Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного бюджетного образовательного
учреждения высшего образования
**«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»**
(КФ МГТУ им. Н.Э. Баумана)

Е.В. Красавин, Е.А. Черепков, А.В. Козина

# БАЗОВАЯ НАСТРОЙКА СЕТИ. МАРШРУТИЗАЦИЯ

Методические указания к лабораторной работе

по дисциплине «Операционные системы»

Калуга – 2019

УДК 004.62
ББК 32.972.1
К78

Методические указания составлены в соответствии с учебным планом КФ МГТУ им. Н.Э. Баумана по направлению подготовки 09.03.04 «Программная инженерия» кафедры «Программного обеспечения ЭВМ, информационных технологий».

Методические указания рассмотрены и одобрены:

- Кафедрой «Программного обеспечения ЭВМ, информационных технологий» (ИУ4-КФ) протокол № 51.4/6 от «20» февраля 2019 г.

Зав. кафедрой ИУ4-КФ _____ к.т.н., доцент Ю.Е. Гагарин

- Методической комиссией факультета ИУ-КФ протокол № 9 от «04» 03 2019 г.

Председатель методической
комиссии факультета ИУ-КФ _____ к.т.н., доцент М.Ю. Адкин

- Методической комиссией
КФ МГТУ им.Н.Э. Баумана протокол № 5 от «5» 03 2019 г.

Председатель методической комиссии
КФ МГТУ им.Н.Э. Баумана _____ д.э.н., профессор О.Л. Перерва

Рецензент:
к.т.н., доцент кафедры ИУ6-КФ _____ А.Б. Лачихина

Авторы
к.т.н., доцент кафедры ИУ4-КФ _____ Е.В. Красавин
ассистент кафедры ИУ4-КФ _____ Е.А. Черепков
ассистент кафедры ИУ4-КФ _____ А.В. Козина

Аннотация

Методические указания к выполнению лабораторной работы по курсу «Операционные системы» содержат описание настройки сетевых интерфейсов и маршрутизации.

Предназначены для студентов 3-го курса бакалавриата КФ МГТУ им. Н.Э. Баумана, обучающихся по направлению подготовки 09.03.04 «Программная инженерия».

**ОГЛАВЛЕНИЕ**

# ВВЕДЕНИЕ

Настоящие методические указания составлены в соответствии с программой проведения лабораторных работ по курсу «Операционные системы» на кафедре «Программное обеспечение ЭВМ, информационные технологии» факультета «Информатика и управление» Калужского филиала МГТУ им. Н.Э. Баумана.

Методические указания, ориентированные на студентов 3-го курса направления подготовки 09.03.04 «Программная инженерия», содержат описание настройки сетевых интерфейсов и маршрутизации.

Методические указания составлены для ознакомления студентов с настройкой сетевых интерфейсов и маршрутизацией в операционной системе FreeBSD. Для выполнения лабораторной работы студенту необходимы минимальные знания об операционной системе Linux.

## ЦЕЛЬ И ЗАДАЧИ РАБОТЫ, ТРЕБОВАНИЯ К РЕЗУЛЬТАТАМ ЕЕ ВЫПОЛНЕНИЯ

Целью выполнения лабораторной работы является приобретение практических навыков работы в среде ОС FreeBSD и ее администрирования.

Основными задачами выполнения лабораторной работы являются:

1. Научиться изучать и настраивать сетевые интерфейсы компьютера
2. Научиться изучать и настраивать таблицу маршрутизации

Результатами работы являются:

1. Демонстрация настроенного сетевого интерфейса и таблицы маршрутизации.
2. Подготовленный отчет.

## КРАТКАЯ ХАРАКТЕРИСТИКА ОБЪЕКТА ИЗУЧЕНИЯ, ИССЛЕДОВАНИЯ

В наши дни мы не представляем себе компьютера без сетевого подключения. Добавление и настройка сетевой карты — это обычная задача любого администратора FreeBSD.

**Поиск подходящего драйвера**

В первую очередь определите тип используемой карты (PCI или ISA), модель карты и используемый в ней чип. FreeBSD поддерживает многие PCI и ISA карты. Обратитесь к Списку поддерживаемого оборудования вашего релиза чтобы узнать, поддерживается ли карта.

Как только вы убедились, что карта поддерживается, потребуется определить подходящий драйвер. В файлах /usr/src/sys/conf/NOTES и /usr/src/sys/arch/conf/NOTES находится список драйверов сетевых интерфейсов с информацией о поддерживаемых чипсетах/картах. Если вы сомневаетесь в том, какой драйвер подойдет, прочтите страницу справочника к драйверу. Страница справочника содержит больше информации о поддерживаемом оборудовании и даже о проблемах, которые могут возникнуть.

Если ваша карта широко распространена, вам скорее всего не потребуется долго искать драйвер. Драйверы для широко распространенных карт представлены в ядре GENERIC, так что ваша карта должна определиться при загрузке, примерно так:

```
  dc0: <82c169 PNIC 10/100BaseTX> port 0xa000-0xa0ff mem
0xd3800000-0xd38 000ff irq 15 at device 11.0 on pci0
  dc0: Ethernet address: 00:a0:cc:da:da:da
  miibus0: <MII bus> on dc0
  ukphy0: <Generic IEEE 802.3u media interface> on
miibus0
  ukphy0:  10baseT, 10baseT-FDX, 100baseTX, 100baseTX-
FDX, auto
  dc1: <82c169 PNIC 10/100BaseTX> port 0x9800-0x98ff mem
0xd3000000-0xd30 000ff irq 11 at device 12.0 on pci0
```

```
dc1: Ethernet address: 00:a0:cc:da:da:db
miibus1: <MII bus> on dc1
ukphy1: <Generic IEEE 802.3u media interface> on miibus1
ukphy1: 10baseT, 10baseT-FDX, 100baseTX, 100baseTX-FDX, auto
```

В этом примере две карты используют имеющийся в системе драйвер dc.

Если драйвер вашей сетевой карты отсутствует в GENERIC, для ее использования потребуется загрузить подходящий драйвер. Это может быть сделано одним из двух способов:

Простейший способ — просто загрузить модуль ядра сетевой карты с помощью kldload.

Не все драйверы доступны в виде модулей; например, модули отсутствуют для ISA карт.

Вместо этого, вы можете статически включить поддержку карты, скомпилировав собственное ядро. Информацию о том, какие параметры нужно включать в ядро, можно получить из /usr/src/sys/conf/NOTES, /usr/src/sys/arch/conf/NOTES и страницы справочника драйвера сетевой карты. Если карта была обнаружена вашим ядром (GENERIC) во время загрузки, собирать ядро не потребуется.

## НАСТРОЙКА СЕТЕВЫХ ИНТЕРФЕЙСОВ

Как только для сетевой карты загружен подходящий драйвер, ее потребуется настроить. Как и многое другое, сетевая карта может быть настроена во время установки с помощью bsdinstall.

Для вывода информации о настройке сетевых интерфейсов системы, введите следующую команду:

```
% ifconfig
 dc0:   flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,
        MULTICAST> mtu 1500 inet 192.168.1.3 netmask
        0xffffff00 broadcast 192.168.1.255
        ether 00:a0:cc:da:da:da
        media: Ethernet autoselect (100baseTX <full-
        duplex>)
        status: active

 dc1:   flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,
        MULTICAST> mtu 1500 inet 10.0.0.1 netmask
        0xffffff00 broadcast 10.0.0.255
        ether 00:a0:cc:da:da:db
        media: Ethernet 10baseT/UTP
        status: no carrier

 lp0:   flags=8810<POINTOPOINT,SIMPLEX,MULTICAST> mtu
        1500
 lo0:   flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu
        16384
        inet 127.0.0.1 netmask 0xff000000
 tun0:  flags=8010<POINTOPOINT,MULTICAST> mtu 1500
```

*Примечание*

Старые версии FreeBSD могут потребовать запуска ifconfig с параметром -a, за более подробным описанием синтаксиса ifconfig обращайтесь к странице справочника. Учтите также, что строки, относящиеся к IPv6 (inet6 и т.п.) убраны из этого примера.

В этом примере были показаны следующие устройства:

- dc0: первый Ethernet интерфейс
- dc1: второй Ethernet интерфейс
- lp0: интерфейс параллельного порта
- lo0: устройство loopback
- tun0: туннельное устройство, используемое ppp

Для присвоения имени сетевой карте FreeBSD использует имя драйвера и порядковый номер, в котором карта обнаруживается при инициализации устройств. Например, sis2 это третья сетевая карта, использующая драйвер sis.

В этом примере, устройство dc0 включено и работает. Ключевые признаки таковы:

1. UP означает, что карта настроена и готова.
2. У карты есть интернет (inet) адрес (в данном случае 192.168.1.3).
3. Установлена маска подсети (netmask; 0xffffff00, то же, что и 255.255.255.0).
4. Широковещательный адрес (в данном случае, 192.168.1.255).
5. Значение MAC адреса карты (ether) 00:a0:cc:da:da:da
6. Выбор физической среды передачи данных в режиме авто выбора (media: Ethernet autoselect (100baseTX <full-duplex>)). Мы видим, что dc1 была настроена для работы с 10baseT/UTP. За более подробной информацией о доступных драйверу типах среды обращайтесь к странице справочника.
7. Статус соединения (status) active, т.е. несущая обнаружена. Для dc1, мы видим status: no carrier. Это нормально, когда Ethernet кабель не подключен к карте.

Если ifconfig показывает примерно следующее:

```
dc0: flags=8843<BROADCAST,SIMPLEX,MULTICAST> mtu 1500
ether 00:a0:cc:da:da:da
```

это означает, что карта не была настроена.

Для настройки карты вам потребуются привилегии пользователя root. Настройка сетевой карты может быть выполнена из командной строки с помощью ifconfig, но вам потребуется делать это после каждой перезагрузки системы. Подходящее место для настройки сетевых карт — это файл /etc/rc.conf.

Откройте /etc/rc.conf в текстовом редакторе. Вам потребуется добавить строку для каждой сетевой карты, имеющейся в системе, например, в нашем случае, было добавлено две строки:

```
ifconfig_dc0="inet 192.168.1.3 netmask 255.255.255.0"
ifconfig_dc1="inet 10.0.0.1 netmask 255.255.255.0 media 10baseT/UTP"
```

Замените dc0, dc1, и так далее на соответствующие имена ваших карт, подставьте соответствующие адреса. Обратитесь к страницам справочника сетевой карты и ifconfig, за подробной информацией о доступных опциях и к странице справочника rc.conf за дополнительной информацией о синтаксисе /etc/rc.conf.

Если вы настроили сетевую карту в процессе установки системы, некоторые строки, касающиеся сетевой карты, могут уже присутствовать. Внимательно проверьте /etc/rc.conf перед добавлением каких-либо строк.

Отредактируйте также файл /etc/hosts для добавления имен и IP адресов различных компьютеров сети, если их еще там нет. За дополнительной информацией обращайтесь к man.hosts.5; и к /usr/share/examples/etc/hosts.

**Тестирование и решение проблем**

Как только вы внесете необходимые изменения в /etc/rc.conf, перегрузите компьютер. Изменения настроек интерфейсов будут применены, кроме того будет проверена правильность настроек.

Как только система перезагрузится, проверьте сетевые интерфейсы.

**Проверка Ethernet карты**

Для проверки правильности настройки сетевой карты, попробуйте выполнить ping для самого интерфейса, а затем для другой машины в локальной сети.

