ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ
Ордена Трудового Красного Знамени федеральное государственное
бюджетное образовательное учреждение высшего образования
Московский технический университет связи и информатики

Кафедра технологий электронного обмена данными

ЛАБОРАТОРНЫЙ ПРАКТИКУМ

ОСНОВЫ ТЕХНОЛОГИЙ СЕТИ ИНТЕРНЕТ

Москва 2019

УДК 004.77
ББК 32.971.353

Лабораторный практикум

Основы технологий сети Интернет

Направление подготовки: 11.03.02 – Инфокоммуникационные технологии и системы связи

Составители: А.С. Кремер, А.В. Иванюк, И.В. Захаров, К.А. Севрук, К.А. Трушкин, Нейман-заде Мурад Искендер-оглы, В.И. Перекатов, А.В. Смирнов, М.И. Кухаренко, Г.В. Курячий, А.А. Волков, В.С. Малиночкин, М.Р. Магафуров, Д.А. Ермолаев, Т.В. Черствов

Издание утверждено советом факультета Сети и системы связи Московского технического университета связи и информатики. Протокол № 10 от 21 мая 2019 года.

Рецензент: В.А. Ерёменко, к.т.н., доцент

ISBN

Содержание

Введение

Пособие для лабораторного практикума состоит из трех глав и содержит 24 практикума по следующим направлениям:

- понимание реализации стека сетевых протоколов и принципов работы сети передачи данных на базе IP, организации адресации и маршрутизации, сетевых интерфейсов;
- понимание принципов построения и управления сетью Интернет, взаимодействия сетей связи и обмена трафиком;
- понимание принципов функционирования системы DNS, системы координации и администрирования доменных имен верхнего уровня (TLDs);
- понимание преимуществ и недостатков открытой модели разработки программного обеспечения, сути свободных лицензий и их ограничений, процессов разработки и внедрения новых сервисов и протоколов;
- умение работать с сетевыми утилитами, анализаторами трафика и другими специальными инструментами.

Пособие включает сведения о программной платформе лабораторного практикума – отечественной операционной системе ОС Альт (Глава 2) и аппаратной платформе лабораторного практикума – вычислительных комплексах Эльбрус (Глава 3).

Каждый практикум содержит: цель работы, задание на работу, термины и используемые команды, порядок выполнения работы, контрольные вопросы.

По результатам выполнения лабораторной работы создается отчет, который оформляется в формате Open Doc и сохраняется в папке на рабочем столе компьютера.

Отчет должен содержать следующий материал:

- титульный лист;
- цель работы;
- схему сети;
- копии экранов с результатами работы команд ОС Альт, анализатора протоколов и программ моделирования трафика;
- выводы по результатам выполнения лабораторной работы;
- контрольные вопросы и ответы на них.

Лабораторный практикум по дисциплине “Основы технологий сети Интернет” предназначен для студентов, обучающихся по направлению подготовки 11.03.02 – Инфокоммуникационные технологии и системы связи.

1. Лабораторный практикум по дисциплине «Основы технологии сети Интернет»

Практикум № 1
Знакомство с терминалом и основными командами

1. ЦЕЛЬ РАБОТЫ

Ознакомление с терминалом ОС Альт. Изучение основных команд терминала.

2. ЗАДАНИЕ НА РАБОТУ

1. Ознакомиться с терминалом ОС Альт;
2. Изучить основные команды терминала.

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

3.1. Ознакомление с терминалом ОС Альт

Основная работа в курсе лабораторных будет проходить в терминале. Для его вызова необходимо нажать ПКМ на свободном от ярлыков, папок и панели задач месте на рабочем столе (рисунок 1). Далее выбрать «Открыть в Терминале», после чего будет открыт терминал (рисунок 2).

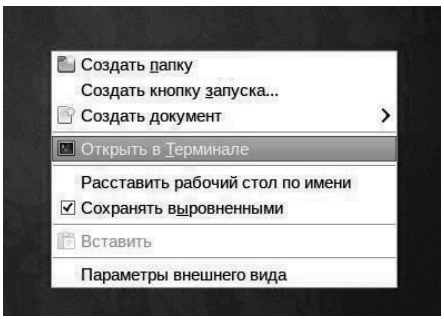


Рисунок 1. Диалоговое окно для открытия терминала

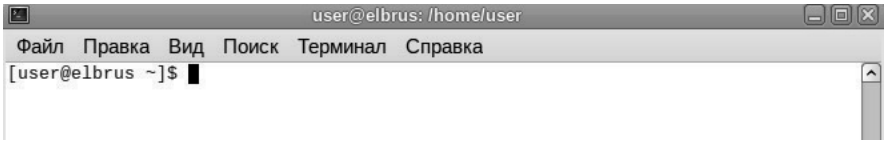


Рисунок 2. Открытый терминал

Альтернативный способ открытия терминала изображён на рисунке 3, для этого на панели задач вверху экрана кликнуть на «Приложения – Системные – Терминал среды MATE».

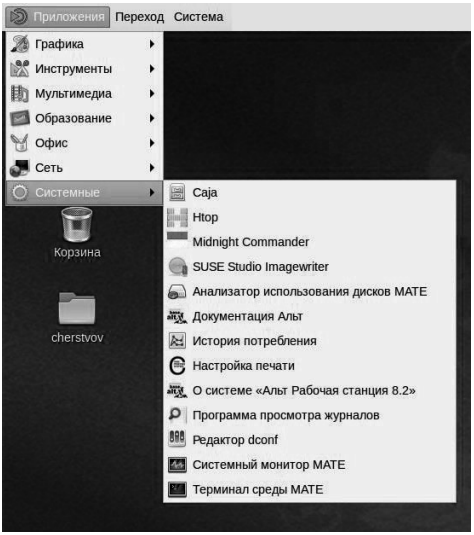


Рисунок 3. Альтернативный способ открытия терминала

В терминале нас ожидает приглашение для ввода команд, которое изображено на рисунке 4.

```
[user@elbrus ~]$
```

Рисунок 4. Приглашение для ввода команды в терминал

Разберём из чего состоит приглашение:

`user` – имя пользователя, от имени которого происходит работа в терминале;

`@` - разделитель;

`elbrus` – название машины;

~ - текущая директория. Символом “~” обозначается домашняя директория для активного пользователя. В данном случае под ”~” понимается /home/user.
$ - приглашение к выполнению команды от имени простого пользователя. Если пользователь имеет права администратора, то символ будет изменён на “#”.

Для работы в терминале от имени администратора (суперпользователя) необходимо зайти под его именем, сменив пользователя. Для этого используется команда

```
su -
```

Далее будет предложено ввести пароль от профиля администратора (на всех компьютерах – 123). Процесс входа от имени суперпользователя изображён на рисунке 5.

```
[user@elbrus ~]$ su -
Password:
[root@elbrus ~]#
```

Рисунок 5. Процесс перехода под суперпользователя

Как видно из рисунка 5, пользователь был изменён с user на root, а символ “$” на “#”. При этом домашняя директория так же была изменена с /home/user на /root. Изменение домашних директорий изображено на рисунке 6.

```
[user@elbrus ~]$ ~
-bash: /home/user: является директорией
[user@elbrus ~]$ su -
Password:
[root@elbrus ~]# ~
-bash: /root: является директорией
```

Рисунок 6. Разница между домашними директориями у обычного и суперпользователей

Для выхода из режима суперпользователя или из-под любого другого пользователя, необходимо нажать CTRL+D.

3.2. Изучение основных команд

Все команды в терминале имеют общую структуру:

```
команда -ключ значение
```

`команда` – название исполняемого файла из каталогов, записанных в переменной $PATH (/bin, /sbin, /usr/bin, /usr/sbin, /usr/local/bin, /usr/local/sbin или полный путь к исполняемому файлу.

`ключ` – пишется после названия команды. У каждой команды свой набор ключей, подробнее о которых можно прочитать в инструкции к команде. С помощью ключей выставляются настройки, с использованием которых будет выполнена команда.

Ключи бывает короткие – состоящие из одного символа, например, -a, или же длинные, описывающие полное название ключа, например, -all. При этом ключ -a и -all, могут быть эквивалентными для какой-то команды, а для другой – нет. Короткие ключи можно перечислять подряд, например, `netstat -tpln`.

`Значение` – величина, которая будет передана при выполнении команды. Например, для команды `ping mtuci.ru` значением будет являться адрес `mtuci.ru`, который необходимо пропинговать.

Разберём основные команды, которые потребуются для работы в терминале:

`man <команда>` – показывает инструкцию по применению команды. Пример вызова команды изображён на рисунке 7, а полученная информация на рисунке 8. Для выхода из инструкции необходимо нажать клавишу `q`.

```
[user@elbrus ~]$ man man
```

Рисунок 7. Пример вызова команды `man`

```
man(1)                    General Commands Manual                   man(1)

NAME
       man - format and display the on-line manual pages

SYNOPSIS
       man  [-acdfFhkKtwW]  [--path]  [-m system] [-p string] [-C config file]
       [-M pathlist] [-P pager] [-B browser] [-H htmlpager] [-S  section list]
       [section] name ...

DESCRIPTION
       man formats and displays the on-line manual pages.  If you specify sec-
       tion, man only looks in that section of the manual.  name  is  normally
       the  name of the manual page, which is typically the name of a command,
       function, or file.  However, if name contains  a  slash  (/)  then  man
       interprets  it  as a file specification, so that you can do man ./foo.5
       or even man /cd/foo/bar.1.gz.

       See below for a description of where man  looks  for  the  manual  page
       files.

MANUAL SECTIONS
       The standard sections of the manual include:
lines 1-27
```

Рисунок 8. Пример полученной инструкции при вводе команды `man man`

`cd <путь к директории>` – переход в указанную директорию. Без указания директории переходит в домашнюю директорию для пользователя, от которого осуществляется ввод команд.

Команда

```
cd .
```

Эквивалентна актуальному пути к каталогу, где находится пользователь. Вместо того, чтобы набирать весь путь к актуальной директории можно поставить символ "`.`" и писать путь после него. Например, в папке `/home/user` находится директория `.ssh`, перейти туда можно двумя способами (рисунок 9). Первый способ – полностью указать путь:

```
cd /home/user/.ssh
```

Второй способ:

```
cd ./.ssh
```

```
[user@elbrus ~]$ cd /home/user/.ssh
[user@elbrus .ssh]$ cd ..
[user@elbrus ~]$ cd ./.ssh
[user@elbrus .ssh]$
```

Рисунок 9. Два альтернативных способа вызова команды `cd`

При вводе команды

```
cd ..
```

Происходит переход в директорию, которая находится выше. Если до ввода команды активной директорией была `/home/user`, то после выполнения станет `/home`. Использование данной команды изображено на рисунке 10.

```
[user@elbrus ~]$ cd ..
[user@elbrus home]$
```

Рисунок 10. Использование команды `cd ..`, при нахождении в директории `/home/user`

`pwd` – выводит текущую директорию. Пример использования команды изображён на рисунке 11.

```
[user@elbrus ~]$ pwd
/home/user
```

Рисунок 11. Пример использования команды `pwd`

`ls` – выводит список директорий и файлов в текущем каталоге. Информация, полученная при вводе команды `ls` для домашней директории пользователя изображена на рисунке 12.

```
[user@elbrus ~]$ ls
Документы  Загрузки  Общедоступные  Рабочий стол
```

Рисунок 12. Результат выполнения команды `ls`

`ip` – утилита, которая используется для работы с маршрутизацией, туннелями, устройствами. С помощью неё легко определить ip адрес устройства, для этого необходимо ввести команду

```
ip a
```

Результат выполнения команды изображён на рисунке 13.

```
[user@elbrus Рабочий стол]$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNO
WN group default qlen 1000
    link/ether 98:a7:b0:00:de:e0 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::9aa7:b0ff:fe00:dee0/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNO
WN group default qlen 1000
    link/ether 98:a7:b0:00:de:e1 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::9aa7:b0ff:fe00:dee1/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
4: eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNO
WN group default qlen 1000
    link/ether 98:a7:b0:00:de:e2 brd ff:ff:ff:ff:ff:ff
    inet 172.16.1.31/24 brd 172.16.1.255 scope global dynamic eth2
       valid_lft 38271sec preferred_lft 38271sec
    inet6 fe80::9aa7:b0ff:fe00:dee2/64 scope link
       valid_lft forever preferred_lft forever
```

Рисунок 13. Пример вызова команды `ip a`

Как видно из рисунка 13, у `eth2` есть ip адрес `172.16.1.31/24`

3.3. Использование WireShark

Для начала необходимо убедиться в том, что Wireshark установлен, для этого получим права суперпользователя, использовав команду

```
su -
```

Далее, введём команду

```
which wireshark
```

Если пакет не удалось обнаружить (рисунок 14), то установим его с помощью команды

```
apt-get install wireshark
```

```
[user@elbrus ~]$ which wireshark
which: no wireshark in (/home/user/bin:/bin:/usr/bin:/usr/local/bin:/usr/X11R6/b
in:/usr/games)
```

Рисунок 14. Отсутствие Wireshark в системе

Для выбора сетевого интерфейса для анализа в WireShark, изначально, необходимо иметь права суперпользователя. Отключим это, введя команду

```
chmod u+s /usr/bin/dumpcap
```

Пример выполнения команды представлен на рисунке 15.

```
[root@elbrus ~]# chmod u+s /usr/bin/dumpcap
```

Рисунок 15. Пример выполнения команды

Выйдем из режима суперпользователя, нажав `CTRL+D`.

Включение WireShark производится в консоли с помощью команды `wireshark`. При запуске потребуется выбрать сетевой интерфейс, с которого будет сниматься информация для последующего анализа. Для простоты можно выбрать `any`, что означает снятие со всех интерфейсов (рисунок 16).

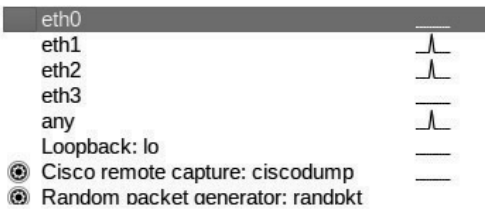


Рисунок 16. Список доступных интерфейсов

Для начала захвата пакетов воспользуемся кнопкой в виде синего плавника (рисунок 17) на панели инструментов. Для завершения захвата существует кнопка, смежная с плавником, в виде красного квадрата (рисунок 18).

Рисунок 17. Кнопка захвата пакетов

Рисунок 18. Кнопка остановки захвата пакетов

В основном поле программы (рисунок 19) находятся перехваченные пакеты, которые можно использовать для дальнейшего анализа.

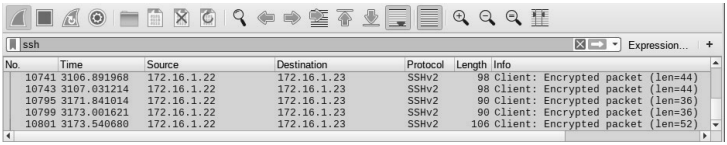


Рисунок 19. Фрагмент основного окна программы

Опишем основные элементы, отображённые на рисунке 19:

- `No.` – номер пакета;
- `Time` – время перехвата с момента включения режима перехвата пакетов;
- `Source` – источник пакета;
- `Destination` – назначение пакета;
- `Protocol` – протокол, по которому передаётся пакет;
- `Length` – длина пакета;
- `Info` – некоторая информация о пакете.

Отдельно следует упомянуть поле фильтрации, которое позволяет выводить пакеты, удовлетворяющие некоторым условиям. На рисунке указано, что следует отображать все пакеты, которые передаются по протоколу `ssh`.

При двойном клике на пакете откроется меню с его содержимым. Пример изображён на рисунке 20.

Рисунок 20. Пример содержания пакета

3.4. Использование утилиты iperf

Для генерирования трафика между двумя и более компьютерами можно использовать утилиту `iperf`. С её помощью можно легко проверить правильность настройки маршрутов между рабочими станциями, а также проводить анализ трафика. Ниже описано, как пользоваться утилитой.

Один из компьютеров должен выступать в качестве сервера. На нём необходимо выполнить команду (`-s` – сервер, `-p` – порт)

```
iperf -s -p port
```

Клиенту необходимо ввести

```
iperf -c ip -p port
```

где

`ip` – ip-адрес сервера,

`port` – порт сервера.

Если ip-адрес сервера – 10.10.0.1, а port, для примера, 12345, то получится следующее.

Для сервера

```
iperf -s -p 12345
```

Для клиента

```
iperf -c 10.10.1.1 -p 12345
```

Пример выполнения изображён на рисунках 21-22.

```
[user@elbrus Рабочий стол]$ iperf -s -p 12345
------------------------------------------------
Server listening on TCP port 12345
TCP window size: 85.3 KByte (default)
------------------------------------------------
```

Рисунок 21. Активация iperf у сервера

```
[user@elbrus Рабочий стол]$ iperf -c 10.10.1.1 -p 12345
------------------------------------------------------------
Client connecting to 10.10.1.1, TCP port 12345
TCP window size: 2.50 MByte (default)
------------------------------------------------------------
[  3] local 10.10.1.1 port 39624 connected with 10.10.1.1 port 12345
```

Рисунок 22. Активация iperf у клиента

Практикум № 2
Преобразование имен NSS

1. ЦЕЛЬ РАБОТЫ

Ознакомится с файлами, расположенными в каталоге `etc`: `nsswitch.conf`, `services`, `passwd`. Научиться создавать пользователей с паролями, определять их идентификатор, заходить в сеанс под этим пользователем и выполнять различные команды. Научиться создавать и удалять группы пользователей, а также определять идентификатор групп.

2. ЗАДАНИЕ НА РАБОТУ

1. Познакомьтесь с файлом `/etc/nsswitch.conf` и запишите основные базы данных для этого файла и источники информации для них.

2. Познакомьтесь с файлом `/etc/services` и заполните таблицу 1. соответствия служб и номеров портов (можно воспользоваться фильтром `grep`).

Таблица 1. Соответствие служб и номеров портов

Служба	Номер порта
	1
ftp	
ssh	
	23
	25
imap	
	631

3. Познакомьтесь с файлом `/etc/passwd` и заполните таблицу 2.

Таблица 2. Соответствие идентификаторов UID и имён пользователей

UID	Имя пользователя
0	
1	
8	
485	
500	

4. Создайте нового пользователя с именем `test`. Убедитесь в его присутствии в `/etc/passwd`. Задайте пароль для пользователя. Определите его UID.

Завершите сеанс суперпользователя и начните сеанс нового пользователя. Перейдите в его домашний каталог и создайте там файл test_file. Завершите сеанс пользователя.

5. Создайте новую группу `newgr`. Проверьте результат в `/etc/group`. Определите GID.

Таблицы необходимо создать с помощью текстового редактора LibreOffice Writer и сохранить на рабочем столе. Название файла должно содержать номер группы и фамилию студента.

3. ТЕРМИНЫ И ИСПОЛЬЗУЕМЫЕ КОМАНДЫ

Термины

NSS (Name Service Switch – преобразование разрешенных имен) – установка, которая обеспечивает работу различных источников для общей конфигурации баз данных и механизмов разрешения имен. NSS позволяет системному администратору выбирать различные сервисы имен (поиск в текстовом файле, DNS, NIS, NIS+) для поиска в базе данных доменных имен, имен пользователей, имен групп пользователей и т.д.

Используемые команды

`cat` – команда для работы с файлами.

`grep` – команда для фильтрации информации из файла. Применяется как обёртка для других команд, например, для команды `cat`.

`useradd` – команда для создания нового пользователя.

`passwd` – команда для управления паролями пользователей.

`groupadd` – команда для создания новой группы.

`nano` – текстовый редактор.

`getent база элемент` – получает указанные элементы из указанной базы данных.

4. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

4.1. Файл nsswitch.conf

Откроем файл, находящийся в каталоге /etc/nsswitch.conf, с помощью команды cat и посмотрим основные базы данных и источники информации для них (рисунок 1).

```
[user@elbrus ~]$ cat /etc/nsswitch.conf
```

Рисунок 1. Пример выполнения команды cat

Например, файл passwd является базой данный для файла nsswitch.conf. Источником информации для passwd являются локальные файлы. В таком же формате приведена информация для других баз данных (рисунок 2).

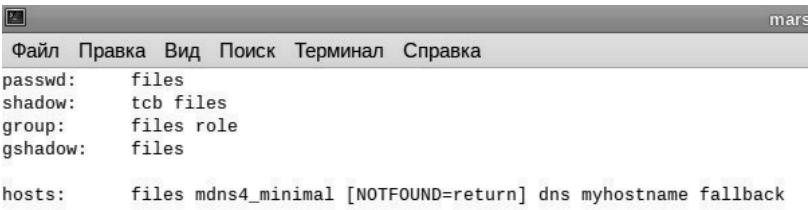


Рисунок 2. Файл nsswitch.conf

4.2. Файл services

Откроем файл /etc/services и ознакомимся с ним (рисунок 3).

```
[user@elbrus ~]$ cat /etc/services
```

Рисунок 3. Пример открытия файла services

Узнаем из этого файла названия служб и портов, которые эти службы используют. Для этого воспользуемся фильтром grep. Пример выполнения команды изображён на рисунке 4.

```
cat /etc/services | grep 1
```

Определим службу, которая имеет порт с номером 1. В соответствии с командой выше, изменяя значение параметра после команды `grep`, необходимо получить службы и номера портов служб для заполнения в таблицу 1.

```
[user@elbrus ~]$ cat /etc/services | grep 1
# The Well Known Ports are those from 0 through 1023.
# The Registered Ports are those from 1024 through 49151.
# The Dynamic and/or Private Ports are those from 49152 through 65535.
tcpmux          1/tcp                           # TCP port service multiplexer
tcpmux          1/udp                           # TCP port service multiplexer
systat          11/tcp          users           # Active Users
systat          11/udp          users           # Active Users
```

Рисунок 4. Использование `grep` для фильтрации информации из файла

Для поиска данных из файлов, которые описаны в nsswitch.conf, также существует специальная команда `getent`. С помощью данной команды, выполним поиск номера порта для службы ftp и названия службы, которая использует порт 1

```
getent services ftp
getent services 1
```

Результат выполнения представлен на рисунке 5.

```
[root@elbrus ~]# getent services ftp
ftp                     21/tcp fsp fspd
[root@elbrus ~]# getent services 1
tcpmux                  1/tcp
```

Рисунок 5. Пример использования команды `getent`

4.3. Файл `passwd`

Откроем файл `/etc/passwd`, используя команду

```
cat /etc/passwd
```

Используя фильтр `grep` (как в пункте 4.2), необходимо заполнить таблицу 2. Пример использования команды с фильтром изображен на рисунке 6.

```
[user@elbrus ~]$ cat /etc/passwd | grep 0
root:x:0:0:System Administrator:/root:/bin/bash
uucp:x:10:14:uucp:/var/spool/uucp:/dev/null
games:x:12:100:games:/usr/games:/dev/null
ftp:x:14:50:FTP User:/var/ftp:/dev/null
geoclue:x:490:480:User for GeoClue service:/var/lib/geoclue:/dev/null
mrmalina:x:500:500:malina:/home/mrmalina:/bin/bash
user1:x:501:501::/home/user1:/bin/bash
elbrus:x:502:502::/home/elbrus:/bin/bash
user:x:503:503::/home/user:/bin/bash
```

Рисунок 6. Фрагмент файла `passwd`

Разберём одну строку для примера

`user:x:503:503::/home/user:/bin/bash`

Все поля разделены с помощью двоеточий и имеют следующий порядок:

- Имя пользователя (`user`);
- Необязательный зашифрованный пароль (`x`);
- Идентификатор пользователя UID (`503`);
- Идентификатор группы GID (`503`);
- Комментарий (`отсутствует`);
- Домашний каталог (`/home/user`);
- Используемая оболочка (`/bin/bash`).

Если в используемой оболочке указано `/bin/null` – это значит, что данный пользователь не использует никакую оболочку.

4.4. Создание нового пользователя

Для добавления нового пользователя, вам потребуется права суперпользователя. Для этого вводим команду

`su -`

Создать нового пользователя с именем `test`, можно набрав команду

`useradd test`

Переход под суперпользователя и создание нового пользователя изображены на рисунке 7.

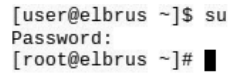


Рисунок 7. Создание нового пользователя

Убедимся, что он присутствует в `/etc/passwd`, набрав команду

```
cat /etc/passwd
```

Новый пользователь находится в конце файла (рисунок 8).

Рисунок 8. UID нового пользователя

Проверить можно и не открывая файл, для этого используем команду

```
getent passwd test
```

Команда `getent` будет искать пользователя `test` в базе данных `passwd`. Если оно обнаружено, то будет выведен идентификатор (UID) этого пользователя.

Для задания или смены пароля существует команда

```
passwd пользователь
```

В нашей случае, имя пользователя – `test`. Команда будет иметь вид

```
passwd test
```

После ввода команды потребуется дважды ввести пароль (рисунок 9).

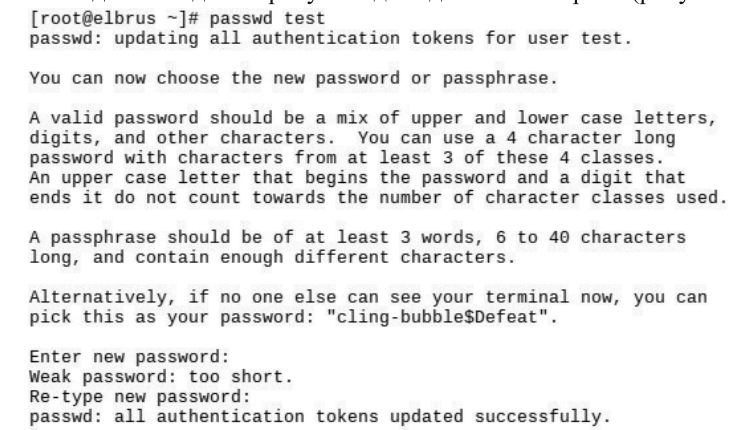


Рисунок 9. Создание пароля для нового пользователя

Обратим внимание, что при использовании команды `passwd` терминал выводит информацию о пожеланиях к паролю – пароль должен состоять не менее чем из трёх слов и содержать от 6 до 40 различных символов.

Выйдем из-под суперпользователя, нажав комбинацию клавиш `CTRL+D`, а затем войдём под именем пользователя test, введя команду (рисунок 10)

```
su - test
```

```
[root@elbrus ~]# su - test
```

Рисунок 10. Переход под нового пользователя

Зайдя под пользователем test, текущая директория была изменена на домашнюю директорию пользователя. Для проверки введём команду

```
pwd
```

Теперь создадим файл `test_file` используя текстовый редактор `nano`

```
nano test_file
```

Запишем туда какой-либо текст и, затем, сохраним его, нажав на `CTRL+X`, после `y` и нажатие `enter`. Чтобы удостовериться в том, что файл был создан, введён команду `ls` (рисунок 11).

```
[test@elbrus ~]$ pwd
/home/test
[test@elbrus ~]$ nano test_file
[test@elbrus ~]$ ls
test_file
```

Рисунок 11. Создание нового файла

4.5. Создание новой группы

Перейдём под суперпользователя, для чего, в начале, выйдем с пользователя test, нажав `CTRL+D`, а затем введём

```
su -
```

Для создания новых групп существует команда `groupadd`. С её помощью создадим группу `newgr`

```
groupadd newgr
```

Получим идентификатор группы, введя команду

```
cat /etc/group
```

Новая группа находится в конце файла (рисунок 12).

Рисунок 12. Создание новой группы и вывод информации обо всех группах

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Назначение команд ОС Альт, используемых в работе.
2. Назначение NSS.
3. Виды баз данных в `nsswitch.conf`.
4. Формат записи в `/etc/passwd`.
5. Какие рекомендации к паролю предлагает утилита passwd?

Практикум № 3
Знакомство с сетевыми интерфейсами

1. ЦЕЛЬ РАБОТЫ

Познакомиться с сетевыми интерфейсами и некоторыми командами для совершения операций над ними. Изучить возможности утилит netstat и ss.

2. ЗАДАНИЕ НА РАБОТУ

1. Узнать имена сетевых интерфейсов в системе и посчитать их количество.
2. Вывести локальный сетевой интерфейс, используя утилиту `grep`.
3. Включить и выключить сетевой интерфейс `eth0`. Убедитесь, что ваши действия отображаются в состоянии интерфейса (UP или DOWN).
4. Запустить `ping` локального адреса. Перевести этот процесс в фоновый режим. Убедиться, с помощью команды `pgrep` (или `ps`) в том, что действие выполняется. Завершить процесс командой `kill`.
5. С помощью команды `netstat`, узнать список соединений типа TCP и приложений, связанных с ними.
6. Выполнить предыдущее задание с помощью команды `ss`.

3. ТЕРМИНЫ И ИСПОЛЬЗУЕМЫЕ КОМАНДЫ

Термины

Сетевой интерфейс – точка соединения между компьютером пользователя и частной или общественной сетью. Существуют внешние и внутренние (локальные) интерфейсы.

Локальный сетевой интерфейс – это локальная петля, (петля говоря простым языком – это когда посланный сигнал отправляется и возвращается обратно, то есть происходит движение по кругу) которая имеет IP-адрес 127.0.0.1 и предназначена для сетевого доступа к своему же компьютеру.

Фоновый режим - это режим, при котором определенные сервисы и программы работают автоматически, без участия человека.

Используемые команды

`ip` – команда для работы с сетевыми интерфейсами.

`ip link` – команда для управления состоянием сетевых интерфейсов и получения информации о них.
`ping` – команда для отправки пакетов по указанному сетевому адресу.
`grep` – команда для фильтрации информации из файла. Часто применяется как обёртка для других команд.
`pgrep` – команда, позволяющая просматривать активные процессы в системе и выдающая идентификаторы процессов, атрибуты которых соответствуют указанным в командной строке запросам.
`kill` – закрытие (“убийство”) процесса.
`ps` – тоже, что и `pgrep`.
`netstat` – выводит на дисплей состояние TCP-соединений (как входящих, так и исходящих), таблицы маршрутизации, число сетевых интерфейсов и сетевую статистику по протоколам. В настоящее время команда устаревшая и не рекомендована к применению.
`ss` – современный аналог команды `netstat`.

4. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

4.1. Определение имён сетевых интерфейсов

Чтобы узнать имена сетевых интерфейсов в системе, воспользуемся утилитой `ip`, используя параметр `link` (рисунок 1).

```
[user@elbrus Рабочий стол]$ ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN mode DEFAULT group default qlen 1000
    link/ether 98:a7:b0:00:dd:60 brd ff:ff:ff:ff:ff:ff
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN mode DEFAULT group default qlen 1000
    link/ether 98:a7:b0:00:dd:61 brd ff:ff:ff:ff:ff:ff
4: eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN mode DEFAULT group default qlen 1000
    link/ether 98:a7:b0:00:dd:62 brd ff:ff:ff:ff:ff:ff
```

Рисунок 1. Пример использования команды `ip link`

Имена сетевых интерфейсов находятся после цифр 1, 2, … и т.д. Чтобы посчитать количество интерфейсов, воспользуемся командой (рисунок 2)

```
ip -o link | wc -l
```

```
[user@elbrus Рабочий стол]$ ip -o link | wc -l
4
```

Рисунок 2. Пример выполнения команды для подсчёта количества сетевых интерфейсов

Параметр `-o` (буква) в команде `ip link` выводит каждую запись на отдельной строке. Для подсчёта количества строк воспользуемся командой для подсчёта строк `wc -l` (буква).

4.2. Получение информации о локальном интерфейсе

Для вывода информации о локальном интерфейсе воспользуемся утилитой `grep`, которой обернём команду `ip link` (рисунок 3)

```
ip link | grep lo
```

```
[user@elbrus ~]$ ip link | grep lo
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT
 group default qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
```

Рисунок 3. Полученная информация о локальном интерфейсе

4.3. Работа с сетевым интерфейсом eth0

Для работы с сетевым интерфейсом необходимо войти в режим суперпользователя (пароль – 123)

```
su -
```

Для выключения сетевого интерфейса используем команду (рисунок 4)

```
ip link set eth0 down
```

Команда `ip link set` изменяет параметры указанного сетевого интерфейса.

```
[root@elbrus ~]# ip link set eth0 down
```

Рисунок 4. Выключение сетевого интерфейса eth0

Проверим, что интерфейс был выключен, введя команду `ip a` (рисунок 5). Как видно из рисунка, сетевой интерфейс eth0 перешёл в состояние DOWN. Значит он выключен.

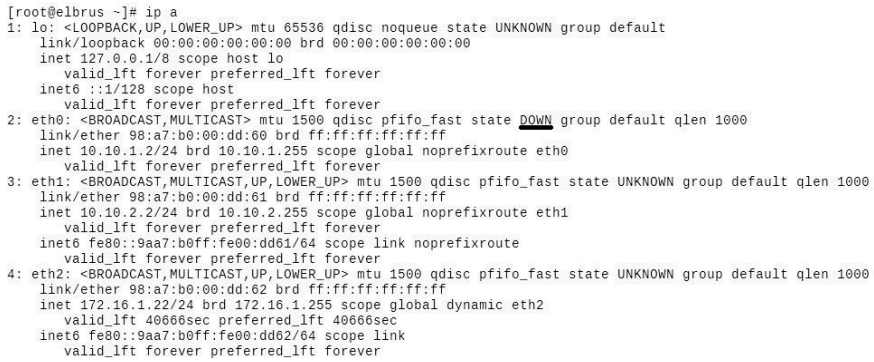


Рисунок 5. Пример выполнения команды `ip a`

Теперь включим его обратно, воспользовавшись командой

```
ip link set eth0 up
```

Результат выполнения команды представлен на рисунке 6. Если на интерфейсе есть ip адрес, то будет написано UP, иначе UNKNOWN.

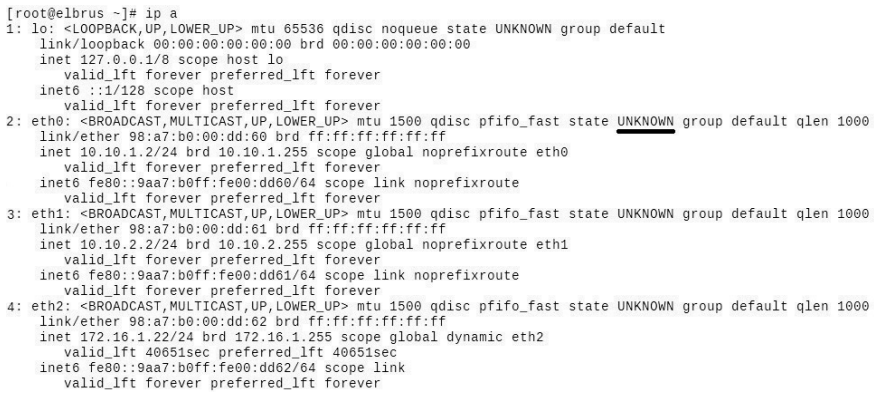


Рисунок 6. Включение сетевого интерфейса

4.4. Использование утилиты ping

Выйдем из-под суперпользователя, нажав `CTRL+D`. Воспользуемся командой `ping` и укажем ip адрес сетевого интерфейса, а затем переведём её в фоновый режим, нажав на `CTRL+Z` (рисунок 7)

```
ping 127.0.0.1
```
CTRL+Z

```
[user@elbrus ~]$ ping 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_req=1 ttl=64 time=0.021 ms
64 bytes from 127.0.0.1: icmp_req=2 ttl=64 time=0.024 ms
64 bytes from 127.0.0.1: icmp_req=3 ttl=64 time=0.022 ms
64 bytes from 127.0.0.1: icmp_req=4 ttl=64 time=0.023 ms
^Z
[1]+  Stopped                 ping 127.0.0.1
```

Рисунок 7. Перевод процесса `ping` в фоновый режим

Используем команду `ps`, чтобы обнаружить процесс ping в фоновом режиме, а также для определения его идентификатора PID (рисунок 8)

```
[user@elbrus Рабочий стол]$ ps
  PID TTY          TIME CMD
 3902 pts/1    00:00:00 bash
 5738 pts/1    00:00:00 ping
 5739 pts/1    00:00:00 ps
```

Рисунок 8. Список процессов

Завершим процесс `ping` командой `kill -9` (-9 – номер сигнала для убийства процесса, который означает безусловное завершение без возникновения ошибок), введя команду

```
kill -9 5738
```

5738 – номер процесса команда `ping`, который был получен в результате использования команды `ps` (поле PID).

Результат выполнения команды `kill` изображён на рисунке 9.

```
[user@elbrus Рабочий стол]$ kill -9 5738
[user@elbrus Рабочий стол]$ [1]+  Killed                  ping 127.0.0.1
```

Рисунок 9. Завершение процесса `ping`

4.5. Получение списка TCP соединений

Определим список всех слушающих сокетов типа TCP с помощью утилиты `netstat`

```
netstat -lt
```

Распишем ключи, использованные в команде:

`-l` – показывать только слушающие сокеты;

`-t` – отображать только TCP соединений.

Результат выполнения команды изображён на рисунке 10.

```
[user@elbrus Рабочий стол]$ netstat -lt
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address               Foreign Address             State
tcp        0      0 *:ssh                       *:*                         LISTEN
tcp        0      0 *:microsoft-ds              *:*                         LISTEN
tcp        0      0 *:netbios-ssn               *:*                         LISTEN
tcp        0      0 *:hostmon                   *:*                         LISTEN
tcp        0      0 *:ssh                       *:*                         LISTEN
tcp        0      0 *:ipp                       *:*                         LISTEN
tcp        0      0 *:microsoft-ds              *:*                         LISTEN
tcp        0      0 *:netbios-ssn               *:*                         LISTEN
tcp        0      0 *:hostmon                   *:*                         LISTEN
```

Рисунок 10. Список TCP соединений, полученных с помощью утилиты `netstat`

4.6. Использование утилиты ss

Пункт 4.5. можно выполнить с помощью современной утилиты ss. Её синтаксис и флаги аналогичны команде `netstat`

```
ss -lt
```

Результат выполнения представлен на рисунке 11.

```
[user@elbrus Рабочий стол]$ ss -lt
State    Recv-Q    Send-Q        Local Address:Port                Peer Address:Port
LISTEN   0         128                 0.0.0.0:ssh                      0.0.0.0:*
LISTEN   0         50                  0.0.0.0:microsoft-ds             0.0.0.0:*
LISTEN   0         50                  0.0.0.0:netbios-ssn              0.0.0.0:*
LISTEN   0         128                 0.0.0.0:hostmon                  0.0.0.0:*
LISTEN   0         128                       *:ssh                            *:*
LISTEN   0         128                       *:ipp                            *:*
LISTEN   0         50                        *:microsoft-ds                   *:*
LISTEN   0         50                        *:netbios-ssn                    *:*
LISTEN   0         128                       *:hostmon                        *:*
```

Рисунок 11. Список TCP соединений, полученных с помощью утилиты `ss`

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Назначение команд ОС Альт, используемых в работе.
2. Сетевой интерфейс. Виды сетевых интерфейсов.
3. Состояния сетевого интерфейса.
4. Фоновой режим работы. Перевод утилиты в фоновый режим.
5. Утилита ip. Примеры применения.

Практикум № 4
Настройка маршрутизации

1. ЦЕЛЬ РАБОТЫ

Познакомиться с понятиями публичных и частные сетей. Изучить механизмы функционирования маршрутизации в ОС Альт. Научится настраивать статическую маршрутизацию.

2. ЗАДАНИЕ НА РАБОТУ

Компьютеры соединены по схеме, изображённой на рисунке 1.

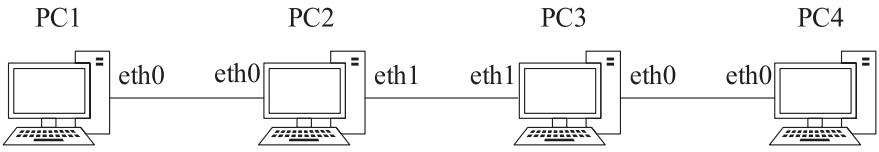


Рисунок 1. Схема соединения компьютеров

1. Настройка сетевых интерфейсов
   - ☐ Присвоение ip адресов;
   - ☐ Настройка конфигурационных файлов;
   - ☐ Разрешение передачи пакетов между сетевыми интерфейсами;
   - ☐ Настройка сетевых маршрутов.
2. Проверка правильности выполнения работы

3. ТЕРМИНЫ И ИСПОЛЬЗУЕМЫЕ КОМАНДЫ

Термины

Для большинства компьютеров локальным адресом считается адрес, расположенный в диапазоне 127.0.0.0, …,127.255.255.25, именно поэтому ip адреса, расположенные в данном диапазоне, нельзя использовать для узла локальной сети.

Использование локального адреса позволяет устанавливать соединение и передавать информацию для программ-серверов, работающим на том же

компьютере, что и программа-клиент, независимо от конфигурации аппаратных сетевых средств компьютера.

Публичные IP-адреса, это те адреса, которые уникальны в пределах всего мира, иногда их еще называют белыми адресами. Можно было бы сказать, что публичные адреса – это все те адреса, которые не обозначены, как частные, но это далеко не так, поскольку в протоколе IP есть еще целый ряд сетей, у которых особое назначение, и большая часть из них не находится в Интернете.

Частные IP-адреса – это такие адреса, которые не уникальны в пределах всего мира, но они должны быть уникальны в пределах локальной сети.

Используемые команды

`nano` – текстовый редактор.

`ip a` – команда для настройки сетевых интерфейсов.

`ip r` – команда для настройки маршрутизации.

`iperf` – утилита для генерации трафика.

4. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

4.1. Настройка сетевых интерфейсов

Для выполнения всех команд необходимо зайти в режим суперпользователя (пароль – 123)

```
su -
```

Далее, в зависимости от того, какой сетевой интерфейс указан в задании и за каким компьютером находится студент, создадим файл `ipv4address` в директории `/etc/net/ifaces/ethX`, X – номер сетевого интерфейса, согласно рисунку 1 и номеру компьютера студента. Для примера возьмём интерфейс `eth0`, тогда команда будет иметь вид

```
nano /etc/net/ifaces/eth0/ipv4address
```

```
[user@elbrus Рабочий стол]$ su -
Password:
[root@elbrus ~]# nano /etc/net/ifaces/eth0/ipv4adress
```

Рисунок 2. Создание файла ipv4address

В файле необходимо записать ip адрес. Определим, какой адрес необходимо указать для каждой из машин. На всех рабочих местах указан номер. Согласно этому номеру выбирается подсеть по следующему правилу –

подсеть для двух компьютеров выбирается в соответствии с минимальным номером рабочего места. Общая формула выбора

```
10.10.X.Y/24
```

где

`X` – минимальный номер между двумя рабочими местами;

`Y` – номер рабочего места.

Пусть схема подключения выглядит следующим образом

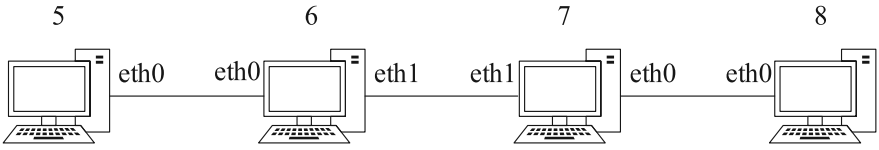


Рисунок 3. Пример схемы с указанием нумерации компьютеров

Согласно рисунку 3 и описанному выше правилу, компьютеры получат следующие ip адреса:

5:

- `10.10.5.5/24` на интерфейсе `eth0`

6:

- `10.10.5.6/24` на интерфейсе `eth0`
- `10.10.6.6/24` на интерфейсе `eth1`

7:

- `10.10.6.7/24` на интерфейсе `eth1`
- `10.10.7.7/24` на интерфейсе `eth0`

8:

- `10.10.7.8/24` на интерфейсе `eth0`

Далее, после записи адреса в файл, сохраним введённые данные с помощью комбинации `CTRL+O`, после чего нажать `enter`, и выходим из редактирования – `CTRL+X`.

Произведём настройку сетевых интерфейсов, для этого откроем файл `options` в директории `/etc/net/ifaces/ethX`, `X` – номер сетевого интерфейса, согласно рисунку 1 и номеру компьютера студента. К примеру, для интерфейса `eth0` команда будет выглядеть следующим образом

```
nano /etc/net/ifaces/eth0/options
```

В указанном файле обязательно должны присутствовать следующие строки с приведёнными ниже значениями:

```
NM_CONTROLLED=no
DISABLED=no
```

Также в файле могут быть указаны и другие параметры, указанные ниже, если они отсутствуют, то прописывать их не требуется:

```
BOOTPROTO=static
ONBOOT=yes
TYPE=eth
CONFIG_WIRELESS=no
CONFIG_IPV4=yes
```

На компьютерах PC2 (в примере – компьютер 6) и PC3 (в примере – компьютер 7) включим пересылку пакетов между сетевыми интерфейсами, изменив в файле `sysctl.conf` параметр `net.ipv4.ip_forward` на 1. Для этого откроем файл, введя команду

```
nano /etc/net/sysctl.conf
```

И изменим значение параметра `net.ipv4.ip_forward` на 1, если значение уже не равно единице (рисунок 4).

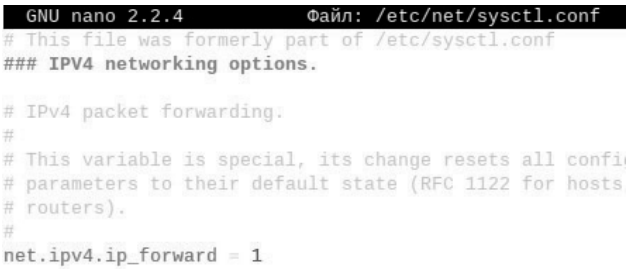


Рисунок 4. Фрагмент файла `sysctl.conf`

Перезапустим сетевую службу на всех компьютерах для применения параметров

```
service network restart
```

Пропишем следующие маршруты в консоли для каждого из компьютеров в соответствии с примером из рисунка 3 (Для других номеров маршруты будут отличаться от приведённых ниже)

5:

```
- ip r add 10.10.6.0/24 via 10.10.5.6 dev eth0
- ip r add 10.10.7.0/24 via 10.10.5.6 dev eth0
```

6:

```
- ip r add 10.10.7.0/24 via 10.10.6.7 dev eth1
```

7:

```
- ip r add 10.10.5.0/24 via 10.10.6.6 dev eth1
```

8:

```
- ip r add 10.10.6.0/24 via 10.10.7.7 dev eth0
- ip r add 10.10.5.0/24 via 10.10.7.7 dev eth0
```

4.2. Проверка правильности выполнения работы

Для проверки правильности выполнения работы воспользуемся утилитой `iperf`, указав сервером машину с номером 5, а клиентом – машину с номером 8. Подробнее об использовании утилиты написано в практикуме № 1.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Назначение команд ОС Альт, используемых в работе.
2. Формат заголовка протокола IP версии 4.
3. Публичные и частные IP адреса.
4. Классы IP адресов.
5. Назначение и содержание таблицы маршрутизации.

Практикум № 5
Динамическая настройка сети

1. ЦЕЛЬ РАБОТЫ

Целью лабораторной работы является знакомство с механизмом динамической маршрутизации. Приобретение практических навыков автоматизации получения компьютерами сетевых настроек.

2. ЗАДАНИЕ НА РАБОТУ

Компьютеры в классе соединяются по следующей схеме:

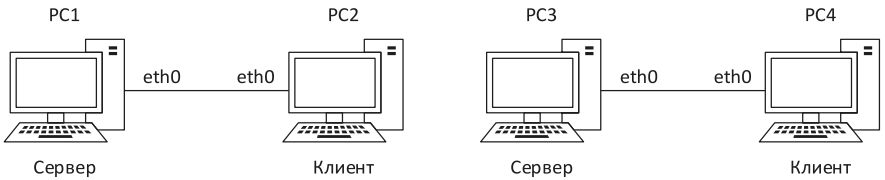


Рисунок 1. Схема подключения компьютеров

1. Настройка DHCP-сервера
   - ☐ Присвоить ip адрес серверу;
   - ☐ Настроить dnsmasq;
   - ☐ Активировать DHCP-сервер.
2. Получение ip адреса
   - ☐ Настроить сетевой интерфейс для получения адреса по DHCP;
   - ☐ Отключить сетевой менеджер;
   - ☐ Запустить DHCP-клиент.

3. ТЕРМИНЫ И ИСПОЛЬЗУЕМЫЕ КОМАНДЫ

Термины

DHCP (англ. Dynamic Host Configuration Protocol — протокол динамической настройки узла) — сетевой протокол, позволяющий сетевым устройствам автоматически получать IP-адрес и другие параметры, необходимые для работы в сети TCP/IP. Данный протокол работает по модели

«клиент-сервер». Для автоматической конфигурации компьютер-клиент на этапе конфигурации сетевого устройства обращается к так называемому серверу DHCP, и получает от него нужные параметры. Сетевой администратор может задать диапазон адресов, распределяемых сервером среди компьютеров. Это позволяет избежать ручной настройки компьютеров сети и уменьшает количество ошибок. Протокол DHCP используется в большинстве сетей TCP/IP.

Используемые команды

`nano` – текстовый редактор.

`service служба restart` – перезапускает указанную службу.

`service служба stop` – отключает указанную службу.

`ip a` – выводит все сетевые интерфейсы и информацию о них.

`mkdir директория` – создание указанной директории.

`killall имя` – закрытие всех процессов с указанным именем.

`dhcpcd` – DHCP-клиент.

`journalctl` – журнал всех системных событий.

`grep` – команда для фильтрации информации из файла. Применяется как обёртка для других команд.

4. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

4.0. Предварительная настройка

Все команды в лабораторной работе выполняются с правами суперпользователя. Подробнее о том, как получить права суперпользователя написано в практикуме № 1.

Для примера выберем, что компьютеры имеют порядковые номера, как изображено на рисунке 2.

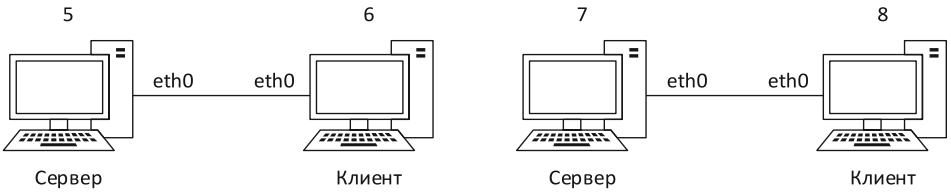


Рисунок 2. Пример схемы соединения с указанием номером рабочих мест

4.1. Настройка DHCP

Произведём настройку сервера (в примере – рабочие места с номерами 5 и 7). Пропишем ip адрес `10.10.N.1` (N – номер рабочего места) на интерфейс `eth0`, предварительно убедившись, что у этого интерфейса нет адреса, введя команду

```
ip a
```

Если адрес присутствует (рисунок 3), то удалим его, введя команду

```
ip a del addr dev eth0
```

где `addr` – ip адрес у сетевого интерфейса (подчёркнут на рисунке 3).

```
[root@elbrus ~]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST> mtu 1500 qdisc pfifo_fast state DOWN group default qlen 1000
    link/ether 98:a7:b0:00:dd:60 brd ff:ff:ff:ff:ff:ff
    inet 10.10.1.2/24 brd 10.10.1.255 scope global noprefixroute eth0
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN group default qlen 1000
    link/ether 98:a7:b0:00:dd:61 brd ff:ff:ff:ff:ff:ff
    inet 10.10.2.2/24 brd 10.10.2.255 scope global noprefixroute eth1
       valid_lft forever preferred_lft forever
    inet6 fe80::9aa7:b0ff:fe00:dd61/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
4: eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN group default qlen 1000
    link/ether 98:a7:b0:00:dd:62 brd ff:ff:ff:ff:ff:ff
    inet 172.16.1.22/24 brd 172.16.1.255 scope global dynamic eth2
       valid_lft 40666sec preferred_lft 40666sec
    inet6 fe80::9aa7:b0ff:fe00:dd62/64 scope link
       valid_lft forever preferred_lft forever
```

Рисунок 3. Информация о сетевых интерфейсах

Для примера будем считать, что N равно 5. В таком случае пропишем адрес для сервера

```
ip a add 10.10.5.1/24 dev eth0
```

Произведём настройку сервера для работы на интерфейсе `eth0`. Для этого зайдём в файл `/etc/dnsmasq.conf`

```
nano /etc/dnsmasq.conf
```

Перейдём в конец файла (для быстрого перехода можно зажать `CTRL+V`) и запишем туда следующие строки

```
dhcp-range=10.10.5.2,10.10.5.254,12h
interface=eth0
```

где

`10.10.5.2` – адрес, обозначающий начало диапазона, с которого будут ‘выдаваться’ адреса клиентам этого DHCP-сервера;

`10.10.5.254` – адрес, обозначающий конец диапазона;

`12h` – время, на которое выданный ip адрес действителен и не потребуется его подтверждение. Время можно указывать в секундах (10s), в минутах (32m) или часах (12h) или ‘infinite’ – бесконечность. Минимальное время – 2m;

`interface=eth0` – сетевой интерфейс, на котором будет работать DHCP-сервер.

Для записи файлов журналирования работы DHCP-сервера необходимо создать директорию `/var/lib/dhcp`. Для этого используем команду

```
mkdir /var/lib/dhcp
```

После выполнения всех команд, перезапустим утилиту dnsmasq, отвечающую за DHCP, введя команду

```
service dnsmasq restart
```

4.2. Получение ip адреса

На машинах-клиентах зайдём в файл настройки сетевого интерфейса `eth0`

```
nano /etc/net/ifaces/eth0/options
```

В нём пропишем следующие строки

```
BOOTPROTO=dhcp
NM_CONTROLLED=no
DISABLED=no
ONBOOT=yes
```

Если каких-либо строк нет, их необходимо дописать. Если в файле присутствует строка `TYPE=eth`, то удалить её.

Отключим сетевой менеджер с помощью команды

```
service NetworkManager stop
```

Данная служба отвечает за автоматическое подключение к сети. Если её не выключить, то ip адрес будет получен автоматически.

Если на сетевом интерфейсе `eth0` присутствует ip адрес, то его необходимо удалить. Пример удаления описан в пункте 4.1.

Отключим все процессы, связанные с DHCP-клиентом

```
killall dhcpcd
```

Для полного перезапуска сети, уничтожим все DHCP-клиенты, которые работают на машине.

Включим новый процесс DHCP-клиента

```
dhcpcd
```

Убедиться в получении адреса можно, введя команду

```
ip a
```

Вся информация по получению ip адреса отображена в журнальном файле. Получим всю информацию по аренде сетевого адреса

```
journalctl | grep dhcpcd
```

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Назначение команд ОС Альт, используемых в работе.
2. DHCP. Модель работы.
3. Процесс настройки DHCP-сервера.
4. Назначение строки BOOTPROTO в файле options сетевого интерфейса.
5. Назначение сетевого менеджера в ОС Альт.

Практикум № 6
Организация сеанса

1. ЦЕЛЬ РАБОТЫ

Познакомится с программой Wireshark. Разобраться с помощью wireshark с трехзвенным рукопожатием и фактом разрыва соединения.

2. ЗАДАНИЕ НА РАБОТУ

1. Установите соединение с помощью утилиты iperf между двумя компьютерами и. Проследите передачу пакетов по протоколу TCP.
2. Зафиксируйте трехзвенное рукопожатие. Перечислите пакеты, участвующие в установке соединения вместе с флагами.
3. Зафиксируйте факт разрыва соединения. Перечислите пакеты, участвующие в разрыве соединения вместе с флагами.

3. ТЕРМИНЫ И ИСПОЛЬЗУЕМЫЕ КОМАНДЫ

Термины

Transmission Control Protocol (TCP, протокол управления передачей) –один из основных протоколов передачи данных интернета, предназначенный для управления передачей данных.

В стеке протоколов TCP/IP выполняет функции транспортного уровня модели OSI.

Механизм TCP предоставляет поток данных с предварительной установкой соединения, осуществляет повторный запрос данных в случае потери данных и устраняет дублирование при получении двух копий одного пакета, гарантируя тем самым, в отличие от UDP, целостность передаваемых данных и уведомление отправителя о результатах передачи.

Wireshark – это мощный сетевой анализатор, который может использоваться для анализа трафика, проходящего через сетевой интерфейс вашего компьютера. Он может понадобиться для обнаружения и решения проблем с сетью, отладки ваших веб-приложений, сетевых программ или сайтов.

Флаги (управляющие биты):

URG – Поле "Указатель важности" задействовано (Urgent pointer field is significant).

ACK – Поле "Номер подтверждения" задействовано (Acknowledgement field is significant).

PSH – (Push function) инструктирует получателя протолкнуть данные, накопившиеся в приемном буфере, в приложение пользователя.

RST – Оборвать соединения, сбросить буфер (очистка буфера) (Reset the connection).

SYN – Синхронизация номеров последовательности (Synchronize sequence numbers).

FIN (final, бит) – флаг, будучи установлен, указывает на завершение соединения (FIN bit used for connection termination).

SYN и FIN используются только при открытии и закрытии соединения.

ACK – Флаг в TCP сегменте, установка которого означает, что поле «Номер подтверждения» задействовано. Если установлен флаг ACK, то это поле содержит порядковый номер, ожидаемый получателем в следующий раз. Помечает этот сегмент как подтверждение получения.

Процесс начала сеанса TCP, также называемый «рукопожатие» (англ. handshake), состоящий из трёх шагов:

1. Клиент, который намеревается установить соединение, посылает серверу сегмент с номером последовательности и флагом SYN.

- Сервер получает сегмент, запоминает номер последовательности и пытается создать сокет (буферы и управляющие структуры памяти) для обслуживания нового клиента;
- В случае успеха сервер посылает клиенту сегмент с номером последовательности и флагами SYN и ACK, и переходит в состояние SYN-RECEIVED;
- В случае неудачи сервер посылает клиенту сегмент с флагом RST.

2. Если клиент получает сегмент с флагом SYN, то он запоминает номер последовательности и посылает сегмент с флагом ACK.

- Если он одновременно получает и флаг ACK (что обычно и происходит), то он переходит в состояние ESTABLISHED;
- Если клиент получает сегмент с флагом RST, то он прекращает попытки соединиться;
- Если клиент не получает ответа в течение 10 секунд, то он повторяет процесс соединения заново.

3. Если сервер в состоянии SYN-RECEIVED получает сегмент с флагом ACK, то он переходит в состояние ESTABLISHED.

В противном случае после тайм-аута он закрывает сокет и переходит в состояние CLOSED.

Процесс называется «трёхэтапным рукопожатием» (англ. three way handshake), так как несмотря на то что возможен процесс установления соединения с использованием четырёх сегментов (SYN в сторону сервера, ACK в сторону клиента, SYN в сторону клиента, ACK в сторону сервера), на практике для экономии времени используется три сегмента.

Используемые команды

`which утилита` – команда для обнаружения установленных утилит.
`apt-get install утилита` – команда для установки указанной утилиты.
`iperf` – утилита для генерации трафика.

4. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

4.0. TCP протокол

Откроем Wireshark (рисунок 1). Если он не установлен, то необходимо установить его в соответствии с порядком, описанном в практикуме № 1.

```
[user@elbrus Рабочий стол]$ wireshark
```

Рисунок 1. Открытие Wireshark из терминала

В Wireshark выберем сетевой интерфейс `eth2`. В строку фильтра пропишем `tcp` и включим сбор трафика (рисунок 2). После чего воспользуемся утилитой iperf для генерации tcp-трафика. Для этого один компьютер будет сервером, а другой клиентом. Подробнее об использовании ipref и wireshark написано в практикуме № 1.

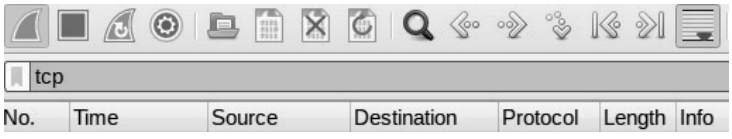


Рисунок 2. Фильтр по протоколу tcp

После некоторого времени работы Wireshark, остановим захват трафика. Из полученного трафика определим, когда произошло трёхзвенное рукопожатие. На рисунке 3 оно изображено на пакетах с номерами 10-12.

```
[SYN] seq=0
[SYN,ACK] Seq=0, Ack=1
```

```
[ACK] Seq=1, Ack=1
```

No.	Time	Source	Destination	Protocol	Length	Info
10	11.304630	10.10.1.1	10.10.2.1	TCP	68	39626 → 12345 [SYN] Seq=0 Win=43690 Len=0 MSS=65495 SACK_PERM…
11	11.304638	10.10.1.1	10.10.2.1	TCP	68	12345 → 39626 [SYN, ACK] Seq=0 Ack=1 Win=43690 Len=0 MSS=6549…
12	11.304646	10.10.1.1	10.10.2.1	TCP	56	39626 → 12345 [ACK] Seq=1 Ack=1 Win=43776 Len=0
13	11.304696	10.10.1.1	10.10.2.1	TCP	80	39626 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=43776 Len=24
14	11.304709	10.10.1.1	10.10.2.1	TCP	56	12345 → 39626 [ACK] Seq=1 Ack=25 Win=43776 Len=0
15	11.304851	10.10.1.1	10.10.2.1	TCP	21944	39626 → 12345 [ACK] Seq=25 Ack=1 Win=43776 Len=21888

Рисунок 3. Пример tcp-трафика

4.1. Установление соединения

Теперь увидим трёхзвенное рукопожатие при входе на какой-либо сайт. Для этого в фильтре введём `ip.addr==194.226.28.20`, вы можете набрать ip-адрес любого другого сайта (для определения ip адреса другого сайта можно воспользоваться утилитой `dig`), затем нажимаем на синий плавник, который расположен в верхнем левом углу (рисунок 4).

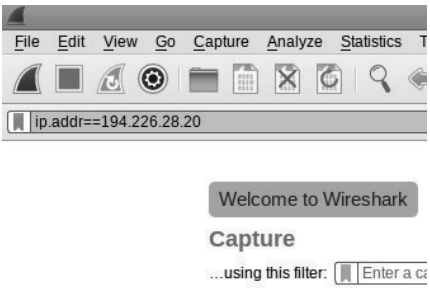


Рисунок 4. Пример введённого фильтра

Затем нажимаем на синюю стрелочку, расположенную справа в адресной строке Wireshark, тогда будут отображаться только пакеты, принадлежащие указанному ip-адресу.

После чего откроем браузер Mozilla Firefox, либо через терминал, набрав команду Mozilla, либо через меню (Приложения→ Сеть → Mozilla Firefox). Набираем в адресной строке браузера тот сайт, ip которого был выбран, и переходим на него, дожидаясь полной загрузки. В нашем случае это сайт mtuci.ru (рисунок 5).

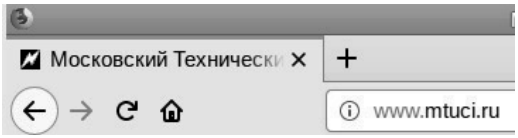


Рисунок 5. Открытие страницы сайта mtuci.ru

Возвращаемся в Wireshark, останавливаем передачу пакетов, нажав на красный квадрат (расположен слева вверху) и видим трехзвенное рукопожатие (рисунок 6. Из рисунка видно, что в соединении участвуют порты 43854 и 80 (порт http), пакеты с ip-адресами 10.0.2.15 (наш ip) и 194.226.28.20 (ip сайта).

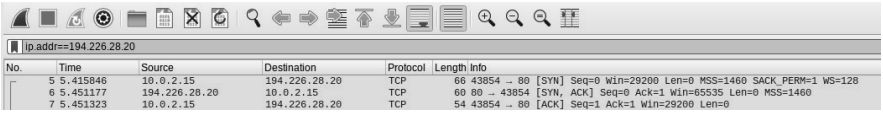


Рисунок 6. Пример трёхзвенного рукопожатия

4.2. Разрыв соединения

Факт разрыва соединения, можно зафиксировать, когда в процессе разрыва участвуют флаги и пакеты с ip-адресами 10.0.2.15 и 194.226.28.20 (рисунок 7).

Рисунок 7. Процесс разрыва соединения

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Назначение и основные функции анализатора протоколов wireshark.
2. Формат заголовка протокола TCP.
3. Три этапа взаимодействия по протоколу TCP.
4. Поля и флаги установления TCP соединения.
5. Поля и флаги разрыва TCP соединения.

Практикум № 7
Преобразование имен на основе DNS

1. ЦЕЛЬ РАБОТЫ

Ознакомится с файлом настроек /etc/resolv.conf. Научится определять адреса сайтов; локального, почтового, FTP и поискового сервера. Научится узнавать количество серверов корневой зоны и их название, а также находить сервер, ответственный за корневую зону и его адрес.

2. ЗАДАНИЕ НА РАБОТУ

1. Ознакомиться с содержимым файла настроек `/etc/resolv.conf`. Определить адреса локального сервера(серверов) и поискового сервера, заполнить таблицу 1.

Таблица 1. Поисковые и локальные сервера

Поисковый	
Локальный	

2. Определить адреса 4 случайных сайтов с помощью утилиты `dig` и заполнить таблицу 2.

Таблица 2. Адреса четырёх случайных сайтов

Имя	Адрес	Команда

3. Определить адреса почтового сервера и сервера FTP basealt.ru и записать их в таблицу 3.

Таблица 3. Почтовые сервера и FTP сервер basealt.ru

Имя	Адрес	Команда

4. Определите все сервера, принадлежащие mail.ru, и занести их в таблицу 4

Таблица 4. Сервера, принадлежащие mail.ru

Имя	Адрес	Команда

5. Узнать название сетевых ресурсов по их адресам

Таблица 5. Адреса сетевых ресурсов

Адреса	Сетевые ресурсы
194.226.28.20	
104.154.127.47	
91.201.52.134	
92.255.60.103	

6. Определить количество серверов корневой зоны и сервер, ответственный за корневую зону.

7. Определить количество серверов зоны su и сервер, ответственный за эту зону.

8. Определить количество серверов arpa.

Таблицы необходимо создать с помощью текстового редактора LibreOffice Writer и сохранить на рабочем столе. Название файла должно содержать номер группы и фамилию студента.

3. ТЕРМИНЫ И ИСПОЛЬЗУЕМЫЕ КОМАНДЫ

Термины

Адрес сайта – это уникальное наименование интернет-ресурса, лицо, по которому пользователь может найти его в сети. Локальной, почтовой, FTP и поисковой сервер

DNS (англ. Domain Name System – система доменных имен) – компьютерная распределенная система для получения информации о доменах. Все привыкли, что адрес веб-сайта написан в виде набора букв, очень удобных

для восприятия, например, google.com или mail.ru. Эти буквенные адреса работают именно благодаря системе доменных имен. Для адресов интернет-узлов применяется специальная цифровая кодировка, так называемые IP-адреса, а задача ДНС в том, чтобы связать названия интернет-сайтов в буквенном виде с IP в виде цифр.

Корневые серверы (первичные) DNS – DNS-серверы, обеспечивающие работу корневой зоны DNS в сети Интернет. Корневые сервера DNS отвечают на запросы других DNS-серверов в ходе трансляции доменных имён в IP-адреса и позволяют получить список DNS-серверов для любого домена верхнего уровня (TLD): RU, COM, NET, MUSEUM и др.

Вторичный сервер – это сервер, который передает полную информацию о зоне для других серверов (первичных или вторичных) и накапливает файл на своем локальном диске. Они создаются для обеспечения избыточности и разгрузки первичной зоны. Каждая копия базы данных DNS, однако, является доступной только для чтения, поскольку все изменения в записи вносятся в первичной зоне.

Типы DNS:

MX (почтовый сервер). Эта запись создает поддомен, который обслуживается внутренним (своим) почтовым сервером.

CNAME (каноническое имя -canonical name record). Запись типа CNAME позволяет иметь и использовать на сервере более одного имени домена (хоста).

Сначала создается одна запись типа A, для одного IP адреса. Имя домена в записи типа A, называется каноническим именем. Другие домены называют мнемонические. Мнемонические имена могут быть псевдонимами (произвольными именами) или субдоменами.

Тип записи NS (сервер имён). Это, пожалуй, самый важный тип записи. Он определяет домены (адреса) DNS серверов, обслуживающих этот домен.

Запись типа SOA показывает, где храниться на каком сервере лежит основная информация об этом домене. В записи типа SOA указывается полное, уточненное доменное имя зоны. Уточненное доменное имя должно оканчиваться точкой. В записи SOA может стоять символ @, вместо уточненного имени. В этом случае, доменное имя будет взято из файла конфигурации.

Тип записи A (address record или адрес Internet 4) привязывает конкретное доменное имя на определенный, точный IP-адрес.

PTR запись – это как обратная версия A записи. A запись связывает доменное имя с IP адресом, а PTR запись связывает IP адрес с именем хоста. Однако эти две записи являются независимыми друг от друга. К примеру, A

запись host.ru может быть привязана к IP адресу 21.21.128.xx, тогда как 23.23.128.xx может быть связан с совершенно другим именем хоста.

Используемые команды

`dig` - утилита (DNS-клиент), предоставляющая пользователю интерфейс командной строки для обращения к системе DNS. Позволяет задавать различные типы запросов и запрашивать произвольно указываемые сервера.

`nslookup` — утилита, предоставляющая пользователю интерфейс командной строки для обращения к системе DNS (проще говоря, DNS-клиент). Позволяет задавать различные типы запросов и опрашивать произвольно указываемые сервера.

`cat` – команда для работы с файлами.

`host` – команда, позволяющая преобразовывать ip адрес в название сетевого ресурса и наоборот.

4. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

4.1. Изучение файла resolv.conf

Используем команду `cat` для открытия файла `resolv.conf`

```
cat /etc/resolv.conf
```

Результат выполнения изображён на рисунке 1. Из рисунка видно, что в файле находится название локального (nameserver) и поискового (search) серверов. Полученные данные необходимо занести в таблицу 1.

```
[user@elbrus Рабочий стол]$ cat /etc/resolv.conf
# Generated by resolvconf
# Do not edit manually, use
# /etc/net/ifaces/<interface>/resolv.conf instead.
search home host-15.localdomain
nameserver 8.8.8.8
```

Рисунок 1. Содержимое файла resolv.conf

4.2. Определение адресов различных сайтов

Для определения адреса сайта можно воспользоваться командой `dig` (рисунок 2) или командой `nslookup` (рисунок 3). Например, получим адрес сайта ya.ru

```
dig ya.ru
```

```
nslookup ya.ru
```

Необходимо определить адреса четырёх различных сайтов и заполнить таблицу 2.

```
[user@elbrus Рабочий стол]$ dig ya.ru

; <<>> DiG 9.10.6 <<>> ya.ru
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 58426
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;ya.ru.                         IN      A

;; ANSWER SECTION:
ya.ru.                  255     IN      A       87.250.250.242

;; Query time: 31 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Mon Jan 07 11:39:42 MSK 2019
;; MSG SIZE  rcvd: 50
```

Рисунок 2. Применение команды `dig` для определения адреса сайта

Из рисунка 2 видно, что адрес сайта ya.ru – 87.250.250.242.

```
[user@elbrus Рабочий стол]$ nslookup ya.ru
Server:         8.8.8.8
Address:        8.8.8.8#53

Non-authoritative answer:
Name:   ya.ru
Address: 87.250.250.242
```

Рисунок 3. Применение команды `nslookup` для определения адреса сайта

4.3. Определение почтового и FTP серверов basealt.ru

Для того чтобы узнать адрес почтового сервера и сервера FTP можно использовать либо `dig`, либо `nslookup`. Обе команды работают примерно одинаково, для примера рассмотрим вариант с командой `nslookup`. Для определения почтового сервера используем команду (рисунок 4)

```
nslookup -type=mx basealt.ru
```

где

mx – mail exchange (почтовый обмен) – это запись на сервере имен (DNS), которая указывает на сервер почты для данного домена.

-type — тип информации, которую хотим получить.

```
[user@elbrus Рабочий стол]$ nslookup -type=mx basealt.ru
Server:         8.8.8.8
Address:        8.8.8.8#53

Non-authoritative answer:
basealt.ru      mail exchanger = 10 air.basealt.ru.

Authoritative answers can be found from:
```

Рисунок 4. Почтовый сервер basealt.ru

Чтобы узнать адрес FTP сервера, используем команду `dig`. Команда выглядит следующим образом (рисунок 5)

```
dig ftp.basealt.ru
```

```
[user@elbrus Рабочий стол]$ dig ftp.basealt.ru

; <<>> DiG 9.10.6 <<>> ftp.basealt.ru
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 24235
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;ftp.basealt.ru.                        IN      A

;; ANSWER SECTION:
ftp.basealt.ru.         21599   IN      CNAME   ftp.altlinux.org.
ftp.altlinux.org.       21599   IN      A       62.152.55.238

;; Query time: 126 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Mon Jan 07 13:41:22 MSK 2019
;; MSG SIZE  rcvd: 89
```

Рисунок 5. Определение ftp сервера

Из полученных данных заполните таблицу 3.

4.4. Определение серверов сайта mail.ru

Чтобы узнать все сервера, потребуется несколько команд

```
nslookup mail.ru
nslookup ftp.mail.ru
nslookup mx.mail.ru
nslookup -type=ns mail.ru
```

Команда `nslookup mail.ru` выводит адреса серверов сайта mail.ru (рисунок 6).

Команда `nslookup ftp.mail.ru` выводит адреса ftp серверов сайта mail.ru (рисунок 7).

Команда `nslookup mx.mail.ru` выводит почтовые сервера сайта mail.ru (рисунок 8).

```
[user@elbrus Рабочий стол]$ nslookup mail.ru
Server:         8.8.8.8
Address:        8.8.8.8#53

Non-authoritative answer:
Name:   mail.ru
Address: 217.69.139.200
Name:   mail.ru
Address: 217.69.139.201
Name:   mail.ru
Address: 94.100.180.199
Name:   mail.ru
Address: 94.100.180.201
```

Рисунок 6. Адреса серверов сайта mail.ru

```
[user@elbrus Рабочий стол]$ nslookup ftp.mail.ru
Server:         8.8.8.8
Address:        8.8.8.8#53

Non-authoritative answer:
Name:   ftp.mail.ru
Address: 217.69.139.87
Name:   ftp.mail.ru
Address: 94.100.180.87
```

Рисунок 7. Адреса ftp серверов сайта mail.ru

```
[user@elbrus Рабочий стол]$ nslookup mx.mail.ru
Server:         8.8.8.8
Address:        8.8.8.8#53

Non-authoritative answer:
Name:   mx.mail.ru
Address: 94.100.180.87
Name:   mx.mail.ru
Address: 217.69.139.87
```

Рисунок 8. Адреса почтовых серверов сайта mail.ru

Определим, сколько ns-серверов у сайта mail.ru, введя команду

`nslookup -type=ns mail.ru`

Получим, что серверов три. Для каждого из них используем команду

`nsX.mail.ru`

где X – номер сервера.

Результат выполнения команд представлен на рисунке 9.

```
[user@elbrus Рабочий стол]$ nslookup -type=ns mail.ru
Server:         8.8.8.8
Address:        8.8.8.8#53

Non-authoritative answer:
mail.ru nameserver = ns2.mail.ru.
mail.ru nameserver = ns1.mail.ru.
mail.ru nameserver = ns3.mail.ru.

Authoritative answers can be found from:

[user@elbrus Рабочий стол]$ nslookup ns1.mail.ru
Server:         8.8.8.8
Address:        8.8.8.8#53

Non-authoritative answer:
Name:   ns1.mail.ru
Address: 217.69.139.112

[user@elbrus Рабочий стол]$ nslookup ns2.mail.ru
Server:         8.8.8.8
Address:        8.8.8.8#53

Non-authoritative answer:
Name:   ns2.mail.ru
Address: 94.100.180.138
```

Рисунок 9. Список ns-серверов и их адреса для сайта mail.ru

4.5. Определение имени сетевого ресурса

Для преобразования адресов в названия ресурсов воспользуемся утилитой host

```
host –t ptr ip
```

Параметр `-t` указывает на тип записи. Запись `ptr` связывает ip адрес с именем хоста. Для примера выполним команду `host –t ptr 217.197.227.132` (рисунок 10).

```
[user@elbrus Рабочий стол]$ host -t ptr 217.197.227.132
132.227.197.217.in-addr.arpa domain name pointer hermitagemuseum.org.
```

Рисунок 10. Пример выполнения команды `host`

4.6. Изучение серверов корневой зоны

Чтобы узнать количество серверов корневой зоны, воспользуемся командой (рисунок 11)

```
nslookup -type=ns .
```

```
[user@elbrus Рабочий стол]$ nslookup -type=ns .
Server:         8.8.8.8
Address:        8.8.8.8#53

Non-authoritative answer:
.       nameserver = a.root-servers.net.
.       nameserver = b.root-servers.net.
.       nameserver = c.root-servers.net.
.       nameserver = d.root-servers.net.
.       nameserver = e.root-servers.net.
.       nameserver = f.root-servers.net.
.       nameserver = g.root-servers.net.
.       nameserver = h.root-servers.net.
.       nameserver = i.root-servers.net.
.       nameserver = j.root-servers.net.
.       nameserver = k.root-servers.net.
.       nameserver = l.root-servers.net.
.       nameserver = m.root-servers.net.

Authoritative answers can be found from:
```

Рисунок 11. Список серверов корневой зоны

Для определения сервера, ответственного за корневую зону воспользуемся командой (рисунок 12)

```
nslookup -type=soa .
```

После определения этого сервера, получим его ip адрес, используя команду `nslookup`.

```
[user@elbrus Рабочий стол]$ nslookup -type=soa .
Server:         8.8.8.8
Address:        8.8.8.8#53

Non-authoritative answer:
.
        origin = a.root-servers.net
        mail addr = nstld.verisign-grs.com
        serial = 2019050200
        refresh = 1800
        retry = 900
        expire = 604800
        minimum = 86400
[user@elbrus Рабочий стол]$ nslookup a.root-servers.net
Server:         8.8.8.8
Address:        8.8.8.8#53

Non-authoritative answer:
Name:   a.root-servers.net
Address: 198.41.0.4
```

Рисунок 12. Сервер, ответственный за корневую зону и его ip адрес

4.7. Изучение серверов, ответственных за зону su

Для выполнения этого пункта будем использовать команды, которыми пользовались в пункте 4.5, но с одним отличием – поменяем “.” на “su” (рисунок 13)

```
nslookup -type=ns su
```

```
[user@elbrus Рабочий стол]$ nslookup -type=ns su
Server:         8.8.8.8
Address:        8.8.8.8#53

Non-authoritative answer:
su      nameserver = a.dns.ripn.net.
su      nameserver = b.dns.ripn.net.
su      nameserver = d.dns.ripn.net.
su      nameserver = e.dns.ripn.net.
su      nameserver = f.dns.ripn.net.

Authoritative answers can be found from:
```

Рисунок 13. Список серверов, ответственных за зону su

Определим сервер, ответственный за зону su и его адрес (рисунки 14 и 15)

```
nslookup -type=soa su
```

```
[user@elbrus Рабочий стол]$ nslookup -type=soa su
Server:         8.8.8.8
Address:        8.8.8.8#53

Non-authoritative answer:
su
        origin = a.dns.ripn.net
        mail addr = hostmaster.ripn.net
        serial = 650176902
        refresh = 86400
        retry = 14400
        expire = 2592000
        minimum = 3600

Authoritative answers can be found from:
```

Рисунок 14. Сервер ответственный за зону su

```
[user@elbrus Рабочий стол]$ nslookup a.dns.ripn.net
Server:         8.8.8.8
Address:        8.8.8.8#53

Non-authoritative answer:
Name:   a.dns.ripn.net
Address: 193.232.128.6
```

Рисунок 15. Адрес сервера, ответственного за зону su

4.8. Определение корневых серверов зоны arpa

Для получения всех корневых серверов зоны arpa воспользуемся командой

```
nslookup -type=ns arpa
```

Результат выполнения команды представлен на рисунке 16.

```
[user@elbrus Рабочий стол]$ nslookup -type=ns arpa
Server:         8.8.8.8
Address:        8.8.8.8#53

Non-authoritative answer:
arpa    nameserver = h.root-servers.net.
arpa    nameserver = l.root-servers.net.
arpa    nameserver = a.root-servers.net.
arpa    nameserver = i.root-servers.net.
arpa    nameserver = c.root-servers.net.
arpa    nameserver = e.root-servers.net.
arpa    nameserver = m.root-servers.net.
arpa    nameserver = k.root-servers.net.
arpa    nameserver = b.root-servers.net.
arpa    nameserver = f.root-servers.net.
arpa    nameserver = d.root-servers.net.
arpa    nameserver = g.root-servers.net.

Authoritative answers can be found from:
a.root-servers.net      internet address = 198.41.0.4
b.root-servers.net      internet address = 199.9.14.201
c.root-servers.net      internet address = 192.33.4.12
d.root-servers.net      internet address = 199.7.91.13
e.root-servers.net      internet address = 192.203.230.10
f.root-servers.net      internet address = 192.5.5.241
g.root-servers.net      internet address = 192.112.36.4
h.root-servers.net      internet address = 198.97.190.53
i.root-servers.net      internet address = 192.36.148.17
k.root-servers.net      internet address = 193.0.14.129
l.root-servers.net      internet address = 199.7.83.42
m.root-servers.net      internet address = 202.12.27.33
a.root-servers.net      has AAAA address 2001:503:ba3e::2:30
b.root-servers.net      has AAAA address 2001:500:200::b
c.root-servers.net      has AAAA address 2001:500:2::c
```

Рисунок 16. Список корневых серверов arpa

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. DNS. Регулирующие документы.
2. Типы DNS серверов.
3. Виды зон и записей в DNS.

