Для системных администраторов UNIX

3-е издание

TCP/IP

Сетевое администрирование

Символ®

O'REILLY®

Крэйг Хант

TCP/IP

Network Administration

Third Edition

Craig Hunt

O'REILLY®

TCP/IP

Сетевое администрирование

Третье издание

Крэйг Хант

Санкт-Петербург — Москва
2004

Крэйг Хант

TCP/IP. Сетевое администрирование, 3-е издание

Перевод М. Зислиса

Главный редактор	*А. Галунов*
Зав. редакцией	*Н. Макарова*
Научный редактор	*С. Маккавеев*
Редакторы	*А. Лосев, А. Петухов*
Корректор	*С. Беляева*
Верстка	*Н. Гриценко*

Крэйг Хант
TCP/IP. Сетевое администрирование, 3-е издание. – Пер. с англ. – СПб: Символ-Плюс, 2004. – 816 с., ил.
ISBN 5-93286-056-1

Третье издание книги «TCP/IP. Сетевое администрирование» – это полноценное руководство по настройке и сопровождению сети TCP/IP, которое предназначается как системным администраторам, так и пользователям домашних компьютеров с доступом к сети Интернет. Повествование начинается с основ: зачем нужны протоколы, как они работают, как адресация и маршрутизация позволяют передавать данные по сети и как настроить сетевое соединение.

Помимо базовой настройки книга рассказывает о современных протоколах маршрутизации (RIPv2, OSPF и BGP) и пакете gated, который реализует работу с ними. Кроме того, книга является руководством по настройке многих важных сетевых служб, в том числе DNS, Apache, sendmail, Samba, PPP и DHCP. Две главы посвящены безопасности и разрешению проблем. Третье издание включает новую главу, посвященную настройке сервера Apache и раздел, в котором обсуждается настройка Samba с целью организации совместного доступа к файлам и принтерам в гетерогенной сети Unix/Windows. Справочные приложения подробно описывают синтаксис таких программ, как gated, pppd, named, dhcpd и sendmail. Книга охватывает реализации TCP/IP для систем Linux, Solaris, BSD и System V.

ISBN 5-93286-056-1
ISBN 0-596-00297-1 (англ)

Издательство «Символ-Плюс». 199034, Санкт-Петербург, 16 линия, 7,
тел. (812) 324-5353, edit@symbol.ru. Лицензия ЛП N 000054 от 25.12.98.
Налоговая льгота – общероссийский классификатор продукции
ОК 005-93, том 2; 953000 – книги и брошюры.
Подписано в печать 08.01.2004. Формат 70x100 1/16 . Печать офсетная.
Объем 51 печ. л. Тираж 2000 экз. Заказ N
Отпечатано с диапозитивов в Академической типографии «Наука» РАН
199034, Санкт-Петербург, 9 линия, 12.

Посвящается Алане. Ты начало новой жизни.

Оглавление

Предисловие

Первое издание книги «TCP/IP. Сетевое администрирование» было написано в 1992 году. За истекшие десять лет многое изменилось, но некоторые вещи остались все теми же. TCP/IP по-прежнему сохраняет свое лидерство среди протоколов связи, объединяющих разнотипные компьютерные системы. Он остается фундаментом для взаимодействия и обмена данными, для глобальных компьютерных сетей. Примечательно, что протоколы IP (Internet Protocol, протокол Интернета, или межсетевой протокол), TCP (Transmission Control Protocol, протокол управления передачей) и UDP (User Datagram Protocol, протокол пользовательских дейтаграмм), составляющие базу TCP/IP, не изменились. Изменились способы применения TCP/IP и управления этими протоколами.

Символичен для этих перемен тот факт, что дома у моей тещи есть подключение к сети TCP/IP, которое позволяет ей обмениваться электронной почтой, изображениями и гипертекстовыми документами с другими людьми своего поколения. Для нее это просто «выход в Интернет», но правда такова, что в ее домашней машине реализован полноценный стек протоколов TCP/IP, работает динамическое получение IP-адреса, а кроме того, используются типы данных, которые десять лет назад попросту не существовали.