Сначала проверьте локальный интерфейс:

```
% ping -c5 192.168.1.3
PING 192.168.1.3 (192.168.1.3): 56 data bytes
bytes from 192.168.1.3: icmp_seq=0 ttl=64 time=0.082 ms
bytes from 192.168.1.3: icmp_seq=1 ttl=64 time=0.074 ms
bytes from 192.168.1.3: icmp_seq=2 ttl=64 time=0.076 ms
bytes from 192.168.1.3: icmp_seq=3 ttl=64 time=0.108 ms
bytes from 192.168.1.3: icmp_seq=4 ttl=64 time=0.076 ms

--- 192.168.1.3 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.074/0.083/0.108/0.013 ms
```

Затем проверьте другую машину в локальной сети:

```
% ping -c5 192.168.1.2
PING 192.168.1.2 (192.168.1.2): 56 data bytes
bytes from 192.168.1.2: icmp_seq=0 ttl=64 time=0.726 ms
bytes from 192.168.1.2: icmp_seq=1 ttl=64 time=0.766 ms
bytes from 192.168.1.2: icmp_seq=2 ttl=64 time=0.700 ms
bytes from 192.168.1.2: icmp_seq=3 ttl=64 time=0.747 ms
bytes from 192.168.1.2: icmp_seq=4 ttl=64 time=0.704 ms

--- 192.168.1.2 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.700/0.729/0.766/0.025 ms
```

Вы можете также использовать имя машины вместо 192.168.1.2, если настроен файл /etc/hosts.

## НАСТРОЙКА ВИРТУАЛЬНЫХ СЕРВЕРОВ

Очень часто FreeBSD используется для размещения сайтов, когда один сервер работает в сети как несколько серверов. Это достигается присвоением нескольких сетевых адресов одному интерфейсу.

У сетевого интерфейса всегда есть один "настоящий" адрес, хотя он может иметь любое количество "синонимов" (alias). Эти синонимы обычно добавляются путём помещения соответствующих записей в /etc/rc.conf.

Синоним для интерфейса fxp0 выглядит следующим образом:

```
ifconfig_fxp0_alias0="inet xxx.xxx.xxx.xxx netmask xxx.xxx.xxx.xxx"
```

Заметьте, что записи синонимов должны начинаться с alias0 и идти далее в определенном порядке (например, _alias1, _alias2, и т.д.). Конфигурационный процесс остановится на первом по порядку отсутствующем числе.

Определение маски подсети для синонима очень важно, но к счастью, так же просто. Для каждого интерфейса должен быть один адрес с истинной маской подсети. Любой другой адрес в сети должен иметь маску подсети, состоящую из всех единичек (что выражается как 255.255.255.255 или как 0xffffffff).

Например, рассмотрим случай, когда интерфейс fxp0 подключён к двум сетям, к сети 10.1.1.0 с маской подсети 255.255.255.0 и к сети 202.0.75.16 с маской 255.255.255.240. Мы хотим, чтобы система была видна по IP, начиная с 10.1.1.1 по 10.1.1.5 и с 202.0.75.17 по 202.0.75.20. Как было сказано выше, только первый адрес в заданном диапазоне (в данном случае, 10.0.1.1 и 202.0.75.17) должен иметь реальную маску сети; все остальные (с 10.1.1.2 по 10.1.1.5 и с 202.0.75.18 по 202.0.75.20) должны быть сконфигурированы с маской сети 255.255.255.255.

Для этого в файл /etc/rc.conf должны быть внесены следующие записи:

```
  ifconfig_fxp0="inet 10.1.1.1 netmask 255.255.255.0"
  ifconfig_fxp0_alias0="inet      10.1.1.2      netmask
255.255.255.255"
  ifconfig_fxp0_alias1="inet      10.1.1.3      netmask
255.255.255.255"
  ifconfig_fxp0_alias2="inet      10.1.1.4      netmask
255.255.255.255"
  ifconfig_fxp0_alias3="inet      10.1.1.5      netmask
255.255.255.255"
  ifconfig_fxp0_alias4="inet     202.0.75.17     netmask
255.255.255.240"
  ifconfig_fxp0_alias5="inet     202.0.75.18     netmask
255.255.255.255"
  ifconfig_fxp0_alias6="inet     202.0.75.19     netmask
255.255.255.255"
  ifconfig_fxp0_alias7="inet     202.0.75.20     netmask
255.255.255.255"
```

**Автоматическая настройка сети (DHCP)**

DHCP, или Dynamic Host Configuration Protocol (Протокол Динамической Конфигурации Хостов), описывает порядок, по которому система может подключиться к сети и получить необходимую информацию для работы в ней. Во FreeBSD используется dhclient , импортированный из OpenBSD 3.7. Вся информация здесь, относительно dhclient относится либо к ISC, либо к DHCP клиентам.DHCP сервер включён в ISC дистрибутив.

Когда на клиентской машине выполняется программа dhclient, являющаяся клиентом DHCP, она начинает широковещательную рассылку запросов на получение настроечной информации. По умолчанию эти запросы делаются на 68 порт UDP. Сервер отвечает на UDP 67, выдавая клиенту адрес IP и другую необходимую информацию, такую, как сетевую маску, маршрутизатор и серверы DNS. Вся эта информация даётся в форме "аренды" DHCP и верна только определенное время (что настраивается администратором сервера DHCP). При таком подходе устаревшие адреса IP тех клиентов,

которые больше не подключены к сети, могут автоматически использоваться повторно.

Клиенты DHCP могут получить от сервера очень много информации. Подробный список находится в странице Справочника dhcp-options.

**Интеграция с FreeBSD**

DHCP клиент от OpenBSD, dhclient, полностью интегрирован во FreeBSD. Поддержка клиента DHCP есть как в программе установки, так и в самой системе, что исключает необходимость в знании подробностей конфигурации сети в любой сети, имеющей сервер DHCP. Утилита dhclient включена во все версии FreeBSD, начиная с 3.2.

DHCP поддерживается утилитой sysinstall. При настройке сетевого интерфейса из программы sysinstall второй вопрос, который вам задается: "Do you want to try DHCP configuration of the interface?" ("Хотите ли вы попробовать настроить этот интерфейс через DHCP?"). Утвердительный ответ приведёт к запуску программы dhclient, и при удачном его выполнении к автоматическому заданию информации для настройки интерфейса.

Есть две вещи, которые вы должны сделать для того, чтобы ваша система использовала DHCP при загрузке:

- Убедитесь, что устройство bpf включено в компиляцию вашего ядра. Чтобы это сделать, добавьте строчку device bpf в конфигурационный файл ядра и перестройте ядро.
- Устройство bpf уже является частью ядра GENERIC, которое поставляется вместе с FreeBSD, так что, если вы не используете другое ядро, то вам и не нужно его делать для того, чтобы работал DHCP.

*Примечание*

Те, кто беспокоится о безопасности, должны иметь в виду, что устройство bpf является также тем самым устройством, которое позволяет работать программам-снифферам пакетов (хотя для этого они должны быть запущены пользователем root). Наличие устройства

bpf необходимо для использования DHCP, но если вы чересчур беспокоитесь о безопасности, то вам нельзя добавлять устройство bpf в ядро только для того, чтобы в неопределённом будущем использовать DHCP.

По умолчанию, конфигурирование FreeBSD по протоколу DHCP выполняется фоновым процессом, или асинхронно. Остальные стартовые скрипты продолжают работу, не ожидая завершения процесса конфигурирования, тем самым ускоряя загрузку системы.

Фоновое конфигурирование не создает проблем в случае, если сервер DHCP быстро отвечает на запросы, и процесс конфигурирования происходит быстро. Однако, в некоторых случаях настройка по DHCP может длиться значительное время. При этом запуск сетевых сервисов может потерпеть неудачу, если будет произведен ранее завершения конфигурирования по DHCP. Запуск DHCP в синхронном режиме предотвращает проблему, откладывая выполнение остальных стартовых скриптов до момента завершения конфигурирования по DHCP.

Для осуществления фонового конфигурирования по DHCP (асинхронный режим), используйте значение «DHCP» в /etc/rc.conf :

```
ifconfig_fxp0="DHCP"
```

Для откладывания запуска стартовых скриптов до завершения конфигурирования по DHCP (синхронный режим), укажите значение «SYNCDHCP»:

```
ifconfig_fxp0="SYNCDHCP"
```

*Примечание*

Обязательно замените fxp0 на имя интерфейса, который вы хотите настраивать динамически.

Если dhclient в вашей системе находится в другом месте или если вы хотите задать дополнительные параметры для dhclient, то также укажите следующее (изменив так, как вам нужно):

```
dhcp_program="/sbin/dhclient"
dhcp_flags=""
```

Сервер DHCP, dhcpd, включён как часть порта net/isc-dhcp3-server в коллекцию портов. Этот порт содержит DHCP-сервер от ISC и документацию.

**Файлы**

- */etc/dhclient.conf*

dhclient требует наличия конфигурационного файла, /etc/dhclient.conf. Как правило, файл содержит только комментарии, а настройки по умолчанию достаточно хороши. Этот настроечный файл описан на страницах справочной системы по dhclient.conf.

- */sbin/dhclient*

dhclient скомпонован статически и находится в каталоге /sbin. На страницу справочника dhclient дается более подробная информация о dhclient.

- */sbin/dhclient-script*

dhclient-script является специфичным для FreeBSD скриптом настройки клиента DHCP. Он описан в dhclient-script, но для нормального функционирования никаких модификаций со стороны пользователя не требуется.

- */var/db/dhclient.leases*

В этом файле клиент DHCP хранит базу данных выданных к использованию адресов в виде журнала. На странице dhclient.leases дается гораздо более подробное описание.

Полное описание протокола DHCP дается в RFC 2131 (http://www.freesoft.org/CIE/RFC/2131/). Кроме того, дополнительная информация есть на сервере http://www.dhcp.org/.

**Установка и настройка сервера DHCP**

Серверная часть пакета не поставляется как часть FreeBSD, так что вам потребуется установить порт net/isc-dhcp3-relay для получения этого сервиса.

Для того, чтобы настроить систему FreeBSD на работу в качестве сервера DHCP, вам необходимо обеспечить присутствие устройства bpf, вкомпилированного в ядро. Для этого добавьте строку device bpf в файл конфигурации вашего ядра.

Устройство bpf уже входит в состав ядра GENERIC, поставляемого с FreeBSD, так что вам не нужно создавать собственное ядро для обеспечения работы DHCP.

*Примечание*

Те, кто обращает особое внимание на вопросы безопасности, должны заметить, что bpf является тем устройством, что позволяет нормально работать снифферам пакетов (хотя таким программам требуются привилегированный доступ). Наличие устройства bpf обязательно для использования DHCP, но если вы очень обеспокоены безопасностью, наверное, вам не нужно включать bpf в ваше ядро только потому, что в отдалённом будущем вы собираетесь использовать DHCP.

Следующим действием, которое вам нужно выполнить, является редактирование примерного dhcpd.conf, который устанавливается в составе порта net/isc-dhcp3-server. По умолчанию это файл /usr/local/etc/dhcpd.conf.sample, и вы должны скопировать его в файл /usr/local/etc/dhcpd.conf перед тем, как его редактировать.