4. Структура сообщения DNS. Формат запроса и ответа.
5. Технология anycast.
6. Угрозы и уязвимые места DNS.
7. Расширение безопасности DNSSEC.

Практикум № 8
Передача нешифрованной и зашифрованной информации между компьютерами

1. ЦЕЛЬ РАБОТЫ

Изучение и сравнение зашифрованных и незашифрованных соединений с помощью Wireshark. Ознакомление с утилитой ssh в дистрибутиве ОС Альт.

2. ЗАДАНИЕ НА РАБОТУ

1. Подключиться к консоли соседнего компьютера с помощью утилиты ssh;
2. Запустить удалённое графическое приложение на компьютере соседа;
3. Сравнить с помощью Wireshark сеансы netcat и ssh. Зафиксировать особенности этих соединений (порт отправителя, порт получателя, протокол передачи данных, указать какое из соединений шифруется);
4. Сравнить с помощью Wireshark HTTP и HTTPS соединения. Зафиксировать особенности таких соединений (протокол передачи данных, указать какое из соединений шифруется).

3. ТЕРМИНЫ И ИСПОЛЬЗУЕМЫЕ КОМАНДЫ

Термины

Шифрование – обратимое преобразование информации с целью сокрытия её содержания для защиты от несанкционированного доступа к ней.

Дешифрование – процесс восстановления зашифрованной информации в её первоначальный вид.

Ассиметричное шифрование (шифрование с открытым ключом) – алгоритм шифрования в котором, в отличие от симметричного применяется два вида ключей: открытые и закрытые. Обладает худшей крипто стойкостью по сравнению с симметричным шифрованием, однако не требует защищенного канала связи для передачи ключа. Наиболее популярным алгоритмом ассиметричного шифрования является алгоритм RSA.

Публичный ключ (открытый ключ) – ключ, генерирующийся по особому алгоритму каждым из абонентов и передающийся в незашифрованном виде по каналу связи. Иными словами, злоумышленник может перехватить этот ключ, и использовать его для взлома. Применяется для шифрования информации.

Приватный ключ (закрытый ключ) – ключ, генерирующийся по особому алгоритму каждым из абонентов и не передающийся по каналу связи (хранится у каждого абонента). Применяется для дешифрования информации.

Используемые команды

`ssh` – утилита для регистрации на удаленной машине и выполнения на ней команд.

`ssh-keygen` – создание, обслуживание и преобразование аутентификационных ключей.

`ssh-copy-id` – копирование вашего публичного ключа для ssh соединения на указанную машину для указанного пользователя.

`netcat` – утилита для передачи текстовых сообщений по протоколу tcp между двумя машинами.

`netstat` – утилита для отображения содержания различных структур данных, связанных с сетью.

`wireshark` – приложение для “захвата” трафика с целью его последующего анализа.

`firefox` – веб-браузер.

4. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

4.1. Генерация ключа и подключение к соседнему компьютеру

Изначально для выполнения соединения по протоколу SSH потребуется создать SSH-ключ. Для его создания воспользуемся следующей командой

```
ssh-keygen
```

При её вводе консоль предложит создать новый SSH-ключ, состоящий из публичной и приватной частей. Публичная часть хранится в `/$HOME/.ssh/id_rsa.pub`, приватная часть в: `/$HOME /.ssh/id_rsa`, где `$HOME` – домашний каталог пользователя, под которым студент находится в терминале.

Без указания специальный параметров-ключей будет создан ключ для шифрования по алгоритму `rsa`.

Результат выполнения команды представлен на рисунке 1.

```
[user@elbrus Рабочий стол]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/user/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/user/.ssh/id_rsa.
Your public key has been saved in /home/user/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:uWEhTYi1GTQFcNCjUHHhuDbFEf0WgBbYq2yF8sNp+NY user@elbrus.localdomain
The key's randomart image is:
+---[RSA 2048]----+
|    .=@&X+.      |
|   . o*BO. .     |
|    ..+Boo. .    |
|   . ooo. oo     |
|    *++  S.      |
|   ..X. . o      |
|    + o  .       |
|     o E         |
|    .            |
+----[SHA256]-----+
```

Рисунок 1. Создание новое ssh ключа

После создания ключа, скопируем его на удалённую машину соседа, чтобы, в дальнейшем, подключаться по этому ключу без ввода пароля. Для этого используем следующую команду:

```
ssh-copy-id user@ipaddr
```

Выше в команде `user` – пользователь, под которым мы будем находиться на удалённой машине, `ipaddr` – ip адрес машины соседа. Для примера примем, что имя пользователя – `user1`, а ip адрес – `10.10.2.2`. Таким образом команда пример следующий вид

```
ssh-copy-id user@10.10.2.2
```

При вводе команды копирования SSH-ключа на удалённую машину потребуется ввести пароль от того пользователя, который был указан в команде `ssh-copy-id`. После выполнения всех манипуляций подключение к машине с ip адресом `192.168.253.1` под пользователем `user1` можно производить без ввода пароля.

На рисунке 2 представлен процесс использования команды `ssh-copy-id user@10.10.2.2`.

```
[user@elbrus Рабочий стол]$ ssh-copy-id user@10.10.2.2
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/user/.ssh/i
d_rsa.pub"
The authenticity of host '10.10.2.2 (10.10.2.2)' can't be established.
ED25519 key fingerprint is SHA256:zg1wHBLfA+5B1Vez/11d5c2BOT3Z83jh/Z3F/VMcc+k.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompt
ed now it is to install the new keys
user@10.10.2.2's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'user@10.10.2.2'"
and check to make sure that only the key(s) you wanted were added.
```

Рисунок 2. Передача ключа на удалённую машину

Для подключения используем команду (рисунок 3)

```
ssh user@10.10.2.2
```

```
[user@elbrus Рабочий стол]$ ssh user@10.10.2.2
[user@elbrus ~]$
```

Рисунок 3. Подключение к удалённой машине

При правильном последовательном выполнении всех вышеизложенных пунктов студент должен подключиться к удалённой машине соседа.

4.2. Подключение в качестве графической оболочки

В первом пункте уже было установлено ssh-соединение. Его необходимо закрыть, нажав комбинацию клавиш `CTRL+D`.

Для запуска удалённого графического приложения на компьютере соседа, необходимо подключиться к нему с помощью следующей команды:

```
ssh -YC user@192.168.253.1
```

При подключении не потребуется ввод пароля, так как манипуляции в первом пункте избавили нас от этой надобности.

Чтобы определить установленные соединения используем утилиту `netstat`

```
netstat -tpln
```

Расшифруем ключи, использованные в команде:

-t – отображение только tcp соединений;
-p – отображение pid и программ;
-l – отображение только «прослушивающих» сокетов;
-n – отображение сетевых адресов как числа.

Итого, получаем: отображение всех открытых tcp соединений в числовой форме вместе с pid и программой, которая использует это соединение. На рисунке 4 изображен пример выполнения команды и подчёркнуты используемые порты.

```
[user@elbrus ~]$ netstat -tpln
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address               Foreign Address             State       PID/Program name
tcp        0      0 0.0.0.0:22                  0.0.0.0:*                   LISTEN      -
tcp        0      0 127.0.0.1:6010              0.0.0.0:*                   LISTEN      -
tcp        0      0 0.0.0.0:445                 0.0.0.0:*                   LISTEN      -
tcp        0      0 0.0.0.0:139                 0.0.0.0:*                   LISTEN      -
tcp        0      0 0.0.0.0:5355                0.0.0.0:*                   LISTEN      -
tcp        0      0 :::22                       :::*                        LISTEN      -
tcp        0      0 :::631                      :::*                        LISTEN      -
tcp        0      0 ::1:6010                    :::*                        LISTEN      -
tcp        0      0 :::445                      :::*                        LISTEN      -
tcp        0      0 :::139                      :::*                        LISTEN      -
tcp        0      0 :::5355                     :::*                        LISTEN      -
```

Рисунок 4. Пример выполнения команды `netstat -tpln`

Откроем приложение `pluma` на удалённой машине

```
pluma
```

4.3. Анализ трафика netcat и ssh

Для выполнения этого пункта необходимо включить Wireshark. Воспользуемся терминалом, введя в него команду

```
wireshark
```

На одном из компьютеров требуется активировать утилиту `netcat` на «прослушивание» всех входящих запросов по указанному порту

```
netcat -l port
```

Выше, `port` – номер порта «прослушивания», по которому ожидается поступление запроса. Например, `netcat -l 12345`.

На другом компьютере требуется подключиться к компьютеру, где была активирована утилита netcat на прослушивание входящих запросов

```
netcat ipaddr port
```

Выше, `ipaddr` – ip адрес машины, на которой активен `netcat`, `port` – номер порта, по которому ожидается поступление запроса. Для примера, как и ранее, примем, что ip адрес машины – `10.10.2.2`, а порт – `12345`. Получим следующие команды:

- Для удалённой машины

```
netcat -l 12345
```

- Для локальной машины

```
netcat 10.10.2.2 12345
```

Теперь отследим этот процесс в Wireshark. Подробнее об использовании Wireshark написано в практикуме № 1. При вводе какой-либо строки в терминале с включенной утилитой `netcat`, в Wireshark отразится пакет, который передаётся с локальной машины на удалённую на порт `12345`. При этом всю передаваемую информацию и текст можно увидеть на экране.

Перейдём ко второй части задания – вновь подключимся по ssh к удалённой машине соседа (процедура подключения подробно описана в пункте 4.1). Затем так же, как и выше, необходимо определить порт подключения.

Пример перехвата пакетов для утилиты `netcat` изображён на рисунке 5. Содержимое одного из пакетов изображено на рисунке 6. Как видно из содержимого, передаваемое сообщение не шифруется. На рисунке 7 изображены пакеты утилиты `ssh`, а на рисунке 8 – содержимое одного из пакетов.

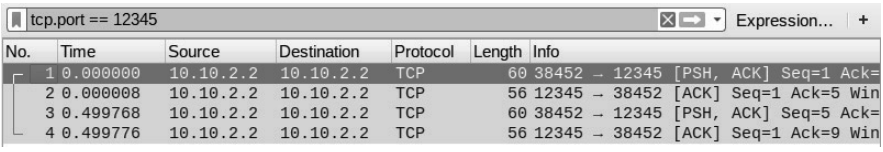


Рисунок 5. Пакеты, передающиеся при использовании `netcat`

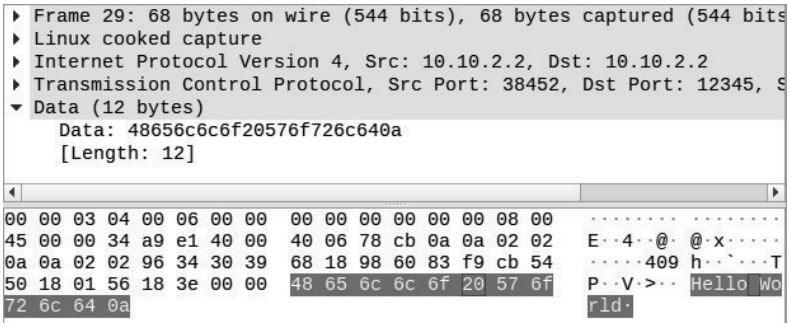


Рисунок 6. Содержимое пакета при использовании `netcat`

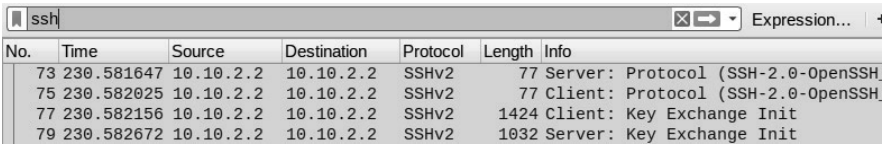


Рисунок 7. Пакеты, передающиеся при использовании `ssh`

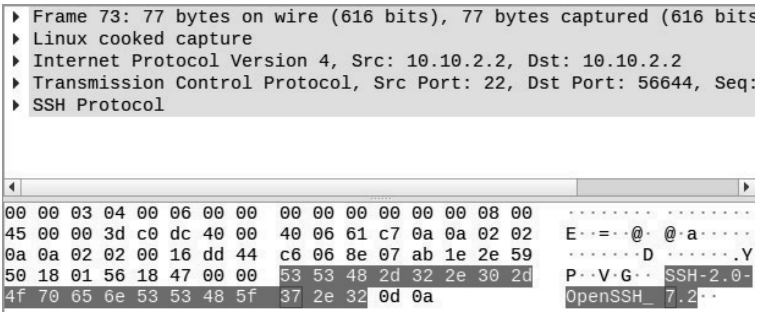


Рисунок 8. Содержимое пакета при использовании `ssh`

4.4. Анализ трафика HTTP и HTTPS

В данном пункте, как и в пункте три, потребуется использовать приложение Wireshark. Для начала выберем два сайта. Первый – соединение к которому происходит по протоколу HTTP, второй – соединение по HTTPS. Например, сайт `http://www.mtuci.ru` и `https://habr.com/ru`.

Откроем Mozilla Firefox, для этого в терминале введём команду:

```
firefox
```

В открывшемся окне в адресную строку введём первый адрес. В нашем примере это `http://www.mtuci.ru`. При этом в wireshark будут отображены пакеты, которые были получены при подключении к сайту. В отчёт необходимо занести протокол, по которому состоялось соединение.

Теперь подключимся ко второму сайту - `https://habr.com/ru`. Для него так же определим протокол, по которому произошло соединение. Полученный протокол требуется занести в отчёт. Дополнительно в отчёте необходимо указать какое из этих двух соединений шифруется.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Шифрование, дешифрование. Типы алгоритмов.
2. Симметричное шифрование. Особенности, угрозы.
3. Ассиметричное шифрование. Особенности, угрозы.
4. Протокол SSH. Назначение.
5. SSH-1. Процесс установления соединения, угрозы.
6. SSH-2. Назначение входящих в него протоколов.
7. Архитектура протокола. Формат пакетов.

Практикум № 9
Трансляция адресов

1. ЦЕЛЬ РАБОТЫ

Целью лабораторной работы является освоение навыков настройки статической и динамической трансляции адресов. Изучение механизма NAT.

2. ЗАДАНИЕ НА РАБОТУ

Компьютеры соединены по схеме, изображённой на рисунке 1.

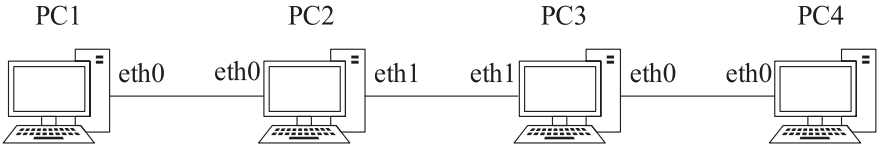


Рисунок 1. Схема сети

1. Настройка сетевых интерфейсов
   - ☐ Необходимо настроить конфигурационные файлы в /etc/net/ifaces.
2. Настройка трансляции адресов
   - ☐ При обращении с узла 1 к службе, расположенной на узле 3, обеспечьте возможность скрытия узла 1 средствами узла 2;
   - ☐ Обеспечьте возможность переноса службы с узла 2 на узел 3 так, чтобы при обращении к ней с узла 1 узел 2 считался «прозрачным».

3. ТЕРМИНЫ И ИСПОЛЬЗУЕМЫЕ КОМАНДЫ

Термины

Трансляция сетевых адресов (Network Address Translation, NAT) — это подмена какого-либо адреса или порта в пакете. Она, как правило, требуется на границе между сетью компании и провайдером Интернет.

DNAT (Destination Network Address Translation) используется для преобразования адреса места назначения в IP заголовке пакета. Если пакет подпадает под критерий правила, выполняющего DNAT, то этот пакет, и все последующие пакеты из этого же потока, будут подвергнуты преобразованию адреса назначения и переданы на требуемое устройство, хост или сеть. Данное

действие может, к примеру, успешно использоваться для предоставления доступа к вашему web-серверу, находящемуся в локальной сети, и не имеющему реального IP адреса. Для этого вы строите правило, которое перехватывает пакеты, идущие на HTTP порт брандмауэра и выполняя DNAT передаете их на локальный адрес web-сервера. Для этого действия так же можно указать диапазон адресов, тогда выбор адреса назначения для каждого нового потока будет производиться случайным образом.

SNAT используется для преобразования сетевых адресов (Source Network Address Translation), т.е. изменение исходящего IP адреса в IP заголовке пакета. Например, это действие можно использовать для предоставления выхода в Интернет другим компьютерам из локальной сети, имея лишь один уникальный IP адрес. Для этого. необходимо включить пересылку пакетов (forwarding) в ядре и затем создать правила, которые будут транслировать исходящие IP адреса нашей локальной сети в реальный внешний адрес. В результате, внешний мир ничего не будет знать о нашей локальной сети, он будет считать, что запросы пришли с нашего брандмауэра.

SNAT допускается выполнять только в таблице nat, в цепочке POSTROUTING. Другими словами, только здесь допускается преобразование исходящих адресов. Если первый пакет в соединении подвергся преобразованию исходящего адреса, то все последующие пакеты, из этого же соединения, будут преобразованы автоматически и не пойдут через эту цепочку правил.

Маскарадинг – тип трансляции сетевого адреса, при которой адрес отправителя подставляется динамически, в зависимости от назначенного интерфейсу адреса.

Используемые команды

`ip` – утилита для работы с сетевыми интерфейсами.

`iptables` – утилита для управления работы межсетевым экранированием.

Некоторые опции команды `iptables`:

`-t таблица` – задаёт таблицу, к которой будет применена команда.

`-A цепочка` – добавляет одно или несколько правил в конец указанной цепочки. Виды цепочек – PREROUTING (для изменения всех входящих пакетов), INPUT (для пакетов, направленных в локальную сеть), FORWARD (для проходящих пакетов), OUTPUT (для пакетов, исходящих из локальной сети), POSTROUTING (для изменения всех исходящих пакетов).

`-j цель` – определяет цель правила; т.е., что делать, когда пакет попадает под условия правила. Целью может быть цепочка, определённая пользователем (отличная от цепочки правила), одна из встроенных целей, определяющая окончательное действие над пакетом, или расширение.

`-s ip` – адрес источника пакета.

`-d ip` – адрес назначения пакета.

`netcat` – утилита для передачи текстовых сообщений по протоколу tcp между двумя машинами.

`sysctl` – команда для считывания/записи параметров системы.

4. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

4.0. Предварительная настройка

Все команды в лабораторной работе выполняются с правами суперпользователя. Подробнее о том, как получить права суперпользователя написано в практикуме № 1.

Для примера выберем, что компьютеры имеют порядковые номера, как изображено на рисунке 2.

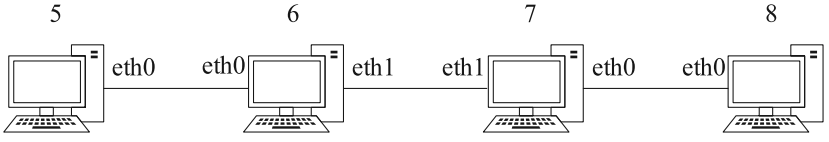


Рисунок 2. Пример схемы соединения с указанием номером рабочих мест

4.1. Настройка конфигурации

Пропишем ip адрес и маршрут для каждого компьютера. Адрес будем определять согласно номеру рабочего места компьютера по следующему правилу

`10.10.X.Y/24`

где

`X` – минимальный номер между двумя рабочими местами;

`Y` – номер рабочего места.

Согласно примеру с рисунка 2, компьютерам необходимо присвоить следующие адреса:

5:
- `10.10.5.5/24` на интерфейсе `eth0`

6:
- `10.10.5.6/24` на интерфейсе `eth0`
- `10.10.6.6/24` на интерфейсе `eth1`

7:
- `10.10.6.7/24` на интерфейсе `eth1`
- `10.10.7.7/24` на интерфейсе `eth0`

8:
- `10.10.7.8/24` на интерфейсе `eth0`

Для шестого и седьмого компьютеров разрешим перемещение трафика с интерфейса на интерфейс

```
sysctl –w net.ipv4.ip_forward=1
```

4.2. Настройка трансляции адресов

Для настройки трансляции сетевых адресов существует утилита iptables. Воспользуемся ей, для подмены источника пакета при передаче пакетов в локальной сети.

На шестом и седьмом компьютерах выполним

```
iptables -t nat -A POSTROUTING –o eth1 -j MASQUERADE
```

Для проверки правильности выполнения работы воспользуемся утилитой `iperf`, указав сервером машину с номером 5, а клиентом – машину с номером 7 или сервер – машина 8, а клиент – 6. Подробнее об использовании утилиты написано в практикуме № 1.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Трансляция адресов. DNAT, SNAT.
2. Маскарадинг и его отличия от DNAT и SNAT.
3. Преимущества и недостатки трансляции адресов.
4. Порядок прохождения таблиц и цепочек iptables.

Практикум № 10
Веб-сервер Apache

1. ЦЕЛЬ РАБОТЫ

Изучение и настройка веб-сервера Apache на операционной системе ОС Альт.

2. ЗАДАНИЕ НА РАБОТУ

1. Настройте веб-сервер таким образом, чтобы по адресу sX.example.com (где X – номер рабочего места) открывалась веб-страница с текстом “Test”.
2. При помощи механизма htaccess настройте пароль для доступа к веб-странице, созданной в пункте 1.
3. Настройте https доступ к странице, созданной в пункте 1.

3. ТЕРМИНЫ И ИСПОЛЬЗУЕМЫЕ КОМАНДЫ

Термины

Веб-сервер – это сервер, принимающий HTTP-запросы от клиентов и выдающий им HTTP-ответы, обычно вместе с HTML-страницей, изображением, файлом, медиа-потоком или другими данными. Веб-сервером называют как программное обеспечение, выполняющее функции веб-сервера, так и непосредственно компьютер, на котором это программное обеспечение работает. Клиент, которым обычно является веб-браузер, передаёт веб-серверу запросы на получение ресурсов, обозначенных URL-адресами.

Используемые команды

`which утилита` – команда для обнаружения установленных утилит.

`apt-get install утилита` – команда для установки указанной утилиты.

`nano, pluma` – текстовые редакторы.

`firefox` – веб-браузер.

`cat > файл` – команда для перевода всех сообщений в указанный файл.

`a2ensite` – активация сайта в apache.

`a2enmod` – активация модуля в apache.

`a2enport` – активация порта в apache.

`systemctl restart httpd2` – перезапуск apache.
`htpasswd файл` – команда для создания нового пользователя и сохранения его пароля в указанный файл.
`openssl` – команда для работы с ssl сертификатами.

4. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

4.1. Установка необходимых утилит

Перед началом выполнения лабораторной работы необходимо убедиться в том, что все необходимые утилиты были установлены, для этого получим права суперпользователя, использовав команду (подробнее об использовании команды см. практикум № 1)

```
su -
```

Далее, введём команду

```
which apache2
```

Если пакет не удалось обнаружить (рисунок 1), то установим его с помощью команды

```
apt-get install apache2
```

После ввода команды (рисунок 2) необходимо ввести `y`.

```
[root@elbrus ~]# which apache2
which: no apache2 in (/root/bin:/sbin:/usr/sbin:/usr/local/sbin:/bin:/usr/bin:/u
sr/local/bin)
```

Рисунок 1. Отсутствие пакета apache2 в системе

```
[root@elbrus ~]# apt-get install apache2
Чтение списков пакетов... Завершено
Построение дерева зависимостей... Завершено
Следующие дополнительные пакеты будут установлены:
  apache2-ab apache2-base apache2-cgi-bin apache2-cgi-bin-printenv apache2-cgi-bin-test-cgi
  apache2-datadirs apache2-htcacheclean apache2-htcacheclean-control apache2-html
  apache2-htpasswd apache2-httpd-worker apache2-icons apache2-mod_cache_disk apache2-mods
  condstopstart-common condstopstart-web libapr1 libaprutil1 perl-DBM perl-base
  webserver-cgi-bin-control webserver-common webserver-common-control
Следующие пакеты будут ОБНОВЛЕНЫ:
  perl-base
Следующие НОВЫЕ пакеты будут установлены:
  apache2 apache2-ab apache2-base apache2-cgi-bin apache2-cgi-bin-printenv
  apache2-cgi-bin-test-cgi apache2-datadirs apache2-htcacheclean
  apache2-htcacheclean-control apache2-html apache2-htpasswd apache2-httpd-worker
  apache2-icons apache2-mod_cache_disk apache2-mods condstopstart-common condstopstart-web
  libapr1 libaprutil1 perl-DBM webserver-cgi-bin-control webserver-common
  webserver-common-control
1 будет обновлено, 23 новых установлено, 0 пакетов будет удалено и 637 не будет обновлено.
Необходимо получить 3659kB архивов.
После распаковки потребуется дополнительно 4869kB дискового пространства.
Продолжить? [Y/n]
```

Рисунок 2. Установка пакета apache2

Результат установки изображён на рисунке 3.

```
auto
daemon
Running /usr/lib/rpm/posttrans-filetriggers
Завершено.
```

Рисунок 3. Результат установки пакета apache2

4.2. Создание веб-страницы

Стандартный конфигурационный файл для сайтов веб-сервера Apache находится в директории `/etc/httpd2/conf/sites-available`. Если актуальная директория – не домашняя (обозначается символом “~” в терминале), то необходимо предварительно выполнить команду `cd ~`.

```
cd /etc/httpd2/conf/sites-available
```

```
[root@elbrus ~]# cd /etc/httpd2/conf/sites-available/
[root@elbrus sites-available]#
```

Рисунок 4. Процесс перемещения в директорию sites-available

В этом каталоге находится файл `default.conf`. Скопируем его, в файл `site1.conf` (название выбрано для примера), использовав команду (рисунок 5)

```
cp default.conf site1.conf
```

```
[root@elbrus sites-available]# cp default.conf site1.conf
[root@elbrus sites-available]#
```

Рисунок 5. Копирование содержимого файла `default.conf` в файл `site1.conf`

Откроем ранее созданный файл в текстовом редакторе pluma, введя команду (рисунок 6)

```
pluma site1.conf
```

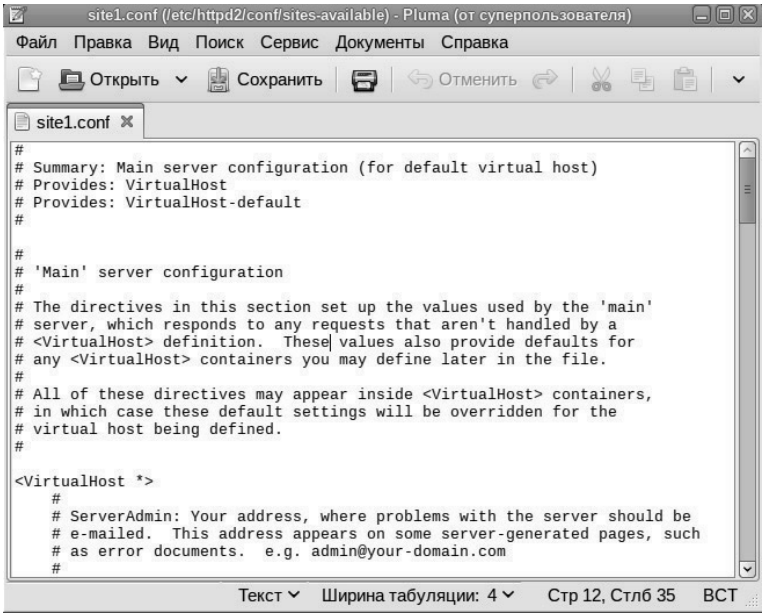


Рисунок 6. Фрагмент файла site1.conf

В открывшемся редакторе в разделе <VirtualHost *> необходимо заменить:

```
#ServerName www.example.com:80 →
ServerName www.s00.example.com

DocumentRoot "/var/www/html" →
DocumentRoot "/var/www/site1"

<Directory "/var/www/html"> →
<Directory "/var/www/site1">
```

Если студент выбрал другое название для файла site1.conf, то название необходимо сменить, чтобы они совпадали (пусть ранее была выполнена

команда `cp default.conf test007.conf`, тогда название директории должно иметь вид `/var/www/test007`). Результат изменений отображён на рисунке 7.

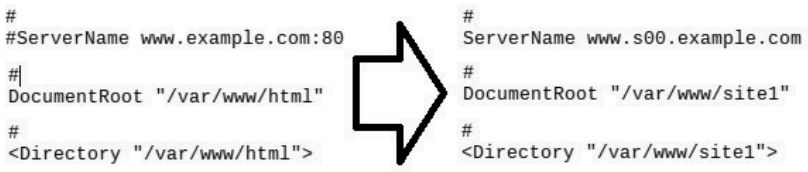


Рисунок 7. Строки, которые необходимо изменить

После всех манипуляций сохраним изменённый файл, кликнув на “Сохранить” (рисунок 8).

Рисунок 8. Внешний вид панели редактора Pluma

Теперь создадим директорию, которую указали в `DirectoryRoot`. Для этого перейдём в каталог `/var/www/`, использовав команду

```
cd /var/www/
```

После чего создадим новую директорию `site1` и перейдём в неё. Процесс выполнения отображён на рисунке 9.

```
[root@elbrus sites-available]# cd /var/www/
[root@elbrus www]# mkdir site1
[root@elbrus www]# cd site1
[root@elbrus site1]#
```

Рисунок 9. Процесс создания и перемещения в директорию `site1`

В каталоге создадим файл `index.html`, куда запишем `Test`. Для создания можно использовать текстовый редактор `pluma`, `nano`, либо утилиту `cat`. Рассмотрим вариант с использованием утилиты `cat`.

Введя команду

```
cat > index.html
```

Весь последующий набранный текст будет сохранён в указанный файл. Для окончания ввода потребуется нажать CTRL+D. Пример выполнения изображён на рисунке 10.

```
[root@elbrus site1]# cat > index.html
Test
[root@elbrus site1]#
```

Рисунок 10. Запись строки в текстовый файл

Теперь необходимо сопоставить созданной веб-странице ip адрес, чтобы браузер знал, где искать запрашиваемый сайт. Для этого откроем файл hosts, находящийся в /etc/hosts и допишем туда следующий текст: 127.0.0.5 www.s00.example.com (название сайта должно соответствовать тому, что было указано в ServerName). Для открытия файла воспользуемся утилитой nano

```
nano /etc/hosts
```

Результат выполнения изображён на рисунке 11. Количество строк может отличаться от того, что показаны на рисунке. Для сохранения файла нажмём CTRL+X, далее, следуя инструкциям, введём y и нажмём enter.

```
 GNU nano 2.2.4                Файл: /etc/hosts
127.0.0.1       localhost.localdomain localhost
127.0.0.5       www.s00.example.com
```

Рисунок 11. Содержимое файла /etc/hosts

Остался заключительный шаг – включение сайта и перезагрузка веб-сервера. Команда

```
a2ensite site1
```

Активирует сайт, но для её корректной работы необходим перезапуск веб-сервера с помощью команды (рисунок 12)

```
systemctl restart httpd2
```

```
[root@elbrus site1]# a2ensite site1
Site site1 installed;
       run service httpd2 condreload to fully enable.
[root@elbrus site1]# systemctl restart httpd2
```

Рисунок 12. Активация сайта site1

Для проверки правильности выполненной работы, откроем веб-браузер `firefox` и введём в адресную строку `www.s00.example.com`. Если, в итоге, будет открыта страница, где написано `Test`, то пункт задания выполнен верно. Пример страницы с надписью `Test` приведён на рисунке 13.

Рисунок 13. Фрагмента сайта, открытого в веб-браузере

4.3. Настройка доступа к веб-странице

Для начала требуется включить модули, которые отвечают за доступ по паролю. Для их активации модулей используется команда

```
a2enmod модуль
```

Список модулей представлен ниже:

- `auth_basic`;
- `authn_core`;
- `authn_file`;
- `authz_user`.

Листинг выполнения команды представлен на рисунке 14.

```
[root@elbrus ~]# a2enmod auth_basic
Module auth_basic installed;
        run service httpd2 condreload to fully enable.
[root@elbrus ~]# a2enmod authn_core
Module authn_core installed;
        run service httpd2 condreload to fully enable.
[root@elbrus ~]# a2enmod authn_file
Module authn_file installed;
        run service httpd2 condreload to fully enable.
[root@elbrus ~]# a2enmod authz_user
Module authz_user installed;
        run service httpd2 condreload to fully enable.
```

Рисунок 14. Активация модулей

Для активации запроса пароля потребуется создать файл в директории сайта: /var/www/site1 с названием .htaccess. Для этого воспользуемся текстовым редактором nano. В файл необходимо записать:

```
AuthType Basic
AuthName “Private area”
AuthUserFile “/var/www/site1/htpasswd”
Require user test
```

Выше:

AuthType Basic – для аутентификации использовать базовый модуль auth_basic (существует модули auth_digest и auth_form. Первый перенаправляет пользователя на отдельную страницу аутентификации, а второй на отдельную форму);

AuthName “Private area” – текст, который будет выведен как информация при запросе аутентификации;

AuthUserFile “/var/www/site1/htpasswd” – путь к файлу, в котором находятся пользователи и пароли от них. Только от имени тех, кто перечислен в файле можно зайти на веб-страницу.

Require user test – какой пользователь имеет доступ к веб-странице. В нашем примере – test.

Пример заполнения файла приведён на рисунке 15.

Рисунок 15. Содержимое файла .htaccess

Теперь создадим пользователя и пароль к нему, чтобы предоставить доступ на веб-странице. Для это используем команду htpasswd, которая создаст нового пользователя (ключ -c – создать) с именем test. Имя пользователя здесь и в строке Require user test должно совпадать. Листинг выполнения команды представлен на рисунке 16. Полная команда будет выглядеть так

```
htpasswd -c /var/www/site1/htpasswd test
```

```
[root@elbrus site1]# htpasswd -c /var/www/site1/htpasswd test
New password:
Re-type new password:
Adding password for user test
```

Рисунок 16. Создание пароля для пользователя `test`

Последний шаг – дописать `AllowOverride all` в файл `/etc/httpd2/conf/sites-available/site1.conf`, строку допишем после `Include conf/include/Directory_html_default.conf` (рисунок 17). С помощью этой строки веб-сервер переопределяет стандартные конфигурационные данные и подключает файл `.htaccess`. Для записи воспользуемся редактором `nano`

```
nano /etc/httpd2/conf/sites-available/site1.conf
```

```
<Directory "/var/www/site1">
        # Summary: Configure for html documents in DocumentRoot
        # Requires: Directory-html
        Include conf/include/Directory_html_default.conf
        AllowOverride all
</Directory>
```

Рисунок 17. Фрагмент файла `site1.conf`

Для вступления в силу всех изменений и подключенных модулей, необходимо выполнить команду

```
systemctl restart httpd2
```

О правильности выполненного задания можно судить по изменённой форме входа на веб-страницу. Теперь вместо отображения строки `Test`, предлагается ввести имя пользователя и пароль (рисунок 18).

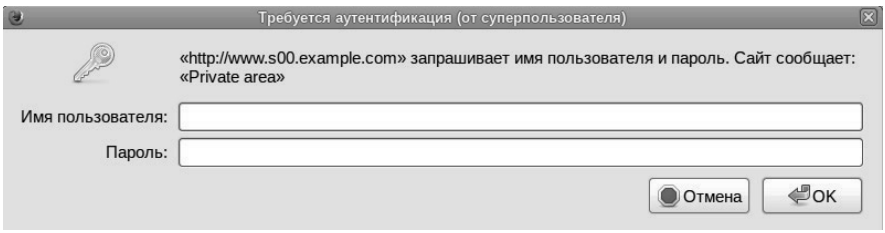


Рисунок 18. Меню аутентификации на сайте

После аутентификации веб-страница выдаст `Test`, как и в первом пункте задания (рисунок 19).

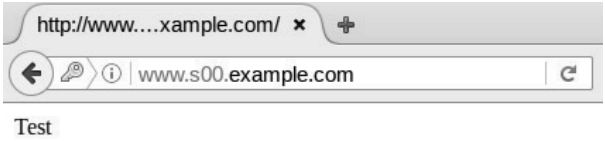


Рисунок 19. Фрагмент сайта в веб-браузере после успешной аутентификации

4.4. Настройка HTTPS доступа

Для работы с https необходимо установить модуль `apache_mod.ssl`, проверим, не установлен ли этот модуль, выполнив следующую команду

```
which apache2-mod_ssl
```

Если модуль не установлен (рисунок 20), то установим его, введя команду (рисунок 21)

```
apt-get install apache2-mod_ssl
```

```
[root@elbrus site1]# which apache2-mod_ssl
which: no apache2-mod_ssl in (/root/bin:/sbin:/usr/sbin:/usr/local/sbin:/bin:/us
r/bin:/usr/local/bin)
```

Рисунок 20. Проверка установленного модуля ssl

```
[root@elbrus site1]# apt-get install apache2-mod_ssl
Чтение списков пакетов... Завершено
Построение дерева зависимостей... Завершено
Следующие НОВЫЕ пакеты будут установлены:
  apache2-mod_ssl
0 будет обновлено, 1 новых установлено, 0 пакетов будет удалено и 637 не будет о
бновлено.
Необходимо получить 0B/113kB архивов.
После распаковки потребуется дополнительно 250kB дискового пространства.
Совершаем изменения...
Preparing...                ########################################### [100%]
1: apache2-mod_ssl          ########################################### [100%]
Running /usr/lib/rpm/posttrans-filetriggers
Завершено.
```

Рисунок 21. Установка модуля ssl для apache2

Для доступа по HTTPS перейдём в директорию, где находится файл `site1.conf` и создадим новый файл `site1_https.conf`, скопировав его из стандартного `default_https.conf`. Результат выполнения команды представлен на рисунке 22.

```
cd /etc/httpd2/conf/sites-available
cp default_https.conf site1_https.conf
```

Название `site1` должно совпадать с тем, что было создано для обычной веб-страницы в пункте 4.2

```
[root@elbrus site1]# cd /etc/httpd2/conf/sites-available/
[root@elbrus sites-available]# cp default_https.conf site1_https.conf
```

Рисунок 22. Копирование содержимого файла

Откроем файл через редактор `nano` (можно использовать любой другой текстовый редактор, например, `pluma`) и изменим следующие строки (рисунок 23):

```
DocumentRoot "/var/www/html" →
DocumentRoot "/var/www/site1"

ServerName www.example.com:443 →
ServerName www.s00.example.com:443

SSLCertificateFile
"/var/lib/ssl/certs/httpd2.cert" →
SSLCertificateFile "/var/www/site1/ssl.cert"

SSLCertificateKeyFile
"/var/lib/ssl/private/httpd2.key" →
SSLCertificateKeyFile "/var/www/site1/ssl.key"
```

Рисунок 23. Строки, которые необходимо изменить

Активируем модуль для работы с ssl и порт для https запросов (рисунок 24)

```
a2enmod ssl
a2enport https
```

```
[root@elbrus sites-available]# a2enmod ssl
Module ssl installed;
        run service httpd2 condreload to fully enable.
[root@elbrus sites-available]# a2enport https
Port config https installed;
        run service httpd2 condreload to fully enable.
```

Рисунок 24. Активация модуля ssl и порта https

Наконец, создадим самоподписанный ssl сертификат

```
openssl req -new -x509 -days 365 -nodes -out
/var/www/site1/ssl.cert -keyout /var/www/site1/ssl.key
```

Расшифруем команду выше:

`req` – создание запроса на подпись сертификата;

`-new` – создание нового сертификата;

`-x509` – создание самостоятельно подписанного сертификата;

`-days 365` – срок действия сертификата;

`-nodes` – не создавать кодовую фразу для защиты сертификата;

`-out <файл>` – имя файла создаваемого сертификата;

`-keyout <файл>` – имя файла создаваемого приватного ключа.

Процесс создания сертификата отображён на рисунке 25.

```
[root@elbrus sites-available]# openssl req -new -x509 -nodes -days 365 -out /va
r/www/site1/ssl.cert -keyout /var/www/site1/ssl.key
Generating a 2048 bit RSA private key
.....+++
.....
.....+++
writing new private key to '/var/www/site1/ssl.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [RU]:RU
State or Province Name (full name) []:
Locality Name (eg, city) []:
Organization Name (eg, company) []:
Organizational Unit Name (eg, section) []:
Common Name (e.g., your name or your server's hostname) []:
Email Address []:
```

Рисунок 25. Процесс создания сертификата

Для вступления в силу всех изменений и подключенных модулей, необходимо выполнить команду

```
systemctl restart httpd2
```

О правильности выполнения можно судить, зайдя в `firefox` и введя адрес `https://www.s00.example.com` (рисунок 26).

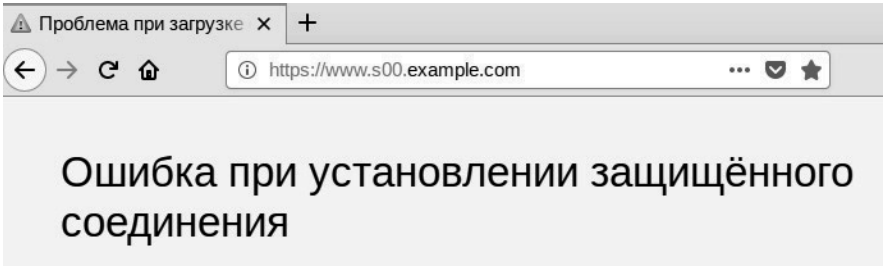


Рисунок 26. Результат установления защищённого соединения

Подключиться к веб-сайту с нашим самоподписанным сертификатом веб-браузер не разрешит. Но подключение по протоколу https состоялось, что говорит о правильности выполнения работы.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Назначение команд ОС Альт, используемых в работе.
2. Веб-сервер и его назначение.
3. Веб-сервер Apache.
4. Протокол SSL и его применение в Apache.
5. Гипертекстовый протокол HTTP. Регулирующие документы. Схема обмена сообщениями. Виды соединений.
6. Формат стартовой строки заголовка в HTTP. Список стандартных методов.
7. Формат стартовой строки ответа в HTTP. Коды состояний.

Практикум № 11
Использование пакетного менеджера RPM

1. ЦЕЛЬ РАБОТЫ

Научиться пользоваться пакетным менеджером RPM.

2. ЗАДАНИЕ НА РАБОТУ

1. Определите, где находится исполняемый файл, который запускает текстовый редактор `Pluma`.
2. Определите, какому пакету принадлежит исполняемый файл `pluma`.
3. Определите, какие файлы входят в этот пакет.
4. Определите версию, лицензию этого пакета, автора и дату сборки.
5. Определите, какие файлы и каким образом были изменены с момента установки пакета `setup`.
6. Вычислите общее количество пакетов в системе.
7. Определите, от каких файлов и пакетов зависит пакет `ipcalc`.

3. ТЕРМИНЫ И ИСПОЛЬЗУЕМЫЕ КОМАНДЫ

Термины

RPM (рекурсивный Package Manager — RPM — менеджер пакетов;) обозначает две сущности: формат пакетов программного обеспечения и программа, созданная для управления этими пакетами. Программа позволяет устанавливать, удалять и обновлять программное обеспечение. RPM является основным форматом пакетов в LSB.

Исполняемый файл — это файл, используемый для выполнения различных функций или операций на компьютере. В отличие от файлов с данными, исполняемый файл нельзя прочитать, потому что он был скомпилирован.

Анализ файлов - неотъемлемая часть работы с ними. Иногда возникает необходимость подсчитать количество строк или слов в тексте. С этой задачей эффективно справляется команда wc.

Используемые команды

`which утилита` – команда, обнаруживающая исполняющий файл для указанной утилиты.

`ll` – разновидность команды `ls`, с выводом полной информации о файлах.

`rmp` – команда для работы с пакетами. Список некоторых флагов:

`-i путь` – установить пакет;

`-u` – обновить пакет;

`-e` – удалить пакет;

`-ql` – выводит список файлов, входящих в пакет;

`-qi` – информация о пакете;

`-qa` – выводит список всех установленных пакетов в системе;

`-qf` – выводит, какому пакеты принадлежит файл;

`-V` – проверка целостности файла, выводит все изменения, которые произошли с файлом после установки пакета;

`-p` – выводит путь до файла, если он не установлен.

4. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

4.1. Определение местонахождения исполняемого файла

Чтобы определить, где находится исполняемый файл, который запускает текстовый редактор `Pluma`. Нам потребуется команда

```
which pluma
```

Затем проверим, является ли файл исполняемым, для этого перейдем в данный каталог, используя команду

```
cd /usr/bin/
```

Далее, выполним команду, которая выведет все файлы из данной папки с указанием прав использования для различных групп пользователей и обернём её в команду-фильтр `grep`, чтобы вывести только то, что соответствует редактору `pluma`

```
ll | grep pluma
```

Если в правах файла находится символ `x`, то файл является исполняемым. Результат выполнения команд, описанных выше, приведён на рисунке 1.

```
[user@elbrus Рабочий стол]$ which pluma
/usr/bin/pluma
[user@elbrus Рабочий стол]$ cd /usr/bin/
[user@elbrus bin]$ ll | grep pluma
-rwsr-xr-x  1 root root      772404 янв  8 05:56 pluma
```

Рисунок 1. Определение прав файла `pluma`

Если же символа нет, то необходимо перейти в режим суперпользователя, и набрать команду

```
chmod u+x /usr/bin/pluma
```

Выполнив её, файл `pluma` станет исполняемым для после чего необходимо вновь использовать команду `ll | grep pluma` (рисунок 2)

```
[root@elbrus bin]# chmod u+x /usr/bin/pluma
[root@elbrus bin]# cd /usr/bin/
[root@elbrus bin]# ll | grep pluma
-rwsr-xr-x  1 root root      772404 янв  8 06:06 pluma
```

Рисунок 2. Изменение прав файла `pluma`

4.2. Определение принадлежности пакету

Чтобы определить к какому пакету принадлежит исполняемый файл `pluma`, наберем следующую команду (рисунок 3)

```
rpm -qf /usr/bin/pluma
```

```
[user@elbrus bin]$ rpm -qf /usr/bin/pluma
mate-text-editor-1.12.2-alt1_1
```

Рисунок 3. Определение принадлежности пакету

4.3. Определение файлов, входящих в пакет

Чтобы определить какие файлы входят в этот пакет, наберем команду (рисунок 4)

```
rpm -ql mate-text-editor-1.12.2-alt1_1
```

```
[user@elbrus bin]$ rpm -ql mate-text-editor-1.12.2-alt1_1
/usr/bin/pluma
/usr/lib/pluma
/usr/lib/pluma/plugin-loaders
/usr/lib/pluma/plugin-loaders/libcloader.so
/usr/lib/pluma/plugin-loaders/libpythonloader.so
/usr/lib/pluma/plugins
```

Рисунок 4. Определение файлов, входящих в пакет

4.4. Определение сведений о пакете

Для определения версии, лицензии этого пакета, автора и даты сборки, воспользуемся командой (рисунок 5)

```
rpm -qi mate-text-editor-1.12.2-alt1_1
```

```
[user@elbrus bin]$ rpm -qi mate-text-editor-1.12.2-alt1_1
Name        : mate-text-editor             Relocations: (not relocatable)
Version     : 1.12.2                            Vendor: ALT Linux Team
Release     : alt1_1                         Build Date: Вт 05 апр 2016 17:27:59
Install date: Вт 01 янв 2019 22:42:11      Build Host: viy-sisyphus.hasher.altlinux.org
Group       : Редакторы            Source RPM: mate-text-editor-1.12.2-alt1_1.src.rpm
Size        : 2154175                        License: GPLv2+ and LGPLv2+
Packager    : Igor Vlasenko <viy@altlinux.ru>
URL         : http://mate-desktop.org
Summary     : Text editor for the MATE desktop
Description :
mate-text-editor is a small, but powerful text editor designed specifically for
the MATE desktop. It has most standard text editor functions and fully
supports international text in Unicode. Advanced features include syntax
highlighting and automatic indentation of source code, printing and editing
of multiple documents in one window.

mate-text-editor is extensible through a plugin system, which currently includes
support for spell checking, comparing files, viewing CVS ChangeLogs, and
adjusting indentation levels.
```

Рисунок 5. Определение сведений о пакете

4.5. Определение изменений в файлах

Чтобы определить какие файлы и каким образом были изменены с момента установки пакета `setup`, нам потребуется команда (рисунок 6)

```
rpm -V setup
```

```
[user@elbrus bin]$ rpm -V setup
..?..... c /etc/motd
..?..... c /etc/securetty
отсутствует /var/log/faillog
.M....G. g /var/log/lastlog
```

Рисунок 6. Определение изменений в файлах

4.6. Определение количества пакетов в системе

Для определения общего количества пакетов, установленных в системе, используем команду `rmp -qa`, обернув её в команду для подсчёта строк `wc -l` (рисунок 7)

```
rpm -qa | wc -l
```

```
[user@elbrus bin]$ rpm -qa| wc -l
1747
```

Рисунок 7. Количество установленных пакетов

4.7. Определение зависимости пакета

Для проверки от каких файлов и пакетов зависит пакет `ipcalc`, в начале, потребуется узнать путь до исполняемого файла, использовав команду

```
which ipcalc
```

Для определения списка пакетов, от которых зависит наш пакет (без которых его работа невозможна), используем команду

```
rpm –R ipcalc
```

Результат выполнения команды представлен на рисунке 8.

```
[user@elbrus bin]$ which ipcalc
/usr/bin/ipcalc
[user@elbrus bin]$ rpm -R ipcalc
rpmlib(PayloadFilesHavePrefix) <= 4.0-1
rpmlib(CompressedFileNames) <= 3.0.4-1
/usr/bin/perl
rpmlib(PayloadIsLzma) <= 4.4.2-1
```

Рисунок 8. Список пакетов, от которых зависит пакет `ipcalc`

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Назначение команд ОС Альт, используемых в работе.
2. Пакетный менеджер RPM.
3. Состав названия пакета-RPM.
4. Утилита RMP. Примеры применения.

Практикум № 12
Удалённый доступ по протоколу VNC»

1. ЦЕЛЬ РАБОТЫ

Изучить сетевой протокол VNC. Определить основные этапы и особенности использования протокола. Настройка удаленного доступа по протоколу VNC.

2. ЗАДАНИЕ НА РАБОТУ

1. Настройте VNC сервер таким образом, чтобы ваш сосед мог подключиться к новому сеансу на вашей машине с правами только на просмотр. Зафиксируйте соответствующие команды.
2. Зайдите в тот же сеанс, что и ваш сосед, запустите некоторые приложения. Проконтролируйте, что ваш сосед, видит тот же сеанс, что и вы. Зафиксируйте команды.

3. ТЕРМИНЫ И ИСПОЛЬЗУЕМЫЕ КОМАНДЫ

Термины

Virtual Network Computing (VNC) — система удалённого доступа к рабочему столу компьютера, использующая протокол RFB (англ. Remote FrameBuffer, удалённый кадровый буфер). Управление осуществляется путём передачи нажатий клавиш на клавиатуре и движений мыши с одного компьютера на другой и ретрансляции содержимого экрана через компьютерную сеть.

Используемые команды

`vncserver` – команда, позволяющая создать vnc-сервер удаленного доступа. При выполнении команды потребуется ввести два пароля. Первый пароль нужен для полного доступа с возможностью управления. Второй пароль для доступа в качестве зрителя.

`vncviewer ip:display` – команда для подключения к указанному дисплею vnc-сервера с указанным ip.

4. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

4.1. Создание vnc-сервера и подключение к нему

Для активации vnc-сервера зайдём в режим суперпользователя

```
su -
```

Далее выполним команду `vncserver`, после чего введём первый и второй пароли (подробнее о паролях в списке использованных команд)

```
vncserver
```

Результат выполнения команды представлен на рисунке 1.

```
[root@elbrus ~]# vncserver

You will require a password to access your desktops.

Password:
Verify:
Would you like to enter a view-only password (y/n)? y
Password:
Verify:

New 'host-15.localdomain:3 (root)' desktop is host-15.localdomain:3

Creating default startup script /root/.vnc/xstartup
Creating default config /root/.vnc/config
Starting applications specified in /root/.vnc/xstartup
Log file is /root/.vnc/host-15.localdomain:3.log
```

Рисунок 1. Активация vnc-сервера

Для подключения к текущему сеансу, определим ip компьютера, введя команду

```
ip a
```

Для примера, возьмём, что `ip - 172.16.1.22`.

На соседнем компьютере подключимся к созданному vnc-серверу

```
vncviewer 172.16.1.22:1
```

4.2. Запуск приложения на удалённом компьютере

Для начала создадим новый сеанс на компьютере с включённым vnc-сервером, используя команду (рисунок 2)

```
vncserver –list –DISPLAY:2
```

```
[root@elbrus ~]# vncserver -list -DISPLAY:2

TigerVNC server sessions:

X DISPLAY #     PROCESS ID
:2              1705
:1              1266
```

Рисунок 2. Список активных vnc-серверов

Подключимся на данной машине и на удалённой машине к вновь созданному сеансу, используя на первой машине пароль к полному доступу, а на второй к доступу в качестве зрителя

```
vncviewer 172.16.1.22:2
```

Активируем `pluma` и проследим, что на удалённой машине отобразился открытый текстовый редактор.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Назначение команд ОС Альт, используемых в работе.
2. VNC и Протокол RFB.
3. Протокол TELNET. Виртуальный сетевой терминал NVT.
4. Символы и команды управления NVT.
5. Недостатки и преимущества протокола TELNET.

Практикум № 13
Статическая маршрутизация IPv4

1. ЦЕЛЬ РАБОТЫ

Данная работа предназначена для изучения статической маршрутизации трафика по протоколу IPv4 в сети Интернет, получения практического опыта в работе с маршрутизацией трафика IPv4 в ОС Альт; работа с основными конфигурационными файлами сетевой подсистемы, настройка и управление сетевыми интерфейсами, умение работать с утилитами для диагностики сетевых соединений, работающих по протоколу IPv4.

2. ЗАДАНИЕ НА РАБОТУ

В работе используется топология сети, представленная на рисунке 1. Сеть содержит три подсети, работающих по протоколу IPv4. Маршрутизаторы R2 и R3 являются транзитными маршрутизаторами и осуществляют пересылку пакетов между подсетями 1 и 3, а также выполняют роль шлюзов для маршрутизаторов R1 и R4. Маршрутизаторам R1, R2, R3 и R4 соответствуют рабочие станции под номерами PC1, PC2, PC3 и PC4 соответственно. Адресация в моделируемой сети осуществляется следующим образом:

- Подсеть 1: 10.10.(N_{min}).0/30 или 10.10.1.0/30
- Подсеть 2: 10.10.($N_{min}+1$).0/30 или 10.10.2.0/30
- Подсеть 3: 10.10.($N_{min}+2$).0/30 или 10.10.3.0/30

где $N_{min} = 1$ – наименьший из четырех номер рабочих станций.

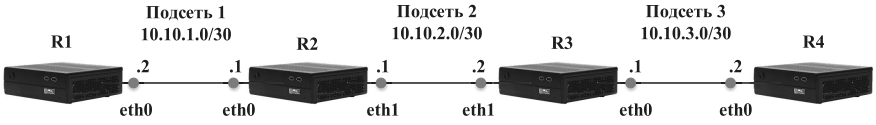


Рисунок 1. Схема сети лабораторной работы

Основные задачи лабораторной работы:

- Настроить все сетевые интерфейсы машин в соответствии со схемой лабораторной сети;

- Сделать маршрутизаторы R1 и R2 шлюзами по умолчанию для маршрутизаторов R1 и R4, прописав на них соответствующие маршруты;

- На транзитных маршрутизаторах R1 и R2 настроить пересылку пакетов между интерфейсами и прописать маршруты в удаленные подсети;

- Проверить работу сети с помощью утилит диагностики сетевых соединений: ping, traceroute.

3. ТЕРМИНЫ И ИСПОЛЬЗУЕМЫЕ КОМАНДЫ

Термины

Статическая маршрутизация – вид маршрутизации трафика в локальных и глобальных сетях, когда маршруты к тем или иным подсетям прописываются вручную администратором сети. Вся маршрутизация при этом происходит без участия каких-либо протоколов маршрутизации.

Статическая маршрутизация имеет свои достоинства и недостатки. К достоинствам можно отнести следующие пункты:

Легкость настройки и отладки;

Отсутствие дополнительной нагрузки на сеть из-за отсутствия трафика управления протоколов маршрутизации;

Низкая нагрузка на аппаратное обеспечение сетевых устройств.

К недостаткам статической маршрутизации относятся:

Сложность масштабирования, трудоемкий процесс конфигурирования при большом количестве маршрутов;

Сложность при изменении топологии сети, требуется вмешательство администратора сети и настройка новых актуальных маршрутов;

Отсутствие динамической балансировки трафика в зависимости от загруженности каналов связи.

Статические маршруты могут быть маршрутами по умолчанию или иметь конкретные маршруты для различных подсетей.

Статический маршрут по умолчанию является маршрутом, который будет соответствовать IP-адресам всех пакетов. Так, имея маршрут по умолчанию, маршрутизатор будет перенаправлять все пакеты с неизвестной ему сетью назначения на заданный в маршруте по умолчанию адрес.

Маршруты по умолчанию часто используется между граничным маршрутизатором компании и маршрутизатором Интернет-провайдера, перенаправляя на него все пакеты, исходящие из локальной сети предприятия.

Также маршруты по умолчанию используются маршрутизаторами, когда в своей таблице маршрутизации они не находят соответствия для между адресом назначения и адресом следующего маршрутизатора, на который необходимо послать пакет для достижения сети назначения. Маршрутизатор проверяет

свою таблицу маршрутизации и вместо того, чтобы отбросить пакет, отправляет его на свой маршрут по умолчанию.

Таблица маршрутизации содержит основную информацию для работы маршрутизатора, а именно:

- Адрес и маску сети назначения;
- Шлюз, обозначающий следующее устройство, через которое доступна сеть назначения;
- Интерфейс, через который доступен шлюз;
- Метрики маршрутов, чем ниже метрика маршрута в определенную сеть, тем приоритетнее данный маршрут.

Также таблицы маршрутизации содержат подключенные напрямую к маршрутизатору сети, интерфейс, к которому они подключены и его IP-адрес.

Используемые команды

`cd` – (Change directory) команда перемещения между директориями;

`ls` – (list) команда просмотра содержимого директории;

`nano` – текстовый редактор для создания и редактирования текстовых файлов;

`ip` – утилита для настройки и управления сетевыми интерфейсами:

`ip address` – отображение информации о сетевых интерфейсах;

`ip route` – отображение информации о маршрутах;

`ip neighbor` – отображение ARP таблицы устройства.

`grep` – утилита для поиска информации;

`ping` – утилита для проверки сетевых соединений;

`traceroute` – утилита для трассировки маршрута в сети;

`mtr` – утилита для трассировки маршрута, показывает информацию в реальном времени.

4. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

4.1. Настройка маршрутизатора R1

Для настройки маршрутизации в сети сначала необходимо настроить все сетевые интерфейсы на четырех маршрутизаторах.

Конфигурационные фалы всех сетевых интерфейсов находятся в директории /etc/net/ifaces. Переход в директорию осуществляется командой `cd`:

```
[user@elbrus]$ cd /etc/net/ifaces/
```

Воспользуемся командой `ls` для того, чтобы просмотреть содержимое директории:

```
[user@elbrus ifaces]$ ls
```

Вывод команды `ls` покажет директории всех сетевых интерфейсов: eth0, eth1, eth2 и петлевого loopback-интерфейса lo.

Так как на R1 необходимо настроить только интерфейс eth0, то переходим в его директорию командой `cd` и выведем её содержимое командой `ls`:

```
[user@elbrus ifaces]$ cd eth0
[user@elbrus ifaces]$ ls
Options
```

В директории находится файл options - основной конфигурационный файл интерфейса eth0.

Для настройки интерфейса eth0 необходимо в конфигурационном файле /eth0/options изменить основные параметры интерфейса. Для редактирования конфигурационного файла нужны права суперпользователя root.

Вход в режим суперпользователя осуществляется командой `su-`, после чего требуется ввести пароль суперпользователя:

```
[user@elbrus eth0]$ su-
```

Отредактировать конфигурационный файл можно открыв его с помощью редактора nano:

```
[root@elbrus]# nano /etc/net/ifaces/eth0/options
```

При стандартной настройке системы все сетевые интерфейсы управляются автоматически через Network Manager, необходимо включить конфигурирование интерфейсов вручную с помощью etcnet. Для этого в открывшемся окне редактора необходимо изменить основные параметры:

`BOOTPROTO=static` – использовать статически заданный IP-адрес;

`NM_CONTROLLED=no` – отключаем автоматическое управление интерфейсом через Network Manager;

`DISABLED=no` – включаем управление интерфейсом через etcnet;

ONBOOT=yes – включать интерфейс при загрузке системы;

CONFIG_IPV4=yes – режим работы с IPv4.

После всех настроек используем Ctrl+O чтобы сохранить изменения и Ctrl+X чтобы выйти из файла конфигурации.

Далее необходимо назначить статический IP-адрес на интерфейс eth0. Для этого в /etc/net/ifaces/eth0/ нужно создать файл с именем ipv4address и в него вписать необходимый IP-адрес в формате <x.x.x.x/маска>

Используя утилиту nano, чтобы создать файл с именем ipv4address и сразу перейти к его редактированию:

```
[root@elbrus eth0]# nano ipv4address
```

10.10.1.2/30 – прописываем IP-адрес в созданном файле ipv4address

Сохраняем Ctrl+O и выходим из редактирования Ctrl+X.

Так как R2 является шлюзом по умолчанию для маршрутизатора R1, то на R1 необходимо прописать маршрут, по умолчанию отправляющий все пакеты на маршрутизатор R2. Для этого в /etc/net/ifaces/eth0 создается файл с именем ipv4route и в него прописывается маршрут по умолчанию:

```
[root@elbrus eth0]# nano ipv4route
```

default via 10.10.1.1 – маршрут по умолчанию на R2

Чтобы изменения применились, необходим перезапуск сетевой службы:

```
[root@elbrus eth0]#service network restart
```

Выход из суперпользователя осуществляется командой exit или Ctrl+D.

Проверка настройки IP-адреса командой ip address или сокращенно ip a:

```
[user@ elbrus]$ ip a
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500
qdisc pfifo_fast state UNKNOWN group default qlen 1000
       link/ether          98:a7:b0:00:de:a0          brd
ff:ff:ff:ff:ff:ff
       inet 10.10.1.2/30 brd 10.10.1.3 scope global eth0
          valid_lft forever preferred_lft forever
```

Первая запись link/ether 00:0c:29:df:4e:8a brd ff:ff:ff:ff:ff:ff – это физические адреса (MAC адреса) интерфейса eth0, где ff:ff:ff:ff:ff:ff – широковещательный MAC адрес,

необходимый для работы протокола преобразования адресов ARP (Address Resolution Protocol).

Вторая запись `inet 10.10.1.2/30 brd 10.10.1.3 scope global eth0` это настроенный нами интерфейс eth0, где `inet 10.10.1.2/30` – сам IP-адрес интерфейса, а `brd 10.10.1.3` – широковещательный адрес для нашей подсети, рассчитанный автоматически в соответствии с указанной маской.

Проверка таблицы маршрутизации на R1. Используется команда `ip route` или сокращенно `ip r`. Так как в работе используются IP-адреса подсетей, начинающиеся с 10.10, то для уточнения поиска по таблице маршрутизации дополним запрос `ip r` командой `grep 10.10`:

```
[user@ elbrus]$ ip r | grep 10.10
default via 10.10.1.1 dev eth0
10.10.1.0/30 dev eth0 proto kernel scope link src 10.10.1.2
```

В таблице маршрутизации есть две записи. Первая запись – наш прописанный маршрут по умолчанию на маршрутизатор R2. Вторая запись – непосредственно подключенная напрямую к интерфейсу eth0 сеть 10.10.1.0/30.

4.2. Настройка маршрутизатора R2

Для настройки сетевых интерфейсов на маршрутизаторе R2 редактируются их конфигурационные файлы в директории /etc/net/ifaces/.

Для интерфейса eth0 в /etc/net/ifaces/eth0 редактируется файл options:

```
[root@elbrus]# nano /etc/net/ifaces/eth0/options
BOOTPROTO=static
NM_CONTROLLED=no
DISABLED=no
ONBOOT=yes
CONFIG_IPV4=yes
```

Настройка IPv4-адреса на eth0:

```
[root@elbrus]# nano /etc/net/ifaces/eth0/ipv4address
10.10.1.1/30
```

Для интерфейса eth1 в /etc/net/ifaces/eth1 редактируется файл options:

```
[root@elbrus]# nano /etc/net/ifaces/eth1/options
BOOTPROTO=static
```

```
NM_CONTROLLED=no
DISABLED=no
ONBOOT=yes
CONFIG_IPV4=yes
```

Интерфейсу eth1 на R2 принадлежит адрес из второй подсети. Настройка IPv4-адреса на eth1:

```
[root@elbrus]# nano /etc/net/ifaces/eth1/ipv4address
10.10.2.10/30
```

Так как R2 является транзитным маршрутизатором, у него должна быть возможность перенаправлять пакеты из одного интерфейса в другой. Для этого необходимо включить форвардинг пакетов на уровне ядра системы.

В файле /etc/net/sysctl.conf нужно отредактировать строку: `net.ipv4.ip_forward=0`, в которой 0 необходимо заменить на 1.

```
[root@elbrus]# nano /etc/net/sysctl.conf
net.ipv4.ip_forward=1
```

После произведенных настроек интерфейсов и форвардинга пакетов, необходимо перезапустить сетевую службу и проверить настроенные IP-адреса:

```
[root@elbrus]# service network restart
[root@elbrus]# ip a
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500
qdisc pfifo_fast state UNKNOWN group default qlen 1000
       link/ether          98:a7:b0:00:dd:60          brd
ff:ff:ff:ff:ff:ff
       inet 10.10.1.1/30 brd 10.10.1.3 scope global eth0
          valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500
qdisc pfifo_fast state UNKNOWN group default qlen 1000
       link/ether          98:a7:b0:00:dd:61          brd
ff:ff:ff:ff:ff:ff
       inet 10.10.2.1/30 brd 10.10.2.3 scope global eth1
          valid_lft forever preferred_lft forever
```

После настроек необходима проверка сетевой связности между подсетями 1 и 2. Для этого с маршрутизатора R1 проверяется доступность подсети 10.10.2.0/30, находящейся за интерфейсом eth1 маршрутизатора R2.

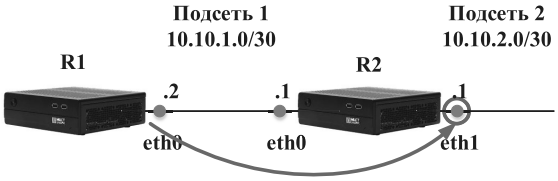


Рисунок 2. Проверка связности сети командой ping

Чтобы проверить работу сети IPv4, используется утилита ping, работающая по протоколу ICMP.

На R1 выполняется команда `ping` и указывается адрес интерфейса eth1 маршрутизатора R2:

```
[root@elbrus ~]# ping 10.10.2.10 -c 3
PING 10.10.2.1 (10.10.2.1) 56(84) bytes of data.
64 bytes from 10.10.2.1: icmp_req=1 ttl=64 time=0.099
ms
64 bytes from 10.10.2.1: icmp_req=2 ttl=64 time=0.095
ms
64 bytes from 10.10.2.1: icmp_req=3 ttl=64 time=0.137
ms
--- 10.10.2.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss,
time 1998ms
rtt min/avg/max/mdev = 0.095/0.110/0.137/0.020 ms
```

Ключ –c (count) задает количество посылаемых ICMP запросов. Интерфейс eth1 маршрутизатора R2 доступен, ICMP пакеты успешно возвращаются.

После того как взаимодействие между R1 и R2 установлено, устройства добавят друг друга в свою таблицу соответствия сетевых и физических адресов – ARP таблицу, установив соответствие между MAC-адресами и IP-адресами интерфейсов маршрутизаторов.

Проверка ARP таблицы на маршрутизаторе R1 осуществляется командой `ip n (neighbor)`. Так как нам необходимы только адреса, начинающиеся с 10.10, то дополним запрос `ip n` командой `grep 10.10`:

```
[root@elbrus ~]# ip n | grep 10.10
10.10.1.1 dev eth0 lladdr 98:a7:b0:00:dd:60 REACHABLE
```

ARP таблица содержит записи, устанавливающие соответствие между MAC-адресами и IP-адресами соседнего узла, т.е. маршрутизатора R2.

В ARP таблице маршрутизатора R2 также будет находиться IP-адрес и соответствующий ему MAC-адрес маршрутизатора R1:

```
[root@elbrus ~]# ip n | grep 10.10
10.10.1.2 dev eth0 lladdr 98:a7:b0:00:de:a0 REACHABLE
```

- Отметка `REACHABLE` показывает, что запись о соответствии появилась недавно, и IP-адрес `10.10.1.2` на R1 доступен.
- Отметка `STALE` означает, что запись о доступности соседнего устройства устарела.
- Отметка `UNREACHABLE` означает недоступность соседнего устройства по этому IP-адресу.

Поскольку R2 является транзитным маршрутизатором между подсетями 1 и 3, у него должен быть маршрут в третью подсеть. Взаимосвязь с подсетью 3 происходит через интерфейс eth1, поэтому маршрут добавляется в конфигурационном файле этого интерфейса:

```
[root@elbrus]# nano /etc/net/ifaces/eth1/ipv4route
10.10.3.0/30 via 10.10.2.2
```

Запись добавляет маршрут в подсеть 3 через интерфейс eth1 маршрутизатора R3.

На R2, после настройки маршрута, необходима перезагрузка сетевой службы и проверка таблицы маршрутизации. Так же дополняем запрос `ip r` командой `grep 10.10`:

```
[root@elbrus]# service network restart
[root@elbrus ~]# ip r | grep 10.10
10.10.1.0/30 dev eth0 proto kernel scope link src
10.10.1.1
10.10.2.0/30 dev eth1 proto kernel scope link src
10.10.2.1
10.10.3.0/30 via 10.10.2.2 dev eth1
```

В таблице маршрутизации первые две записи означают подключенные напрямую сети к интерфейсам eth0 и eth1. Последняя запись – прописанный маршрут в подсеть 3 с интерфейса eth1 через транзитный маршрутизатор R3.

4.3. Настройка маршрутизатора R3

Для настройки сетевых интерфейсов на маршрутизаторе R3 редактируются их конфигурационные файлы в директории /etc/net/ifaces/.

Для интерфейса eth1 в /etc/net/ifaces/eth1 редактируется файл options:

```
[root@elbrus]# nano /etc/net/ifaces/eth1/options
BOOTPROTO=static
NM_CONTROLLED=no
DISABLED=no
ONBOOT=yes
CONFIG_IPV4=yes
```

Настройка IPv4-адреса на eth1:

```
[root@elbrus]# nano /etc/net/ifaces/eth1/ipv4address
10.10.2.2/30
```

Для интерфейса eth0 в /etc/net/ifaces/eth0 редактируется файл options:

```
[root@elbrus]# nano /etc/net/ifaces/eth0/options
BOOTPROTO=static
NM_CONTROLLED=no
DISABLED=no
ONBOOT=yes
CONFIG_IPV4=yes
```

Интерфейсу eth0 на R3 принадлежит адрес из третьей подсети. Настройка IPv4-адреса на eth0:

```
[root@elbrus]# nano /etc/net/ifaces/eth0/ipv4address
10.10.3.1/30
```

Так как R3 является транзитным маршрутизатором, у него должна быть возможность перенаправлять пакеты из одного интерфейса в другой. Для этого необходимо включить форвардинг пакетов на уровне ядра системы.

В файле /etc/net/sysctl.conf нужно отредактировать строку: `net.ipv4.ip_forward=0`, в которой 0 необходимо заменить на 1.

```
[root@elbrus]# nano /etc/net/sysctl.conf
net.ipv4.ip_forward=1
```

Аналогично транзитному маршрутизатору R2, у маршрутизатора R3 должен быть маршрут в первую подсеть. Взаимосвязь с подсетью 1 происходит через интерфейс eth1, поэтому маршрут добавляется в конфигурационном файле этого интерфейса:

```
[root@elbrus]# nano /etc/net/ifaces/eth1/ipv4route
10.10.1.0/30 via 10.10.2.1
```

Запись добавляет маршрут в подсеть 1 через маршрутизатор R2.

После произведенных настроек интерфейсов и форвардинга пакетов, необходим перезапуск сетевой службы и проверка настроенных IP-адресов и маршрутов:

```
[root@elbrus]# service network restart
[root@elbrus]# ip address
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500
qdisc pfifo_fast state UNKNOWN group default qlen 1000
       link/ether          98:a7:b0:00:de:20          brd
ff:ff:ff:ff:ff:ff
       inet 10.10.3.1/30 brd 10.10.3.3 scope global eth0
          valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500
qdisc pfifo_fast state UNKNOWN group default qlen 1000
       link/ether          98:a7:b0:00:de:21          brd
ff:ff:ff:ff:ff:ff
       inet 10.10.2.2/30 brd 10.10.2.3 scope global eth1
          valid_lft forever preferred_lft forever
```

Таблица маршрутизации на R3. В таблице маршрутизации информация о подключенных напрямую подсетях, а также о том, что первая подсеть 10.10.1.0/30 доступна через маршрутизатор R2:

```
[root@elbrus]# ip r
[root@elbrus eth1]# ip r | grep 10.10
10.10.1.0/30 via 10.10.2.1 dev eth1
10.10.2.0/30 dev eth1 proto kernel scope link src
10.10.2.2
10.10.3.0/30 dev eth0 proto kernel scope link src
10.10.3.1
```

После настроек необходима проверка сетевой связности между подсетями 1 и 3 командой `ping`. Для этого с маршрутизатора R1 проверяется доступность подсети 10.10.3.0/30, находящейся за интерфейсом eth0 маршрутизатора R3:

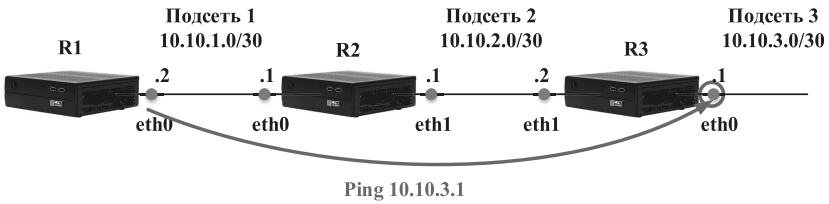


Рисунок 3. Проверка связности сети командой ping

```
[root@elbrus ~]# ping 10.10.3.1 -c 3
PING 10.10.3.1 (10.10.3.1) 56(84) bytes of data.
64 bytes from 10.10.3.1: icmp_req=1 ttl=63 time=0.261
ms
64 bytes from 10.10.3.1: icmp_req=2 ttl=63 time=0.130
ms
64 bytes from 10.10.3.1: icmp_req=3 ttl=63 time=0.133
ms
--- 10.10.3.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss,
time 1998ms
rtt min/avg/max/mdev = 0.130/0.174/0.261/0.062 ms
```

ICMP пакеты успешно возвращаются, интерфейс eth0 маршрутизатора R3 доступен.

4.4. Настройка маршрутизатора R4

Для настройки сетевых интерфейсов на маршрутизаторе R4 редактируются их конфигурационные файлы в директории /etc/net/ifaces/.

Для интерфейса eth0 в /etc/net/ifaces/eth0 редактируется файл options:

```
[root@elbrus]# nano /etc/net/ifaces/eth0/options
BOOTPROTO=static
NM_CONTROLLED=no
DISABLED=no
ONBOOT=yes
CONFIG_IPV4=yes
```

Настройка IPv4-адреса на eth0:

```
[root@elbrus]# nano /etc/net/ifaces/eth0/ipv4address
10.10.3.2/30
```

Так как R3 является шлюзом по умолчанию для маршрутизатора R4, то на R4 необходимо прописать маршрут, по умолчанию отправляющий все пакеты на маршрутизатор R3. Для этого в /etc/net/ifaces/eth0 создается файл с именем ipv4route и в него прописывается маршрут по умолчанию:

```
[root@elbrus eth0]# nano ipv4route
default via 10.10.3.1 -  маршрут по умолчанию на R3
```

После произведенных настроек необходим перезапуск сетевой службы и проверка настроенных IP-адресов и маршрутов:

```
[root@elbrus]# service network restart
[root@elbrus]# ip address
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500
qdisc pfifo_fast state UNKNOWN group default qlen 1000
       link/ether          98:a7:b0:00:df:20          brd
ff:ff:ff:ff:ff:ff
       inet 10.10.3.2/30 brd 10.10.3.3 scope global eth0
          valid_lft forever preferred_lft forever
```

Проверка маршрута по умолчанию на R4 с помощью команды `ip route`:

```
[root@elbrus ~]# ip r | grep 10.10
default via 10.10.3.1 dev eth1
10.10.3.0/30 dev eth1 proto kernel scope link src
10.10.3.2
```

В таблице маршрутизации в дополнении к непосредственно подключенной напрямую сети, появился маршрут по умолчанию на R3, прописанный первой строкой.

4.5. Проверка работы построенной сети

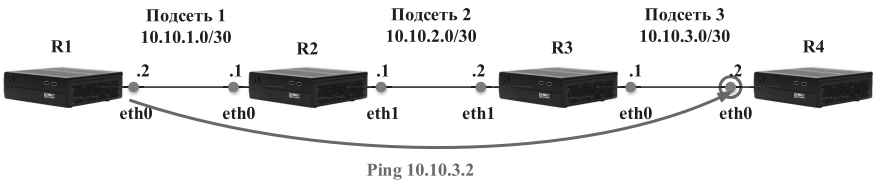


Рисунок 4. Проверка работы сети командой ping

Проверка работы настроенной сети осуществляется командой `ping` с маршрутизатора R1 на маршрутизатор R4:

```
[root@srv ~]# ping 10.10.3.2 -c 5
PING 10.10.3.2 (10.10.3.2) 56(84) bytes of data.
64 bytes from 10.10.3.2: icmp_req=1 ttl=62 time=0.349
ms
64 bytes from 10.10.3.2: icmp_req=2 ttl=62 time=0.219
ms
64 bytes from 10.10.3.2: icmp_req=3 ttl=62 time=0.183
ms
64 bytes from 10.10.3.2: icmp_req=4 ttl=62 time=0.185
ms
64 bytes from 10.10.3.2: icmp_req=5 ttl=62 time=0.194
ms
--- 10.10.3.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss,
time 3999ms
rtt min/avg/max/mdev = 0.183/0.226/0.349/0.062 ms
```

Отправленные ICMP пакеты успешно возвращаются, маршрутизатор R4 доступен, настроенная сеть функционирует правильно.

Проверить каким путем проходят пакеты между подсетями можно с помощью утилиты traceroute:

```
[root@elbrus ~]# traceroute 10.10.3.2
traceroute to 10.10.3.2        , 30 hops max, 60 byte
packets
 1  gateway (10.10.1.1)           0.157 ms  0.115 ms
0.086 ms
 2  10.10.2.2 (10.10.2.2)         0.207 ms  0.133 ms
0.111 ms
 3  10.10.3.2 (10.10.3.2)         0.184 ms  0.178 ms
0.126 ms
```

Маршрутизатор R1, не зная маршрута до R4, отправляет свои пакеты на шлюз по умолчанию – маршрутизатор R2. Транзитные маршрутизаторы R2 и R3 знают о всех подсетях, и используя свои транзитные маршруты,

перенаправляют пакеты между подсетями. Получив пакеты от R1, R2 отправляет их на R3, а R3 на R4. Так как R4 также не знает о подсети 10.10.1.0/30, в которой находится R1, он отправляет пакеты на свой шлюз по умолчанию – маршрутизатор R3, а тот уже через R2 на R1.

4.6. Утилита mtr для диагностики сетевых соединений

Путь прохождения пакетов можно также отследить с помощью утилиты mtr. Утилита mtr является аналогом traceroute и в реальном времени показывает потери и задержки пакетов на определенных участках сети:

```
[root@elbrus]# mtr 10.10.3.2
My   traceroute   [v0.82]   elbrus.localdomain   (0.0.0.0)
Wed May 22 16:51:53 2019
Host                                                Packets
Pings
              loss%  Snt   Last   Avg  Best  Wrst   StDev
1. 10.10.1.1  0.0%    14   0.4    1.1   0.4   1.4    0.3
2. 10.10.2.2  0.0%    13   2.2    2.3   2.2   2.4    0.1
3. 10.10.3.2  0.0%    13   2.2    2.3   2.2   2.4    0.1
```

Loss% – процент потерь пакетов;
Snt – количество отправленных пакетов;
Last – время задержки последнего полученного пакета, мс;
Avg – среднее время задержки, мс;
Best – наименьшее зафиксированное время задержки, мс;
Wrst – наибольшее зафиксированное время задержки, мс;
StDev – среднеквадратичное отклонение времени задержки, мс.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Формат заголовка протокола IP версии 4.
2. Типы IP-адресов, unicast, broadcast и multicast.
3. Для чего используются маски IP-адресов?
4. Назначение широковещательных MAC-адресов. Протокол ARP.
5. Назначение и содержание таблиц маршрутизации.
6. Применение маршрутов по умолчанию.