В 1991 году протоколы TCP/IP были инструментом для опытных пользователей. Сетевые администраторы заведовали ограниченным числом систем и могли рассчитывать, что пользователи этих систем обладают определенным уровнем специальных знаний. Но это в прошлом. В 2002 году потребность в профессиональных сетевых администраторах выше, чем когда-либо ранее, поскольку контингент пользователей становится все более разношерстным и не столь подготовленным к самостоятельному решению технических проблем. В этой книге содержится информация для тех, кто хочет эффективно решать задачи сетевого администрирования TCP/IP.

«TCP/IP. Сетевое администрирование» стала первым сборником полезной информации для профессиональных сетевых администраторов TCP/IP и по сей день остается лучшей из подобных книг. За первым изданием последовал целый поток книг о TCP/IP и Интернете. Однако очень немногие из них сосредоточены на том, что действительно необходимо знать системному администратору об администрировании TCP/IP. Большинство книг – либо академические тексты, написанные с точки зрения архитектора протокола,

либо инструкции по использованию приложений TCP/IP. В них отсутствует практическая информация о сетях, которая необходима системным администраторам Unix. В настоящей книге сделан упор на TCP/IP и Unix, а также на поиск правильного соотношения между теорией и практикой.

Я горжусь предшествующими изданиями этой книги. Что же касается настоящего издания, я постарался сделать все возможное, чтобы не только сохранить настрой книги, но и улучшить ее. Рассмотрено динамическое назначение адресов при помощи протокола DHCP (Dynamic Host Configuration Protocol, протокол динамической настройки узлов). Материал, посвященный системе доменных имен (DNS), теперь охватывает BIND версии 8 и, в меньшей степени, BIND 9. Настройка электронной почты рассмотрена на примере текущей версии sendmail (8), а примеры, связанные с операционной системой, базируются на текущих версиях Solaris и Linux. Из протоколов маршрутизации описаны RIPv2 (Routing Information Protocol version 2, протокол маршрутной информации версии 2), OSPF (Open Shortest Path First, протокол предпочтения кратчайшего пути) и BGP (Border Gateway Protocol, протокол граничных шлюзов). Кроме того, добавлена глава, посвященная настройке веб-сервера Apache, новый материал по `xinetd`, а также информация о создании брандмауэров на базе `iptables`. Отмечу, что эти дополнительные темы не очень сильно увеличили объем книги.

TCP/IP – это набор протоколов связи, определяющих правила общения различных видов компьютеров между собой. «TCP/IP. Сетевое администрирование» – книга о том, как создать собственную сеть на базе TCP/IP. Эта книга является одновременно руководством, отвечающим на вопросы «как» и «почему» из области сетей TCP/IP, и справочником по отдельным сетевым приложениям.

Для кого эта книга

Эта книга предназначена всем владельцам Unix-машин, подключенных к сети TCP/IP.[1] Очевидно, в эту категорию попадают администраторы сетей и систем, отвечающие за настройку и сопровождение машин сети, но также и пользователи, которые желают узнать, каким образом их компьютеры общаются с другими системами. Провести границу между «системным администратором» и «конечным пользователем» довольно сложно. Человек может считать себя пользователем, но, работая на Unix-машине, ему, скорее всего, приходится заниматься и задачами системного администрирования.

В последние несколько лет, словно грибы после дождя, появляются книги для «чайников» и «идиотов». Эта книга не для людей, которые считают себя «идиотами» в отношении Unix. Кроме того, эта книга едва ли пригодится

[1] Большая часть текста применима не только к Unix-системам. Многие форматы файлов и команды, а также все описания протоколов справедливы для операционных систем Windows 98/NT/2000 и других. Администраторам NT-систем можно порекомендовать книгу «Windows NT TCP/IP Network Administration», O'Reilly.

«гениям» от сетевого администрирования. Однако читатели, не относящиеся к этим крайностям, найдут в книге немало полезной информации.

Предполагается, что читатели хорошо разбираются в работе компьютеров и знакомы с основами администрирования Unix-систем. Если это не так, изучить основы поможет книга Элин Фриш (Jeen Frisch) «Essential System Administration» (Основы системного администрирования), O'Reilly, серия Nutshell Handbook).