**Настройка сервера DHCP**

dhcpd.conf состоит из деклараций относительно подсетей и хостов, и проще всего описывается на примере:

```
option domain-name "example.com"; (1)
option domain-name-servers 192.168.4.100; (2)
option subnet-mask 255.255.255.0; (3)
```

```
default-lease-time 3600; (4)
max-lease-time 86400; (5)
ddns-update-style none; (6)

subnet 192.168.4.0 netmask 255.255.255.0 {
range  192.168.4.129 192.168.4.254; (7)
option routers 192.168.4.1; (8)
}

host mailhost {
hardware ethernet 02:03:04:05:06:07; (9)
fixed-address mailhost.example.com; (10)
}
```

1. Этот параметр задаёт домен, который будет выдаваться клиентам в качестве домена, используемого по умолчанию при поиске. Обратитесь к страницам справочной системы по resolv.conf для получения дополнительной информации о том, что это значит.
2. Этот параметр задаёт список разделённых запятыми серверов DNS, которые должен использовать клиент.
3. Маска сети, которая будет выдаваться клиентам.
4. Клиент может запросить определённое время, которое будет действовать выданная информация. В противном случае сервер выдаст настройки с этим сроком (в секундах).
5. Это максимальное время, на которое сервер будет выдавать конфигурацию. Если клиент запросит больший срок, он будет подтверждён, но будет действовать только max-lease-time секунд.
6. Этот параметр задаёт, будет ли сервер DHCP пытаться обновить DNS при выдаче или освобождении конфигурационной информации. В реализации ISC этот параметр является *обязательным*.

7. Это определение того, какие IP-адреса должны использоваться в качестве резерва для выдачи клиентам. IP-адреса между и включая границы, будут выдаваться клиентам.
8. Объявление маршрутизатора, используемого по умолчанию, который будет выдаваться клиентам.
9. Аппаратный MAC-адрес хоста (чтобы сервер DHCP мог распознать хост, когда тот делает запрос).
10. Определение того, что хосту всегда будет выдаваться один и тот же IP-адрес. Заметьте, что указание здесь имени хоста корректно, так как сервер DHCP будет разрешать имя хоста самостоятельно до того, как выдать конфигурационную информацию.

Когда вы закончите составлять свой dhcpd.conf, нужно разрешить запуск сервера DHCP в файле /etc/rc.conf, добавив в него строки

```
dhcpd_enable="YES"
dhcpd_ifaces="dc0"
```

Замените dc0 именем интерфейса (или именами интерфейсов, разделяя их пробелами), на котором(ых) сервер DHCP должен принимать запросы от клиентов.

Затем вы можете стартовать сервер DHCP при помощи команды

```
# /usr/local/etc/rc.d/isc-dhcpd.sh start
```

Если в будущем вам понадобится сделать изменения в настройке вашего сервера, то важно заметить, что посылка сигнала SIGHUP приложению dhcpd не приведёт к перезагрузке настроек, как это бывает для большинства даемонов. Вам нужно послать сигнал SIGTERM для остановки процесса, а затем перезапустить его при помощи вышеприведённой команды.

**Файлы**

- */usr/local/sbin/dhcpd*

dhcpd скомпонован статически и расположен в каталоге /usr/local/sbin. Страницы справочной системы dhcpd, устанавливаемые портом, содержат более полную информацию о dhcpd.

- */usr/local/etc/dhcpd.conf*

dhcpd требует наличия конфигурационного файла, /usr/local/etc/dhcpd.conf, до того, как он будет запущен и начнёт предоставлять сервис клиентам. Необходимо, чтобы этот файл содержал все данные, которая будет выдаваться обслуживаемым клиентам, а также информацию о работе сервера. Этот конфигурационный файл описывается на страницах справочной системы dhcpd.conf, которые устанавливаются портом.

- */var/db/dhcpd.leases*

Сервер DHCP ведёт базу данных выданной информации в этом файле, который записывается в виде протокола. Страницы справочной системы dhcpd.leases, устанавливаемые портом, дают гораздо более подробное описание.

- */usr/local/sbin/dhcrelay*

dhcrelay используется в сложных ситуациях, когда сервер DHCP пересылает запросы от клиента другому серверу DHCP в отдельной сети. Если вам нужна такая функциональность, то установите порт net/isc-dhcp3-server. На страницах справочной системы dhcrelay, которые устанавливаются портом, даётся более полное описание.

# МАРШРУТИЗАЦИЯ

## Сетевые шлюзы и маршруты

Чтобы некоторая машина могла найти в сети другую, должен иметься механизм описания того, как добраться от одной машине к другой. Такой механизм называется маршрутизацией. "Маршрут" задаётся парой адресов: "адресом назначения" (destination) и "сетевым шлюзом" (gateway). Эта пара указывает на то, что если Вы пытаетесь соединиться с адресом назначения, то вам нужно устанавливать связь через "сетевой шлюз". Существует три типа адресов назначения: отдельные хосты, подсети и "маршрут по умолчанию" (default). "Маршрут по умолчанию" (default route) используется, если не подходит ни один из других маршрутов. Мы поговорим немного подробнее о маршрутах по умолчанию позже. Также имеется и три типа сетевых шлюзов: отдельные хосты, интерфейсы (также называемые "подключениями" (links)) и аппаратные адреса Ethernet (MAC-адреса).

Пример. Для иллюстрации различных аспектов маршрутизации мы будем использовать следующий пример использования команды netstat:

```
netstat -r
Routing tables
Destination    Gateway    Flags     Refs Use      Netif
default        outside-gw UGSc      37   418      ppp0
localhost      localhost  UH        0    181      lo0
test0  0:e0:b5:36:cf:4f  UHLW      5    63288    ed0
10.20.30.255   link#1     UHLW      1    2421
example.com    link#1     UC        0    0
               0:e0:a8:37
host1          :8:1e      UHLW      3    4601     lo0
host2  0:e0:a8:37:8:1e    UHLW      0    5        lo0
host2.example.com link#1  UC        0    0
224            link#1     UC        0    0
```

В первых двух строках задаются маршрут по умолчанию и маршрут на localhost.

Интерфейс (колонка Netif), который указан в этой таблице маршрутов для использования localhost и который назван lo0, имеет также второе название, устройство loopback. Это значит сохранение всего трафика для указанного адреса назначения внутри, без посылки его по сети, так как он все равно будет направлен туда, где был создан.

Следующими выделяющимися адресами являются адреса, начинающиеся с 0:e0:.... Это аппаратные адреса Ethernet, или MAC-адреса. FreeBSD будет автоматически распознавать любой хост (в нашем примере это test0) в локальной сети Ethernet и добавит маршрут для этого хоста, указывающий непосредственно на интерфейс Ethernet, ed0. С этим типом маршрута также связан параметр таймаута (колонка Expire), используемый в случае неудачной попытки услышать этот хост в течении некоторого периода времени. Если такое происходит, то маршрут до этого хоста будет автоматически удалён. Такие хосты поддерживаются при помощи механизма, известного как RIP (Routing Information Protocol), который вычисляет маршруты к хостам локальной сети при помощи определения кратчайшего расстояния.

FreeBSD добавит также все маршруты к подсетям для локальных подсетей (10.20.30.255 является широковещательным адресом для подсети 10.20.30, а имя example.com является именем домена, связанным с этой подсетью). Назначение link#1 соответствует первому адаптеру Ethernet в машине. Отметьте отсутствие дополнительного интерфейса для этих строк.

В обеих этих группах (хосты и подсети локальной сети) маршруты конфигурируются автоматически даемоном, который называется routed. Если он не запущен, то будут существовать только статически заданные (то есть введенные явно) маршруты.

Строка host1 относится к нашему хосту, который известен по адресу Ethernet. Так как мы являемся посылающим хостом, FreeBSD знает, что нужно использовать loopback-интерфейс (lo0) вместо того, чтобы осуществлять посылку в интерфейс Ethernet.

Две строки host2 являются примером того, что происходит при использовании алиасов в команде ifconfig (обратитесь к разделу об Ethernet для объяснения того, почему мы это делаем). Символ => после интерфейса lo0 указывает на то, что мы используем не просто интерфейс loopback (так как это адрес, обозначающий локальный хост), но к тому же это алиас. Такие маршруты появляются только на хосте, поддерживающем алиасы; для всех остальных хостов в локальной сети для таких маршрутов будут показаны просто строчки link#1.

Последняя строчка (подсеть назначения 224) имеет отношение к многоадресной посылке, которая будет рассмотрена в другом разделе.

наконец, различные атрибуты каждого маршрута перечисляются в колонке Flags. Ниже приводится краткая таблица некоторых из этих флагов и их значений:

| | |
|---|---|
| U | Up: Маршрут актуален. |
| H | Host: Адресом назначения является отдельный хост. |
| G | Gateway: Посылать все для этого адреса назначения на указанную удаленную систему, которая будет сама определять дальнейший путь прохождения информации. |
| S | Static: Маршрут был настроен вручную, а не автоматически сгенерирован системой. |
| C | Clone: Новый маршрут сгенерирован на основе указанного для машин, к которым мы подключены. Такой тип маршрута обычно используется для локальных сетей. |
| W | WasCloned: Указывает на то, что маршрут был автоматически сконфигурирован на основе маршрута в локальной сети (Clone). |
| L | Link: Маршрут включает ссылку на аппаратный адрес Ethernet. |

**Маршруты по умолчанию**

Когда локальной системе нужно установить соединение с удаленным хостом, она обращается к таблице маршрутов для того, чтобы определить, существует ли такой маршрут. Если удаленный хост попадает в подсеть, для которой известен способ ее достижения (маршруты типа Cloned), то система определяет возможность подключиться к ней по этому интерфейсу.

Если все известные маршруты не подходят, у системы имеется последняя возможность: маршрут "default". Это маршрут с особым типом сетевого шлюза (обычно единственным, присутствующим в системе), и в поле флагов он всегда помечен как с. Для хостов в локальной сети этот сетевой шлюз указывает на машину, имеющую прямое подключение внешнему миру (неважно, используется ли связь по протоколу PPP, канал DSL, кабельный модем, T1 или какой-то другой сетевой интерфейс).

Если вы настраиваете маршрут по умолчанию на машине, которая сама является сетевым шлюзом во внешний мир, то маршрутом по умолчанию будет являться сетевой шлюз у Вашего провайдера Интернет (ISP).

Давайте взглянем на примеры маршрутов по умолчанию. Типичная конфигурация представлена на рисунке 1.

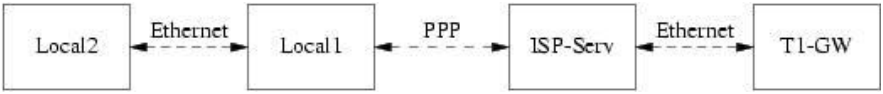


**Рис.1.** Типичная конфигурация маршрутов по умолчанию

Хосты Local1 и Local2 находятся в нашей сети. Local1 подключён к ISP через коммутируемое соединение по протоколу PPP. Этот компьютер с сервером PPP подключён посредством локальной сети к другому шлюзовому компьютеру через внешний интерфейс самого ISP к Интернет.

Маршруты по умолчанию для каждой из ваших машин будут следующими:

```
Хост      Маршрут по умолчанию     Интерфейс
Local2    Local1                   Ethernet
Local1    T1-GW                    PPP
```

Часто задаётся вопрос "Почему (или каким образом) в качестве шлюза по умолчанию для машины Local1 мы указываем T1-GW, а не сервер провайдера, к которому подключаемся?".

Запомните, что из-за использования PPP-интерфейсом адреса в сети провайдера Интернет вашей стороны соединения, маршруты для всех других машин в локальной сети провайдера будут сгенерированы автоматически. Таким образом, вы уже будете знать, как достичь машины T1-GW, так что нет нужды в промежуточной точке при посылке трафика к серверу ISP.