Практикум № 14
Протокол динамической маршрутизации RIPv2

1. ЦЕЛЬ РАБОТЫ

Изучение особенностей работы протокола маршрутной информации RIPv2 (Routing Information Protocol) версии 2 с помощью отечественных программно-аппаратных средств. Изучение пакета quagga, реализующего протоколы маршрутизации, анализ его функционала в рамках настройки сети, работающей по протоколу динамической маршрутизации RIPv2.

2. ЗАДАНИЕ НА РАБОТУ

В работе используется топология сети, представленная на рисунке 1. Маршрутизаторам R1, R2, R3 и R4 соответствуют рабочие станции под номерами PC1, PC2, PC3 и PC4, соответственно. Адресация в моделируемой сети осуществляется следующим образом:

- Подсеть 1: $10.10.(N_{min}).0/24$ или 10.10.1.0/24
- Подсеть 2: $10.10.(N_{min}+1).0/24$ или 10.10.2.0/24
- Подсеть 3: $10.10.(N_{min}+2).0/24$ или 10.10.3.0/24
- Подсеть 4: $10.10.(N_{min}+3).0/24$ или 10.10.4.0/24

где $N_{min} = 1$ – наименьший из четырех номер рабочих станций.

Последние цифры IP-адресов интерфейсов соответствуют номеру машины.

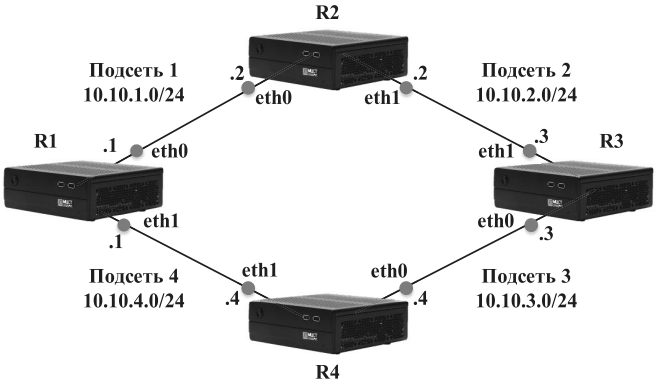


Рисунок 1. Схема лабораторной сети

Основные задачи работы:

- Подготовить машины к работе в сети и установить необходимые пакеты;
- Изменить имя устройств и настроить сетевые интерфейсы, задействованные в работе протокола RIPv2 с помощью демона zebra;
- С помощью демона ripd настроить работу протокола RIPv2 на всех устройствах, настроить анонсирование необходимых подсетей и проверить работу протокола с помощью анализатора трафика Wireshark;
- Проверить отказоустойчивость построенной сети RIPv2, смоделировав отказ канала связи между R1 и R2 путем отключения интерфейса eth0 на R1;
- Проверить изменение метрик маршрутов протокола RIPv2 при перестроении сетевой топологии из-за отключения канала связи между R1 и R2.

3. ТЕРМИНЫ И ИСПОЛЬЗУЕМЫЕ КОМАНДЫ

Термины

Для организации динамической маршрутизации между машинами, работающими под ОС Альт, используется пакет quagga. Пакет представляет собой набор утилит, предназначенных для настройки протоколов динамической маршрутизации в Unix-подобных системах. Quagga поддерживает основные распространенные версии протоколов, такие как: OSPFv2, OSPFv3, RIPv1, RIPv2, RIPng, BGP.

Для обеспечения работы протоколов динамической маршрутизации пакет программ quagga использует специальные сетевые демоны. Демонами в Unix-подобных операционных системах называются компьютерные программы, запускаемые системой при ее старте или самим пользователем из терминала и работающие в фоновом режиме без прямого взаимодействия с ним.

Quagga работает следующим образом: каждый протокол маршрутизации обслуживается собственным демоном маршрутизации (ripd, ospfd и т.д.), который просчитывает маршруты по своему протоколу и передает результаты управляющему демону zebra. Zebra формирует таблицу маршрутизации и выбирает наилучший маршрут на основании административной дистанции и метрик

Протокол динамической маршрутизации RIPv2 относится к категории протоколов маршрутизации внутреннего шлюза (Interior Gateway Protocol –

IGP), которые предназначены для работы внутри автономной системы AS. Автономная система (AS) – это сеть под административным контролем одной организации, сеть каждого интернет провайдера является автономной системой.

Алгоритмы протоколов маршрутизации определяют то, как они решают задачи для изучения всех возможных маршрутов и выбора наилучшего, а также для реакции на изменения в работе сети (конвергенции). Конвергенция (сходимость) – процесс перестроения маршрутов, происходящий при изменении топологии сети, т.е. когда отказывает маршрутизатор или канал связи, либо наоборот – когда они восстанавливаются. Для этих целей протокол RIPv2 использует дистанционно-векторный алгоритм (алгоритм Беллмана-Форда), выбирая наилучший маршрут к подсетям на основании самой низкой метрики маршрута.

Протокол RIPv2 для определения метрики маршрута использует счетчик количества маршрутизаторов (транзитных участков) между текущим маршрутизатором и подсетью назначения. Маршрутизатор каждые 30 секунд отправляют обновления RIP на групповой IP-адрес 224.0.0.9, используя UDP порт 520, содержащие полную таблицу маршрутизации со всеми известными маршрутами соседним маршрутизаторам.

Таблица маршрутизации содержит адрес сети пункта назначения, кратчайший путь до подсети назначения, отсчитываемый в транзитных участках, следующий маршрутизатор и счетчик транзитных участков.

Когда маршрутизатор получает обновление RIP от соседа, он узнает из него о новых сетях, добавляет эти сети в свою таблицу маршрутизации и отправляет обновление другим маршрутизаторам, увеличивая счетчик переходов для каждой полученной им сети на единицу.

Из-за вероятности возникновения маршрутных петель, максимальная метрика маршрута имеет значение 15, т.е. максимальное число транзитных участков не должно превышать 15. При отказе маршрутов, маршрутизаторы анонсируют отказавший маршрут со специальным значением метрики – бесконечностью. Протокол RIP определяет бесконечность как значение метрики 16.

Для управления частотой рассылки обновлений RIP и улучшения времени конвергенции протокол RIPv2 использует специальные таймеры:

- Update timer – таймер отправки маршрутных обновлений RIP (по умолчанию составляет 30 секунд). Каждые 30 секунд процесс RIP на маршрутизаторе рассылает сообщения другим маршрутизаторам RIP, содержащие полную таблицу маршрутизации;

- Timeout timer – таймер тайм-аута маршрута (по умолчанию составляет 180 секунд). По истечению таймера, маршрут, по которому не получены обновления в Update сообщениях помечается как не доступный и помечается метрикой 16, но сохраняется в таблице маршрутизации до тех пор, пока не истекает Flush таймер.
- Flush timer – таймер сброса маршрута (по умолчанию равен 120 секунд). По истечении таймера, недоступный маршрут окончательно удаляется из таблицы маршрутизации.

Используемые команды

`apt-get install` <имя пакета> – команда пакетного менеджера Apt, используемого в ОС Альт. Инициирует установку указанных пакетов;

`chown` – команда смены владельца (change owner), когда также необходимо сменить права вложенные файлы и папки, используется ключ -R (рекурсивно);

`nano` – текстовый редактор, также может создавать текстовые файлы;

`systemctl <start/stop>` – команда управления системными сервисами

`netstat` – команда отображения сетевых соединений системы. Основные ключи: -t (TCP-соединения), -u (UDP-соединения), -l (слушающие сокеты), -p (номер процесса в системе), -n (не переводить IP-адреса в доменные имена);

`grep` – утилита для поиска информации;

`ip` – утилита для настройки и управления сетевыми интерфейсами:

`ip address` – отображение информации о сетевых интерфейсах;

`ip route` – отображение информации о маршрутах;

`telnet` – утилита для подключения к сетевым узлам по протоколу telnet;

`ping` – утилита для проверки сетевых соединений;

`traceroute` – утилита для трассировки маршрута в сети.

Основные команды сетевых демонов quagga

`Show running config` – показать текущую конфигурацию устройства;

`Hostname` <имя> – смена имени устройства;

`Interface` <имя> – переход к конфигурированию интерфейса;

`Ip address` <IP-адрес/маска> – Настройка IP-адреса интерфейса;

`Router rip` – запуск протокола RIP на устройстве и переход к его конфигурированию;

`Version 2` – включение второй версии протокола RIP;

`Timers basic` <Update> <Timeout> <Flush> – настройка таймеров протокола RIPv2;

`Network` <сеть/маска> – настройка анонсирования заданной сети;

`Show ip route rip` – показать таблицу маршрутизации RIPv2;

`Write memory` – сохранение конфигурации устройства.

4. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

4.1. Установка пакетов и подготовка машин к работе

Для установки пакета quagga, как и любых дополнительных пакетов, необходимы права суперпользователя root, поэтому переходим в режим работы от root командой su- и вводим пароль суперпользователя:

```
[user@elbrus ~]$ su-
Password:
```

Используя пакетный менеджер apt, командой `apt-get install` устанавливаем необходимые пакеты. Кроме пакета quagga, для выполнения лабораторной работы понадобятся пакеты telnet и wireshark. Используем ключ –y, чтобы автоматически подтверждать установку пакетов:

```
[root@elbrus ~]# apt-get install -y quagga telnet wireshark
```

Конфигурационные файлы пакета quagga находятся в директории /etc/quagga/, а логи – в /var/log/quagga/.

Чтобы пакет мог корректно работать, необходимо сменить права на файлы конфигурации и логов, которые он использует. Командой `chown` делаем пользователя quagga и его одноименную группу владельцами файлов:

```
[root@elbrus ~]# chown -R quagga:quagga /etc/quagga/
[root@elbrus ~]# chown -R quagga:quagga /var/log/quagga/
```

После того, как пакеты установлены, необходимо на всех машинах включить форвардинг пакетов на уровне ядра системы. Это необходимо для

того, чтобы маршрутизаторы могли пересылать пакеты между своими интерфейсами.

Включение форвадинга производится редактированием конфигурационного файла системы sysctl.conf, находящегося в директории /etc/net/. В конфигурационном файле в строке net.ipv4.ip_forward=0 необходимо изменить 0 на 1:

```
[root@elbrus ~]# nano /etc/net/sysctl.conf
net.ipv4.ip_forward=1
```

Сохраняем изменения Ctrl+O и выходим с помощью Ctrl+X. После настройки нужно перезапустить сетевые сервисы системы командой `service network restart`:

```
[root@elbrus ~]# service network restart
```

4.2. Подключение к демонам и работа с конфигурационными файлами пакета quagga

Для настройки динамической маршрутизации RIPv2 нужны управляющий демон zebra и демон протокола RIPv2 – ripd.

Запускаем необходимые демоны:

```
[root@elbrus ~]# systemctl start zebra
[root@elbrus ~]# systemctl start ripd
```

Настраивать сетевые демоны можно как редактируя их конфигурационные файлы, находящиеся в директории /etc/quagga/, так и подключаясь к их консоли по telnet.

После запуска демонов на нашей машине появятся сокеты (IP-адрес + порт), прослушивающие входящие соединения, с помощью которых и осуществляется подключение к консоли демонов.

Проверим появление необходимых сокетов командой `netstat`, так как демоны quagga используют порты, начинающиеся от 2601, то для точного поиска дополняем запрос командой `grep 260`:

```
[root@elbrus ~]# netstat -tlnp | grep 260
Proto      Local Address   Foreign Address   State   PID/Program
name
tcp         127.0.0.1:2601        0.0.0.0:*       LISTEN        -
735/zebra
tcp         127.0.0.1:2602        0.0.0.0:*       LISTEN        -
1302/ripd
```

На локальном хосте (127.0.0.1) демон ripd слушает порт 2602, а zebra – порт 2601.

Используем telnet для подключения к консоли zebra или ripd, указывается IP-адрес и порт необходимого демона:

`[root@elbrus ~]# telnet 127.0.0.1 2601` – подключение к zebra

`[root@elbrus ~]# telnet 127.0.0.1 2602` – подключение к ripd

Подключимся к zebra. Вводим пароль zebra (по умолчанию) и попадаем в его консоль:

```
[root@elbrus ~]# telnet 127.0.0.1 2601
Connected to 127.0.0.1.
User Access Verification
Password:
Router>
```

Для того чтобы начать настройку, сначала нужно войти в привилегированный режим командой `enable`. Привилегированный режим также защищен паролем (по умолчанию zebra):

```
Router> enable
Password:
Router#
```

Чтобы посмотреть текущую конфигурацию маршрутизатора, используется команда `show running-config`, или сокращенно `show run`:

```
Router# show running config
```

Вывод команды покажет конфигурацию, записанную в файле /etc/quagga/zebra.conf, рассмотрим ее подробнее:

Имя хоста:

```
hostname Router
```

Зашифрованный пароль для подключения к нашему маршрутизатору:

```
password 8 LrjDz/a2KALVQ
```

Зашифрованный пароль для входа в привилегированный режим enable:

```
enable password 8 LrjDz/a2KALVQ
```

Команда, включающая шифрование паролей при просмотре конфигурации:

```
service password-encryption
```

Путь к файлу с логами:

```
log file /var/log/quagga/zebra.log
```

Список контроля доступа (ACL) с именем localhost, разрешающий доступ к консоли демона zebra только с локального хоста, и запрещающий все остальные соединения:

```
access-list localhost permit 127.0.0.1/32
access-list localhost deny any
```

Команды, указывающие применять список доступа localhost для входящих соединений по виртуальным терминальным линиям vty. Подключаясь по telnet, мы занимаем одну виртуальную терминальную линию vty:

```
line vty
access-class localhost
```

Внесение изменений в конфигурацию устройства производится в режиме глобального конфигурирования (config). Вход в режим осуществляется командой `configure terminal` или сокращённо `conf t`:

```
router# configure terminal
router(config)#
```

4.3. Настройка маршрутизатора R1

Для работы протокола RIPv2 на маршрутизаторе R1 с помощью демона zebra настраиваются все сетевые интерфейсы, задействованные в работе протокола RIPv2. Подключаемся к zebra и входим в режим глобального конфигурирования:

```
[root@elbrus ~]# telnet 127.0.0.1 2601
Trying 127.0.0.1...
Connected to 127.0.0.1.
User Access Verification
Password:
Router>
Router> enable
Router#
Router# configure terminal
```

```
Router(config)#
```

Изменим имя устройства:

```
Router(config)# hostname R1
```

В процессе работы протокола RIPv2 будут задействованы два интерфейса маршрутизатора: eth0 и eth1.

Настройка интерфейса eth0:

`R1(config)#interface eth0` – переход к конфигурированию интерфейса;

`R1(config-if)#ip address 10.10.1.1/24` – присваиваем адрес и маску подсети;

`R1(config-if)#exit` – выход из режима настройки интерфейса eth0.

```
R1(config)#
```

Настройка интерфейса eth1:

```
R1(config)#interface eth1
R1(config-if)#ip address 10.10.4.1/24
R1(config-if)#exit
R1(config)#
```

Чтобы сохранить все произведенные настройки, нужно сохранить текущую конфигурацию zebra командой `write memory` или сокращенно `wr`:

```
R1(config)# write memory
Configuration saved to /etc/quagga/zebra.conf
R1(config)# exit
```

Вся конфигурация была перезаписана в /etc/quagga/zebra.conf.

Настройка демона zebra завершена, отключиться от демона можно с помощью комбинации Ctrl+D.

Далее на маршрутизаторе R1 необходимо настроить работу протокола RIPv2 с помощью демона ripd. Подключаемся к ripd по telnet:

```
[root@elbrus ~]# telnet 127.0.0.1 2602
Trying 127.0.0.1...
Connected to 127.0.0.1.
User Access Verification
Password:
```

```
router-RIP>
router-RIP> enable
router-RIP#
```

Командой `show running-config` можно посмотреть текущую конфигурацию, записанную в /etc/quagga/ripd.conf. Для внесения изменений в конфигурацию, переходим в режим глобального конфигурирования, используя команду `configure terminal`:

```
router-RIP# configure terminal
router-RIP(config)#
```

Изменим имя устройства:

```
router-RIP(config)# hostname R1-RIP
```

Теперь необходимо включить процесс RIP на маршрутизаторе:

`R1-RIP(config)# router rip` – включаем RIP на устройстве и переходим к его конфигурированию;

По умолчанию используется устаревшая первая версия протокола RIP, для включения версии 2 используем команду:

```
R1-RIP(config)# version 2
```

Далее необходимо прописать все подключенные сети, которые необходимо анонсировать маршрутизатору R1 по протоколу:

```
R1-RIP(config-router)#       network       10.10.1.0/24
R1-RIP(config-router)# network 10.10.4.0/24
```

Чтобы увеличить скорость конвергенции сети при изменениях топологии, уменьшим время стандартных таймеров. Таймера Update с 30 до 10 секунд, таймера Timeout с 180 до 60 секунд и таймера Flush с 120 до 30 секунд:

```
R1-RIP(config-router)# timers basic 10 60 30
R1-RIP(config-router)# exit
```

Настройка ripd завершена. Сохраняем конфигурацию и отключаемся от ripd с помощью Ctrl+D:

```
R1-RIP(config)# write memory
Configuration saved to /etc/quagga/ripd.conf
R1-RIP(config)# exit
```

4.4. Настройка маршрутизатора R2

Теперь производится настройка интерфейсов маршрутизатора R2. На R2 запускаем демон zebra, подключаемся к нему по telnet и входим в режим глобального конфигурирования:

```
[root@elbrus ~]# systemctl start zebra
[root@elbrus ~]# telnet 127.0.0.1 2601
Trying 127.0.0.1...
Connected to 127.0.0.1.
User Access Verification
Password:
Router>
Router enable
Router#
Router# configure terminal
Router(config)#
```

Изменим имя устройства:

```
Router(config)# hostname R2
```

Настройка интерфейса eth0:

```
R2(config)#interface                                eth0
R2(config-if)#       ip       address       10.10.1.2/24
R2(config-if)# exit
```

Настройка интерфейса eth1:

```
R2(config)#interface                                eth1
R2(config-if)#ip           address           10.10.3.2/24
R2(config-if)#exit
R2(config)#
```

Настройка демона zebra завершена, необходимо сохранить конфигурацию. Отключиться от демона можно с помощью комбинации Ctrl+D.

```
R2(config)# write memory
Configuration saved to /etc/quagga/zebra.conf
R2(config)# exit
```

Далее на маршрутизаторе R2 необходимо настроить работу протокола RIPv2 с помощью демона ripd. Запускаем демон, подключаемся к ripd по telnet и входим в режим глобального конфигурирования:

```
[root@elbrus ~]# systemctl start ripd
[root@elbrus ~]# telnet 127.0.0.1 2602
Trying 127.0.0.1...
Connected to 127.0.0.1.
User Access Verification
Password:
router-RIP>
router-RIP> enable
router-RIP#
router-RIP# configure terminal
router-RIP(config)#
```

Изменим имя устройства:

```
router-RIP(config)# hostname R2-RIP
```

Теперь необходимо включить процесс RIP, указать вторую версию протокола, и прописать сети, которые будет анонсировать наш маршрутизатор:

```
R2-RIP(config)# router rip
R2-RIP(config-router)# version 2
```

Пропишем все нужные для анонсирования сети, подключенные к маршрутизатору:

```
R2-RIP(config-router)#       network       10.10.1.0/24
R2-RIP(config-router)# network 10.10.3.0/24
```

Чтобы увеличить скорость конвергенции сети при изменениях топологии, уменьшим время стандартных таймеров: таймера Update с 30 до 10 секунд, таймера Timeout с 180 до 60 секунд и таймера Flush с 120 до 30 секунд:

```
R2-RIP(config-router)# timers basic 10 60 30
```

Проверим текущую конфигурацию демона ripd командой `show run`, если все настроено верно, то конфигурация должна содержать следующие строки:

```
R2-RIP(config)# show run
!
router rip
```

```
 version 2
 network 10.10.1.0/24
 network 10.10.3.0/24
!
```

Настройка ripd завершена. Сохраняем конфигурацию и отключаемся от ripd с помощью Ctrl+D:

```
R2-RIP(config)# write memory
Configuration saved to /etc/quagga/ripd.conf
R2-RIP(config)# exit
```

4.5. Настройка маршрутизатора R3

Настройка интерфейсов маршрутизатора R3 производится с помощью демона zebra. На R3 запускаем демон zebra, подключаемся к нему по telnet и входим в режим глобального конфигурирования:

```
[root@elbrus ~]# systemctl start zebra
[root@elbrus ~]# telnet 127.0.0.1 2601
Trying 127.0.0.1...
Connected to 127.0.0.1.
User Access Verification
Password:
Router>
Router> enable
Router#
Router# configure terminal
Router(config)#
```

Изменим имя устройства:

```
Router(config)# hostname R3
```

Настройка интерфейса eth0:

```
R3(config)#interface                              eth0
R3(config-if)#ip          address          10.10.3.3/24
R3(config-if)#exit
```

Настройка интерфейса eth1:

```
R3(config)#interface                              eth1
R3(config-if)#ip          address          10.10.2.3/24
R3(config-if)#exit
```

```
R3(config)#
```

Настройка демона zebra завершена, необходимо сохранить конфигурацию. Отключиться от демона можно с помощью комбинации Ctrl+D:

```
R3(config)# write memory
Configuration saved to /etc/quagga/zebra.conf
R3(config)# exit
```

Далее на маршрутизаторе R3 необходимо настроить работу протокола RIPv2 с помощью демона ripd. Запускаем демон, подключаемся к ripd по telnet и входим в режим глобального конфигурирования:

```
[root@elbrus ~]# systemctl start ripd
[root@elbrus ~]# telnet 127.0.0.1 2602
User Access Verification
Password:
router-RIP>
router-RIP> enable
router-RIP#
router-RIP# configure terminal
router-RIP(config)#
```

Изменим имя устройства:

```
router-RIP(config)# hostname R3-RIP
```

Теперь необходимо включить процесс RIP, указать вторую версию протокола, и прописать сети, которые будет анонсировать наш маршрутизатор:

```
R3-RIP(config)# router rip
R3-RIP(config-router)# version 2
```

Пропишем все нужные для анонсирования сети, подключенные к маршрутизатору:

```
R3-RIP(config-router)#       network       10.10.2.0/24
R3-RIP(config-router)# network 10.10.3.0/24
```

Чтобы увеличить скорость конвергенции сети при изменениях топологии, уменьшим время стандартных таймеров: таймера Update с 30 до 10 секунд, таймера Timeout с 180 до 60 секунд и таймера Flush с 120 до 30 секунд:

```
R2-RIP(config-router)# timers basic 10 60 30
```

Настройка демона ripd завершена, необходимо сохранить конфигурацию. Отключиться от демона можно с помощью комбинации Ctrl+D.

```
R3-RIP(config)# write memory
Configuration saved to /etc/quagga/ripd.conf
R3-RIP(config)# exit
```

4.6. Настройка маршрутизатора R4

Настройка интерфейсов маршрутизатора R4 производится с помощью демона zebra. На R4 запускаем демон zebra, подключаемся к нему по telnet и входим в режим глобального конфигурирования:

```
[root@elbrus ~]# systemctl start zebra
[root@elbrus ~]# telnet 127.0.0.1 2601
Trying 127.0.0.1...
Connected to 127.0.0.1.
User Access Verification
Password:
Router>
Router> enable
Router#
Router# configure terminal
Router(config)#
```

Изменим имя устройства:

```
Router(config)# hostname R4
```

Настройка интерфейса eth0:

```
R4(config)#interface                                eth0
R4(config-if)#ip           address           10.10.3.4/24
R4(config-if)#exit
```

Настройка интерфейса eth1:

```
R4(config)#interface                                eth1
R4(config-if)#ip           address           10.10.4.4/24
R4(config-if)#exit
R4(config)#
```

Настройка демона zebra завершена, необходимо сохранить конфигурацию. Отключиться от демона можно с помощью комбинации Ctrl+D:

```
R4(config)# write memory
Configuration saved to /etc/quagga/zebra.conf
R4(config)# exit
```

Далее на маршрутизаторе R4 необходимо настроить работу протокола RIPv2 с помощью демона ripd. Запускаем демон, подключаемся к ripd по telnet и входим в режим глобального конфигурирования:

```
[root@elbrus ~]# systemctl start ripd
[root@elbrus ~]# telnet 127.0.0.1 2602
Trying 127.0.0.1...
Connected to 127.0.0.1.
User Access Verification
Password:
router-RIP>
router-RIP> enable
router-RIP#
router-RIP# configure terminal
router-RIP(config)#
```

Изменим имя устройства:

```
router-RIP(config)# hostname R4-RIP
```

Теперь необходимо включить процесс RIP, указать вторую версию протокола, и прописать сети, которые будет анонсировать наш маршрутизатор:

```
R4-RIP(config)# router rip
R4-RIP(config-router)# version 2
```

Пропишем все нужные для анонсирования сети, подключенные к маршрутизатору:

```
R4-RIP(config-router)#       network       10.10.3.0/24
R4-RIP(config-router)# network 10.10.4.0/24
```

Чтобы увеличить скорость конвергенции сети при изменениях топологии, уменьшим время стандартных таймеров: таймера Update с 30 до 10 секунд, таймера Timeout с 180 до 60 секунд и таймера Flush с 120 до 30 секунд:

```
R4-RIP(config-router)# timers basic 10 60 30
```

Настройка демона ripd завершена, необходимо сохранить конфигурацию. Отключиться от демона можно с помощью комбинации Ctrl+D.

```
R4-RIP(config)# write memory
Configuration saved to /etc/quagga/ripd.conf
R4-RIP(config)# exit
```

4.7. Проверка работы протокола RIPv2

Как только протокол RIPv2 настроен, маршрутизаторы начинают рассылку специальных Update-сообщений, содержащих маршрутную информацию. По умолчанию каждые 30 секунд рассылаются пакеты Response на групповой IP-адрес 224.0.0.9, предназначенный для всех маршрутизаторов, поддерживающих протокол RIPv2. Сообщения RIPv2 response инкапсулируются в протокол UDP и используют порт 520.

Проверим работу протокола RIPv2 на R1 с помощью анализатора трафика, для этого на интерфейсе eth1 запустим Wireshark и зададим фильтр по протоколу RIP:

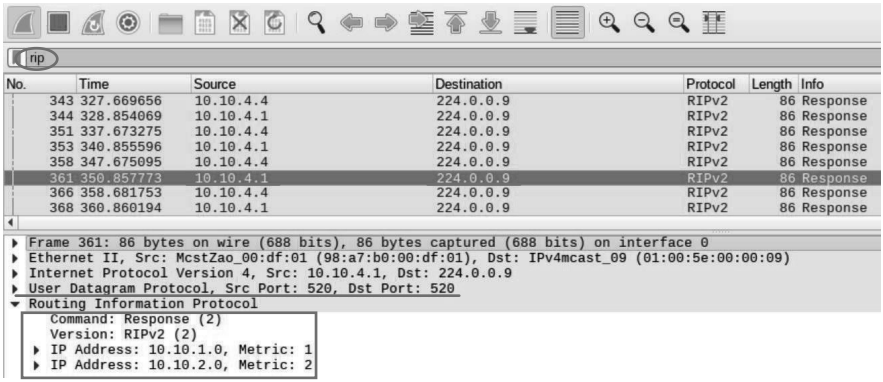


Рисунок 2. Анализ работы протокола RIPv2 на R1

Анализатор трафика показывает, как маршрутизатор R1 с интерфейса eth1, который имеет IP-адрес 10.10.4.1, отправляет пакет RIPv2 response на групповой IP-адрес 224.0.0.9. на порт 520. Пакеты RIPv2 инкапсулированы протоколом UDP, и содержат основою информацию: тип пакета (Command), версию протокола (Version 2, для работы в сетях IPv4) и маршрутную информацию о состоянии известных сетей.

В анализируемом пакете R1 отправляет в сторону R4 маршрутную информацию о известных ему сетях: о сети 10.10.1.0/24 и о сети 10.10.2.0/24, а

также указывает соответствующую метрику. При этом сеть 10.10.1.0/24 подключена напрямую к интерфейсу eth0 и имеет метрику 1, а о сети 10.10.2.0/24 маршрутизатор узнал из анонсов RIP от маршрутизатора R2 и анонсирует ее с метрикой 2, так как до нее от R1 существует два транзитных участка.

4.8. Проверка сети, функционирующей по протоколу RIPv2

Проверим работу протокола RIPv2 проверкой доступности сети 10.10.2.0/24 с маршрутизатора R1. Так как данная сеть не подключена напрямую к маршрутизатору R1, то с помощью протокола RIPv2 он должен получить к ней маршрут через маршрутизатор R2:

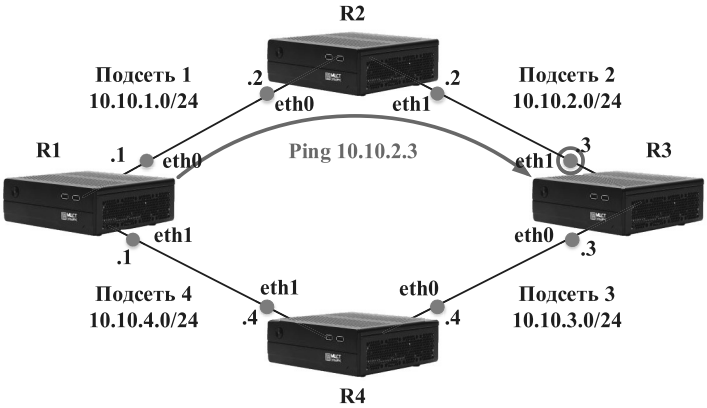


Рисунок 3. Схема моделируемой сети

Чтобы проверить работу сети, используется утилита ping, работающая по протоколу ICMP. На R1 выполняется команда `ping` и указывается адрес интерфейса eth1 маршрутизатора R3:

```
    root@elbrus ~]# ping 10.10.2.3 -c 3
    PING 10.10.2.3 (10.10.2.3) 56(84) bytes of data.
    64 bytes from 10.10.2.3: icmp_req=1 ttl=63 time=0.208
ms
    64 bytes from 10.10.2.3: icmp_req=2 ttl=63 time=0.130
ms
    64 bytes from 10.10.2.3: icmp_req=3 ttl=63 time=0.151
ms
    --- 10.10.2.3 ping statistics ---
```

```
    3 packets transmitted, 3 received, 0% packet loss,
time 1999ms
    rtt min/avg/max/mdev = 0.130/0.163/0.208/0.032 ms
```

Ключ –c (count) задает количество посылаемых ICMP запросов. Интерфейс eth1 маршрутизатора R3 доступен, ICMP пакеты успешно возвращаются, значит R1 получил необходимый маршрут и сеть работает корректно.

Проверим таблицу маршрутизации RIPv2 на R1. Подключаемся по telnet к zebra, и в привилегированном режиме (enable) используем команду `show ip route rip`:

```
    [root@elbrus ~]# telnet 127.0.0.1 2601
    Router>
    Router> enable
    R1# show ip ro rip
    Codes: K - kernel route, C - connected, S - static, R
- RIP, O - OSPF, I - IS-IS, B - BGP, P - PIM, A - Babel,
> - selected route, * - FIB route

    R>*   10.10.2.0/24   [120/2]   via   10.10.1.2,   eth0,
00:08:48
    R>*   10.10.3.0/24   [120/2]   via   10.10.4.4,   eth1,
00:05:31
```

В таблице маршрутизации указаны маршруты, полученные R1 из анонсов RIP от своих соседей R2 и R4. Сеть 10.10.2.0/24 доступна через 10.10.1.2 (маршрутизатор R2), а сеть 10.10.3.0/24 доступна через 10.10.4.4 (маршрутизатор R4).

Убедимся в записи таблицы маршрутизации для подсети 10.10.2.0/24, которая доступна через 10.10.1.2, т.е. R2. Выполним с R1 трассировку маршрута утилитой traceroute до адреса 10.10.2.3, находящегося в подсети 10.10.2.0/24:

```
    [root@elbrus ~]# traceroute 10.10.2.3
    traceroute to 10.10.2.3 (10.10.2.3), 30 hops max, 60
byte packets
     1  10.10.1.2 (10.10.1.2)  0.102 ms  0.058 ms  0.049
ms
     2  10.10.2.3 (10.10.2.3)  0.149 ms  0.112 ms  0.112
ms
```

Как показывает трассировка маршрута, пакеты сначала попадают на адрес 10.10.1.2 маршрутизатора R2, а потом достигают адреса назначения 10.10.2.3 на маршрутизаторе R3.

Проверим как протокол RIPv2 перестраивает маршруты до подсетей при изменении сетевой топологии, путем отключения интерфейса eth0 на маршрутизаторе R1. После отключения порта, протокол RIPv2 перестроит маршрут к подсети 10.10.2.0/24 через маршрутизатор R4.

Как только происходит отключение интерфейса, на маршрутизаторе R2 происходит вытеснение маршрута. Вытеснение маршрута подразумевает анонсирование отказавшего маршрута со специальным значением метрики – бесконечностью. Маршрутизаторы считают маршруты с бесконечной метрикой отказавшими. Протокол RIP определяет бесконечность как значение 16.

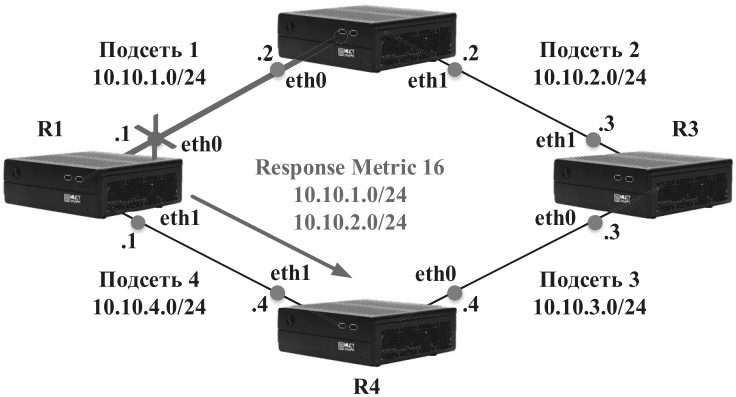


Рисунок 4. Схема сети при отключении интерфейса eth0

Запускаем Wireshark на интерфейсе eth1 маршрутизатора R1 и зададим фильтр по протоколу RIP. После отключаем интерфейс eth0 маршрутизатора R2 и с помощью Wireshark смотрим анонсируемую информацию с интерфейса eth1 маршрутизатора R1:

```
[root@elbrus ~]# ip link set dev eth0 down
```

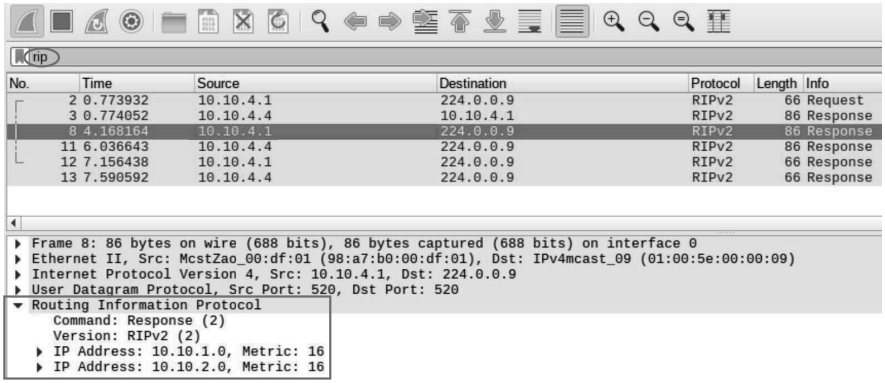


Рисунок 5. Анализ работы протокола RIPv2 при отключении интерфейса

После отключения интерфейса eth0, маршрутизатор R1 с интерфейса eth1, который имеет адрес 10.10.4.1, отправляет пакет с обновлением маршрутной информации на групповой адрес 224.0.0.9, в котором указаны сети 10.10.1.0/24 и 10.10.2.0/24 с метрикой 16, что означает их недоступность. Маршрутизатор R1 анонсирует эти сети, так как до отключения они были доступны через интерфейс eth0. После этого начнется перестроение маршрутов до этих подсетей через маршрутизаторы R4 и R3.

Из-за специфики работы протокола RIPv2, маршрут до подсетей 10.10.1.0/24 и 10.10.2.0/24 не будет перестроен сразу, а только после истечения таймера таймаута, так как отправив обновление маршрутов для недоступных сетей с метрикой 16, маршрутизатор R1 будет хранить эти маршруты пока не истечет таймер таймаута (по умолчанию 180 секунд). Поэтому протокол RIPv2 имеет большое время сходимости, что является одной из причин, по которой он практически не используется в современных сетях. В нашем случае таймер снижен до 60 секунд, по истечении которых маршрут будет перестроен.

После перестроения маршрутов, используем утилиту ping с маршрутизатора R1, проверим доступность адреса 10.10.1.2 из подсети 10.10.1.0/24:

```
    [root@elbrus ~]# ping 10.10.1.2 -c 5
    PING 10.10.1.2 (10.10.1.2) 56(84) bytes of data.
    64 bytes from 10.10.1.2: icmp_req=1 ttl=62 time=0.193
ms
    64 bytes from 10.10.1.2: icmp_req=2 ttl=62 time=0.174
ms
```

```
    64 bytes from 10.10.1.2: icmp_req=3 ttl=62 time=0.183
ms
    64 bytes from 10.10.1.2: icmp_req=4 ttl=62 time=0.180
ms
    64 bytes from 10.10.1.2: icmp_req=5 ttl=62 time=0.170
ms
    -- 10.10.1.2 ping statistics ---
    5 packets transmitted, 5 received, 0% packet loss,
time 3996ms
    rtt min/avg/max/mdev = 0.170/0.180/0.193/0.007 ms
```

Интерфейс eth0 маршрутизатора R2 доступен, ICMP пакеты успешно возвращаются, протокол RIPv2 перестроил маршрут к сети 10.10.1.0/24 через маршрутизаторы R4 и R3.

После перестроения маршрута, используя утилиту traceroute с маршрутизатора R1 проверим путь трафика до адреса 10.10.1.2:

```
    [root@elbrus ~]# traceroute 10.10.1.2
    traceroute to 10.10.1.2 (10.10.1.2), 30 hops max, 60
byte packets
     1  10.10.4.4 (10.10.4.4)  0.133 ms  0.052 ms  0.053
ms
     2  10.10.3.3 (10.10.3.3)  0.170 ms  0.119 ms  0.109
ms
     3  10.10.1.2 (10.10.1.2)  0.204 ms  0.168 ms  0.141
ms
```

Трассировка показывает, что пакеты сначала попадают на узел 10.10.4.4 (маршрутизатор R4), а потом через маршрутизатор R3 (10.10.3.3) попадают в нужную подсеть на заданный IP-адрес 10.10.1.2.

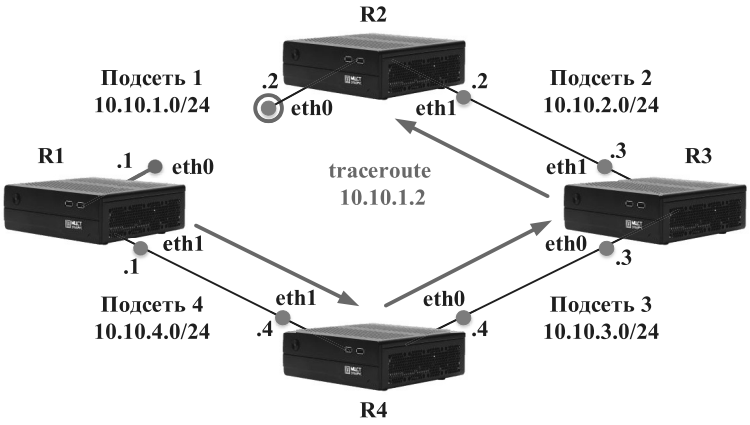


Рисунок 6. Схема прохождения трафика в сети

Когда маршруты перестроены, проверим таблицу маршрутизации RIPv2 на R1. Для этого подключимся к zebra и в привилегированном режиме (enable) используем команду `show ip route rip`:

```
R1# show ip ro rip
Codes: K - kernel route, C - connected, S - static, R
- RIP,
       O - OSPF, I - IS-IS, B - BGP, P - PIM, A -
Babel,
       > - selected route, * - FIB route

R>*   10.10.1.0/24   [120/4]   via   10.10.4.4,   eth1,
00:02:34
R>*   10.10.2.0/24   [120/3]   via   10.10.4.4,   eth1,
00:02:36
R>*   10.10.3.0/24   [120/2]   via   10.10.4.4,   eth1,
00:09:36
```

Теперь в таблице маршрутизации указано, что все существующие сети доступны через 10.10.4.4 (маршрутизатор R4) с интерфейса eth1.

Управление передачей трафика в сети происходит с помощью метрик маршрутов, в таблице маршрутизации метрика маршрута указана вторым числом в квадратных скобках, первое число — это административная дистанция протокола маршрутизации.

Административная дистанция используется маршрутизатором для того, чтобы определить какому протоколу маршрутизации доверять больше, если у обоих протоколов есть маршрут до подсети назначения. Например, протокол OSPF имеет административную дистанцию 110, а устаревший протокол RIPv2 – административную дистанцию 120. Таким образом, если в сети используются два и более протокола маршрутизации, у которых есть свои маршруты к сети назначения, то маршрутизатор выберет маршрут по протоколу с наименьшей административной дистанцией. У статических маршрутов административная дистанция равна единице, так как они прописываются вручную, а значит степень доверия к ним выше.

Если же в сети используется один протокол маршрутизации, и у него есть несколько разных маршрутов до подсети назначения, то маршрутизатор выбирает маршрут уже на основании метрик. Чем меньше метрика, тем больше маршрутизатор доверяет этому маршруту.

У разных протоколов маршрутизации метрика рассчитывается по-разному. Например, у протокола OSPF метрика рассчитывается на основании суммарной цены маршрута, которая в свою очередь зависит от ширины полосы пропускания интерфейсов и каналов связи на маршруте до подсети назначения. А у протокола RIPv2 метрика рассчитывается исходя из числа транзитных участков между маршрутизатором и подсетью назначения.

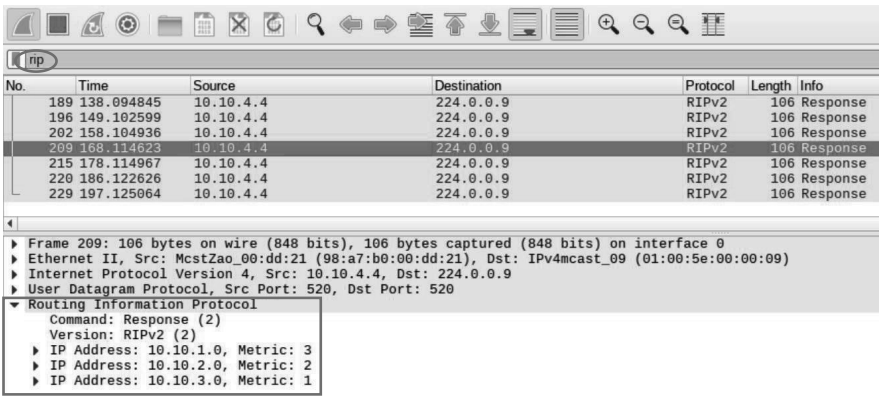


Рисунок 7. Анализ трафика на R1

При этом подсеть 10.10.1.0/24 R4 анонсирует с метрикой 3, так как между интерфейсом eth0 маршрутизатора R4 и подсетью 10.10.1.0/24 есть три транзитных участка: первый между R4 и R3, второй между R3 и R2, а третий начинается за интерфейсом eth0 маршрутизатора R2. Сеть 10.10.2.0/24

анонсирована с метрикой 2, т.к. имеет два транзитных участка, а сеть 10.10.3.0/24 он анонсирует с метрикой 1, так как она подключена напрямую к маршрутизатору R4.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Основные концепции протокола RIPv2. Применение и принцип работы.
2. Каким образом происходит выбор наилучших маршрутов в сети, работающей по протоколу RIPv2?
3. Назначение метрики маршрутов и административной дистанции при работе протоколов динамической маршрутизации.
4. Пакеты обновлений RIP Response, назначение и передаваемая в них информация.
5. Функционирование таймеров протокола RIPv2 при изменениях в топологии сети.

Практикум № 15
Протокол динамической маршрутизации OSPF

1. ЦЕЛЬ РАБОТЫ

Изучение особенностей работы и настройки протокола поиска первого кратчайшего пути OSPF (Open Shortest Path First) с помощью отечественных программно-аппаратных средств. Изучение пакета quagga, реализующего протоколы маршрутизации, анализ его функционала в рамках настройки сети, работающей по протоколу динамической маршрутизации OSPF.

2. ЗАДАНИЕ НА РАБОТУ

В работе используется топология сети, представленная на рисунке 1. Маршрутизаторам R1, R2, R3 и R4 соответствуют рабочие станции под номерами PC1, PC2, PC3 и PC4, соответственно. Адресация в моделируемой сети осуществляется следующим образом:

- Подсеть 1: $10.10.(N_{min}).0/24$ или 10.10.1.0/24
- Подсеть 2: $10.10.(N_{min}+1).0/24$ или 10.10.2.0/24
- Подсеть 3: $10.10.(N_{min}+2).0/24$ или 10.10.3.0/24
- Подсеть 4: $10.10.(N_{min}+3).0/24$ или 10.10.4.0/24

где $N_{min} = 1$ – наименьший из четырех номер рабочих станций.

Последние цифры IP-адресов интерфейсов соответствуют номеру машины.

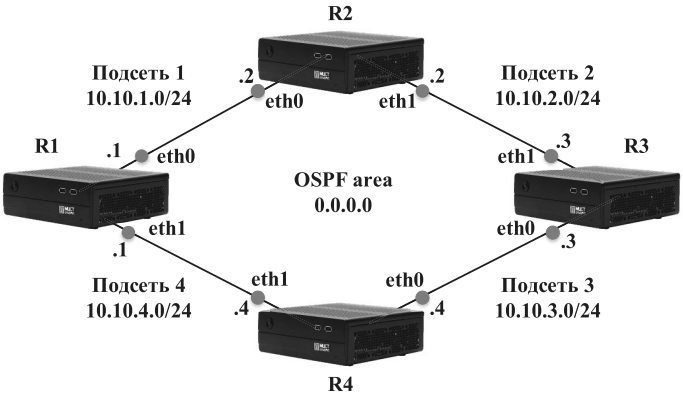


Рисунок 1. Схема лабораторной сети

Основные задачи работы:

- Подготовить машины к работе в сети и установить необходимые пакеты;

- Изменить имя устройств и настроить сетевые интерфейсы, задействованные в работе протокола OSPF с помощью демона zebra;
- С помощью демона ospfd настроить работу протокола OSPF на всех устройствах, настроить анонсирование необходимых подсетей и проверить работу протокола с помощью анализатора трафика Wireshark;
- Проверить отказоустойчивость построенной сети OSPF, смоделировав отказ канала связи между R1 и R2 путем отключения интерфейса eth0 на R1;
- Проверить работу протокола OSPF при изменении полосы пропускания каналов связи между маршрутизаторами R1, R2 и R4.

3. ТЕРМИНЫ И ИСПОЛЬЗУЕМЫЕ КОМАНДЫ

Термины

Для организации динамической маршрутизации между машинами, работающими под ОС Альт, используется пакет quagga. Пакет представляет собой набор утилит, предназначенных для настройки протоколов динамической маршрутизации в Unix-подобных системах. Quagga поддерживает основные распространенные версии протоколов, такие как: OSPFv2, OSPFv3, RIPv1, RIPv2, RIPng, BGP.

Для обеспечения работы протоколов динамической маршрутизации пакет программ quagga использует специальные сетевые демоны. Демонами в Unix-подобных операционных системах называются компьютерные программы, запускаемые системой при ее старте или самим пользователем из терминала и работающие в фоновом режиме без прямого взаимодействия с ним.

Quagga работает следующим образом: каждый протокол маршрутизации обслуживается собственным демоном маршрутизации (ospfd, ripngd и т.д.), который просчитывает маршруты по своему протоколу и передает результаты управляющему демону zebra. Zebra формирует таблицу маршрутизации и выбирает наилучший маршрут на основании административной дистанции и метрик

Протокол динамической маршрутизации OSPF относится к категории протоколов маршрутизации внутреннего шлюза (Interior Gateway Protocol – IGP), которые предназначены для работы внутри автономной системы AS. Автономная система (AS) – это сеть под административным контролем одной организации, сеть каждого интернет провайдера является автономной системой.

Алгоритмы протоколов маршрутизации определяют то, как они решают задачи для изучения всех возможных маршрутов и выбора наилучшего, а также

для реакции на изменения в работе сети (конвергенции). Конвергенция (сходимость) – процесс перестроения маршрутов, происходящий при изменении топологии сети, т.е. когда отказывает маршрутизатор или канал связи, либо наоборот – когда они восстанавливаются. Для этих целей протокол OSPF использует алгоритм состояния канала, выбирая наилучший маршрут к подсетям на основании самой низкой метрики маршрута.

Протокол OSPF для определения метрики подсчитывает цену каждого интерфейса на всем маршруте до сети назначения, а также цену на основании ширины полосы пропускания канала связи и с помощью математического алгоритма Дейкстры выбирает наилучший маршрут.

Протокол OSPF использует концепцию соседей. Соседские отношения позволяют соседним маршрутизаторам обмениваться анонсами состояния каналов LSA (Link State Advertisement), которые содержат изменения в топологии сети. Получая анонсы, маршрутизатор формирует у себя базу данных состояния каналов LSDB (Link State Date Base). Модель соседей OSPF также обеспечивает новым маршрутизаторам динамическое обнаружение, что увеличивает масштабируемость сети.

Для динамического поиска соседей маршрутизаторы OSPF используют пакеты OSPF Hello. Пакеты Hello инкапсулируются заголовком IP и посылаются на групповой IP-адрес 224.0.0.5, предназначенный для всех маршрутизаторов, работающих по протоколу OSPF. Получая пакеты Hello, маршрутизаторы OSPF узнают из них о новых соседях.

Работа сети OSPF организована с помощью зон (OSPF area). Разделение на зоны в протоколе OSPF позволяет снизить нагрузку на маршрутизаторы и оптимизировать работу сети. Если сеть слишком велика и находится целиком в одной области, то маршрутизатор должен будет хранить большую топологическую базу LSDB, занимающую много оперативной памяти.

Протокол OSPF работает с помощью алгоритма Дейкстры, любое изменение состояния интерфейса маршрутизатора запускает этот алгоритм. Обработка больших топологических баз алгоритмом Дейкстры занимает больше процессорного времени, а загрузка процессора растет экспоненциально при увеличении размера топологической базы. При разделении на зоны маршрутизатор будет просчитывать топологию только для своей зоны, также значительно уменьшится количество группового трафика, создаваемого пакетами OSPF Hello, рассылка которых будет ограничена границами зоны.

Используемые команды

`apt-get install` <имя пакета> – команда пакетного менеджера Apt, используемого в ОС Альт. Инициирует установку указанных пакетов;

`chown` – команда смены владельца (change owner), когда также необходимо сменить права вложенные файлы и папки, используется ключ -R (рекурсивно);

`nano` – текстовый редактор, также может создавать текстовые файлы;

`systemctl <start/stop>` – команда управления системными сервисами

`netstat` – команда отображения сетевых соединений системы. Основные ключи: -t (TCP-соединения), -u (UDP-соединения), -l (слушающие сокеты), -p (номер процесса в системе), -n (не переводить IP-адреса в доменные имена);

`grep` – утилита для поиска информации;

`ip` – утилита для настройки и управления сетевыми интерфейсами:

`ip address` – отображение информации о сетевых интерфейсах;

`ip route` – отображение информации о маршрутах;

`telnet` – утилита для подключения к сетевым узлам по протоколу telnet;

`ping` – утилита для проверки сетевых соединений;

`traceroute` – утилита для трассировки маршрута в сети.

Основные команды сетевых демонов quagga

`Show running config` – показать текущую конфигурацию устройства;

`Hostname` <имя> – смена имени устройства;

`Interface` <имя> – переход к конфигурированию интерфейса;

`Ip address` <IP-адрес/маска> – Настройка IP-адреса интерфейса;

`Router rip` – запуск протокола RIP на устройстве и переход к его конфигурированию;

`Version 2` – включение второй версии протокола RIP;

`Router-id` <x.x.x.x> – настройка идентификатора маршрутизатора OSPF;

`Network` <сеть/маска> `area` <№ зоны OSPF> – настройка анонсирования заданной сети и присвоение ей конкретной зоны;

`Show ip route ospf` – показать таблицу маршрутизации OSPF;

`Show ip ospf neighbor` – информация о соседях маршрутизатора OSPF;

`Write memory` – сохранение конфигурации устройства.

4. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

4.1. Установка пакетов и подготовка машин к работе

Для установки пакета quagga, как и любых дополнительных пакетов, необходимы права суперпользователя root, поэтому переходим в режим работы от root командой su- и вводим пароль суперпользователя:

```
[user@elbrus ~]$ su-
Password:
```

Устанавливаем необходимые пакеты. Кроме пакета quagga, для выполнения лабораторной работы понадобятся пакеты telnet и wireshark. Используем ключ –y, чтобы автоматически подтверждать установку пакетов:

```
[root@elbrus ~]# apt-get install –y quagga telnet wireshark
```

Конфигурационные файлы пакета quagga находятся в директории /etc/quagga/, а логи – в /var/log/quagga/.

Чтобы пакет мог корректно работать, необходимо сменить права на файлы конфигурации и логов, которые он использует. Командой `chown` делаем пользователя quagga и его одноименную группу владельцами файлов:

```
[root@elbrus ~]# chown -R quagga:quagga /etc/quagga/
[root@elbrus ~]# chown -R quagga:quagga /var/log/quagga/
```

После того, как пакеты установлены, необходимо на всех машинах включить форвардинг пакетов на уровне ядра системы. Это необходимо для того, чтобы маршрутизаторы могли пересылать пакеты между своими интерфейсами.

Включение форвадинга производится редактированием конфигурационного файла системы sysctl.conf, находящегося в директории /etc/net/. В конфигурационном файле в строке net.ipv4.ip_forward=0 необходимо изменить 0 на 1:

```
[root@elbrus ~]# nano /etc/net/sysctl.conf
net.ipv4.ip_forward=1
```

Сохраняем изменения Ctrl+O и выходим с помощью Ctrl+X. После настройки нужно перезапустить сетевые сервисы системы командой `service network restart`:

```
[root@elbrus ~]# service network restart
```

4.2. Подключение к демонам и работа с конфигурационными файлами пакета quagga

Для настройки динамической маршрутизации OSPF нужны управляющий демон zebra и демон протокола OSPF – ospfd.

Запускаем необходимые демоны:

```
[root@elbrus ~]# systemctl start zebra
[root@elbrus ~]# systemctl start ospfd
```

Настраивать сетевые демоны можно как редактируя их конфигурационные файлы, находящиеся в директории /etc/quagga/, так и подключаясь к их консоли по telnet.

После запуска демонов на нашей машине появятся сокеты (IP-адрес + порт), прослушивающие входящие соединения, с помощью которых и осуществляется подключение к консоли демонов.

Проверим появление необходимых сокетов командой `netstat`, так как демоны quagga используют порты, начинающиеся от 2601, то для точного поиска дополняем запрос командой `grep 260:`

```
[root@elbrus ~]# netstat -tlnp | grep 260
Proto    Local Address   Foreign Address   State   PID/Program name
tcp         127.0.0.1:2601       0.0.0.0:*      LISTEN         -
735/zebra
tcp         127.0.0.1:2604       0.0.0.0:*      LISTEN         -
737/ospfd
```

На локальном хосте (127.0.0.1) демон ospfd слушает порт 2604, а zebra – порт 2601.

Используем telnet для подключения к консоли zebra или ospfd, указывается IP-адрес и порт необходимого демона:

`[root@elbrus ~]# telnet 127.0.0.1 2601` – подключение к zebra

`[root@elbrus ~]# telnet 127.0.0.1 2604` – подключение к ospfd

Подключимся к zebra. Вводим пароль zebra (по умолчанию) и попадаем в его консоль:

```
[root@elbrus ~]# telnet 127.0.0.1 2601
Trying 127.0.0.1...
Connected to 127.0.0.1.
User Access Verification
Password:
Router>
```

Для того чтобы начать настройку, сначала нужно войти в привилегированный режим командой `enable`. Привилегированный режим также защищен паролем (по умолчанию zebra):

```
Router> enable
Password:
Router#
```

Чтобы посмотреть текущую конфигурацию маршрутизатора, используется команда `show running-config`, или сокращенно `show run`:

```
Router# show running config
```

Вывод команды покажет конфигурацию, записанную в файле /etc/quagga/zebra.conf, рассмотрим ее подробнее:

Имя хоста:

```
hostname Router
```

Зашифрованный пароль для подключения к нашему маршрутизатору:

```
password 8 LrjDz/a2KALVQ
```

Зашифрованный пароль для входа в привилегированный режим enable:

```
enable password 8 LrjDz/a2KALVQ
```

Команда, включающая шифрование паролей при просмотре конфигурации:

```
service password-encryption
```

Путь к файлу с логами:

```
log file /var/log/quagga/zebra.log
```

Список контроля доступа (ACL) с именем localhost, разрешающий доступ к консоли демона zebra только с локального хоста, и запрещающий все остальные соединения:

```
    access-list      localhost      permit      127.0.0.1/32
access-list localhost deny any
```

Команды, указывающие применять список доступа localhost для входящих соединений по виртуальным терминальным линиям vty. Подключаясь по telnet, мы занимаем одну виртуальную терминальную линию vty:

```
    line                                                    vty
access-class localhost
```

Внесение изменений в конфигурацию устройства производится в режиме глобального конфигурирования (config). Вход в режим осуществляется командой `configure terminal` или сокращённо `conf t`:

```
    router# configure terminal
    router(config)#
```

4.3. Настройка маршрутизатора R1

Для работы протокола OSPF на маршрутизаторе R1 с помощью демона zebra настраиваются все сетевые интерфейсы, задействованные в работе протокола OSPF. Подключаемся к zebra и входим в режим глобального конфигурирования:

```
    [root@elbrus ~]# telnet 127.0.0.1 2601
    Trying 127.0.0.1...
    Connected to 127.0.0.1.
    User Access Verification
    Password:
    Router>
    Router> enable
    Router#
    Router# configure terminal
    Router(config)#
```

Изменим имя устройства:

```
    Router(config)# hostname R1
```

В процессе работы протокола OSPF будут задействованы два интерфейса маршрутизатора: eth0 и eth1.

Настройка интерфейса eth0:

`R1(config)# interface eth0` – переход к конфигурированию интерфейса;

`R1(config-if)# ip address 10.10.1.1/24` – присваиваем адрес и маску подсети;

`R1(config-if)# bandwidth 100000` – задается скорость работы интерфейса в кбит/с (100 Мбит/с). Команда необходима для работы механизмов QoS (Quality of Service) и не меняет реальную физическую скорость интерфейса, а лишь заставляет маршрутизатор думать, что скорость является такой.

`R1(config-if)# exit` – выход из режима настройки интерфейса eth0.

```
R1(config)#
```

Настройка интерфейса eth1:

```
R1(config)# interface eth1
R1(config-if)# ip address 10.10.4.1/24
R1(config-if)# bandwidth 100000
R1(config-if)# exit
R1(config)#
```

Чтобы сохранить все произведенные настройки, нужно сохранить текущую конфигурацию zebra командой `write memory` или сокращенно `wr`:

```
R1(config)# write memory
Configuration saved to /etc/quagga/zebra.conf
R1(config)# exit
```

Вся конфигурация была перезаписана в /etc/quagga/zebra.conf.

Настройка демона zebra завершена, отключиться от демона можно с помощью комбинации Ctrl+D.

Далее на маршрутизаторе R1 необходимо настроить работу протокола OSPF с помощью демона ospfd. Подключаемся к ospfd по telnet:

```
[root@elbrus ~]# telnet 127.0.0.1 2604
Trying 127.0.0.1...
Connected to 127.0.0.1.
```

```
User Access Verification
Password:
router-OSPF>
router-OSPF> enable
router-OSPF#
```

Командой `show running-config` можно посмотреть текущую конфигурацию, записанную в /etc/quagga/ospfd.conf. Для внесения изменений в конфигурацию, переходим в режим глобального конфигурирования:

```
router-OSPF# configure terminal
router-OSPF(config)#
```

Изменим имя устройства:

```
router-OSPF(config)# hostname R1-OSPF
```

Теперь необходимо включить процесс OSPF и прописать сети, которые будет анонсировать наш маршрутизатор:

- `R1-OSPF(config)# router ospf` – включаем OSPF на устройстве и переходим к его конфигурированию;
- Для правильной работы у маршрутизатора OSPF должен быть задан идентификатор маршрутизатора Router ID (RID).
- Изначально, при запуске процесса OSPF, маршрутизаторы в качестве Router ID используют самый большой IP-адрес физического интерфейса. Оставлять настройку по умолчанию не рекомендуется, так как в случае отключения физического интерфейса нарушится и процесс OSPF, идентификатор которого был привязан к IP-адресу физического интерфейса, это может привести к нарушению работы протокола OSPF на маршрутизаторе.
- Router ID назначается следующим образом в порядке приоритета:
- с помощью команды `router-id`. Если команда задана, то маршрутизатор использует ее значение;
- если на каком-либо loopback-интерфейсе маршрутизатора сконфигурирован IP-адрес и этот интерфейс в состоянии up, то маршрутизатор выбирает loopback-интерфейс с самым большим IP-адресом;
- маршрутизатор сам выбирает самый большой IP-адрес из имеющихся физических интерфейсов.

Таким образом, лучше назначать RID вручную командой `router-id` или использовать логические loopback-интерфейсы, т.к. они не полагаются на аппаратные средства, и переходят в рабочее состояние сразу при запуске устройства.

Зададим Router ID с помощью команды `router-id`:

```
R1-OSPF(config-router)# router-id 1.1.1.1
```

Далее необходимо прописать все сети, которые будет анонсировать маршрутизатор по OSPF. При настройке сетей в OSPF необходимо указывать к какой зоне `area` они принадлежат.

В лабораторной работе будет использоваться одна нулевая зона, называемая магистральной (backbone area). Если в сети существует несколько зон, все они должны быть подсоединены к магистральной зоне.

Пропишем все сети, подключенные к R1, которые необходимо анонсировать, и определим их в нулевую зону:

```
R1-OSPF(config-router)# network 10.10.1.0/24 area 0
R1-OSPF(config-router)# network 10.10.4.0/24 area 0
R1-OSPF(config-router)# exit
```

Настройка демона ospfd завершена. Сохраняем конфигурацию и отключаемся от ospfd с помощью Ctrl+D.

```
R1-OSPF(config)# write memory
Configuration saved to /etc/quagga/ospfd.conf
R1-OSPF(config)# exit
```

4.4. Проверка работы протокола OSPF

Как только маршрутизатор настроен, он начинает поиск соседей, рассылая пакеты «Hello» на групповой IP-адрес 224.0.0.5, предназначенный для всех маршрутизаторов, поддерживающих протокол OSPF. Маршрутизаторы OSPF получают пакеты Hello, посылаемые на этот IP-адрес и узнают из них о других соседях в сети.

Проверим работу протокола OSPF на R1 с помощью анализатора трафика, для этого на интерфейсе eth1 запустим Wireshark и зададим фильтр по протоколу OSPF.

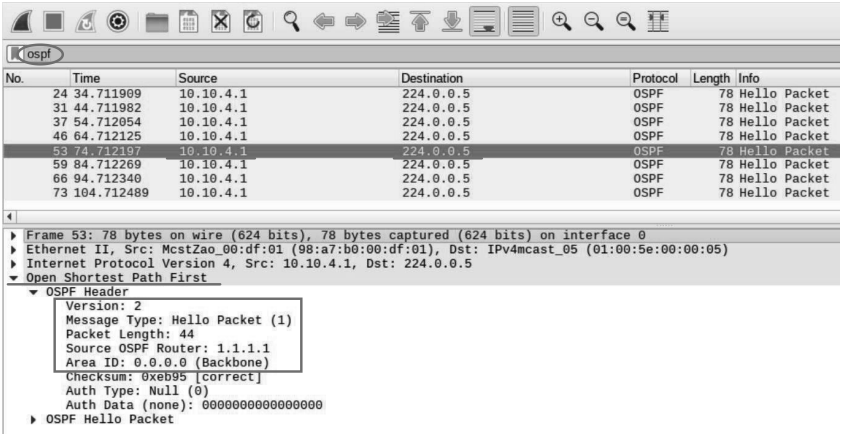


Рисунок 2. Анализ работы протокола OSPF на R1

Анализатор трафика показывает, как маршрутизатор R1 с интерфейса eth1, который имеет IP-адрес 10.10.4.1, отправляет пакеты OSPF Hello на групповой IP-адрес 224.0.0.5. Заголовок пакета OSPF Hello содержит основные параметры маршрутизатора, а именно: версию протокола OSPF (Version 2, версия OSPF для протокола IPv4), тип пакета (Hello Packet), Router ID маршрутизатора (Source OSPF Router) и идентификатор зоны (Area ID).

Hello пакеты используются для обнаружения соседей, а также выступают в роли keepalive-пакетов для проверки доступности, и отправляются по умолчанию каждые 10 секунд. Пакет несет в себе параметры, о которых маршрутизаторы OSPF должны договориться перед тем как становиться соседями.

4.5. Настройка маршрутизатора R2

Теперь производится настройка интерфейсов маршрутизатора R2. На R2 запускаем демон zebra, подключаемся к нему по telnet и входим в режим глобального конфигурирования:

```
[root@elbrus ~]# systemctl start zebra
[root@elbrus ~]# telnet 127.0.0.1 2601
Trying 127.0.0.1...
Connected to 127.0.0.1.
User Access Verification
Password:
Router>
Router enable
```

```
Router#
Router# configure terminal
Router(config)#
```

Изменим имя устройства:

```
Router(config)# hostname R2
```

Настройка интерфейса eth0:

```
R2(config)#interface eth0
R2(config-if)# ip address 10.10.1.2/24
R2(config-if)# bandwidth 100000
R2(config-if)# exit
```

Настроим интерфейс eth1:

```
R2(config)# interface eth1
R2(config-if)# ip address 10.10.2.2/24
R2(config-if)# bandwidth 100000
R2(config-if)# exit
R2(config)#
```

Настройка демона zebra завершена, необходимо сохранить конфигурацию. Отключиться от демона можно с помощью комбинации Ctrl+D.

```
R2(config)# write memory
Configuration saved to /etc/quagga/zebra.conf
R2(config)# exit
```

Далее на маршрутизаторе R2 необходимо настроить работу протокола OSPF с помощью демона ospfd. Запускаем демон, подключаемся к ospfd по telnet и входим в режим глобального конфигурирования:

```
[root@elbrus ~]# systemctl start ospfd
[root@elbrus ~]# telnet 127.0.0.1 2604
Trying 127.0.0.1...
Connected to 127.0.0.1.
User Access Verification
Password:
router-OSPF>
router-OSPF> enable
router-OSPF#
router-OSPF# configure terminal
```

```
router-OSPF(config)#
```

Изменим имя устройства:

```
router-OSPF(config)# hostname R2-OSPF
```

Теперь необходимо включить процесс OSPF, задать идентификатор Router ID и прописать сети, которые будет анонсировать наш маршрутизатор:

```
R2-OSPF(config)# router ospf
R2-OSPF(config-router)# router-id 2.2.2.2
```

Пропишем все нужные для анонсирования сети, подключенные к маршрутизатору и определим их в нулевую зону:

```
R2-OSPF(config-router)# network 10.10.1.0/24 area 0
R2-OSPF(config-router)# network 10.10.2.0/24 area 0
```

Проверим текущую конфигурацию демона ospfd командой `show run`, если все настроено верно, то основная конфигурация должна содержать следующие строки:

```
R2-OSPF(config)# show run
!
router ospf
 ospf router-id 2.2.2.2
 network 10.10.1.0/24 area 0.0.0.0
 network 10.10.2.0/24 area 0.0.0.0
!
```

Настройка демона ospfd завершена, необходимо сохранить конфигурацию. Отключиться от демона можно с помощью комбинации Ctrl+D.

```
R2-OSPF(config)# write memory
Configuration saved to /etc/quagga/ospfd.conf
R2-OSPF(config)# exit
```

4.6. Настройка маршрутизатора R3

Настройка интерфейсов маршрутизатора R3 производится с помощью демона zebra. На R3 запускаем демон zebra, подключаемся к нему по telnet и входим в режим глобального конфигурирования:

```
[root@elbrus ~]# systemctl start zebra
```

```
[root@elbrus ~]# telnet 127.0.0.1 2601
Trying 127.0.0.1...
Connected to 127.0.0.1.
User Access Verification
Password:
Router>
Router> enable
Router#
Router# configure terminal
Router(config)#
```

Изменим имя устройства:

```
Router(config)# hostname R3
```

Настройка интерфейса eth0:

```
R3(config)#interface                                eth0
R3(config-if)# ip address 10.10.3.3/24
R3(config-if)#             bandwidth             100000
R3(config-if)# exit
```

Настроим интерфейс eth1:

```
R3(config)#                interface                eth1
R3(config-if)# ip address 10.10.2.3/24
R3(config-if)#             bandwidth             100000
R3(config-if)# exit
R3(config)#
```

Настройка демона zebra завершена, необходимо сохранить конфигурацию. Отключиться от демона можно с помощью комбинации Ctrl+D:

```
R3(config)# write memory
Configuration saved to /etc/quagga/zebra.conf
R3(config)# exit
```

Далее на маршрутизаторе R3 необходимо настроить работу протокола OSPF с помощью демона ospfd. Запускаем демон, подключаемся к ospfd по telnet и входим в режим глобального конфигурирования:

```
[root@elbrus ~]# systemctl start ospfd
[root@elbrus ~]# telnet 127.0.0.1 2604
Trying 127.0.0.1...
Connected to 127.0.0.1.
```

```
User Access Verification
Password:
router-OSPF>
router-OSPF> enable
router-OSPF#
router-OSPF# configure terminal
router-OSPF(config)#
```

Изменим имя устройства:

```
router-OSPF(config)# hostname R3-OSPF
```

Теперь необходимо включить процесс OSPF, задать идентификатор Router ID и прописать сети, которые будет анонсировать наш маршрутизатор:

```
R3-OSPF(config)# router ospf
R3-OSPF(config-router)# router-id 3.3.3.3
```

Пропишем все нужные для анонсирования сети, подключенные к маршрутизатору и определим их в нулевую зону:

```
R3-OSPF(config-router)# network 10.10.2.0/24 area 0
R3-OSPF(config-router)# network 10.10.3.0/24 area 0
R3-OSPF(config-router)# exit
```

Настройка демона ospfd завершена, необходимо сохранить конфигурацию. Отключиться от демона можно с помощью комбинации Ctrl+D.

```
R3-OSPF(config)# write memory
Configuration saved to /etc/quagga/ospfd.conf
R3-OSPF(config)# exit
```

4.7. Настройка маршрутизатора R4

Настройка интерфейсов маршрутизатора R4 производится с помощью демона zebra. На R4 запускаем демон zebra, подключаемся к нему по telnet и входим в режим глобального конфигурирования:

```
[root@elbrus ~]# systemctl start zebra
[root@elbrus ~]# telnet 127.0.0.1 2601
Connected to 127.0.0.1.
User Access Verification
Password:
Router>
```

```
Router> enable
Router#
Router# configure terminal
Router(config)#
```

Изменим имя устройства:

```
Router(config)# hostname R4
```

Настройка интерфейса eth0:

```
R4(config)#                interface                eth0
R4(config-if)# ip address 10.10.3.4/24
R4(config-if)#              bandwidth             100000
R4(config-if)# exit
```

Настроим интерфейс eth1:

```
R4(config)#                interface                eth1
R4(config-if)# ip address 10.10.4.4/24
R4(config-if)#              bandwidth             100000
R4(config-if)# exit
R4(config)#
```

Настройка демона zebra завершена, необходимо сохранить конфигурацию. Отключиться от демона можно с помощью комбинации Ctrl+D:

```
R4(config)# write memory
Configuration saved to /etc/quagga/zebra.conf
R4(config)# exit
```

Далее на маршрутизаторе R4 необходимо настроить работу протокола OSPF с помощью демона ospfd. Запускаем демон, подключаемся к ospfd по telnet и входим в режим глобального конфигурирования:

```
[root@elbrus ~]# systemctl start ospfd
[root@elbrus ~]# telnet 127.0.0.1 2604
Connected to 127.0.0.1.
User Access Verification
Password:
router-OSPF>
router-OSPF> enable
router-OSPF#
router-OSPF# configure terminal
```

```
router-OSPF(config)#
```

Изменим имя устройства:

```
router-OSPF(config)# hostname R4-OSPF
```

Теперь необходимо включить процесс OSPF, задать идентификатор Router ID и прописать сети, которые будет анонсировать наш маршрутизатор:

```
R4-OSPF(config)# router ospf
R4-OSPF(config-router)# router-id 4.4.4.4
```

Пропишем все нужные для анонсирования сети, подключенные к маршрутизатору и определим их в нулевую зону:

```
R4-OSPF(config-router)# network 10.10.4.0/24 area 0
R4-OSPF(config-router)# network 10.10.3.0/24 area 0
R4-OSPF(config-router)# exit
```

Настройка демона ospfd завершена, необходимо сохранить конфигурацию. Отключиться от демона можно с помощью комбинации Ctrl+D.

```
R4-OSPF(config)# write memory
Configuration saved to /etc/quagga/ospfd.conf
R4-OSPF(config)# exit
```

4.8. Анализ работы сети, работающей по протоколу OSPF

Когда все маршрутизаторы сконфигурированы, между ними должны установиться соседские отношения с помощью пакетов OSPF Hello. Проверим наличие соседских отношений на маршрутизаторе R1. Для этого подключаемся к ospfd и из привилегированного режима (enable) вводим команду `show ip ospf neighbor`:

```
[root@elbrus ~]# telnet 127.0.0.1 2604
R1-OSPF>
R1-OSPF> enable
R1-OSPF# show ip ospf neighbor
Neighbor  ID    Pri     State           Dead  Time       Address
Interface
2.2.2.2           1     Full/Backup      36.459s      10.10.1.2
eth0:10.10.1.1
4.4.4.4           1     Full/Backup      35.128s      10.10.4.4
eth1:10.10.4.1
```

Вывод команды показывает, что R1 установил соседские отношения с R2 и R4. Таблица содержит следующие обозначения:

Neighbor ID: Идентификатор (Router ID) маршрутизатора, с которым установлено соседство;

Pri: Поле приоритета, указывает приоритет соседнего маршрутизатора. По умолчанию приоритеты установлены на 1;

Dead Time: Оставшееся время, в течении которого маршрутизатор будет ждать пакет OSPF Hello от соседа, прежде чем объявить его недоступным (по умолчанию 40 секунд, т.е. 4 интервала Hello);

State: Поле State показывает функциональное состояние соседнего маршрутизатора. Состояние Full означает полную синхронизацию с соседом;

Address: Поле Address указывает IP-адрес интерфейса соседа, по которому с ним осуществляется соединение;

Interface: Поле Интерфейс указывает локальный интерфейс, на котором было установлено OSPF соседство.

Проверим работу протокола OSPF проверкой доступности сети 10.10.2.0/24 с маршрутизатора R1. Так как данная сеть не подключена напрямую к маршрутизатору R1, то с помощью протокола OSPF он должен получить к ней маршрут через маршрутизатор R2:

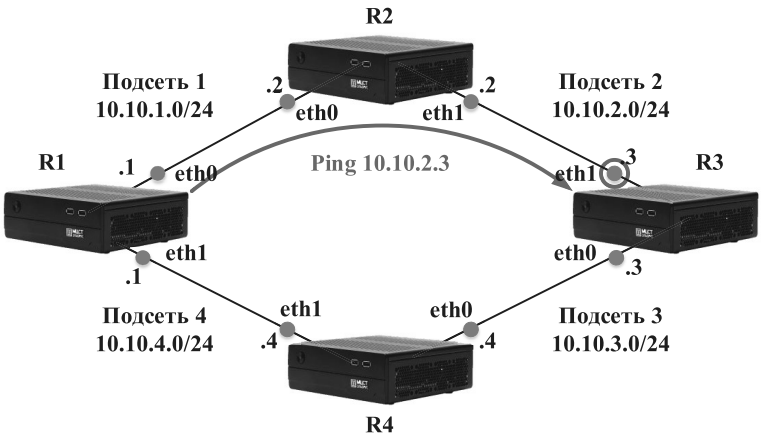


Рисунок 3. Проверка связности настроенной сети

На R1 выполним команду `ping` и укажем адрес интерфейса eth1 на R3:

```
[root@elbrus ~]# ping 10.10.2.3 -c 3
PING 10.10.2.3 (10.10.2.3) 56(84) bytes of data.
```

```
    64 bytes from 10.10.2.3: icmp_req=1 ttl=63 time=0.176
ms
    64 bytes from 10.10.2.3: icmp_req=2 ttl=63 time=0.138
ms
    64 bytes from 10.10.2.3: icmp_req=3 ttl=63 time=0.121
ms
    --- 10.10.2.3 ping statistics ---
    3 packets transmitted, 3 received, 0% packet loss,
time 1998ms
    rtt min/avg/max/mdev = 0.121/0.145/0.176/0.023 ms
```

Ключ –c (count) задает количество посылаемых ICMP запросов. Интерфейс eth1 маршрутизатора R3 доступен, ICMP пакеты успешно возвращаются, значит R1 получил необходимый маршрут и сеть работает верно.

Проверим таблицу маршрутизации OSPF на R1. Подключаемся по telnet к zebra, и в привилегированном режиме (enable) используем команду `show ip route ospf`:

```
    [root@elbrus ~]# telnet 127.0.0.1 2601
    R1>
    R1> enable
    R1# show ip ro ospf
    Codes: K - kernel route, C - connected, S - static, R
- RIP,O - OSPF, I - IS-IS, B - BGP, P - PIM, A - Babel,>
- selected route, * - FIB route

    O   10.10.1.0/24 [110/1] is directly connected, eth0,
00:08:16
    O>* 10.10.2.0/24 [110/2] via 10.10.1.2, eth0,
00:07:26
    O>* 10.10.3.0/24 [110/2] via 10.10.4.4, eth1,
00:06:36
    O   10.10.4.0/24 [110/1] is directly connected, eth1,
00:08:16
```

В таблице маршрутизации указаны маршруты, полученные R1 по OSPF от своих соседей R2 и R4. Сеть 10.10.2.0/24 доступна через 10.10.1.2 (маршрутизатор R2), а сеть 10.10.3.0/24 доступна через 10.10.4.4 (маршрутизатор R4).

Убедимся в записи таблицы маршрутизации для подсети 10.10.2.0/24, которая доступна через 10.10.1.2, т.е. R2. Выполним с R1 трассировку маршрута утилитой traceroute до адреса 10.10.2.3, находящегося в подсети 10.10.2.0/24:

```
    [root@elbrus ~]# traceroute 10.10.2.3
    traceroute to 10.10.2.3 (10.10.2.3), 30 hops max, 60
byte packets
     1   10.10.1.2 (10.10.1.2)   0.128 ms   0.082 ms   0.109
ms
     2   10.10.2.3 (10.10.2.3)   0.124 ms   0.097 ms   0.100
ms
```

Как показывает трассировка маршрута, пакеты сначала попадают на адрес 10.10.1.2 маршрутизатора R2, а потом достигают адреса назначения 10.10.2.3 на маршрутизаторе R3.

Проверим как протокол OSPF перестраивает маршруты до подсетей при изменении сетевой топологии, путем отключения интерфейса eth0 на маршрутизаторе R1. После отключения порта, протокол OSPF перестроит маршрут к подсети 10.10.2.0/24 через маршрутизатор R4.

Как только происходит отключение интерфейса, алгоритм OSPF создает анонсы LSA (Link State Advertisement), в которых содержатся изменения в базе данных LSDB (Link State Data Base) маршрутизатора, а именно – отключенный интерфейс eth0, и рассылает их своим соседям, а они своим соседям, пока у всех маршрутизаторов не появится одинаковая копия базы LSDB, содержащая информацию об измененной топологии сети.