Структура книги

Книга состоит из трех логических частей: основные понятия, руководство, справочник. Три первых главы в общих чертах рассказывают о протоколах и службах TCP/IP. Они содержат основные понятия, необходимые для понимания последующих глав. Последующие главы содержат практические инструкции по различным темам. Главы с 4 по 7 посвящены планированию сети и настройке основных программных пакетов, необходимых для ее работы. Главы с 8 по 11 рассказывают о настройке основных сетевых служб. Главы 12 и 13 – о двух насущных вопросах, связанных с обеспечением надежной работы сети: безопасности и разрешении проблем. Завершает книгу ряд приложений-справочников, посвященных важным командам и программам.

Книга состоит из следующих глав:

Глава 1 «Обзор TCP/IP» содержит историю TCP/IP, описание архитектуры протокола, а также объясняет принципы его функционирования.

Глава 2 «Доставка данных» описывает адресацию и передачу данных по сети адресатам.

Глава 3 «Сетевые службы» посвящена отношениям между системами клиент–сервер, а также различным службам, жизненно важным для существования современной сети Интернет.

Глава 4 «Начинаем работу» открывает обсуждение установки и настройки сети, рассказывая о предварительном планировании, которое является необходимым шагом при создании сетей.

Глава 5 «Базовая настройка» посвящена вопросам настройки TCP/IP на уровне ядра Unix и настройки системы для запуска сетевых служб.

Глава 6 «Настройка интерфейса» расскажет о том, как связать сетевой интерфейс и сетевое программное обеспечение. Глава содержит примеры настройки интерфейсов Ethernet и PPP.

Глава 7 «Настройка маршрутизации» описывает, как организовать маршрутизацию, позволяющую машинам сети корректно взаимодействовать с другими сетями. В частности, освещены статические таблицы маршрутизации, распространенные протоколы маршрутизации, а также `gated` – пакет, реализующий последние версии некоторых из протоколов маршрутизации.

Глава 8 «Настройка DNS» посвящена администрированию программы сервера имен, который преобразует имена машин сети в адреса Интернета.

Глава 9 «Локальные службы сети» описывает настройку многих из распространенных сетевых серверов, в частности сервера настройки DHCP, сервера печати LPD, почтовых серверов POP и IMAP, сетевой файловой системы NFS (Network File System), серверов файлов и печати Samba, а также сетевой информационной службы NIS (Network Information System).

Глава 10 «sendmail» рассказывает о настройке sendmail – демона, отвечающего за доставку сообщений электронной почты.

Глава 11 «Настройка Apache» описывает настройку веб-сервера Apache.

В главе 12 «Сетевая безопасность» речь идет о том, как использовать современный Интернет, не подвергаясь излишнему риску. Глава посвящена угрозам безопасности, связанным с работой в сети, и возможным способам защиты от них.

Глава 13 «Разрешение проблем TCP/IP» рассказывает, какие действия можно предпринять, если что-то идет не так. Описаны методы и инструменты диагностирования проблем TCP/IP, приводятся примеры реальных проблем и их решений.

Приложение A «Инструментарий PPP» – это справочное руководство по различным программам, применяемым в настройке последовательных портов для работы по TCP/IP. Описаны программы `dip`, `pppd` и `chat`.

Приложение B «gated, справочник» – это справочное руководство по языку настройки пакета маршрутизации `gated`.

Приложение C «named, справочник» – это справочное руководство по серверу имен BIND (Berkeley Internet Name Domain).

Приложение D «dhcpd, справочник» – это справочное руководство по демону `dhcpd` (Dynamic Host Configuration Protocol Daemon).

Приложение E «sendmail, справочник» является справочным руководством по синтаксису, параметрам и ключам настройки sendmail.

В приложении F «Файл httpd.conf в Solaris» приводится содержимое файла настройки Apache, о котором идет речь в главе 11.

Приложение G «Выдержки из RFC» содержит информативные справочные фрагменты по протоколам из документов RFC, которые дополняют примеры диагностирования и разрешения проблем главы 13. Кроме того, в приложении содержатся сведения о том, где взять полноценные документы RFC.

Версии Unix

Большинство примеров книги относятся к Red Hat Linux, наиболее популярному в настоящее время дистрибутиву Linux, а также к Solaris 8, операционной системе от Sun, основанной на Unix System V. По счастью, программные средства TCP/IP достаточно стандартны в разных системах, что делает приво-

димые примеры универсальными – они должны работать в любых системах Linux, System V и BSD. Незначительные различия в выходных данных команд и параметрах командной строки не должны представлять затруднений.