В локальных сетях адрес X.X.X.1 часто используется в качестве адреса сетевого шлюза. Тогда (при использовании того же самого примера) если пространство адресов класса C вашей локальной сети было задано как 10.20.30, а ваш провайдер использует 10.9.9, то маршруты по умолчанию будут такие:

```
Хост                              Маршрут по умолчанию
Local2 (10.20.30.2)               Local1 (10.20.30.1)
Local1(10.20.30.1, 10.9.9.30)     T1-GW (10.9.9.1)
```

Вы можете легко задать используемый по умолчанию маршрутизатор посредством файла /etc/rc.conf. В нашем примере на машине Local2 мы добавили такую строку в файл /etc/rc.conf:

```
defaultrouter="10.20.30.1"
```

Это также возможно сделать и непосредственно из командной строки при помощи команды route:

```
# route add default 10.20.30.1
```

Для получения дополнительной информации об управлении таблицами маршрутизации обратитесь к справочной странице по команде route.

**Хосты с двойным подключением**

Есть еще один тип подключения, который мы должны рассмотреть, и это случай, когда хост находится в двух различных сетях. Технически, любая машина, работающая как сетевой шлюз (в примере выше использовалось PPP-соединение), считается хостом с двойным подключением. Однако этот термин реально используется для описания машины, находящейся в двух локальных сетях.

В одном случае у машины имеется два адаптера Ethernet, каждый имеющий адрес в разделенных подсетях. Как альтернативу можно рассмотреть вариант с одним Ethernet-адаптером и использованием алиасов в команде ifconfig. В первом случае используются два физически разделённые сети Ethernet, в последнем имеется один физический сегмент сети, но две логически разделённые подсети.

В любом случае таблицы маршрутизации настраиваются так, что для каждой подсети эта машина определена как шлюз (входной маршрут) в другую подсеть. Такая конфигурация, при которой машина выступает в роли маршрутизатора между двумя подсетями, часто используется, если нужно реализовать систему безопасности на основе фильтрации пакетов или функций брандмауэра в одном или обоих направлениях.

Если вы хотите, чтобы эта машина действительно перемещала пакеты между двумя интерфейсами, то вам нужно указать FreeBSD на включение этой функции. Обратитесь к следующей главе, чтобы узнать, как это сделать.

**Построение маршрутизатора**

Сетевой маршрутизатор является обычной системой, которая пересылает пакеты с одного интерфейса на другой. Стандарты Интернет и хорошая инженерная практика не позволяют Проекту FreeBSD включать эту функцию по умолчанию во FreeBSD. Вы можете включить эту возможность, изменив значение следующей переменной в YES в файле rc.conf:

```
gateway_enable=YES          # Set to YES if this host will be a gateway
```

Этот параметр изменит значение sysctl-переменной net.inet.ip.forwarding в 1. Если вам временно нужно выключить маршрутизацию, вы можете на время сбросить это значение 0.

Вашему новому маршрутизатору нужна информация о маршрутах для того, чтобы знать, куда пересылать трафик. Если ваша сеть достаточно проста, то вы можете использовать статические маршруты. С FreeBSD также поставляется стандартный даемон BSD для маршрутизации routed, который умеет работать с RIP (как версии 1, так и версии 2) и IRDP. Поддержка BGP v4, OSPF v2 и других сложных протоколов маршрутизации имеется в пакете net/zebra. Также существуют и коммерческие продукты, применяемые как более комплексное решение проблемы маршрутизации в сети, такие как GateD

**Ручная настройка статических маршрутов**

Предположим, что у нас есть следующая сеть:

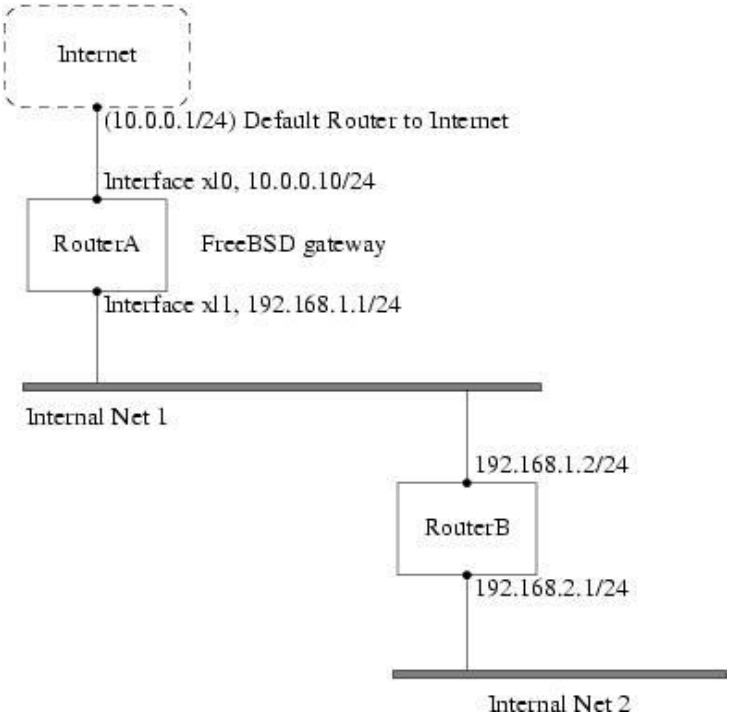


**Рис.2.** Пример сети

В этом сценарии, RouterA это наш компьютер с FreeBSD, который выступает в качестве маршрутизатора в сеть Интернет. Его маршрут по умолчанию настроен на 10.0.0.1, что позволяет ему соединяться с внешним миром. Мы будем предполагать, что RouterB уже правильно настроен и знает все необходимые маршруты (на этом рисунке все просто; добавьте на RouterB маршрут по умолчанию, используя 192.168.1.1 в качестве шлюза).

Если мы посмотрим на таблицу маршрутизации RouterA, то увидим примерно следующее:

```
#netstat -nr
Routing tables

Internet:
Destination  Gateway    Flags  Refs Use     Netif
default      10.0.0.1   UGS    0    49378   xl0
127.0.0.1    127.0.0.1  UH     0    6       lo0
10.0.0/24    link#1     UC     0    0       xl0
192.168.1/24 link#2     UC     0    0       xl1
```

С текущей таблицей маршрутизации RouterA не сможет достичь внутренней сети 2 (Internal Net 2). Один из способов обхода этой проблемы — добавление маршрута вручную. Следующая команда добавляет внутреннюю сеть 2 к таблице маршрутизации RouterA с 192.168.1.2 в качестве следующего узла:

```
# route add -net 192.168.2.0/24 192.168.1.2
```

Теперь RouterA сможет достичь любого хоста в сети 192.168.2.0/24.

**Постоянная конфигурация**

Проблема предыдущего примера заключается в том, что маршрутная информация не сохранится после перезагрузки FreeBSD. Способ сохранения добавленного маршрута заключается в добавлении его в файл /etc/rc.conf:

```
#Добавление статического маршрута в Internal Net 2
static_routes="internalnet2"
route_internalnet2="-net 192.168.2.0/24 192.168.1.2"
```

В переменной static_routes находятся строки, разделенные пробелами. Каждая строка означает имя маршрута. В примере выше в static_routes есть только одна строка, это internalnet2. Затем мы добавили переменную route_internalnet2, куда помещены все

параметры, которые необходимо передать команде route. В примере выше была использована команда:

```
# route add -net 192.168.2.0/24 192.168.1.2
```

поэтому нам потребуется "-net 192.168.2.0/24 192.168.1.2".

Как было сказано выше, мы можем добавить в static_routes более чем одну строку. Это позволит создать несколько статических маршрутов. В следующем примере показано добавление маршрутов для сетей 192.168.0.0/24 и 192.168.1.0/24 (этот маршрутизатор не показан на рисунке выше):

```
static_routes="net1 net2"
route_net1="-net 192.168.0.0/24 192.168.0.1"
route_net2="-net 192.168.1.0/24 192.168.1.1"
```

**Распространение маршрутов**

Мы уже знаем, что таблицы маршрутизации могут быть настроены так, что весь трафик для некоторого диапазона адресов (в нашем примере это подсеть класса C) может быть направлен заданному хосту в той сети, которая будет перенаправлять входящие пакеты дальше.

При получении адресного пространства, выделенного Вашей сети, Ваш провайдер настроит свои таблицы маршрутизации так, что весь трафик для Вашей подсети будет пересылаться по PPP-соединению к Вашей сети. Но как серверы по всей стране узнают, что Ваш трафик нужно посылать Вашему ISP?

Существует система (подобная распределению информации DNS), которая отслеживает все назначенные пространства адресов и определяет точку подключения к магистрали Интернет. "Магистралью" называют главные каналы, по которым идет трафик Интернет внутри страны и по всему миру. Каждая магистральная машина имеет копию основного набора таблиц, согласно которой трафик для конкретной

сети направляется по конкретному магистральному каналу, и затем, передаваясь по цепочке провайдеров, он достигает вашей сети.

Задачей вашего провайдера является объявить на магистрали о том, что он отвечает за подключение вашей сети. Этот процесс называется распространением маршрута.

**Устранение неполадок**

Иногда с распространением маршрута возникают проблемы, и некоторые сайты не могут вам подключиться. Для определения точки неверной работы маршрутизации можно воспользоваться командой traceroute. Она также полезна и когда вы сами не можете подключиться к удаленной машине (то есть команда ping не срабатывает).

Команда traceroute запускается с именем удаленного хоста, с которым вы хотите установить соединение, в качестве параметра. Она показывает промежуточные сетевые шлюзы по пути следования, в конце концов достигая адрес назначения или прерывая свою работу из-за отсутствия соединения.

За дополнительной информацией обратитесь к странице Справочника по traceroute.

**Маршрутизация многоадресного трафика**

FreeBSD изначально поддерживает как приложения, работающие с многоадресным трафиком, так и его маршрутизацию. Такие приложения не требуют особой настройки FreeBSD; обычно они работают сразу. Для маршрутизации многоадресного трафика требуется, чтобы поддержка этого была включена в ядро:

```
options MROUTING
```

Кроме того, даемон многоадресной маршрутизации, mrouted, должен быть настроен посредством файла */etc/mrouted.conf* на использование туннелей и DVMRP. Дополнительную информацию о настройки многоадресного трафика можно найти на страницах справочной системы, посвящённых даемону mrouted.

# ПРОТОКОЛЫ МАРШРУТИЗАЦИИ

## Протоколы внутренней маршрутизации

Протоколы маршрутизации делятся на две базовых категории: протоколы внутренней маршрутизации и протоколы внешней маршрутизации. Протокол внутренней маршрутизации используется в рамках независимой сетевой системы. В терминологии TCP/IP такие независимые сетевые системы называются автономными системами. В пределах автономной системы информация маршрутизации циркулирует на основе протокола маршрутизации, выбранного при администрировании этой автономной системы.

Все протоколы внутренней маршрутизации выполняют одни и те же и основные функции: определяют "лучший" маршрут в каждый пункт назначения и распространяют информацию маршрутизации среди систем сети. То, как они выполняют эти функции (в частности, как определяют лучшие машруты), - критерий, по которому различаются протоколы маршрутизации. Протоколов внутренней маршрутизации существует несколько:

- *Протокол маршрутной информации* (RIP, *Routing Information Protocol*) протокол внутренней маршрутизации, наиболее широко распространенный на платформах Unix. Реализации RIP поставляются в составе большинства систем Unix. Протокол адекватен в локальных сетях (LAN) и прост в настройке. RIP считает лучшим маршрут с минимальным числом транзитных участков (*метрикой маршрутизации*). Число транзитных участков в случае RIP - это число шлюзов, через которые должны пройти данные, прежде чем достигнут пункта назначения. RIP предполагает, что лучший маршрут проходит через минимальное число шлюзов. Такой подход к выбору маршрута носит название *алгоритма вектора расстояния* (*distance-vector algorithm*).
- *Hello* - протокол, в котором выбор лучшего маршрута выполняется на основе анализа задержек. Задержка- это время, за которое дейтаграмма проходит от источника к адресату и обратно. Пакет

Hello содержит отметку времени отправки. Когда пакет доходит до адресата, получившая его система вычисляет время путешествия пакета. Hello используется достаточно редко. В свое время он использовался для внутренней маршрутизации исходной магистрали NSFNET (56 Кбит) и, пожалуй, больше практически нигде.