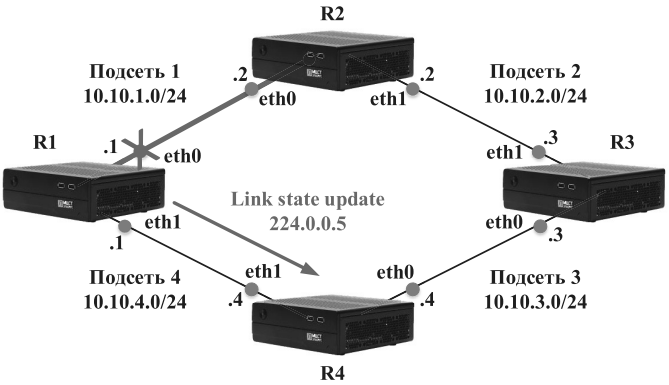


Рисунок 4. Схема сети при отключении интерфейса eth0

Запускаем Wireshark на интерфейсе eth1 маршрутизатора R1 и зададим фильтр по протоколу ospf. После отключаем интерфейс eth0 маршрутизатора R1:

```
[root@elbrus ~]# ip link set dev eth0 down
```

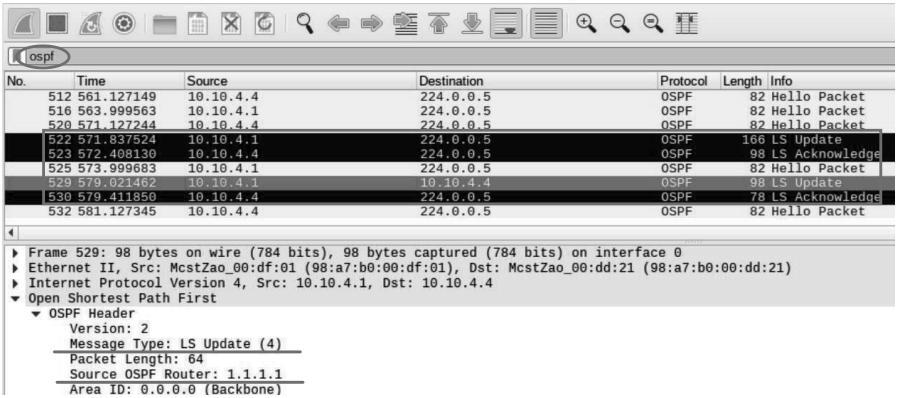


Рисунок 5. Анализ работы протокола OSPF

В частности, сообщения LSA, передающие информацию об изменении, называются обновлениями состояния канала LSU (Link State Update). Как показывает анализатор пакетов, при отключении интерфейса маршрутизатор R1 сразу начал отправку своим соседям на групповой адрес 224.0.0.5 Update сообщений LSU, содержащих изменения базы данных LSDB. Маршрутизаторы R1 и R4 обмениваются пакетами LS Update и подтверждениями LS Acknowledgment, после чего произойдет перестроение маршрутов в сети. Теперь базы LSDB всех маршрутизаторов содержат информацию об отключённом интерфейсе eth0 маршрутизатора R1 и запускают алгоритм Дейкстры для повторного расчета любых маршрутов, на которые повлиял отказ интерфейса.

После того, как интерфейс отключен, проверим доступна ли сеть 10.10.1.0/24 с маршрутизатора R1 при помощи команды `ping`:

```
[root@elbrus ~]# ping 10.10.1.2 -c 5
PING 10.10.1.2 (10.10.1.2) 56(84) bytes of data.
64 bytes from 10.10.1.2: icmp_req=1 ttl=62 time=0.241 ms
64 bytes from 10.10.1.2: icmp_req=2 ttl=62 time=0.187 ms
64 bytes from 10.10.1.2: icmp_req=3 ttl=62 time=0.185 ms
64 bytes from 10.10.1.2: icmp_req=4 ttl=62 time=0.184 ms
64 bytes from 10.10.1.2: icmp_req=5 ttl=62 time=0.202 ms
```

```
--- 10.10.1.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time
4000ms
rtt min/avg/max/mdev = 0.184/0.199/0.241/0.028 ms
```

Интерфейс eth0 маршрутизатора R2 доступен, ICMP пакеты успешно возвращаются и значит, что протокол OSPF перестроил маршрут к подсети 10.10.1.0/24 через маршрутизатор R4 и R3.

Чтобы проверить, каким путем прошли пакеты до подсети 10.10.1.0/24, с R1 выполним трассировку маршрута до адреса 10.10.1.2 на R2, который находится в этой подсети. На R1 используем команду `traceroute`:

```
[root@elbrus ~]# traceroute 10.10.1.2
traceroute to 10.10.1.2 (10.10.1.2), 30 hops max, 60
byte packets
 1  10.10.4.4 (10.10.4.4)  0.132 ms  0.056 ms  0.054
ms
 2  10.10.3.3 (10.10.3.3)  0.161 ms  0.113 ms  0.102
ms
 3  10.10.1.2 (10.10.1.2)  0.176 ms  0.152 ms  0.143
ms
```

Трассировка показывает, что пакеты сначала попадают на узел 10.10.4.4 (маршрутизатор R4), а потом через маршрутизатор R3 (10.10.3.3) попадают в нужную подсеть на заданный IP-адрес 10.10.1.2.

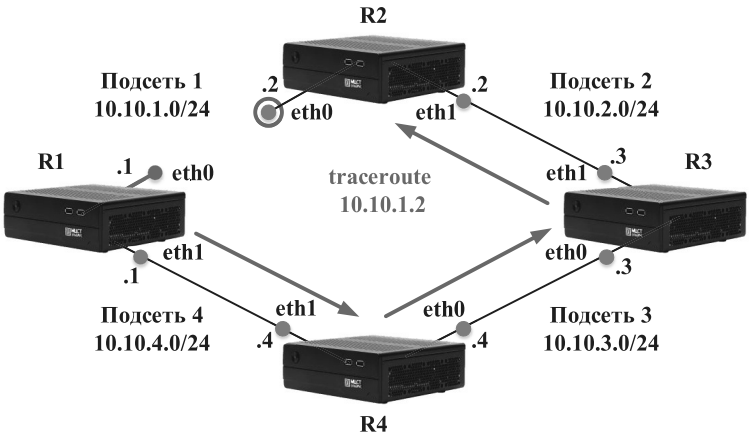


Рисунок 6. Схема прохождения трафика в сети

Таблица маршрутизации на R1 также покажет произошедшие изменения в сетевой топологии. Для этого подключимся к zebra и в привилегированном режиме (enable) используем команду `show ip route ospf`:

```
R1# show ip ro ospf
Codes: K - kernel route, C - connected, S - static, R -
RIP,O - OSPF, I - IS-IS, B - BGP, P - PIM, A - Babel,> -
selected route, * - FIB route
O>*   10.10.1.0/24   [110/4]   via   10.10.4.4,   eth1,
00:02:37
O>*   10.10.2.0/24   [110/3]   via   10.10.4.4,   eth1,
00:03:09
O>*   10.10.3.0/24   [110/2]   via   10.10.4.4,   eth1,
00:13:27
O   10.10.4.0/24 [110/1] is directly connected, eth1,
00:15:07
```

Теперь в таблице маршрутизации указано, что все существующие сети доступны через 10.10.4.4 (маршрутизатор R4) с интерфейса eth1.

Протокол OSPF также может выбирать наилучший маршрут до подсетей исходя их полосы пропускания канала на пути прохождения трафика. Если снизить скорость линии между маршрутизаторами, то протокол OSPF также сначала начнет рассылку своим соседям Update сообщений, содержащих изменения в топологии сети, а после перестроит маршрут до сети назначения по более скоростным каналам.

На маршрутизаторе R1 вернем в активное состояние интерфейс eth0:

```
[root@elbrus ~]# ip link set dev eth0 up
```

Проверим таблицу маршрутизации OSPF на R1. Подключаемся по telnet к zebra, и в привилегированном режиме (enable) используем команду `show ip route ospf`. Подсеть 10.10.1.0/24 появилась в статусе подключенной напрямую (directly connected), а подсеть 10.10.2.0/24 снова доступна через R2:

```
R1# show ip ro ospf
Codes: K - kernel route, C - connected, S - static, R -
RIP, O - OSPF, I - IS-IS, B - BGP, P - PIM, A - Babel, > -
selected route, * - FIB route

O     10.10.1.0/24 [110/1] is directly connected, eth0,
00:00:15
O>* 10.10.2.0/24 [110/2] via 10.10.1.2, eth0, 00:00:15
```

```
    O>* 10.10.3.0/24 [110/2] via 10.10.4.4, eth1, 00:00:15
    O     10.10.4.0/24  [110/1]  is  directly  connected,  eth1,
00:15:13
```

Изменим пропускную способность канала связи между R1 и R2. На интерфейсе eth0 маршрутизатора R1 программно снизим скорость интерфейса со 100 Мбит/с до 5 Мбит/с, в zebra в режиме глобального конфигурирования используем команду `bandwidth`, полоса указывается в кбит/с:

```
[root@elbrus ~]# telnet 127.0.0.1 2601
R1>
R1> enable
R1# configure terminal
R1(config)# interface eth0
R1(config-if)# bandwidth 5000
R1(config-if)# exit
```

Физически скорость интерфейса не изменилась, но маршрутизатор теперь считает скорость интерфейса равной 5 Мбит/с. После этого протокол OSPF снова перестроит маршрут ко всем подсетям через R4. Несмотря на три транзитных участка по новому маршруту через R4 и R3, линия между R1 и R2 все равно будет иметь меньший приоритет из-за низкой скорости передачи, хотя и имеет всего один транзитный участок.

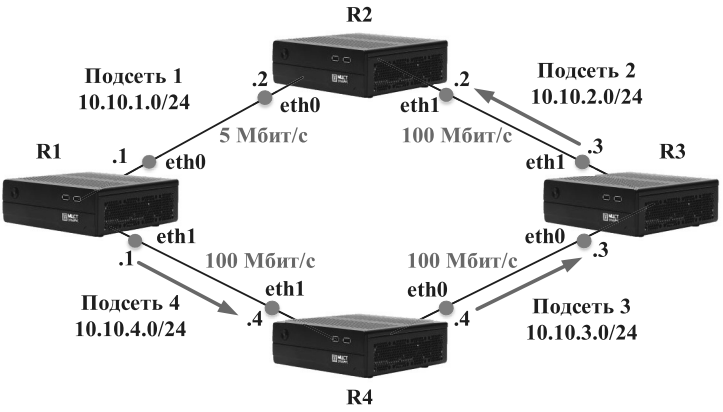


Рисунок 7. Изменение пропускной способности канала между R1 и R2

Таблица маршрутизации на R1 показывает изменения, произошедшие в сети. Теперь маршрут до всех подсетей идет снова идет через 10.10.4.4, т.е. через маршрутизатор R4:

```
R1(config-if)# do sh ip ro ospf
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, P - PIM, A - Babel, N - NHRP, > - selected route, * - FIB route

O>* 10.10.1.0/24 [110/4] via 10.10.4.4, eth1, 00:00:07
O>* 10.10.2.0/24 [110/3] via 10.10.4.4, eth1, 00:00:07
O>* 10.10.3.0/24 [110/2] via 10.10.4.4, eth1, 00:00:07
O   10.10.4.0/24 [110/1] is directly connected, eth1, 00:17:07
```

Управление передачей трафика в сети происходит с помощью метрик маршрутов, в таблице маршрутизации метрика маршрута указана вторым числом в квадратных скобках, первое число — это административная дистанция протокола маршрутизации.

Административная дистанция используется маршрутизатором для того, чтобы определить какому протоколу маршрутизации доверять больше, если у обоих протоколов есть маршрут до подсети назначения. Например, протокол OSPF имеет административную дистанцию 110, а устаревший протокол RIPv2 – административную дистанцию 120. Таким образом, если в сети используются два и более протокола маршрутизации, у которых есть свои маршруты к сети назначения, то маршрутизатор выберет маршрут по протоколу с наименьшей административной дистанцией. У статических маршрутов административная дистанция равна единице, так как они прописываются вручную, а значит степень доверия к ним выше.

Если же в сети используется один протокол маршрутизации, и у него есть несколько разных маршрутов до подсети назначения, то маршрутизатор выбирает маршрут уже на основании метрик. Чем меньше метрика, тем больше маршрутизатор доверяет этому маршруту.

У разных протоколов маршрутизации метрика рассчитывается по-разному. Например, у протокола RIPv2 метрика рассчитывается исходя из числа транзитных участков между маршрутизатором и подсетью назначения. А у протокола OSPF метрика рассчитывается на основании суммарной цены маршрута, которая в свою очередь зависит от ширины полосы пропускания интерфейсов и каналов связи на маршруте до подсети назначения.

Чтобы проверить, как меняется метрика маршрута в сети, на маршрутизаторе R1 программно изменим цену интерфейса eth1 путем уменьшения его скорости со 100 Мбит/с до 10 Мбит/с и проанализируем таблицу маршрутизации:

```
R1(config)#
R1(config)# interface eth1
R1(config-if)# bandwidth 10000
R1(config-if)# exit
```

После изменений трафик все равно будет проходить через маршрутизатор R3, так как суммарная цена маршрута через R3 ниже из-за более высокой пропускной способности каналов, проходящих через него.

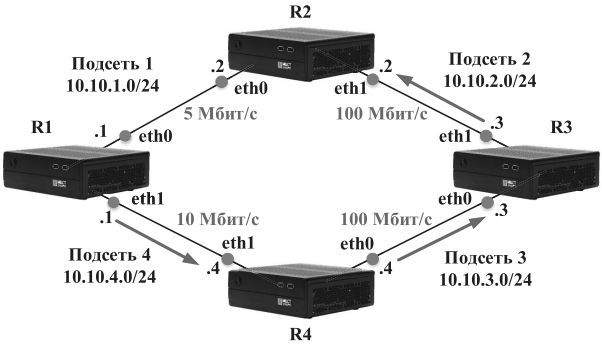


Рисунок 8. Изменение пропускной способности канала между R1 и R4

Уменьшая скорость интерфейса eth1 маршрутизатора R1, мы увеличиваем цену маршрута на участке сети между R1 и R4, что приводит к увеличению метрик на 9 единиц:

```
R1(config-if)# do show ip ro ospf
Codes: K - kernel route, C - connected, S - static, R
- RIP, O - OSPF, I - IS-IS, B - BGP, P - PIM, A - Babel,>
- selected route, * - FIB route

O       10.10.1.0/24  [110/13]  via  10.10.4.4,  eth1,
00:02:37
O>*  10.10.2.0/24  [110/12]  via  10.10.4.4,  eth1,
00:02:37
```

```
   O>*  10.10.3.0/24  [110/11]  via  10.10.4.4,  eth1,
00:02:37
   O 10.10.4.0/24 [110/10] is directly connected, eth1,
00:02:37
```

После снижения скорости интерфейса eth1, метрики всех маршрутов увеличились, но трафик по-прежнему идет через R4:

```
   [root@elbrus ~]# traceroute 10.10.1.2
   Traceroute to 10.10.1.2 (10.10.1.2), 30 hops max, 60
byte packets
    1  10.10.4.4 (10.10.4.4)  0.132 ms  0.056 ms  0.054
ms
    2  10.10.3.3 (10.10.3.3)  0.161 ms  0.113 ms  0.102
ms
    3  10.10.1.2 (10.10.1.2)  0.176 ms  0.152 ms  0.143
ms
```

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Основные концепции протокола OSPF. Применение и принцип работы;
2. Для каких целей применяется деление на области при построении сети по протоколу OSPF?
3. Каким образом происходит выбор наилучших маршрутов в сети OSPF? Назначение метрики маршрутов и административной дистанции.
4. Назначение пакетов OSPF Hello. Основные параметры, передаваемые в заголовке пакета OSPF Hello;
5. Концепция соседей протокола OSPF;
6. Назначение анонсов состояния канала LSA и баз данных состояния каналов LSDB протокола OSPF.

Практикум № 16
Протокол динамической маршрутизации BGP

1. ЦЕЛЬ РАБОТЫ

Изучение особенностей работы и настройки протокола граничного шлюза BGP (Border Gateway Protocol) с помощью отечественных программно-аппаратных средств. Изучение пакета quagga, реализующего протоколы маршрутизации, анализ его функционала в рамках настройки сети, работающей по протоколу динамической маршрутизации BGP.

2. ЗАДАНИЕ НА РАБОТУ

В работе используется топология сети, представленная на рисунке 1. Маршрутизаторам R1, R2, R3 и R4 соответствуют рабочие станции под номерами PC1, PC2, PC3 и PC4 соответственно. Адресация в моделируемой сети осуществляется следующим образом:

Подсеть 1: $10.10.(N_{min}).0/30$ или 10.10.1.0/30

Подсеть 2: $10.10.(N_{min}+1).0/30$ или 10.10.2.0/30

Подсеть 3: $10.10.(N_{min}+2).0/30$ или 10.10.3.0/30

Подсеть 4: $10.10.(N_{min}+3).0/30$ или 10.10.4.0/30

Номера автономных систем AS распределяются следующим образом:

AS R1: $6500(N_{min})$ или 65001 AS R2: $6500(N_{min}+1)$ или 65002

AS R3: $6500(N_{min}+2)$ или 65003 AS R4: $6500(N_{min}+3)$ или 65004

Сети в автономных системах реализуются с помощью loopback интерфейсов маршрутизаторов и имеют следующую адресацию:

Сеть в AS R1 : $192.168.(N_{min}).1$ или 192.168.1.1

Сеть в AS R2 : $192.168.(N_{min}+1).1$ или 192.168.2.1

Сеть в AS R4 : $192.168.(N_{min}+3).1$ или 192.168.4.1

Сети в AS R3 : 192.168.10.1/30 и 192.168.20.1/30

где $N_{min}=1$ – наименьший из четырех номер рабочих станций.

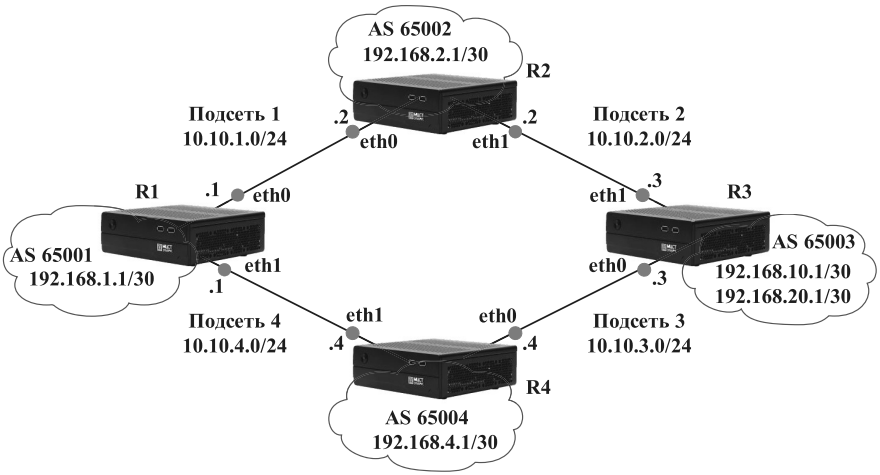


Рисунок 1. Схема моделируемой сети

Основные задачи работы:

- Подготовить машины к работе в сети и установить необходимые пакеты;
- Изменить имя устройств и настроить сетевые интерфейсы, задействованные в работе протокола BGP с помощью демона zebra;
- С помощью демона bgpd настроить работу протокола BGP на всех устройствах, настроить BGP-соседство и анонсирование необходимых префиксов, а также проверить работу протокола с помощью анализатора трафика Wireshark;
- На маршрутизаторе R1 с помощью списков префиксов запретить принимать BGP анонсы сети 192.168.10.0/30 от маршрутизатора R2;
- На маршрутизаторе R1 с помощью карт маршрутов настроить распределение трафика таким образом, чтобы трафик до сети 192.168.10.0/30 передавался через AS 65002, а до 192.168.20.0/30 через AS 65004.

3. ТЕРМИНЫ И ИСПОЛЬЗУЕМЫЕ КОМАНДЫ

Термины

Для организации динамической маршрутизации между машинами, работающими под ОС Альт, используется пакет quagga. Пакет представляет собой набор утилит, предназначенных для настройки протоколов динамической

маршрутизации в Unix-подобных системах. Quagga поддерживает основные распространенные версии протоколов, такие как: OSPFv2, OSPFv3, RIPv1, RIPv2, RIPng, BGP.

Для обеспечения работы протоколов динамической маршрутизации пакет программ quagga использует специальные сетевые демоны. Демонами в Unix-подобных операционных системах называются компьютерные программы, запускаемые системой при ее старте или самим пользователем из терминала и работающие в фоновом режиме без прямого взаимодействия с ним.

Quagga работает следующим образом: каждый протокол маршрутизации обслуживается собственным демоном маршрутизации (bgpd, ripngd и т.д.), который просчитывает маршруты по своему протоколу и передает результаты управляющему демону zebra. Zebra формирует таблицу маршрутизации и выбирает наилучший маршрут на основании административной дистанции и метрик

Протокол динамической маршрутизации BGP относится к категории протоколов маршрутизации внешнего шлюза (Exterior Gateway Protocol – EGP), которые предназначены для работы между автономными системами AS, связывая их в единую мировую сеть Интернет. Одно из главных отличий протокола BGP от протоколов IGP в том, что он анонсирует маршруты к другим маршрутизаторам в других компаниях. В протоколе BGP анонсируемые блоки адресов называются префиксами. Автономные системы играют важную роль

Автономная система (AS) – это сеть под административным контролем одной организации, сеть каждого интернет провайдера является автономной системой. Автономные системы играют ключевую роль в протоколе BGP. Протокол BGP использует номера автономных систем для многих целей, включая часть процесса выбора наилучшего маршрута, а также они используются механизмом предотвращения петлевого маршрута.

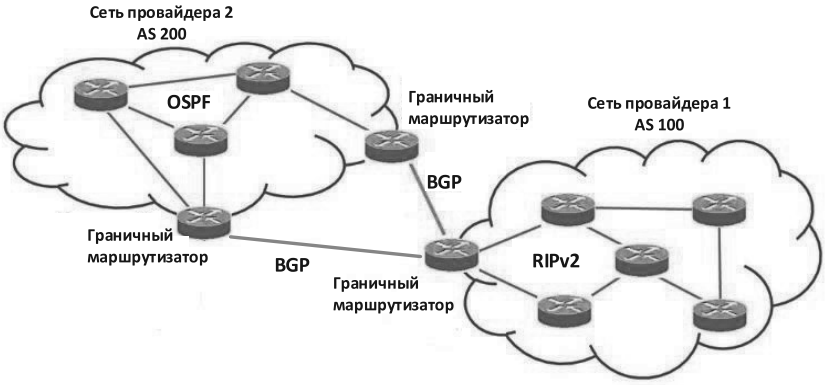


Рисунок 2. Схема работы протокола BGP

Для выбора наилучшего маршрута протокол BGP не использует единую концепцию метрик как в протоколах IGP. Вместо этого он использует атрибуты пути. Протокол BGP анонсирует каждый префикс наряду со списком различных атрибутов пути. Атрибуты пути – это различные факты о маршруте для достижения определенной подсети. Одним из таких атрибутов протокола BGP является атрибут пути AS Path. Атрибут AS Path – это передаваемый с маршрутами BGP атрибут пути, содержащий список номеров AS на маршруте. Процесс выбора наилучшего маршрута считает лучшим более короткий атрибут AS Path.

Протокол BGP использует концепцию соседских отношений, но не использует динамическое обнаружение и установление BGP сессии с соседом. Маршрутизаторы должны быть сконфигурированы вручную.

Для передачи своих сообщений между двумя партнерами BGP протокол BGP использует протокол TCP и порт 179. Когда настраивается протокол BGP, он открывает порт 179 и ожидает входящие запросы на установление соединения от других маршрутизаторов.

После установления соединения и проверки всех необходимых параметров, маршрутизаторы BGP становятся соседями и начинают обмен маршрутной информацией в сообщениях об обновлении, которые содержат информацию о префиксах и соответствующие атрибуты пути.

Используемые команды

`apt-get install` <имя пакета> – команда пакетного менеджера Apt, используемого в ОС Альт. Инициирует установку указанных пакетов;

chown – команда смены владельца (change owner), когда также необходимо сменить права вложенные файлы и папки, используется ключ -R (рекурсивно);

nano – текстовый редактор, также может создавать текстовые файлы;

systemctl <start/stop> – команда управления системными сервисами

netstat – команда отображения сетевых соединений системы. Основные ключи: -t (TCP-соединения), -u (UDP-соединения), -l (слушающие сокеты), -p (номер процесса в системе), -n (не переводить IP-адреса в доменные имена);

grep – утилита для поиска информации;

telnet – утилита для подключения к сетевым узлам по протоколу telnet;

ping – утилита для проверки сетевых соединений;

traceroute – утилита для трассировки маршрута в сети.

Основные команды сетевых демонов quagga

Show running config – показать текущую конфигурацию устройства;

Hostname <имя> – смена имени устройства;

Interface <имя> – переход к конфигурированию интерфейса;

Ip address <IP-адрес/маска> – Настройка IP-адреса интерфейса;

Router bgp <№ AS> – запуск протокола BGP на устройстве и переход к его конфигурированию;

Neighbor <IP-адрес> remote-as <№ AS> – настройка соседа BGP;

Network <IP-адрес> mask <m.m.m.m> - настройка анонса сети;

Show ip bgp neighbors – информация о соседе BGP;

Show ip bgp – показать таблицу маршрутизации BGP;

Show ip bgp summary – информация о соседях маршрутизатора BGP;

Write memory – сохранение конфигурации устройства;

Ip prefix-list <имя списка> [seq <value>] <deny | permit> <network/length> [ge <value>] [le <value>] – настройка списка префиксов;

Route-map <имя карты> <deny | permit> <seq> – настройка карты маршрутов.

4. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

4.1. Установка пакетов и подготовка машин к работе

Для установки пакета quagga, как и любых дополнительных пакетов, необходимы права суперпользователя root, поэтому переходим в режим работы от root командой su- и вводим пароль суперпользователя:

```
[user@elbrus ~]$ su-
Password:
```

Используя пакетный менеджер apt, командой `apt-get install` устанавливаем необходимые пакеты. Кроме пакета quagga, для выполнения лабораторной работы понадобятся пакеты telnet и wireshark. Используем ключ –y, чтобы автоматически подтверждать установку пакетов:

```
[root@elbrus ~]# apt-get install -y quagga telnet wireshark
```

Конфигурационные файлы пакета quagga находятся в директории /etc/quagga/, а логи – в /var/log/quagga/.

Чтобы пакет мог корректно работать, необходимо сменить права на файлы конфигурации и логов, которые он использует. Командой `chown` делаем пользователя quagga и его одноименную группу владельцами файлов:

```
[root@elbrus ~]# chown -R quagga:quagga /etc/quagga/
[root@elbrus ~]# chown -R quagga:quagga /var/log/quagga/
```

После того, как пакеты установлены, необходимо на всех машинах включить форвардинг пакетов на уровне ядра системы. Это необходимо, чтобы маршрутизаторы могли пересылать пакеты между своими интерфейсами.

Включение форвадинга производится редактированием конфигурационного файла системы sysctl.conf, находящегося в директории /etc/net/. В конфигурационном файле в строке net.ipv4.ip_forward=0 необходимо изменить 0 на 1:

```
[root@elbrus ~]# nano /etc/net/sysctl.conf
net.ipv4.ip_forward=1
```

Сохраняем изменения Ctrl+O и выходим с помощью Ctrl+X. После настройки нужно перезапустить сетевые сервисы системы командой `service network restart`:

```
[root@elbrus ~]# service network restart
```

4.2. Подключение к демонам и работа с конфигурационными файлами пакета quagga

Для настройки динамической маршрутизации OSPF нужны управляющий демон zebra и демон протокола BGP – bgpd.

Запускаем необходимые демоны:

```
[root@elbrus ~]# systemctl start zebra
[root@elbrus ~]# systemctl start bgpd
```

Настраивать сетевые демоны можно как редактируя их конфигурационные файлы, находящиеся в директории /etc/quagga/, так и подключаясь к их консоли по telnet.

После запуска демонов на нашей машине появятся сокеты (IP-адрес + порт), прослушивающие входящие соединения, с помощью которых и осуществляется подключение к консоли демонов.

Проверим появление необходимых сокетов командой `netstat`:

```
[root@elbrus ~]# netstat -tlnp
Proto      Local Address   Foreign Address   State   PID/Program name
tcp         127.0.0.1:2601        0.0.0.0:*       LISTEN        -
735/zebra
tcp         127.0.0.1:2605        0.0.0.0:*       LISTEN        -
1217/bgpd
```

На локальном хосте (127.0.0.1) демон bgpd слушает порт 2605, а zebra – порт 2601.

Используем telnet для подключения к консоли zebra или bgpd, указывается IP-адрес и порт необходимого демона:

`[root@elbrus ~]# telnet 127.0.0.1 2601` – подключение к zebra

`[root@elbrus ~]# telnet 127.0.0.1 2605` – подключение к bgpd

Подключимся к zebra. Вводим пароль zebra (по умолчанию) и попадаем в его консоль:

```
[root@elbrus ~]# telnet 127.0.0.1 2601
Trying 127.0.0.1...
Connected to 127.0.0.1.
User Access Verification
Password:
Router>
```

Для того чтобы начать настройку, сначала нужно войти в привилегированный режим командой `enable`. Привилегированный режим также защищен паролем (по умолчанию zebra):

```
Router> enable
Password:
Router#
```

Чтобы посмотреть текущую конфигурацию маршрутизатора, используется команда `show running-config`, или сокращенно `show run`:

```
Router# show running config
```

Вывод команды покажет конфигурацию, записанную в файле /etc/quagga/zebra.conf, рассмотрим ее подробнее:

Имя хоста:

```
hostname Router
```

Зашифрованный пароль для подключения к нашему маршрутизатору:

```
password 8 LrjDz/a2KALVQ
```

Зашифрованный пароль для входа в привилегированный режим enable:

```
enable password 8 LrjDz/a2KALVQ
```

Команда, включающая шифрование паролей при просмотре конфигурации:

```
service password-encryption
```

Путь к файлу с логами:

```
log file /var/log/quagga/zebra.log
```

Список контроля доступа (ACL) с именем localhost, разрешающий доступ к консоли демона zebra только с локального хоста, и запрещающий все остальные соединения:

```
access-list localhost permit 127.0.0.1/32
access-list localhost deny any
```

Команды, указывающие применять список доступа localhost для входящих соединений по виртуальным терминальным линиям vty. Подключаясь по telnet, мы занимаем одну виртуальную терминальную линию vty:

```
line vty
access-class localhost
```

Внесение изменений в конфигурацию устройства производится в режиме глобального конфигурирования (config). Вход в режим осуществляется командой `configure terminal` или сокращённо `conf t`:

```
router# configure terminal
router(config)#
```

4.3. Настройка маршрутизатора R1

Для работы протокола BGP на маршрутизаторе R1 с помощью демона zebra настраиваются все сетевые интерфейсы, задействованные в работе протокола BGP. Подключаемся к zebra и входим в режим глобального конфигурирования:

```
[root@elbrus ~]# telnet 127.0.0.1 2601
Trying 127.0.0.1...
Connected to 127.0.0.1.
User Access Verification
Password:
Router>
Router> enable
Router#
Router# configure terminal
Router(config)#
```

Изменим имя устройства:

```
Router(config)# hostname R1
```

В процессе работы протокола BGP будет задействован интерфейс маршрутизатора eth0.

Настройка интерфейса eth0:

`R1(config)#interface eth0` – переход к конфигурированию интерфейса;

`R1(config-if)#ip address 10.10.1.1/24` – присваиваем адрес и маску подсети;

`R1(config-if)#exit` – выход из режима настройки интерфейса eth0.

```
R1(config)#
```

Настройка интерфейса eth1:

```
R1(config)#interface eth1
R1(config-if)# ip address 192.168.4.1/24
R1(config-if)# exit
```

Настройка логического loopback-интерфейса lo:

```
R1(config)#interface lo
R1(config-if)# ip address 192.168.1.1/30
R1(config-if)# exit
```

Выход из режима глобального конфигурирования осуществляется с помощью команды `exit` или комбинации Ctrl+C.

Чтобы сохранить все произведенные настройки, нужно сохранить текущую конфигурацию zebra командой `write memory` или сокращенно `wr`:

```
R1(config)# write memory
Configuration saved to /etc/quagga/zebra.conf
R1(config)# exit
```

Вся конфигурация была перезаписана в /etc/quagga/zebra.conf.

Настройка демона zebra завершена, отключиться от демона можно с помощью комбинации Ctrl+D.

Далее на маршрутизаторе R1 необходимо настроить работу протокола BGP с помощью демона bgpd. Подключаемся к bgpd по telnet:

```
[root@elbrus ~]# telnet 127.0.0.1 2605
Trying 127.0.0.1...
Connected to 127.0.0.1.
User Access Verification
```

```
Password:
router-BGP>
router-BGP> enable
router-BGP#
```

Командой `show running-config` можно посмотреть текущую конфигурацию, записанную в /etc/quagga/bgpd.conf. Для внесения изменений в конфигурацию, переходим в режим глобального конфигурирования:

```
router-BGP# configure terminal
router-BGP(config)#
```

Изменим имя устройства:

```
router-BGP(config)# hostname R1-BGP
```

Теперь необходимо включить процесс BGP на маршрутизаторе и назначить номер автономной системы в нашей сети, а также прописать сети, которые будет анонсировать данный маршрутизатор.

Включаем BGP и задаем номер автономной системы AS 65001, переходим к его конфигурированию:

```
R1-BGP(config)# router bgp 65001
R1-BGP(config-router)#
```

Для работы протоколу BGP сначала необходимо установить соседские отношения с подключёнными непосредственно к нему маршрутизаторами. Соседи прописываются вручную, указывается IP-адрес и номер AS соседа. В нашей сети соседом маршрутизатора R1 является маршрутизатор R2 и маршрутизатор R4, соответственно, указываются их IP-адреса и номер их автономных систем AS:

```
R1-BGP(config-router)# neighbor 10.10.1.2 remote-as 65002
R1-BGP(config-router)# neighbor 10.10.4.4 remote-as 65004
```

Далее в формате `network <x.x.x.x> mask <m.m.m.m>` прописываются сети, которые необходимо анонсировать по протоколу BGP, в нашем случае за сеть в AS отвечает loopback-адрес маршрутизатора:

```
R1-BGP(config-router)# network 192.168.1.0 mask 255.255.255.252
```

```
R1-BGP(config-router)# exit
```

Настройка демона bgpd завершена, необходимо сохранить конфигурацию. Отключиться от демона можно комбинации Ctrl+D.

```
R1-BGP(config)# write memory
Configuration saved to /etc/quagga/bgpd.conf
R1-BGP(config)# exit
```

4.4. Настройка маршрутизатора R2

Теперь настраивается маршрутизатор R2. Для работы протокола BGP на маршрутизаторе R2 с помощью демона zebra настраиваются все сетевые интерфейсы, задействованные в работе протокола BGP. Подключаемся к zebra и входим в режим глобального конфигурирования:

```
[root@elbrus ~]# systemctl start zebra
[root@elbrus ~]# telnet 127.0.0.1 2601
Trying 127.0.0.1...
Connected to 127.0.0.1.
User Access Verification
Password:
Router>
Router enable
Router#
Router# configure terminal
Router(config)#
```

Изменим имя устройства:

```
Router(config)# hostname R2
```

Настройка интерфейса eth0:

```
R2(config)#interface                                eth0
R2(config-if)#       ip       address       10.10.1.2/24
R2(config-if)# exit
```

Настроим интерфейс eth1:

```
R2(config)#              interface              eth1
R2(config-if)#       ip       address       10.10.2.2/24
R2(config-if)# exit
```

Настройка логического loopback-интерфейса lo:

```
R2(config)#interface lo
R2(config-if)# ip address 192.168.2.1/30
R2(config-if)# exit
```

Настройка демона zebra завершена, необходимо сохранить конфигурацию. Отключиться от демона можно комбинации Ctrl+D.

```
R2(config)# write memory
Configuration saved to /etc/quagga/zebra.conf
R2(config)# exit
```

Далее на маршрутизаторе R2 необходимо настроить работу протокола BGP с помощью демона bgpd. Запускаем демон, подключаемся к bgpd по telnet и входим в режим глобального конфигурирования:

```
[root@elbrus ~]# systemctl start bgpd
[root@elbrus ~]# telnet 127.0.0.1 2605
Trying 127.0.0.1...
Connected to 127.0.0.1.
User Access Verification
Password:
router-BGP>
router-BGP> enable
router-BGP#
```

Для внесения изменений в конфигурацию, переходим в режим глобального конфигурирования:

```
router-BGP# configure terminal
router-BGP(config)#
```

Изменим имя устройства:

```
router-BGP(config)# hostname R2-BGP
```

Теперь необходимо включить процесс BGP на маршрутизаторе и назначить номер автономной системы в нашей сети, а также прописать сети, которые будет анонсировать данный маршрутизатор.

Включаем BGP и задаем номер автономной системы AS 65002, переходим к его конфигурированию;

```
R2-BGP(config)# router bgp 65002
R2-BGP(config-router)#
```

Для работы протоколу BGP сначала необходимо установить соседские отношения с подключёнными непосредственно к нему маршрутизаторами.

В нашей сети соседями маршрутизатора R2 являются маршрутизаторы R1 и R3, соответственно, указываются их IP-адреса и номера AS:

```
R2-BGP(config-router)# neighbor 10.10.1.1 remote-as 65001
R2-BGP(config-router)# neighbor 10.10.2.3 remote-as 65003
```

Далее в формате `network <x.x.x.x> mask <m.m.m.m>` прописываются сети, которые необходимо анонсировать по протоколу BGP, в нашем случае за сеть в AS отвечает loopback адрес маршрутизатора:

```
R2-BGP(config-router)# network 192.168.2.0 mask 255.255.255.252
```

Так как R2 находится в транзитной автономной системе между R1 и R3, необходимо прописать транзитные сети между ними:

```
R2-BGP(config-router)#network 10.10.1.0 mask 255.255.255.0
R2-BGP(config-router)#network 10.10.2.0 mask 255.255.255.0
```

Настройка демона bgpd завершена, необходимо сохранить конфигурацию. Отключиться от демона можно комбинации Ctrl+D.

```
R2-BGP(config)# write memory
Configuration saved to /etc/quagga/bgpd.conf
R2-BGP(config)# exit
```

4.5. Анализ работы протокола BGP

После настроек маршрутизаторы R1 и R2 начнут процесс установления соседских отношений, инициируя TCP-сессию по 179 порту. Проверим работу протокола BGP на R1 с помощью анализатора трафика, для этого на интерфейсе eth0 запустим Wireshark и зададим фильтр по протоколу TCP:

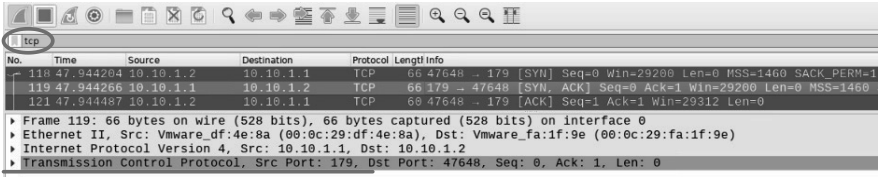


Рисунок 3. Процесс установления TCP-сессии

R1 отправляет SYN на 179 порт R2, R2 подтверждает установление сессии, отвечая R1 SYN+ACK, R1 отвечает подтверждением ACK. Таким образом происходит тройное рукопожатие, после чего TCP-сессия установлена.

После того, как TCP-сессия установлена, маршрутизаторы BGP начинают обмен сообщениями OPEN. OPEN – первый тип сообщений BGP, они отсылаются только в самом начале BGP-сессии для согласования параметров. В нём передаются версия протокола, номер AS, Hold Timer и Router ID. Чтобы BGP-сессия установилась, должны соблюдаться следующие условия:

Версии протокола должна быть одинаковой;

Номера AS в сообщении OPEN должны совпадать с настройками на удалённой стороне;

Router ID должны различаться.

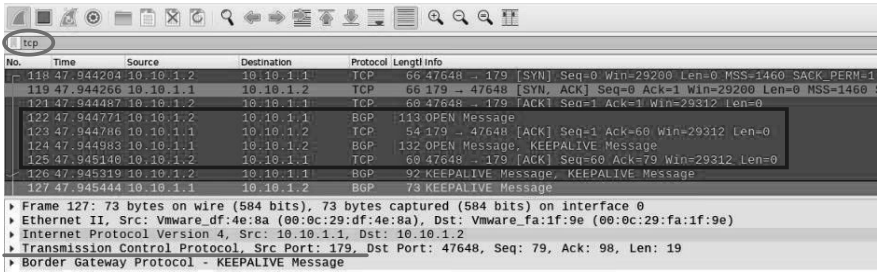


Рисунок 4. Процесс обмена сообщениями OPEN

Получив OPEN от R1, R2 отправляет свой OPEN, а также KEEPALIVE, говорящий о том, что OPEN от R1 получен – это сигнал для R1 переходить к следующему состоянию – ESTABLISHED.

После всех шагов маршрутизаторы переходят в стабильное состояние ESTABLISHED, означающее, что запущена правильная версия BGP и все настройки идентичны.

Для каждого соседа можно посмотреть Uptime – как долго он находится в состоянии ESTABLISHED, для этого подключимся к bgpd и в режиме enable и используя команду `show ip bgp neighbors`:

```
[root@elbrus ~]# telnet 127.0.0.1 2605
R1-BGP>
R1-BGP> enable
R1-BGP#
R1-BGP# show ip bgp neighbors
BGP neighbor is 10.10.1.2, remote AS 65002, local AS 65001, external link
    BGP version 4, remote router ID 192.168.2.1
    BGP state = Established, up for 00:18:53
    Last read 00:00:53, hold time is 180, keepalive interval is 60 seconds
```

Команда выводит IP-адрес соседнего маршрутизатора, номер его AS и номер AS на локальном маршрутизаторе. Также показывается версия протокола BGP (4 версия для IPv4), router ID соседа, Uptime и таймеры рассылки сообщений keepalive и таймер удержания сессии Hold, по истечении которого сосед будет считаться недоступным.

После того как все состояния успешно пройдены, BGP-маршрутизаторы регулярно будут рассылать сообщения KEEPALIVE. Эти сообщения означают, что маршрутизатор находится в рабочем состоянии. Это происходит с истечением таймера Keepalive – по умолчанию 60 секунд.

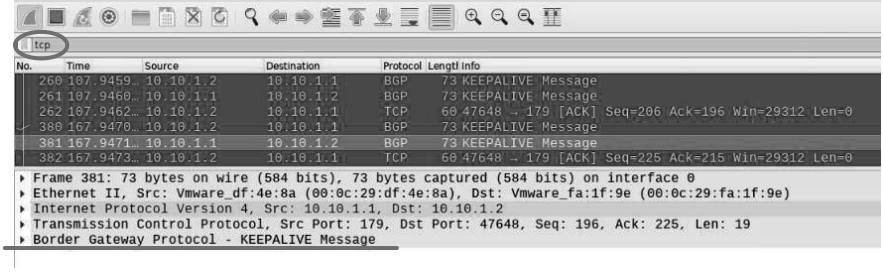


Рисунок 5. Процесс отправки сообщений KEEPALIVE

Как только происходят какие-либо изменения в сети, начинается обмен маршрутной информацией. Для это используются сообщения UPDATE. Каждое

сообщение UPDATE одновременно может нести информацию об одном новом маршруте или о удалении группы старых.

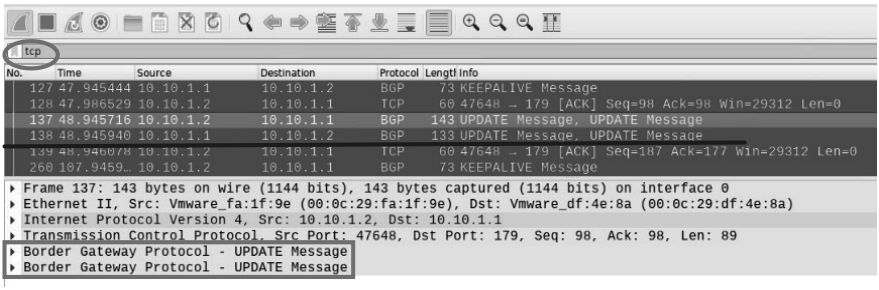


Рисунок 6. Обмен маршрутной информацией в пакетах UPDATE

4.6. Настройка маршрутизатора R3

Теперь настраивается маршрутизатор R3. Для работы протокола BGP на маршрутизаторе R3 с помощью демона zebra настраиваются все сетевые интерфейсы, задействованные в работе протокола BGP. Подключаемся к zebra и входим в режим глобального конфигурирования:

```
[root@elbrus ~]# systemctl start zebra
[root@elbrus ~]# telnet 127.0.0.1 2601
Trying 127.0.0.1...
Connected to 127.0.0.1.
User Access Verification
Password:
Router>
Router enable
Router#
Router# configure terminal
Router(config)#
```

Изменим имя устройства:

```
Router(config)# hostname R3
```

Настройка интерфейса eth0:

```
R3(config)#interface eth0
R3(config-if)# ip address 10.10.3.3/24
R3(config-if)# exit
```

Настройка интерфейса eth1:

```
R3(config)#interface eth1
R3(config-if)# ip address 10.10.2.3/24
R3(config-if)# exit
```

Настройка логического loopback-интерфейса lo:

```
R3(config)#interface lo
R3(config-if)# ip address 192.168.10.1/30
R3(config-if)# ip address 192.168.20.1/30
R3(config-if)# exit
```

Настройка демона zebra завершена, необходимо сохранить конфигурацию. Отключиться от демона можно комбинации Ctrl+D.

```
R3(config)# write memory
Configuration saved to /etc/quagga/zebra.conf
R3(config)# exit
```

Далее на маршрутизаторе R3 необходимо настроить работу протокола BGP с помощью демона bgpd. Запускаем демон, подключаемся к bgpd по telnet и входим в режим глобального конфигурирования:

```
[root@elbrus ~]# systemctl start bgpd
[root@elbrus ~]# telnet 127.0.0.1 2605
Trying 127.0.0.1...
Connected to 127.0.0.1.
User Access Verification
Password:
router-BGP>
router-BGP> enable
```

Для внесения изменений в конфигурацию, переходим в режим глобального конфигурирования:

```
router-BGP# configure terminal
router-BGP(config)#
```

Изменим имя устройства:

```
router-BGP(config)# hostname R3-BGP
```

Теперь необходимо включить процесс BGP на маршрутизаторе и назначить номер автономной системы в нашей сети, а также прописать сети, которые будет анонсировать данный маршрутизатор.

Включаем BGP и задаем номер автономной системы AS 65003, переходим к его конфигурированию;

```
R3-BGP(config)# router bgp 65003
R3-BGP(config-router)#
```

Для работы протоколу BGP сначала необходимо установить соседские отношения с подключёнными непосредственно к нему маршрутизаторами.

В нашей сети соседями маршрутизатора R3 является маршрутизатор R2 и R4, соответственно, указывается их IP-адреса и номера AS:

```
R3-BGP(config-router)# neighbor 10.10.2.2 remote-as
65002
R3-BGP(config-router)# neighbor 10.10.3.4 remote-as
65004
```

Далее в формате `network <x.x.x.x> mask <m.m.m.m>` прописываются сети, которые необходимо анонсировать по протоколу BGP, в нашем случае за сети в AS отвечает loopback адрес маршрутизатора:

```
R3-BGP(config-router)# network 192.168.10.0 mask
255.255.255.252
R3-BGP(config-router)# network 192.168.20.0 mask
255.255.255.252
R3-BGP(config-router)#exit
R3-BGP(config)#
```

Настройка демона bgpd завершена, необходимо сохранить конфигурацию. Отключиться от демона можно комбинации Ctrl+D.

```
R3-BGP(config)# write memory
Configuration saved to /etc/quagga/bgpd.conf
R3-BGP(config)# exit
```

4.7. Настройка маршрутизатора R4

Теперь настраивается маршрутизатор R4. Для работы протокола BGP на маршрутизаторе R4 с помощью демона zebra настраиваются все сетевые

интерфейсы, задействованные в работе протокола BGP. Подключаемся к zebra и входим в режим глобального конфигурирования:

```
[root@elbrus ~]# systemctl start zebra
[root@elbrus ~]# telnet 127.0.0.1 2601
Trying 127.0.0.1...
Connected to 127.0.0.1.
User Access Verification
Password:
Router>
Router enable
Router#
Router# configure terminal
Router(config)#
```

Изменим имя устройства:

```
Router(config)# hostname R4
```

Настройка интерфейса eth0:

```
R4(config)#interface                                eth0
R4(config-if)#       ip       address       10.10.3.4/24
R4(config-if)# exit
```

Настроим интерфейс eth1:

```
R4(config)#               interface               eth1
R4(config-if)#       ip       address       10.10.4.4/24
R4(config-if)# exit
```

Настройка логического loopback-интерфейса lo:

```
R4(config)#interface                                  lo
R4(config-if)#       ip       address       192.168.4.1/30
R4(config-if)# exit
```

Настройка демона zebra завершена, необходимо сохранить конфигурацию. Отключиться от демона можно комбинации Ctrl+D.

```
R4(config)# write memory
Configuration saved to /etc/quagga/zebra.conf
R4(config)# exit
```

Далее на маршрутизаторе R4 необходимо настроить работу протокола BGP с помощью демона bgpd. Запускаем демон, подключаемся к bgpd по telnet и входим в режим глобального конфигурирования:

```
[root@elbrus ~]# systemctl start bgpd
[root@elbrus ~]# telnet 127.0.0.1 2605
Trying 127.0.0.1...
Connected to 127.0.0.1.
User Access Verification
Password:
router-BGP>
router-BGP> enable
router-BGP#
```

Для внесения изменений в конфигурацию, переходим в режим глобального конфигурирования:

```
router-BGP# configure terminal
router-BGP(config)#
```

Изменим имя устройства:

```
router-BGP(config)# hostname R4-BGP
```

Теперь необходимо включить процесс BGP на маршрутизаторе и назначить номер автономной системы в нашей сети, а также прописать сети, которые будет анонсировать данный маршрутизатор.

Включаем BGP и задаем номер автономной системы AS 65004, переходим к его конфигурированию;

```
R4-BGP(config)# router bgp 65004
R4-BGP(config-router)#
```

Для работы протоколу BGP сначала необходимо установить соседские отношения с подключёнными непосредственно к нему маршрутизаторами.

В нашей сети соседями маршрутизатора R4 являются маршрутизаторы R1 и R3, соответственно, указываются их IP-адреса и номера AS:

```
R4-BGP(config-router)# neighbor 10.10.4.1 remote-as
65001
R4-BGP(config-router)# neighbor 10.10.3.3 remote-as
65003
```

Далее в формате network <x.x.x.x> mask <m.m.m.m> прописываются сети, которые необходимо анонсировать по протоколу BGP, в нашем случае за сеть в AS отвечает loopback адрес маршрутизатора:

```
R4-BGP(config-router)# network 192.168.4.0 mask 255.255.255.252
```

Так как R4 находится в транзитной автономной системе между R1 и R3, необходимо прописать транзитные сети между ними:

```
R4-BGP(config-router)#network 10.10.3.0 mask 255.255.255.0
R4-BGP(config-router)#network 10.10.4.0 mask 255.255.255.0
```

Настройка демона bgpd завершена, необходимо сохранить конфигурацию. Отключиться от демона можно комбинации Ctrl+D.

```
R4-BGP(config)# write memory
Configuration saved to /etc/quagga/bgpd.conf
R4-BGP(config)# exit
```

4.8. Проверка работы протокола BGP

После настроек все маршрутизаторы в сети установят BGP-сессии между собой и будут с помощью анонсов BGP анонсировать друг другу префиксы (блоки адресов).

Проверим все полученные маршруты на R1 командой show ip bgp:

```
[root@elbrus ~]# telnet 127.0.0.1 2605
R1-BGP>
R1-BGP> enable
R1-BGP# show ip bgp
BGP table version is 0, local router ID is 192.168.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, = multipath
Network Next Hop Metric LocPrf Weight Path
*  10.10.1.0/24     10.10.1.2  0      0 65002 i
*>                  0.0.0.0    0          32768 i
*  10.10.2.0/24     10.10.4.4         0 65004 65003 i
*>                  10.10.1.2  0      0 65002 i
```

```
*  10.10.3.0/24     10.10.1.2          0 65002 65003 i
*>                  10.10.4.4  0       0 65004 i
*  10.10.4.0/24     10.10.4.4  0       0 65004 i
*>                  0.0.0.0    0           32768 i
*> 192.168.1.0/30   0.0.0.0    0           32768 i
*  172.168.2.0/30   10.10.1.2  0       0 65002 i
*>                  10.10.4.4  0       0 65004 i
*  172.168.10.0/30  10.10.1.2          0 65002 65003 i
*>                  10.10.4.4          0 65004 65003 i
*  172.168.20.0/30  10.10.1.2          0 65002 65003 i
*>                  10.10.4.4          0 65004 65003 i
```

Таблица маршрутов BGP показывает все известные сети, полученные по протоколу BGP (столбец `Network`), следующий маршрутизатор на маршруте к подсети (столбец `Next Hop`), путь до сети через номера автономных систем (столбец `Path`), а также остальные атрибуты пути, такие как метрика, вес и приоритет маршрута `LocPrf`.

Для того, чтобы проверить расширенную статистику BGP, используется команда `show ip bgp summary`. Выполним ее на маршрутизаторе R1:

```
R1-BGP# show ip bgp summary
BGP router identifier 192.168.1.1, local AS number 65001
RIB entries 17, using 1904 bytes of memory
Peers 2, using 14 KiB of memory
Neighbor    V   AS    MsgRcvd MsgSent  Up/Down  State/PfxRcd
10.10.1.2   4 65002      46      59     00:11:32        8
10.10.4.4   4 65004      38      44     00:10:14        8
```

Команда показывает информацию о соседних маршрутизаторах, количество принятых и отправленных сообщений (`MsgRcvd/MsgSent`), время в активном состоянии (`Up/Down`), а также количество принятых префиксов (маршрутов) – `State/PfxRcd`.

Чтобы проверить работу сети, организуем обмен трафиком между автономными системами AS 65001 и AS 65003. С маршрутизатора R1 проверим доступность сети 192.168.10.0/30, находящейся в автономной системе AS 65003 маршрутизатора R3:

```
[root@elbrus ~]# ping 192.168.10.1 -c 3
PING 192.168.10.1 (192.168.10.1) 56(84) bytes of data.
64 bytes from 192.168.10.1: icmp_req=1 ttl=63 time=0.362
ms
```

```
    64 bytes from 192.168.10.1: icmp_req=2 ttl=63 time=0.345
ms
    64 bytes from 192.168.10.1: icmp_req=3 ttl=63 time=0.356
ms
    --- 192.168.10.1 ping statistics ---
    3 packets transmitted, 3 received, 0% packet loss, time
4103ms
    rtt min/avg/max/mdev = 0.315/0.343/0.362/0.020 ms
```

Также проверим доступность сети 192.168.20.0/30:

```
    [root@elbrus ~]# ping 192.168.20.1 -c 3
    PING 192.168.20.1 (192.168.20.1) 56(84) bytes of data.
    64 bytes from 192.168.20.1: icmp_req=1 ttl=63 time=0.331
ms
    64 bytes from 192.168.20.1: icmp_req=2 ttl=63 time=0.352
ms
    64 bytes from 192.168.20.1: icmp_req=3 ttl=63 time=0.343
ms
    --- 192.168.20.1 ping statistics ---
    3 packets transmitted, 3 received, 0% packet loss, time
4103ms
    rtt min/avg/max/mdev = 0.317/0.353/0.345/0.021 ms
```

Ключ –c (count) задает количество посылаемых ICMP запросов. Сети 192.168.10.0/30 и 192.168.20.0/30 в автономной системе AS 65003 доступны, ICMP пакеты успешно возвращаются.

4.9. Распределение трафика в BGP

4.9.1. Фильтрация маршрутов с помощью Prefix-list

Prefix-list или списки префиксов представляют собой списки управления маршрутной информацией, в которых можно что-либо запретить или разрешить.

Для создания списка префиксов используется следующий синтаксис:

`ip prefix-list` <list-name> `[seq` <value>`]` <deny | permit> <network/length> `[ge` <value>`] [le` <value>`]`

List-name – название списка префиксов;

Seq – порядковый номер создаваемого правила в списке префиксов. На соответствие заданным условиям первыми проверяются правила с наименьшим

порядковым номером. Если правило одно, то параметр seq указывать не обязательно, по умолчанию равен 5;

Deny/permit – запретить или разрешить определенные маршруты;

Network/length – запись вида сеть/размер в формате <x.x.x.x>/размер;

Ge и le – не обязательные операторы, необходимы для определения диапазона фильтрации трафика. Ge (Greater or Equal), что означает больше или равно, le (Less or Equal) – меньше или равно.

Пример, запись вида:

```
ip prefix-list PL_test permit 172.16.5.0/24 ge 25 le 26
```

В данном списке префиксов создается разрешающее правило с именем PL_test, которое проверяет первые 24 бита адреса сети (первые три октета) и сравнивает их с входящим анонсом, если первые три октета совпадают с 172.16.5, а маска находится в диапазоне от 25 до 26, то правило выполняется.

Под данное правило попадают следующие сети:

Сеть 172.16.5.0/26 с диапазоном адресов 172.16.5.1 – 172.16.5.62;

Сеть 172.16.5.64/26 с диапазоном адресов 172.16.5.65 – 172.16.5.126;

Сеть 172.16.5.128/26 с диапазоном адресов 172.16.5.129 – 172.16.5.190;

Сеть 172.16.5.192/26 с диапазоном адресов 172.16.5.193 – 172.16.5.254;

Сеть 172.16.5.0/25 с диапазоном адресов 172.16.5.1 – 172.16.5.126;

Сеть 172.16.5.128/25 с диапазоном адресов 172.16.5.129 – 172.16.5.254.

После формирования списка префиксов необходимо применить его к BGP-соседу на входящий или исходящий трафик. Список применяется следующим образом:

```
router bgp <Номер автономной системы>
neighbor <IP-адрес BGP-соседа> prefix-list <Имя списка> in/out
```

После того, как прописаны префикс листы на входящий трафик, политики фильтрации еще не будут работать, необходимо перезапросить маршрутную информацию от соседа. Существует несколько способов сделать это:

Hard reset – жесткий сброс. Выполняется из привилегированного режима (enable) командой `clear ip bgp *`:

```
Router-BGP# clear ip bgp *
```

При этом разрываются все BGP-сессии со всеми соседями, очищается таблица BGP и маршрутизатору необходимо будет заново установить BGP-сессии с соседями и выучить всю маршрутную информацию. Выполнить жесткий сброс для определенного соседа можно командой `clear ip bgp` <IP-адрес соседа>.

Soft reset – мягкий сброс. Позволяет обновить политики маршрутизации, не разрывая BGP-сессию с соседом:

```
Router-BGP# conf t
Router-BGP(config)# router bgp <Номер автономной системы>
Router-BGP(config-router)# neighbor <IP-адрес> soft-reconfiguration inbound
```

При таком способе обновления маршрутизатор сохранит BGP сессию с соседом и всю маршрутную информацию, полученную от него. При этом обновятся входящие политики фильтрации.

Если у маршрутизатора немного BGP соседей, и они анонсируют не большое количество маршрутной информации, то можно использовать жесткий сброс. Если же в сети много соседей, от которых приходит много маршрутов, то реконфигурация таким способом может сильно нагрузить маршрутизатор и занять длительное время, поэтому в таком случае лучше использовать мягкий сброс.

На данный момент маршрутизатор R1 получает анонсы сети 192.168.10.0/30 от своих соседей 10.10.1.2 (маршрутизатор R2) и 10.10.4.4 (маршрутизатор R4). Это можно увидеть из таблицы BGP в поле Next Hop:

```
[root@elbrus ~]# telnet 127.0.0.1 2605
R1-BGP>
R1-BGP> enable
R1-BGP# show ip bgp
BGP table version is 0, local router ID is 192.168.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, S Stale, R Removed
Network          Next Hop      Metric LocPrf Weight Path
* 192.168.10.0/30 10.10.1.2          0   65002 65003 i
*>                10.10.4.4          0   65004 65003 i
```

Для настроенной лабораторной сети с помощью списка префиксов запретим принимать BGP-анонсы сети 192.168.10.0/30 от соседа R2, находящегося в автономной системе AS 65002. Для этого настроим соответствующий список префиксов:

```
R1-BGP# conf t
R1-BGP(config)# ip prefix-list from_R2 seq 5 deny 192.168.10.0/30
R1-BGP(config)# ip prefix-list from_R2 seq 10 permit 0.0.0.0/0 le 32
```

Создаётся список префиксов с именем from_R2, содержащий в себе 2 правила.

Первое правило имеет номер последовательности 5 и запрещает прием BGP-анонсов сети 192.168.10.0/30. Второе правило с номером последовательности 10 разрешает прием всех подсетей с любой длиной маски (0-32).

Когда трафик приходит на маршрутизатор, то сначала проверяется первым правилом с наименьшим номером последовательности. Если пришел анонс о запрещенной сети, то трафик, соответственно, сразу попадает под первое правило и дальнейшая обработка завершается, а запрещенные анонсы не попадают в BGP таблицу маршрутизатора.

Приходящий анонс о любой другой сети также сначала обрабатывается первым правилом и не попадает под него, после переходит на обработку вторым правилом, в котором разрешены любые сети и, соответственно, уже под него попадает, после чего обработка трафика завершается и все не запрещенные маршруты попадают в таблицу BGP.

После того как список префиксов сформирован, необходимо применить его к соседу R2. Так как маршрутизатор R1 принимает анонсы о сети 192.168.10.0/30 от R2, то применяем prefix-list на входящий трафик (in):

```
R1-BGP(config)# router bgp 65001
R1-BGP(config-router)# neighbor 10.10.1.2 prefix-list from_R2 in
R1-BGP(config-router)# exit
```

Чтобы политики фильтрации применились, выполняем перезапрос маршрутной информации у маршрутизатора R2:

```
R1-BGP# clear ip bgp 10.10.1.2
```

После произойдет переустановление BGP сессии, R1 получит анонсы от R2 и обновит свои политики фильтрации трафика. С помощью команды `show ip bgp` проверим какие префиксы получил R1. Часть строк таблицы опущена для краткости:

```
R1-BGP# show ip bgp
BGP table version is 0, local router ID is 192.168.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, = multipath
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop    Metric  LocPrf Weight Path
*> 192.168.1.0/30   0.0.0.0          0         32768 i
*  192.168.2.0/30   10.10.1.2        0             0 65002 i
*>                  10.10.4.4        0             0 65004 i
*> 192.168.10.0/30  10.10.4.4                      0 65004 65003 i
*  192.168.20.0/30  10.10.1.2                      0 65002 65003 i
*>                  10.10.4.4                      0 65004 65003 i
```

Как видно из таблицы BGP, маршрутизатор R1 получил анонс сети 192.168.10.0/30, но теперь эта сеть доступна только через 10.10.4.4 (маршрутизатор R4), так как анонс от R2 об этой сети пришел на маршрутизатор, но был отфильтрован списком префиксов и не добавлен в таблицу BGP.

Теперь трафик до сети 192.168.10.1 идет только через AS 65004:

```
[root@elbrus ~]# traceroute 192.168.10.1
traceroute to 192.168.10.1 (192.168.10.1), 30 hops max, 60 byte packets
 1  10.10.4.4          0.120 ms  0.078 ms  0.050 ms
 192.168.10.1       0.122 ms  0.083 ms  0.105 ms
```

4.9.2. Распределение маршрутов BGP с помощью Route map

Одним из способов распределения трафика в сетях, работающих по протоколу BGP, являются карты маршрутов route map. В контексте BGP карта маршрутов представляет собой способ контроля изменения информации о маршрутах. Распределение маршрутов применяется, например, когда

необходимо организовать высокую стабильность сервисов, перенаправляя трафик через заведомо более надежные участки сети.

Синтаксис настройки карты маршрутов выглядит следующим образом:

route-map <имя карты> <deny | permit> <seq>

<match {критерий}>

<set {критерий}>

deny | permit – разрешающее или запрещающее правило

seq – порядковый номер карты маршрутов. При применении карты маршрутов к входящим или исходящим маршрутам, первый набор условий применяется через экземпляр с наименьшим номером. Если первый набор условий не удовлетворяется, происходит переход к экземпляру карты маршрутов с более высоким номером.

Все карты маршрутов состоят из списка команд настройки match и set. Командой match задаются критерии соответствия, а командой set задаются действия, выполняемые при удовлетворении условий, задаваемых командой match.

В таблице 1 указаны некоторые основные критерии для команды соответствия match.

Таблица 1. Критерии соответствия и команды конфигурирования

Критерий соответствия	Команда конфигурации
Network/mask	match ip address prefix-list
AS-path	match as-path
BGP next-hop address	match ip next-hop

Network/mask – соответствие по сети и маске;

AS-path – соответствие по атрибуту пути AS-path;

BGP next-hop address – соответствие по следующему маршрутизатору BGP.

В таблице 2 указаны некоторые основные атрибуты пути для команды действия set:

Таблица 2. Атрибуты и команды конфигурации

Атрибуты пути	Команда конфигурации
Weight	set weight
Local Preference	set local-preference
MED	set metric
BGP next-hop	set next-hop

Weight – установка стоимости пути;

Local Preference – установка локального приоритета;

MED – установка метрики;

BGP next-hop – установка следующего BGP-маршрутизатора.

В нашей сети настроим пути трафика таким образом, чтобы трафик до подсети 192.168.10.0/30 шел через автономную систему AS 65002, а трафик до сети 192.168.20.0/30 шел через автономную систему AS 65004. Для этого воспользуемся рассмотренными ранее списками префиксов и картами маршрутов.

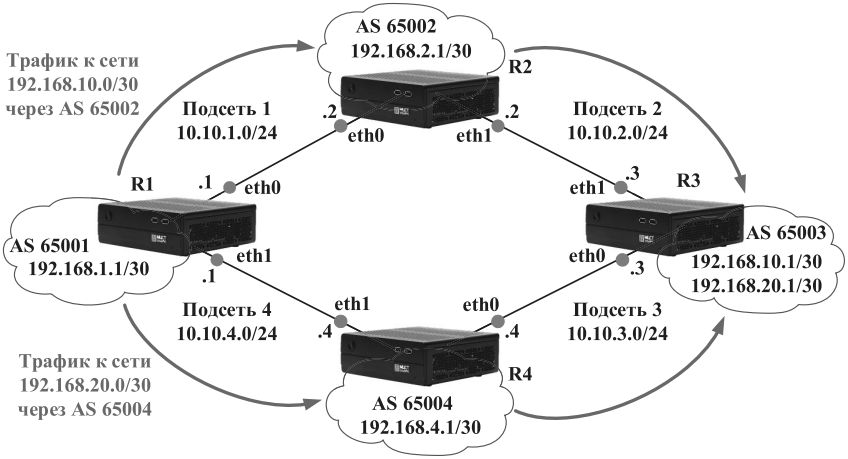


Рисунок 7. Организация распределения трафика в сети

Для начала отвяжем от соседа R2 настроенный ранее список префиксов, запрещающий принимать от него анонсы сети 192.168.10.0/30:

```
R1-BGP# conf terminal
```

```
R1-BGP(config)# router bgp 65001
R1-BGP(config-router)# no neighbor 10.10.1.2 prefix-list from_R2 in
R1-BGP(config-router)# exit
R1-BGP(config)# exit
```

Перезапросим маршруты у R2 и обновим политики фильтрации:

```
R1-BGP#
R1-BGP# clear ip bgp 10.10.1.2
```

Теперь R1 снова получает анонсы сети 192.168.10.0/30 от R2 и доступен через автономные системы 65002 и 65003:

```
R1-BGP# show ip bgp
BGP table version is 0, local router ID is 192.168.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, S Stale, R Removed
Network             Next Hop     Metric LocPrf Weight Path

* 192.168.10.0/30 10.10.1.2            0    65002 65003 i
*>               10.10.4.4             0    65004 65003 i
*  192.168.20.0/30 10.10.1.2           0    65002 65003 i
*>                10.10.4.4            0    65004 65003 i
```

Для реализации распределения трафика создадим список префиксов, которым выделим анонсы сети 192.168.20.0/30 от R2:

```
R1-BGP# conf terminal
R1-BGP(config)# ip prefix-list PL_from_R2 seq 5 permit 192.168.20.0/30
```

Далее создадим карты маршрутов, в которых разрешим получение анонсов, попадающих под правила созданного списка префиксов. Для этого воспользуемся критерием соответствия `Network/mask`:

```
R1-BGP(config)# route-map for_AS65002 permit 10
```

```
R1-BGP(config-route-map)# match ip address prefix-
list PL_from_R2
```

Воспользуемся атрибутом пути `Local preference` для установки приоритета маршрута, попадающего под правило списка префикса:

```
R1-BGP(config-route-map)# set local-preference 50
R1-BGP(config-route-map)# exit
```

Создадим второй экземпляр этой карты маршрута для всех остальных приходящих анонсов, но уже с порядковым номером 20:

```
R1-BGP(config)# route-map for_AS65002 permit 20
```

Также воспользуемся атрибутом пути `Local preference` для установки приоритета всех остальных маршрутов на R1:

```
R1-BGP(config-route-map)# set local-preference 100
R1-BGP(config-route-map)# exit
```

Рассмотрим подробнее произведенные настройки. Создана карта маршрута с именем for_AS65002, имеющая порядковый номер 10, т.е. сначала ко всему приходящему трафику будет применяться набор действий, описанный в экземпляре карты с меньшим приоритетом.

В первом экземпляре карты маршрута командой match задается критерий соответствия `Network/mask`, для которого используется команда `ip address prefix-list`, проверяющая соответствие списку префикса PL_from_R2.

Если условие `match` выполняется, то командой `set` устанавливается приоритет маршрута для сети, удовлетворяющей правилу заданного списка префиксов с помощью атрибута пути `local-preference`. Атрибут пути `local-preference` имеет числовые значения, чем выше значение атрибута пути, тем этот путь приоритетнее для трафика.

Второй экземпляр карты маршрута необходим для всех остальных анонсов, не попадающих под правила списка префиксов, этому трафику устанавливается другое значение атрибута пути `local preference`.

После того как список карта маршрута сформирована, необходимо применить ее к соседу R2. Так как маршрутизатор R1 принимает анонсы о сети 192.168.20.0/30 от R2, то применяем карту маршрута на входящий трафик (in):

```
R1-BGP(config)# router bgp 65001
```

```
R1-BGP(config-router)# neighbor 10.10.1.2 route map
fpr_AS65002 in
R1-BGP(config-router)# exit
```

Рассмотрим подробнее как работает данная карта маршрута:

- Маршрутизатор R3 анонсирует сеть 192.168.20.0/30 маршрутизатору R2, а R2 анонсирует ее маршрутизатору R1;
- Приходя на маршрутизатор R1, анонс сети 192.168.20.0/30 проверяется первым экземпляром карты маршрутов и удовлетворяет правилу списка префиксов PL_from_R2, так как данный список выделяет сеть 192.168.20.0/30;
- Так как анонс попал под правило списка префиксов, то к нему применяется команда `set`, которая устанавливает приоритет данного полученного маршрута на значение 50, а дальнейшая обработка анонса завершается ввиду выполненного условия;
- Если на маршрутизатор R1 от R2 приходит анонс любой другой сети, то сначала он также проверяется первым экземпляром карты маршрута, под правила которой не попадает, так как не соответствует сети 192.168.20.0/30;
- Далее анонс проверяется вторым экземпляром карты маршрутов, в которой не производится никаких проверок, а сразу командой `set` устанавливается приоритет маршрута на значение 100.

Таким образом, данной картой маршрута мы понизили приоритет трафика до сети 172.16.20.0/30 через AS65002.

Теперь произведем аналогичные настройки для анонсов от маршрутизатора R4. Создадим список прификсов и карту маршрутов, в которой понизим приоритет трафика до сети 192.168.10.0/30 через AS65004, а приоритет всех остальных маршрутов повысим.

Настроим список префиксов, которым выделим анонсы сети 192.168.10.0/30 от маршрутизатора R4:

```
R1-BGP# conf terminal
R1-BGP(config)# ip prefix-list PL_from_R4 seq 5
permit 192.168.10.0/30
```

Далее создадим карты маршрутов, в которых разрешим получение анонсов, попадающих под правила созданного списка префиксов. Для этого воспользуемся критерием соответствия `Network/mask`:

```
R1-BGP(config)# route-map for_AS65004 permit 10
```

```
R1-BGP(config-route-map)# match ip address prefix-list PL_from_R4
```

Установка приоритета для выделенных анонсов:

```
R1-BGP(config-route-map)# set local-preference 50
R1-BGP(config-route-map)# exit
```

Создадим второй экземпляр этой карты маршрутов для всех остальных приходящих анонсов, но уже с порядковым номером 20 и установим для них приоритет 100:

```
R1-BGP(config)# route-map for_AS65004 permit 20
R1-BGP(config-route-map)# set local-preference 100
R1-BGP(config-route-map)# exit
```

Применим созданную карту маршрутов к соседу R4. Так как маршрутизатор R1 принимает анонсы о сети 192.168.10.0/30 от R4, то применяем карту маршрута на входящий трафик (in):

```
R1-BGP(config)# router bgp 65001
R1-BGP(config-router)# neighbor 10.10.4.4 route map fpr_AS65004 in
R1-BGP(config-router)# exit
R1-BGP(config)# exit
```

После сконфигурированных карт маршрутов, перезапустим BGP-сессии с маршрутизаторами R2 и R4, обновив тем самым настроенные политики распределения трафика:

```
R1-BGP# clear ip bgp *
```

Когда BGP-сессии восстановятся, R1 получит анонсы всех сетей от своих соседей. Анонсы обработаются картами маршрутов и маршрутизатор сформирует новую таблицу BGP уже содержащую настроенные приоритеты трафика:

```
R1-BGP# show ip bgp
BGP table version is 0, local router ID is 192.168.1.1 Status codes: s suppressed, d damped, h history, * valid, > best, S Stale, R Removed
Network Next Hop Metric LocPrf Weight Path
```

```
   *> 192.168.1.0/30    0.0.0.0                0         32768 i
   *   192.168.2.0/30   10.10.1.2      0    100             0
65002 i
   *>                    10.10.4.4       0   100           0
65004 i
   *> 192.168.10.0/30  10.10.1.2             100    0  65002
65003 i
   *                     10.10.4.4             50    0  65004
65003 i
   *   192.168.20.0/30  10.10.1.2              50    0  65002
65003 i
   *>                    10.10.4.4            100    0  65004
65003 i
```

В таблице BGP появились значения в столбце локального приоритета `LocPrf` (Local Preference). Как видно, все маршруты получили свои приоритеты исходя из правил карт маршрутов.

Так маршрут к сети 192.168.10.0/30 через R2 (10.10.1.2) имеет приоритет 100, а через R4 (10.10.4.4) – приоритет 50. Для маршрута к сети 192.168.20.0/30 приоритеты имеют обратное значение. Это позволит распределить исходящий от R1 трафик к подсетям 192.168.10.0/30 и 192.168.20.0/30 через разные автономные системы.

Проверим пути трафика с помощью команды `traceroute`.

Трассировка маршрута до 192.168.10.0/30:

```
[root@elbrus ~]# traceroute 192.168.10.1
traceroute to 192.168.10.1 (192.168.10.1), 30 hops
max, 60 byte packets
 1  10.10.1.2        0.130 ms  0.092 ms  0.065 ms
 2  192.168.10.1     0.155 ms  0.147 ms  0.152 ms
```

Трассировка маршрута до 192.168.20.0/30:

```
[root@elbrus ~]# traceroute 192.168.20.1
traceroute to 192.168.20.1 (192.168.20.1), 30 hops
max, 60 byte packets
 1  10.10.4.4         0.125 ms  0.091 ms  0.065 ms
 2  192.168.20.1      0.115 ms  0.095 ms  0.091 ms
```

Как показывает трассировка маршрутов, трафик до сетей 192.168.10.0/30 и 192.168.20.0/30, находящихся в автономной системе AS 65003, распределяется между разными автономными системами.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Основные концепции протокола BGP. Применение и принцип работы.
2. Назначение автономных систем AS и их роль в работе протокола BGP.
3. Каким образом происходит выбор наилучших маршрутов в сети, работающей по протоколу BGP?
4. Этапы установления BGP сессии.
5. Атрибуты пути протокола BGP, рассмотренные в работе, их назначение.

Практикум № 17
Статическая маршрутизация IPv6

1. ЦЕЛЬ РАБОТЫ

Данная работа предназначена для изучения статической маршрутизации трафика по протоколу IPv6 в сети Интернет, получения практического опыта в работе с маршрутизацией трафика IPv6 в ОС Альт; работа с основными конфигурационными файлами сетевой подсистемы, настройка и управление сетевыми интерфейсами, умение работать с утилитами для диагностики сетевых соединений, работающих по протоколу IPv6.

2. ЗАДАНИЕ НА РАБОТУ

В работе используется топология сети, представленная на рисунке 1. Сеть состоит из трех подсетей, работающих по протоколу IPv6. Маршрутизаторы R2 и R3 являются транзитными маршрутизаторами и осуществляют пересылку пакетов между подсетями 1 и 3, а также выполняют роль шлюзов для маршрутизаторов R1 и R4. Маршрутизаторам R1, R2, R3 и R4 соответствуют рабочие станции под номерами PC1, PC2, PC3 и PC4, соответственно. Адресация в лабораторной сети осуществляется следующим образом:

- Подсеть 1: 2001:fb6:31c0:(N_{min})::/64 или 2001:fb6:31c0:1::/64
- Подсеть 2: 2001:fb6:31c0:(N_{min} + 1)::/64 или 2001:fb6:31c0:2::/64
- Подсеть 2: 2001:fb6:31c0:(N_{min} + 2)::/64 или 2001:fb6:31c0:3::/64

где $N_{min} = 1$ – наименьший из четырех номер рабочих станций.

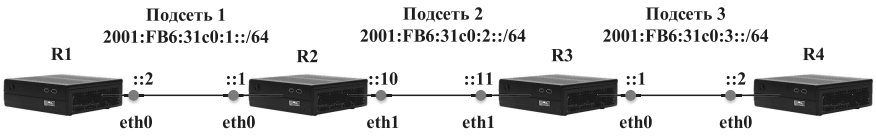


Рисунок 1. Схема сети лабораторной работы

Основные задачи работы:

- Настроить все сетевые интерфейсы машин в соответствии со схемой лабораторной сети;
- Сделать маршрутизаторы R1 и R2 шлюзами по умолчанию для маршрутизаторов R1 и R4, прописав на них соответствующие маршруты;

- На транзитных маршрутизаторах R1 и R2 настроить пересылку пакетов между интерфейсами и прописать маршруты в удаленные подсети;
- Проверить работу сети с помощью утилит диагностики сетевых соединений: ping6, traceroute6.

3. ТЕРМИНЫ И ИСПОЛЬЗУЕМЫЕ КОМАНДЫ

Термины

Статическая маршрутизация – вид маршрутизации трафика в локальных и глобальных сетях, когда маршруты к тем или иным подсетям прописываются вручную администратором сети. Вся маршрутизация при этом происходит без участия каких-либо протоколов маршрутизации.

Статическая маршрутизация имеет свои достоинства и недостатки. К достоинствам можно отнести следующие пункты:

- Легкость настройки и отладки;
- Отсутствие дополнительной нагрузки на сеть из-за отсутствия трафика управления протоколов маршрутизации;
- Низкая нагрузка на аппаратное обеспечение сетевых устройств.

К недостаткам статической маршрутизации относятся:

- Сложность масштабирования, трудоемкий процесс конфигурирования при большом количестве маршрутов;
- Сложность при изменении топологии сети, требуется вмешательство администратора сети и настройка новых актуальных маршрутов;
- Отсутствие динамической балансировки трафика в зависимости от загруженности каналов связи.

Статические маршруты могут быть маршрутами по умолчанию или иметь конкретные маршруты для различных подсетей.

Статический маршрут по умолчанию является маршрутом, который будет соответствовать IP-адресам всех пакетов. Так, имея маршрут по умолчанию, маршрутизатор будет перенаправлять все пакеты с неизвестной ему сетью назначения на заданный в маршруте по умолчанию адрес.

Маршруты по умолчанию часто используется между граничным маршрутизатором компании и маршрутизатором Интернет-провайдера, перенаправляя на него все пакеты, исходящие из локальной сети предприятия.

Также маршруты по умолчанию используются маршрутизаторами, когда в своей таблице маршрутизации они не находят соответствия для между адресом

назначения и адресом следующего маршрутизатора, на который необходимо послать пакет для достижения сети назначения. Маршрутизатор проверяет свою таблицу маршрутизации и вместо того, чтобы отбросить пакет, отправляет его на свой маршрут по умолчанию.

Таблица маршрутизации содержит основную информацию для работы маршрутизатора, а именно:

- Адрес и маску сети назначения;
- Шлюз, обозначающий следующее устройство, через которое доступна сеть назначения;
- Интерфейс, через который доступен шлюз;
- Метрики маршрутов, чем ниже метрика маршрута в определенную сеть, тем приоритетнее данный маршрут.

Также таблицы маршрутизации содержат подключенные напрямую к маршрутизатору сети, интерфейс, к которому они подключены и его IP-адрес.