Некоторые дополнительные сетевые приложения имеют собственные номера версий, не зависящие от версий операционных систем. Номера версий дополнительных пакетов упоминаются, когда это уместно. Наиболее важные из подобных пакетов:

BIND

Приводимое описание пакета BIND относится к версиям ветви 8, работающим в ОС Solaris 8. BIND 8 – это версия пакета BIND, поставляемая в составе Solaris и поддерживающая все стандартные типы записей ресурсов. В отношении базовых настроек версии BIND 8 и более новая BIND 9 не сильно различаются.

sendmail

Информация о пакете sendmail соответствует версии 8.11.3 и должна быть применима для всех других версий sendmail ветви 8.

Типографские соглашения

В книге использованы следующие типографские соглашения:

Курсив

Применяется для отображения имен файлов, каталогов, узлов, доменов, а также для выделения новых терминов.

`Моноширинный шрифт`

Применяется для отображения содержимого файлов и вывода команд, а также для выделения команд, параметров и ключевых слов в тексте.

`Моноширинный полужирный шрифт`

Применяется в примерах для выделения команд, набираемых пользователем.

`Моноширинный курсив`

Используется в примерах и в тексте для выделения переменных, значения которых должны быть подставлены в зависимости от обстоятельств. (Например, переменную *`filename`* необходимо заменять конкретным именем файла.)

`%, #`

Команды, вводимые в диалоговом режиме, отмечены приглашением стандартного интерпретатора команд C shell (`%`). Если команда должна выполняться с полномочиями администратора, она отмечается стандартным приглашением суперпользователя (`#`). В примерах, где участвует сразу несколько машин сети, приглашению может предшествовать имя той машины, на которой выполняется команда.

`[ ключ ]`

В описании синтаксиса команд необязательные аргументы заключаются в квадратные скобки. Так, запись `ls [ -l ]` означает, что ключ `-l` является необязательным.

Нам важно знать ваше мнение

Мы тщательно, насколько представляется возможным, проверили всю информацию в настоящей книге, но вы можете обнаружить, что возможности программ изменились (или наши ошибки!). Пожалуйста, сообщайте обо всех найденных ошибках, а также присылайте предложения, связанные с последующими изданиями книги по адресу:

O'Reilly & Associates, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
(800) 998-9938 (в США или Канаде)
(707) 829-0515 (международный/местный)
(707) 829-0104 (факс)

Издательством O'Reilly создана веб-страница, посвященная этой книге, на которой доступна информация о найденных ошибках и будут появляться разнообразные дополнительные сведения. Страница доступна по адресу:

http://www.oreilly.com/catalog/tcp3

Комментарии и технические вопросы, связанные с книгой, присылайте по адресу электронной почты:

bookquestions@oreilly.com

На веб-сайте издательства O'Reilly представлена дополнительная информация о книгах, конференциях, программном обеспечении, источниках информации и Сети O'Reilly (O'Reilly Network):

http://www.oreilly.com

Чтобы узнать, чем еще занимается Крэйг Хант, посетите его веб-сайт, *http://www.wrotethebook.com*.

Благодарности

Я хотел бы поблагодарить многих людей, которые помогли подготовить эту книгу. Те, кто участвовал в подготовке двух первых изданий, в первую очередь заслуживают благодарности; их вклад живет и в этом издании. Первое издание: Джон Уок (John Wack), Мэтт Бишоп (Matt Bishop), Вьетс Венема (Wietse Venema), Эрик Оллман (Eric Allman), Джефф Хониг (Jeff Honig),

Скотт Брим (Scott Brim) и Джон Дорган (John Dorgan). Второе издание: снова Эрик Оллман, Брайан Косталес (Bryan Costales), Крикет Ли (Cricket Liu), Пол Альбитц (Paul Albitz), Тед Лемон (Ted Lemon), Элизабет Цвики (Elizabeth Zwicky), Брент Чепмен (Brent Chapman), Симсон Гарфинкель (Simson Garfinkel), Джефф Седайо (Jeff Sedayao), а также Элин Фриш (Жleen Frisch).