- *Протокол общения промежуточных систем* IS-IS (*Intermediate System to Intermediate System*) - протокол внутренней маршрутизации из набора протоколов OSI. Протокол IS-IS работает на основе алгоритма состояния канала и является *протоколом кратчайшего пути* (Shortest Path First, SPF). Данный протокол использовался для внутренней маршрутизации магистрали NSFNET T1 и сегодня все еще применяется некоторыми из крупных поставщиков услуг.
- *Протокол предпочтения кратчайшего пути* OSPF (*Open Shortest Path First*) - другой протокол состояния канала, разработанный для TCP/IP. Он подходит для применения в очень крупных сетях и имеет ряд преимуществ перед RIP.

Из перечисленных протоколов в подробностях мы рассмотрим RIP и OSPF. OSPF широко применяется на маршрутизаторах, а RIP - в системах Unix. Мы начнем с протокола RIP.

**Протокол маршрутной информации (RIP)**

В поставку многих систем Unix протокол Routing Information Protocol входит в качестве демона маршрутизации routed (произносится «рут-ди»). При запуске routed генерирует запрос на обновление маршрутов и ожидает получения ответов на этот запрос. Получив запрос, система, настроенная на распространение информации RIP, отвечает пакетом обновлений, созданным на основе информации из локальной таблицы маршрутизации. Пакет обновлений содержит конечные адреса из таблицы маршрутизации и связанные с ними метрики маршрутизации. Пакеты обновлений генерируются в ответ на

запросы, а также в целях периодического уточнения информации маршрутизации.

Для создания таблицы маршрутизации routed использует информацию из пакетов обновлений. Если обновление содержит маршрут к пункту назначения, не существующий в локальной таблице маршрутизации, происходит добавление нового маршрута. Если обновление описывает маршрут, конечный пункт которого уже есть в локальной таблице, новый маршрут будет использован только в случае, если является лучшим из двух. Как уже говорилось, RIP считает лучшим маршрут с меньшим числом транзитных участков, а само это число в терминологии RIP называется *стоимостью маршрута*, или *метрикой маршрутизации*. Ранее мы видели, что метрика маршрутизации в локальной таблице поддается ручной корректировке посредством аргумента metric команды route. Чтобы выбрать лучший маршрут, протокол RIP должен сначала определить стоимость маршрута. Стоимость маршрута определяется суммой стоимости маршрута до шлюза, сгенерировавшего обновление, и метрики, содержащейся в пакете обновлений RIP. Если совокупная стоимость меньше стоимости уже существующего в таблице маршрута, используется новый маршрут.

Кроме того, RIP удаляет маршруты из таблицы маршрутизации. Во-первых, маршрут удаляется, если шлюз, через который пролегает маршрут, утверждает, что стоимость маршрута превышает 15. Во-вторых, RIP считает шлюз, не присылающий обновления, неработоспособным. Удаляются все маршруты, пролегающие через этот шлюз, если за определенный период времени не было получено ни одного обновления. Как правило, RIP генерирует обновления раз в 30 секунд. Во многих реализациях молчание шлюза в течение 180 секунд является поводом для удаления всех пролегающих через этот шлюз маршрутов из локальной таблицы маршрутизации.

**RIP и routed**

Чтобы запустить службу RIP при помощи демона маршрутизации (routed) наберите такую команду:

```
# routed
```

Команда routed часто фигурирует без аргументов командной строки, ключ -q может вам пригодиться. Ключ -q запрещает routed распространять маршруты и разрешает демону только принимать маршруты, распространяемые другими системами. Если машина не является шлюзом, имеет смысл использовать ключ -q.

В разделе, посвященном статической маршрутизации, мы не стали блокировать команду routed в файле inetinit, поскольку Solaris запускает routed только в случае, когда в системе установлено более одного сетевого интерфейса либо когда существует файл /etc/gateways. Если система Unix запускает routed в любом случае, для запуска службы RIP не требуется дополнительных действий - достаточно просто загрузить систему. Иначе необходимо убедиться, что команда routed присутствует в одном из загрузочных файлов и что выполнены все условия ее запуска. Простейший способ запустить routed в системе Solaris - создать файл gateways, пусть даже пустой.

routed читает файл /etc/gateways при запуске и добавляет информацию из файла в таблицу маршрутизации. routed может создать работоспособную таблицу маршрутизации на основе только обновлений RIP, полученных от прочих RIP-систем. Но иногда бывает полезно дополнить эту информацию, скажем, начальным маршрутом по умолчанию либо сведениями о шлюзе, который не распространяет данные о своих маршрутах. Такие дополнительные сведения хранятся в файле /etc/gateways.

Чаще всего файл /etc/gateways содержит определение активного маршрута по умолчанию, и это обстоятельство мы используем в наших примерах. Одного этого примера вполне достаточно, поскольку все записи файла /etc/gateways имеют однородный формат. Следующая запись определяет систему crab в качестве шлюза по умолчанию:

net 0.0.0.0 gateway 172.16.12.1 metric 1 active

Запись начинается ключевым словом net. Все записи начинаются ключевым словом net либо ключевым словом host: первое предшествует адресу сети, второе - адресу узла. Конечный адрес 0.0.0.0 - это адрес маршрута по умолчанию. Применяя команду route, для обозначения этого маршрута мы использовали ключевое слово default, но в файле /etc/gateways маршрут по умолчанию обозначается адресом сети 0.0.0.0.

Далее следует ключевое слово gateway и IP-адрес шлюза. В данном случае -адрес узла crab (172.16.12.1).

Затем следуют ключевое слово metric и численное значение метрики маршрутизации. Метрика определяет стоимость маршрута. В статической маршрутизации метрика почти не востребована, но в случае RIP метрики используются для принятия решений. Метрика RIP определяет число шлюзов, через которые должны пройти данные, чтобы попасть в пункт назначения. Но, как мы видели при изучении ifconfig, на деле метрика- это произвольное чение, используемое администратором для расстановки приоритетов маршрутов. (Системный администратор волен назначать маршруту любое чение метрики.) Однако имеет смысл варьировать метрику для нескольких маршрутов, ведущих к одному пункту назначения. У нас есть только один шлюз в сеть Интернет, так что верной метрикой для узла crab будет 1.

Все записи /etc/gateways заканчиваются ключевым словом passive либо active. Первое означает, что от указанного шлюза локальная система не ожидает RIP-обновлений. Используйте ключевое слово passive, чтобы запретить RIP удалять маршруты в случае, когда шлюз не присылает пакеты обновлений и не должен их присылать. Пассивные маршруты добавляются в таблицу маршрутизации и существуют в течение всего времени работы системы. По сути дела, они становятся постоянными статическими маршрутами.

С другой стороны, ключевое слово active создает маршруты, обновляемые протоколом RIP. Ожидается, что активный шлюз предоставляет информацию маршрутизации, которая удаляется из таблицы, если пакеты обновлений не поступают в течение заранее определенного интервала времени. Активные маршруты используются для «стимулирования» на этапе запуска RIP: предполагается, что они будут обновляться после перехода протокола в фазу активного существования.

Приведенная запись завершается ключевым словом active, то есть данный маршрут по умолчанию будет удален в случае отсутствия обновлений от узла crab. Умолчания маршрутов удобны, в особенности для статической маршрутизации. Однако при динамической маршрутизации их следует использовать осторожно, особенно если речь идет о нескольких шлюзах, предоставляющих маршруты одного направления. Пассивный маршрут по умолчанию не позволяет протоколу маршрутизации выполнять динамическое обновление и подстраиваться под изменения условий сетевой среды. Используйте активные маршруты по умолчанию, которые могут обновляться протоколом маршрутизации.

RIP легок в установке и настройке. Идеальный вариант? Не совсем так. Rip имеет три серьезных недостатка:

**Ограниченный диаметр сети**

Максимальная длина маршрута RIP - 15 транзитных участков. Маршрутизатор RIP не способен создать полную таблицу маршрутизации для сети, работающей с более длинными маршрутами. Число транзитных участков не может быть увеличено из-за следующего недостатка.

**Медленная сходимость**

Удаление неверного маршрута иногда требует многократного обмена пакетами обновлений, прежде чем стоимость маршрута достигнет значения 16. Это называется «счетом до бесконечности», поскольку RIP продолжает увеличивать стоимость маршрута, пока она

не превысит максимально допустимого значения метрики в RIP. (В данном случае бесконечность представлена числом 16.) Кроме того, в RIP отсрочка удаления маршрута может достигать 180 секунд. Говоря на сетевом жаргоне, эти условия замедляют «сходимость маршрутизации», то есть требуется значительное время для того, чтобы таблица маршрутизации полностью отразила изменившееся состояние сети.

**Классовая маршрутизация**

RIP интерпретирует все адреса по правилам для классов, приведённым в главе 2. С точки зрения RIP все адреса принадлежат классам A, B и C, что делает протокол RIP несовместимым с современной практикой интерпретации адресов на основе битовых адресных масок.

Обойти ограничение диаметра сети невозможно. Небольшие значений метрик - насущная необходимость, позволяющая сократить воздействие счета до бесконечности. Однако ограниченный размер сети - наименьший из недостатков протокола RIP. Настоящая задача по улучшению RIP связана решением двух других проблем - медленной сходимости и классовой маршрутизации.

RIP были добавлены возможности, позволяющие бороться с медленной сходимостью. Прежде чем перейти к изучению этих возможностей, мы должны понять, как возникает проблема «счета до бесконечности». На рис. 3 отражена сеть, в которой может возникнуть такая проблема.

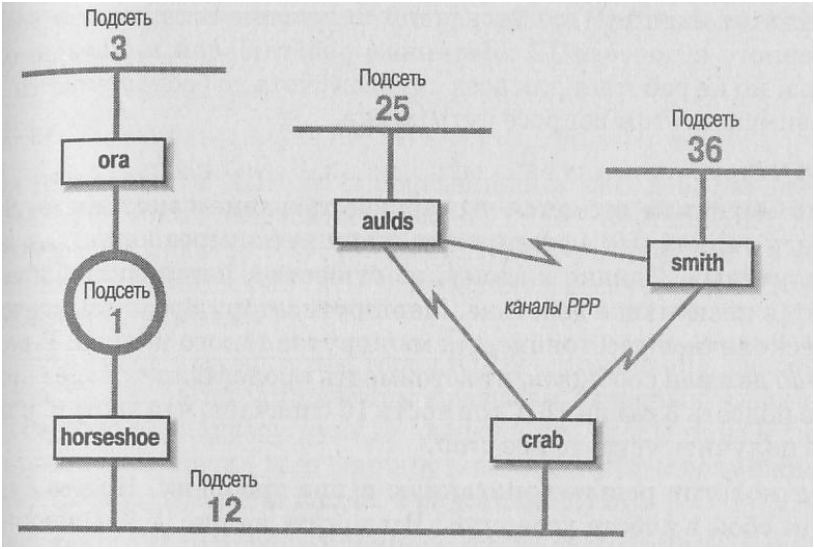


**Рис.3.** Пример сети

Узел crab обращается к подсети 3 через horseshoe и далее через узел ora. Подсеть 3 удалена на два транзитных участка от узла crab и на один транзитный участок от узла horseshoe. Следовательно, horseshoe афиширует стоимость 1 для подсети 3, а crab - стоимость 2, и маршрутизация трафика через horseshoe продолжается. До тех пор, пока не возникнут проблемы. Если неожиданно перестает работать узел ora, horseshoe ожидает обновлений от ora в течение 180 секунд. В процессе ожидания horseshoe продолжает посылать узлу обновления, и маршрут в подсеть 3 сохраняется в таблице маршрутизации crab. Когда интервал ожидания horseshoe наконец истекает, horseshoe удаляет все маршруты, пролегающие через ora, из своей таблицы маршрутизации, включая и маршрут в сеть 3. Затем horseshoe получает от узла crab обновление, уведомляющее, что crab находится в двух транзитных участках от подсети 3. horseshoe создает этот маршрут и объявляет, что находится в трех транзитных участках от подсети 3. crab получает это обновление, создает маршрут и объявляет, что находится в четырех транзитных участках от подсети 3. И так по кругу, пока стоимость

маршрута в подсеть 3 не достигнет 16 в обеих таблицах маршрутизации. Если интервал обновления равен 30 секундам, процесс может затянуться надолго!