Используемые команды

`cd – (Change directory)` команда перемещения между директориями;

`ls – (list)` команда просмотра содержимого директории;

`nano –` текстовый редактор для создания и редактирования текстовых файлов;

`ip` – утилита для настройки и управления сетевыми интерфейсами. Для работы с протоколом IPv6 используется ключ -6:

`ip -6 address` – отображение информации о сетевых интерфейсах;

`ip -6 route –` отображение информации о маршрутах;

`ip -6 neighbor` – отображение NDP таблицы устройства.

`grep` – утилита для поиска информации;

`ping6` – утилита для проверки сетевых соединений в сетях IPv6;

`traceroute6` – утилита для трассировки маршрута в сетях IPv6;

`mtr6` – утилита для трассировки маршрута, показывает информацию в реальном времени в сетях IPv6.

4. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

4.1. Настройка маршрутизатора R1

Для настройки маршрутизации в сети сначала необходимо настроить все сетевые интерфейсы на четырех маршрутизаторах.

Конфигурационные фалы всех сетевых интерфейсов находятся в директории /etc/net/ifaces. Переход в директорию осуществляется командой `cd`:

```
[user@elbrus]$ cd /etc/net/ifaces/
```

Воспользуемся командой `ls` для того, чтобы просмотреть содержимое директории:

```
[user@elbrus ifaces]$ ls
```

Вывод команды `ls` покажет директории всех сетевых интерфейсов: eth0, eth1, eth2 и петлевого loopback-интерфейса lo.

Так как на R1 необходим только интерфейс eth0, то переходим в его директорию командой `cd` и выведем её содержимое командой `ls`:

```
[user@elbrus ifaces]$ cd eth0
[user@elbrus ifaces]$ ls
Options
```

В директории находится файл options - основной конфигурационный файл интерфейса eth0.

Для настройки интерфейса eth0 необходимо в конфигурационном файле /eth0/options изменить основные параметры интерфейса. Для редактирования конфигурационного файла нужны права суперпользователя root.

Вход в режим суперпользователя осуществляется командой `su-`, после чего требуется ввести пароль суперпользователя:

```
[user@elbrus eth0]$ su-
```

Отредактировать конфигурационный файл можно открыв его с помощью редактора nano:

```
[root@elbrus]# nano /etc/net/ifaces/eth0/options
```

При стандартной настройке системы все сетевые интерфейсы управляются автоматически через Network Manager, необходимо включить конфигурирование интерфейсов вручную с помощью etcnet. Для этого в открывшемся окне редактора необходимо изменить основные параметры:

`BOOTPROTO=static` – использовать статически заданный IP-адрес;

NM_CONTROLLED=no – отключаем автоматическое управление интерфейсом через Network Manager;

DISABLED=no – включаем управление интерфейсом через etcnet;

ONBOOT=yes – включать интерфейс при загрузке системы;

CONFIG_IPV6=yes – режим работы с IPv6-адресами.

После настройки используется Ctrl+O чтобы сохранить изменения и Ctrl+X для выхода из файла конфигурации.

Далее необходимо назначить статический IPv6-адрес на интерфейс eth0. Для этого в /etc/net/ifaces/eth0 нужно создать файл с именем ipv6address и в него вписать необходимый IPv6-адрес.

Используя редактор nano, создается файл с именем ipv6address, в котором прописывается сетевой адрес:

```
[root@elbrus eth0]# nano ipv6address
```

2001:fb6:31c0:1::2/64 – прописывается статический IPv6-адрес;

Сохранение настроек Ctrl+O и выход из редактирования Ctrl+X.

Так как R2 является шлюзом по умолчанию для маршрутизатора R1, то на R1 необходимо прописать маршрут, по умолчанию отправляющий все пакеты на маршрутизатор R2. Для этого в /etc/net/ifaces/eth0 создается файл с именем ipv6route и в него прописывается маршрут по умолчанию:

```
[root@elbrus eth0]# nano ipv6route
```

default via 2001:fb6:31c0:1::1 – маршрут по умолчанию на R2;

Чтобы изменения применились, необходим перезапуск сетевой службы:

```
[root@elbrus eth0]#service network restart
```

Выход из суперпользователя осуществляется командой exit или Ctrl+D.

Проверяется настройка IPv6-адреса командой ip address или сокращенно ip a с ключом -6 (Показать только IPv6-адреса):

```
[user@ elbrus]$ ip -6 a
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500
qlen 1000
       inet6 2001:fb6:31c0:1::2/64 scope global
          valid_lft forever preferred_lft forever
       inet6 fe80::9aa7:b0ff:fe00:dea0/64 scope link
          valid_lft forever preferred_lft forever
```

Первая запись: `inet6 2001:fb6:31c0:1::2/64 scope global` – это global unicast адрес, сконфигурированный нами и использующийся для адресации в сетях IPv6.

Вторая запись: `inet6 fe80::9aa7:b0ff:fe00:dea0/64 scope link` – это link-local адрес, назначенный системой автоматически и предназначенный для взаимодействия между устройствами в пределах локального сегмента сети. Link-local адреса используют подсеть FE80::/64, а оставшаяся часть адреса берется из MAC-адреса интерфейса. Данные адреса не маршрутизируются в сети Интернет и нужны для адресации на канале между двумя устройствами, а также для обеспечения работы необходимых механизмов IPv6.

Для проверки таблицы маршрутизации на R1. Используется команда `ip route` с ключом -6 (для отображения IPv6 маршрутов). Так как в работе используются IPv6-адреса подсетей, начинающиеся с 2001, то для уточнения поиска по таблице маршрутизации дополним запрос `ip -6 r` командой `grep 2001`:

```
[root@elbrus ~]# ip -6 r | grep 2001
2001:fb6:31c0:1::/64 dev eth0 proto kernel metric 256
default via 2001:fb6:31c0:1::1 dev eth0 metric 1024
```

В таблице маршрутизации присутствуют две записи. Первая запись – непосредственно подключенная напрямую к интерфейсу eth0 IPv6 сеть 2001:fb6:31c0:1::/64. Вторая запись – наш прописанный маршрут по умолчанию на маршрутизатор R2.

4.2. Настройка маршрутизатора R2

Для настройки сетевых интерфейсов на маршрутизаторе R2 редактируются их конфигурационные файлы в директории /etc/net/ifaces/.

Для интерфейса eth0 в /etc/net/ifaces/eth0 редактируется файл options:

```
[root@elbrus]# nano /etc/net/ifaces/eth0/options
BOOTPROTO=static
NM_CONTROLLED=no
DISABLED=no
ONBOOT=yes
CONFIG_IPV6=yes
```

Настройка IPv6-адреса на eth0:

```
[root@elbrus]# nano /etc/net/ifaces/eth0/ipv6address
2001:fb6:31c0:1::1/64
```

Для интерфейса eth1 в /etc/net/ifaces/eth1 редактируется файл options:

```
[root@elbrus]# nano /etc/net/ifaces/eth1/options
BOOTPROTO=static
NM_CONTROLLED=no
DISABLED=no
ONBOOT=yes
CONFIG_IPV6=yes
```

Интерфейсу eth1 на R2 принадлежит адрес из второй подсети. Настройка IPv6-адреса на eth1:

```
[root@elbrus]# nano /etc/net/ifaces/eth1/ipv6address
2001:fb6:31c0:2::10/64
```

Так как R2 является транзитным маршрутизатором, у него должна быть возможность перенаправлять пакеты из одного интерфейса в другой. Для этого необходимо включить форвардинг пакетов на уровне ядра системы. В файле /etc/net/sysctl.conf нужно добавить строку:

```
net.ipv6.conf.all.forwarding=1
[root@elbrus]# nano /etc/net/sysctl.conf
net.ipv6.conf.all.forwarding=1
```

После произведенных настроек интерфейсов и форвардинга пакетов, необходимо перезапустить сетевую службу и проверить настроенные IP-адреса:

```
[root@elbrus]# service network restart
[root@elbrus]# ip -6 address
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500
qlen 1000
       inet6 2001:fb6:31c0:1::1/64 scope global
          valid_lft forever preferred_lft forever
       inet6 fe80::9aa7:b0ff:fe00:dd60/64 scope link
          valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500
qlen 1000
       inet6 2001:fb6:31c0:2::10/64 scope global
          valid_lft forever preferred_lft forever
```

```
inet6 fe80::9aa7:b0ff:fe00:dd61/64 scope link
   valid_lft forever preferred_lft forever
```

После настроек необходима проверка сетевой связности между подсетями 1 и 2. Для этого с маршрутизатора R1 проверяется доступность подсети 2001:fb6:31c0:2::/64, находящейся за интерфейсом eth1 маршрутизатора R2.

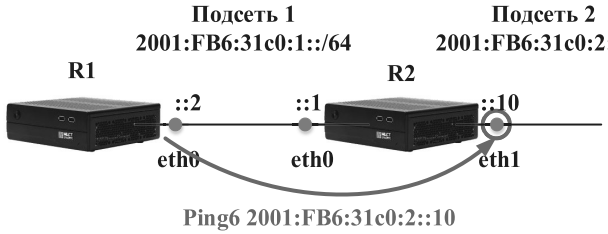


Рисунок 2. Проверка связности сети командой ping6

Чтобы проверить работу сети IPv6, используется утилита ping6, работающая по протоколу ICMPv6, специально разработанного для работы в сетях IPv6.

На R1 выполняется команда `ping6` и указывается адрес интерфейса eth1 маршрутизатора R2:

```
[root@elbrus ~]# ping6 2001:fb6:31c0:2::10 -c 3
PING 2001:fb6:31c0:2::10(2001:fb6:31c0:2::10) 56 data bytes
64 bytes from 2001:fb6:31c0:2::10: icmp_seq=1 ttl=64
time=0.184 ms
64 bytes from 2001:fb6:31c0:2::10: icmp_seq=2 ttl=64
time=0.132 ms
64 bytes from 2001:fb6:31c0:2::10: icmp_seq=3 ttl=64
time=0.102 ms
--- 2001:fb6:31c0:2::10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 0.102/0.139/0.184/0.035 ms
```

Ключ –c (count) задает количество посылаемых ICMP запросов. Интерфейс eth1 маршрутизатора R2 доступен, ICMP пакеты успешно возвращаются.eth1 доступен, ICMP пакеты успешно возвращаются.

После того как взаимодействие между R1 и R2 установлено, устройства добавят друг друга в свою таблицу соответствия сетевых и физических адресов (ARP таблица для IPv4), установив соответствие между MAC-адресами и IPv6-адресами интерфейсов маршрутизаторов.

В сетях IPv6 протокол преобразования адресов ARP заменен на протокол обнаружения соседних устройств NDP (Neighbor Discovery Protocol).

Проверка таблицы NDP на маршрутизаторе R1 осуществляется командой `ip n` (neighbor) с ключом -6 (для протокола IPv6):

```
[root@elbrus ~]# ip -6 n
fe80::9aa7:b0ff:fe00:dd60 dev eth0 lladdr 98:a7:b0:00:dd:60
REACHABLE
2001:fb6:31c0:1::1 dev eth0 lladdr 98:a7:b0:00:dd:60 REACHABLE
```

NDP таблица содержит две записи, устанавливающие соответствие между MAC-адресами и IPv6-адресами соседнего узла, т.е. маршрутизатора R2. Первая запись устанавливает соответствие для link-local адреса R2, а вторая для global unicast IPv6-адреса R2.

В NDP таблице маршрутизатора R2 также будут находиться IPv6-адреса и соответствующие им MAC-адреса маршрутизатора R1:

```
[root@elbrus ~]# ip -6 n
fe80::9aa7:b0ff:fe00:dea0 dev eth0 lladdr 98:a7:b0:00:de:a0
REACHABLE
2001:fb6:31c0:1::2 dev eth0 lladdr 98:a7:b0:00:de:a0 REACHABLE
```

Отметка `REACHABLE` показывает, что запись о соответствии появилась недавно, и IPv6-адреса на R1 доступны.

Отметка `STALE` означает, что запись о доступности соседнего устройства устарела.

Отметка `UNREACHABLE` означает недоступность соседнего устройства по этому IP-адресу.

Поскольку R2 является транзитным маршрутизатором между подсетями 1 и 3, у него должен быть маршрут в третью подсеть. Взаимосвязь с подсетью 3 происходит через интерфейс eth1, поэтому маршрут добавляется в конфигурационном файле этого интерфейса:

```
[root@elbrus]# nano /etc/net/ifaces/eth1/ipv6route
2001:fb6:31c0:3::/64 via 2001:fb6:31c0:2::11
```

Запись добавляет маршрут в подсеть 3 через интерфейс eth0 маршрутизатора R3.

На R2, после настройки маршрута, необходима перезагрузка сетевой службы и проверка таблицы маршрутизации. Так же дополняем запрос `ip -6 r` командой `grep 2001`:

```
[root@elbrus]# service network restart
[root@elbrus]# ip -6 r | grep 2001
```

```
2001:fb6:31c0:1::/64 dev eth0 proto kernel metric 256
2001:fb6:31c0:2::/64 dev eth1 proto kernel metric 256
2001:fb6:31c0:3::/64 via 2001:fb6:31c0:2::11 dev eth1
metric 1024
```

В таблице маршрутизации первые две записи означают подключенные напрямую IPv6 сети к интерфейсам eth0 и eth1. Последняя запись – прописанный маршрут в подсеть 3 с интерфейса eth1 через транзитный маршрутизатор R3.

4.3. Настройка маршрутизатора R3

Для настройки сетевых интерфейсов на маршрутизаторе R3 редактируются их конфигурационные файлы в директории /etc/net/ifaces/.

Для интерфейса eth0 в /etc/net/ifaces/eth1 редактируется файл options:

```
[root@elbrus]# nano /etc/net/ifaces/eth1/options
BOOTPROTO=static
NM_CONTROLLED=no
DISABLED=no
ONBOOT=yes
CONFIG_IPV6=yes
```

Настройка IPv6-адреса на eth1:

```
[root@elbrus]# nano /etc/net/ifaces/eth1/ipv6address
2001:fb6:31c0:2::11/64
```

Для интерфейса eth0 в /etc/net/ifaces/eth0 редактируется файл options:

```
[root@elbrus]# nano /etc/net/ifaces/eth0/options
BOOTPROTO=static
NM_CONTROLLED=no
DISABLED=no
ONBOOT=yes
CONFIG_IPV6=yes
```

Интерфейсу eth0 на R3 принадлежит адрес из третьей подсети. Настройка IPv6-адреса на eth0:

```
[root@elbrus]# nano /etc/net/ifaces/eth0/ipv6address
2001:fb6:31c0:3::1/64
```

Поскольку R3 является транзитным маршрутизатором, у него должна быть возможность перенаправлять пакеты из одного интерфейса в другой. Для этого, как и на R2, включается форвардинг пакетов на уровне ядра системы. В файле /etc/net/sysctl.conf нужно добавить строку:

```
net.ipv6.conf.all.forwarding=1
[root@elbrus]# nano /etc/net/sysctl.conf
net.ipv6.conf.all.forwarding=1
```

Аналогично транзитному маршрутизатору R2, у маршрутизатора R3 должен быть маршрут в первую подсеть. Взаимосвязь с подсетью 1 происходит через интерфейс eth1, поэтому маршрут добавляется в конфигурационном файле этого интерфейса:

```
[root@elbrus]# nano /etc/net/ifaces/eth1/ipv6route
2001:fb6:31c0:1::/64 via 2001:fb6:31c0:2::10
```

Запись добавляет маршрут в подсеть 1 через маршрутизатор R2.

После произведенных настроек интерфейсов и форвардинга пакетов, необходим перезапуск сетевой службы и проверка настроенных IPv6-адресов и маршрутов:

```
[root@elbrus]# service network restart
[root@elbrus]# ip -6 address
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500
qlen 1000
       inet6 2001:fb6:31c0:3::1/64 scope global
          valid_lft forever preferred_lft forever
       inet6 fe80::9aa7:b0ff:fe00:de20/64 scope link
          valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500
qlen 1000
       inet6 2001:fb6:31c0:2::11/64 scope global
          valid_lft forever preferred_lft forever
       inet6 fe80::9aa7:b0ff:fe00:de21/64 scope link
          valid_lft forever preferred_lft forever
```

Таблица маршрутизации на R3. В таблице маршрутизации информация о подключенных напрямую подсетях, а также о том, что первая подсеть 2001:fb6:31c0:1::/64 доступна через маршрутизатор R2:

```
[root@elbrus ~]# ip -6 r | grep 2001
2001:fb6:31c0:1::/64 via 2001:fb6:31c0:2::10 dev eth1
metric 1024
2001:fb6:31c0:2::/64 dev eth1 proto kernel metric 256
2001:fb6:31c0:3::/64 dev eth0 proto kernel metric 256
```

После настроек необходима проверка сетевой связности между подсетями 1 и 3 командой `ping6`. Для этого с маршрутизатора R1 проверяется доступность подсети 2001:fb6:31c0:3::/64, находящейся за интерфейсом eth1 маршрутизатора R3:

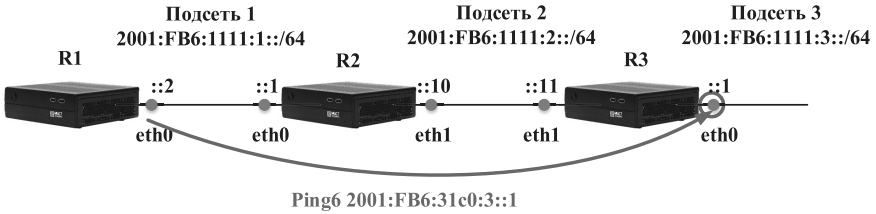


Рисунок 3. Проверка связности сети командой ping6

```
[root@elbrus ~]# ping6 2001:fb6:31c0:3::1 -c 3
PING 2001:fb6:31c0:3::1(2001:fb6:31c0:3::1) 56 data bytes
64 bytes from 2001:fb6:31c0:3::1: icmp_seq=1 ttl=63 time=0.145
ms
64 bytes from 2001:fb6:31c0:3::1: icmp_seq=2 ttl=63 time=0.132
ms
64 bytes from 2001:fb6:31c0:3::1: icmp_seq=3 ttl=63 time=0.130
ms
--- 2001:fb6:31c0:3::1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 0.130/0.135/0.145/0.015 ms
```

ICMP пакеты успешно возвращаются, интерфейс eth0 маршрутизатора R3 доступен.

4.4. Настройка маршрутизатора R4

Для настройки сетевых интерфейсов на маршрутизаторе R4 редактируются их конфигурационные файлы в директории /etc/net/ifaces/.

Для интерфейса eth0 в /etc/net/ifaces/eth0 редактируется файл options:

```
[root@elbrus]# nano /etc/net/ifaces/eth0/options
BOOTPROTO=static
```

```
NM_CONTROLLED=no
DISABLED=no
ONBOOT=yes
CONFIG_IPV6=yes
```

Настройка IPv6-адреса на eth0:

```
[root@elbrus]# nano /etc/net/ifaces/eth0/ipv6address
2001:fb6:31c0:3::2/64
```

Так как R3 является шлюзом по умолчанию для маршрутизатора R4, то на R4 необходимо прописать маршрут, по умолчанию отправляющий все пакеты на маршрутизатор R3. Для этого в /etc/net/ifaces/eth0 создается файл с именем ipv6route и в него прописывается маршрут по умолчанию:

```
[root@elbrus eth0]# nano ipv6route
default via 2001:fb6:31c0:3::1 -
```

маршрут по умолчанию на R3;

После произведенных настроек необходим перезапуск сетевой службы и проверка настроенных IPv6-адресов и маршрутов:

```
[root@elbrus]# service network restart
[root@elbrus]# ip -6 a
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500
qlen 1000
    inet6 2001:fb6:31c0:3::2/64 scope global
       valid_lft forever preferred_lft forever
    inet6 fe80::9aa7:b0ff:fe00:df20/64 scope link
       valid_lft forever preferred_lft forever
```

Проверка маршрута по умолчанию на R4 с помощью команды `ip -6 route`:

```
[root@elbrus ~]# ip -6 r | grep 2001
2001:fb6:31c0:3::/64 dev eth0 proto kernel metric 256
default via 2001:fb6:31c0:3::1 dev eth0 metric 1024
```

В таблице маршрутизации в дополнении к непосредственно подключенной напрямую сети, появился маршрут по умолчанию на R3, прописанный последней строкой.

4.5. Проверка работы построенной сети

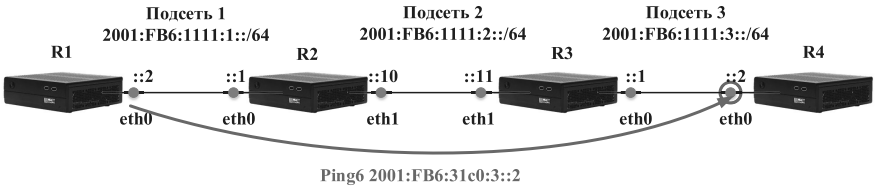


Рисунок 4. Проверка работы сети командой ping6

Проверка работы настроенной сети осуществляется командой `ping6` с маршрутизатора R1 на маршрутизатор R4:

```
    [root@srv ~]# ping6 2001:fb6:31c0:3::2 -c 5
    PING 2001:fb6:31c0:3::2(2001:fb6:31c0:3::2) 56 data bytes
    64 bytes from 2001:fb6:31c0:3::2: icmp_seq=1 ttl=62 time=0.180
ms
    64 bytes from 2001:fb6:31c0:3::2: icmp_seq=2 ttl=62 time=0.190
ms
    64 bytes from 2001:fb6:31c0:3::2: icmp_seq=3 ttl=62 time=0.162
ms
    64 bytes from 2001:fb6:31c0:3::2: icmp_seq=4 ttl=62 time=0.174
ms
    64 bytes from 2001:fb6:31c0:3::2: icmp_seq=5 ttl=62 time=0.216
ms
    --- 2001:fb6:31c0:3::2 ping statistics ---
    5 packets transmitted, 5 received, 0% packet loss, time 3996ms
    rtt min/avg/max/mdev = 0.162/0.184/0.216/0.021 ms
```

Отправленные ICMP пакеты успешно возвращаются, маршрутизатор R4 доступен, настроенная сеть функционирует верно.

Проверить каким путем проходят пакеты между подсетями можно с помощью утилиты traceroute6, предназначенной для работы с IPv6:

```
    [root@elbrus ~]# traceroute6  2001:fb6:31c0:3::2

    traceroute to 2001:fb6:31c0:3::2 (2001:fb6:31c0:3::2), 30 hops
max, 80 byte packets
     1  gateway (2001:fb6:31c0:1::1)  0.148 ms  0.088 ms  0.055 ms
     2  2001:fb6:31c0:2::11           0.196 ms  0.092 ms  0.099 ms
     3  2001:fb6:31c0:3::2            0.187 ms  0.158 ms  0.135 ms
```

Маршрутизатор R1, не зная маршрута до R4, отправляет свои пакеты на шлюз по умолчанию – маршрутизатор R2. Транзитные маршрутизаторы R2 и R3 знают о всех подсетях, и используя свою маршрутную информацию,

перенаправляют пакеты между подсетями. Получив пакеты от R1, R2 отправляет их на R3, а R3 на R4. Так как R4 также не знает о подсети 2001:fb6:31c0:1::/64, в которой находится R1, он отправляет пакеты на свой шлюз по умолчанию – маршрутизатор R3, а тот уже через R2 на R1.

4.6. Утилита mtr6 для диагностики сетевых соединений

Путь прохождения пакетов можно также отследить с помощью утилиты mtr6. Утилита mtr6 является аналогом traceroute6 и в реальном времени показывает потери и задержки пакетов на определенных участках сети:

```
    [root@elbrus]# mtr6 2001:fb6:31c0:3::2
    My traceroute [v0.82]
    elbrus.localdomain (::)                    Wed May 22 16:12:15
2019
Host                          Packets                  Pings
                         loss%  Snt   Last   Avg  Best  Wrst StDev
1. 2001:fb6:31c0:1::1   0.0%    11     0.4     1.1    0.4  1.4     0.3
2. 2001:fb6:31c0:2::11 0.0%   10    2.2    2.3   2.2  2.4    0.1
3. 2001:fb6:31c0:3::2  0.0%   10    2.2    2.3   2.2  2.4    0.1
```

Loss% – процент потерь пакетов;
Snt – количество отправленных пакетов;
Last – время задержки последнего полученного пакета, мс;
Avg – среднее время задержки, мс;
Best – наименьшее зафиксированное время задержки, мс;
Wrst – наибольшее зафиксированное время задержки, мс;
StDev – среднеквадратичное отклонение времени задержки, мс.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Причины перехода к протоколу IPv6. Формат заголовка протокола IPv6;
2. Типы IPv6-адресов, unicast global, link-local, multicast;
3. Для чего используются маски IP-адресов?
4. Назначение link-local адресов IPv6. Протокол NDP;
5. Назначение и содержание таблиц маршрутизации;
6. Применение маршрутов по умолчанию.

Практикум № 18
Протокол динамической маршрутизации RIPng

1. ЦЕЛЬ РАБОТЫ

Изучение особенностей настройки и работы протокола маршрутной информации RIPng (Routing Information Protocol next generation), предназначенного для работы в сетях IPv6, с помощью отечественных программно-аппаратных средств. Изучение пакета quagga, реализующего протоколы маршрутизации, анализ его функционала в рамках настройки IPv6 сети, работающей по протоколу динамической маршрутизации RIPng.

2. ЗАДАНИЕ НА РАБОТУ

В работе используется топология сети, представленная на рисунке 1. Маршрутизаторам R1, R2, R3 и R4 соответствуют рабочие станции под номерами PC1, PC2, PC3 и PC4, соответственно. Адресация в моделируемой сети осуществляется следующим образом:

- Подсеть 1: $2001{:}\,fb6{:}\,31c0{:}\,(N_{min}){:}:/64$ или $2001{:}\,fb6{:}\,31c0{:}\,1{::}/64$
- Подсеть 2: $2001{:}\,fb6{:}\,31c0{:}\,(N_{min}+1){:}:/64$ или $2001{:}\,fb6{:}\,31c0{:}\,2{::}/64$
- Подсеть 3: $2001{:}\,fb6{:}\,31c0{:}\,(N_{min}+2){:}:/64$ или $2001{:}\,fb6{:}\,31c0{:}\,3{::}/64$
- Подсеть 4: $2001{:}\,fb6{:}\,31c0{:}\,(N_{min}+3){:}:/64$ или $2001{:}\,fb6{:}\,31c0{:}\,4{::}/64$

где $N_{min} = 1$ – наименьший из четырех номер рабочих станций.

Последние цифры IPv6-адресов интерфейсов соответствуют номеру машины.

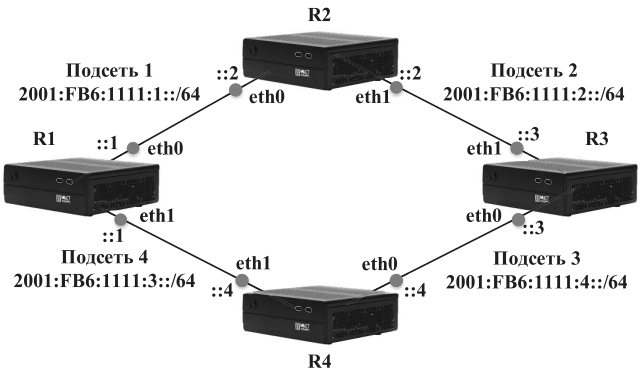


Рисунок 1. Схема лабораторной сети

Основные задачи работы:

- Подготовить машины к работе в сети и установить необходимые пакеты;
- Изменить имя устройств и настроить сетевые интерфейсы, задействованные в работе протокола RIPng с помощью демона zebra;
- С помощью демона ripngd настроить работу протокола RIPng на всех устройствах, настроить анонсирование необходимых подсетей и проверить работу протокола с помощью анализатора трафика Wireshark;
- Проверить отказоустойчивость построенной сети RIPng, смоделировав отказ канала связи между R1 и R2 путем отключения интерфейса eth0 на R1;
- Проверить изменение метрик маршрутов протокола RIPng при перестроении сетевой топологии из-за отключения канала связи между R1 и R2.

3. ТЕРМИНЫ И ИСПОЛЬЗУЕМЫЕ КОМАНДЫ

Термины

Для организации динамической маршрутизации между машинами, работающими под ОС Альт, используется пакет quagga. Пакет представляет собой набор утилит, предназначенных для настройки протоколов динамической маршрутизации в Unix-подобных системах. Quagga поддерживает основные распространенные версии протоколов, такие как: OSPFv2, OSPFv3, RIPv1, RIPv2, RIPng, BGP.

Для обеспечения работы протоколов динамической маршрутизации пакет программ quagga использует специальные сетевые демоны. Демонами в Unix-подобных операционных системах называются компьютерные программы, запускаемые системой при ее старте или самим пользователем из терминала и работающие в фоновом режиме без прямого взаимодействия с ним.

Quagga работает следующим образом: каждый протокол маршрутизации обслуживается собственным демоном маршрутизации (ripngd, ospfd и т.д.), который просчитывает маршруты по своему протоколу и передает результаты управляющему демону zebra. Zebra формирует таблицу маршрутизации и выбирает наилучший маршрут на основании административной дистанции и метрик

Протокол динамической маршрутизации RIPng предназначен для работы в сетях IPv6 и относится к категории протоколов маршрутизации внутреннего шлюза (Interior Gateway Protocol – IGP), которые предназначены для работы внутри автономной системы AS. Автономная система (AS) – это сеть под административным контролем одной организации, сеть каждого интернет провайдера является автономной системой.

Алгоритмы протоколов маршрутизации определяют то, как они решают задачи для изучения всех возможных маршрутов и выбора наилучшего, а также для реакции на изменения в работе сети (конвергенции). Конвергенция (сходимость) – процесс перестроения маршрутов, происходящий при изменении топологии сети, т.е. когда отказывает маршрутизатор или канал связи, либо наоборот – когда они восстанавливаются. Для этих целей протокол RIPng использует дистанционно-векторный алгоритм (алгоритм Беллмана-Форда), выбирая наилучший маршрут к подсетям на основании самой низкой метрики маршрута.

Протокол RIPng для определения метрики маршрута использует счетчик количества маршрутизаторов (транзитных участков) между текущим маршрутизатором и подсетью назначения. Маршрутизатор каждые 30 секунд отправляют обновления RIP на групповой IPv6-адрес FF02::09, используя UDP порт 521, содержащие полную таблицу маршрутизации со всеми известными маршрутами соседним маршрутизаторам.

Таблица маршрутизации содержит адрес сети пункта назначения, кратчайший путь до подсети назначения, отсчитываемый в транзитных участках, следующий маршрутизатор и счетчик транзитных участков.

Когда маршрутизатор получает обновление RIP от соседа, он узнает из него о новых сетях, добавляет эти сети в свою таблицу маршрутизации и отправляет обновление другим маршрутизаторам, увеличивая счетчик переходов для каждой полученной им сети на единицу.

Из-за вероятности возникновения маршрутных петель, максимальная метрика маршрута имеет значение 15, т.е. максимальное число транзитных участков не должно превышать 15. При отказе маршрутов, маршрутизаторы анонсируют отказавший маршрут со специальным значением метрики – бесконечностью. Протокол RIPng определяет бесконечность как значение метрики 16.

Для управления частотой рассылки обновлений RIP и улучшения времени конвергенции протокол RIPng использует специальные таймеры:

- Update timer – таймер отправки маршрутных обновлений RIP (по умолчанию составляет 30 секунд). Каждые 30 секунд процесс RIP на

маршрутизаторе рассылает сообщения другим маршрутизаторам RIPng, содержащие полную таблицу маршрутизации;

- Timeout timer – таймер тайм-аута маршрута (по умолчанию составляет 180 секунд). По истечению таймера, маршрут, по которому не получены обновления в Update сообщениях помечается как не доступный и помечается метрикой 16, но сохраняется в таблице маршрутизации до тех пор, пока не истекает Flush таймер.
- Flush timer – таймер сброса маршрута (по умолчанию равен 120 секунд). По истечении таймера, недоступный маршрут окончательно удаляется из таблицы маршрутизации.

Используемые команды

`apt-get install` <имя пакета> – команда пакетного менеджера Apt, используемого в ОС Альт. Инициирует установку указанных пакетов;

`chown` – команда смены владельца (change owner), когда также необходимо сменить права вложенные файлы и папки, используется ключ -R (рекурсивно);

`nano` – текстовый редактор, также может создавать текстовые файлы;

`systemctl <start/stop>` – команда управления системными сервисами

`netstat` – команда отображения сетевых соединений системы. Основные ключи: -t (TCP-соединения), -u (UDP-соединения), -l (слушающие сокеты), -p (номер процесса в системе), -n (не переводить IP-адреса в доменные имена);

`grep` – утилита для поиска информации;

`ip` – утилита для настройки и управления сетевыми интерфейсами. Для работы с протоколом IPv6 используется ключ -6:

`ip -6 address` – отображение информации о сетевых интерфейсах;

`ip -6 route` – отображение информации о маршрутах;

`telnet` – утилита для подключения к сетевым узлам по протоколу telnet;

`ping6` – утилита для проверки сетевых соединений в сетях IPv6;

`traceroute6` – утилита для трассировки маршрута в сетях IPv6.

Основные команды сетевых демонов quagga

`Show running config` – показать текущую конфигурацию устройства;

`Hostname` <имя> – смена имени устройства;

`Ipv6 forwarding` – режим работы маршрутизатора в сетях IPv6;

`Interface` <имя> – переход к конфигурированию интерфейса;

`Ipv6 address` <IPv6-адрес/маска> – Настройка IPv6-адреса интерфейса;

`Router ripng` – запуск протокола RIPng на устройстве и переход к его конфигурированию;

`Timers basic` <Update> <Timeout> <Flush> – настройка таймеров протокола RIPng

`Network` <интерфейс> – настройка анонсирования сети на определенном интерфейсе

`Show ipv6 route ripng` – показать таблицу маршрутизации RIPng;

`Write memory` – сохранение конфигурации устройства.

4. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

4.1. Установка пакетов и подготовка машин к работе

Для установки пакета quagga, как и любых дополнительных пакетов, необходимы права суперпользователя root, поэтому переходим в режим работы от root командой su- и вводим пароль суперпользователя:

```
[user@elbrus ~]$ su-
Password:
```

Используя пакетный менеджер apt, командой `apt-get install` устанавливаем необходимые пакеты. Кроме пакета quagga, для выполнения лабораторной работы понадобятся пакеты telnet и wireshark. Используем ключ –y, чтобы автоматически подтверждать установку пакетов:

```
[root@elbrus ~]# apt-get install -y quagga telnet wireshark
```

Конфигурационные файлы пакета quagga находятся в директории /etc/quagga/, а логи – в /var/log/quagga/.

Чтобы пакет мог корректно работать, необходимо сменить права на файлы конфигурации и логов, которые он использует. Командой `chown` делаем пользователя quagga и его одноименную группу владельцами файлов:

```
[root@elbrus ~]# chown -R quagga:quagga /etc/quagga/
[root@elbrus ~]# chown -R quagga:quagga /var/log/quagga/
```

После того, как пакеты установлены, необходимо на всех машинах включить форвардинг IPv6 пакетов на уровне ядра системы. Это необходимо для того, чтобы маршрутизаторы могли пересылать пакеты между своими интерфейсами.

Включение форвадинга производится редактированием конфигурационного файла системы sysctl.conf, находящегося в директории /etc/net/. В конфигурационном файле необходимо добавить строку net.ipv6.conf.all.ip_forwarding=1:

```
[root@elbrus ~]# nano /etc/net/sysctl.conf
net.ipv6.conf.all.ip_forwarding=1
```

Сохраняем изменения Ctrl+O и выходим с помощью Ctrl+X.

Также, для запуска демонов, работающих по протоколу IPv6, в системе необходимо включить поддержку IPv6, для этого в файле /etc/sysconfig/network необходимо добавить строку NETWORKING_IPV6=yes

```
[root@elbrus ~]# nano /etc/sysconfig/network
NETWORKING_IPV6=yes
```

Сохраняем изменения Ctrl+O и выходим с помощью Ctrl+X. После настроек перезапускаются сетевые сервисы системы командой `service network restart`:

```
[root@elbrus ~]# service network restart
```

4.2. Подключение к демонам и работа с конфигурационными файлами пакета quagga

Для настройки динамической маршрутизации RIPng нужны управляющий демон zebra и демон протокола RIPng – ripngd:

Запускаем необходимые демоны:

```
[root@elbrus ~]# systemctl start zebra
[root@elbrus ~]# systemctl start ripngd
```

Настраивать сетевые демоны можно как редактируя их конфигурационные файлы, находящиеся в директории /etc/quagga/, так и подключаясь к их консоли по telnet.

После запуска демонов на нашей машине появятся сокеты (IP-адрес + порт), прослушивающие входящие соединения, с помощью которых и осуществляется подключение к консоли демонов.

Проверим появление необходимых сокетов командой `netstat`, так как демоны quagga используют порты, начинающиеся от 2601, то для точного поиска дополняем запрос командой `grep 260`:

```
[root@elbrus ~]# netstat -tlnp | grep 260
Proto      Local Address   Foreign Address   State   PID/Program
name
tcp            127.0.0.1:2601          0.0.0.0:*         LISTEN
3957/zebra
tcp                  ::1:2603               :::*        LISTEN
4032/ripngd
```

На локальном хосте IPv6-демон ripngd по loopback IPv6-адресу слушает порт 2603, а zebra по loopback IPv4-адресу – порт 2601.

Используем telnet для подключения к консоли zebra или ripngd. Указывается IP-адрес и порт необходимого демона:

`[root@elbrus ~]# telnet 127.0.0.1 2601` – подключение к zebra

`[root@elbrus ~]# telnet ::1 2603` – подключение к ripngd

Подключимся к zebra. Вводим пароль zebra (по умолчанию) и попадаем в его консоль:

```
[root@elbrus ~]# telnet 127.0.0.1 2601
Trying 127.0.0.1...
Connected to 127.0.0.1.
User Access Verification
Password:
Router>
```

Для того чтобы начать настройку, сначала нужно войти в привилегированный режим командой `enable`. Привилегированный режим также защищен паролем (по умолчанию zebra):

```
Router> enable
Password:
Router#
```

Чтобы посмотреть текущую конфигурацию маршрутизатора, используется команда `show running-config`, или сокращенно `show run`:

```
Router# show run
```

Вывод команды покажет конфигурацию, записанную в файле /etc/quagga/zebra.conf, рассмотрим ее подробнее:

Имя хоста:

```
hostname Router
```

Зашифрованный пароль для подключения к нашему маршрутизатору:

```
password 8 LrjDz/a2KALVQ
```

Зашифрованный пароль для входа в привилегированный режим enable:

```
enable password 8 LrjDz/a2KALVQ
```

Команда, включающая шифрование паролей при просмотре конфигурации:

```
service password-encryption
```

Путь к файлу с логами:

```
log file /var/log/quagga/zebra.log
```

Список контроля доступа (ACL) с именем localhost, разрешающий доступ к консоли демона zebra только с локального хоста, и запрещающий все остальные соединения:

```
access-list localhost permit 127.0.0.1/32
access-list localhost deny any
```

Команды, указывающие применять список доступа localhost для входящих соединений по виртуальным терминальным линиям vty. Подключаясь по telnet, мы занимаем одну виртуальную терминальную линию vty:

```
line vty
access-class localhost
```

Внесение изменений в конфигурацию устройства производится в режиме глобального конфигурирования (config). Вход в режим осуществляется командой `configure terminal` или сокращённо `conf t`:

```
Router# configure terminal
Router(config)#
```

4.3. Настройка маршрутизатора R1

Для работы протокола RIPng на маршрутизаторе R1 с помощью демона zebra настраиваются все сетевые интерфейсы, задействованные в работе протокола RIPng. Подключаемся к zebra и входим в режим глобального конфигурирования:

```
[root@elbrus ~]# telnet 127.0.0.1 2601
Trying 127.0.0.1...
Connected to 127.0.0.1.
User Access Verification
Password:
Router>
Router> enable
Router#
Router# configure terminal
Router(config)#
```

Изменим имя устройства:

```
Router(config)# hostname R1
```

Далее необходимо включить поддержку передачи пакетов в IPv6-сетях:

```
R1(config)# ipv6 forwarding
```

В процессе работы протокола RIPng будут задействованы два интерфейса маршрутизатора: eth0 и eth1.

Настройка интерфейса eth0:

`R1(config)#interface eth0` – переход к конфигурированию интерфейса;

`R1(config-if)#ipv6 address 2001:fb6:31c0:1::1/64` – присваиваем адрес и маску подсети;

`R1(config-if)#exit` – выход из режима настройки интерфейса eth0.

```
R1(config)#
```

Настройка интерфейса eth1:

```
R1(config)#interface eth1
R1(config-if)#ipv6 address 2001:fb6:31c0:4::1/64
R1(config-if)#exit
R1(config)#
```

Чтобы сохранить все произведенные настройки, нужно сохранить текущую конфигурацию zebra командой `write memory` или сокращенно `wr`:

```
R1(config)# write memory
Configuration saved to /etc/quagga/zebra.conf
R1(config)# exit
```

Вся конфигурация была перезаписана в /etc/quagga/zebra.conf.

Настройка демона zebra завершена, отключиться от демона можно с помощью комбинации Ctrl+D.

Далее на маршрутизаторе R1 необходимо настроить работу протокола RIPng с помощью демона ripngd. Подключаемся к ripngd по telnet:

```
[root@elbrus ~]# telnet ::1 2603
Trying ::1...
Connected to ::1.
User Access Verification
Password:
router-RIPng>
router-RIPng> enable
router-RIPng#
```

Командой `show running-config` можно посмотреть текущую конфигурацию, записанную в /etc/quagga/ripngd.conf. Для внесения изменений в конфигурацию, переходим в режим глобального конфигурирования, используя команду `configure terminal`:

```
router-RIPng# configure terminal
router-RIPng(config)#
```

Изменим имя устройства:

```
router-RIPng(config)# hostname R1-RIPng
```

Теперь необходимо включить процесс RIPng на маршрутизаторе:

`R1-RIPng(config)# router ripng` – включаем RIPng на устройстве и переходим к его конфигурированию;

Далее необходимо прописать все подключенные сети, которые должен анонсировать маршрутизатор R1. Анонс сетей производится путем указания необходимых интерфейсов, находящихся в соответствующих сетях:

```
R1-RIPng(config-router)# network eth0
R1-RIPng(config-router)# network eth1
R1-RIPng(config-router)# exit
```

Для управления частотой рассылки обновлений RIPng, а также для предотвращения маршрутных петель и улучшения времени конвергенции сети, протокол RIPng использует специальные таймеры.

Чтобы увеличить скорость конвергенции сети при изменениях топологии, уменьшим время стандартных таймеров. Таймера Update с 30 до 10 секунд, таймера Timeout с 180 до 60 секунд и таймера Flush с 120 до 30 секунд:

```
R1-RIPng(config)# timers basic 10 60 30
```

Настройка ripngd завершена. Сохраняем конфигурацию и отключаемся от ripngd с помощью Ctrl+D:

```
R1-RIPng(config)# write memory
Configuration saved to /etc/quagga/ripngd.conf
R1-RIPng(config)# exit
```

4.4. Анализ работы протокола RIPng

Проверим настройку IPv6-адресов на маршрутизаторе R1 командой `ip adress` с ключом -6 (для протокола IPv6):

```
[root@elbrus ~]# ip -6 address
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500
qlen 1000
       inet6 2001:fb6:31c0:1::1/64 scope global
          valid_lft forever preferred_lft forever
       inet6 fe80::9aa7:b0ff:fe00:dea0/64 scope link
          valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500
qlen 1000
       inet6 2001:fb6:31c0:4::1/64 scope global
          valid_lft forever preferred_lft forever
       inet6 fe80::9aa7:b0ff:fe00:dea1/64 scope link
          valid_lft forever preferred_lft forever
```

Первая запись: `inet6 2001:fb6:31c0:1::1/64 scope global` – это global unicast адрес, сконфигурированный нами и использующийся для адресации в сетях IPv6.

Вторая запись: `inet6 fe80::9aa7:b0ff:fe00:dea0/64 scope link` – это link-local адрес, назначенный системой автоматически и предназначенный для взаимодействия между устройствами в пределах локального сегмента сети. Link-local адреса используют подсеть FE80::/64, а оставшаяся часть адреса берется из MAC-адреса интерфейса. Данные адреса не маршрутизируются в сети Интернет и нужны для адресации на канале между

двумя устройствами, а также для обеспечения работы необходимых механизмов IPv6.

Как только протокол RIPng настроен на маршрутизаторе, он начинает рассылку специальных Update-сообщений, содержащих маршрутную информацию. Для отправки Update-сообщений в качестве источника пакета, маршрутизаторы используют link-local адреса интерфейсов. Эти сообщения называются RIPng response и по умолчанию рассылаются каждые 30 секунд на специальный групповой IPv6-адрес FF02::9, предназначенный для всех маршрутизаторов, поддерживающих протокол RIPng. Сообщения RIPng response инкапсулируются в протокол UDP и используют порт 521.

Проверим работу протокола RIPng на R1 с помощью анализатора трафика, для этого на интерфейсе eth1 запустим Wireshark и зададим фильтр по протоколу RIPng:

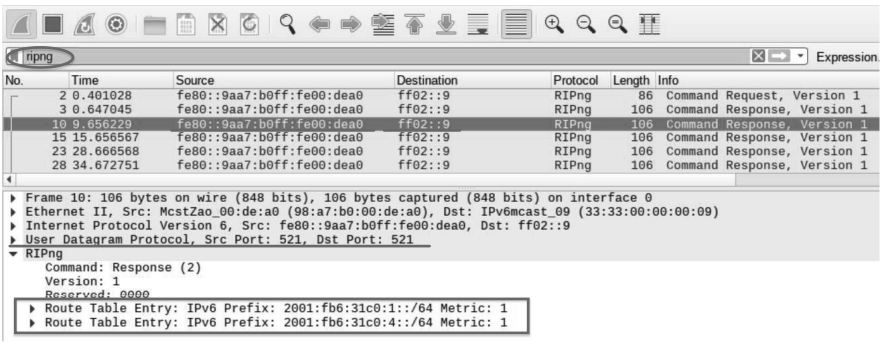


Рисунок 2. Анализ работы протокола RIPng на R1

Анализатор показывает, как с link-local адреса интерфейса eth1 `fe80::9aa7:b0ff:fe00:dea0` отправляется пакет RIPng response на групповой IPv6-адрес FF02::9 по порту 521. Пакеты RIPng инкапсулированы протоколом UDP и содержат маршрутную информацию о состоянии всех известных сетей.

В анализируемом пакете R1 отправляет в сторону R4 маршрутную информацию о известных ему сетях: о сети 2001:fb6:31c0:1::0/64 и о сети 2001:fb6:31c0:4::0/64, так как на данный момент он знает только о сетях, подключенных напрямую к нему, а также указывает соответствующую метрику.

4.5. Настройка маршрутизатора R2

Теперь производится настройка интерфейсов маршрутизатора R2. На R2 запускаем демон zebra, подключаемся к нему по telnet и входим в режим глобального конфигурирования:

```
[root@elbrus ~]# systemctl start zebra
[root@elbrus ~]# telnet 127.0.0.1 2601
Trying 127.0.0.1...
Connected to 127.0.0.1.
User Access Verification
Password:
Router>
Router enable
Router#
Router# configure terminal
Router(config)#
```

Изменим имя устройства:

```
Router(config)# hostname R2
```

Включение поддержки передачи пакетов в IPv6-сетях:

```
 R2(config)# ipv6 forwarding
```

Настройка интерфейса eth0:

```
R2(config)#interface                                eth0
R2(config-if)#    ipv6    address    2001:fb6:31c0:1::2/64
R2(config-if)# exit
R2(config)#
```

Настройка интерфейса eth1:

```
R2(config)#interface                                eth1
R2(config-if)#ipv6      address      2001:fb6:31c0:2::2/64
R2(config-if)#exit
R2(config)#
```

Настройка демона zebra завершена, необходимо сохранить конфигурацию. Отключиться от демона можно с помощью комбинации Ctrl+D.

```
R2(config)# write memory
```

```
Configuration saved to /etc/quagga/zebra.conf
R2(config)# exit
```

Далее на маршрутизаторе R2 необходимо настроить работу протокола RIPng с помощью демона ripngd. Запускаем демон, подключаемся к ripngd по telnet и входим в режим глобального конфигурирования:

```
[root@elbrus ~]# systemctl start ripngd
[root@elbrus ~]# telnet ::1 2603
Trying ::1...
Connected to ::1.
User Access Verification
Password:
router-RIPng>
router-RIPng> enable
router-RIPng#
router-RIPng# configure terminal
router-RIPng(config)#
```

Изменим имя устройства:

```
router-RIPng(config)# hostname R2-RIPng
```

Теперь необходимо включить процесс RIPng и прописать сети, которые будет анонсировать наш маршрутизатор:

```
R2-RIPng(config)# router ripng
```

Пропишем все нужные для анонсирования сети, подключенные к маршрутизатору, путем добавления соответствующих интерфейсов:

```
R2-RIPng(config-router)#          network          eth0
R2-RIPng(config-router)# network eth1
```

Чтобы увеличить скорость конвергенции сети при изменениях топологии, уменьшим время стандартных таймеров: таймера Update с 30 до 10 секунд, таймера Timeout с 180 до 60 секунд и таймера Flush с 120 до 30 секунд:

```
R2-RIPng(config-router)# timers basic 10 60 30
R2-RIPng(config-router)# exit
```

Проверим текущую конфигурацию демона ripngd командой `show run`, если все настроено верно, то основная конфигурация ripngd должна содержать следующие строки:

```
R2-RIPng(config)# show run
!
router ripng
 network eth0
 network eth1
 timers basic 10 60 30
!
```

Настройка демона ripngd завершена, необходимо сохранить конфигурацию:

```
R2-RIPng(config)# write memory
Configuration saved to /etc/quagga/ripngd.conf
R2-RIPng(config)# exit
```

Отключиться от демона можно с помощью комбинации Ctrl+D.

4.6. Настройка маршрутизатора R3

Настройка интерфейсов маршрутизатора R3 производится с помощью демона zebra. На R3 запускаем демон zebra, подключаемся к нему по telnet и входим в режим глобального конфигурирования:

```
[root@elbrus ~]# systemctl start zebra
[root@elbrus ~]# telnet 127.0.0.1 2601
Trying 127.0.0.1...
Connected to 127.0.0.1.
User Access Verification
Password:
Router>
Router> enable
Router#
Router# configure terminal
Router(config)#
```

Изменим имя устройства:

```
Router(config)# hostname R3
```

Включение поддержки передачи пакетов в IPv6-сетях:

```
 R3(config)# ipv6 forwarding
```

Настройка интерфейса eth0:

```
    R3(config)#interface                                eth0
R3(config-if)#ipv6      address      2001:fb6:31c0:3::3/64
R3(config-if)#exit
```

Настройка интерфейса eth1:

```
    R3(config)#interface                                eth1
R3(config-if)#ip       address       2001:fb6:31c0:2::3/64
R3(config-if)#exit
    R3(config)#
```

Настройка демона zebra завершена, необходимо сохранить конфигурацию. Отключиться от демона можно с помощью комбинации Ctrl+D:

```
    R3(config)# write memory
    Configuration saved to /etc/quagga/zebra.conf
    R3(config)# exit
```

Далее на маршрутизаторе R3 необходимо настроить работу протокола RIPng с помощью демона ripngd. Запускаем демон, подключаемся к ripngd по telnet и входим в режим глобального конфигурирования:

```
    [root@elbrus ~]# systemctl start ripngd
    [root@elbrus ~]# telnet ::1 2603
    User Access Verification
    Password:
    router-RIPng>
    router-RIPng> enable
    router-RIPng#
    router-RIPng# configure terminal
    router-RIPng(config)#
```

Изменим имя устройства:

```
    router-RIPng(config)# hostname R3-RIPng
```

Теперь необходимо включить процесс RIPng и прописать сети, которые будет анонсировать маршрутизатор R3:

```
R3-RIPng(config)# router ripng
```

Пропишем все нужные для анонсирования сети, подключенные к маршрутизатору, путем добавления соответствующих интерфейсов:

```
R3-RIPng(config-router)# network eth0
R3-RIPng(config-router)# network eth1
```

Чтобы увеличить скорость конвергенции сети при изменениях топологии, уменьшим время стандартных таймеров: таймера Update с 30 до 10 секунд, таймера Timeout с 180 до 60 секунд и таймера Flush с 120 до 30 секунд:

```
R3-RIPng(config-router)# timers basic 10 60 30
```

Настройка демона ripngd завершена, необходимо сохранить конфигурацию. Отключиться от демона можно с помощью комбинации Ctrl+D.

```
R3-RIPng(config)# write memory
Configuration saved to /etc/quagga/ripngd.conf
R3-RIPng(config)# exit
```

4.7. Настройка маршрутизатора R4

Настройка интерфейсов маршрутизатора R4 производится с помощью демона zebra. На R4 запускаем демон zebra, подключаемся к нему по telnet и входим в режим глобального конфигурирования:

```
[root@elbrus ~]# systemctl start zebra
[root@elbrus ~]# telnet 127.0.0.1 2601
Trying 127.0.0.1...
Connected to 127.0.0.1.
User Access Verification
Password:
Router>
Router> enable
Router#
Router# configure terminal
Router(config)#
```

Изменим имя устройства:

```
Router(config)# hostname R4
```

Включение поддержки передачи пакетов в IPv6-сетях:

```
R4(config)# ipv6 forwarding
```

Настройка интерфейса eth0:

```
R4(config)#interface eth0
R4(config-if)#ipv6 address 2001:fb6:31c0:3::4/64
R4(config-if)#exit
```

Настройка интерфейса eth1:

```
R4(config)#interface eth1
R4(config-if)#ip address 2001:fb6:31c0:4::4/64
R4(config-if)#exit
R4(config)#
```

Настройка демона zebra завершена, необходимо сохранить конфигурацию. Отключиться от демона можно с помощью комбинации Ctrl+D:

```
R4(config)# write memory
Configuration saved to /etc/quagga/zebra.conf
R4(config)# exit
```

Далее на маршрутизаторе R4 необходимо настроить работу протокола RIPng с помощью демона ripngd. Запускаем демон, подключаемся к ripngd по telnet и входим в режим глобального конфигурирования:

```
[root@elbrus ~]# systemctl start ripngd
[root@elbrus ~]# telnet ::1 2603
User Access Verification
Password:
router-RIPng>
router-RIPng> enable
router-RIPng#
router-RIPng# configure terminal
router-RIPng(config)#
```

Изменим имя устройства:

```
router-RIPng(config)# hostname R4-RIPng
```

Теперь необходимо включить процесс RIPng и прописать сети, которые будет анонсировать наш маршрутизатор:

```
R4-RIPng(config)# router ripng
```

Пропишем все нужные для анонсирования сети, подключенные к маршрутизатору, путем добавления соответствующих интерфейсов:

```
R4-RIPng(config-router)#          network          eth0
R4-RIPng(config-router)# network eth1
```

Чтобы увеличить скорость конвергенции сети при изменениях топологии, уменьшим время стандартных таймеров: таймера Update с 30 до 10 секунд, таймера Timeout с 180 до 60 секунд и таймера Flush с 120 до 30 секунд:

```
R4-RIPng(config-router)# timers basic 10 60 30
```

Настройка демона ripngd завершена, необходимо сохранить конфигурацию. Отключиться от демона можно с помощью комбинации Ctrl+D.

```
R4-RIPng(config)# write memory
Configuration saved to /etc/quagga/ripngd.conf
R4-RIPng(config)# exit
```

4.8. Проверка сети, функционирующей по протоколу RIPng

Проверим работу протокола RIPng проверкой доступности сети 2001:fb6:31c0:2::/64 с маршрутизатора R1. Так как данная сеть не подключена напрямую к маршрутизатору R1, то с помощью протокола RIPng он должен получить к ней маршрут через маршрутизатор R2:

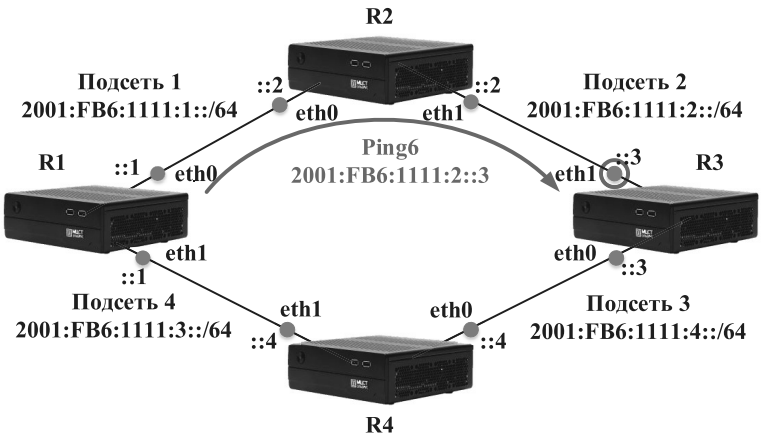


Рисунок 3. Проверка связности настроенной сети

Чтобы проверить работу сети IPv6, используется утилита ping6, работающая по протоколу ICMPv6, который предназначен для работы в сетях IPv6.

На R1 выполняется команда `ping6` и указывается адрес интерфейса eth1 маршрутизатора R3:

```
[root@elbrus ~]# ping6 2001:fb6:31c0:2::3 -c 3
PING 2001:fb6:31c0:2::3(2001:fb6:31c0:2::3) 56 data bytes
64 bytes from 2001:fb6:31c0:2::3: icmp_seq=1 ttl=63 time=0.263
ms
64 bytes from 2001:fb6:31c0:2::3: icmp_seq=2 ttl=63 time=0.151
ms
64 bytes from 2001:fb6:31c0:2::3: icmp_seq=3 ttl=63 time=0.134
ms

--- 2001:fb6:31c0:2::3 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 0.134/0.182/0.263/0.059 ms
```

Ключ –c (count) задает количество посылаемых ICMP запросов. Интерфейс eth1 маршрутизатора R3 доступен, ICMP пакеты успешно возвращаются, значит R1 получил необходимый маршрут и сеть работает корректно.

Проверим таблицу маршрутизации RIPng на R1. Для этого подключимся к zebra и в привилегированном режиме (enable) используем команду `show ipv6 route ripng`:

```
[root@elbrus ~]# telnet 127.0.0.1 2601
R1>
R1> enable
R1# show ipv6 route ripng
Codes: K - kernel route, C - connected, S - static, R -
RIPng, O - OSPFv6, I - IS-IS, B - BGP, A - Babel, > -
selected route, * - FIB route
R>*         2001:fb6:31c0:2::/64          [120/2]         via
fe80::9aa7:b0ff:fe00:dd60, eth0, 00:03:01
R>*         2001:fb6:31c0:3::/64          [120/2]         via
fe80::9aa7:b0ff:fe00:df21, eth1, 00:03:01
```

R1 получил маршруты до сетей 2001:fb6:31c0:2::/64 и 2001:fb6:31c0:3::/64.

Согласно таблице маршрутизации, сеть 2001:fb6:31c0:2::/64 доступна через R2. Проверим это, выполнив с R1 трассировку маршрута утилитой traceroute6 до адреса 2001:fb6:31c0:2::3, находящегося в подсети 2001:fb6:31c0:2::/64:

```
[root@elbrus ~]# traceroute6 2001:fb6:31c0:2::3
traceroute to 2001:fb6:31c0:2::3 (2001:fb6:31c0:2::3), 30
hops max, 80 byte packets
 1  2001:fb6:31c0:1::2 0.133 ms  0.096 ms  0.120 ms
 2  2001:fb6:31c0:2::3 0.106 ms  0.087 ms  0.120 ms
```

Как показывает трассировка, пакеты сначала попадают на адрес 2001:fb6:31c0:1::2 маршрутизатора R2, а потом достигают адреса назначения 2001:fb6:31c0:2::3 на маршрутизаторе R3.

Проверим как протокол RIPng перестраивает маршруты до сетей при изменении сетевой топологии, путем отключения интерфейса eth0 на маршрутизаторе R1. После отключения порта, протокол RIPng перестроит маршрут к подсети 2001:fb6:31c0:2::/64 через маршрутизатор R4.

Как только происходит отключение интерфейса, на маршрутизаторе R1 происходит вытеснение маршрута. Вытеснение маршрута подразумевает анонсирование отказавшего маршрута со специальным значением метрики – бесконечностью. Маршрутизаторы считают маршруты с бесконечной метрикой отказавшими. Протокол RIPng определяет бесконечность как значение 16.

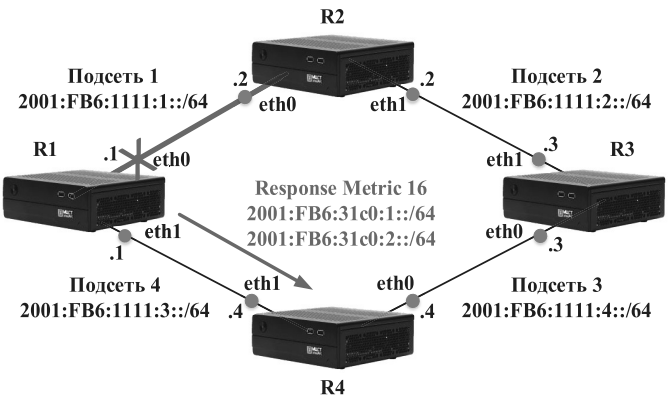


Рисунок 4. Схема сети при отключении интерфейса eth0

Запустим анализатор трафика Wireshark на интерфейсе eth1 маршрутизатора R1, и зададим фильтр по протоколу RIPng. После этого отключаем интерфейс eth0 и с помощью Wireshark смотрим анонсируемую информацию с интерфейса eth1 маршрутизатора R1:

```
[root@elbrus ~]# ip link set dev eth0 down
```

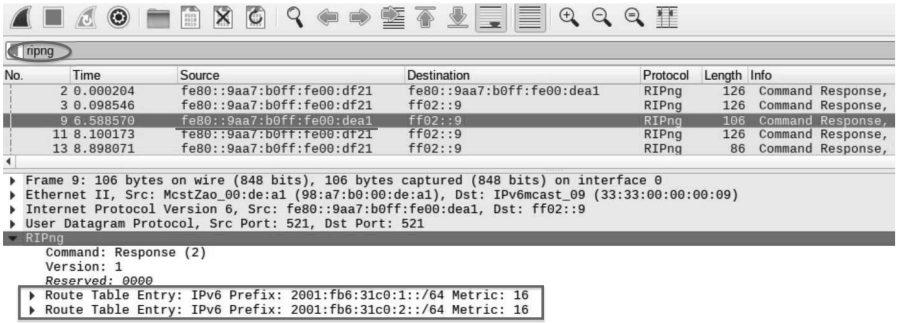


Рисунок 5. Анализ работы протокола RIPng при отключении интерфейса

После отключения интерфейса eth0, маршрутизатор R1 с link-local адреса интерфейса eth1, который имеет IPv6-адрес `fe80::9aa7:b0ff:fe00:dea1`, отправляет пакет с обновлением маршрутной информации, в котором указаны сети 2001:fb6:31c0:1::/64 и 2001:fb6:31c0:2::/64 с метрикой 16, что означает их недоступность. Маршрутизатор R1 анонсирует эти сети, так как до отключения они были доступны через интерфейс eth0. После этого начнется перестроение маршрутов до этих подсетей через маршрутизаторы R4 и R3.

Из-за специфики работы протокола RIPng, маршрут до подсетей 2001:fb6:31c0:1::/64 и 2001:fb6:31c0:2::/64 не будет перестроен сразу, а только после истечения таймера таймаута, так как отправив обновление маршрутов для недоступных сетей с метрикой 16, маршрутизатор R1 будет хранить эти маршруты пока не истечет таймер таймаута (по умолчанию 180 секунд). Поэтому протокол RIPng имеет большое время сходимости, что является одной из причин, по которой он практически не используется в современных сетях. В нашем случае таймер снижен до 60 секунд, по истечении которых маршрут будет перестроен.

После перестроения маршрутов, используем утилиту ping6 с маршрутизатора R1, проверим доступность адреса 2001:fb6:31c0:1::2, который находится в подсети 2001:fb6:31c0:1::/64:

```
    [root@elbrus ~]# ping6 2001:fb6:31c0:1::2 -c 5
    PING 2001:fb6:31c0:1::2(2001:fb6:31c0:1::2) 56 data bytes
    64 bytes from 2001:fb6:31c0:1::2: icmp_seq=1 ttl=62 time=0.178
ms
    64 bytes from 2001:fb6:31c0:1::2: icmp_seq=2 ttl=62 time=0.221
ms
    64 bytes from 2001:fb6:31c0:1::2: icmp_seq=3 ttl=62 time=0.202
ms
    64 bytes from 2001:fb6:31c0:1::2: icmp_seq=4 ttl=62 time=0.197
ms
```

```
    64 bytes from 2001:fb6:31c0:1::2: icmp_seq=5 ttl=62 time=0.200
ms

    --- 2001:fb6:31c0:1::2 ping statistics ---
    5 packets transmitted, 5 received, 0% packet loss, time 3998ms
    rtt min/avg/max/mdev = 0.178/0.199/0.221/0.020 ms
```

Интерфейс eth0 маршрутизатора R2 доступен, ICMP пакеты успешно возвращаются, протокол RIPng перестроил маршрут к сети 2001:fb6:31c0:1::0/64 через маршрутизаторы R4 и R3.

После перестроения маршрута, используя утилиту traceroute6 с маршрутизатора R1 проверим путь трафика до адреса 2001:fb6:31c0:2::2:

```
    [root@elbrus ~]# traceroute6 2001:fb6:31c0:1::2
    traceroute to 2001:fb6:31c0:1::2 (2001:fb6:31c0:1::2), 30
hops max, 80 byte packets
     1   2001:fb6:31c0:4::4          0.193 ms   0.122 ms   0.092
ms
     2   2001:fb6:31c0:3::3          0.193 ms   0.168 ms   0.149
ms
     3   2001:fb6:31c0:1::2          0.263 ms   0.245 ms   0.222
ms
```

Трассировка показывает, что пакеты сначала попадают на узел 2001:fb6:31c0:4::4 (маршрутизатор R4), а потом через маршрутизатор R3 (2001:fb6:31c0:3::3) попадают в нужную подсеть на заданный IPv6-адрес 2001:fb6:31c0:1::2.

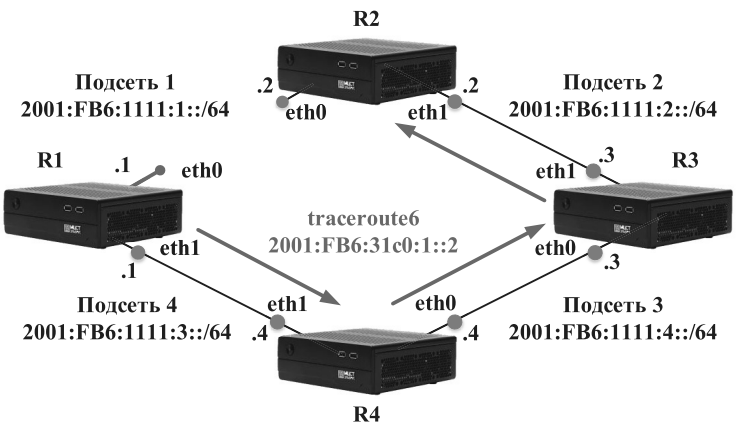


Рисунок 6. Схема прохождения трафика в сети

Когда маршруты перестроены, проверим таблицу маршрутизации RIPng на R1. Для этого подключимся к zebra и в привилегированном режиме (enable) используем команду `show ipv6 route ripng`:

```
R1# show ipv6 route ripng
Codes: K - kernel route, C - connected, S - static, R - RIPng, O - OSPFv6, I - IS-IS, B - BGP, A - Babel, > - selected route, * - FIB route
R>*         2001:fb6:31c0:1::/64         [120/4]         via fe80::9aa7:b0ff:fe00:df21, eth1, 00:01:02
R>*         2001:fb6:31c0:2::/64         [120/3]         via fe80::9aa7:b0ff:fe00:df21, eth1, 00:01:00
R>*         2001:fb6:31c0:3::/64         [120/2]         via fe80::9aa7:b0ff:fe00:df21, eth1, 00:02:32
```

Теперь в таблице маршрутизации указано, что все существующие сети доступны через fe80::9aa7:b0ff:fe00:df21 (маршрутизатор R4) с интерфейса eth1.

Управление передачей трафика в сети происходит с помощью метрик маршрутов, в таблице маршрутизации метрика маршрута указана вторым числом в квадратных скобках, первое число — это административная дистанция протокола маршрутизации.

Административная дистанция используется маршрутизатором для того, чтобы определить какому протоколу маршрутизации доверять больше, если у обоих протоколов есть маршрут до подсети назначения. Например, более современный протокол OSPFv3 имеет административную дистанцию 110, а протокол RIPng – административную дистанцию 120. Таким образом, если в сети используются два и более протокола маршрутизации, у которых есть свои маршруты к сети назначения, то маршрутизатор выберет маршрут по протоколу с наименьшей административной дистанцией. У статических маршрутов административная дистанция равна единице, так как они прописываются вручную, а значит степень доверия к ним выше.

Если же в сети используется один протокол маршрутизации, и у него есть несколько различных маршрутов до подсети назначения, то маршрутизатор выбирает маршрут уже на основании метрик. Чем меньше метрика, тем больше маршрутизатор доверяет этому маршруту.

У разных протоколов маршрутизации метрика рассчитывается по-разному. Например, у протокола OSPFv3 метрика рассчитывается на основании суммарной цены маршрута, которая в свою очередь зависит от ширины полосы пропускания интерфейсов и каналов связи на маршруте до подсети назначения.

А у протокола RIPng метрика рассчитывается исходя из числа транзитных участков между маршрутизатором и подсетью назначения.

На данном рисунке видно, как маршрутизатор R1 получает на интерфейс eth1 анонс маршрутной информации от R4 об известных ему сетях:

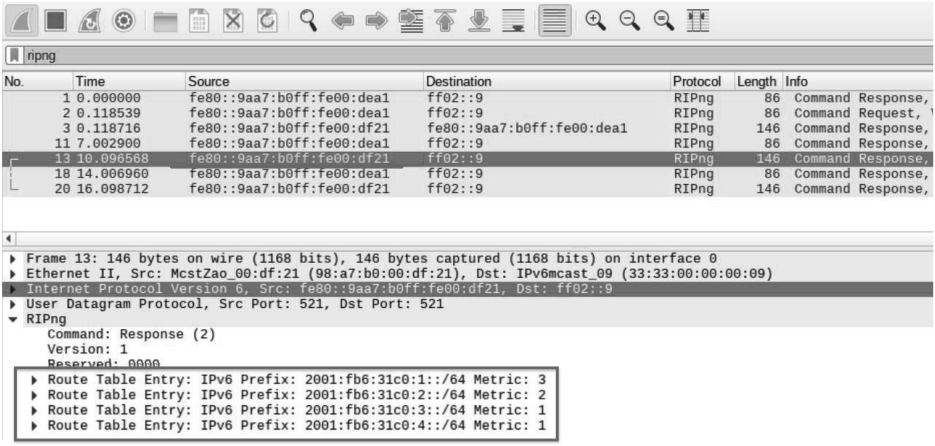


Рисунок 7. Анализ трафика на R1

При этом сеть 2001:fb6:31c0:1::/64 R4 анонсирует с метрикой 3, так как между интерфейсом eth0 маршрутизатора R4 и подсетью 2001:fb6:31c0:1::/64 есть три транзитных участка: первый между R4 и R3, второй между R3 и R2, а третий начинается за интерфейсом eth0 маршрутизатора R2. Сеть 2001:fb6:31c0:2::/64 анонсирована с метрикой 2, т.к. имеет два транзитных участка, а сети 2001:fb6:31c0:3::/64 и 2001:fb6:31c0:4::/64 он анонсирует с метрикой 1, так как они подключены напрямую к маршрутизатору R4.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Основные концепции протокола RIPng. Применение и принцип работы.
2. Каким образом происходит выбор наилучших маршрутов в сети, работающей по протоколу RIPng?
3. Назначение метрики маршрутов и административной дистанции при работе протоколов динамической маршрутизации.
4. Пакеты обновлений RIPng Response, назначение и передаваемая в них информация.
5. Функционирование таймеров протокола RIPng при изменениях в топологии сети.

Практикум № 19
Протокол динамической маршрутизации OSPFv3

1. ЦЕЛЬ РАБОТЫ

Изучение особенностей работы и настройки протокола поиска первого кратчайшего пути OSPFv3 (Open Shortest Path First) версии 3, предназначенного для работы в сетях IPv6, с помощью отечественных программно-аппаратных средств. Изучение пакета quagga, реализующего протоколы маршрутизации, анализ его функционала в рамках настройки IPv6 сети, работающей по протоколу динамической маршрутизации RIPng.

2. ЗАДАНИЕ НА РАБОТУ

В работе используется топология сети, представленная на рисунке 1. Маршрутизаторам R1, R2, R3 и R4 соответствуют рабочие станции под номерами PC1, PC2, PC3 и PC4, соответственно. Адресация в моделируемой сети осуществляется следующим образом:

- Подсеть 1: $2001{:}\,fb6{:}\,31c0{:}\,(N_{min}){:}:/64$ или $2001{:}\,fb6{:}\,31c0{:}\,1{:}:/64$
- Подсеть 2: $2001{:}\,fb6{:}\,31c0{:}\,(N_{min}+1){:}:/64$ или $2001{:}\,fb6{:}\,31c0{:}\,2{:}:/64$
- Подсеть 3: $2001{:}\,fb6{:}\,31c0{:}\,(N_{min}+2){:}:/64$ или $2001{:}\,fb6{:}\,31c0{:}\,3{:}:/64$
- Подсеть 4: $2001{:}\,fb6{:}\,31c0{:}\,(N_{min}+3){:}:/64$ или $2001{:}\,fb6{:}\,31c0{:}\,4{:}:/64$

где $N_{min} = 1$ – наименьший из четырех номер рабочих станций.

Последние цифры IPv6-адресов интерфейсов соответствуют номеру машины.

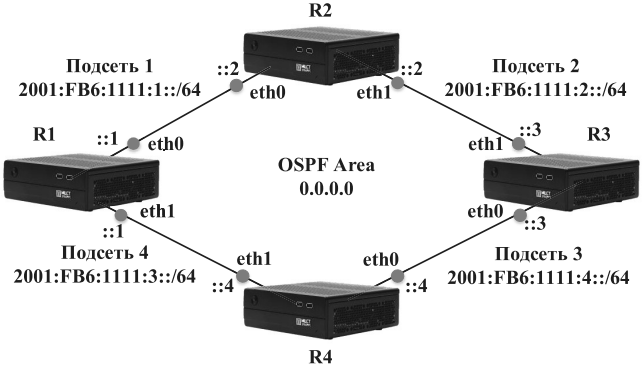


Рисунок 1. Схема сети.

Основные задачи работы:

- Подготовить машины к работе в сети IPv6 и установить необходимые пакеты;
- Изменить имя устройств и настроить сетевые интерфейсы, задействованные в работе протокола OSPFv3 с помощью демона zebra;
- С помощью демона ospf6d настроить работу протокола OSPFv3 на всех устройствах, настроить анонсирование необходимых подсетей и проверить работу протокола с помощью анализатора трафика Wireshark;
- Проверить отказоустойчивость построенной сети OSPFv3, смоделировав отказ канала связи между R1 и R2 путем отключения интерфейса eth0 на R1;
- Проверить работу протокола OSPFv3 при изменении полосы пропускания каналов связи между маршрутизаторами R1, R2 и R4.

3. ТЕРМИНЫ И ИСПОЛЬЗУЕМЫЕ КОМАНДЫ

Термины

Для организации динамической маршрутизации между машинами, работающими под ОС Альт, используется пакет quagga. Пакет представляет собой набор утилит, предназначенных для настройки протоколов динамической маршрутизации в Unix-подобных системах. Quagga поддерживает основные распространенные версии протоколов, такие как: OSPFv2, OSPFv3, RIPv1, RIPv2, RIPng, BGP.

Для обеспечения работы протоколов динамической маршрутизации пакет программ quagga использует специальные сетевые демоны. Демонами в Unix-подобных операционных системах называются компьютерные программы, запускаемые системой при ее старте или самим пользователем из терминала и работающие в фоновом режиме без прямого взаимодействия с ним.

Quagga работает следующим образом: каждый протокол маршрутизации обслуживается собственным демоном маршрутизации (ospf6d, ripngd и т.д.), который просчитывает маршруты по своему протоколу и передает результаты управляющему демону zebra. Zebra формирует таблицу маршрутизации и выбирает наилучший маршрут на основании административной дистанции и метрик

Протокол динамической маршрутизации OSPFv3 предназначен для работы в сетях IPv6 и относится к категории протоколов маршрутизации внутреннего шлюза (Interior Gateway Protocol – IGP), которые предназначены для работы внутри автономной системы AS. Автономная система (AS) – это сеть под административным контролем одной организации, сеть каждого интернет провайдера является автономной системой.

Алгоритмы протоколов маршрутизации определяют то, как они решают задачи для изучения всех возможных маршрутов и выбора наилучшего, а также для реакции на изменения в работе сети (конвергенции). Конвергенция (сходимость) – процесс перестроения маршрутов, происходящий при изменении топологии сети, т.е. когда отказывает маршрутизатор или канал связи, либо наоборот – когда они восстанавливаются. Для этих целей протокол OSPFv3 использует алгоритм состояния канала, выбирая наилучший маршрут к подсетям на основании самой низкой метрики маршрута.

Протокол OSPFv3 для определения метрики подсчитывает цену каждого интерфейса на всем маршруте до сети назначения, а также цену на основании ширины полосы пропускания канала связи и с помощью математического алгоритма Дейкстры выбирает наилучший маршрут.

Протокол OSPFv3 использует концепцию соседей. Соседские отношения позволяют соседним маршрутизаторам обмениваться анонсами состояния каналов LSA (Link State Advertisement), которые содержат изменения в топологии сети. Получая анонсы, маршрутизатор формирует у себя базу данных состояния каналов LSDB (Link State Date Base). Модель соседей OSPF также обеспечивает новым маршрутизаторам динамическое обнаружение, что увеличивает масштабируемость сети.

Для динамического поиска соседей маршрутизаторы OSPF используют пакеты OSPF Hello. Пакеты Hello инкапсулируются заголовком IP и посылаются на групповой IPv6-адрес FF02::5, предназначенный для всех маршрутизаторов, работающих по протоколу OSPFv3. Получая пакеты Hello, маршрутизаторы OSPF узнают из них о новых соседях.

Работа сети OSPFv3 организована с помощью зон (OSPF area). Разделение на зоны в протоколе OSPFv3 позволяет снизить нагрузку на маршрутизаторы и оптимизировать работу сети. Если сеть слишком велика и находится целиком в одной области, то маршрутизатор должен будет хранить большую топологическую базу LSDB, занимающую много оперативной памяти.

Протокол OSPFv3 работает с помощью алгоритма Дейкстры, любое изменение состояния интерфейса маршрутизатора запускает этот алгоритм. Обработка больших топологических баз алгоритмом Дейкстры занимает

больше процессорного времени, а загрузка процессора растет экспоненциально при увеличении размера топологической базы.

При разделении на зоны маршрутизатор будет просчитывать топологию только для своей зоны, также значительно уменьшится количество группового трафика, создаваемого пакетами OSPF Hello, рассылка которых будет ограничена границами зоны.

Протокол OSPFv3 использует в работе как глобальные IPv6 адреса, так и специальные Link-local адреса для рассылки пакетов OSPF Hello. Данные адреса не маршрутизируются в сети Интернет и нужны для адресации на канале между двумя устройствами, а также для обеспечения работы необходимых механизмов IPv6.

Используемые команды

`apt-get install` <имя пакета> – команда пакетного менеджера Apt, используемого в ОС Альт. Инициирует установку указанных пакетов;

`chown` – команда смены владельца (change owner), когда также необходимо сменить права вложенные файлы и папки, используется ключ -R (рекурсивно);

`nano` – текстовый редактор, также может создавать текстовые файлы;

`systemctl <start/stop>` – команда управления системными сервисами

`netstat` – команда отображения сетевых соединений системы. Основные ключи: -t (TCP-соединения), -u (UDP-соединения), -l (слушающие сокеты), -p (номер процесса в системе), -n (не переводить IP-адреса в доменные имена);

`grep` – утилита для поиска информации;

`ip` – утилита для настройки и управления сетевыми интерфейсами. Для работы с протоколом IPv6 используется ключ -6:

`ip -6 address` – отображение информации о сетевых интерфейсах;

`ip -6 route` – отображение информации о маршрутах;

`telnet` – утилита для подключения к сетевым узлам по протоколу telnet;

`ping6` – утилита для проверки сетевых соединений в сетях IPv6;

`traceroute6` – утилита для трассировки маршрута в сетях IPv6.