В третьем издании книга стала лучше благодаря участию людей, многие из которых сами являются авторами. Они не только помогли мне с техническими вопросами – благодаря им я стал лучше писать. Три автора заслуживают отдельной благодарности. Многочисленные комментарии Крикета Ли, одного из авторов лучшей в мире книги о DNS, позволили улучшить раздел, посвященный системе доменных имен. Дэвид Колье-Браун (David Collier-Brown), один из авторов книги «Using Samba», написал исчерпывающую техническую рецензию на материал по Samba. Чарльз Олдс (Charles Aulds), автор бестселлера об администрировании сервера Apache, оказал помощь при написании главы о настройке Apache. Все эти люди помогли мне улучшить книгу в третьем издании. Спасибо!

Сотрудники издательства O'Reilly & Associates постоянно помогали мне. Деб Кэмерон (Deb Cameron), мой редактор, заслуживает отдельной благодарности. Ее усилий хватало и на развитие книги, и на общение со своей новорожденной красавицей, Вифанией Розой. Эмили Квилл (Emily Quill) играла роль выпускающего редактора и куратора проекта. Джефф Холкомб (Jeff Holcomb) и Джейн Эллин (Jane Ellin) выполняли проверку качества. За техническую помощь спасибо Леанне Сойлемез (Leanne Soylemez). Том Динз (Tom Dinse) создал указатель. Автором обложки является Эди Фридман (Edie Freedman), а стилевое оформление текста делала Мелани Вонг (Melanie Wang). Нил Уоллз (Neil Walls) занимался преобразованием текста из формата Microsoft Word в формат редактора Framemaker. Иллюстрации предшествующих изданий, созданные Крисом Райли (Chris Reilley) и Робертом Романо (Robert Romano), были обновлены стараниями Роберта Романо и Джессамина Рида (Jessamyn Read).

Наконец, я хочу поблагодарить мою семью – Кэти, Сару, Дэвида и Ребекку. Только их стараниями давление сроков сдачи материала до сих пор не свело меня с ума. Ребята, вы лучше всех.

- *Варианты настройки маршрутизации*
- *Простейшая таблица маршрутизации*
- *Создание статической таблицы маршрутизации*
- *Протоколы внутренней маршрутизации*
- *Протоколы внешней маршрутизации*
- *Демон шлюзовой маршрутизации*
- *Настройка gated*

7

Настройка маршрутизации

Маршрутизация связывает Интернет воедино. Без маршрутизации трафик TCP/IP ограничивался бы единственной физической сетью. Маршрутизация позволяет данным из локальной сети достигать адресатов во всех концах света, проходя через многочисленные промежуточные сети.

Важность маршрутизации и сложность взаимосвязей сетей Интернета делают проектирование протоколов маршрутизации серьезным испытанием для разработчиков сетевых программ. Как следствие, в большинстве материалов по маршрутизации говорится об архитектуре протоколов. Очень немного написано о важной задаче правильной настройки протоколов маршрутизации. И это при том, что большая часть возникающих проблем может быть обусловлена неверной настройкой маршрутизаторов, а вовсе не плохо спроектированными алгоритмами маршрутизации. Задача системного администратора – убедиться в том, что маршрутизация на вверенных ему системах настроена правильно. Именно этой задаче посвящена настоящая глава.

Варианты настройки маршрутизации

Прежде всего, следует провести черту между *маршрутизацией* и *протоколами маршрутизации*. Все системы выполняют маршрутизацию данных, но далеко не на каждой функционирует протокол маршрутизации. *Маршрутизация* – это действие по пересылке дейтаграммы, основанное на информации из таблицы маршрутизации. *Протоколы маршрутизации* – это программы, которые обмениваются информацией для построения таблиц маршрутизации.

Настройка маршрутизации сети не обязательно требует присутствия протокола маршрутизации. В случаях, когда информация маршрутизации не изме-

няется – например, если существует лишь один возможный маршрут, – администратор системы, как правило, создает таблицу маршрутизации вручную. Некоторые сети не имеют доступа к другим сетям TCP/IP, а следовательно, не требуют от администратора даже создания таблицы маршрутизации. Вот три наиболее распространенных варианта настройки маршрутизации.[1]

Примитивная маршрутизация

Сеть, полностью изолированная от всех других сетей TCP/IP, требует лишь примитивной маршрутизации. Простейшая таблица маршрутизации обычно создается при настройке сетевых интерфейсов: на каждый из интерфейсов добавляется по одному маршруту. Если отсутствует прямой доступ к другим сетям TCP/IP и разделение на подсети, других таблиц маршрутизации может не потребоваться.