Механизмы Split horizon (Расщепленные горизонты) и Poison reverse (Отравленный обратный путь) - вот те две технологии, которые позволяют во многих случаях избежать счета до бесконечности.

**Split horizon**

Данный механизм не позволяет маршрутизатору афишировать маршруты через канал, по которому эти маршруты были получены, и решает описанную выше проблему счета до бесконечности. Следуя этому правилу, crab не станет уведомлять подсеть 12 о маршруте в подсеть 3, поскольку узнал этот маршрут из обновлений, полученных от узла horseshop, расположенного в подсети 12. Механизм работает для приведенного выше примера, но не работает для всех случаев счета до бесконечности. Мы еще остановимся на этом вопросе чуть позже.

**Poison reverse**

Данный механизм является усовершенствованием механизма Split horizon. Идея та же: «Не афишировать маршруты через канал, по которому они получены». Однако к этому, по существу, негативному правилу, добавляется позитивное действие. Маршрутизатору предписывается объявлять бесконечное расстояние для маршрутов такого канала. В результате узел crab должен сообщать, что стоимость пролегающих через него маршрутов в подсеть 3 равна 16. Стоимость 16 означает, что доступ к подсети нельзя получить через шлюз crab.

Эти две технологии решают описанную выше проблему. Но что будет, если произойдет сбой в работе узла crab? Следуя правилу «split horizon», узлы aulds и smith не объявят маршрут в подсеть 12 шлюзу crab, поскольку сами узнали этот маршрут от узла crab. Однако они обмениваются маршрутом в подсеть 12 друг с другом. Если crab перестает работать, aulds и smith начинают свой счет до бесконечности, который заканчиваетеся удалением маршрута в подсеть 12. Эту

проблему призвана решить технология triggered updates (обновления по условию, или мгновенные обновления).

Triggered updates - большой шаг вперед, поскольку обновления посылаются немедленно, а не по истечении стандартного 30-секундного интервала. Таким образом, если происходит сбой маршрутизатора более высокого уровня или локального канала, маршрутизатор передает своим соседям обновления сразу после того, как внесет их в собственную таблицу маршрутизации. Без обновлений по условию счет до бесконечности может занять до восьми минут! Обновления по условию позволяют уведомить соседей за несколько секунд. Кроме того, данный механизм позволяет более эффективно использовать сетевые каналы. Обновления по условию не содержат полных таблиц маршрутизации - лишь сведения об изменившихся маршрутах.

Обновления по условию позволяют предпринимать четкие действия по уничтожению непроходимых маршрутов. Маршрутизатор объявляет маршруты, удаленные из таблицы маршрутизации, с бесконечной стоимостью, что вынуждает прочие маршрутизаторы также удалить эти маршруты. Взгляните еще раз на рис. 7.2. При сбое шлюза crab узлы smith и aulds выжидают 180 секунд, прежде чем удалить маршруты в подсети 1, 3 и 12 из своих таблиц маршрутизации. Затем они обмениваются обновлениями по условию, содержащими метрику 16 для подсетей 1, 3 и 12. Таким образом они сообщают друг другу, что не способны общаться с этими сетями, а необходимость в счете до бесконечности исчезает. Технологии split horizons, poison reverse и triggered updates играют важную роль в уничтожении счета до бесконечности.

Последний недостаток - несовместимость RIP с сетями CIDR и подсетями переменной длины - привел к тому, что в 1996 году протокол RIP получил статус «исторического». RIP несовместим с существующим стеком протоколов, равно как с планами по его развитию. Для решения этой последней проблемы была разработана новая версия RIP.

**RIP Version 2**

Протокол RIP версии 2 (RIP-2), определенный в RFC 2453, является новой версией RIP. Протокол разрабатывался не с нуля - он лишь определяет расширения формата пакетов RIP. К адресу пункта назначения и метрике, существовавшим в пакетах RIP, RIP-2 добавляет маску сети и адрес следующего транзитного участка.

Маска сети снимает с маршрутизаторов RIP-2 ограничение, связанное с интерпретацией адресов по устаревшим правилам адресных классов. Теперь маска применяется к адресу пункта назначения, чтобы определить способ его интерпретации. Маска дает маршрутизаторам RIP-2 возможность работать с подсетями переменной длины и над сетями CIDR.

Адрес следующего транзитного участка - это IP-адрес шлюза, через который проходит маршрут. Если это адрес 0.0.0.0, источник пакета обновлений является шлюзом для маршрута. Транзитный адрес позволяет источнику данных RIP-2 распространять информацию маршрутизации о шлюзах, которые не говорят на языке протокола RIP-2. Функциональность транзитных адресов схожа с функциональностью сообщений ICMP Redirect, они указывают на лучшие шлюзы для маршрутов и сокращают число транзитных участков.

RIP-2 содержит и другие нововведения. Протокол передает обновления на групповой адрес 224.0.0.9, чтобы сократить нагрузку на системы, не способные обрабатывать пакеты RIP-2. Кроме того, RIP-2 предоставляет механизм проверки подлинности пакетов, позволяющий сократить возможность приема ошибочных обновлений от некорректно настроенных систем.

Несмотря на все изменения протокол RIP-2 совместим с RIP. Исходные спецификации RIP закладывали возможности такого развития протокола. В заголовке пакета RIP присутствует номер версии и несколько пустых полей для потенциальных расширений. Новые значения RIP-2 не потребовали переработки структуры пакета; они передаются в пустых полях, которые в исходном протоколе были зарезервированы для использования в будущем. Корректные

реализации маршрутизаторов RIP способны принимать пакеты RIP-2 и извлекать из пакетов данные, не обращая внимания на новую информацию.

Split horizon, poison reverse, triggered updates, а также протокол RIP-2 решают большинство проблем изначального протокола RIP. Однако RIP-2 остается протоколом вектора расстояния. Существуют современные технологии маршрутизации, которые считаются более приемлемыми для крупных сетей. В частности- протоколы маршрутизации, выполняющие *анализ состояния каналов*, поскольку они обеспечивают быструю сходимость и сокращают риск возникновения петель маршрутизации.

**Протокол предпочтения кратчайшего пути**

*Протокол предпочтения кратчайшего пути* OSPF (*Open Shortest Path First*), определенный документом RFC 2328, является протоколом состояния канала и в корне отличается от протокола RIP. Маршрутизатор, использующий RIP, делится информацией обо всей сети со своими соседями. Напротив, маршрутизатор, использующий OSPF, делится информацией о своих соседях со всей сетью. «Вся сеть» означает максимум одну автономную систему. RIP не пытается получить полные сведения о сети Интернет, а OSPF не пытается распространить информацию по всей сети Интернет. Задача этих протоколов в другом. Протоколы внутренней маршрутизации призваны решать вопросы маршрутизации в рамках отдельных автономных систем. OSPF подходит к задаче более скрупулезно, определяя иерархию областей маршрутизации автономной системы:

*Область (Area)* - это произвольный набор взаимосвязанных сетей, узлов и маршрутизаторов. Обмен информацией маршрутизации между областями одной автономной системы происходит посредством пограничных маршрутизаторов областей.

*Магистраль (Backbone)* - это особая область, объединяющая все прочие области автономной системы. Каждая область должна быть

связана с магистралью, поскольку магистраль отвечает за распространение информации маршрутизации между областями.

*Оконечная область (Stub area)* имеет лишь один пограничный маршрутизатор, то есть из области существует единственный маршрут. В данном случае пограничный маршрутизатор области может не сообщать о внешних маршрутах прочим маршрутизаторам оконечной области. Достаточно заявить о себе как о точке, через которую пролегает маршрут по умолчанию.

Деление на области необходимо лишь для крупных автономных систем. Сеть, показанная на рис. 7.2, невелика, нет смысла делить ее на области. Тем не менее она может послужить иллюстрацией различных областей. Мы можем разделить данную автономную систему на любые области - как нам заблагорассудится. Предположим, мы разделили ее на три области: область содержит подсеть 3; область 2 содержит подсети 1 и 12; область 3 содержит подсети 25 и 36, а также каналы PPP. Далее, мы можем определить область в качестве оконечной, поскольку она имеет лишь один пограничный маршрутизатор - ora. Кроме того, мы можем определить область 2 в качестве магистральной, поскольку она объединяет две оставшиеся области и передает всю информацию маршрутизации между областями 1 и 3. Область 2 содержит два пограничных маршрутизатора, crab и ora, плюс один внутренний маршрутизатор - horseshoe. Область 3 содержит три маршрутизатора: crab, smith и aulds.

Очевидно, OSPF обеспечивает высокую гибкость в плане разграничения автономной системы. Для чего же нужна такая гибкость? Одной из проблем протоколов, анализирующих состояние каналов, является большой объем данных, накапливаемых в базе данных состояний каналов, и объем времени, необходимый для вычисления маршрутов на основе этих данных. Сейчас станет ясно, почему возникает такая проблема.

Каждый OSPF-маршрутизатор выполняет построение ориентированного графа всей сети при помощи алгоритма Дейкстры, служащего для обнаружения кратчайшего пути (Shortest Path First,

SPF). Ориентированный граф - это карта сети с точки зрения маршрутизатора. То есть корнем графа является маршрутизатор. Построение графа выполняется на основе данных из базы данных состояния каналов, содержащей информацию о каждом маршрутизаторе сети и обо всех соседях каждого маршрутизатора. Эта база данных для автономной системы, представленной на рис. 3, содержит пять маршрутизаторов и десять соседей: у ora один сосед, horseshoe; у horseshoe два соседа, ora и crab; у crab три соседа - horseshoe, aulds и smith; у aulds -два соседа, crab и smith; у smith - два соседа, aulds и crab. Граф этой автономной системы в представлении маршрутизатора ora отражен на рис. 4.