Основные команды сетевых демонов quagga

`Show running config` – показать текущую конфигурацию устройства;

`Hostname` <имя> – смена имени устройства;

`Ipv6 forwarding` – режим работы маршрутизатора в сетях IPv6;

`Interface` <имя> – переход к конфигурированию интерфейса;

`Ipv6 address` <IPv6-адрес/маска> – Настройка IPv6-адреса интерфейса;

`Bandwidth` <1-10000000 кбит/с> – настройка скорости интерфейса;

`Router ospf6` – запуск протокола OSPFv3 на устройстве и переход к его конфигурированию;

`Router-id` <x.x.x.x> – настройка идентификатора маршрутизатора OSPF;

`Network` <интерфейс> `area` <x.x.x.x> – настройка анонсирования сети на определенном интерфейсе и присвоение ей конкретной зоны;

`Show ipv6 route ospf6` – показать таблицу маршрутизации OSPFv3;

`Show ipv6 ospf6 neighbor` – информация о соседях OSPF;

`Write memory` – сохранение конфигурации устройства.

4. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

4.1. Установка пакетов и подготовка машин к работе

Для установки пакета quagga, как и любых дополнительных пакетов, необходимы права суперпользователя root, поэтому переходим в режим работы от root командой su- и вводим пароль суперпользователя:

```
[user@elbrus ~]$ su-
Password:
```

Используя пакетный менеджер apt, командой `apt-get install` устанавливаем необходимые пакеты. Кроме пакета quagga, для выполнения лабораторной работы понадобятся пакеты telnet и wireshark. Используем ключ –y, чтобы автоматически подтверждать установку пакетов:

```
[root@elbrus ~]# apt-get install -y quagga telnet wireshark
```

Конфигурационные файлы пакета quagga находятся в директории /etc/quagga/, а логи – в /var/log/quagga/.

Чтобы пакет мог корректно работать, необходимо сменить права на файлы конфигурации и логов, которые он использует. Командой `chown` делаем пользователя quagga и его одноименную группу владельцами файлов:

```
[root@elbrus ~]# chown -R quagga:quagga /etc/quagga/
[root@elbrus ~]# chown -R quagga:quagga
/var/log/quagga/
```

После того, как пакеты установлены, необходимо на всех машинах включить форвардинг IPv6 пакетов на уровне ядра системы. Это необходимо для того, чтобы маршрутизаторы могли пересылать пакеты между своими интерфейсами.

Включение форвадинга производится редактированием конфигурационного файла системы sysctl.conf, находящегося в директории /etc/net/. В конфигурационном файле необходимо добавить строку net.ipv6.conf.all.ip_forwarding=1:

```
[root@elbrus ~]# nano /etc/net/sysctl.conf
net.ipv6.conf.all.ip_forwarding=1
```

Сохраняем изменения Ctrl+O и выходим с помощью Ctrl+X.

Также, для запуска демонов, работающих по протоколу IPv6, в системе необходимо включить поддержку IPv6, для этого в файле /etc/sysconfig/network необходимо добавить строку NETWORKING_IPV6=yes

```
[root@elbrus ~]# nano /etc/sysconfig/network
NETWORKING_IPV6=yes
```

Сохраняем изменения Ctrl+O и выходим с помощью Ctrl+X. После настроек перезапускаются сетевые сервисы системы командой `service network restart`:

```
[root@elbrus ~]# service network restart
```

4.2. Подключение к демонам и работа с конфигурационными файлами пакета quagga

Для настройки динамической маршрутизации OSPFv3 нужны управляющий демон zebra и демон протокола OSPFv3 – ospf6d:

Запускаем необходимые демоны:

```
[root@elbrus ~]# systemctl start zebra
[root@elbrus ~]# systemctl start ospf6d
```

Настраивать сетевые демоны можно как редактируя их конфигурационные файлы, находящиеся в директории /etc/quagga/, так и подключаясь к их консоли по telnet.

После запуска демонов на нашей машине появятся сокеты (IP-адрес + порт), прослушивающие входящие соединения, с помощью которых и осуществляется подключение к консоли демонов.

Проверим появление необходимых сокетов командой `netstat`, так как демоны quagga используют порты, начинающиеся от 2601, то для точного поиска дополняем запрос командой `grep 260:`

```
[root@elbrus ~]# netstat -tlnp | grep 260
Proto      Local Address   Foreign Address   State   PID/Program
name
tcp             127.0.0.1:2601              0.0.0.0:*      LISTEN
4026/zebra
tcp                    ::1:2606                   :::*       LISTEN
4113/ospf6d
```

На локальном хосте IPv6-демон ospf6d по loopback IPv6-адресу слушает порт 2606, а zebra по loopback IPv4-адресу – порт 2601.

Используем telnet для подключения к консоли zebra или ospf6d. Указывается IP-адрес и порт необходимого демона:

`[root@elbrus ~]# telnet 127.0.0.1 2601` – подключение к zebra

`[root@elbrus ~]# telnet ::1 2606` – подключение к ospf6d

Подключимся к zebra. Вводим пароль zebra (по умолчанию) и попадаем в его консоль:

```
[root@elbrus ~]# telnet 127.0.0.1 2601
Trying 127.0.0.1...
Connected to 127.0.0.1.
User Access Verification
Password:
Router>
```

Для того чтобы начать настройку, сначала нужно войти в привилегированный режим командой `enable.` Привилегированный режим также защищен паролем (по умолчанию zebra):

```
Router> enable
Password:
```

```
Router#
```

Чтобы посмотреть текущую конфигурацию маршрутизатора, используется команда `show running-config`, или сокращенно `show run`:

```
Router# show run
```

Вывод команды покажет конфигурацию, записанную в файле /etc/quagga/zebra.conf, рассмотрим ее подробнее:

Имя хоста:

```
hostname Router
```

Зашифрованный пароль для подключения к нашему маршрутизатору:

```
password 8 LrjDz/a2KALVQ
```

Зашифрованный пароль для входа в привилегированный режим enable:

```
enable password 8 LrjDz/a2KALVQ
```

Команда, включающая шифрование паролей при просмотре конфигурации:

```
service password-encryption
```

Путь к файлу с логами:

```
log file /var/log/quagga/zebra.log
```

Список контроля доступа (ACL) с именем localhost, разрешающий доступ к консоли демона zebra только с локального хоста, и запрещающий все остальные соединения:

```
access-list localhost permit 127.0.0.1/32
access-list localhost deny any
```

Команды, указывающие применять список доступа localhost для входящих соединений по виртуальным терминальным линиям vty. Подключаясь по telnet, мы занимаем одну виртуальную терминальную линию vty:

```
line vty
access-class localhost
```

Внесение изменений в конфигурацию устройства производится в режиме глобального конфигурирования (config). Вход в режим осуществляется командой `configure terminal` или сокращённо `conf t`:

```
Router# configure terminal
Router(config)#
```

4.3. Настройка маршрутизатора R1

Для работы протокола OSPFv3 на маршрутизаторе R1 с помощью демона zebra настраиваются все сетевые интерфейсы, задействованные в работе протокола OSPFv3. Подключаемся к zebra и входим в режим глобального конфигурирования:

```
[root@elbrus ~]# telnet 127.0.0.1 2601
Trying 127.0.0.1...
Connected to 127.0.0.1.
User Access Verification
Password:
Router>
Router> enable
Router#
Router# configure terminal
Router(config)#
```

Изменим имя устройства:

```
Router(config)# hostname R1
```

Далее необходимо включить поддержку передачи пакетов в IPv6-сетях:

```
R1(config)# ipv6 forwarding
```

В процессе работы протокола OSPFv3 будут задействованы два интерфейса маршрутизатора: eth0 и eth1.

Настройка интерфейса eth0:

`R1(config)#interface eth0` – переход к конфигурированию интерфейса;

`R1(config-if)#ipv6 address 2001:fb6:31c0:1::1/64` – присваиваем адрес и маску подсети;

`R1(config-if)# bandwidth 100000` – задается скорость работы интерфейса в кбит/с (100 Мбит/с). Команда необходима для работы механизмов

QoS (Quality of Service) и не меняет реальную физическую скорость интерфейса, а лишь заставляет маршрутизатор думать, что скорость является такой.

```
R1(config-if)#exit – выход из режима настройки интерфейса eth0.
R1(config)#
```

Настройка интерфейса eth1:

```
R1(config)#interface eth1
R1(config-if)#ipv6 address 2001:fb6:31c0:4::1/64
R1(config-if)# bandwidth 100000
R1(config-if)#exit
R1(config)#
```

Чтобы сохранить все произведенные настройки, нужно сохранить текущую конфигурацию zebra командой `write memory` или сокращенно `wr`:

```
R1(config)# write memory
Configuration saved to /etc/quagga/zebra.conf
R1(config)# exit
```

Вся конфигурация была перезаписана в /etc/quagga/zebra.conf.

Настройка демона zebra завершена, отключиться от демона можно с помощью комбинации Ctrl+D.

Далее на маршрутизаторе R1 необходимо настроить работу протокола OSPFv3 с помощью демона ospf6d. Подключаемся к ospf6d по telnet:

```
[root@elbrus ~]# telnet ::1 2606
Trying ::1...
Connected to ::1.
User Access Verification
Password:
router-OSPF6>
router-OSPF6> enable
router-OSPF6#
```

Командой `show running-config` можно посмотреть текущую конфигурацию, записанную в /etc/quagga/ospf6d.conf. Для внесения изменений в конфигурацию, переходим в режим глобального конфигурирования, используя команду `configure terminal`:

```
router-OSPF6# configure terminal
router-OSPF6(config)#
```

Изменим имя устройства:

```
router-OSPF6(config)# hostname R1-OSPF6
```

Теперь необходимо включить процесс OSPFv3 на маршрутизаторе:

`R1-OSPF6(config)# router ospf6` – включаем OSPFv3 на устройстве и переходим к его конфигурированию;

Для правильной работы у маршрутизатора OSPF должен быть задан идентификатор маршрутизатора Router ID (RID).

Изначально, при запуске процесса OSPF, маршрутизаторы в качестве Router ID используют самый большой IP-адрес физического интерфейса. Оставлять настройку по умолчанию не рекомендуется, так как в случае отключения физического интерфейса нарушится и процесс OSPF, идентификатор которого был привязан к IP-адресу физического интерфейса, это может привести к нарушению работы протокола OSPF на маршрутизаторе.

Router ID назначается следующим образом в порядке приоритета:

- с помощью команды `router-id`. Если команда задана, то маршрутизатор использует ее значение;
- если на каком-либо loopback-интерфейсе маршрутизатора сконфигурирован IP-адрес и этот интерфейс в состоянии up, то маршрутизатор выбирает loopback-интерфейс с самым большим IP-адресом;
- маршрутизатор сам выбирает самый большой IP-адрес из имеющихся физических интерфейсов.

Таким образом, лучше назначать RID вручную командой `router-id` или использовать логические loopback-интерфейсы, т.к. они не полагаются на аппаратные средства, и переходят в рабочее состояние сразу при запуске устройства.

Зададим Router ID с помощью команды `router-id`:

```
R1-OSPF6(config-router)# router-id 1.1.1.1
```

Далее необходимо прописать все сети, которые будет анонсировать маршрутизатор по OSPF. При настройке сетей в OSPFv3 необходимо указывать к какой зоне `area` они принадлежат.

В лабораторной работе будет использоваться одна нулевая зона, называемая магистральной (backbone area). Если в сети существует несколько зон, все они должны быть подсоединены к магистральной зоне.

Пропишем все сети, подключенные к R1, которые необходимо анонсировать, и определим их в нулевую зону. Анонс сетей производится путем указания необходимых интерфейсов, находящихся в соответствующих сетях:

```
    R1-OSPF6(config-router)# network eth0 area 0.0.0.0
R1-OSPF6(config-router)# network eth1 area 0.0.0.0
    R1-OSPF6(config-router)# exit
```

Настройка демона ospf6d завершена. Сохраняем конфигурацию и отключаемся от ospf6d с помощью Ctrl+D.

```
    R1-OSPF6(config)# write memory
    Configuration saved to /etc/quagga/ospf6d.conf
    R1-OSPF6(config)# exit
```

4.4. Проверка работы протокола OSPFv3

Проверим настройку IPv6-адресов на маршрутизаторе R1 командой ip adress с ключом -6 (для протокола IPv6):

```
    [root@elbrus ~]# ip -6 address
    2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500
qlen 1000
        inet6 2001:fb6:31c0:1::1/64 scope global
           valid_lft forever preferred_lft forever
        inet6 fe80::9aa7:b0ff:fe00:df00/64 scope link
           valid_lft forever preferred_lft forever
    3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500
qlen 1000
        inet6 2001:fb6:31c0:4::1/64 scope global
           valid_lft forever preferred_lft forever
        inet6 fe80::9aa7:b0ff:fe00:df01/64 scope link
           valid_lft forever preferred_lft forever
```

Первая запись: `inet6 2001:fb6:31c0:1::1/64 scope global` – это global unicast адрес, сконфигурированный нами и использующийся для адресации в сетях IPv6.

Вторая запись: `inet6 fe80::9aa7:b0ff:fe00:df00/64 scope link` – это link-local адрес, назначенный системой автоматически и предназначенный для взаимодействия между устройствами в пределах локального сегмента сети. Link-local адреса используют подсеть FE80::/64, а оставшаяся часть адреса берется из MAC-адреса интерфейса. Данные адреса не маршрутизируются в сети Интернет и нужны для адресации на канале между двумя устройствами, а также для обеспечения работы необходимых механизмов IPv6.

Как только маршрутизатор настроен, он начинает поиск соседей, рассылая пакеты «Hello» на групповой IPv6-адрес FF02::5, предназначенный для всех маршрутизаторов, поддерживающих протокол OSPFv3. Для отправки пакетов Hello в качестве источника пакета, маршрутизаторы используют link-local адреса интерфейсов. Маршрутизаторы OSPFv3 получают пакеты Hello, посылаемые на этот IPv6-адрес и узнают из них о других соседях в сети.

Проверим работу протокола OSPFv3 на R1 с помощью анализатора трафика, для этого на интерфейсе eth1 запустим Wireshark и зададим фильтр по протоколу OSPF:

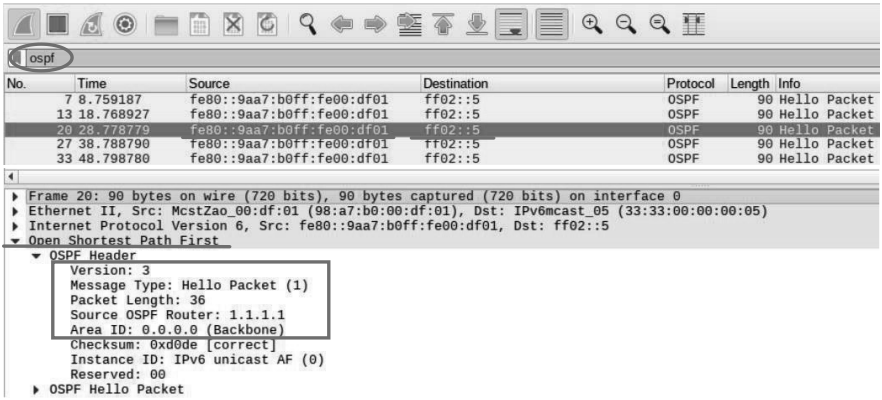


Рисунок 2. Анализ работы протокола OSPFv3 на R1

Анализатор трафика показывает, как маршрутизатор R1 с Анализатор показывает, как с link-local адреса интерфейса eth1, который имеет адрес `fe80::9aa7:b0ff:fe00:df01`, отправляет пакеты OSPFv3 Hello на групповой IPv6-адрес FF02::5. Заголовок пакета OSPFv3 Hello содержит основные параметры маршрутизатора, а именно: версию протокола OSPF (Version 3, версия OSPF для протокола IPv6), тип пакета (Hello Packet), Router ID маршрутизатора (Source OSPF Router) и идентификатор зоны (Area ID).

Hello пакеты используются для обнаружения соседей, а также выступают в роли keepalive-пакетов для проверки доступности, и отправляются по умолчанию каждые 10 секунд. Пакет несет в себе параметры, о которых маршрутизаторы OSPFv3 должны договориться перед тем, как становиться соседями.

4.5. Настройка маршрутизатора R2

Теперь производится настройка интерфейсов маршрутизатора R2. На R2 запускаем демон zebra, подключаемся к нему по telnet и входим в режим глобального конфигурирования:

```
[root@elbrus ~]# systemctl start zebra
[root@elbrus ~]# telnet 127.0.0.1 2601
Trying 127.0.0.1...
Connected to 127.0.0.1.
User Access Verification
Password:
Router>
Router enable
Router#
Router# configure terminal
Router(config)#
```

Изменим имя устройства:

```
Router(config)# hostname R2
```

Включение поддержки передачи пакетов в IPv6-сетях:

```
 R2(config)# ipv6 forwarding
```

Настройка интерфейса eth0:

```
R2(config)#interface                                eth0
R2(config-if)# ipv6 address 2001:fb6:31c0:1::2/64
R2(config-if)#            bandwidth            100000
R2(config-if)# exit
R2(config)#
```

Настройка интерфейса eth1:

```
R2(config)#interface                                eth1
R2(config-if)#ipv6 address 2001:fb6:31c0:2::2/64
```

```
    R2(config-if)#              bandwidth              100000
R2(config-if)#exit
```

Настройка демона zebra завершена, необходимо сохранить конфигурацию. Отключиться от демона можно с помощью комбинации Ctrl+D.

```
    R2(config)# write memory
    Configuration saved to /etc/quagga/zebra.conf
    R2(config)# exit
```

Далее на маршрутизаторе R2 необходимо настроить работу протокола OSPFv3 с помощью демона ospf6d. Запускаем демон, подключаемся к ospf6d по telnet и входим в режим глобального конфигурирования:

```
    [root@elbrus ~]# systemctl start ospf6d
    [root@elbrus ~]# telnet ::1 2606
    Trying ::1...
    Connected to ::1.
    User Access Verification
    Password:
    router-OSPF6>
    router-OSPF6> enable
    router-OSPF6#
    router-OSPF6# configure terminal
    router-OSPF6(config)#
```

Изменим имя устройства:

```
    router-OSPF6(config)# hostname R2-OSPF6
```

Теперь необходимо включить процесс OSPFv3, задать идентификатор Router ID и прописать сети, которые будет анонсировать наш маршрутизатор:

```
    R2-OSPF6(config)# router ospf6
    R2-OSPF6(config-router)# router-id 2.2.2.2
```

Пропишем все нужные для анонсирования сети, подключенные к маршрутизатору, путем добавления соответствующих интерфейсов и определим их в нулевую зону:

```
    R2-OSPF6(config-router)#  network  eth0  area  0.0.0.0
R2-OSPF6(config-router)# network eth1 area 0.0.0.0
```

Проверим текущую конфигурацию демона ospf6d командой `show run`, если все настроено верно, то основная конфигурация ospf6d должна содержать следующие строки:

```
R2-OSPF6(config)# show run
!
router ospf6
 router-id 2.2.2.2
 interface eth0 area 0.0.0.0
 interface eth1 area 0.0.0.0
!
```

Настройка демона ospf6d завершена, необходимо сохранить конфигурацию:

```
R2-OSPF6(config)# write memory
Configuration saved to /etc/quagga/ospf6d.conf
R2-OSPF6(config)# exit
```

Отключиться от демона можно с помощью комбинации Ctrl+D.

4.6. Настройка маршрутизатора R3

Настройка интерфейсов маршрутизатора R3 производится с помощью демона zebra. На R3 запускаем демон zebra, подключаемся к нему по telnet и входим в режим глобального конфигурирования:

```
[root@elbrus ~]# systemctl start zebra
[root@elbrus ~]# telnet 127.0.0.1 2601
Trying 127.0.0.1...
Connected to 127.0.0.1.
User Access Verification
Password:
Router>
Router> enable
Router#
Router# configure terminal
Router(config)#
```

Изменим имя устройства:

```
Router(config)# hostname R3
```

Включение поддержки передачи пакетов в IPv6-сетях:

```
 R3(config)# ipv6 forwarding
```

Настройка интерфейса eth0:

```
    R3(config)#interface                                  eth0
R3(config-if)#ipv6 address 2001:fb6:31c0:3::3/64
    R3(config-if)#             bandwidth             100000
R3(config-if)#exit
```

Настройка интерфейса eth1:

```
    R3(config)#interface                                  eth1
R3(config-if)#ip address 2001:fb6:31c0:2::3/64
    R3(config-if)#             bandwidth             100000
R3(config-if)#exit
    R3(config)#
```

Настройка демона zebra завершена, необходимо сохранить конфигурацию. Отключиться от демона можно с помощью комбинации Ctrl+D:

```
    R3(config)# write memory
    Configuration saved to /etc/quagga/zebra.conf
    R3(config)# exit
```

Далее на маршрутизаторе R3 необходимо настроить работу протокола OSPFv3 с помощью демона ospf6d. Запускаем демон, подключаемся к ospf6d по telnet и входим в режим глобального конфигурирования:

```
    [root@elbrus ~]# systemctl start ospf6d
    [root@elbrus ~]# telnet ::1 2606
    User Access Verification
    Password:
    router-OSPF6>
    router-OSPF6> enable
    router-OSPF6#
    router-OSPF6# configure terminal
    router-OSPF6(config)#
```

Изменим имя устройства:

```
    router-OSPF6(config)# hostname R3-OSPF6
```

Теперь необходимо включить процесс OSPFv3 и прописать сети, которые будет анонсировать маршрутизатор R3:

```
R3-OSPF6(config)# router ospf6
```

Теперь необходимо включить процесс OSPFv3, задать идентификатор Router ID и прописать сети, которые будет анонсировать наш маршрутизатор:

```
R3-OSPF6(config)# router ospf6
R3-OSPF6(config-router)# router-id 3.3.3.3
```

Пропишем все нужные для анонсирования сети, подключенные к маршрутизатору, путем добавления соответствующих интерфейсов и определим их в нулевую зону:

```
R3-OSPF6(config-router)# network eth0 area 0.0.0.0
R3-OSPF6(config-router)# network eth1 area 0.0.0.0
```

Настройка демона ospf6d завершена, необходимо сохранить конфигурацию. Отключиться от демона можно с помощью комбинации Ctrl+D.

```
R3-OSPF6(config)# write memory
Configuration saved to /etc/quagga/ospf6d.conf
R3-OSPF6(config)# exit
```

4.7. Настройка маршрутизатора R4

Настройка интерфейсов маршрутизатора R4 производится с помощью демона zebra. На R4 запускаем демон zebra, подключаемся к нему по telnet и входим в режим глобального конфигурирования:

```
[root@elbrus ~]# systemctl start zebra
[root@elbrus ~]# telnet 127.0.0.1 2601
Trying 127.0.0.1...
Connected to 127.0.0.1.
User Access Verification
Password:
Router>
Router> enable
Router#
Router# configure terminal
Router(config)#
```

Изменим имя устройства:

```
Router(config)# hostname R4
```

Включение поддержки передачи пакетов в IPv6-сетях:

```
R4(config)# ipv6 forwarding
```

Настройка интерфейса eth0:

```
R4(config)#interface                                    eth0
R4(config-if)#ipv6 address 2001:fb6:31c0:3::4/64
R4(config-if)#            bandwidth            100000
R4(config-if)#exit
```

Настройка интерфейса eth1:

```
R4(config)#interface                                    eth1
R4(config-if)#ip address 2001:fb6:31c0:4::4/64
R4(config-if)#            bandwidth            100000
R4(config-if)#exit
R4(config)#
```

Настройка демона zebra завершена, необходимо сохранить конфигурацию. Отключиться от демона можно с помощью комбинации Ctrl+D:

```
R4(config)# write memory
Configuration saved to /etc/quagga/zebra.conf
R4(config)# exit
```

Далее на маршрутизаторе R4 необходимо настроить работу протокола OSPFv3 с помощью демона ospf6d. Запускаем демон, подключаемся к ospf6d по telnet и входим в режим глобального конфигурирования:

```
[root@elbrus ~]# systemctl start ospf6d
[root@elbrus ~]# telnet ::1 2606
User Access Verification
Password:
router-OSPF6>
router-OSPF6> enable
router-OSPF6#
router-OSPF6# configure terminal
router-OSPF6(config)#
```

Изменим имя устройства:

```
router-OSPF6(config)# hostname R4-OSPF6
```

Теперь необходимо включить процесс OSPFv3 и прописать сети, которые будет анонсировать маршрутизатор R3:

```
R4-OSPF6(config)# router ospf6
```

Теперь необходимо включить процесс OSPFv3, задать идентификатор Router ID и прописать сети, которые будет анонсировать наш маршрутизатор:

```
R4-OSPF6(config)# router ospf6
R4-OSPF6(config-router)# router-id 4.4.4.4
```

Пропишем все нужные для анонсирования сети, подключенные к маршрутизатору, путем добавления соответствующих интерфейсов и определим их в нулевую зону:

```
R4-OSPF6(config-router)# network eth0 area 0.0.0.0
R4-OSPF6(config-router)# network eth1 area 0.0.0.0
```

Настройка демона ospf6d завершена, необходимо сохранить конфигурацию. Отключиться от демона можно с помощью комбинации Ctrl+D.

```
R4-OSPF6(config)# write memory
Configuration saved to /etc/quagga/ospf6d.conf
R4-OSPF6(config)# exit
```

4.8. Анализ работы сети, работающей по протоколу OSPF

Когда все маршрутизаторы сконфигурированы, между ними должны установиться соседские отношения с помощью пакетов OSPF Hello. Проверим наличие соседских отношений на маршрутизаторе R1. Для этого подключаемся к ospf6d и из привилегированного режима (enable) вводим команду show ipv6 ospf6 neighbor:

```
[root@elbrus ~]# telnet 127.0.0.1 2606
R1-OSPF6>
R1-OSPF6> enable
R1-OSPF6# show ipv6 ospf6 neighbor
Neighbor   ID     Pri     DeadTime     State/IfState     Duration
I/F[State]
2.2.2.2           1       00:00:39     Full/BDR          00:00:50
eth0[DR]
4.4.4.4           1       00:00:38     Full/BDR          00:00:30
eth1[DR]
```

Вывод команды показывает, что R1 установил соседские отношения с R2 и R4. Таблица содержит следующие обозначения:

- Neighbor ID: Идентификатор (Router ID) маршрутизатора, с которым установлено соседство;
- Pri: Поле приоритета, указывает приоритет соседнего маршрутизатора. По умолчанию приоритеты установлены на 1;
- Dead Time: Оставшееся время, в течении которого маршрутизатор будет ждать пакет OSPF Hello от соседа, прежде чем объявить его недоступным (по умолчанию 40 секунд, т.е. 4 интервала Hello);
- State: Поле State показывает функциональное состояние соседнего маршрутизатора. Состояние Full означает полную синхронизацию с соседом;
- Duration: Поле Duration указывает время, прошедшее с установления соседских отношений с другими маршрутизаторами;
- I/F[State]: Поле Интерфейса указывает локальный интерфейс, на котором было установлено OSPF соседство.

Проверим работу протокола OSPFv3 проверкой доступности сети 2001:fb6:31c0:2::0/64 с маршрутизатора R1. Так как данная сеть не подключена напрямую к маршрутизатору R1, то с помощью протокола OSPFv3 он должен получить к ней маршрут от своих соседей.

Чтобы проверить работу сети IPv6, используется утилита ping6, работающая по протоколу ICMPv6, который предназначен для работы в сетях IPv6.

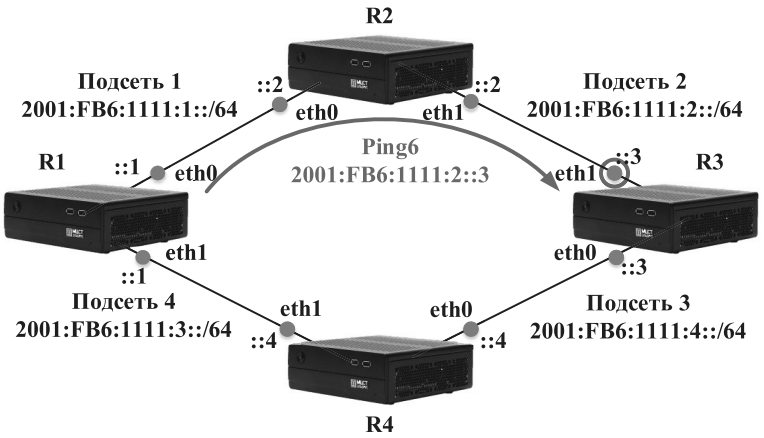


Рисунок 3. Проверка связности настроенной сети

На R1 выполняется команда ping6 и указывается адрес интерфейса eth1 маршрутизатора R3:

```
[root@elbrus ~]# ping6 2001:fb6:31c0:2::3 -c 3
PING 2001:fb6:31c0:2::3(2001:fb6:31c0:2::3) 56 data bytes
64 bytes from 2001:fb6:31c0:2::3: icmp_seq=1 ttl=63 time=0.495
ms
64 bytes from 2001:fb6:31c0:2::3: icmp_seq=2 ttl=63 time=0.171
ms
64 bytes from 2001:fb6:31c0:2::3: icmp_seq=3 ttl=63 time=0.135
ms
--- 2001:fb6:31c0:2::3 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 0.135/0.267/0.495/0.161 ms
```

Ключ –c (count) задает количество посылаемых ICMP запросов. Интерфейс eth1 маршрутизатора R3 доступен, ICMP пакеты успешно возвращаются, значит R1 получил необходимый маршрут и сеть работает корректно.

Проверим таблицу маршрутизации OSPFv3 на R1. Подключаемся по telnet к zebra, и в привилегированном режиме (enable) используем команду show ipv6 route ospf6:

```
[root@elbrus ~]# telnet 127.0.0.1 2601
R1>
R1> enable
R1# show ipv6 route ospf6
Codes: K - kernel route, C - connected, S - static, R -
RIPng,O - OSPFv6, I - IS-IS, B - BGP, A - Babel, > - selected
route, * - FIB route
O   2001:fb6:31c0:1::/64 [110/1] is directly connected, eth0,
00:01:48
O>*          2001:fb6:31c0:2::/64          [110/2]          via
fe80::9aa7:b0ff:fe00:ddc0, eth0, 00:00:59
O>*          2001:fb6:31c0:3::/64          [110/2]          via
fe80::9aa7:b0ff:fe00:dd21, eth1, 00:00:50
O   2001:fb6:31c0:4::/64 [110/1] is directly connected, eth1,
00:01:28
```

В таблице маршрутизации указаны маршруты, полученные R1 по OSPFv3 от своих соседей R2 и R4. Сеть 2001:fb6:31c0:2::/64 доступна через fe80::9aa7:b0ff:fe00:ddc0 (link-local адрес маршрутизатора R2), а сеть 2001:fb6:31c0:3::/64 доступна через fe80::9aa7:b0ff:fe00:dd21 (link-local адрес маршрутизатора R4).

Согласно таблице маршрутизации, сеть 2001:fb6:31c0:2::/64 доступна через R2. Проверим это, выполнив с R1 трассировку маршрута утилитой traceroute6 до адреса 2001:fb6:31c0:2::3, находящегося в подсети 2001:fb6:31c0:2::/64:

```
    [root@elbrus ~]# traceroute6 2001:fb6:31c0:2::3
    traceroute to 2001:fb6:31c0:2::3 (2001:fb6:31c0:2::3), 30
hops max, 80 byte packets
     1   2001:fb6:31c0:1::2          0.123 ms   0.077 ms   0.052
ms
     2   2001:fb6:31c0:2::3          0.140 ms   0.098 ms   0.088
ms
```

Как показывает трассировка, пакеты сначала попадают на адрес 2001:fb6:31c0:1::2 маршрутизатора R2, а потом достигают адреса назначения 2001:fb6:31c0:2::3 на маршрутизаторе R3.

Проверим как протокол OSPFv3 перестраивает маршруты до подсетей при изменении сетевой топологии, путем отключения интерфейса eth0 на маршрутизаторе R1. После отключения порта, протокол OSPFv3 перестроит маршрут к подсети 2001:fb6:31c0:2::/64 через маршрутизатор R4.

Как только происходит отключение интерфейса, алгоритм OSPF создает анонсы LSA (Link State Advertisement), в которых содержатся изменения в базе данных LSDB (Link State Data Base) маршрутизатора, а именно – отключенный интерфейс eth0, и рассылает их своим соседям, а они своим соседям, пока у всех маршрутизаторов не появится одинаковая копия базы LSDB, содержащая информацию об измененной топологии сети.

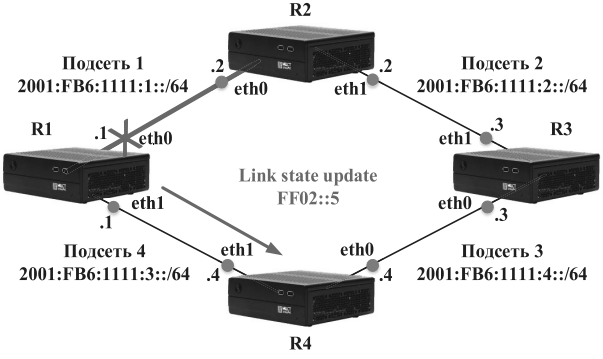


Рисунок 4. Схема сети при отключении интерфейса eth0

Запустим анализатор трафика Wireshark на интерфейсе eth1 маршрутизатора R1, и зададим фильтр по протоколу OSPF. После этого отключаем интерфейс eth0 и с помощью Wireshark смотрим анонсируемую информацию с интерфейса eth1 маршрутизатора R1:

```
[root@elbrus ~]# ip link set dev eth0 down
```

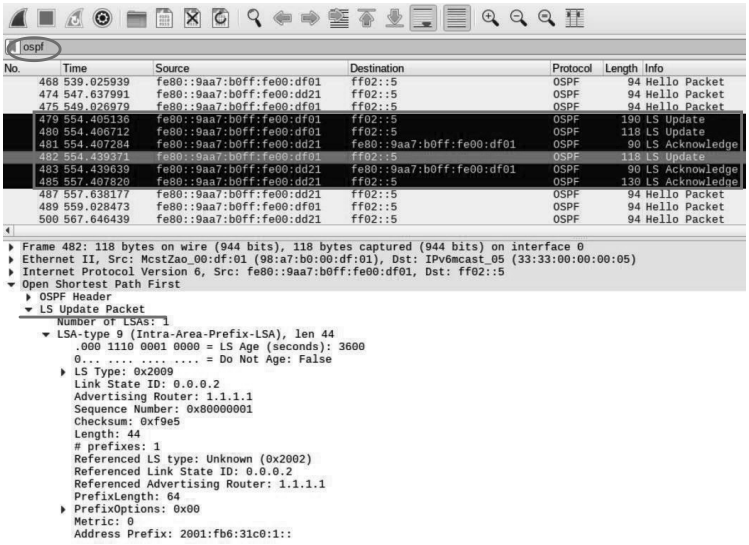


Рисунок 5. Анализ работы протокола OSPFv3 при отключении интерфейса

В частности, сообщения LSA, передающие информацию об изменении, называются обновлениями состояния канала LSU (Link State Update). Как показывает анализатор пакетов, при отключении интерфейса маршрутизатор R1 сразу начал отправку своим соседям на групповой адрес Update сообщений LSU, содержащих изменения базы данных LSDB. Маршрутизаторы R1 и R4, используя link-local адреса интерфейсов, обмениваются пакетами LS Update и подтверждениями LS Acknowledgment, после чего происходит перестроение маршрутов в сети.

После перестроения маршрутов, используем утилиту ping6 с маршрутизатора R1, проверим доступность адреса 2001:fb6:31c0:1::2, который находится в подсети 2001:fb6:31c0:1::/64:

```
[root@elbrus ~]# ping6 2001:fb6:31c0:1::2 -c 5
PING 2001:fb6:31c0:1::2(2001:fb6:31c0:1::2) 56 data bytes
64 bytes from 2001:fb6:31c0:1::2: icmp_seq=1 ttl=62 time=0.178
ms
64 bytes from 2001:fb6:31c0:1::2: icmp_seq=2 ttl=62 time=0.221
ms
```

```
    64 bytes from 2001:fb6:31c0:1::2: icmp_seq=3 ttl=62 time=0.202
ms
    64 bytes from 2001:fb6:31c0:1::2: icmp_seq=4 ttl=62 time=0.197
ms
    64 bytes from 2001:fb6:31c0:1::2: icmp_seq=5 ttl=62 time=0.200
ms

    --- 2001:fb6:31c0:1::2 ping statistics ---
    5 packets transmitted, 5 received, 0% packet loss, time 3998ms
    rtt min/avg/max/mdev = 0.178/0.199/0.221/0.020 ms
```

Интерфейс eth0 маршрутизатора R2 доступен, ICMP пакеты успешно возвращаются, протокол OSPFv3 перестроил маршрут к сети 2001:fb6:31c0:1::0/64 через маршрутизаторы R4 и R3.

После перестроения маршрута, используя утилиту traceroute6 с маршрутизатора R1 проверим путь трафика до адреса 2001:fb6:31c0:2::2:

```
    [root@elbrus ~]# traceroute6 2001:fb6:31c0:1::2
    traceroute to 2001:fb6:31c0:1::2 (2001:fb6:31c0:1::2), 30
hops max, 80 byte packets
     1   2001:fb6:31c0:4::4          0.193 ms   0.122 ms   0.092
ms
     2   2001:fb6:31c0:3::3          0.193 ms   0.168 ms   0.149
ms
     3   2001:fb6:31c0:1::2          0.263 ms   0.245 ms   0.222
ms
```

Трассировка показывает, что пакеты сначала попадают на узел 2001:fb6:31c0:4::4 (маршрутизатор R4), а потом через маршрутизатор R3 (2001:fb6:31c0:3::3) попадают в нужную подсеть на заданный IPv6-адрес 2001:fb6:31c0:1::2.

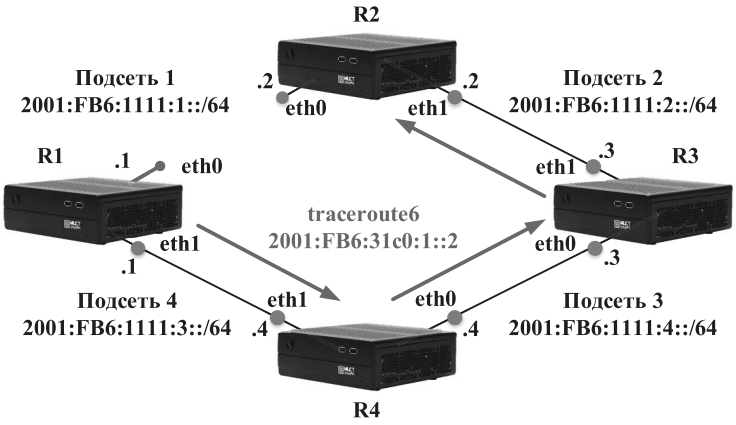


Рисунок 6. Схема прохождения трафика в сети

Когда маршруты перестроены, проверим таблицу маршрутизации OSPFv3 на R1. Для этого подключимся к zebra и в привилегированном режиме (enable) используем команду `show ipv6 route ospf6`:

```
    R1# show ipv6 route ospf6
    Codes: K - kernel route, C - connected, S - static, R -
RIPng, O - OSPFv6, I - IS-IS, B - BGP, A - Babel, > -selected
route, * - FIB route
    O>*         2001:fb6:31c0:1::/64         [110/4]         via
fe80::9aa7:b0ff:fe00:dd21, eth1, 00:03:58
    O>*         2001:fb6:31c0:2::/64         [110/3]         via
fe80::9aa7:b0ff:fe00:dd21, eth1, 00:04:33
    O>*         2001:fb6:31c0:3::/64         [110/2]         via
fe80::9aa7:b0ff:fe00:dd21, eth1, 00:08:30
    O    2001:fb6:31c0:4::/64 [110/1] is directly connected,
eth1, 00:09:08
```

Теперь в таблице маршрутизации указано, что все существующие сети доступны через fe80::9aa7:b0ff:fe00:df21 (link-local адрес маршрутизатора R4) с интерфейса eth1.

Протокол OSPFv3 также может выбирать наилучший маршрут до подсетей исходя их полосы пропускания канала на пути прохождения трафика. Если снизить скорость линии между маршрутизаторами, то протокол OSPFv3 также сначала начнет рассылку своим соседям Update сообщений, содержащих изменения в топологии сети, а после перестроит маршрут до сети назначения по более скоростным каналам.

На маршрутизаторе R1 вернем в активное состояние интерфейс eth0 и для правильной работы перезапустим демон zebra:

```
[root@elbrus ~]# ip link set dev eth0 up
[root@elbrus ~]# systemctl restart zebra
```

Проверим таблицу маршрутизации OSPFv3 на R1. Подключаемся по telnet к zebra, и в привилегированном режиме (enable) используем команду `show ipv6 route ospf6`. Подсеть 2001:fb6:31c0:1::/64 появилась в статусе подключенной напрямую (directly connected), а подсеть 2001:fb6:31c0:2::/64 снова доступна через R2:

```
[root@elbrus ~]# telnet 127.0.0.1 2601
R1>
R1> enable
R1# show ipv6 route ospf6
Codes: K - kernel route, C - connected, S - static, R
- RIPng,O - OSPFv6, I - IS-IS, B - BGP, A - Babel, > -
selected route, * - FIB route
O       2001:fb6:31c0:1::/64   [110/1]   is   directly
connected, eth0, 00:01:35
O>*       2001:fb6:31c0:2::/64       [110/2]       via
fe80::9aa7:b0ff:fe00:ddc0, eth0, 00:01:35
O>*       2001:fb6:31c0:3::/64       [110/2]       via
fe80::9aa7:b0ff:fe00:dd21, eth1, 00:01:35
O       2001:fb6:31c0:4::/64   [110/1]   is   directly
connected, eth1, 00:01:35
```

Изменим пропускную способность канала связи между R1 и R2. На интерфейсе eth0 маршрутизатора R1 программно снизим скорость интерфейса со 100 Мбит/с до 5 Мбит/с, в zebra в режиме глобального конфигурирования используем команду `bandwidth`, полоса указывается в кбит/с:

```
R1# configure terminal
R1(config)# interface eth0
R1(config-if)# bandwidth 5000
R1(config-if)# exit
```

Физически скорость интерфейса не изменилась, но маршрутизатор теперь считает скорость интерфейса равной 5 Мбит/с. После этого протокол OSPF снова перестроит маршрут ко всем подсетям через R4. Несмотря на три

транзитных участка по новому маршруту через R4 и R3, линия между R1 и R2 все равно будет иметь меньший приоритет из-за низкой скорости передачи, хотя и имеет всего один транзитный участок.

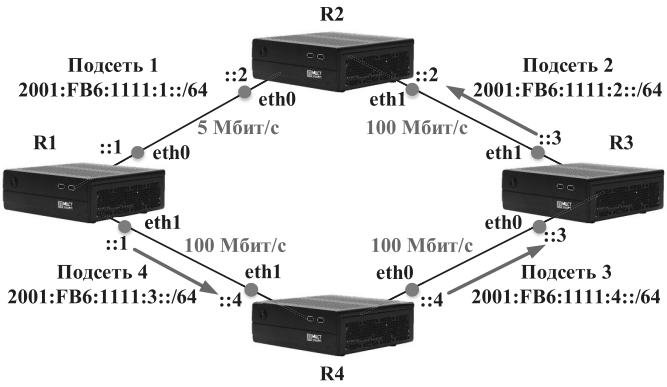


Рисунок 7. Изменение пропускной способности канала между R1 и R2

Таблица маршрутизации на R1 показывает изменения, произошедшие в сети. Теперь маршрут до всех подсетей идет снова идет через fe80::9aa7:b0ff:fe00:dd21, т.е. через маршрутизатор R4:

```
R1# show ipv6 route ospf6
Codes: K - kernel route, C - connected, S - static, R
- RIPng, O - OSPFv6, I - IS-IS, B - BGP, A - Babel,> -
selected route, * - FIB route
O>*       2001:fb6:31c0:1::/64       [110/4]       via
fe80::9aa7:b0ff:fe00:dd21, eth1, 00:00:33
O>*       2001:fb6:31c0:2::/64       [110/3]       via
fe80::9aa7:b0ff:fe00:dd21, eth1, 00:00:33
O>*       2001:fb6:31c0:3::/64       [110/2]       via
fe80::9aa7:b0ff:fe00:dd21, eth1, 00:00:33
O        2001:fb6:31c0:4::/64   [110/1]   is   directly
connected, eth1, 00:15:12
```

Управление передачей трафика в сети происходит с помощью метрик маршрутов, в таблице маршрутизации метрика маршрута указана вторым числом в квадратных скобках, первое число — это административная дистанция протокола маршрутизации.

Административная дистанция используется маршрутизатором для того, чтобы определить какому протоколу маршрутизации доверять больше, если у

обоих протоколов есть маршрут до подсети назначения. Например, более современный протокол OSPFv3 имеет административную дистанцию 110, а протокол RIPng – административную дистанцию 120. Таким образом, если в сети используются два и более протокола маршрутизации, у которых есть свои маршруты к сети назначения, то маршрутизатор выберет маршрут по протоколу с наименьшей административной дистанцией. У статических маршрутов административная дистанция равна единице, так как они прописываются вручную, а значит степень доверия к ним выше.

Если же в сети используется один протокол маршрутизации, и у него есть несколько различных маршрутов до подсети назначения, то маршрутизатор выбирает маршрут уже на основании метрик. Чем меньше метрика, тем больше маршрутизатор доверяет этому маршруту.

У разных протоколов маршрутизации метрика рассчитывается по-разному. Например, у протокола OSPFv3 метрика рассчитывается на основании суммарной цены маршрута, которая в свою очередь зависит от ширины полосы пропускания интерфейсов и каналов связи на маршруте до подсети назначения. А у протокола RIPng метрика рассчитывается исходя из числа транзитных участков между маршрутизатором и подсетью назначения.

Чтобы проверить, как меняется метрика маршрута в сети, на маршрутизаторе R1 программно изменим цену интерфейса eth1 путем уменьшения его скорости со 100 Мбит/с до 10 Мбит/с и проанализируем таблицу маршрутизации:

```
R1(config)#
R1(config)# interface eth1
R1(config-if)# bandwidth 10000
R1(config-if)# exit
```

После изменений трафик все равно будет проходить через маршрутизатор R3, так как суммарная цена маршрута через R3 ниже из-за более высокой пропускной способности каналов, проходящих через него.

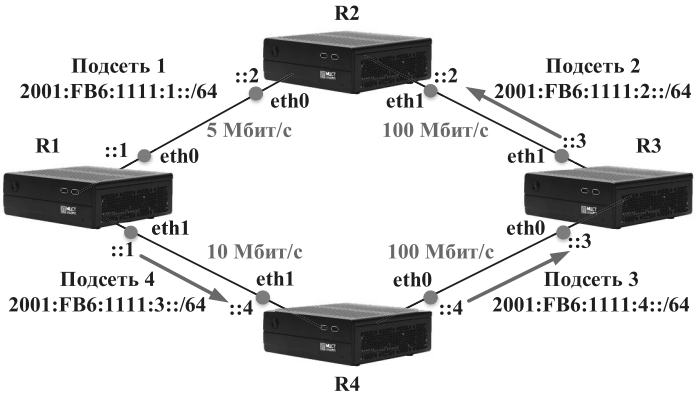


Рисунок 8. Изменение пропускной способности канала между R1 и R4

Уменьшая скорость интерфейса eth1 маршрутизатора R1, мы увеличиваем цену маршрута на участке сети между R1 и R4, что приводит к увеличению метрик на 9 единиц. Сделанные изменения будут видны в таблице маршрутизации R1:

```
[root@elbrus ~]# telnet 127.0.0.1 2601
R1>
R1> enable
R1# show ipv6 route ospf6
Codes: K - kernel route, C - connected, S - static, R
- RIPng, O - OSPFv6, I - IS-IS, B - BGP, A - Babel, > -
selected route, * - FIB route
O             2001:fb6:31c0:1::/64     [110/13]     via
fe80::9aa7:b0ff:fe00:dd21, eth1, 00:00:45
O>*       2001:fb6:31c0:2::/64       [110/12]       via
fe80::9aa7:b0ff:fe00:dd21, eth1, 00:00:45
O>*       2001:fb6:31c0:3::/64       [110/11]       via
fe80::9aa7:b0ff:fe00:dd21, eth1, 00:00:45
O       2001:fb6:31c0:4::/64   [110/10]   is   directly
connected, eth1, 00:00:45
```

После снижения скорости интерфейса eth1, метрики всех маршрутов увеличились, но трафик по-прежнему идет через R4:

```
[root@elbrus ~]# traceroute6 2001:fb6:31c0:2::2
traceroute            to            2001:fb6:31c0:2::2
(2001:fb6:31c0:2::2), 30 hops max, 80 byte packets
```

```
    1   2001:fb6:31c0:4::4      0.190 ms   0.110 ms   0.074
ms
    2   2001:fb6:31c0:3::3      0.181 ms   0.157 ms   0.150
ms
    3   2001:fb6:31c0:2::2      0.244 ms   0.222 ms   0.219
ms
```

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Основные концепции протокола OSPFv3. Применение и принцип работы. Назначение link local адресов в работе протокола OSPFv3.
2. Для каких целей применяется деление на области при построении сети по протоколу OSPFv3?
3. Каким образом происходит выбор наилучших маршрутов в сети OSPFv3? Назначение метрики маршрутов и административной дистанции.
4. Назначение пакетов OSPF Hello. Основные параметры, передаваемые в заголовке пакета OSPF Hello.
5. Концепция соседей протокола OSPFv3.
6. Назначение анонсов состояния канала LSA и баз данных состояния каналов LSDB протокола OSPFv3.

Практикум № 20
Настройка статической маршрутизации с использованием командной строки утилиты nmcli

1. ЦЕЛЬ РАБОТЫ

Познакомиться с понятиями публичных и частные сетей. Изучить механизмы функционирования маршрутизации в ОС Альт. Научится настраивать статическую маршрутизацию.

2. ЗАДАНИЕ НА РАБОТУ

Соедините три компьютера по схеме, изображённой на рисунке 1.

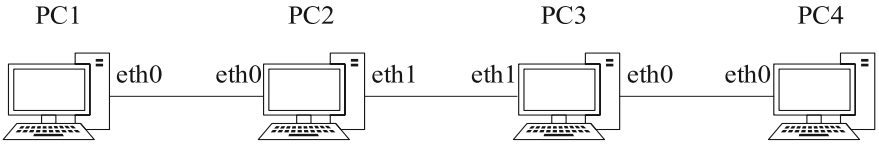


Рисунок 1. Схема соединения компьютеров

Настройте передачу пакетов между сетевыми интерфейсами.

Проверьте правильность настройки маршрутов.

3. ТЕРМИНЫ И ИСПОЛЬЗУЕМЫЕ КОМАНДЫ

Термины

IP-адреса, используемые в локальных сетях

Все используемые в Интернете адреса, должны регистрироваться, что гарантирует их уникальность в масштабе всей планеты. Такие адреса называются реальными или публичными IP-адресами.

Для локальных сетей, не подключенных к Интернету, регистрация IP-адресов, естественно, не требуется, так как, в принципе, здесь можно использовать любые возможные адреса. Однако, чтобы не допускать возможность конфликтов при последующем подключении такой сети к интернету, рекомендуется применять в локальных сетях только следующие диапазоны так называемых частных IP-адресов (в интернете эти адреса не существуют и использовать их там нет возможности), представленных в таблице 1.

Таблица 1. Диапазоны IP-адресов, используемых в локальных сетях

10.0.0.0 – 10.255.255.255
172.16.0.0 – 172.31.255.255
192.168.0.0 – 192.168.255.255

Особый смысл имеет IP-адрес, первый октет которого равен 127.x.x.x. Он используется для тестирования программ и взаимодействия процессов в пределах одной машины. Когда программа посылает данные по IP-адресу 127.0.0.1, то образуется как бы «петля». Данные не передаются по сети, а возвращаются модулям верхнего уровня, как только что принятые. Поэтому в IP-сети запрещается присваивать машинам IP-адреса, начинающиеся со 127. Этот адрес имеет название loopback. Можно отнести адрес 127.0.0.0 ко внутренней сети модуля маршрутизации узла, а адрес 127.0.0.1 — к адресу этого модуля на внутренней сети. На самом деле любой адрес сети 127.0.0.0 служит для обозначения своего модуля маршрутизации, а не только 127.0.0.1, например, 127.0.0.3.

Используемые команды

`nmcli` – утилита для управления сетевыми подключениями

`nano` – текстовый редактор.

`ping` – команда для отправки пакетов по указанному сетевому адресу.

`ip r` – команда для настройки маршрутизации.

4. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Для выполнения всех команд необходимо зайти в режим суперпользователя

```
[user@elbrus]# su-
Password:123
[root@elbrus]#
```

Определим, какой адрес необходимо указать для каждой из машин. На всех рабочих местах указан номер. Согласно этому номеру выбирается подсеть по следующему правилу – подсеть для двух компьютеров выбирается в соответствии с минимальным номером рабочего места. Общая формула выбора

`10.10.X.Y/24`

где

`X` – минимальный номер между двумя рабочими местами;

Y – номер рабочего места.

Пусть схема подключения выглядит следующим образом

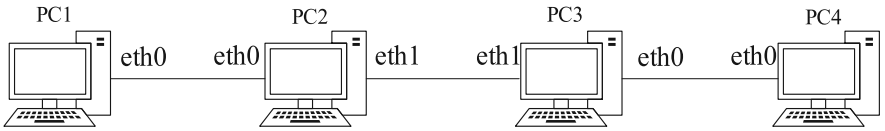


Рисунок 2. Пример схемы нумерации компьютеров

Согласно рисунку 2 и описанному выше правилу, компьютеры получат следующие ip адреса:

PC1:
- 10.10.1.1 на интерфейсе eth0

PC2:
- 10.10.1.2 на интерфейсе eth0
- 10.10.2.2 на интерфейсе eth1

PC3:
- 10.10.2.3 на интерфейсе eth1
- 10.10.3.3 на интерфейсе eth0

PC4:
- 10.10.3.4 на интерфейсе eth0

Для присвоения ip – адресов на необходимые интерфейсы, с последующим их подключением, воспользуемся утилитой “nmcli”

Добавим необходимые интерфейсы и настроим согласно рисунку 2. Рассмотрим на примере PC1 и PC2:

PC1:

```
[root@elbrus]# nmcli c add type Ethernet con-name eth0 ifname eth0 ip4 10.10.1.1/24
```

Теперь активируем настроенный интерфейс

```
[root@elbrus]# nmcli device connect eth0
```

PC2:

```
[root@elbrus]# nmcli c add type Ethernet con-name eth0 ifname eth0 ip4 10.10.1.2/24
[root@elbrus]# nmcli c add type Ethernet con-name eth1 ifname eth1 ip4 10.10.2.2/24
```

Теперь активируем настроенные интерфейсы:

```
[root@elbrus]# nmcli device connect eth0
[root@elbrus]# nmcli device connect eth1
```

Проверить правильность присвоения ip – адресов соответствующим интерфейсам можно при помощи команды:

```
[root@elbrus]# ip a
```

Для PC3 и PC4 проделайте аналогичные операции самостоятельно.

На компьютерах PC2 и PC3 включим пересылку пакетов между сетевыми интерфейсами, изменив в файле `sysctl.conf` параметр `net.ipv4.ip_forward` на 1. Для этого откроем файл, введя команду

```
[root@elbrus]# nano /etc/net/sysctl.conf
```

И изменим значение параметра `net.ipv4.ip_forward` на 1, если значение уже не равно единице (рисунок 3).

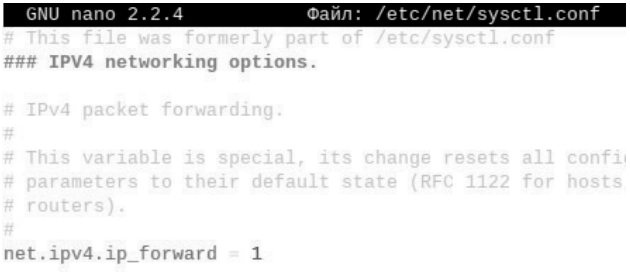


Рисунок 3. Фрагмент файла `sysctl.conf`

Далее, после записи в файле, сохраним введённые данные с помощью комбинации `CTRL+O` и выходим из редактирования – `CTRL+X`.

Пропишем следующие маршруты в консоли для каждого из компьютеров в соответствии с примером из рисунка 2.

PC1: `ip r add 10.10.2.0/24 via 10.10.1.2 dev eth0`
PC1: `ip r add 10.10.3.0/24 via 10.10.1.2 dev eth0`
PC2: `ip r add 10.10.3.0/24 via 10.10.2.3 dev eth1`
PC3: `ip r add 10.10.1.0/24 via 10.10.2.2 dev eth1`
PC4: `ip r add 10.10.2.0/24 via 10.10.3.3 dev eth0`
PC4: `ip r add 10.10.1.0/24 via 10.10.3.3 dev eth0`

Перезапустим сетевую службу на всех PC для применения параметров

```
[root@elbrus]# service network restart
```

Для проверки правильности выполнения работы используем команду `ping`:

Для PC1:

```
[root@elbrus]# ping 10.10.3.4
```

Для PC4:

```
[root@elbrus]# ping 10.10.1.1
```

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Назначение команд ОС Альт используемых в работе
2. Формат заголовка протокола IP версии 4.
3. Публичные и частные IP адреса.
4. Классы IP адресов.
5. Назначение и содержание таблицы маршрутизации.

Практикум № 21
Межсетевое экранирование на основе netfilter/iptables

1. ЦЕЛЬ РАБОТЫ

Познакомиться с понятием межсетевого экрана. Получить начальные практические навыки управлением iptables. Освоить взаимодействие правил и цепочек iptables.

2. ЗАДАНИЕ НА РАБОТУ

Соедините компьютеры по схеме, изображённой на рисунке 1 и 2.

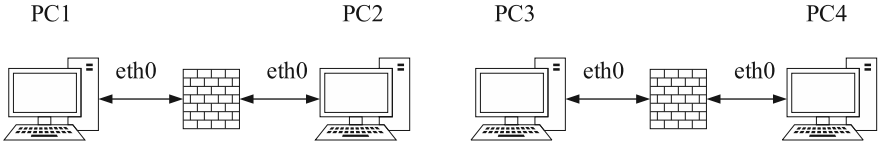


Рисунок 1. Схема соединения компьютеров 1

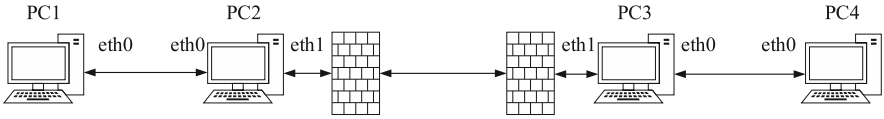


Рисунок 2. Схема соединения компьютеров 2

Развернуть межсетевые экраны на PC2 и PC3 которые будут наглядно демонстрировать работу каждой цепочки netfilter:

Используйте Iperf и Wireshark для более детального просмотра прохождения пакетов.

3. ТЕРМИНЫ И ИСПОЛЬЗУЕМЫЕ КОМАНДЫ

Общие сведение о подсистеме netfilter

Подсистема netfilter представляет собой средство пакетной фильтрации в ядре Linux Утилита iptables используется для управления netfilter.

Примечание: Netfilter/iptables кроме создание межсетевых экранов на основе пакетной фильтрации (в том числе и с учетом состояния соединения), также позволяет реализовать различные сложные схемы манипуляции сетевыми пакетами, в частности трансляцию сетевых адресов (NAT) для разделяемого доступа в Интернет и организации прозрачных прокси.

Сетевые пакеты в подсистеме netfilter проходят последовательность цепочек. Каждая цепочка включает заданный набор таблиц, содержащих упорядоченные списки правил. Каждое правило содержит параметры выбора и выполняемое действие при соответствии содержимого пакета параметрам.

Таблицы

CONNTRACK – механизм определения состояний

RAW – просматривается до передачи пакета системе определения состояний. Содержится в цепочках PREROUTING и OUTPUT.

MANGLE - содержит правила модификации заголовка IP-пакетов. Среди прочего, поддерживает изменения полей TTL и TOS, и маркеров пакета. Содержится во всех стандартных цепочках.

NAT — используется для трансляции (подмены) сетевых адресов. Содержится в цепочках PREROUTING, OUTPUT, и POSTROUTING.

FILTER — основная таблица, предназначенная для фильтрации сетевых пакетов; используется по умолчанию если название таблицы не указано. Содержится в цепочках INPUT, FORWARD, и OUTPUT.

Цепочки:

PREROUTING — для изначальной обработки входящих пакетов

INPUT — для входящих пакетов, адресованных непосредственно локальному компьютеру (входящих)

FORWARD — для проходящих (маршрутизируемых) пакетов

OUTPUT — для пакетов, создаваемых локальным компьютером (исходящих)

POSTROUTING — для окончательной обработки исходящих пакетов

Механизм определения состояний (CONNTRACK)

Механизм определения состояний (state machine, connection tracking) — система трассировки соединений, важная часть netfilter, при помощи которой реализуется межсетевой экран на сеансовом уровне (stateful firewall). Система позволяет определить, к какому соединению или сеансу принадлежит пакет.

Механизм определения состояний анализирует все пакеты, кроме тех, которые были помечены NOTRACK в таблице raw.

В системе netfilter каждый пакет, проходящий через механизм определения состояний, может иметь одно из четырёх возможных состояний:

NEW — пакет открывает новый сеанс. Классический пример — пакет TCP с флагом SYN.

ESTABLISHED — пакет является частью уже существующего сеанса.

RELATED — пакет открывает новый сеанс, связанный с уже открытым сеансом. Например, во время сеанса пассивного FTP клиент подсоединяется к порту 21 сервера, сервер сообщает клиенту номер второго, случайно выбранного порта, после чего клиент подсоединяется ко второму порту для передачи файлов. В этом случае второй сеанс (передача файлов по второму порту) связан с уже существующим сеансом (изначальное подсоединение к порту 21).

INVALID — все прочие пакеты.

Рассмотрим прохождение пакета в подсистеме netfilter.

Входящий пакет (Рисунок 3.) начинает обрабатываться с цепочки PREROUTING в таблицах raw, conntrack (определение состояний) и mangle. Затем он обрабатывается правилами таблицы nat данной цепочки. На этом этапе проверяется, необходима ли модификация адреса получателя пакета (DNAT). Важно сменить адрес получателя на данном этапе, так как маршрут пакета определяется сразу после того, как он покинет цепочку PREROUTING.

Далее возможны два варианта:

Если целью пакета является этот компьютер, то пакет будет отправлен в цепочку INPUT; после маршрутизации, он обрабатывается правилами цепочек INPUT. В случае прохождения цепочек пакет передается приложению.

Если целью пакета является другой компьютер, то пакет фильтруется правилами цепочки FORWARD таблиц mangle и filter, а затем к нему применяются правила цепочки POSTROUTING. На данном этапе можно использовать SNAT/MASQUERADE (подмена источника/маскировка). После этих действий пакет будет отправлен в сеть.

Непосредственно для фильтрации используются таблицы filter. Поэтому в рамках данной темы важно понимать, что для фильтрации пакетов, предназначенных данному узлу необходимо модифицировать таблицу filter

цепочки INPUT, для проходящих пакетов — цепочки FORWARD, для пакетов, созданных данным узлом — OUTPUT.

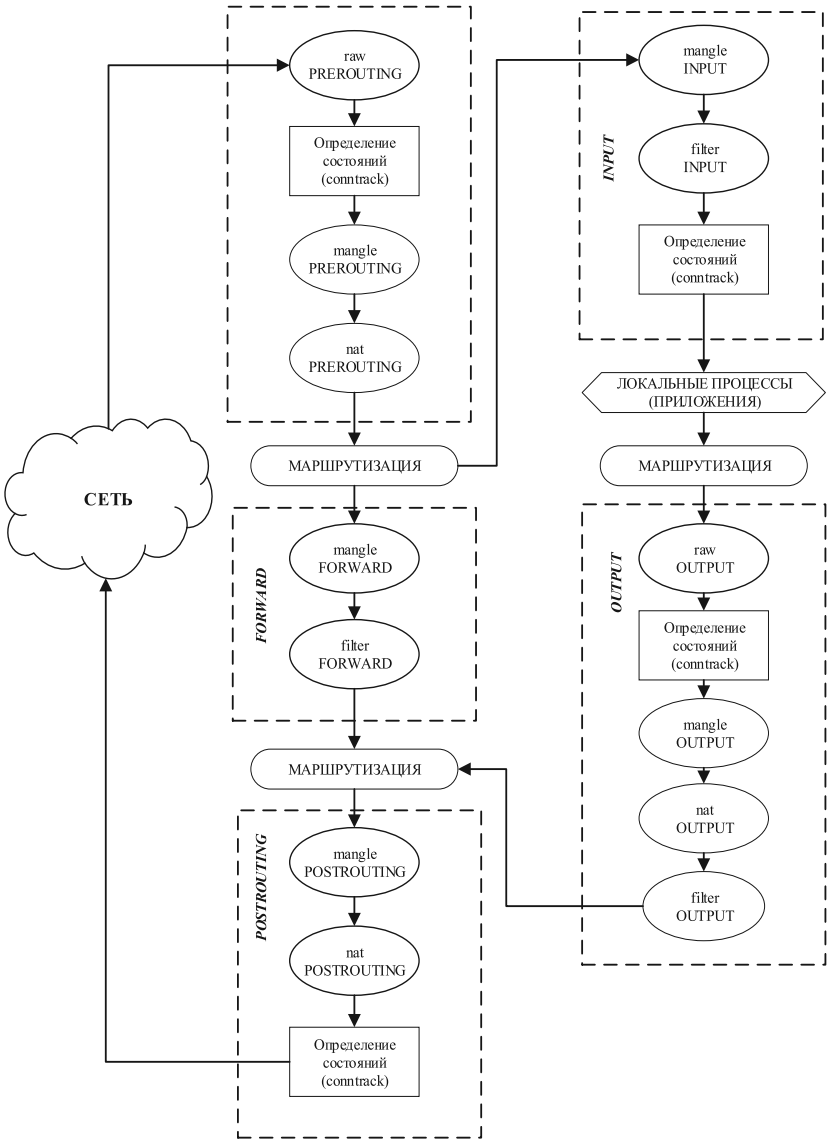


Рисунок 3. Прохождение пакета в подсистеме netfilter

Следует отметить, что одноименные таблицы каждой цепочки независимы.

Управление правилами iptables

Команда iptables позволяет редактировать правила таблиц netfilter. Каждое правило представляет собой запись, содержащую в себе параметры отбора (критерии), определяющие, подпадает ли пакет под данное правило, и действие, которое необходимо выполнить в случае соответствия параметрам отбора.

В общем виде правила записываются в виде:

iptables [-t table] command [match] [target/jump]

По умолчанию используется таблица filter, если же необходимо указать другую таблицу, то следует использовать спецификатор -t с указанием имени таблицы. После имени таблицы указывается команда, определяющая действие: вставить правило, или добавить правило в конец цепочки, или удалить правило и т. п. Match задает параметры отбора. Target указывает, какое действие должно быть выполнено при условии выполнения критериев в правиле: передать пакет в другую цепочку правил, "сбросить" пакет, выдать на источник сообщение об ошибке и т. п.

Примечание: для работы с iptables требуются права суперпользователя.

Для просмотра существующих правил, команда iptables используется с ключом -L. Так, для просмотра всех цепочек таблицы filter (-v — для более детального отображения):

```
# iptables -L -v
```

Примечание: Далее по тексту данной темы, если не названа таблица, то имеется в виду таблица filter.

Для просмотра цепочки FORWARD нужно указать имя цепочки

```
# iptables -v -L FORWARD
```

Сбросить все правила (-F) и удалить определенные пользователем цепочки (-X)

```
# iptables -F
# iptables -X
```

Таблица 1. Параметры утилиты iptables

Параметр	Описание	Пример
--protocol (сокращено -p)	Определяет протокол. Опции tcp, udp, icmp, или любой другой протокол определенный в /etc/protocols	iptables -A INPUT -p tcp
--source (-s)	IP адрес источника пакета. Может быть определен несколькими путями. Одиночный хост: host.domain.tld, или IP адрес: 10.10.10.3 Пул-адресов (подсеть): 10.10.10.3/24 или 10.10.10.3/255.255.255.0	iptables -A INPUT –s 10.10.10.3
--destination(-d)	IP адрес назначения пакета. Может быть определен несколькими путями - смотри — source	iptables -A INPUT --destination192.168.1.0/24
--source-port (--sport)	Порт источник, возможно только для протоколов — protocol tcp, или — protocol udp	iptables -A INPUT --protocol tcp --source-port 25
--destination-port (--dport)	Порт назначения, возможно только для протоколов — protocol tcp, или — protocol udp	iptables -A INPUT --protocol udp --destination-port 67
--state	Состояние соединения. Доступно, если модуль 'state' загружен с помощью '-m state'. Доступные опции: NEW, RELATED, ESTABLISHED, INVALID	iptables -A INPUT -m state --state NEW,ESTABLISHED
--in-interface (сокращенно -i)	Определяет интерфейс, на который прибыл пакет. Полезно для NAT и машин с несколькими сетевыми интерфейсами	iptables -t nat -A PREROUTING --in-interface eth0

Таблица 1. Параметры утилиты iptables (продолжение)

Параметр	Описание	Пример
--out-interface (сокращенно -o)	Определяет интерфейс, с которого уйдет пакет. Полезно для NAT и машин с несколькими сетевыми интерфейсами	iptables -t nat -A POSTROUTING --out-interface eth1
--syn	Сокращение для '--tcp-flags SYN,RST,ACK SYN'. Поскольку проверяет TCP флаги, используется с — protocol tcp. В примере показан фильтр для пакетов с флагом NEW, но без флага SYN. Обычно такие пакеты должны быть выброшены (DROP).	iptables -A INPUT --protocol tcp ! --syn -m state --state NEW

Действие, которое система выполнит, если пакет в одной из цепочек удовлетворяет условию, устанавливается с помощью ключа -j (--jump). Можно также передать пакет в другую цепочку.

Стандартные действия

-**ACCEPT** — пакет покидает данную цепочку и передается в следующую.
-**DROP** — отбросить удовлетворяющий условию пакет.
-**REJECT** — отбросить пакет, отправив отправителю ICMP-сообщение.
-**LOG** — протоколировать пакет.
-**RETURN** — возвратить пакет в предыдущую цепочку.
-**SNAT** — применить трансляцию адреса источника в пакете. Может использоваться только в цепочках POSTROUTING и OUTPUT таблицы nat.
-**DNAT** — применить трансляцию адреса назначения в пакете. Может использоваться только в цепочке POSTROUTING таблицы nat.
-**MASQUERADE** — используется вместо SNAT при наличии соединения с динамическим IP.
-**MARK** — используется для установки меток на пакеты.

Основные действия

- A — добавить правило в цепочку;
- C — проверить все правила;
- D — удалить правило;

- **I** — вставить правило с нужным номером;
- **L** — вывести все правила в текущей цепочке;
- **S** — вывести все правила;
- **F** — очистить все правила;
- **N** — создать цепочку;
- **X** — удалить цепочку;
- **P** — установить действие по умолчанию.

4. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Для выполнения всех команд необходимо зайти в режим суперпользователя

su –

Разворачивать наши межсетевые экраны будем на PC2 и PC3 для схемы, изображённой на рисунке 1 они будут являться серверами.

Рассмотрим 1 вариант прохождения пакетов в подсистеме netfilter.

Для начала выставим запретительно политику чтобы запретить принимать и отправлять пакеты с серверов. Для этого необходимо прописать в терминале:

```
[root@elbrus]# iptables -P INPUT DROP
[root@elbrus]# iptables - P OUTPUT DROP
```

Чтобы убедиться в правильности можно отправить эхо запросы, результат моделирования правила представлен на рисунке 4.

Теперь разрешим перемещаться пакетам в локальном интерфейсе для этого

```
[root@elbrus]# iptables -A INPUT –i lo –j ACCEPT
[root@elbrus]# iptables -A OUTPUT –o lo –j ACCEPT
```

Результат моделирования правила представлен на рисунке 4.

Важно прописывать правила в обе цепочки для установления правил в обоих направлениях.

Разрешим серверу доступ в интернет

```
[root@elbrus]# iptables -A INPUT –i eth2 –j ACCEPT
[root@elbrus]# iptables -A OUTPUT –o eth2 –j ACCEPT
```

Для проверки данного правила откройте любой установленный браузер и перейдите на любой поисковой сайт.

Разрешим отправку icmp сообщений

```
[root@elbrus]# iptables -A INPUT –p icmp –j ACCEPT
```

```
[root@elbrus]# iptables -A OUTPUT –p icmp –j ACCEPT
```

Результат моделирования правила представлен на рисунке 4.

Разрешим устанавливать соединения для всех установленных сессий

```
[root@elbrus]# iptables -A INPUT –m conntrack –-ctstate
RELATED,ESTABLISHED –j ACCEPT
[root@elbrus]# iptables -A OUTPUT –m conntrack –-
ctstate RELATED,ESTABLISHED –j ACCEPT
```

Разрешим SSH соединения до сервера:

```
[root@elbrus]# iptables -A INPUT –p tcp -m conntrack –-
ctstate NEW –m tcp –-dport 22 –j ACCEPT
```

Результат моделирования правила представлен на рисунке 5. Сохраним наши правила

```
[root@elbrus]# iptables-save> etc/sysconfig/iptables
```

Проверим работу межсетевого экрана:

Для установления SSH соединений пропишем на PC1 и на предложенный вопрос об установлении нового соединения ответим “yes”. Результат моделирования правила представлен на рисунке 5.

```
[root@elbrus]# iptables ssh <имя компьютера(PC2)>@<ip-
адрес компьютера(PC2)>
[root@elbrus]# iptables ssh elbrus@10.10.X.Y
```

где:

X – номер подсети

Y – номер компьютера указанный на мониторе

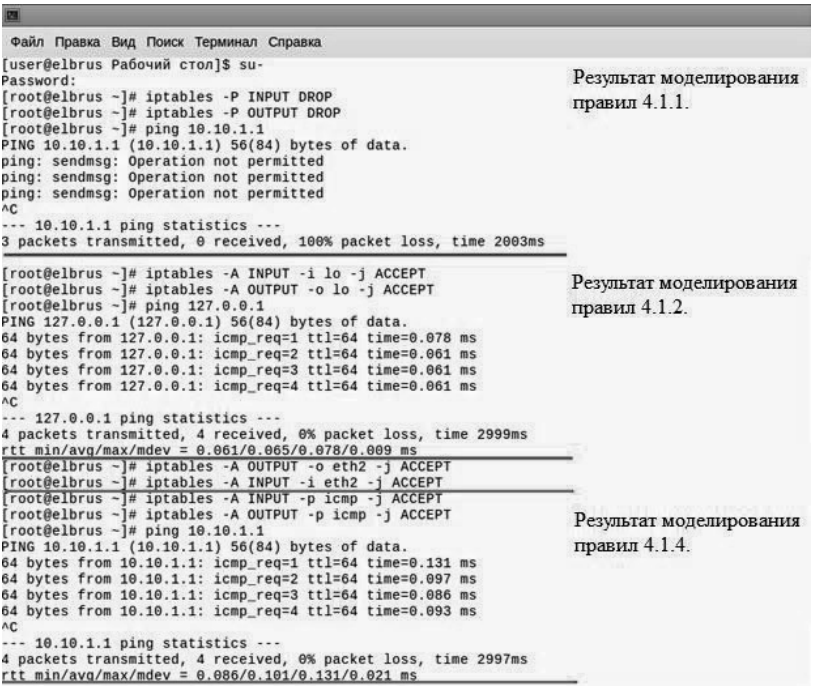


Рисунок 4. Результаты моделирования рассмотренных правил: 4.1.1 - 4.1.4

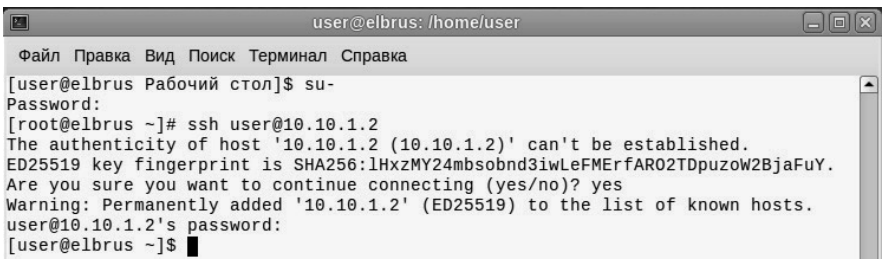


Рисунок 5. Результат установления SSH соединения, пункт 4.1.6

Для удобства дальнейшей работы после выполнения данной части работы сотрем все правила

```
[root@elbrus]# iptables -F
```

Рассмотрим 2 вариант прохождения пакетов в подсистеме netfilter.

В предыдущем случаем мы рассматривали по умолчанию запретительную политику и частично разрешали некоторые функции, на примере транзитного

узла рассмотрим обратную ситуацию, когда по умолчанию все разрешено, а мы будем частично запрещать некоторые функции.

Для начала выставим разрешительную политику чтобы разрешить принимать и отправлять пакеты. Для этого необходимо прописать в терминале PC2 и PC3

```
[root@elbrus]# -P INPUT ACCEPT
[root@elbrus]# -P OUTPUT ACCEPT
[root@elbrus]# -P FORWARD ACCEPT
```

Чтобы убедиться в правильности можно отправить эхо запросы с PC1 на PC4.

Теперь введем правило запрещающее TCP трафик по 80 порту от PC4 к PC1 и PC2 для этого

```
[root@elbrus]# iptables -A FORWARD -p tcp -s 10.10.3.4
-d 10.10.1.0/24 --dport 80 -j DROP
```

На рисунке 6 представлены результаты моделирования TCP трафика между PC4 – клиент и PC1 – сервер до применения правил. На рисунке 7 представлены результаты моделирования TCP трафика между PC4- клиент и PC1 –сервер после применения правил.

Теперь введем правило запрещающее ICMP трафик от PC4 к PC1 и PC2 перемещаться пакетам в локальном интерфейсе для этого

```
[root@elbrus]# iptables -I FORWARD -i eth0 -s
10.10.3.0/24 -d 10.10.1.0/24 -p icmp -j DROP
```

Проверим правильность работы нашего экрана. Для этого на всех PC установим утилиту iperf.

```
[root@elbrus]# apt-get install iperf
```

PC на который мы отправляем запрос переводим в режим сервера и указываем желаемый порт принятия трафика.

```
[root@elbrus]# iperf -s -p 80
```

PC с которого мы отправлять запросы переводим в режим клиента и указываем на какой PC и какой порт необходимо отправить запрос.

```
[root@elbrus]# iperf -c 10.10.1.1 -p 80
```

На рисунке 8 представлены результаты моделирования ICMP трафика между PC4- клиент и PC1 –сервер до применения правил 4.2.3. На рисунке 9 представлены результаты моделирования ICMP трафика между PC4- клиент и PC1 –сервер после применения правил 4.2.3.

PC4:

```
[root@elbrus ~]# iperf -c 10.10.1.1
------------------------------------------------------------
Client connecting to 10.10.1.1, TCP port 80
TCP window size: 85.0 KByte (default)
------------------------------------------------------------
[  3] local 10.10.3.4 port 57904 connected with 10.10.1.1 port 80
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec   584 MBytes   490 Mbits/sec
[root@elbrus ~]#
```

PC1:

```
[root@elbrus ~]# iperf -s -p 80
------------------------------------------------------------
Server listening on TCP port 80
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-10.0 sec   584 MBytes   490 Mbits/sec
^C[root@elbrus ~]#
```

Рисунок 6. Результаты моделирования TCP трафика между PC4- клиент и PC1 – сервер до применения правил 4.2.2

PC4:

```
[root@elbrus ~]# iperf -c 10.10.1.1 -p 80
^C^C[root@elbrus ~]#
```

PC1:

```
[root@elbrus ~]# iperf -s -p 80
------------------------------------------------------------
Server listening on TCP port 80
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
^C[root@elbrus ~]#
```

Рисунок 7. Результаты моделирования TCP трафика между PC4- клиент и PC1 – сервер после применения правил 4.2.2

PC4:

```
[root@elbrus ~]# ping 10.10.1.1
PING 10.10.1.1 (10.10.1.1) 56(84) bytes of data.
64 bytes from 10.10.1.1: icmp_req=1 ttl=62 time=0.295 ms
64 bytes from 10.10.1.1: icmp_req=2 ttl=62 time=0.175 ms
64 bytes from 10.10.1.1: icmp_req=3 ttl=62 time=0.171 ms
^C
--- 10.10.1.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1999ms
rtt min/avg/max/mdev = 0.171/0.213/0.295/0.059 ms
```

Рисунок 8. Результаты моделирования ICMP трафика между PC4- клиент и PC1 – сервер до применения правил 4.2.3.

PC4:

```
[root@elbrus ~]# ping 10.10.1.1
PING 10.10.1.1 (10.10.1.1) 56(84) bytes of data.
^C
--- 10.10.1.1 ping statistics ---
10 packets transmitted, 0 received, 100% packet loss, time 9006ms

[root@elbrus ~]#
```

Рисунок 9. Результаты моделирования ICMP трафика между PC4- клиент и PC1 –сервер после применения правил 4.2.3.