Алгоритм Дейкстры создает карту следующим образом:

1. Локальная система устанавливается в качестве корня карты и получает нулевую стоимость.
2. В карту добавляются соседи только что установленной системы. Стоимость сообщения с соседями представлена суммой стоимости сообщения с только что установленной системой и стоимостью, которую эта система афиширует для каждого из соседей. Например, предположим, что crab афиширует стоимость 20 для aulds, а стоимость сообщения с crab - 15. Тогда стоимость aulds на карте ora - 35.
3. Выполняется обход карты с выбором самых дешевых маршрутов для всех направлений. Например, когда в карту добавляется aulds, среди его соседей находится smith. Путь к узлу smith через aulds временно добавляется в карту. На третьем шаге алгоритма стоимость сообщения с узлом smith через crab сравнивается со стоимостью сообщения с узлом smith через aulds. Выбор делается в пользу более дешевого пути. На рис. 7.3 отброшенные маршруты представлены пунктирными линиями. Шаги 2 и 3 алгоритма повторяются для каждой системы из базы данных.

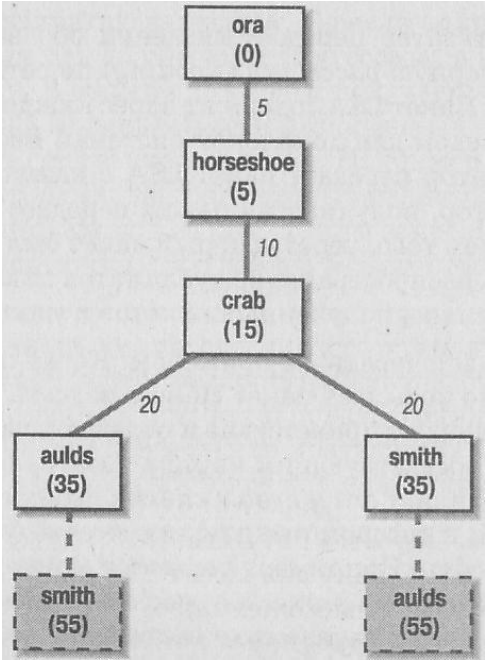


**Рис.4.** Граф сети

Информация базы данных состояния каналов накапливается и распространяется по простым и эффективным правилам. OSPF-маршрутизатор выполняет обнаружение своих соседей при помощи пакетов Hello. Он отправляет пакеты Hello и ожидает получения пакетов Hello от соседствующих маршрутизаторов. Пакет Hello идентифицирует локальный маршрутизатор и перечисляет соседние маршрутизаторы, от которых были получены пакеты. Получив пакет Hello с информацией о себе в качестве соседствующего, маршрутизатор понимает, что обнаружил соседа. И это вполне логично - ведь он может получать пакеты от этого соседа, а сосед считает его своим соседом - и, значит, может получать ответные пакеты. Обнаруженные соседи добавляются в соответствующий локальный список системы.

Затем OSPF-маршрутизатор передает сведения обо всех своих соседях, а именно выполняет веерную рассылку (flooding) по сети пакетов LSA (Link-State Advertisement). Пакет LSA содержит адрес каждого соседа и стоимость сообщения с этим соседом для локальной системы. Веерная рассылка означает, что маршрутизатор передает пакет LSA с каждого интерфейса и что каждый маршрутизатор, получивший пакет, передает его с каждого интерфейса - за исключением того, через который пакет был изначально получен. Чтобы предотвратить распространение дубликатов пакетов LSA, маршрутизаторы хранят экземпляры полученных пакетов и удаляют дубликаты.

Когда протокол OSPF запускается на horseshoe, то посылает пакет Hello в подсеть 1 и еще один - в подсеть 12. ora и crab получают приветствие и отвечают пакетами Hello, в которых horseshoe указан соседствующим маршрутизатором, horseshoe, получив эти пакеты Hello, добавляет ora и crab в список своих соседей. Затем horseshoe создает пакет LSA, в котором каждому из соседей (ora и crab) поставлена в соответствие стоимость. Например, horseshoe может присвоить ora стоимость 5, а crab - стоимость 10. horseshoe рассылает пакеты LSA в подсети и 12. ora получает LSA и рассылает его в подсети 3. crab получает LSA и рассылает его через оба своих канала PPP. aulds рассылает LSA по каналу к smith, а smith - по тому же каналу aulds. aulds и smith, получив вторую копию LSA, удаляют ее, поскольку она дублирует уже полученную от узла crab. Таким образом, каждый маршрутизатор всей сети получает LSA-пакеты всех других маршрутизаторов.

OSPF-маршрутизаторы отслеживают состояние своих соседей, принимая пакеты Hello. Пакеты Hello генерируются всеми маршрутизаторами периодически. Если маршрутизатор перестал генерировать пакеты, он, или связанный с ним канал, переходит в разряд неработоспособных. Соседи этого маршрутизатора обновляют свои записи LSA и посылают их в сеть. Новые LSA включаются в базу данных состояний каналов на каждом маршрутизаторе сети, и каждый

маршрутизатор заново выполняет построение карты сети, исходя из новых сведений. Очевидно, ограничение размера сети и, как следствие, количества маршрутизаторов снижает нагрузки, связанные с построением карт. Для одних сетей вся автономная система оказывается достаточно невелика. Другим требуется разделение автономной системы на области.

Еще одной чертой OSPF, благоприятно влияющей на производительность, является возможность определить назначенный маршрутизатор. Назначенный маршрутизатор - это один из маршрутизаторов сети, который считает соседями все остальные маршрутизаторы, тогда как все остальные маршрутизаторы сети считают соседом только его. Назначенный маршрутизатор позволяет сократить размер базы данных состояний каналов и повышает скорость работы алгоритма вычисления кратчайшего пути. Рассмотрим для примера широковещательную сеть с пятью маршрутизаторами. Пять маршрутизаторов - по четыре соседа на каждого - являются источником базы данных с двадцатью записями. Но если один из маршрутизаторов является назначенным, тогда у него четыре соседа, а у каждого из его соседей - всего по одному. В общей сложности получается десять записей базы данных. И хотя в столь маленькой сети назначенный маршрутизатор не нужен, чем крупнее сеть, тем выше экономия. Например, широковещательная сеть с 25 маршрутизаторами, один из которых является назначенным, имеет базу данных состояний каналов из пятидесяти записей, тогда как в отсутствие обозначенного маршрутизатора размер базы данных - шестьсот записей.

OSPF дает маршрутизатору полную картину маршрута из конца в конец - сравните с ограниченным видом следующего транзитного участка в RIP. Рассылка LSA позволяет быстро распространить информацию по сети. Ограничение размера базы данных при помощи разделения на области и отметки маршрутизаторов ускоряет вычисление кратчайших путей. В целом, OSPF называется весьма эффективным протоколом маршрутизации.

OSPF предоставляет также функции, которых нет в RIP. Простая аутентификация по паролю, состоящему из восьми символов и передаваемому открытым текстом, позволяет убедиться, что обновление исходит от доверенного маршрутизатора. Кроме того, реализован более надежный механизм аутентификации на основе контрольных сумм MD5 (Message Digest 5).

OSPF поддерживает многолучевую маршрутизацию для лучей равной стоимости (equal-cost multi-path routing). Эта неудобоваримая фраза означает, что маршрутизаторы OSPF способны работать более чем с одним маршрутом определенного направления. При соблюдении определенных условий такая возможность может использоваться для распределения нагрузки по ряду сетевых каналов. Однако многие системы не смогут воспользоваться этим вариантом из-за своих собственных недостатков. Чтобы определить, поддерживает ли ваш маршрутизатор распределение нагрузки посредством равноценных маршрутов OSPF, обратитесь к соответствующей документации.

С учетом описанных возможностей, OSPF является предпочтительным протоколом внутренней маршрутизации TCP/IP для выделенных маршрутизаторов.

**Протоколы внешней маршрутизации**

Протоколы внешней маршрутизации реализуют обмен информацией маршрутизации между автономными системами. Такая информация маршрутизации известна как информация достижимости. Информация достижимости - это сведения о том, какие сети доступны через конкретную автономную систему.

RFC 1771 дает определение протокола пограничных шлюзов BGP (Border Gateway Protocol), ведущего протокола внешней маршрутизации, и содержит следующее описание функции маршрутизации автономной системы:

Классическое определение автономной системы (АС): набор маршрутизаторов, подчиненных единому техническому управлению, использующих протокол внутренних шлюзов и общие метрики для

маршрутизации пакетов в пределах АС, а также протокол внешних шлюзов для маршрутизации пакетов в другие АС... Одна АС видится другой как имеющая четкое внутреннее планирование маршрутизации и представляющая ясную картину достижимых через нее сетей. С точки зрения внешней маршрутизации АС можно считать монолитной конструкцией...

Обмен информацией с такими монолитными конструкциями как раз и является функцией протоколов внешней маршрутизации. Протоколы внешней маршрутизации называют также протоколами внешних шлюзов. Не надо путать понятие протокола внешних шлюзов с Протоколом Внешних Шлюзов EGP (Exterior Gateway Protocol). EGP - не общее название; это имя собственное конкретного протокола внешней маршрутизации, причем довольно старого.

**Протокол внешних шлюзов (EGP)**

Шлюз, работающий с протоколом EGP, сообщает, что способен передавать информацию в сети, которые входят в его автономную систему. Шлюз не говорит, что способен обращаться к сетям за пределами своей автономной системы. Например, внешний шлюз гипотетической автономной системы book-as способен обращаться ко всей сети Интернет через свой внешний канал, но в его автономной системе существует только одна сеть (172.16.0.0), доступ к которой он и будет афишировать, работая с EGP.

Прежде чем передавать информацию маршрутизации, системы обмениваются EGP-сообщениями Hello и I-Heard-You (I-H-U). Эти сообщения начинают диалог между парами шлюзов EGP. Компьютеры, выполняющие обмен данными по EGP, называются EGP-соседями, а обмен сообщениями Hello и I-H-U - знакомством с соседом.

Когда соседи познакомились, информация маршрутизации доступна в результате опроса (poll). Сосед отвечает пакетом информации достижимости, или обновлением (update). Локальная система включает маршруты из обновления в локальную таблицу маршрутизации. Если сосед не ответил на три запроса подряд, система делает вывод, что

сосед недоступен, и удаляет поступившие от него маршруты из таблицы маршрутизации. Получив запрос от EGP-соседа, система отвечает собственным пакетом обновлений.

В отличие от протоколов внутренней маршрутизации, описанных выше, EGP не пытается выбрать «лучший» маршрут. Обновления EGP содержат данные векторных расстояний, но EGP не производит вычислений на основе этой информации. Метрики маршрутизации из различных автономных систем не поддаются прямому сравнению, поскольку в различных АС могут использоваться различные критерии вывода таких значений. Таким образом, EGP оставляет выбор «лучшего» маршрута кому-то другому.

Когда проектировался протокол EGP, функционирование сети зависело от группы доверенных стержневых шлюзов, которые обрабатывали и распространяли маршруты, полученные от всех автономных систем. Предполагалось, что стержневые шлюзы обладают всей необходимой для выбора лучших внешних маршрутов информацией. Информация достижимости EGP передавалась стержневым шлюзам, обрабатывалась и возвращалась автономным системам.

Структура маршрутизации, находящаяся в зависимости от одной группы шлюзов с централизованным управлением, не очень хорошо масштабируется, а потому является неадекватным решением, учитывая темпы роста сети Интернет. По мере роста числа автономных систем и сетей, подключенных к Интернету, центральным шлюзам становилось все труднее справляться с рабочей нагрузкой. В том числе это обстоятельство послужило причиной перехода сети Интернет на более эффективную распределенную архитектуру, возлагающую нагрузку обработки маршрутов на отдельные автономные системы. Вторая причина - отсутствие выделенного управляющего органа коммерциализированной сети Интернет. Интернет состоит из многих равноправных сетей. В распределенной архитектуре автономной системе требуются протоколы маршрутизации, как внутренней, так и

внешней, позволяющие принимать осмысленные решения в выборе маршрутов. По этой причине протокол EGP вышел из моды.