Для удобства удалим правила 4.2.2. и 4.2.3. Далее сохраним наши поправки и выйдем обратно в терминал.

Рассмотрим работу действий SNAT и DNAT.

Пусть у маршрутизатора есть дополнительный IP-адрес 10.10.2.3 и необходимо в него транслировать все пакеты, принадлежащие локальной сети 10.10.3.0/24. Для явного задания исходного IP-адреса трансляции необходимо использовать действие **SNAT** с параметром --to-source

```
[root@elbrus]#iptables -t nat -I POSTROUTING -s
10.10.3.0/24 -o eth1 -j SNAT --to-source 10.10.2.3
```

На рисунке 10 показаны результаты моделирования преобразования сетевых адресов SNAT между PC4- клиент и PC1 –сервер до применения правил 4.2.4. На рисунке 11 показаны результаты моделирования преобразования сетевых адресов SNAT между PC4- клиент и PC1 –сервер после применения правил 4.2.4.

Пусть необходимо, чтобы iperf-сервер на порту 8222 узла локальной сети 10.10.2.2 был доступен на внешнем адресе маршрутизатора 10.10.1.1. Данное правило развернем на PC2. Для этого необходимо использовать действие **DNAT** с параметром --to-destination

```
[root@elbrus]# iptables -t nat -I PREROUTING -i eth1 -p
tcp -d 10.10.2.2 --dport 8222 -j DNAT --to-destination
10.10.1.1
```

На рисунке 12 показаны результаты моделирования преобразования адреса назначения – DNAT между PC4 – клиент и PC1 – сервер после применения правил 4.2.5. До применения правил 4.2.5 результаты будут такими же как показано на рисунке 10.

Проверим работу межсетевого экрана:

PC4:

```
[root@elbrus ~]# iperf -c 10.10.1.1
------------------------------------------------------------
Client connecting to 10.10.1.1, TCP port 5001
TCP window size: 85.0 KByte (default)
------------------------------------------------------------
[  3] local 10.10.3.4 port 57904 connected with 10.10.1.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec   584 MBytes   490 Mbits/sec
[root@elbrus ~]#
```

PC1:

```
^C[root@elbrus ~]# iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  4] local 10.10.1.1 port 5001 connected with 10.10.3.4 port 57904
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-10.0 sec   584 MBytes   490 Mbits/sec
^C[root@elbrus ~]#
```

Рисунок 10. Результаты моделирования преобразования сетевых адресов SNAT между PC4- клиент и PC1 –сервер до применения правил 4.2.4

PC4:

```
^C^C[root@elbrus ~]# iperf -c 10.10.1.1
------------------------------------------------------------
Client connecting to 10.10.1.1, TCP port 5001
TCP window size: 85.0 KByte (default)
------------------------------------------------------------
[  3] local 10.10.3.4 port 57902 connected with 10.10.1.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec   581 MBytes   487 Mbits/sec
[root@elbrus ~]#
```

PC1:

```
^C[root@elbrus ~]# iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  4] local 10.10.1.1 port 5001 connected with 10.10.2.3 port 57902
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-10.0 sec   581 MBytes   486 Mbits/sec
^C[root@elbrus ~]#
```

Рисунок 11. Результаты моделирования преобразования сетевых адресов SNAT между PC4- клиент и PC1 –сервер после применения правил 4.2.4

Обратите внимание (Рисунок 10) до применения правил 4.2.4 на сервер пришли запросы с ip-адресом 10.10.3.4 отправителя - это адрес PC4, но после их

применения (Рисунок 11) произошла “подмена ip-адресов” и на сервер стали приходить запросы с ip-адресом отправителя 10.10.2.3 – это адрес PC3.

PC4:

```
[root@elbrus ~]# iperf -c 10.10.2.2 -p 8222
------------------------------------------------------------
Client connecting to 10.10.2.2, TCP port 8222
TCP window size: 85.0 KByte (default)
------------------------------------------------------------
[  3] local 10.10.3.4 port 34306 connected with 10.10.2.2 port 8222
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec   631 MBytes   529 Mbits/sec
[root@elbrus ~]#
```

```
^C[root@elbrus ~]# iperf -s -p 8222
------------------------------------------------------------
Server listening on TCP port 8222
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  4] local 10.10.1.1 port 8222 connected with 10.10.3.4 port 34306
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-10.0 sec   631 MBytes   528 Mbits/sec
^C[root@elbrus ~]#
```

PC1:

Рисунок 12. Результаты моделирования преобразования адреса назначения – DNAT между PC4 – клиент и PC1 – сервер после применения правил 4.2.5

Обратите внимание что ip – адрес назначения мы указывали 10.10.2.2 – это PC2, но запросы пришли на ip – адрес 10.10.1.1, это PC1.

Проверим результаты моделирования преобразования адреса назначения – DNAT между PC4 – клиент и PC1 – сервер после применения правил 4.2.5. при помощи программы “Wireshark”.

PC4:

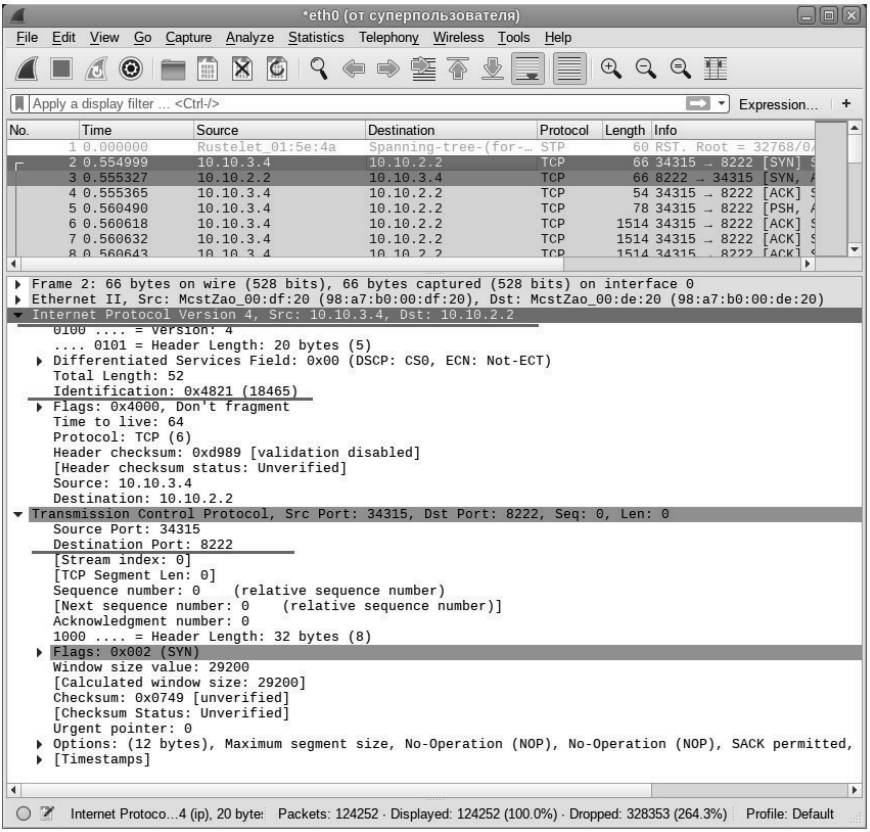


Рисунок 13. Результаты моделирования преобразования адреса назначения – DNAT между PC4 – клиент и PC1 – сервер после применения правил 4.2.5. при помощи программы “Wireshark” на стороне клиента.

PC2:

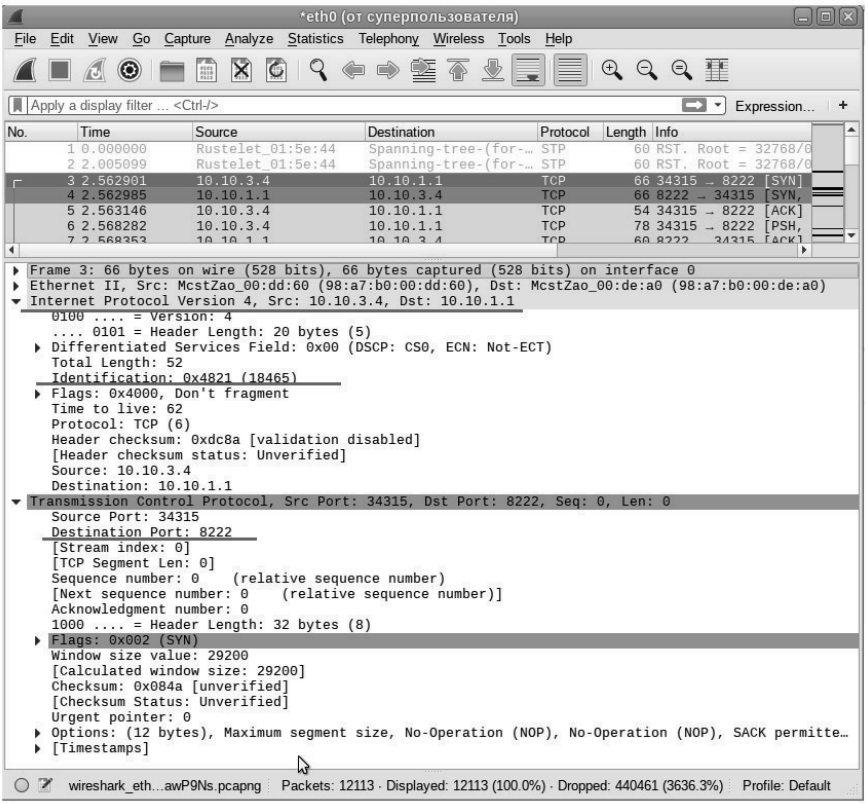


Рисунок 14. Результаты моделирования преобразования адреса назначения – DNAT между PC4 – клиент и PC1 – сервер после применения правил 4.2.5. при помощи программы “Wireshark” на транзитном узле – PC2.

Обратите внимание поля “Dst” и “Src” во вкладке “Internet Protocol Version 4”. На рисунке 13 видно, что запросы отправляются адресу назначения запроса 10.10.2.2 – PC2. На рисунке 14 видно, что PC2 как бы “ретранслирует” запросы с PC2 адрес 10.10.1.1 – PC1. Так же в этой же вкладке в поле “Identification” мы можем убедиться, что это и есть тот самый запрос с PC4 на PC2 с переводом на PC1, а не какой-либо другой. Во вкладке “Transmission Control Protocol” в поле “Destination Port” мы можем убедиться, что отправленные нами запросы пришли на указанный в правиле порт.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Модель взаимодействия открытых систем (ВОС).
2. Стек протоколов TCP/IP и его соответствие модели ВОС.
3. Протоколы стека TCP/IP используемые в лабораторной работе. Структура заголовка и назначение полей протоколов.
4. Преимущества свободного программного обеспечения.
5. Назначение команд Ос Альт используемых в работе.
6. Назначение и основные функции анализатора протоколов.
7. Назначение и основные функции программы моделирования TCP и UDP трафика.

Практикум № 22
Межсетевой экран для рабочей станции

1. ЦЕЛЬ РАБОТЫ

Познакомится с основными принципами настройки системы Netfilter с помощью утилиты iptables для локального компьютера. Понять принцип работы и основное назначение цепочек INPUT и OUTPUT. Научится изменять политику по умолчанию и добавлять правила фильтрации трафика.

2. ЗАДАНИЕ НА РАБОТУ

Используя стенд, изображённый на рисунке 1:

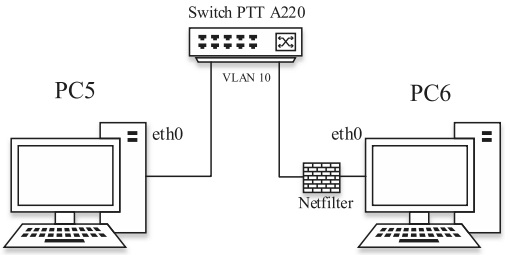


Рисунок 1. Схема лабораторного стенда

- ☐ Проверить состояние соединения
- ☐ Настроить запрещающую политику по умолчанию для PC6
- ☐ Разрешить “ответный” трафик от PC 5
- ☐ Разрешить PC 6 трафик от виртуального сетевого интерфейса
- ☐ Проверить правильность выполненной работы сравнив настройки Netfilter с Таблицей 1

Таблица 1. Правила фильтрации Netfilter

Действие	Протокол	Интерфейс приёма пакета	Интерфейс отправки пакета	Адрес источника	Адрес назначения	Дополнительные критерии
Chain: INPUT , Policy: DROP, Table: filter						
ACCEPT	ICMP	any	any	10.10.5.5	anywhere	RELATED, ESTABLISHED

Таблица 1. Правила фильтрации Netfilter (продолжение)

Действие	Протокол	Интерфейс приёма пакета	Интерфейс отправки пакета	Адрес источника	Адрес назначения	Дополнительные критерии
ACCEPT	all	lo	any	10.10.5.6	anywhere	-
Chain: OUTPUT , Policy: DROP , Table: filter						
ACCEPT	ICMP	any	any	10.10.5.6	10.10.5.5	-
ACCEPT	all	any	lo	10.10.5.6	anywhere	-

3. ТЕРМИНЫ И ИСПОЛЬЗУЕМЫЕ КОМАНДЫ

Термины

Netfilter – межсетевой экран (брандмауэр) в ОС Альт. Управление системой осуществляется посредством консольной программы iptables. С её помощью можно добавлять, удалять, изменять правила фильтрации пакетов. В системе Netfilter пакеты пропускаются через цепочки, которые состоят из таблиц. Существует три варианта прохождения пакета:

- Если пакет адресован локальной машине, на которой установлен экран: PREROUTING -> INPUT
- Если пакет транзитный (маршрутизируемый): PREROUTING-> FORWARD -> POSTROUTING
- Если пакет отправляется с локальной машины на которой установлен экран: OUTPUT -> POSTROUTING

При экранировании локального хоста особое внимание уделяется настройке правил фильтрации в цепочках INPUT и OUTPUT. Цепочка INPUT включает в себя таблицу mangle, filter и механизм определения состояния пакета conntrack (connection tracking). Она предназначена для фильтрации входящего трафика. Цепочка OUTPUT включает в себя таблицы raw, mangle, nat, filter и механизм conntrack. Она предназначена для фильтрации исходящего трафика.

Filter – основная таблица в цепочках INPUT, FORWARD, OUTPUT. Используется для фильтрации пакетов (если таблица в правиле не задана явно, используется по умолчанию).

Механизм определения состояний conntrack (connection tracking), не является таблицей, но входит во все цепочки кроме FORWARD. В системе netfilter пакеты, проходящие через механизм conntrack, могут иметь одно из

четырёх возможных состояний: NEW – первый пакет соединения, ESTABLISHED – пакет является частью уже существующего соединения, RELATED - пакет открывает новый сеанс, связанный с уже открытым сеансом, INVALID – все прочие пакеты.

Используемые команды

`ping` - утилита для проверки целостности и качества соединений в сетях на основе TCP/IP

`ifconfig` – отображение состояния сетевых интерфейсов

`nano` – текстовый редактор

`iptables` – утилита настройки netfilter

Описание используемых команд iptables:

`-A, --append` – добавляет новое правило

`-L, --list` –выводит список правил заданной цепочки или таблицы

`-P, --policy` – задаёт политику по умолчанию заданной цепочки

`-v, --verbose` – ключ (не команда) используется для вывода подробной информации

Описание используемых критериев в правилах iptables:

`-s, --source` – ip адрес источника пакета

`-d, --destination` – ip адрес получателя пакета

`-i, --in-interface` – интерфейс с которого получен пакет

`-o, --out-interface` – интерфейс с которого будет отправлен пакет

`-m conntrack --ctstate` [состояние пакета] – включение в критерий правила механизм определения состояний

4. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

4.1. Проверка и настройка сети

Для выполнения команд необходимо зайти в режим суперпользователя (пароль 123):

```
su-
```

Далее используя команду ifconfig просмотрим информацию о сетевых интерфейсах. Согласно Рисунку 1 компьютеры соединены по eth0. Запишите ip-адрес PC5 и PC6 и переходим к пункту 4.2.

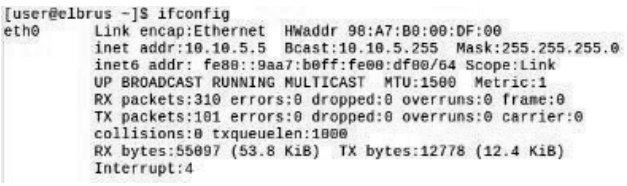


Рисунок 2. Результат выполнения команды ifconfig

Если ip адреса не назначены для интерфейсов eth0 на PC5 и PC6, их надо назначить вручную. Для этого необходимо создать файл ipv4address в директории /etc/net/ifaces/eth0. В командной строке выполним команду

```
nano /etc/net/ifaces/eth0/ipv4address
```

В файле необходимо записать ip адрес. Ip адрес выбирается по следующей формуле

```
10.10.X.Y/24
```

Где X – минимальный номер между двумя рабочими местами, Y – номер рабочего места.

Например, если компьютеры с номерами 5 и 6, то ip адреса будут следующие:

PC1: 10.10.5.5
PC2: 10.10.5.6

После записи ip адреса в файле сохраним его: CTRL+O, enter и выйдем из текстового редактора nano: CTRL+X.

Теперь произведём настройку сетевого интерфейса eth0. Для этого откроем файл options:

```
nano /etc/net/ifaces/eth0/options
```

В указанном файле должны быть следующие строки:

```
NM_CONTROLLED=no
DISABLED=no
BOOTPROTO=static
ONBOOT=yes
TYPE=eth
CONFIG_WIRLESS=no
CONFIG_IPV4=yes
```

Последний шаг - перезапуск сетевой службы на обеих компьютерах для применения параметров:

```
service network restart
```

Запустив команду ping проверьте состояние соединения. Остановить ping можно нажав CTRL+C. Пример

```
ping 10.10.5.6
```

```
[user@elbrus ~]$ ping 10.10.5.6
PING 10.10.5.6 (10.10.5.6) 56(84) bytes of data.
64 bytes from 10.10.5.6: icmp_req=1 ttl=64 time=0.144 ms
64 bytes from 10.10.5.6: icmp_req=2 ttl=64 time=0.098 ms
64 bytes from 10.10.5.6: icmp_req=3 ttl=64 time=0.109 ms
64 bytes from 10.10.5.6: icmp_req=4 ttl=64 time=0.091 ms
^C
--- 10.10.5.6 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2997ms
rtt min/avg/max/mdev = 0.091/0.110/0.144/0.022 ms
```

Рисунок 3. Выполнение команды ping

4.2. Настройка сетевого экрана

Просмотрим на PC2 политику по умолчанию и правила для таблицы filter.

```
iptables -L -v
```

```
[root@elbrus ~]# iptables -L -v
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

Рисунок 4. Просмотр правил таблиц filter

На PC2 заблокируем весь входящий трафик. Для этого настроим политику по умолчанию в цепочке INPUT для таблицы filter. Политика по умолчанию определяет, что делать с пакетами, которые не попадают ни под одно правило таблицы.

```
iptables -P INPUT DROP
```

Теперь попытки ping'a PC6 с PC5 будут неудачны.

```
[root@elbrus ~]# ping 10.10.5.6
PING 10.10.5.6 (10.10.5.6) 56(84) bytes of data.
ping: sendmsg: No buffer space available
ping: sendmsg: No buffer space available
ping: sendmsg: No buffer space available
ping: sendmsg: No buffer space available
^C
--- 10.10.5.6 ping statistics ---
21 packets transmitted, 0 received, 100% packet loss, time 39010ms
```

Рисунок 5. Результат команды ping

Поскольку ping работает по принципу: *запрос – ответ*, а политика по умолчанию запрещает весь входящий трафик на PC6, то и попытки ping'a PC5 с PC6 тоже будут неудачны.

Разрешим пакеты с состоянием RELATED и ESTABLISHED, это позволит PC2 получать ответы на отправленные эхо-запросы. Поскольку *запрос* имеет состояние NEW, а *ответ* – ESTABLISHED.

```
iptables –A INPUT –p icmp –m conntrack --ctstate
        RELATED,ESTABLISHED –j ACCEPT
```

Так как любое соединение по протоколу tcp начинается с запроса и получения ответа на его установление, то попытки установить соединение с PC6 будут неудачны. Но в свою очередь PC6 может установить соединение с PC5.

При выборе метода настройки сетевого экрана “заблокировать всё и разрешить выбранное”, для корректной работы сети необходимо разрешить доступ трафика от виртуального сетевого интерфейса (локальная петля, loopback).

```
iptables –A INPUT –i lo –j ACCEPT
```

Проверить доступность интерфейса можно командой:

```
ping 127.0.0.1
```

```
[root@elbrus ~]# iptables -A INPUT -i lo -j ACCEPT
[root@elbrus ~]# ping 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_req=1 ttl=64 time=0.080 ms
64 bytes from 127.0.0.1: icmp_req=2 ttl=64 time=0.078 ms
^C
--- 127.0.0.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.078/0.079/0.080/0.001 ms
[root@elbrus ~]#
```

Рисунок 6. Добавление разрешающего правила для интерфейса lo

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Политика по умолчанию в системе Netfilter.
2. Назначение механизма определения состояний conntrack.
3. Назначение Netfilter/iptables.
4. Типы состояний пакета в системе Netfilter.
5. Функции и состав цепочек INPUT и OUTPUT.

Практикум № 23
Межсетевой экран для локальной сети

1. ЦЕЛЬ РАБОТЫ

Познакомится с основными принципами экранирования локальной сети. Понять принцип работы и основное назначение цепочки FORWARD. Научится добавлять правила фильтрации трафика с критериями по адресу источника, приёмника, протоколу и порту. Проверить работоспособность экрана с помощью моделирования соединения утилитой iperf.

2. ЗАДАНИЕ НА РАБОТУ

Используя стенд, изображённый на рисунке 1:

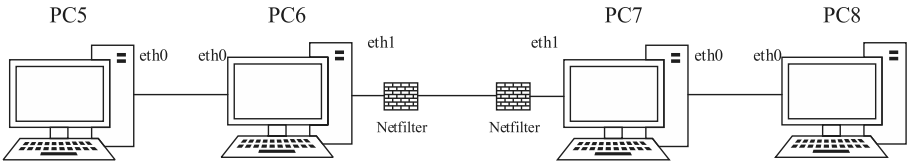


Рисунок 1. Схема лабораторного стенда

- ☐ Смоделировать tcp и udp соединение с помощью утилиты iperf между компьютерами PC5 и PC8
- ☐ Настроить правила фильтрации трафика для маршрутизаторов, роль которых выполняют PC6 и PC7
- ☐ Проверить правильность выполненной работы сравнив настройки Netfilter с Таблицей 1

Таблица 1. Правила фильтрации Netfilter

Действие	Протокол	Интерфейс приёма пакета	Интерфейс отправки пакета	Адрес источника	Адрес назначения	Дополнительные критерии
PC6 , Chain: FORWARD , Policy: DROP , Table: filter						
ACCEPT	UDP, TCP	any	any	10.10.7.0/24	10.10.5.5	-

ACCEPT	UDP, TCP	any	any	10.10.5.0/24	10.10.7.8	-
ACCEPT	ICMP	any	any	10.10.7.0/24	10.10.5.5	-
ACCEPT	ICMP	any	any	10.10.5.0/24	10.10.7.8	-
PC7 , Chain: FORWARD , Policy: DROP , Table: filter						
ACCEPT	UDP,ICMP	any	any	10.10.7.0/24	10.10.5.5	destination-port: 12345
ACCEPT	UDP,ICMP	any	any	10.10.5.0/24	10.10.7.8	source-port: 12345

3. ТЕРМИНЫ И ИСПОЛЬЗУЕМЫЕ КОМАНДЫ

Термины

В данной работе производится экранирование двух локальных сетей. Первая сеть состоит из PC5 и PC6, а вторая соответственно из PC7 и PC8. Компьютеры 6 и 7 так же выполняют роль маршрутизаторов, на которые установлен сетевой экран. Трафик проходящий через маршрутизаторы в Netfilter будет проходить фильтрацию в цепочках PREROUTING -> FORWARD -> POSTROUTING, поскольку этот трафик является маршрутизируемым (транзитным).

Основную роль в настоящей работе играет цепочка FORWARD. Именно она предназначена для транзитных пакетов. Эта цепочка состоит из таблиц mangle и filter.

mangle — содержит правила модификации (обычно заголовка) IP-пакетов. Среди прочего, поддерживает действия TTL (Time to live), TOS (Type of Service), и MARK (для изменения полей TTL и TOS, и для изменения маркеров пакета). Редко необходима и может быть опасна, при неправильной настройке.

filter — основная таблица, используется по умолчанию, если название таблицы не указано.

Проверка правильности настройки сетевых экранов будет осуществляется с помощью утилиты iperf. Она предназначена для моделирования соединения между двумя ПК по заранее выбранному протоколу (TCP или UDP) и порту.

Используемые команды

`ping` - утилита для проверки целостности и качества соединений в сетях на основе TCP/IP.

`ifconfig` – отображение состояния сетевых интерфейсов.

`iptables` – утилита настройки netfilter.

`iperf` - консольная клиент-серверная программа — генератор TCP и UDP трафика.

Описание используемых команд iptables:

`-A, --append` – добавляет новое правило;

`-L, --list` – выводит список правил заданной цепочки или таблицы;

`-P, --policy` – задаёт политику по умолчанию заданной цепочки;

`-v, --verbose` – ключ (не команда) используется для вывода подробной информации.

Описание используемых критериев в правилах iptables:

`-s, --source` – ip адрес источника пакета;

`-d, --destination` – ip адрес получателя пакета;

`-i, --in-interface` – интерфейс с которого получен пакет;

`-o, --out-interface` – интерфейс с которого будет отправлен пакет.

4. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

4.1. Настройка сети

Для выполнения всех команд необходимо зайти в режим суперпользователя (пароль 123)

```
su-
```

Далее используя команду `ifconfig` просмотрим информацию о сетевых интерфейсах (Рисунок 2). Согласно Рисунку 1 компьютеры 5,6 и 7,8 соединены по eth0, а компьютеры 6 и 7 по eth1. Запишите ip-адреса PC5 – PC8. (если ip адреса не назначены для интерфейсов, необходимо назначить их, подробнее о настройке сети написано в Практикуме № 22, пункт 4.1)

Рисунок 2. Результат выполнения команды ifconfig

Далее необходимо проверить маршрутизацию, запустите команду `ip r` на каждом ПК. Если в таблице маршрутизации присутствуют только стандартные настройки, а ping с PC5 к PC8 не проходит, то необходимо добавить правила маршрутизации трафика для каждого ПК. Пример настройки маршрутизации для компьютеров с номерами 5,6,7,8.

PC5:

```
ip r add 10.10.6.0/24 via 10.10.5.6 dev eth0
ip r add 10.10.7.0/24 via 10.10.5.6 dev eth0
```

PC6:

```
ip r add 10.10.7.0/24 via 10.10.6.7 dev eth1
```

PC7:

```
ip r add 10.10.5.0/24 via 10.10.6.6 dev eth1
```

PC8:

```
ip r add 10.10.6.0/24 via 10.10.7.7 dev eth0
ip r add 10.10.5.0/24 via 10.10.7.7 dev eth0
```

Для корректной работы включите на PC6 и PC7 пересылку пакетов между интерфейсами, изменив в файле sysctl.conf параметр net.ipv4.ip_forward на 1. Для этого откроем файл, используя команду

```
nano /etc/net/sysctl.conf
```

Сохраним изменения в файле: CTRL+O, enter. Закроем текстовый редактор командой CTRL+X.

4.2. Настройка сетевого экрана

Первоначально смоделируем tcp соединение между PC5 и PC8. Для этого воспользуемся утилитой iperf. Где PC5 будет выступать в роли сервера, а PC8 в роли клиента. Для PC5 запуск сервера выглядит следующим образом:

```
iperf -s -p 12345
```

где ключ –s означает запуск сервера, а –p порт.

```
[root@elbrus ~]# iperf -s -p 12345
------------------------------------------------------------
Server listening on TCP port 12345
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
```

Рисунок 3. Запуск TCP сервера

На PC8 запустим клиент, который отправит tcp трафик нашему серверу:

```
iperf -c 10.10.5.5 -p 12345
```

```
[root@elbrus ~]# iperf -c 10.10.5.5 -p 12345
------------------------------------------------------------
Client connecting to 10.10.5.5, TCP port 12345
TCP window size: 85.0 KByte (default)
------------------------------------------------------------
[  3] local 10.10.7.8 port 57848 connected with 10.10.5.5 port 12345
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec   628 MBytes   526 Mbits/sec
```

Рисунок 4. Запуск клиента на PC8

где
–c означает запуск и подключение клиента к ip адресу.

В случае успешного подключения на PC5 можно увидеть сообщение:

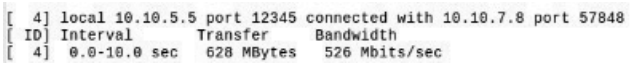


Рисунок 5. Подключение к серверу

Теперь настроим сетевой экран на PC6. Установим политику по умолчанию DROP для всех транзитных пакетов.

```
iptables -P FORWARD DROP
```

Теперь при попытке установить соединение с PC5, будет возникать ошибка, поскольку сетевой экран на PC6 не пропускает маршрутизируемые пакеты.

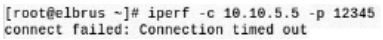


Рисунок 6. Сообщение об ошибке

Чтобы разрешить трафик от PC8 к PC5, добавим его в исключения брандмауэра:

```
iptables -A FORWARD -p tcp -s 10.10.7.0/24 -d
          10.10.5.5 -j ACCEPT
iptables -A FORWARD -p tcp -s 10.10.5.0/24 -d
          10.10.7.8 -j ACCEPT
```

Вторая команда, соответственно, разрешает трафик в обратную сторону.

Далее смоделируем UDP соединение и так же добавим его в исключения сетевого экрана. Соответствующие команды для компьютеров 5 и 8:

PC5 (для ввода команды необходимо открыть второй терминал, либо отключить TCP-сервер командой CTRL+C в текущем терминале):

```
iperf -s -u -p 5001
```

PC8:

```
iperf -c 10.10.5.5 -u -p 5001
```

Поскольку TCP протокол сначала отправляет запрос на установление соединения и только после получения ответа посылает данные, UDP протокол сразу отправляет дейтаграммы по указанному адресу и только после ждёт ack-подтверждение, то результат запуска клиента для PC8 будет выглядеть так:

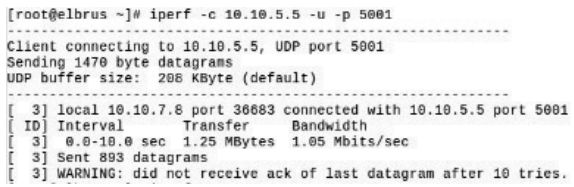


Рисунок 7. Результат запуска UDP клиента

При этом сообщения о подключении на PC5 не будет, поскольку правила Netfilter разрешают только TCP соединения. Разрешим UDP трафик между PC8 и PC5:

```
iptables -A FORWARD -p udp -s 10.10.7.0/24 -d
            10.10.5.5 -j ACCEPT
iptables -A FORWARD -p udp -s 10.10.5.0/24 -d
            10.10.7.8 -j ACCEPT
```

Теперь при выполнении команды на PC8: `iperf -c 10.10.5.5 -u -p 5001`, на PC5 будет соответствующее сообщение о подключении:

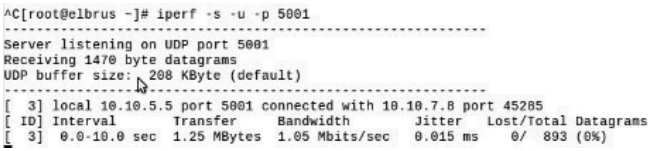


Рисунок 8. Сообщение о подключении к серверу

Заметим, что ping с PC8 к PC5 (и в обратном направлении) проходить не будет, поскольку в разрешающих правилах в критериях только TCP и UDP протокол, а команда ping работает по ICMP протоколу.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Назначение цепочки FORWARD.
2. Назначение и функционал утилиты iperf.
3. Описание TCP и UDP протокола.
4. Описание и назначение таблиц filter и mangle.
5. Основные критерии фильтрации трафика.

Практикум № 24
Межсетевое экранирование и доступ из внешней сети по протоколу SSH

1. ЦЕЛЬ РАБОТЫ

Познакомится с основными принципами настройки Netfilter для защиты локального сервера с доступом во внешнюю сеть. Создать и настроить доступ к локальному серверу из внешней сети по протоколу SSH.

2. ЗАДАНИЕ НА РАБОТУ

Используя стенд, изображённый на рисунке 1.

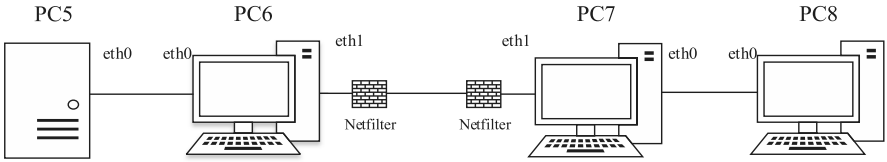


Рисунок 1. Схема лабораторного стенда

- ☐ Смоделировать сервер на PC5 с помощью утилиты iperf
- ☐ Настроить NAT в Netfilter на PC6 и PC7
- ☐ Настроить доступ к серверу, в роли которого выступает PC5, из внешней сети используя протокол SSH
- ☐ Проверить правильность выполненной работы сравнив настройки Netfilter с Таблицей 1

Таблица 1. Правила фильтрации Netfilter

Действие	Протокол	Интерфейс приёма пакета	Интерфейс отправки пакета	Адрес источника	Адрес назначения	Дополнительные критерии
PC6 , Chain: PREROUTING , Policy: ACCEPT , Table: nat						
DNAT	TCP	any	any	anywhere	10.10.6.6	dport:12345, to 10.10.5.5
PC6 , Chain: POSTROUTING , Policy: ACCEPT , Table: nat						
SNAT	TCP	any	eth1	anywhere	anywhere	to: 10.10.6.6
PC6 , Chain: FORWARD , Policy: DROP , Table: filter						

Таблица 1. Правила фильтрации Netfilter (продолжение)

Действие	Протокол	Интерфейс приёма пакета	Интерфейс отправки пакета	Адрес источника	Адрес назначения	Дополнительные критерии
ACCEPT	TCP	any	any	anywhere	anywhere	destination-port: 12345
ACCEPT	TCP	any	any	anywhere	anywhere	source-port: 12345
ACCEPT	TCP	any	any	anywhere	anywhere	source-port: 22
ACCEPT	TCP	any	any	anywhere	anywhere	destination-port: 22

3. ТЕРМИНЫ И ИСПОЛЬЗУЕМЫЕ КОМАНДЫ

Термины

В данной работе рассматривается вопрос защиты локального сервера с обеспечением доступа во внешнюю сеть (PC7 и PC8), а также обеспечение безопасного соединения посредством протокола SSH. Моделирование сервера осуществляется с помощью утилиты iperf на PC5. Компьютеры 6 и 7 выступают в роли маршрутизаторов на которых установлены Netfilter.

Обеспечение безопасного взаимодействия сегментов внутренней сети (PC5 и PC6) с внешней сетью осуществляется настройкой правил в таблице nat (Network Address Translation). Данная таблица входит в цепочки PREROUTING, OUTPUT и POSTROUTING.

Поскольку в настоящей лабораторной работе компьютеры, на которых установлен сетевой экран, играют роль маршрутизаторов, то все пакеты для них будут является транзитными и будут пропускаться через цепочки следующим образом: PREROUTING -> FORWRARD -> POSTROUTING.

Через таблицу nat проходят только пакеты, создающие новое соединение, т.е. имеющие состояние NEW (согласно системе определений состояний conntrack). Если имеется необходимость не отслеживать определённые пакеты с состоянием NEW в таблице nat, необходимо внести правило маркирующие такие пакеты в таблице raw.

Таблица nat поддерживает действия DNAT, SNAT, MASQUERADE и REDIRECT.

Действие SNAT изменяет адрес источника пакета, таким образом скрывая сегменты локальной сети и их адреса от внешней сети. Это действие так же

можно использовать для предоставления выхода в Интернет другим компьютерам из локальной сети, имея лишь один уникальный IP адрес. SNAT допускается выполнять только в таблице nat, в цепочке POSTROUTING.

Действие DNAT изменяет адрес назначения пакета, что позволяет перенаправлять разрешённый трафик на соответствующие сегменты сети. Данное действие может, к примеру, успешно использоваться для предоставления доступа к вашему серверу, находящемуся в локальной сети, и не имеющему реального IP адреса. Действие DNAT может выполняться только в цепочках PREROUTING и OUTPUT таблицы nat.

Действие MASQUERADE решает ту же задачу, что и SNAT, но может работать, например, с dialup подключением или DHCP, т.е. в тех случаях, когда IP адрес присваивается устройству динамически. Действие MASQUERADE допускается указывать только в цепочке POSTROUTING таблицы nat.

Действие REDIRECT выполняет перенаправление пакетов и потоков на другой порт той же самой машины. Действие REJECT может использоваться только в цепочках INPUT, FORWARD и OUTPUT.

Безопасное управление сервером из внешней сети осуществляется по протоколу SSH. Данный протокол шифрует весь трафик, включая и передаваемые пароли, допускает выбор различных алгоритмов шифрования (по умолчанию RSA). SSH позволяет безопасно передавать в незащищённой среде практически любой другой сетевой протокол, таким образом, через защищённый канал можно удалённо работать на компьютере через командную строку.

Используемые команды

`ifconfig` – отображение состояния сетевых интерфейсов

`iptables` – утилита настройки netfilter

`iperf` - консольная клиент-серверная программа — генератор TCP и UDP трафика

`ssh` – утилита для организации защищённого канала с удалённой машиной и выполнения на ней команд

Описание используемых команд iptables:

`-A, --append` – добавляет новое правило

`-P, --policy` – задаёт политику по умолчанию заданной цепочки

Описание используемых критериев в правилах iptables:

`-s, --source` – ip адрес источника пакета

`-d, --destination` – ip адрес получателя пакета

`-i, --in-interface` – интерфейс с которого получен пакет
`-o, --out-interface` – интерфейс с которого будет отправлен пакет
`-dport, --destination-port` – адрес порта назначения
`-sport, --source-port` – адрес порта источника
`-p, --protocol` – протокол

4. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

4.1. Информация о сети

Для выполнения всех команд необходимо зайти в режим суперпользователя (пароль 123):

```
su-
```

Далее используя команду `ifconfig` просмотрим информацию о сетевых интерфейсах (Рисунок 2). Согласно Рисунку 1 компьютеры 5,6 и 7,8 соединены по eth0, а компьютеры 6 и 7 по eth1. Запишите ip-адреса PC5 – PC8. (если ip адреса не назначены для интерфейсов, необходимо назначить их и настроить сеть, подробнее о настройке сети написано в Практикуме № 22, пункт 4.1)

```
[user@elbrus ~]$ ifconfig
eth0      Link encap:Ethernet  HWaddr 98:A7:B0:00:DF:00
          inet addr:10.10.5.5  Bcast:10.10.5.255  Mask:255.255.255.0
          inet6 addr: fe80::9aa7:b0ff:fe00:df00/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:310 errors:0 dropped:0 overruns:0 frame:0
          TX packets:101 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:55097 (53.8 KiB)  TX bytes:12778 (12.4 KiB)
          Interrupt:4

eth1      Link encap:Ethernet  HWaddr 98:A7:B0:00:DF:01
          inet6 addr: fe80::9aa7:b0ff:fe00:df01/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:296 errors:0 dropped:0 overruns:0 frame:0
          TX packets:31 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:58082 (56.7 KiB)  TX bytes:3056 (2.9 KiB)
          Interrupt:10

eth2      Link encap:Ethernet  HWaddr 98:A7:B0:00:DF:02
          inet addr:172.16.1.25  Bcast:172.16.1.255  Mask:255.255.255.0
          inet6 addr: fe80::9aa7:b0ff:fe00:df02/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:280 errors:0 dropped:0 overruns:0 frame:0
          TX packets:148 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:41301 (40.3 KiB)  TX bytes:20886 (20.3 KiB)
          Interrupt:14
```

Рисунок 2. Результат выполнения команды ifconfig

При необходимости: для корректной работы NAT (SNAT/DNAT) включите на PC6 и PC7 пересылку пакетов между интерфейсами командой:

```
sys -w net.ipv4.ip_forward=1
```

4.2. Настройка сетевого экрана

Первоначально запустим сервер на PC5. Для этого воспользуемся утилитой iperf. Для PC5 запуск сервера выглядит следующим образом:

```
iperf -s -p 12345
```

где ключ –s означает запуск сервера, а –p порт.

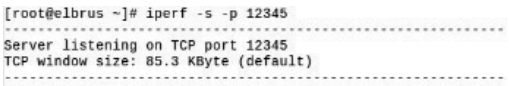


Рисунок 3. Запуск TCP сервера

Далее настроим сетевой экран на PC6. Для цепочки POSTROUTING таблицы nat добавим правило:

```
iptables -t nat -A POSTROUTING -o eth1 -j SNAT --to-
source 10.10.6.6
```

где ключ –t указывает таблицу в цепочке, в которую необходимо добавить правило, 10.10.6.6 – ip адрес PC6 на внешнем интерфейсе eth1 (согласно Рисунку 1).

Таким образом ко всем пакетам, проходящим через PC6, будет применятся адрес источника 10.10.6.6.

Добавим правило в таблицу nat цепочки PREROUTING:

```
iptables -t nat -A PREROUTING -p tcp -d 10.10.6.6 --
 dport 12345 -j DNAT --to-destination 10.10.5.5
```

Данное правило применяет изменение адреса назначения, т.е. направляет трафик на наш сервер, если пакет адресован порту 12345.

Теперь при попытке подключится к серверу по адресу 10.10.6.6:12345, подключение будет направлено к нашему серверу 10.10.5.5:12345, адрес которого скрыт от внешней сети (Рисунок 4). Установим соединение с сервером с PC8 отправив запрос на подключение на сетевой экран PC6. Для PC8 команда имеет вид:

```
iperf -c 10.10.6.6 -p 12345
```

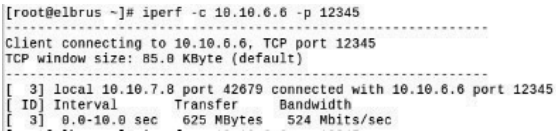


Рисунок 4. – установление соединения с сервером

Далее запретим доступ всем транзитным пакетам, кроме тех, которые адресованы на порт 12345:

```
iptables -P FORWARD DROP
iptables -A FORWARD -p tcp --dport 12345 -j ACCEPT
iptables -A FORWARD -p tcp --sport 12345 -j ACCEPT
```

Перед настройкой и установлением соединения по протоколу SSH (по умолчанию порт 22) к нашему серверу (PC5) из внешней сети необходимо добавить разрешающие правила на PC6:

```
iptables -A FORWARD -p tcp --dport 22 -j ACCEPT
iptables -A FORWARD -p tcp --sport 22 -j ACCEPT
```

Теперь адрес сервера скрыт, а Netfilter разрешает трафик только к публичному серверу на порт 12345 и соединение по SSH на порт 22.

Последний шаг – настройка и подключение к PC5 с PC8 по протоколу SSH. Изначально необходимо создать ключ. На PC8 выполните команду:

```
ssh-keygen
```

При её вводе консоль предложит создать новый SSH-ключ, состоящий из публичной и приватной частей. Результат выполнения команды представлен на рисунке 5:

```
[root@elbrus ~]# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
/root/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:2YFcs6Xh5R2fGrw1mq+QPTQUEKeclQ5HHHnNlaRyLu8 root@elbrus.localdomain
The key's randomart image is:
+---[RSA 2048]----+
|          +oB**o=|
|       . +.X**o=+|
|        o ===*o+.|
|         o .=.B .|
|        S .. O   |
|            * o  |
|           o + . |
|            o o  |
|             E   |
+----[SHA256]-----+
```

Рисунок 5. Создание SSH ключа

После создания ключа, скопируем его на удалённую машину, чтобы в дальнейшем подключаться без пароля. Для этого используем команду на PC8:

```
ssh-copy-id user@10.10.6.6
```

где user –имя чётной записи на PC5, 10.10.6.6 – адрес маршрутизатора, который сделает перенаправление на сервер

При вводе команды копирования SSH-ключа на удалённую машину потребуется ввести пароль от пользователя, который был указан в команде (рисунок 6):

```
[root@elbrus ~]# ssh-copy-id user@10.10.6.6
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/root/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
user@10.10.6.6's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'user@10.10.6.6'"
and check to make sure that only the key(s) you wanted were added.
```

Рисунок 6. Передача ключа на удалённую машину

Для подключения используем команду:

```
ssh user@10.10.6.6
```

Для проверки правильности подключения, создайте удалённо текстовый документ на рабочем столе на PC5 (Рисунок 7). Для этого на PC8 введите:

```
cd 'Рабочий стол'
```

```
nano test
```

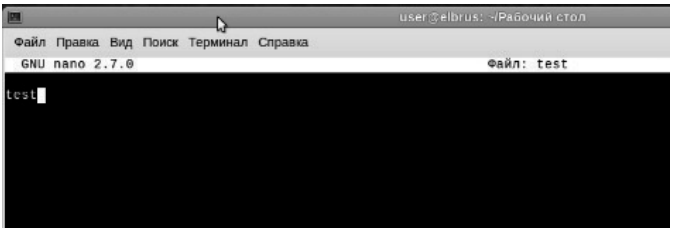


Рисунок 7. Создание документа

Далее сохраните документ нажав CTRL+O, enter. Теперь выйдите из текстового редактора нажав CTRL+X. Закрыть установленное соединение с PC5 можно нажав CTRL + D (Рисунок 8).

```
Connection to 10.10.6.6 closed.
[root@elbrus ~]#
```

Рисунок 8. Закрытие соединения

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Назначение действий DNAT и SNAT.
2. Функции и содержание таблицы nat.
3. Назначение и функционал утилиты iperf.
4. Назначение протокола SSH.
5. Назначение действий MASQUERADE и REDIRECT.

2. Программная платформа лабораторного практикума – операционная система ОС Альт

2.1. Процессы и файлы

ОС Альт Рабочая станция для Эльбрус является многопользовательской интегрированной системой. Это значит, что она разработана в расчете на одновременную работу нескольких пользователей.

Пользователь может либо сам работать в системе, выполняя некоторую последовательность команд, либо от его имени могут выполняться прикладные процессы.

Пользователь взаимодействует с системой через командный интерпретатор. Командный интерпретатор представляет собой прикладную программу, которая принимает от пользователя команды или набор команд и транслирует их в системные вызовы к ядру системы. Интерпретатор позволяет пользователю просматривать файлы, передвигаться по дереву файловой системы, запускать прикладные процессы. Все командные интерпретаторы UNIX имеют развитый командный язык и позволяют писать достаточно сложные программы, упрощающие процесс администрирования системы и работы с ней.

2.1.1. Процессы функционирования ОС

Все программы, которые выполняются в текущий момент времени, называются процессами. Процессы можно разделить на два основных класса: системные процессы и пользовательские процессы.

Системные процессы — программы, решающие внутренние задачи ОС, например, организацию виртуальной памяти на диске или предоставляющие пользователям те или иные сервисы (процессы-службы).

Пользовательские процессы — процессы, запускаемые пользователем из командного интерпретатора для решения задач пользователя или управления системными процессами. Linux изначально разрабатывался как многозадачная система. Он использует технологии, опробованные и отработанные другими реализациями UNIX, которые существовали ранее.

Фоновый режим работы процесса — режим, когда программа может работать без взаимодействия с пользователем. В случае необходимости интерактивной работы с пользователем (в общем случае) процесс будет «остановлен» ядром, и работа его продолжается только после переведения его в «нормальный» режим работы.

2.1.2. Файловая система ОС

В ОС использована файловая система Linux, которая, в отличие от файловых систем DOS и Windows(™), является единым деревом. Корень этого дерева — каталог, называемый root (рут) и обозначаемый /.

Части дерева файловой системы могут физически располагаться в разных разделах разных дисков или вообще на других компьютерах — для пользователя это прозрачно. Процесс присоединения файловой системы раздела к дереву называется монтированием, удаление — размонтированием. Например, файловая система CD-ROM в дистрибутиве монтируется по умолчанию в каталог /media/cdrom (путь в дистрибутиве обозначается с использованием /, а не \, как в DOS/Windows).

Текущий каталог обозначается ./.

2.1.3. Структура каталогов

Корневой каталог /:

- /bin — командные оболочки (shell), основные утилиты;
- /boot — содержит ядро системы;
- /dev — псевдофайлы устройств, позволяющие работать с устройствами напрямую. Файлы в /dev создаются сервисом udev
- /etc — общесистемные конфигурационные файлы для большинства программ в системе;
- /etc/init.d, /etc/rc.boot, /etc/rc.d — каталоги, где расположены командные файлы, выполняемые при запуске системы или при смене её режима работы;
- /etc/passwd — база данных пользователей, в которой содержится информация об имени пользователя, его настоящем имени, личном каталоге, его зашифрованный пароль и другие данные;
- /etc/shadow — теневая база данных пользователей. При этом информация из файла /etc/passwd перемещается в /etc/shadow, который недоступен для чтения всем, кроме пользователя root. В случае использования альтернативной схемы управления теневыми паролями (TCB), все теневые пароли для каждого пользователя располагаются в каталоге /etc/tcb/имя пользователя/shadow;
- /home — домашние каталоги пользователей;
- /lib — содержит файлы динамических библиотек, необходимых для работы большей части приложений, и подгружаемые модули ядра;
- /lost+found — восстановленные файлы;
- /media — подключаемые носители (каталоги для монтирования файловых систем сменных устройств);

• /mnt — точки временного монтирования;

• /opt — вспомогательные пакеты;

• /proc — виртуальная файловая система, хранящаяся в памяти компьютера при загруженной ОС. В данном каталоге расположены самые свежие сведения обо всех процессах, запущенных на компьютере.

• /root — домашний каталог администратора системы;

• /run — файлы состояния приложений;

• /sbin — набор программ для административной работы с системой (системные утилиты);

• /selinux — виртуальная файловая система SELinux;

• /srv — виртуальные данные сервисных служб;

• /sys — файловая система, содержащая информацию о текущем состоянии системы;

• /tmp — временные файлы.

• /usr — пользовательские двоичные файлы и данные, используемые только для чтения (программы и библиотеки);

• /var — файлы для хранения изменяющихся данных (рабочие файлы программ, очереди, журналы).

Каталог /usr:

• /usr/bin — дополнительные программы для всех учетных записей;

• /usr/sbin — команды, используемые при администрировании системы и не предназначенные для размещения в файловой системе root;

• /usr/local — место, где рекомендуется размещать файлы, установленные без использования пакетных менеджеров, внутренняя организация каталогов практически такая же, как и корневого каталога;

• /usr/man — каталог, где хранятся файлы справочного руководства man;

• /usr/share — каталог для размещения общедоступных файлов большей части приложений.

Каталог /var:

• /var/log — место, где хранятся файлы аудита работы системы и приложений;

• /var/spool — каталог для хранения файлов, находящихся в очереди на обработку для того или иного процесса (очереди печати, непрочитанные или не отправленные письма, задачи cron т.д.).

2.1.4. Организация файловой структуры

Система домашних каталогов пользователей помогает организовывать безопасную работу пользователей в многопользовательской системе. Вне

своего домашнего каталога пользователь обладает минимальными правами (обычно чтение и выполнение файлов) и не может нанести ущерб системе, например, удалив или изменив файл.

Кроме файлов, созданных пользователем, в его домашнем каталоге обычно содержатся персональные конфигурационные файлы некоторых программ.

Маршрут (путь) — это последовательность имён каталогов, представляющая собой путь в файловой системе к данному файлу, где каждое следующее имя отделяется от предыдущего наклонной чертой (слешем). Если название маршрута начинается со слеша, то путь в искомый файл начинается от корневого каталога всего дерева системы. В обратном случае, если название маршрута начинается непосредственно с имени файла, то путь к искомому файлу должен начаться от текущего каталога (рабочего каталога).

Имя файла может содержать любые символы за исключением косой черты (/). Однако следует избегать применения в именах файлов большинства знаков препинания и непечатаемых символов. При выборе имен файлов рекомендуется ограничиться следующими символами:

- строчные и ПРОПИСНЫЕ буквы. Следует обратить внимание на то, что регистр всегда имеет значение;
- символ подчеркивания (_);
- точка (.).

Для удобства работы точку можно использовать для отделения имени файла от расширения файла. Данная возможность может быть необходима пользователям или некоторым программам, но не имеет значение для shell.

2.1.5. Имена дисков и разделов

Все физические устройства вашего компьютера отображаются в каталог /dev файловой системы дистрибутива (об этом — ниже). Диски (в том числе IDE/SATA/SCSI/SAS жёсткие диски, USB-диски) имеют имена:

- /dev/sda — первый диск;
- /dev/sdb — второй диск;
- и т.д.

Диски обозначаются /dev/sdX, где X — a, b, c, d, e, … в зависимости от порядкового номера диска на шине.

Раздел диска обозначается числом после его имени. Например, /dev/sdb4 — четвертый раздел второго диска.

2.1.6. Разделы, необходимые для работы ОС

Для работы ОС на жестком диске (дисках) должны быть созданы, по крайней мере, два раздела: корневой (то есть тот, который будет содержать каталог /) и раздел для ядра (/boot). Если на диске много свободного места, то можно создать отдельные разделы для каталогов /usr, /home, /var.

2.2. Работа с наиболее часто используемыми компонентами

2.2.1. Виртуальная консоль

Система Альт Рабочая станция для Эльбрус предоставляет доступ к виртуальным консолям, с которых можно осуществлять одновременно несколько сеансов работы в системе (login session).

Только что установленная система Альт Рабочая станция для Эльбрус, возможно, предоставляет доступ только к первым шести виртуальным консолям, к которым можно обращаться, нажимая комбинации клавиш Alt+F1 — Alt+F6 (Ctrl+Alt+F1 — Ctrl+Alt+F6).

2.2.2. Командные оболочки (интерпретаторы)

Для управления ОС используются командные интерпретаторы (shell).

Зайдя в систему, Вы увидите приглашение — строку, содержащую символ «$» (далее этот символ будет обозначать командную строку). Программа ожидает ваших команд. Роль командного интерпретатора — передавать ваши команды операционной системе. По своим функциям он соответствует command.com в DOS, но несравненно мощнее. При помощи командных интерпретаторов можно писать небольшие программы — сценарии (скрипты). В Linux доступны следующие командные оболочки:

- bash — самая распространенная оболочка под linux. Она ведет историю команд и предоставляет возможность их редактирования;
- pdksh — клон korn shell, хорошо известной оболочки в UNIX™ системах.

Проверить, какая оболочка используется в данный момент можно, выполнив команду:

```
$ echo $SHELL
```

Оболочкой по умолчанию является Bash (Bourne Again Shell) — самая распространённая оболочка под Linux, которая ведет историю команд и предоставляет возможность их редактирования. В дальнейшем описании работы с Альт Рабочая станция для Эльбрус будут использоваться примеры с использованием этой оболочки.

2.3. Командная оболочка Bash

В Bash имеется несколько приемов для работы со строкой команд. Например, можно использовать следующие сочетания:

- Ctrl+A — перейти на начало строки;
- Ctrl+U — удалить текущую строку;
- Ctrl+C — остановить текущую задачу.

Для ввода нескольких команд одной строкой можно использовать разделитель «;». По истории команд можно перемещаться с помощью клавиш ↑ («вверх») и ↓ («вниз»).

Чтобы найти конкретную команду в списке набранных, не пролистывая всю историю, можно нажать Ctrl+R и начать вводить символы ранее введенной команды.

Для просмотра истории команд можно воспользоваться командой history. Команды, присутствующие в истории, отображаются в списке пронумерованными. Чтобы запустить конкретную команду необходимо набрать:

!номер команды

Если ввести:

!!

запустится последняя из набранных команд.

В Bash имеется возможность самостоятельного завершения имен команд из общего списка команд, что облегчает работу при вводе команд, в случае, если имена программ и команд слишком длинны. При нажатии клавиши Tab Bash завершает имя команды, программы или каталога, если не существует нескольких альтернативных вариантов. Например, чтобы использовать программу декомпрессии gunzip, можно набрать следующую команду:

gu

Затем нажать клавишу Tab. Так как в данном случае существует несколько возможных вариантов завершения команды, то необходимо повторно нажать клавишу Tab, чтобы получить список имен, начинающихся с gu.

В предложенном примере можно получить следующий список:

$ gu

guile gunzip gupnp-binding-tool

Если набрать: n (gunzip — это единственное имя, третьей буквой которого является «n»), а затем нажать клавишу Tab, то оболочка самостоятельно дополнит имя. Чтобы запустить команду нужно нажать Enter.

Программы, вызываемые из командной строки, Bash ищет в каталогах, определяемых в системной переменной $PATH. По умолчанию в этот перечень

каталогов не входит текущий каталог, обозначаемый: ./ (точка слеш) (если только не выбран один из двух самых слабых уровней защиты). Поэтому, для запуска программы из текущего каталога, необходимо использовать команду (в примере запускается команда prog):

./prog

2.4. Обзор основных команд системы

Все команды, приведенные ниже, могут быть запущены в режиме консоли. Для получения более подробной информации используйте команду man. Пример:

$ man ls

Примечание

Параметры команд обычно начинаются с символа «-», и обычно после одного символа «-» можно указать сразу несколько опций. Например, вместо команды ls -l -F можно ввести команду ls -lF

Учетные записи пользователей

Команда su

Позволяет получить права администратора. При вводе команды su, будет запрошен пароль суперпользователя (root), и, в случае ввода корректного пароля, оператор получит права администратора. Чтобы вернуться к правам оператора, необходимо ввести команду:

exit

Команда id

Команда id выводит информацию о пользователе и группах, в которых он состоит для заданного пользователя или о текущем пользователе (если ничего не указано).

Синтаксис:

id [ОПЦИИ...] [ПОЛЬЗОВАТЕЛЬ]

Команда passwd

Команда passwd меняет (или устанавливает) пароль, связанный с входным_именем пользователя.

Обычный пользователь может менять только пароль, связанный с его собственным входным_именем.

Команда запрашивает у обычных пользователей старый пароль (если он был), а затем дважды запрашивает новый. Новый пароль должен соответствовать техническим требованиям к паролям, заданным администратором системы.

Основные операции с файлами и каталогами

Команда ls

Команда ls (list) печатает в стандартный вывод содержимое каталогов.

Синтаксис:

ls [ОПЦИИ...] [ФАЙЛ...]

Основные опции:

- -a — просмотр всех файлов, включая скрытые;
- -l — отображение более подробной информации;
- -R — выводить рекурсивно информацию о подкаталогах.

Команда cd

Команда cd предназначена для смены каталога. Команда работает как с абсолютными, так и с относительными путями. Если каталог не указан, используется значение переменной окружения $HOME (домашний каталог пользователя). Если каталог задан полным маршрутным именем, он становится текущим. По отношению к новому каталогу нужно иметь право на выполнение, которое в данном случае трактуется как разрешение на поиск.

Синтаксис:

cd [-L|-P] [КАТАЛОГ]

Если в качестве аргумента задано «-», то это эквивалентно $OLDPWD. Если переход был осуществлен по переменной окружения $CDPATH или в качестве аргумента был задан «-» и смена каталога была успешной, то абсолютный путь нового рабочего каталога будет выведен на стандартный вывод.

Пример. Сделать текущим каталог /usr/bin:

cd /usr/bin/

Сделать текущим родительский каталог:

cd ..

Вернуться в предыдущий каталог:

cd -

Команда pwd

Команда pwd выводит абсолютный путь текущего (рабочего) каталога.

Синтаксис:

pwd [-L|-P]

Опции:

- -P — не выводить символические ссылки;
- -L — выводить символические ссылки.

Команда rm

Команда rm служит для удаления записей о файлах. Если заданное имя было последней ссылкой на файл, то файл уничтожается.

Предупреждение

Удалив файл, вы не сможете его восстановить!

Синтаксис:

rm [ОПЦИИ...] <ФАЙЛ>

Основные опции:

• -f — никогда не запрашивать подтверждения;

• -i — всегда запрашивать подтверждение;

• -r, -R — рекурсивно удалять содержимое указанных каталогов.

Пример. Удалить все файлы html в каталоге ~/html:

rm -i ~/html/*.html

Команда mkdir

mkdir — команда для создания новых каталогов.

Синтаксис:

mkdir [-p] [-m права] <КАТАЛОГ...>

Команда rmdir

Команда rmdir удаляет каталоги из файловой системы. Каталог должен быть пуст перед удалением.

Синтаксис:

rmdir [ОПЦИИ] <КАТАЛОГ...>

Основные опции:

• -p — удалить каталог и его потомки.

Команда rmdir часто заменяется командой rm -rf, которая позволяет удалять каталоги, даже если они не пусты.

Команда cp

Команда cp предназначена для копирования файлов из одного в другие каталоги.

Синтаксис:

cp [-fip] [ИСХ_ФАЙЛ...] [ЦЕЛ_ФАЙЛ...]

cp [-fip] [ИСХ_ФАЙЛ...] [КАТАЛОГ]

cp [-R] [[-H] | [-L] | [-P]] [-fip] [ИСХ_ФАЙЛ...] [КАТАЛОГ]

Основные опции:

• -p — сохранять по возможности времена изменения и доступа к файлу, владельца и группу, права доступа;

• -i — запрашивать подтверждение перед копированием в существующие файлы;

• -r, -R — рекурсивно копировать содержимое каталогов.

Команда mv

Команда mv предназначена для перемещения файлов.

Синтаксис:

mv [-fi] [ИСХ_ФАЙЛ...] [ЦЕЛ_ФАЙЛ...]

mv [-fi] [ИСХ_ФАЙЛ...] [КАТАЛОГ]

В первой синтаксической форме, характеризующейся тем, что последний операнд не является ни каталогом, ни символической ссылкой на каталог, mv перемещает исх_файл в цел_файл (происходит переименование файла).

Во второй синтаксической форме mv перемещает исходные файлы в указанный каталог под именами, совпадающими с краткими именами исходных файлов.

Основные опции:

- -f — не запрашивать подтверждения перезаписи существующих файлов;
- -i — запрашивать подтверждение перезаписи существующих файлов.

Команда cat

Команда cat последовательно выводит содержимое файлов.

Синтаксис:

cat [ОПЦИИ] [ФАЙЛ...]

Основные опции:

- -n, --number — нумеровать все строки при выводе;
- -E, --show-ends — показывать $ в конце каждой строки.

Если файл не указан, читается стандартный ввод. Если в списке файлов присутствует имя «-», вместо этого файла читается стандартный ввод.

Команда head

Команда head выводит первые 10 строк каждого файла на стандартный вывод.

Синтаксис:

head [ОПЦИИ] [ФАЙЛ...]

Основные опции:

- -n, --lines=[-]K — вывести первые K строк каждого файла, а не первые 10;
- -q, --quiet — не печатать заголовки с именами файлов.

Команда chmod

Команда chmod предназначена для изменения прав доступа файлов и каталогов.

Синтаксис:

chmod [ОПЦИИ] РЕЖИМ[,РЕЖИМ]... <ФАЙЛ>

chmod [ОПЦИИ] --reference=ИФАЙЛ <ФАЙЛ>

Основные опции:

- -R — рекурсивно изменять режим доступа к файлам, расположенным в указанных каталогах;

• --reference=ИФАЙЛ — использовать режим файла ИФАЙЛ.

chmod изменяет права доступа каждого указанного файла в соответствии с правами доступа, указанными в параметре режим, который может быть представлен как в символьном виде, так и в виде восьмеричного, представляющего битовую маску новых прав доступа.

Формат символьного режима следующий:

[ugoa...][[+-=][разрешения...]...]

Здесь разрешения — это ноль или более букв из набора «rwxXst» или одна из букв из набора «ugo».

Каждый аргумент — это список символьных команд изменения прав доступа, разделены запятыми. Каждая такая команда начинается с нуля или более букв «ugoa», комбинация которых указывает, чьи права доступа к файлу будут изменены: пользователя, владеющего файлом (u), пользователей, входящих в группу, к которой принадлежит файл (g), остальных пользователей (o) или всех пользователей (a). Если не задана ни одна буква, то автоматически будет использована буква «a», но биты, установленные в umask, не будут затронуты.

Оператор «+» добавляет выбранные права доступа к уже имеющимся у каждого файла, «-» удаляет эти права. «=» присваивает только эти права каждому указанному файлу.

Буквы «rwxXst» задают биты доступа для пользователей: «r» — чтение, «w» — запись, «x» — выполнение (или поиск для каталогов), «X» — выполнение/поиск только если это каталог или же файл с уже установленным битом выполнения, «s» — задать ID пользователя и группы при выполнении, «t» — запрет удаления.

Примеры. Позволить всем выполнять файл f2:

chmod +x f2

Запретить удаление файла f3:

chmod +t f3

Команда chown

Команда chown изменяет владельца и/или группу для каждого заданного файла.

Синтаксис:

chown [КЛЮЧ]…[ВЛАДЕЛЕЦ][:[ГРУППА]] <ФАЙЛ>

Изменить владельца может только владелец файла или суперпользователь. Владелец не изменяется, если он не задан в аргументе. Группа также не изменяется, если не задана, но, если после символьного ВЛАДЕЛЬЦА стоит символ «:», подразумевается изменение группы на основную группу текущего

пользователя. Поля ВЛАДЕЛЕЦ и ГРУППА могут быть как числовыми, так и символьными.

Примеры: 1. Поменять владельца каталога /u на пользователя test:

chown test /u

2. Поменять владельца и группу каталога /u:

chown test:staff /u

3. Поменять владельца каталога /u и вложенных файлов на test:

chown -hR test /u

Поиск файлов

Команда find

Команда find предназначена для поиска всех файлов, начиная с корневого каталога. Поиск может осуществляться по имени, типу или владельцу файла.

Синтаксис:

find [-H] [-L] [-P] [-Оуровень] [-D help|tree|search|stat|rates|opt|exec] [ПУТЬ…] [ВЫРАЖЕНИЕ]

Ключи для поиска:

- -name — поиск по имени файла;
- -type — поиск по типу f=файл, d=каталог, l=ссылка(lnk);
- -user — поиск по владельцу (имя или UID).

Когда выполняется команда find, можно выполнять различные действия над найденными файлами. Основные действия:

- -exec команда \; — выполнить команду. Запись команды должна заканчиваться экранированной точкой с запятой. Строка «{}» заменяется текущим маршрутным именем файла;
- execdir команда \; — то же самое что и -exec, но команда вызывается из подкаталога, содержащего текущий файл;
- -ok команда — эквивалентно -exec за исключением того, что перед выполнением команды запрашивается подтверждение (в виде сгенерированной командной строки со знаком вопроса в конце) и она выполняется только при ответе: y;
- -print — вывод имени файла на экран.

Путем по умолчанию является текущий подкаталог. Выражение по умолчанию -print.

Примеры. Найти в текущем каталоге обычные файлы (не каталоги), имя которых начинается с символа «~»:

find . -type f -name "~*" -print

Найти в текущем каталоге файлы, измененные позже, чем файл file.bak:

find . -newer file.bak -type f -print

Удалить все файлы с именами a.out или *.o, доступ к которым не производился в течение недели:

find / \(-name a.out -o -name '*.o' \) \ -atime +7 -exec rm {} \;

Удалить из текущего каталога и его подкаталогов все файлы нулевого размера, запрашивая подтверждение:

find . -size 0c -ok rm {} \;

Команда whereis

whereis сообщает путь к исполняемому файлу программы, ее исходным файлам (если есть) и соответствующим страницам справочного руководства.

Синтаксис:

whereis [ОПЦИИ] <ИМЯ>

Опции:

- -b — вывод информации только об исполняемых файлах;
- -m — вывод информации только о страницах справочного руководства;
- -s — вывод информации только об исходных файлах.

Мониторинг и управление процессами

Команда ps

Команда ps отображает список текущих процессов.

Синтаксис:

ps [ОПЦИИ]

По умолчанию выводится информация о процессах с теми же действующим UID и управляющим терминалом, что и у подающего команду пользователя.

Основные опции:

- -a — вывести информацию о процессах, ассоциированных с терминалами;
- -f — вывести «полный» список;
- -l — вывести «длинный» список;
- -p список — вывести информацию о процессах с перечисленными в списке PID;
- -u список — вывести информацию о процессах с перечисленными идентификаторами или именами пользователей.

Команда kill

Команда kill позволяет прекратить исполнение процесса или передать ему сигнал.

Синтаксис:

kill [-s] [сигнал] [идентификатор] [...]

kill [-l] [статус_завершения]

kill [-номер_сигнала] [идентификатор] [...]

Идентификатор — PID ведущего процесса задания или номер задания, предварённый знаком «%».

Основные опции:

- -l — вывести список поддерживаемых сигналов;
- -s сигнал, -сигнал — послать сигнал с указанным именем.

Если обычная команда kill не дает желательного эффекта, необходимо использовать команду kill с параметром -9 (kill -9 PID_номер).

Команда df

Команда df показывает количество доступного дискового пространства в файловой системе, в которой содержится файл, переданный как аргумент. Если ни один файл не указан, показывается доступное место на всех смонтированных файловых системах. Размеры по умолчанию указаны в блоках по 1КБ.

Синтаксис:

df [ОПЦИИ] [ФАЙЛ...]

Основные опции:

- --total — подсчитать общий объем в конце;
- -h, --human-readable — печатать размеры в удобочитаемом формате (например, 1K 234M 2G);
- -h, --human-readable — печатать размеры в удобочитаемом формате (например, 1K 234M 2G).

Команда du

Команда du подсчитывает использование диска каждым файлом, для каталогов подсчет происходит рекурсивно.

Синтаксис:

du [ОПЦИИ] [ФАЙЛ...]

Основные опции:

- -a, --all — выводить общую сумму для каждого заданного файла, а не только для каталогов;
- -c, --total — подсчитать общий объем в конце. Может быть использовано для выяснения суммарного использования дискового пространства для всего списка заданных файлов;
- -d, --max-depth=N — выводить объем для каталога (или файлов, если указано --all) только если она на N или менее уровней ниже аргументов командной строки;
- -S, --separate-dirs — выдавать отдельно размер каждого каталога, не включая размеры подкаталогов;
- -s, --summarize — отобразить только сумму для каждого аргумента.

Команда which

Команда which отображает полный путь к указанным командам или сценариям.

Синтаксис:

which [ОПЦИИ] <ФАЙЛ...>

Основные опции:

• -a, --all — выводит все совпавшие исполняемые файлы по содержимому в переменной окружения $PATH, а не только первый из них;

• -c, --total — подсчитать общий объем в конце. Может быть использовано для выяснения суммарного использования дискового пространства для всего списка заданных файлов;

• -d, --max-depth=N — выводить объем для каталога (или файлов, если указано --all) только если она на N или менее уровней ниже аргументов командной строки;

• -S, --separate-dirs — выдавать отдельно размер каждого каталога, не включая размеры подкаталогов;

• --skip-dot — пропускает все каталоги из переменной окружения $PATH, которые начинаются с точки.

Использование многозадачности

Альт Рабочая станция для Эльбрус — это многозадачная система.

Для того, чтобы запустить программу в фоновом режиме, необходимо набрать «&» после имени программы. После этого оболочка даст возможность запускать другие приложения.

Так как некоторые программы интерактивны — их запуск в фоновом режиме бессмысленен. Подобные программы просто остановятся, если их запустить в фоновом режиме.

Можно также запускать нескольких независимых сеансов. Для этого в консоли необходимо набрать Alt и одну из клавиш, находящихся в интервале от F1 до F6. На экране появится новое приглашение системы, и можно открыть новый сеанс. Этот метод также позволяет вам работать на другой консоли, если консоль, которую вы использовали до этого, не отвечает или вам необходимо остановить зависшую программу.

Команда bg

Команда bg позволяет перевести задание на задний план.

Синтаксис:

bg [ИДЕНТИФИКАТОР ...]

Идентификатор — PID ведущего процесса задания или номер задания, предварённый знаком «%».

Команда fg

Команда fg позволяет перевести задание на передний план.

Синтаксис:

fg [ИДЕНТИФИКАТОР ...]

Идентификатор — PID ведущего процесса задания или номер задания, предварённый знаком «%».

Сжатие и упаковка файлов

Команда tar

Сжатие и упаковка файлов выполняется с помощью команды tar, которая преобразует файл или группу файлов в архив без сжатия (tarfile).

Упаковка файлов в архив чаще всего выполняется следующей командой:

tar -cf [имя создаваемого файла архива] [упаковываемые файлы и/или каталоги]

Пример использования команды упаковки архива:

tar -cf moi_dokumenti.tar Docs project.tex

Распаковка содержимого архива в текущий каталог выполняется командой:

tar -xf [имя файла архива]

Для сжатия файлов используются специальные программы сжатия: gzip, xz.

2.3. Стыкование команд в системе Linux

2.3.1. Стандартный ввод и стандартный вывод

Многие команды системы имеют так называемые стандартный ввод (standard input) и стандартный вывод (standard output), часто сокращаемые до stdin и stdout. Ввод и вывод здесь — это входная и выходная информация для данной команды. Программная оболочка делает так, что стандартным вводом является клавиатура, а стандартным выводом — экран монитора.

Пример с использованием команды cat. По умолчанию команда cat читает данные из всех файлов, которые указаны в командной строке, и посылает эту информацию непосредственно в стандартный вывод (stdout). Следовательно, команда:

cat history-final masters-thesis

выведет на экран сначала содержимое файла history-final, а затем — файла masters-thesis.

Если имя файла не указано, программа cat читает входные данные из stdin и возвращает их в stdout. Пример:

cat
Hello there.
Hello there.
Bye.
Bye.
Ctrl-D

Каждую строку, вводимую с клавиатуры, программа cat немедленно возвращает на экран. При вводе информации со стандартного ввода конец текста сигнализируется вводом специальной комбинации клавиш, как правило, Ctrl+D. Сокращённое название сигнала конца текста — EOT (end of text).

3.2. Перенаправление ввода и вывода

При необходимости можно перенаправить стандартный вывод, используя символ >, и стандартный ввод, используя символ <.

Фильтр (filter) — программа, которая читает данные из стандартного ввода, некоторым образом их обрабатывает и результат направляет на стандартный вывод. Когда применяется перенаправление, в качестве стандартного ввода и вывода могут выступать файлы. Как указывалось выше, по умолчанию, stdin и stdout относятся к клавиатуре и к экрану соответственно. Программа sort является простым фильтром — она сортирует входные данные и посылает результат на стандартный вывод. Совсем простым фильтром является программа cat — она ничего не делает с входными данными, а просто пересылает их на выход.

Пред.
• Документация
• След.

2.3.3. Использование состыкованных команд

Стыковку команд (pipelines) осуществляет командная оболочка, которая stdout первой команды направляет на stdin второй команды. Для стыковки используется символ |. Направить stdout команды ls на stdin команды sort:

ls | sort -r
notes
masters-thesis
history-final
english-list

Вывод списка файлов частями:

ls /usr/bin | more

Если необходимо вывести на экран последнее по алфавиту имя файла в текущем каталоге, можно использовать следующую команду:

ls | sort -r | head -1 notes

где команда head -1 выводит на экран первую строку получаемого ей входного потока строк (в примере поток состоит из данных от команды ls), отсортированных в обратном алфавитном порядке.

3.4. Недеструктивное перенаправление вывода

Эффект от использования символа > для перенаправления вывода файла является деструктивным; т.е, команда

ls > file-list

уничтожит содержимое файла file-list, если этот файл ранее существовал, и создаст на его месте новый файл. Если вместо этого перенаправление будет сделано с помощью символов >>, то вывод будет приписан в конец указанного файла, при этом исходное содержимое файла не будет уничтожено.

Примечание

Перенаправление ввода и вывода и стыкование команд осуществляется командными оболочками, которые поддерживают использование символов >, >> и |. Сами команды не способны воспринимать и интерпретировать эти символы.

2.4. Режим суперпользователя

2.4.1. Какие бывают пользователи?

Linux — система многопользовательская, а потому пользователь — ключевое понятие для организации всей системы доступа в Linux. Файлы всех пользователей в Linux хранятся раздельно, у каждого пользователя есть собственный домашний каталог, в котором он может хранить свои данные. Доступ других пользователей к домашнему каталогу пользователя может быть ограничен.

Суперпользователь в Linux — это выделенный пользователь системы, на которого не распространяются ограничения прав доступа. Именно суперпользователь имеет возможность произвольно изменять владельца и группу файла. Ему открыт доступ на чтение и запись к любому файлу или каталогу системы.

Среди учётных записей Linux всегда есть учётная запись суперпользователя — root. Поэтому вместо «суперпользователь» часто говорят «root». Множество системных файлов принадлежат root, множество файлов

только ему доступны для чтения или записи. Пароль этой учётной записи — одна из самых больших драгоценностей системы. Именно с её помощью системные администраторы выполняют самую ответственную работу.

2.4.2. Для чего может понадобиться режим суперпользователя?

Системные утилиты, например, такие, как Центр управления системой или Программа управления пакетами Synaptic требуют для своей работы привилегий суперпользователя, потому что они вносят изменения в системные файлы. При их запуске выводится диалоговое окно с запросом пароля системного администратора.

3. Как получить права суперпользователя?

Для опытных пользователей, умеющих работать с командной строкой, существует два различных способа получить права суперпользователя.

Первый — это зарегистрироваться в системе под именем root.

Второй способ — воспользоваться специальной утилитой su (shell of user), которая позволяет выполнить одну или несколько команд от лица другого пользователя. По умолчанию эта утилита выполняет команду sh от пользователя root, то есть запускает командный интерпретатор. Отличие от предыдущего способа в том, что всегда известно, кто именно запускал su, а значит, ясно, кто выполнил определённое административное действие.

В некоторых случаях удобнее использовать не su, а утилиту sudo, которая позволяет выполнять только заранее заданные команды.

Важно

Для того чтобы воспользоваться командами su и sudo, необходимо быть членом группы wheel. Пользователь, созданный при установке системы, по умолчанию уже включён в эту группу.

В дистрибутивах Альт для управления доступом к важным службам используется подсистема control. control — механизм переключения между неким набором фиксированных состояний для задач, допускающих такой набор.

Команда control доступна только для суперпользователя (root). Для того, чтобы посмотреть, что означает та или иная политика control (разрешения выполнения конкретной команды, управляемой control), надо запустить команду с ключом help:

```
# control su help
```

Запустив control без параметров, можно увидеть полный список команд, управляемых командой (facilities) вместе с их текущим состоянием и набором допустимых состояний.

.4. Как перейти в режим суперпользователя?

Для перехода в режим суперпользователя наберите в терминале команду su -.

Если воспользоваться командой su без ключа, то происходит вызов командного интерпретатора с правами root. При этом значение переменных окружения, в частности $PATH, остаётся таким же, как у пользователя: в переменной $PATH не окажется каталогов /sbin, /usr/sbin, без указания полного имени будут недоступны команды route, shutdown, mkswap и другие. Более того, переменная $HOME будет указывать на каталог пользователя, все программы, запущенные в режиме суперпользователя, сохранят свои настройки с правами root в каталоге пользователя, что в дальнейшем может вызвать проблемы.

Чтобы избежать этого, следует использовать su -. В этом режиме su запустит командный интерпретатор в качестве login shell, и он будет вести себя в точности так, как если бы в системе зарегистрировался root.

2.5. Управление пользователями

2.5.1. Общая информация

Пользователи и группы внутри системы обозначаются цифровыми идентификаторами — UID и GID, соответственно.

Пользователь может входить в одну или несколько групп. По умолчанию он входит в группу, совпадающую с его именем. Чтобы узнать, в какие еще группы входит пользователь, введите команду id, вывод её может быть примерно следующим:

uid=500(test) gid=500(test) группы=500(test),16(rpm)

Такая запись означает, что пользователь test (цифровой идентификатор 500) входит в группы test и rpm. Разные группы могут иметь разный уровень доступа к тем или иным каталогам; чем в большее количество групп входит пользователь, тем больше прав он имеет в системе.

Примечание

В связи с тем, что большинство привилегированных системных утилит в дистрибутивах Альт имеют не SUID-, а SGID-бит, будьте предельно

внимательны и осторожны в переназначении групповых прав на системные каталоги.

Команда passwd

Команда passwd поддерживает традиционные опции passwd и утилит shadow.

Синтаксис:

passwd [ОПЦИИ...] [ИМЯ ПОЛЬЗОВАТЕЛЯ]

Возможные опции:

- -d --delete — удалить пароль для указанной записи;
- -f, --force — форсировать операцию;
- -k, --keep-tokens — сохранить не устаревшие пароли;
- -l, --lock — блокировать указанную запись;
- --stdin — прочитать новые пароли из стандартного ввода;
- -S, --status — дать отчет о статусе пароля в указанной записи;
- -u, --unlock — разблокировать указанную запись;
- -?, --help — показать справку и выйти;
- --usage — дать короткую справку по использованию;
- -V, --version — показать версию программы и выйти.

Код выхода: при успешном завершении passwd заканчивает работу с кодом выхода 0. Код выхода 1 означает, что произошла ошибка. Текстовое описание ошибки выводится на стандартный поток ошибок.

Добавления нового пользователя

Для добавления нового пользователя используйте команды useradd и passwd:

```
# useradd test1
# passwd test1
passwd: updating all authentication tokens for user test1.
You can now choose the new password or passphrase.
A valid password should be a mix of upper and lower case letters,
digits, and other characters.  You can use an 8-character long
password with characters from at least 3 of these 4 classes, or
a 7character long password containing characters from all the
classes.  An upper case letter that begins the password and a
digit that ends it do not count towards the number of character
classes used.
A passphrase should be of at least 3 words, 11 to 40 characters
```

long, and contain enough different characters.

Alternatively, if no one else can see your terminal now, you can

pick this as your password: "holder5dinghy-Arm".

Enter new password:

В результате описанных действий в системе появился пользователь test1 с некоторым паролем. Если пароль оказался слишком слабым с точки зрения системы, она об этом предупредит (как в примере выше). Пользователь в дальнейшем может поменять свой пароль при помощи команды passwd — но, если он попытается поставить слабый пароль, система откажет ему (в отличие от root) в изменении.

В Альт Рабочая станция для Эльбрус для проверки паролей на слабость используется модуль PAM passwdqc.

Программа useradd имеет множество параметров, которые позволяют менять её поведение по умолчанию. Например, можно принудительно указать, какой будет UID или какой группе будет принадлежать пользователь.

Модификация пользовательских записей

Для модификации пользовательских записей применяется утилита usermod:

```
# usermod -G audio,rpm,test1 test1
```

Такая команда изменит список групп, в которые входит пользователь test1 — теперь это audio, rpm, test1.

```
# usermod -l test2 test1
```

Будет произведена смена имени пользователя с test1 на test2.

Команды usermod -L test2 и usermod -U test2 соответственно временно блокируют возможность входа в систему пользователю test2 и возвращают всё на свои места.

Изменения вступят в силу только при следующем входе пользователя в систему.

При неинтерактивной смене или задании паролей для целой группы пользователей используйте утилиту chpasswd. На стандартный вход ей следует подавать список, каждая строка которого будет выглядеть как имя:пароль.

Удаление пользователей

Для удаления пользователей используйте userdel.

Команда userdel test2 удалит пользователя test2 из системы. Если будет дополнительно задан параметр -d, то будет уничтожен и домашний каталог

пользователя. Нельзя удалить пользователя, если в данный момент он еще работает в системе.

Утилиты vigr и vipw используются для ручного редактирования файлов /etc/passwd и /etc/group, в которых хранятся основные записи о пользователях и группах в системе.

Не рекомендуется создавать пользователей с правами сверх необходимых. Предпочтительнее создать серию новых групп и включить в них требуемого пользователя. А для данных групп установить соответствующие права на объектах файловой системы (утилиты chmod и chown).

2.6. Документация

Каждый объект системы Linux обязательно сопровождается документацией, описывающей их назначение и способы использования. От пользователя системы не требуется заучивать все возможные варианты взаимодействия с ней. Достаточно понимать основные принципы её устройства и уметь находить справочную информацию.

Не пренебрегайте чтением документации: она поможет вам избежать многих сложностей, сэкономить массу времени и усилий при установке, настройке и администрировании системы, поможет найти нужное для работы приложение и быстро разобраться в нём.

2.6.1. Экранная документация

Почти все системы семейства UNIX, включая систему Linux, имеют экранную документацию. Её тексты содержат документацию по системным командам, ресурсам, конфигурационным файлам и т. д., а также могут быть выведены на экран в процессе работы.

2.6.1.1. Команда man

Для доступа к экранной документации используется команда man (сокращение от manual). Каждая страница руководства посвящена одному объекту системы. Для того чтобы прочесть страницу руководства по программе, необходимо набрать man название_программы. К примеру, если вы хотите узнать, какие опции есть у команды date, вы можете ввести команду:

```
$ man date
```

Большинство экранной документации написано для пользователей, имеющих некоторое представление о том, что делает данная команда. Поэтому большинство текстов экранной документации содержит исключительно

технические детали команды без особых пояснений. Тем не менее, экранная документация оказывается очень ценной в том случае, если вы помните название команды, но её синтаксис просто выпал у вас из памяти.

Поиск по описаниям man осуществляется командой apropos. Если вы точно не знаете, как называется необходимая вам программа, то поиск осуществляется по ключевому слову, к примеру, apropos date или при помощи ввода слова, обозначающего нужное действие, после команды man -k (например, man -k copy). Слово, характеризующее желаемое для вас действие, можно вводить и на русском языке. При наличии русского перевода страниц руководства man результаты поиска будут выведены на запрашиваемом языке.

«Страница руководства» занимает, как правило, больше одной страницы экрана. Для того чтобы читать было удобнее, man запускает программу постраничного просмотра текстов. Страницы перелистывают пробелом, для выхода из режима чтения описания команд man необходимо нажать на клавиатуре q. Команда man man выдаёт справку по пользованию самой командой man.

Документация в подавляющем большинстве случаев пишется на простом английском языке. Необходимость писать на языке, который будет более или менее понятен большинству пользователей, объясняется постоянным развитием Linux. Дело не в том, что страницу руководства нельзя перевести, а в том, что её придётся переводить всякий раз, когда изменится описываемый ею объект! Например, выход новой версии программного продукта сопровождается изменением его возможностей и особенностей работы, а, следовательно, и новой версией документации.

Тем не менее, некоторые наиболее актуальные руководства существуют в переводе на русский язык. Свежие версии таких переводов на русский язык собраны в пакете man-pages-ru. Установив этот пакет, вы добавите в систему руководства, для которых есть перевод, и man по умолчанию будет отображать их на русском языке.

2.6.1.2. Команда info

Другой источник информации о Linux и составляющих его программах — справочная подсистема info. Страница руководства, несмотря на обилие ссылок различного типа, остаётся «линейным» текстом, структурированным только логически. Документ info — это настоящий гипертекст, в котором множество небольших страниц объединены в дерево. В каждом разделе документа info всегда есть оглавление, из которого можно перейти к нужному подразделу, а затем вернуться обратно (ссылки для перемещения по разделам

текста помечены *). Для получения вспомогательной информации о перемещении по тексту используйте клавишу h. Полное руководство info вызывается командой info info. Команда info, введённая без параметров, предлагает пользователю список всех документов info, установленных в системе.

.2. Документация по пакетам

Дополнительным источником информации об интересующей вас программе, в основном на английском языке, является каталог /usr/share/doc — место хранения разнообразной документации.

Каждый пакет также содержит поставляемую вместе с включённым в него ПО документацию, располагающуюся обычно в каталоге /usr/share/doc/имя_пакета. Например, документация к пакету foo-1.0-alt1 находится в /usr/share/doc/foo-1.0-alt1. Для получения полного списка файлов документации, относящихся к пакету, воспользуйтесь командой rpm -qd имя_установленного_пакета.

В документации к каждому пакету вы можете найти такие файлы как README, FAQ, TODO, ChangeLog и другие. В файле README содержится основная информация о программе — имя и контактные данные авторов, назначение, полезные советы и пр. FAQ содержит ответы на часто задаваемые вопросы; этот файл стоит прочитать в первую очередь, если у вас возникли проблемы или вопросы по использованию программы, поскольку большинство проблем и сложностей типичны, вполне вероятно, что в FAQ вы тут же найдёте готовое решение. В файле TODO записаны планы разработчиков на реализацию той или иной функциональности. В файле ChangeLog записана история изменений в программе от версии к версии.

Для поиска внешней информации о программе, например, адреса сайта программы в сети Интернет можно использовать команду rpm -qi имя_установленного_пакета. В информационном заголовке соответствующего пакета, среди прочей информации, будет выведена искомая ссылка.

Возможно, будет полезно знать расположение собрания практических рекомендаций по самым различным вопросам, связанным с использованием Linux. Файлы HOWTO в формате HTML (от англ. how to — «как сделать») каталога /usr/share/doc/HOWTO/ (при условии их наличия в системе) содержат многообразную информацию о работе Linux-систем.

2.7. Базовые команды ОС Альт

2.7.1. Команды для получения справочной информации

`man command` – показать руководство (man-страницу) для command.

`info command` – показать info-страницу для command.

`whatis command` – поиск man-страницы для command и отображение информации об имени из соответствующей man-страницы.

`apropos command` – поиск, по ключевым словам, в man-страницах и вывод тех, которые содержат command (аналог команды man -k, где ключ k - указывает на поиск в руководстве по ключевому слову).

`editor` – отображает имеющиеся в системе текстовые редакторы.

2.7.2. Команды для получения системной информации

`date` – вывести текущую дату и время.

`cal` – вывести календарь на текущий месяц.

`uptime` – показать текущий «аптайм» - время непрерывной работы системы.

`uname` – показать информацию о ядре.

`-s` – показать имя ядра (информация выдается по умолчанию, если ни одна опция не указана).

`-n` – показать имя хоста.

`-r` – показать номер выпуска ядра. Эта опция часто используется с командами управления модулями.

`-v` – показать версию ядра.

`-m` – показать имя аппаратной платформы.

`-o` – показать имя операционной системы.

`-a` – показать всю возможную информацию.

`cat /proc/cpuinfo` – показать информацию о ЦПУ.

`cat /proc/meminfo` – показать информацию о памяти.

`df` – показать информацию об использовании дисков.

`-t` – указание типа файловой системы.

`du` – показать объём текущего каталога.

`-s` – с подкаталогами.

`whereis file` – отображение расположения файла file.

`which app` – отображает путь к команде app.

`enable` – вывод встроенных в интерпретатор команд.

`lshw` – обтображение списка устройств.

`lsmod` – список подключенных модулей ядра.
`modprobe modulemane` – добавления и удаления модулей ядра Linux.
`modinfo modulename` – информация о модуле modulename.

2.7.3. Команды для работы с файлами

`ls` – список файлов и каталогов: обычно, чтобы выполнить какие-либо действия с файлами, надо выяснить, есть ли они в каталоге.
`-a` – со скрытыми.
`-t` – сортировка по дате модификации, начиная от старых к новым.
`-s` – сортировать по размеру по убыванию.
`-r` – сортировка в обратном порядке.
`-r` – список с подкаталогами.
`cd` – сменить директорию на домашний каталог.
`cd dir` – сменить директорию на dir.
`cp file1 file2`– скопировать file1 в file2.
`mkdir dir` – создать каталог dir.
`pwd` – показать текущий каталог.
`rm file` – удалить file.
`-r` – удалить каталог dir.
`-f` – удалить форсировано, то есть без запроса.
`-i` – интерактивно подтвердить попытку удаления файла.
`more file` – вывести содержимое file.
`head file` – вывести первые 10 строк file.
`tail file` – вывести последние 10 строк file.

2.7.4. Команды архивации

`tar cf file.tar files`– создать tar-архив с именем file.tar содержащий files.
`c` – создать архив.
`f` – название файла архива.
`t` – вывести содержимое архива.
`r` – дописать в архив.
`x` – распаковать архив.
`v` – выводить отладочную информацию.
`gzip file` – сжать file и переименовать в file.gz.
`-l` – отобразить содержимое архива.

-d – распаковать архив, исходный удалить.

zcat file.gz – просмотр содержимого архива file.gz.

gunzip file.gz – распаковать file.gz в file.

2.7.5. Команды поиска

locate file – найти все файлы с именем file.

find path file – поиск файла file в каталоге path.

-path – поиск по пути.

-name – поиск по имени.

-ipath или -iname – то есть с "i" - регистронезависимый поиск.

-type d или f или l – поиск по типу файла: 'f' для регулярных файлов, 'd' для каталогов и 'l' для символьных ссылок соответственно (другие типы - в man find)

-size n – поиск файла по размеру (n может быть, как n, как +n (более n), как -n (менее n), а также для указания единицы измерения: nc - для байта, nk - для килобайт, nb - блоков).

-empty – поиск пустых файлов.

-print – вывод всего что найдено.

-ls – вывод найденного аналогично команде ls -lids.

-mtime n – поиск файла по времени модификации.

-atime n – поиск файла по времени доступа.

2.7.6. Команды для управления процессами

bg number – список остановленных и фоновых задач; продолжить выполнение остановленной задачи в фоне, имеющей номер number.

fg number – выносит на передний план последние задачи.

n – вынести задачу n на передний план.

kill pid – убить процесс с id pid.

nproc - указывает число дочерних процессов, имеющихся в данном процессе.

ps – вывести ваши текущие активные процессы.

-a – связанные с конкретным терминалом, кроме главных системных процессов сеанса.

-u – отображение пользователя (владельца процесса).

aux – вывод всех процессов в системе.

-u user – отобразить процессы пользователя user.

pstree – команда отображает дерево запущенных процессов.

-a – отображение процессов с аргументами запуска командной строки.

-h – подсвечивает текущий процесс и его предков.

-u – показывает UID процесса. (Когда uid процесса отличается от uid родителя, то новый uid показывается после имени процесса, заключенным в круглые скобки).

pwdx – Сообщает о текущем рабочем директории процесса.

sysctl – Модифицирует параметры ядра в режиме реального времени.

top – показать все запущенные процессы в интерактивном режиме.

h – справка о программе.

k – уничтожить процесс.

n – число отображаемых процессов.

u – сортировать по имени пользователя.

m – сортировать по объему ОЗУ.

p – сортировать по загрузке ЦП.

r – изменить приоритет выполнения.

q – выход.

2.7.7. Команды для работы с ssh

ssh user@host – подключится к host как user.

-p port – подключится на порт port.

2.7.8. Команды для работы с сетью

ping host – проверка сетевого соединения с host и вывод результатов.

traceroute host – трассировка маршрута до определенного хоста.

ifconfig eth0 – отображение/настройка параметров сетевого интерфейса etho (либо другого, без указания имени интерфейса - отобразит конфигурацию всех интерфейсов).

ipcalc – калькулятор сети.

ethtool interface – отобразить физическую статистику интерфейса interface.

route – отображение таблицу маршрутизации.

-n – не преобразовывать имена в адреса.

mtr host – отображение статистики трассировки до хоста host.

netstat – отображение статистики сети.

-r – таблица маршрутизации.

-n – без разрешения IP адресов в имена и сетевых портов в названия.

-a – состояние всех (во всех состояниях) соединений на локальной машине.

-t – статистика по протоколу TCP.

-u – статистика по протоколу UDP.

-i – отобразить статистику сетевых интерфейсов.

-l – просмотр сокетов, слушающих (LISTEN) соединения (ожидающих соединения).
-p – отобразить имя программы и PID (process ID), с которой взаимодействует сокет.
nmap host – просканировать порты на хосте host.
-p ports – просканировать указанные порты.
-o – определить ОС при сканировании порта.
host domain – получить DNS информацию о домене domain.
-a – вывести все записи зоны.
-d – отладка.
-v – вывод подробной информации.
host ns.server – узнать информацию о хосте host с DNS сервера ns.server.

2.7.9. Команды для управления ПО

rpm – (без параметров) вывод версии пакетного менеджера.
-i pkg.rpm – установить пакет (RPM) pkg.rpm.
-v – вывод информации на экран при установке/удалении.
-q – вывод краткой информации о пакете (версия).
-a – вывод информации о всех установленных пакетах.
-f file – Запросить пакет, которому принадлежит файл file.
-i pkg.rpm – вывод полной информации об установленном пакете pkg.rpm.
-l pkg.rpm – Запросить список файлов в пакете pkg.rpm.
-ivh – установка пакета с выводом доп. инфо.
-Uvh – обновление имеющегося пакета с выводом доп. инфо.
-e – удаление пакета.
-qpl pkg.rpm – вывод информации о файле пакета pkg.rpm.
-R - запрос списка пакетов, от которых зависит этот пакет.
--initdb –инициализация базы данных RPM (создает структуру файлов в каталоге /var/lib/rpm, содержащих информацию о пакетах, зависимостях и т.п.).
--nodeps – не обращать внимания на зависимости.
Установка из исходников:

```
./configure
make
make install
```

rpm -Uvh pkg.rpm – установить пакет (RPM).
apt-get – вывод информации о команде управления пакетами apt-get.
install pakage – установка пакета pakage.

`remove pakage` – удаление пакета pakage.
`--purge` – удаление вместе с пакетом конфигурационных файлов.
`check` – проверка дерева зависимостей пакетов.
`update` – обновление локального списка пакетов.
`upgrade` – обновление всех пакетов, не требующих инсталляции (обновляются только те, которые установлены и те, которые зависят от уже установленных)
`dist-upgrade` – обновление пакетов системы с инсталляцией новых пакетов (обновляются все пакеты, в том числе и не инсталлированные).
`apt-cachesearch pakage` – выполняет полный поиск текста pakage по всем доступным файлам пакетов по заданному шаблону. Команда просматривает имена пакетов и описания, для поиска определенной строки, а также выводит имя пакета и его краткое описание.
`ldd /path/program` – вывести список библиотек общего доступа от которых зависит программа /path/program.
`strace program` – трассировка системных вызовов команды program.
`-e write` – указать параметр фильтрации, в данном примере - отслеживать системные вызовы write (часто используется open).
`-f` – отслеживать системные вызовы потомков (желательно использовать).
`-o file` – вывод трассировки в файл file.
10 Команды для управления пользователями.
`su user` – создание оболочки (подоболочки текущей оболочки) с правами пользователя user (без указания пользователя - вызывается оболочка root).
`-, -l, --login` – все три параметра имеют одинаковое значение - загрузить окружение вызываемого пользователя.
`-c command` – выполнить команду command с правами суперпользователя и вернуться к исходным правам после завершения команды.
`last` – показывает, какие пользователи последними входили и выходили из системы.
`lastb` – показывает неудачные попытки входа в систему, которые записаны в файле /var/log/btmp.
`w` – показать пользователей онлайн.
`whoami` – имя, под которым вы работаете.
`finger user@host` – показать информацию о user (без указания пользователя - выведет список пользователей, работающих в системе) на хосте host.
`talk user@host` – чат с пользователем user на компьютере host.
`wall message` – разместить сообщение message на всех терминалах.

3. Аппаратная платформа лабораторного практикума – вычислительные комплексы Эльбрус

3.1. Описание возможностей архитектуры Эльбрус

Введение: архитектура Эльбрус, её особенности

Процессоры с архитектурой “Эльбрус” принадлежат к классу архитектур с явным управлением параллелизмом операций. Основными свойствами процессоров из этого класса являются:

– возможность планирования нескольких операций в одной широкой команде, с целью одновременного исполнения в одном такте;

– точное исполнение широких команд в соответствии со статическим планированием машинного кода.

Средняя степень наполнения широкой команды полезными операциями определяет так называемую логическую скорость работы, которая отражает производительность процессора при исполнении кода. С другой стороны, блокировки исполнения спланированного кода, вызванные различными причинами: ожидание кода, ожидание данных, неверное планирование, неподготовленный переход и т.д., снижают производительность процессора. Можно сказать, что задачей повышения производительности кода на архитектурах VLIW является статическое обнаружение параллелизма на уровне операций, планирование операций с учетом найденного параллелизма, обеспечивающее высокую логическую скорость, и одновременно с тем проведение оптимизаций, снижающих количество блокировок исполнения.

Архитектура Эльбрус располагает различными механизмами, позволяющими решать задачу повышения производительности более эффективно:

– предикатное и спекулятивное исполнение операций: пользуясь этими механизмами, можно планировать в одной широкой команде операции, относящиеся к различным ветвям управления, избавляться от дорогостоящих операций перехода, переносить арифметико-логические операции через операции перехода;

– конвейеризация циклов: позволяет наиболее эффективно исполнять циклы с независимыми (или слабо зависимыми) итерациями;

– динамический разрыв зависимостей: позволяет забрасывать операции чтения (и зацепленные за них цепочки вычислений) за операции записи, потенциально конфликтующие с ними;

– подготавливаемый переход: позволяет избавиться от свойственной многим архитектурам проблемы неправильно предсказанных переходов,

– асинхронный доступ к массивам: позволяет независимо от исполнения команд основного потока буферизовать данные из памяти, запросы к которым формируются в цикле, с адресами, линейно зависящими от номера итерации; тем самым устраняются блокировки ожидания данных в основном потоке.

Особенности архитектуры Эльбрус, касающиеся режима двоичной совместимости с x86 и режима тегированной защиты, остаются за пределами данной главы.

Имеется 5 версий микроархитектуры Эльбрус:

Версия 1 – микропроцессор «Эльбрус» 1891ВМ4Я

Версия 2 – микропроцессор «Эльбрус-S» 1891ВМ5Я и микропроцессор «Эльбрус-2С+» 1891ВМ7Я

Версия 3 – микропроцессор «Эльбрус-4С» 1891ВМ8Я

Версия 4 – микропроцессор «Эльбрус-8С» 1891ВМ10Я и микропроцессор «Эльбрус-1С+» 1891ВМ11

Версия 5 – микропроцессор «Эльбрус-8СВ» 1891ВМ12Я

Широкая команда

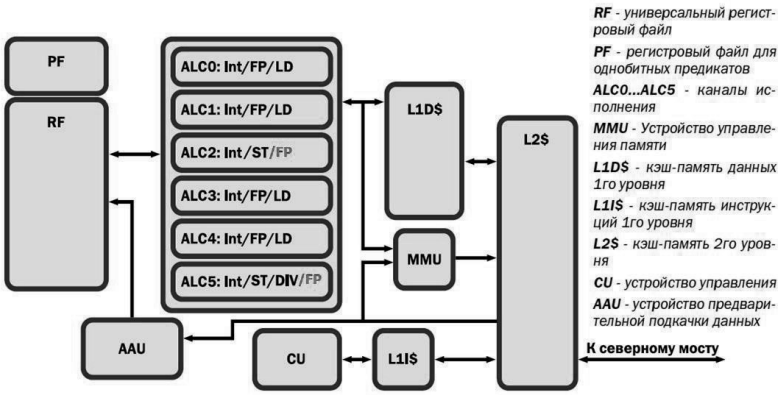
Под широкой командой Эльбрус понимается набор элементарных операций, которые могут быть исполнены одним ядром процессора в одном такте.

Под кодировкой широкой команды Эльбрус понимается множество слогов, описывающих элементарные операции, либо содержащих дополнительную информацию.

Парк устройств

Арифметико-логические устройства (АЛУ)

В процессорах Эльбрус поддержано шесть АЛУ с номерами 0-5 (рисунок 3.1).



Красным указаны изменения для архитектуры в.4/в.5 (Эльбрус-8С/8СВ)

Рисунок 3.1. Схема АЛУ и элементов, взаимодействующих с ними

В парках АЛУ0 и АЛУ3 присутствуют:

- целочисленное арифметическое/побитовое/сдвиговое устройство;
- устройство сравнения;
- устройство для операций ММХ 8бит, 16 бит, 32бита;
- двухэтажное вещественное арифметическое устройство;
- двухэтажное вещественное арифметическое устройство над упакованными 32-разрядными вещественными числами;
- устройство обращения к памяти по чтению.

В парке АЛУ0 также присутствует устройство обращения к специальным регистрам.

В парках АЛУ1 и АЛУ4 присутствуют:

- двухэтажное целочисленное арифметическое/побитовое/сдвиговое устройство;
- устройство сравнения;
- устройство для операций ММХ 8бит, 16 бит, 32бита;
- двухэтажное вещественное арифметическое устройство;

- двухэтажное вещественное арифметическое устройство над упакованными 32-разрядными.

вещественными числами

В парках АЛУ2 и АЛУ5 присутствуют:

- целочисленное арифметическое/побитовое/сдвиговое устройство;
- устройство обращения к памяти по чтению/записи;
- двухэтажное вещественное арифметическое устройство (начиная с 4й версии микроархитектуры);
- двухэтажное вещественное арифметическое устройство над упакованными 32-разрядными вещественными числами (начиная с 4й версии микроархитектуры).

Также в парке АЛУ5 присутствуют устройство деления и устройство извлечения квадратного корня.

Предикатное устройство

В широкой команде можно выполнить до трех логических операций над предикатными регистрами, причем длительность операций составляет ½ такта, и поэтому в одном такте можно планировать логические операции, одна(одни) из которых использует(ют) результат другой(других).

Устройство асинхронной подкачки массивов (АРВ)

В одной широкой команде можно исполнить до 4 операций чтения из буфера АРВ.

Устройство управления

В одной команде можно исполнить не более одной операции передачи управления, и не более одной операции подготовки перехода.

. Кодировка

Широкая команда может кодироваться четным числом слогов (не более 16), каждый размером 4 байта. Во всякой широкой команде первым идет слог заголовка HS, описывающий состав и длину широкой команды. Доступны следующие виды слогов (Рисунок 3.2):

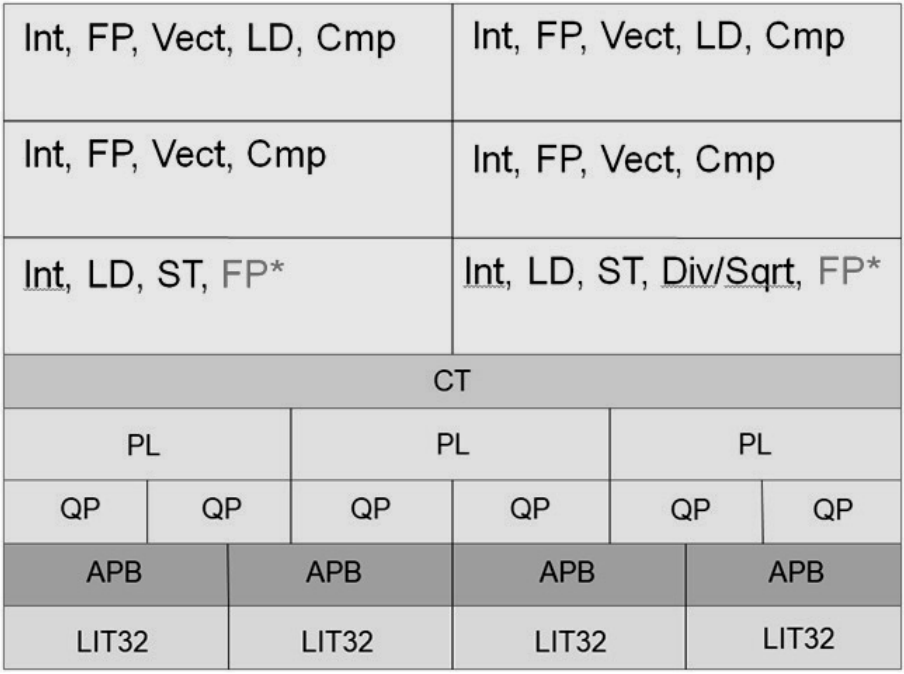


Рисунок 3.2. Виды полей

• Арифметико-логические (ALS), не более 6, номера ALS0-ALS5

В них кодируются операции АЛУ:

- целочисленные (арифметика, побитовая арифметика, сдвиги)
- “двухэтажные” целочисленные (только ALS1, ALS4)
- записи в память (только ALS2, ALS5)
- чтения из памяти (только ALS0, ALS2, ALS3, ALS5)
- вещественной арифметики (только ALS0, ALS1, ALS3, ALS4)
- “двухэтажных” операций вещественной арифметики (только ALS0, ALS1, ALS3, ALS4)
- целочисленного и вещественного деления (только ALS5, не более 1 в два такта)
- целочисленного и вещественного сравнения (только ALS0, ALS1, ALS3, ALS4)
- чтения/записи специальных регистров (только ALS0)

Некоторые операции ограничивает размещение других операций в той же широкой команде. Наиболее существенным из таких ограничений касается

одновременного размещения операций чтения и записи в память; допустимо три варианта:

- две записи,
- одна запись и не более двух чтений,
- не более четырех чтений.

• Литеральные слоги (LTS0-LTS3)

Литералы предназначены для хранения констант, общим объемом 16 байт

• Предикатные слоги

Слоги PLS0-PLS1 предназначены для кодирования логических операций, и для постановки операций в ALS под условие.

• Слоги постановки под предикат

Слоги CDS0-CDS1 служат для связывания слогов ALS и предикатов, управляющих возможной отменой операций в слогах ALS.

• Слог управления CT0, CT1

Слоги CS0-CS1 предназначены для кодирования операций подготовки перехода, операций немедленного перехода, дополнительной информации для операций записи в специальные регистры

• Вспомогательный слог SS

Предназначен для осуществления подготовленного перехода, для кодирования операций продвижения базы вращаемых регистров, поддержки аппаратного циклового счетчика, некоторых других операций.

• Полуслоги доступа к асинхронным массивам AAS0–AAS5

Предназначены для считывания данных из буфера предварительной подкачки и для управления циклом.

• Полуслоги расширения ALES0-ALES3

Предназначены для кодирования дополнительной информации для операций ALU.

Приведем пример дизассемблера однотактного цикла из теста на пиковую производительность:

```
слог    ассемблерные команды слога
HS      M_25e0:loop_mode
SS      ct %ctpr1 ? %pred5 && #NOT_LOOP_END
        ipd 2
        abn abnf=1, abnt=1
        abp abpf=1, abpt=1
        alc alcf=1, alct=1
ALS0    cmpedb,0,sm %db[36], %db[19], %pred0
ALS1    xor_ord,1,sm %db[37], %db[24], %db[37], %db[28]
```

```
ALS2    staad,2 %db[27], %aad0[ %aasti5 ]
        incr,2 %aaincr0
ALS3    cmpedb,3,sm %db[36], %db[40], %pred1
ALS4    sub_andd,4,sm %db[14], %db[28], %db[15], %db[36]
ALS5    subd,5,sm %db[14], %db[15], %db[15]
ALES
AAS
AAS     movad,0 area = 1, ind = 0, am = 1, be = 0, %db[14]
        movad,1 area = 0, ind = 0, am = 1, be = 0, %db[0]
AAS     movad,2 area = 1, ind = 0, am = 1, be = 0, %db[27]
        movad,3 area = 0, ind = 0, am = 1, be = 0, %db[1]
LTS0
PLS2    landp ~@p5, ~@p4, @p6
        pass @p6, %pred2
PLS1    landp @p0, ~@p1, @p5
PLS0    pass %pred1, @p0
        pass %pred2, @p1
        landp ~@p0, @p1, @p4
```

Видно, что все 16 возможных слогов заняты.

Регистры

Основной (рабочий) размер регистра – 64 бита (1 двойное слово, 1dw)

В архитектуре версии 5 размер регистров увеличен до 128-разрядов.

Логическая организация регистров

В рамках одной процедуры (процедурного контекста) доступны следующие регистры:

– глобальные, 32dw, %g0-%g31; регистры доступны всем процедурам одновременно, и могут использоваться по договоренности либо для хранения небольших глобальных данных, либо как локальные регистры, не сохраняющие значений при процедурных вызовах;

– локальные, не более 64dw (количество указывается специальной операцией в первом такте процедуры), %r0-%r63; согласно программным соглашениям, первые 8 локальных регистров могут быть использованы как входные параметры текущей процедуры; локальные регистры сохраняют значения при процедурных вызовах;

– вращаемые, не более 128dw (количество задается специальной операцией, может изменяться внутри процедуры); вращаемые регистры используются для конвейеризации циклов (аппаратное переименование по принципу вращения), первые восемь могут использоваться как параметры вызываемых процедур; вращаемые регистры не сохраняют значений при процедурных вызовах;

Физическая организация регистров

В процессоре Эльбрус реализован регистровый файл из 256 регистров. Регистры разделяются на:

– 32 глобальных (не изменяющих значение при процедурном вызове)

– 224 стековых, которые аппаратно переназначаются при процедурных вызовах, при исчерпании стека снизу аппаратно сохраняются в памяти, при исчерпании стека сверху аппаратно восстанавливаются из памяти. Непрерывный отрезок стековых регистров используется как локальные и вращаемые регистры.

Предикатные регистры

Результаты операций сравнения заносятся в специальные регистры, принимающими только два значения истина/ложь, (и поэтому называющиеся предикатными). Процедуре доступно 32 предикатных регистра, их значения сохраняются при процедурных вызовах.

Физически предикатные регистры хранятся в одном 64-битном регистре, программно недоступном.

Дополнительное назначение предикатных регистров будет разъяснено в разделе 5, пункте “Слияние кода, предикатные операции”

Регистры передачи управления

Для предварительной подготовки перехода в микропроцессоре Эльбрус реализовано 3 регистра: %ctpr0–%ctpr2. Эти регистры могут использоваться только в операциях перехода, подготовки перехода.

Регистры специального назначения

– LSR

Регистр, управляющий исполнением циклов. Содержит аппаратный счетчик, биты значимости конвейеризации, пролога/эпилога, битами контроля за операциями чтения и операциями с побочным эффектом в прологе/эпилоге, дистанцией предварительной подкачки массивов.

– FPSR, PFPSR

Эти регистры отвечают за режимы округления для некоторых операций преобразования чисел из вещественных форматов в целые

– AAD(0-31), AAIND(0-15, AAIND0=0), AAINCR(0-7, AAINCR0=1)

Регистры, управляющие работой асинхронной программы, задают начальные адреса массивов и инкременты адресов через итерации.

Управление

Переход

Передача управления внутри процедуры может производится двумя способами:

– неподготовленный переход (по константному смещению)

Переход ibranch (“немедленный переход”) не потребляет регистра ctpr, но плох с точки зрения производительности тем, что при исполнении переходе возникает безусловная блокировка длиной минимум 6 тактов. Его целесообразно использовать при априорно низкой (например, <1%) вероятности перехода.

– подготовленный переход (по регистру управления)

Переход ct (“подготовленный переход”) возможен по одному из трех доступных регистров передачи %ctpr1-%ctpr3. Для того, чтобы переход произошел без блокировок, соответствующий регистр передачи управления должен быть подготовлен соответствующей операцией (disp, getpl); при этом подготовка перехода должна производиться за 5 тактов до перехода в случае операции disp (константного смещения), и за 9 тактов в случае операции getpl (неконстантного смещения, как, например, в конструкциях switch и goto по переменной метке).

Вызов

Вызов процедуры может производиться по одному из станков перехода операцией call, при этом задержки для disp и getpl такие же, как в случае ct. Операция call не может быть произведена «немедленно» по аналогии с ibranch, и всегда потребляет регистр передачи управления.

Возврат

Возврат из процедуры может производиться операцией ct только по регистру передачи управления ctpr3, который для исполнения без блокировок должен быть подготовлен за 6 тактов операцией return.

Специальные операции

– ldisp

Эта операция присваивает станку ctpr2 местоположение асинхронной программы доступа к массивам; подробнее механизм асинхронного доступа к массивам изложен в разделе 1.7

– sdisp

Эта операция подготавливает произвольный станок для системного вызова

Предикатный режим исполнения операций

При помощи предикатных регистров можно управлять отменой отдельных операций в широкой команде. Любая операция в АЛУ может быть снабжена предикатным регистром (возможно, с битом инверсии), и при исполнении широкой команды операция не будет выполнена, если значение предиката равно 0 (равно 1, если проставлен бит инверсии).

Приведем пример, иллюстрирующий применение предикатного исполнения операций:

```
int a,b;
…
if (a==0)
  {
  b++;
  }
…
```

Такой код может быть выполнен при помощи следующих операций:

```
cmpe Ra,0 => p0 ! сравнение a на равенство 0
ibranch label ~p0 ! условный переход на обходную метку
add Rb,1 => Rb ! инкремент b
label:
```

где Ra,Rg – регистры, хранящие значения переменных a и g. Если же мы воспользуемся предикатным режимом исполнения команд, код может выглядеть так:

```
cmpe Ra,0 => p0 ! сравнение a на равенство 0
add Rb,1 => Rb | p0 ! инкремент b при условии p0
```

Таким образом, наличие предикатного режима позволяет в ряде случаев избавляться в целевом коде от дорогостоящих операций условного перехода.

Спекулятивный режим исполнения операций

Для повышения параллелизма на уровне операций часто бывает полезно начать выполнять операцию до того, как станет известно, нужно ли выполнять эту операцию. В случае, если операция может привести к прерыванию (например, чтение из памяти или деление), может произойти некорректный аварийный выход либо нарушение логики исполнения программы. Для того, чтобы решить эту проблему, в архитектуре Эльбрус допускается выполнение операций в режиме отложенного прерывания, также называемом спекулятивным режимом. Для того, чтобы обеспечить возможность хранения информации об отложенном прерывании, ячейки памяти, числовые и предикатные регистры снабжены диагностическим битом, по одному биту на одно слово (1слово=4байта=32 бита) в памяти, по два на 64-битный числовой регистр (по одному на старшую и младшую части), и по одному на каждый предикатный регистр. Примерная схема работы такова:

вместо выработки прерывания операция, имеющая спекулятивный признак (везде далее «спекулятивная операция») записывает 1 в диагностический бит результата,

диагностический операнд у спекулятивной операции приводит к диагностическому результату,

диагностический операнд у неспекулятивной операции приводит к выработке отложенного прерывания.

Возможности оптимизации, предоставляемые спекулятивным режимом исполнения, таковы: в спекулятивном режиме операция может быть исполнена раньше, чем условный переход на линейный участок, содержащий эту операцию, и логика исполнения программы при этом сохраняется.

Пример:

```
int* a;
…
if (a!=NULL)
  {
  *a++;
  }
…
```

Такой код может быть выполнен при помощи следующих операций:

```
0 disp label => ctpr1 ! подготовка обходной метки
0 cmpe Ra,0 => p0 ! сравнение указателя на равенство 0
4 ct ctpr1 ? p0 ! условный переход на обходную метку
5 ld Ra,0 => Rx ! чтение из указателя a
8 add Rx,1 => Ry ! инкремент результата чтения
9 st Ra,0 <= Ry ! запись в a результата инкремента
label:
```

время работы – 10 тактов

Спекулятивный режим позволяет спланировать код с большей параллельностью:

```
0 disp label => ctpr1 ! подготовка обходной метки
0 cmpe Ra,0 => p0 ! сравнение на равенство
0 ld.s Ra,0 => Rx ! чтение из указателя a
3 add.s Rx,1 => Ry ! инкремент результата чтения
4 ct ctpr1 ? p0 ! условный переход на обходную метку
5 st Ra,0 <= Ry ! запись в a результата инкремента
label:
```

время работы – 6 тактов

Обратим внимание, что операции ld и add, поставленные в спекулятивный режим, будут исполнены независимо от результата сравнения, причем при a==NULL результат чтения будет диагностическим (чтение по адресу NULL вызывает прерывание, которое будет отложено). При этом результат операции add, содержащий диагностическое значение, не будет использован, т.к. операция st исполнится только при a!=NULL.

Скомбинировав предикатный и спекулятивный режимы, можно получить еще более быстрый код:

```
0 ld.s Ra,0 => Rx ! чтение из указателя a
3 add.s Rx,1 => Ry ! инкремент результата чтения
0 cmpe Ra,0 => p0 ! сравнение на равенство
4 st Ra,0 <= Ry ? ~p0 ! запись под предикатом не-p0
```

время работы – 5 тактов

Отметим, что приведенный код является более параллельным, чем в чистом предикатном режиме, поскольку цепочка операций ld.s и add.s не зацеплена за операцию сравнения, вырабатывающую предикат.

Асинхронный доступ к массивам

Для увеличения производительности в случае регулярного доступа к элементам массивов в E2K реализован уникальный механизм асинхронного доступа к элементам массива. Суть его состоит в следующем: доступ к массивам кодируется особым образом в виде кода асинхронной программы (она состоит только из операций “fapb“). Операции fapb запускаются по циклу, пополняя буфера упреждающих данных для разных массивов. При этом основной поток исполнения забирает данные из этого буфера операциями mova (вместо запуска операций чтения).

Преимущества, предоставляемые механизмом асинхронного доступа к массивам:

– вместо операции чтения, занимающей ALU, используется операция mova, занимающая отдельный канал, что освобождает в широкой команде место под арифметическую операцию;

– блокировки из-за отсутствия данных существенно уменьшаются, т.к. операции доступа к памяти, имеющие непредсказуемую длительность, выполняются асинхронно, и не блокируют основной поток исполнения операций.

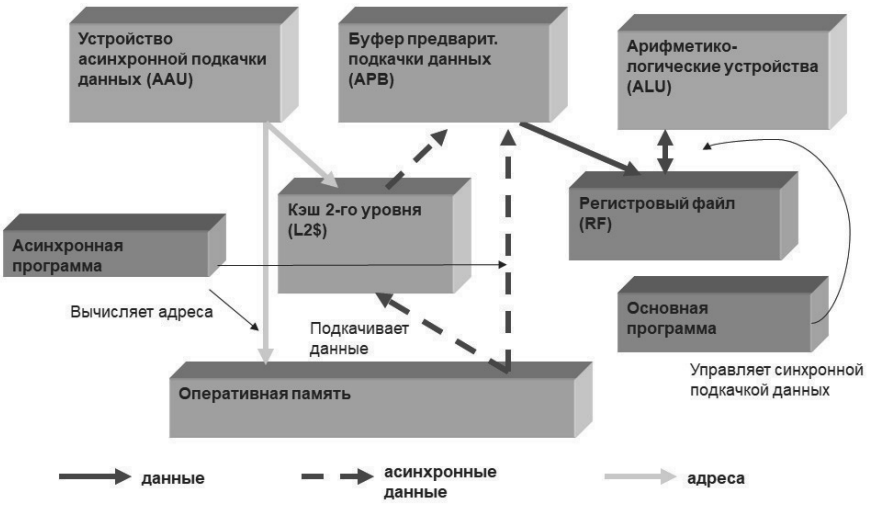


Рисунок 3.3. Возможные виды передачи данных

Динамический разрыв зависимостей

Выявление параллелизма операций может потребовать выполнения какой-либо операции чтения раньше, чем опережающей ее по исходному тексту операции записи. Если при этом адрес одной этих операций неизвестен, подобное действие является некорректным; в таких случаях принято говорить о наличии конфликта по памяти между соответствующими операциями.

Механизм динамического разрыва конфликтов (везде далее DAM, Disambiguating Access to Memory) позволяет поднимать операции чтения (и зацепленные за них цепочки операций) выше операций записи, потенциально конфликтующих с ними. Для этого аппаратно реализована таблица адресов для поднятых чтений (MLT, Memory Lock Table), из которой при выполнении любой операции записи автоматически вычеркиваются адреса, пересекающиеся с адресом этой записи. Использование механизма DAM на практике выглядит следующим образом:

– вместо операции чтения ld строятся две операции – ld.lock (запирающее чтение) и ld.check (проверочное чтение)

– операция ld.lock планируется выше операции записи, равно как и операции, зависящие от нее

– строка с адресом операции ld.lock заносится в таблицу адресов MLT

– операция записи st обновляет таблицу MLT, вычеркивая из нее те строки, в которых хранящийся адрес пересекается с адресом записи

– сразу после записи выполняется операция ld.check, проверяющая наличие строки в MLT, содержащей адрес чтения

– если искомая строка в MLT обнаружена, продолжаем исполнение, поскольку результат операции чтения ld.lock правильный

– если искомая строка в MLT не обнаружена, необходимо заново перезапустить все операции, поднятые выше записи. Для этого управление передается на компенсирующий код, где спланированы указанные операции и возврат к команде, следующей за ld.check

Описанная схема может быть проиллюстрирована на следующем примере:

```
int* a;
int* b;
int x,y;
…
*b=15;
x=(*a+1)*y;
…
```

Обычный код выглядит так:

```
st Rb,0 <= 15 ! запись по указателю b
ld Ra,0 => Rt1 ! чтение из указателя a
add Rt1,1 => Rt2 ! прибавляем 1
mul Rt2,Ry => Rx ! умножаем
```

Код после применения DAM может выглядеть так:

```
ld.lock Ra,0 => Rt1 ! чтение из указателя a т занесение адреса a в таблицу DAM
add Rt1,1 => Rt2 ! прибавляем 1
mul Rt2,Ry => Rx ! умножаем
st Rb,0 <= 15 ! запись по указателю b
ld.check Ra,0 => Rt1 ! проверка целостности адреса a в таблице DAM, повторное чтение при неудачном результате
rbranch comp_code_label ! переход при неудачном результате проверки по метке компенсирующего кода
back_label: ! метка возврата

…
comp_code_label: ! здесь нужно заново выполнить операции, поднятые выше записи
add Rt1,1 => Rt2 ! прибавляем 1
mul Rt2,Ry => Rx ! умножаем
ct back_label ! возвращаемся обратно
…
```

Преимущество в производительности состоит в следующем: компенсирующий код будет исполняться только в тех, как правило, редких случаях, когда адреса a и b совпали либо пересеклись, и перенос операции чтения вверх был неправомочен. Во всех остальных случаях ветка вычислений, зацепленная за чтение, выполняется до операции записи.

Как видно из приведенного примера, степень параллельности на уровне операций при помощи DAM может быть увеличена весьма существенно, правда, ценой добавления новых операций: всякая операция, поднятая выше записи вслед за операцией чтения при помощи механизма DAM, требует построения компенсирующего кода.

Заметим, что операции с побочным эффектом (записи в память, переходы, записи в специальные регистры), присутствующие в зацепленной за чтение цепочке, не могут быть перенесены выше операции записи, т.к. их неверное исполнение, вообще говоря, может приводить к нарушениям логики программы, которые не могут быть исправлены компенсирующим кодом.

Подсистема памяти

Необходимо сразу же отметить тот хорошо известный факт, что правильный учет особенностей подсистемы памяти способен самым кардинальным образом положительно влиять на производительность

Схема работы системы памяти изображена на рисунке 3.4.

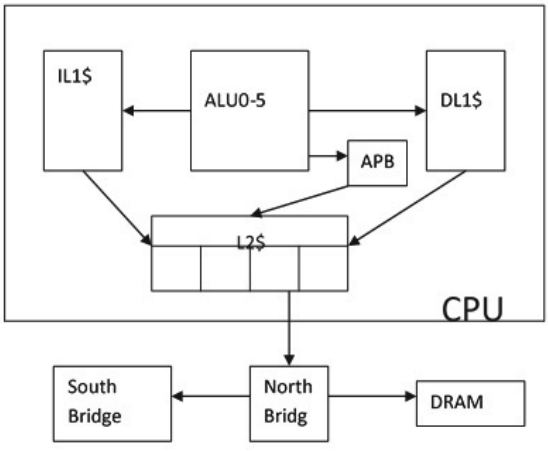


Рисунок 3.4. Схема работы системы памяти

В состав микропроцессора 1891ВМ4Я «Эльбрус» входят:

– кэш кода первого уровня (IL1$), 64K, 4-ассоциативный

– кэш данных первого уровня кода (DL1$), 64K, 4-ассоциативный

– общий (D+I) кэш второго уровня (L2$), 256K, 4-ассоциативный, 4-банковый

– устройство обращения к памяти MAU

В состав микропроцессора 1891ВМ5Я «Эльбрус-S» входят:

– кэш кода первого уровня (IL1$), 64K, 4-ассоциативный

– кэш данных первого уровня кода (DL1$), 64K, 4-ассоциативный

– общий (D+I) кэш второго уровня (L2$), 2048K, 4-ассоциативный, 4-банковый

– устройство обращения к памяти MAU

– контроллер памяти (North Bridge)

Величины длительностей операций чтения при попаданиях в кэш-память различного уровня

L1$ попадание:
3 такта
L1$ промах, L2$ попадание:
>=9 тактов
L2$ промах:
1891ВМ4Я «Эльбрус»: >~75 тактов
1891ВМ5Я «Эльбрус-S»: >~60 тактов

Темп обработки заявок на чтение/запись:
L1$
4 dw в такт по чтению либо записи

L2$
1 dw в такт по чтению либо записи в одном банке
4 dw в такт по чтению либо записи в сумме

Устройство MAU
1891ВМ5Я «Эльбрус»
4/3 dw в такт по чтению
2/3 dw в такт по записи
1891ВМ5Я «Эльбрус-S»
2 dw в такт по чтению либо записи

3.2. Использование возможностей архитектуры при статической оптимизирующей компиляции

Задача построения быстро работающего кода в архитектурах VLIW традиционно возлагается на оптимизирующий компилятор (сокращенно ОК). Оптимизации, которые проводит компилятор, можно условно разделить на две категории:

полезные для произвольной архитектуры, например, удаление мертвого кода, упрощение вычислений, сбор общих подвыражений, и т.д.

нацеленные на особенности конкретной архитектуры.

В данном разделе речь пойдет в первую очередь об оптимизациях либо архитектурно-направленных, либо имеющих качественно большую важность для архитектуры Эльбрус, чем для других архитектур.

Статический поиск и разрыв зависимостей

Определение: операции i1 и i2 зависимы, если операция i2 должна быть выполнена не ранее, чем после некоторого числа тактов после операции i1. Число тактов называется расстоянием между зависимыми операциями.

Зависимость может быть нескольких видов:

– по потоку данных:

– прямые (flow, RAW=ReadAfterWrite), если результат i1 является аргументом i2; расстояние в таких случаях часто зависит только от операции i1, и тогда его принято называть длительностью i1 (latency);

– по выходу (output, WAW=WriteAfterWrite), если i1 и i2 имеют одинаковый результат; расстояние как правило равно 0-1;

– анти (anti, WAR=WriteAfterRead), если результат i2 является аргументом i1; расстояние, как правило, равно 0-1.

– по управлению (control), если операции не пересекаются по своим аргументам, но тем не менее зависимы, например: i1 – запись в память, i2 – чтение из памяти по возможно совпадающему адресу.

Наличие зависимостей между операциями уменьшает параллелизм на уровне операций. Поэтому разрыв необязательных зависимостей является очень важной задачей получения эффективно работающего кода.

Как правило, поиск необязательных зависимостей производится среди зависимостей по управлению: пары запись/чтение, вызов/чтение. В ОК Эльбрус реализованы внутрипроцедурный и межпроцедурный анализы указателей для выяснения возможности/невозможности пересечения адресов операций обращения к памяти. Для разрыва неопределенных зависимостей ОК использует механизм DAM (см пункт 1.8), разрыв зависимостей между линейными обращениями в циклах (оптимизация RTMD, Run Time Memory Disambiguation, разрыв зависимостей во время исполнения).

Слияние условного кода

Участок программы, содержащий большое количество условных операторов, преобразуется к коду, содержащему большое число меток и операций условного перехода, перемежающихся с другими операциями. Такая структура кода является плохой для выявления параллелизма на уровне операций.

Опираясь на предикатный и спекулятивный режимы, поддержанные архитектурой E2K, можно «слить» несколько независимых участков кода в один.

В ОК имеется оптимизация «слияние условного кода» (if_conversion), которая анализирует пользу от совместного планирования линейных участков, связанных операцией условного перехода, по сравнению с независимым планированием этих участков. В случае обнаружения положительного эффекта, оптимизация производит слияние, с устранением операции перехода, и постановкой операций каждого участка под предикат условия перехода, под которым он выполняется. В результате параллельность, обнаруживаемая в коде, существенно возрастает.

Конвейеризация циклов

Цикл, различные итерации которого независимы, может быть исполнен на процессоре архитектуры E2K существенно эффективнее циклов с зависимыми итерациями. Для таких циклов придуман механизм конвейеризации.

Суть конвейеризации состоит в следующем:

число тактов исполнения конвейеризованного цикла (обозначается II=Iteration Interval) определяется исходя из количества операций цикла и возможности распределения их по устройствам широкой команды

итерация цикла разбивается на т.н. «стадии», длина стадии не более ii

всякая операция Odef, результат которых используется на другой стадии, заносит результат в базированный регистр Bk, и операции Ouse, использующие этот результат, получают его в базированном регистре, сдвинутом относительно Bk на число регистров, равно числу стадий, на которое Ouse отстоят от Odef.

Конвейеризацию необходимо проиллюстрировать на примере:

```
for (i=0; i<N; i++)
{
  a[i]=b[i]*14+5;
}
```

Неконвейеризованный цикл может выглядеть так:

```
disp loop_head => ctpr1 ! подготовка метки
loop_head: ! метка головы цикла
ld Rb,Ri4 => Rt1 ! чтение b
mul Rt1,14 => Rt2 ! умножение
add Rt2,5 => Rt3 ! сложение
```

```
st Ra,Ri4 =< Rt3 ! запись в a
add Ri4,4 => Ri4 ! инкремент переменной-адреса
cmpe Ri4,4*N => p0 ! условие выхода из цикла
ct ctpr1 ~p0
```

NB: Здесь Ri4 – учетверенное значение i, поскольку для вычисления адреса требуется умножать i на размер данных.

Заметим, что без наложения итераций тело цикла не может быть спланировано лучше, чем за 3(длительность ld)+6(длительность mul)+1(длительность add)=10 тактов.

Заметим, что арифметико-логических операций в теле цикла всего 6, поэтому ii можно положить равным 1. При этом конвейеризованный цикл с ii=1 может выглядеть так:

```
disp loop_head => ctpr1 ! подготовка метки
loop_head: ! метка головы цикла
ld Rb,Br24 => Br0 ! чтение b
mul Br6,14 => Br6 ! умножение
add Br18,5 => Br18 ! сложение
st Ra,Br22 =< Br20 ! запись в a
add Br24,4 => Br22 ! инкремент переменной-адреса
cmpl Ri4,4*N => bp0 ! условие выхода из цикла
abn ! прокрутка вращаемых числовых регистров
abp ! прокрутка вращаемых предикатных регистров
ct ctpr1 bp2
```

Здесь отрезки базированных регистров хранят значения регистров неконвейеризованной версии цикла:

```
Br0-Br6 : Rt1
Br6-Br18 : Rt2
Br18-Br20 : Rt3
Br22-Br44 : Ri4
bp0-bp2 : p0
```

Работу конвейеризованного цикла можно показать графически (Рисунок 3.5).

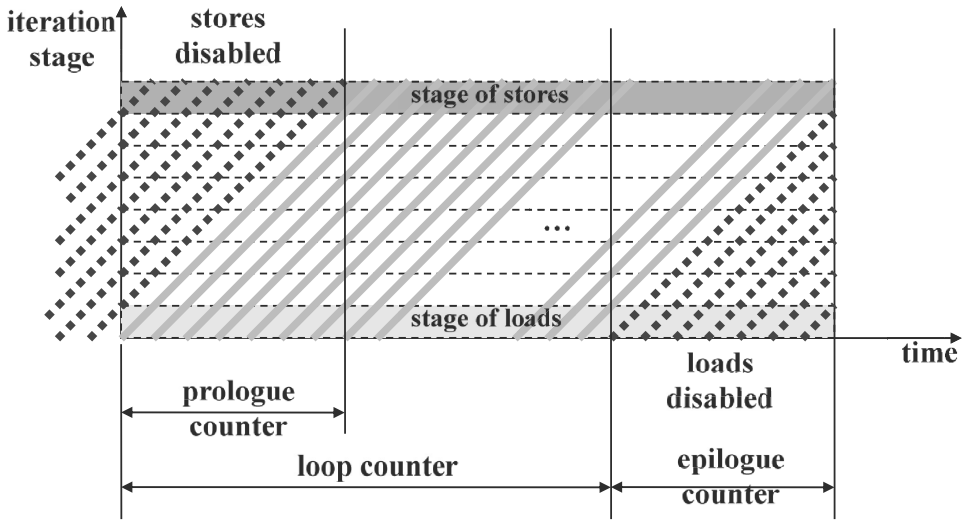


Рисунок 3.5. Работа конвейеризованного цикла

Параллельное планирование широких команд

Основываясь на операциях управления компилируемой задачи, можно расставить в коде все необходимые метки для переходов. Расставленные метки определяют т.н. расширенные линейные участки кода (краткое название - гиперузлы): попасть внутрь гиперузла можно только через начальную метку, и операции перехода внутри гиперузла могут быть направлены только наружу либо на начальную метку. Условных выходов из гиперузла может быть несколько. Среднее время исполнения гиперузла можно определить как сумма по всем выходам из гиперузла времен работы до выхода, умноженных на вероятность этого выхода.

Оптимизирующий компилятор решает задачу планирования широких команд в пределах гиперузла, с целью минимизации среднего времени исполнения гиперузла. Основываясь на графе зависимостей между операциями, с учетом расстояний между зависимыми операциями, можно решать задачу оптимального размещения элементарных операций на устройствах широких команд, добиваясь минимального расстояния в тактах до выходов из гиперузла. Точное решение задачи требует экспоненциальной сложности алгоритма планирования, однако, приближенные методы дают сравнительно хорошие результаты в практическом большинстве случаев.

Заметим, что планирование операции выхода теоретически не может быть лучше, чем длительность критического пути (цепочки зависимых операций максимальной суммарной длительности, завершающейся операцией перехода),

Работа параллельного планирования на практике не требует (и не позволяет) ручной настройки. Эффективно помочь планированию можно за счет уменьшения критических путей, в первую очередь, за счет уменьшения рекуррентных цепочек в циклах, и за счет устранения ненужных зависимостей.

3.3. Возможности настройки оптимизирующего компилятора LCC

Режимы компиляции

Оптимизации.

Оптимизирующий компилятор выполняет анализ исходной программы и ряд оптимизирующих преобразований, номенклатура которых определяется уровнем оптимизации, который выбирается пользователем.

O0 – режим компиляции без оптимизаций;

O1 - локальные внутрипроцедурные, простейшие цикловые оптимизации и необходимый для них анализ;

O2 - включает в себя все виды анализа и преобразований уровня O1, а также всевозможные цикловые оптимизации, часть межпроцедурных оптимизаций;

O3 - включает в себя все виды анализа и преобразований уровня O1 и O2, а также дополнительные глобальные и межпроцедурные оптимизации;

O4 - включает в себя все виды анализа и оптимизаций, за счёт значительного увеличения времени компиляции.

Пример:

lcc -O3 t.c

Сбор и использование компиляторного профиля.

Под компиляторным профилем будем называть информацию о количестве исполнений различных узлов (или гиперузлов, см 2.4) программы. Компиляторный профиль может быть динамическим, полученным в результате исполнения инструментированного кода программы на некотором (желательно представительном) наборе входных данных. Компиляторный профиль может быть статическим, если компилятор предсказывает вероятность всех ветвлений

и распределяет счетчики количества выполнений команд в соответствии с вероятностями.

В случае, если динамического профиля нет, компилятор использует статический профиль.

Для того, чтобы получить реальную профильную информацию, необходимо выполнить два действия:

1) Откомпилировать исходную программу с опцией -fprofile-generate[=DIR] (DIR - каталог для сброса файлов с профильной информацией, по умолчанию DIR=.). При этом исходный текст программы инструментируется счетчиками переходов и вызовов.

2) Исполнить инструментированную программу, используя представительные входные данные; если задача интерактивна, следует использовать представительный усредненный сценарий работы. После выхода из задачи в каталог DIR будет сброшен файл eprof.sum0 (после следующего запуска eprof.sum1 и т.д.)

Использование профильной информации:

Загрузка профиля компилятором осуществляется с помощью опции -fprofile-use[=DIR/FILE] , где DIR/FILE - имя файла загружаемой профильной информации. Если DIR/FILE не задано, то используется eprof.sum из текущей директории.

Если файлов несколько, профильная информация из них они будет просуммирована.

Пример:

Очистка/создание каталога DIR и создание инструментированного кода

1 Компиляция для сбора профиля:

lcc -fprofile-generate=DIR t.c

2 Запуск a.out. Получение файлов профиля для двух входных данных.

./a.out in1

./a.out in2

В каталоге DIR создаются файлы профильной информации eprof.sum0 и eprof.sum1.

3 Суммирование и использование этих файлов профиля.

lcc -fprofile-use=DIR t.c

В каталоге DIR создается файл просуммированного профиля eprof.sum (суммируются eprof.out0 и eprof.out1). и он используется при компиляции программы.

Если файл профиля уже существует его можно сразу использовать:

lcc -fprofile-use=DIR/FILE t.c

Дополнительные опции.

Здесь описаны некоторые наибоее часто используемые опции. Остальные можно получить по опции –help.

-mptr32 Размер всех указателей 32 бита, размер программы вместе с данными не превышает 2^{32} байтов

-mptr64 Размер всех указателей 64 бита (реальный размер виртуальной памяти 2^{48} байтов)

-faligned декларирует выровненные обращения в память

-fstrict-aliasing декларирует обращение к каждому участку памяти по указателям совместимых типов

-ffinite-math декларирует отсутствие знаковых нулей, nan, inf, точного errno

-ffast - Разрешение преобразований с вещественной арифметикой; выравнивание данных; некоторые дополнительные рискованные оптимизации (нарушающие понятия знаковый ноль, бесконечность и т.п.). Как пример, x*0 будет заменено на 0, несмотря на то, что это может быть, как +0, так и -0. Включает в себя опции –faligned, -ffinit-math, -fstrict-aliasing.

-fwhole - Указание сборки в режиме "вся программа". По этой опции компилятор считает, что для всех объектов, определенных в поданных на трансляцию модулях, все использования по имени локализованы в этих модулях (исключение - main и процедуры динамической инициализации).

3.4. Особенности программ с точки зрения производительности кода на процессорах Эльбрус

Учет особенностей архитектуры Эльбрус при проектировании программ

Прежде всего, необходимо хорошо представлять себе, какие участки программы будут работать больше других. Как правило, для программ применимо правило 80/20 – 80% времени работы происходит в 20% кода. Далее будем называть «горячими» те участки кода, на которые приходится большая часть времени работы. Понятно, что производительность программы в первую очередь зависит от оптимальности ее горячих участков.

Виды структур данных

Работа с различными структурами данных может существенно отличаться по средней скорости доступа, темпу чтения и изменения. При выборе той или иной структуры данных может помочь нижеследующая информация о характеристиках таких структур:

– простые одномерные массивы

при регулярном считывании/записи элементов массива могут достигаться теоретически максимальные показатели темпа доступа к памяти – 32 байта в такт при попадании в L2$, 32 байта в 3 такта при гарантированном отсутствии в L2$ (см. 1.9), при условии обращения к соседним элементам, причем для считывания может применяться механизм APB; при регулярном обращении с большим шагом (>64b) APB все еще применим, но темп существенно падает (до 64 раз при побайтовой обработке);

– одномерные массивы структур

при регулярной обработке применим APB, однако, следует следить за тем, чтобы набор одновременно читаемых/записываемых полей в горячих участках был как можно более компактным; весьма полезен (в ущерб наглядности) переход от массивов структур к набору массивов, хранящих отдельные поля;

– многомерные выстраиваемые массивы

многомерные массивы в языке FORTRAN (а также многомерные массивы в языке C при условии константных длин старших размерностей) являются одномерными по сути, но индексируемые несколькими размерностями:

A(i,j,k) “FORTRAN” = a(i+j*dim1+k*dim1*dim2) “C”

для повышения локальности нужно следить за тем, чтобы внутренняя размерность массивов (первая) индексировалась индуктивной переменной самого внутреннего цикла:

```
for i
  for j
    for k
      A(k,j,i) // хорошо
      B(i,j,k) // плохо
```

– многомерные невыстраиваемые массивы

многомерные массивы в C вообще говоря являются массивами указателей на массивы (указателей на массивы и т.д. по размерностям); в связи с этим чтение одного элемента превращается в набор чтений количеством, равным размерности;

анализ зависимостей по адресам становится для компилятора весьма тяжелым;

– списки

обход элементов списка представляет собой цикл с рекуррентностью (зависимостью между итерациями) вида p=p->next, иными словами, адрес, по которому производится чтение на текущей итерации, зависит от результата чтения на предыдущей итерации;

при этом темп перебора элементов списка не превышает 1 элемента в 3 такта (в 24раза (!) меньше максимума при размере указателя 4 байта), при условии, что все читаемые элементы расположены в L1$, а в случае, когда все операции чтения промахиваются и в L1$, и в L2$, темп падает до 1 элемента в t_latency тактов (~75 для e3m, ~60 для e3s; отставание от темпа обработки массива ~100 раз); в связи с нерегулярностью адресов APB неприменим, но может быть эффективен механизм list-prefetch.

– деревья

деревья могут быть реализованы несколькими способами, но каждый из этих способов обладает тем же фундаментальным свойством, что и обычные списки: обход деревьев реализуется циклом с рекуррентностью по чтению из памяти; при этом, расположение перебираемых элементов дерева в памяти, как правило, еще хуже поддается упорядочению, чем множество перебираемых элементов списка;

– хэш-таблицы

хэш-таблицы, как правило, строятся на базе обычных массивов, при этом чтение элемента хэш-таблицы предваряется вычислением хэш-функции; доступ становится нерегулярным, поэтому APB к перебору элементов хэша неприменим, тем не менее, возможна предварительная подкачка элементов хэша, считываемых на следующих итерациях

Виды локальности данных

Там, где логика программы не диктует необходимости определенной локальности данных, можно делать выбор в пользу одного из следующих типов локальности:

– глобальные данные:

– местоположение – сегмент bss кода

– время жизни – вся программа

– адрес общедоступен

– не конфликтуют с другими данными (отсутствие конфликтов очевидно, если не было операций взятия адреса &glob)

– глобальные данные небольшого размера можно разместить на глобальных регистрах

– простые локальные данные без взятия адреса

– местоположение – регистры

– время жизни – до выхода из процедуры

– регистры не отображаются в память, ни с кем не конфликтуют

– сложные локальные данные, либо локальные данные со взятым адресом

– местоположение – пользовательский стек

– время жизни – до выхода из процедуры

– адрес доступен только внутри процедуры, для передачи данных в вызываемые процедуры нужно брать адрес

– в связи с часто необходимой операцией взятия адреса разрешение конфликтов по адресам становится более затруднительным

– динамические глобальные данные (malloc)

– местоположение – динамически выделяемая память

– время жизни – до динамического освобождения free

– адрес доступен через указатели

– конфликты по адресам разрешимы с затруднениями, не разрешимы конфликты между разными экземплярами malloc в цикле

– динамические локальные данные (alloca)

– местоположение – пользовательский стек

– время жизни – до выхода из процедуры

– адрес доступен через указатели

– конфликты по адресам разрешимы с затруднениями, не разрешимы конфликты между разными экземплярами malloc в цикле

Анализ и повышение производительности готовой программы

В этом разделе будет описан возможной метод поиска неоптимальных фрагментов задачи и предложен ряд методов повышения производительности.

Получение профиля; динамический профилировщик dprof

Анализ производительности задачи целесообразно начать со сбора профиля исполнения. Под профилем понимается распределение времени исполнения по тактам и процедурам задачи. Для получения профиля исполнения в составе ОПО для вычислительных комплексов “Эльбрус” предусмотрен динамический профилировщик dprof.

Назначение профилировщика dprof состоит в сборе информации о динамическом поведении программ. Он позволяет узнать общее количество

тактов исполнения задачи, распределение тактов исполнения между полезными тактами и тактами блокировки, собирать адреса команд с периодичностью по таймеру (обычный профиль) или по исчерпанию мониторных счетчиков, наблюдающих за какими-либо событиями (событийный профиль). На профилировщик dprof имеется штатная документацияю

Здесь нас в первую очередь интересует обычный профиль исполнения задачи.

Процесс получения текстового профиля состоит из трех шагов:

1) запуск задачи из-под профилировщика

dprof ./my_program my_parameters

2) получения текстового профиля файл при помощи дизассемблера

ldis –P my_program

При этом профиль будет иметь вид

%time	%summ	time	name
%времени	%времени	время	имя процедуры
работы	суммарный	работы	
32%	32%	624	my_function1
16%	48%	311	my_function2
3%	51%	61	my_function3

Из приведенного примера профиля видно, что почти половину времени задача проводит в процедурах my_function1,2, в то время как следующая по времени работы процедура my_function3 занимает всего 3% времени работы программы.

Следовательно, основные усилия поначалу целесообразно приложить к анализу первых двух процедур.

Может быть полезной более подробная информация о качестве исполнения кода. Для этого соберем доли времени работы процессора при исполнении

- такты исполнения команд
- такты запланированного пропуска команд
- такты блокировки процессора;

Команды и их примерная выдача

```
dprof –m EXEC ./my_program my_parameters
EXEC: 3604123154
dprof –m BUB_B ./my_program my_parameters
BUB_B: 1105421233
dprof –m BUB_E0 ./my_program my_parameters
BUB_E0: 251324143
```

```
dprof –m BUB_E2 ./my_program my_parameters
BUB_E2: 51324179
dprof –m IB_NO_COMMAND ./my_program my_parameters
IB_NO_COMMAND: 587064530
```

Время в тактах вычисляется как

```
EXEC+BUB_B+2*BUB_E0+4*BUB_E2+IB_NO_COMMAND
```

В нашем случае получается примерно 6*10^9 тактов, что составляет ~20с (частота e3m = 300МГц)

Видим, что запланированное время работы составляет ~4.7*10^9 тактов, что соответствует ~15.7с. Остальное время процессор простаивал из-за ожидания данных из памяти (BUB_E0*2, 1.7с), ожидания кода из памяти (IB_NO_COMMAND, 2.0с), другим причинам (BUB_E2*4, 0.7s).

Анализ процедуры: начальный этап

Для получения кода процедуры с профилем необходимо воспользоваться дизассемблером:

```
ldis –I m_program my_function1
```

В первую очередь необходимо определить тип анализируемой процедуры. Примерная классификация процедур с точки зрения анализа производительности выглядит следующим образом

1) Короткая ациклическая процедура (не более 30 тактов)

Такие процедуры просты для анализа; неоптимальность короткой процедуры как правило проявляется в виде

– плохой наполненности широких команд

– вследствие зацепления операций

– ввиду наличия длинных операций (деление, квадратный корень, вызов по косвенности)

– вследствие конфликтов между чтениями/записями

– блокировки(ок) от операций чтения

Общие рекомендации по исправлению найденных дефектов производительности:

– инлайн-подстановка: собирать в режиме –fwhole, использовать двухфазную компиляцию –fprofile,

– уменьшение длины зацепления

– принудительный разрыв конфликтов

– включение режима выноса чтений из процедур –fipo-invup,

– по возможности локализация данных для лучшего использования кэш-памяти.

2) Процедура с горячими простыми циклами/гнездами циклов

Анализ процедуры сводится к анализу работы горячих циклов. Наиболее частые проблемы:

– плохая наполненность широких команд

– не применился механизм apb

– блокировки после операций чтения из-за промахов в кэш

– блокировки из-за превышения пропускной способности устройства памяти

Предлагаемые пути решения означенных проблем:

– малое число итераций может привести к отказу от применения конвейеризации (как следствие к слабой наполненности широких команд), к отказу от использования механизма apb; если число итераций цикла объективно невелико (<5), следует рассмотреть возможность модификации алгоритма; если число итераций объективно велико, следует использовать двухфазную компиляцию –fprofile, либо добавить в исходный текст перед циклом подсказку

#pragma loop count(100)

– конвейеризированный цикл содержит длинную рекуррентность (длинно вычислимую зависимость между итерациями цикла); рекомендуется проверить цикл на наличие рекуррентности, в случае нахождения – оценить ее целесообразность

– механизм apb не применяется из-за нерегулярного изменения адреса; рекомендуется использовать в качестве цикловых счетчиков, определяющих адрес чтения, переменные типа long (не unsigned), не производить инкрементов счетчиков под условиями

– механизм apb не применяется при невозможности статического определения выравненности чтений по размеру; рекомендуется пользоваться опцией –faligned (входит в состав -ffast), подразумевающей выравненность адресов по размеру читаемого объекта

– блокировки от операций чтения из-за кэш-промахов (BUB_E0): рекомендуется попробовать опции –fcache-opt, -flist-prefetch, включающие режим предварительной подкачки данных в кэш

– блокировки по темпу работы памяти (BUB_E2): рекомендуется проверить темп обработки данных – сколько тактов работает цикл, сколько в нем операций чтения и записи, каков размер этих операций, какова локальность данных, какие данные могут быть найдены в кэше. Если темп существенно

ниже ожидаемого, возможно, проблема в неравномерности использования ресурсов кэша второго уровня

3) Сложный цикл с управлением/гнездо с управлением

Сложный цикл – цикл с управлением, несколькими обратными дугами. Некоторые сложные циклы при наличии точной профильной информации

(-fprofile) могут быть сведены к простым применением цикловых оптимизациям loop_nesting, loop_unswitching и некоторых других.

4) Громоздкая процедура

Громоздкие процедуры характеризуются неоднородным сложным управлением и размазанным профилем исполнения. Такие процедуры часто содержат циклы, как правило с небольшим числом итераций, вызовы других процедур. Они сложны как для оптимизации компилятором, так и для анализа производительности, и здесь мы можем дать только общие рекомендации:

– если процедура стала громоздкой вследствие inline-подстановок других процедур, можно попробовать ограничить применение inline опциями

– можно произвести вручную выделение важных фрагментов в отдельные процедуры

5) Процедура с превалирующим оператором switch.

Конструкции switch с большим числом альтернатив, как правило, обрабатываются компилятором достаточно эффективно при наличии адекватного профиля (см опцию -fprofile). Если конструкция switch имеет большое количество альтернатив с равномерно распределенной малой вероятностью, ее можно закодировать таблицей меток и косвенным переходом. Более эффективно при этом работают конструкции switch с плотным множеством значений, т.к. в случае разреженного множества значений switch таблица будет иметь большой размер.

6) Библиотечная процедура.

Библиотечная процедура собирается один раз, но используется в разных контекстах с различными параметрами. Если производительность задачи определяется производительностью библиотечной процедуры, то может быть целесообразно спрофилировать важную функцию и пересобрать ее вместе с задачей.

3.5. Технические характеристики вычислительного комплекса «Эльбрус 101-РС»

Вычислительный комплекс (ВК) «Эльбрус 101-РС» разработан на базе микропроцессора «Эльбрус-1С+» и предназначен для оборудования

автоматизированных рабочих мест операторов, тонких клиентов и информационных терминалов, применения в промышленной автоматизации и в системах с повышенными требованиями к информационной безопасности.

Вычислительный комплекс «Эльбрус 101-РС» предназначен для выполнения задач обмена, обработки, отображения информации в автоматизированном режиме под управлением операционной системы «ОС Альт» и использования в качестве персональной и/или терминальной вычислительной техники.

Системный блок вычислительного комплекса в формате nettop компактен, имеет малый уровень шума и потребления электроэнергии, допускает настенное размещение. Благодаря наличию двух независимых видеовыходов HDMI, возможно оборудование рабочего места двумя мониторами высокой чёткости с разрешением Full HD. Предусмотрено расширение функциональности путём установки внутренних модулей в слот mini PCI Express. ВК «Эльбрус 101-РС» изображен на рисунке 3.6, а технические характеристики представлены в таблицах 3.1 и 3.2.

Рисунок 3.6. Системный блок ВК «Эльбрус 101-РС» — вид сзади

Таблица 3.1. Технические характеристики «Эльбрус 101-РС»

Технические характеристики	
Набор микросхем	1 процессор Эльбрус-1С+ (1891ВМ11Я — 1 ядро, 1000 МГц) 1 южный мост КПИ-2 (1991ВГ2Я)
Оперативная память	2 слота DIMM DDR3-1600 registered ECC установлено 16 Гбайт, поддерживается до 32 Гбайт
Долговременная память	1 слот mSATA 3.0, установлен SSD-накопитель объёмом 120 Гбайт 7 портов SATA 3.0, штатный корпус не имеет разъёмов питания SATA
Видеосистема	интегрированное в центральный процессор

Технические характеристики	
	видеоядро аппаратное ускорение 2D- и 3D-графики вывод на 2 монитора высокой чёткости 1920×1080 (Full HD) вывод на 1 монитор сверхвысокой чёткости 2560×1440 (Quad HD)
Аудиосистема	5.1-канальный звук через интегрированный контроллер 1 вход для микрофона, 1 линейный вход
Внешние интерфейсы	2 порта видео с разъёмами HDMI 6 портов аудио с разъёмами «гнездо» 3,5 мм 3 порта сети Ethernet 1000Base-T с разъёмами RJ-45 6 портов периферии USB 2.0 с разъёмами типа A 2 порта консоли RS-232 с разъёмами DE-9
Внутренние интерфейсы	2 порта LVDS с разъёмами 40-pin для видео высокой чёткости 1 порт HD Audio с разъёмом 10-pin для передних гнёзд 1 колодка USB 2.0 с разъёмом 10-pin для 2 передних портов 1 колодка FP с разъёмом 8-pin для индикаторов и кнопок на корпусе 2 колодки GPIO с разъёмами 8-pin и 3-pin для прямого ввода-вывода 2 порта JTAG с разъёмами 10-pin для диагностики ЦП и КПИ 1 колодка с разъёмом 8-pin для перезаписи ППЗУ
Слоты расширения	1 слот mini PCI Express 2.0 формата x4 1 слот PCI Express 2.0 формата x16 — использование слота x16 в штатном корпусе не предусмотрено
Системный блок	корпус формата nettop, материнская плата формата Mini-ITX габариты 190×196×62 мм, вес 1,87 кг 1 преобразователь питания мощностью 120 Вт, потребление до 40 Вт 1 вентилятор под верхней крышкой (120 мм) 4 места для вентиляторов сбоку (40 мм) 4 отверстия снизу для настенного крепления VESA 100×100 мм
Блок питания	внешний адаптер AC-DC для компактных компьютеров

Технические характеристики	
	вход 220 В, выход 12 В, 5 А, мощность до 60 Вт габариты 120×54×32 мм

Эксплуатационные параметры	
Условия эксплуатации	умеренно холодный климат (УХЛ), группа 1.1 по ГОСТ 15150 температура 0…+35 °С, предельная температура −20…+50 °С влажность до 85 % без конденсата давление 630…800 мм рт. ст.
Условия хранения	согласно группе 1 (Л) по ГОСТ 15150 температура +5…+40 °С влажность до 80 % максимум, до 65 % во влажный период
Электропитание	220 В ± 10 %, 50 Гц ±2 %
Срок службы	12 лет
Срок хранения	5 лет
Документация	ТВГИ.466535.181

Таблица 3.2. Комплектация рабочей станции «Эльбрус 101-РС»

Материнская плата	MBE1C-PC
Интегрированные микросхемы	1 процессор Эльбрус-1С+ (1891ВМ11Я) 1 контроллер периферийных интерфейсов КПИ-2 (1991ВГ2Я)
Оперативная память	2 модуля Kingston ValueRAM KVR16R11D8/8
Долговременная память	1 твердотельный диск Kingston SSDnow SMS200S3/120G
Системный блок	корпус Morex 557P-60W 1 преобразователь питания Realan LR1109-120W12VDC 1 вентилятор Scythe SY1212SL12SL
Блок питания	адаптер Seasonic SSA-0601HE-12

Список литературы

Основная литература

1. Олифер В.Г. Компьютерные сети. Принципы, технологии, протоколы, 4-е издание. - М.: Питер, 2014. - 943с.
2. Семенов Ю.А. Протоколы и алгоритмы маршрутизации в Интернет [Электронный ресурс]/ Семенов Ю.А.— Электрон. текстовые данные.— М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016.— 998 с.— Режим доступа: http://www.iprbookshop.ru/62826.html.— ЭБС «IPRbooks

Дополнительная литература

3. Берлин А.Н. Основные протоколы Интернет [Электронный ресурс]/ Берлин А.Н.— Электрон. текстовые данные.— М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016.— 602 с.— Режим доступа: http://www.iprbookshop.ru/52181.— ЭБС «IPRbooks»
4. Зиглер Роберт Л., Брандмауэры в Linux. : Пер. с англ. : М. : Издательский дом “Вильямс”, 2001. – 384 с.
5. Oskar Andreasson, Руководство по iptables (Iptables Tutorial 1.1.19), : Пер. С англ. : Андрей Киселёв, 2003. – 137 с.
6. Сетевые средства Linux [Электронный ресурс] // ИНТУИТ, Национальный открытый университет. URL: https://www.intuit.ru/studies/courses/681/537/info
7. Основные протоколы интернет [Электронный ресурс] // ИНТУИТ, Национальный открытый университет. URL: https://www.intuit.ru/studies/courses/2/2/info
8. Операционные системы ОС Альт [Электронный ресурс] // Базальт СПО. URL: https://www.basealt.ru/products/alt-server/
9. Postel, J. (ed.), "Internet Protocol - DARPA Internet Program Protocol Specification," RFC 791, USC/Information Sciences Institute, September 1981. - 45 p. https://tools.ietf.org/html/rfc791
10. J. Postel. “User Datagram Protocol”, RFC 768, 1980, - 3 p. https://tools.ietf.org/html/rfc768
11. TRANSMISSION CONTROL PROTOCOL - DARPA Internet Program Protocol Specification, RFC 793, 1981. – 85 p. https://tools.ietf.org/html/rfc793
12. INTERNET CONTROL MESSAGE PROTOCOL - DARPA Internet Program Protocol Specification, RFC 792, 1981, - 21 p. https://tools.ietf.org/html/rfc792
13. Internet Protocol, Version 6 (IPv6) Specification Internet Engineering Task Force (IETF) RFC 8200, 2017. – 41 p. https://tools.ietf.org/html/rfc8200

Подписано в печать 06.06.2019
Формат 60x90/16
Печать офсетная
Усл. печ. л. 23,5
Тираж 200, Заказ № 23871
ООО «Фабрика Офсетной Печати»
www.fop.ru