**Протокол пограничных шлюзов (BGP)**

Протокол пограничных шлюзов BGP (Border Gateway Protocol) является ведущим протоколом внешней маршрутизации сети Интернет. Протокол BGP основан на OSI-протоколе междоменной маршрутизации (InterDomain Routing Protocol, IDRP). BGP поддерживает маршрутизацию, подчиненную правилам, позволяющим принимать решения по маршрутизации, исходя из нетехнических причин (политических, структурных, из соображений безопасности и т. д.). Таким образом, BGP совершенствует способность автономной системы выбирать маршруты и приводить в исполнение правила, не обращаясь к высшему авторитету. В отсутствие центральных шлюзов, выполняющих эти задачи, наличие подобных механизмов чрезвычайно важно.

Правила маршрутизации не являются частью протокола BGP. Правила имеют внешние источники и выступают в роли данных настройки. Как уже говорилось в главе 2, в точках доступа к сети (NAPs, Network Access Points), объединяющих крупных поставщиков услуг Интернета, существуют арбитры маршрутизации (RAs, Routing Arbiters). Правила маршрутизации могут быть получены от арбитров. Кроме того, большинство поставщиков услуг Интернета создают частные наборы правил - двусторонние соглашения с другими поставщиками услуг. BGP может применяться для реализации таких соглашений: управления афишируемыми маршрутами и принимаемыми маршрутами. Позже в этой главе - в разделе, посвященном gated, - мы обсудим команды import и export, которые определяют принимаемые (import) и афишируемые (export) маршруты. Администратор сети приводит правила маршрутизации в исполнение путем соответствующей настройки маршрутизатора.

BGP работает поверх TCP, что обеспечивает его надежной службой доставки. BGP использует широко известный порт TCP 179 и

знакомится с соседями посредством стандартного тройного рукопожатия TCP. Соседи BGP известны в качестве равных (peers). Установив соединение, BGP-равные обмениваются сообщениями OPEN в целях согласования параметров сеанса, таких как версия протокола BGP.

Сообщение UPDATE содержит перечень пунктов назначения, доступных через определенное направление, а также свойства этого направления. BGP является протоколом векторного пути и называется так потому, что предоставляет информацию о сквозном пути в виде последовательности номеров автономных систем. Наличие полного пути AC исключает появление петель маршрутизации и проблем счета до бесконечности. Сообщение BGP UPDATE содержит один вектор пути и перечень всех пунктов назначения, доступных через этот вектор. Для создания таблицы маршрутизации могут передаваться множественные пакеты UPDATE.

BGP-равные обмениваются полными обновлениями таблиц маршрутизации в первом сеансе связи. После этого передаются только изменения. Если изменений нет, передается небольшое (19 байт) сообщение KEEPALIVE, уведомляющее, что BGP-система и канал по-прежнему функционируют. BGP весьма экономно расходует ресурсы систем и полосы пропускания сетевых каналов.

Самое важное, что следует помнить о протоколах внешней маршрутизации, - большинство систем прекрасно обходятся без них. Протоколы внешней маршрутизации требуются лишь для переноса информации между автономными системами. Большинство маршрутизаторов в пределах автономной системы работают на основе протокола внутренней маршрутизации, такого как OSPF. И только шлюзам, связующим различные автономные системы, необходимы протоколы внешней маршрутизации. Ваша сеть, скорее всего, входит в качестве независимой составляющей в автономную систему, которая управляется кем-то иным. Хорошим примером автономных систем, состоящих из многочисленных независимых сетей, являются поставщики услуг Интернета. И если ваша организация не

предоставляет услуги подобного уровня, то и протокол внешней маршрутизации, вероятно, не понадобится.

**Выбор протокола маршрутизации**

Несмотря на разнообразие вариантов выбрать протокол маршрутизации обычно легко. Мотивацией разработки большинства описанных протоколов внутренней маршрутизации служила необходимость разрешения конкретных проблем маршрутизации в крупных сетях. Некоторые из протоколов использовались лишь в крупных национальных и региональных сетях. Для территориальных сетей обычным выбором по-прежнему является RIP, тогда как в более крупных сетях выбор делается в пользу OSPF.

В случае необходимости работать с протоколом внешней маршрутизации его тип зачастую не приходится выбирать. Чтобы две автономные системы могли обмениваться информацией маршрутизации, они должны работать с одним протоколом. Если вторая автономная система уже функционирует, ее администраторы, скорее всего, уже решили, какой протокол использовать, и вам останется лишь согласиться с этим выбором. Чаще всего выбор падает на BGP.

На выбор протоколов влияет и тип оборудования. Маршрутизаторы работают с широким спектром протоколов, хотя отдельные производители могут делать акцент на каких-то конкретных. Простые узлы обычно обходятся без протоколов маршрутизации, а в состав большинства систем Unix входит только RIP. Таким образом, допуск простых узлов на поле динамической маршрутизации может серьезно ограничить варианты выбора. Но gated позволяет работать в Unix-системе со многими протоколами маршрутизации. И хотя производительность специального аппаратного обеспечения маршрутизаторов обычно выше, gated позволяет использовать в качестве маршрутизатора любую систему Unix.

**Даемон шлюзовой маршрутизации GateD**

*Замечание*

GateD это закрытое программно обеспечение, более недоступное в исходных текстах (дополнительная информация находится на вебсайте GateD (http://www.gated.org/)). Этот раздел существует лишь в целях обратной совместимости для тех, кто все еще использует старую версию.

Альтернатива головной боли со статическими маршрутами — это установка GateD на FreeBSD SLIP сервере и настройка его для использования соответствующих протоколов маршрутизации (RIP/OSPF/BGP/EGP) для сообщения другим маршрутизаторам о вашей SLIP подсети. Вам потребуется создать /etc/gated.conf для настройки gated. Ниже дан пример:

```
  #gated configuration file for dc.dsu.edu; for gated
  #version 3.5alpha5
  #Only broadcast RIP information for xxx.xxx.yy out the
  #ed Ethernet interface
  #
  #tracing options
  #
  traceoptions "/var/tmp/gated.output" replace size 100k
files 2 general ;

  rip yes {
     interface sl noripout noripin ;
     interface ed ripin ripout version 1 ;
     traceoptions route ;
  } ;

  #
  #Turn on a bunch of tracing info for the interface to
  #the kernel:

  kernel {
    traceoptions remnants request routes info interface;
   } ;

  #
```

```
#Propagate the route to xxx.xxx.yy out the Ethernet
#interface via RIP

export proto rip interface ed {
proto direct {
   xxx.xxx.yy mask 255.255.252.0 metric 1; # SLIP
   connections
} ; } ;

#
# Accept routes from RIP via ed Ethernet interfaces
import proto rip interface ed {
all ;
} ;
```

В примере выше используется широковещательная рассылка информации о маршрутизации для подсети SLIP xxx.xxx.yy протоколом RIP на сеть Ethernet; если вы используете другой драйвер Ethernet вместо ed, потребуется соответственно изменить запись для ed. В этом примере отладочная информация переправляется в /var/tmp/gated.output; вы можете выключить отладку, если GateD работает. Вам потребуется заменить xxx.xxx.yy в сетевом адресе на вашу подсеть SLIP (убедитесь, что изменение сетевой маски в proto direct работает нормально).

Как только вы установили и настроили GateD, потребуется сообщить стартовым скриптам FreeBSD запускать его вместо routed. Простейший способ сделать это — установить переменные router и router_flags в /etc/rc.conf. Обратитесь к странице справочника GateD за информацией о параметрах командной строки.

## ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ

Научиться настраивать сетевые интерфейсы и таблицу маршрутизации в операционной системе FreeBSD. Выполнить следующие шаги:

1. Ознакомиться с предлагаемым материалом для получения базовой информации о настройке сетевых интерфейсов и маршрутизации.
2. Определить тип используемой сетевой карты (PCI или ISA), модель карты и используемый в ней чип.
3. Проанализировать вывод команды ifconfig.
4. Отредактируйте файл /etc/rc.conf.
5. Перезагрузите компьютер.
6. Использую утилиту ping проверьте правильность настройки.
7. Настроить виртуальные серверы.
8. Установить DHCP сервер.
9. Настроить DHCP сервер.
10. Настроить маршрутизацию.
11. Отредактировать файл /etc/gated.conf
12. Запустить даемон GateD.
13. Проверить работоспособность GateD.
14. Изучить основные протоколы маршрутизации.

Ответить на контрольные вопросы и подготовить отчет.

## КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАНИЯ

1. Дайте определение понятию сетевой интерфейс.
2. Раскройте значение термина виртуальный сервер.
3. Предложите варианты утилит, которые позволяют читать и изменять настройки сетевых интерфейсов.
4. Опишите роль «синонима» (alias) сетевого интерфейса.
5. Опишите назначение DHCP.
6. Предложите методы настройки DHCP сервера.
7. Предложите методы настройки DHCP клиента.
8. Дайте определение понятию маршрутизация.
9. Предложите варианты команд для просмотра и управления записями в таблице маршрутизации.
10. Объясните назначение маски сети.
11. Опишите назначение маршрута по умолчанию.
12. Перечислите протоколы маршрутизации
13. Опишите алгоритмы работы протоколов маршрутизации
14. Объясните принцип работы протоколов RIP и OSPF.
15. Объясните в чем отличие протоколов внешней и внутренней маршрутизации.
16. Выполните анализ протоколов «внешней» маршрутизации.

## ФОРМА ОТЧЕТА ПО ЛАБОРАТОРНОЙ РАБОТЕ

На выполнение лабораторной работы отводится 2 занятия (4 академических часа: 3 часа на выполнение и сдачу лабораторной работы и 1 час на подготовку отчета).

Отчет на защиту предоставляется в печатном виде.

Структура отчета (на отдельном листе(-ах)): титульный лист, формулировка задания, ответы на контрольные вопросы, описание процесса выполнения лабораторной работы, выводы.

## ОСНОВНАЯ ЛИТЕРАТУРА

1. Вирт, Н. Разработка операционной системы и компилятора. Проект Оберон [Электронный ресурс] / Н. Вирт, Ю. Гуткнехт ; пер.с англ. Борисов Е.В., Чернышов Л.Н.. — Электрон. дан. — Москва : ДМК Пресс, 2012. — 560 с. — Режим доступа: https://e.lanbook.com/book/39992

## ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА

2. Крищенко, В.А. Сервисы Windows [Электронный ресурс] : учебное пособие / В.А. Крищенко, Н.Ю. Рязанова. — Электрон. дан. — Москва : МГТУ им. Н.Э. Баумана, 2011. — 47 с. — Режим доступа: https://e.lanbook.com/book/52416..

3. Войтов, Н.М. Администрирование ОС Red Hat Enterprise Linux. Учебный курс [Электронный ресурс] : учебное пособие / Н.М. Войтов. — Электрон. дан. — Москва : ДМК Пресс, 2011. — 192 с. — Режим доступа: https://e.lanbook.com/book/1081

4. Стащук, П.В. Администрирование и безопасность рабочих станций под управлением Mandriva Linux: лабораторный практикум [Электронный ресурс] : учебно-методическое пособие / П.В. Стащук. — Электрон. дан. — Москва : ФЛИНТА, 2015. — 182 с. — Режим доступа: https://e.lanbook.com/book/70397

**Электронные ресурсы:**

5. Научная электронная библиотека http://eLIBRARY.RU
6. Электронно-библиотечная система http://e.lanbook.com
7. Losst - Linux Open Source Software Technologies https://losst.ru