Статическая маршрутизация

В сети с ограниченным числом шлюзов в другие сети TCP/IP имеет смысл применять статическую маршрутизацию. Для сети с единственным шлюзом статическая маршрутизация станет идеальным выбором. Статическая таблица маршрутизации создается вручную администратором – при помощи команды `route`. Статические таблицы маршрутизации не способны реагировать на изменения в сети, поэтому они оптимальны для случаев, когда маршруты постоянны.

Динамическая маршрутизация

В сетях, где существует несколько путей к одному пункту назначения, следует применять динамическую маршрутизацию. Динамическая таблица маршрутизации создается на основе сведений, которыми обмениваются протоколы маршрутизации. Задача этих протоколов – распространение информации, позволяющей автоматически настраивать маршруты в случае изменений в сети. Протоколы маршрутизации справляются со сложными задачами быстрее и точнее, чем при всем желании способен это делать системный администратор. Протоколы маршрутизации позволяют не только переключаться на резервный маршрут, если становится непроходимым основной, но также выбирать «лучший» маршрут из нескольких доступных. Протоколы маршрутизации следует применять во всех сетях, где существуют альтернативные маршруты.

Маршруты создаются вручную системным администратором или динамически – протоколами маршрутизации. Но, независимо от способа создания, они, в конечном итоге, оказываются в таблице маршрутизации.

Простейшая таблица маршрутизации

Взглянем на содержимое таблицы маршрутизации, созданной в процессе настройки сетевых интерфейсов системы Solaris 8 посредством `ifconfig`:

[1] В главе 4 представлены руководящие указания по выбору варианта настройки маршрутизации для ваших сетей.

```
% netstat -rn
Routing Table: IPv4
  Destination           Gateway           Flags  Ref   Use   Interface
-------------------- -------------------- ----- ----- ------ ---------
172.16.12.0          172.16.12.15          U         1      8  dnet0
224.0.0.0            172.16.12.15          U         1      0  dnet0
127.0.0.1            127.0.0.1             UH       20   3577  lo0
```

Первая запись определяет маршрут в сеть 172.16.12.0, пролегающий через интерфейс dnet0. 172.16.12.15 – это не адрес внешнего шлюза, но адрес, назначенный интерфейсу dnet0 на данном узле. Оставшиеся две записи не являются определениями маршрутов; они отражают принятые для программного обеспечения соглашения. 224.0.0.0 – групповой адрес. Данная запись предписывает Solaris выполнять доставку групповых сообщений через интерфейс 172.16.12.15. Последняя запись определяет кольцевой маршрут к узлу *localhost*, созданный при настройке интерфейса lo0.

Обратите внимание на поле Flags этих записей. Для всех записей установлен флаг U (up), показывающий, что маршруты готовы к использованию, но ни разу не встречается флаг G (gateway). Флаг G отмечает внешние шлюзы. Флаг G в данном случае отсутствует потому, что все маршруты пролегают через локальные интерфейсы и ни один не проходит через внешний шлюз.

Кроме того, для кольцевого маршрута установлен флаг H (host), который указывает, что лишь один узел доступен по этому маршруту. Значение флага становится ясным, если взглянуть на поле Destination записи кольцевого маршрута. Оно содержит адрес конкретного узла, а не адрес сети. Адрес кольцевой сети – 127.0.0.0. Данный адрес пункта назначения (127.0.0.1) является адресом конкретного узла – *localhost*. В одних системах используется маршрут в кольцевую сеть, в других – маршрут к локальному узлу, но во всех системах в таблице маршрутизации существует некоторый маршрут для кольцевого интерфейса.

В таблице из примера существует один маршрут к узлу, но в большинстве случаев маршруты прокладываются к сетям. Первая причина тому – необходимость сократить размер таблицы маршрутизации. Единственная сеть организации может состоять из сотен узлов. Сеть Интернет состоит из тысяч сетей и из миллионов узлов. Таблица маршрутизации, содержащая маршрут для каждого из узлов, была бы неуправляема.

Приведенная таблица содержит лишь один маршрут в физическую сеть, 172.16.12.0. Следовательно, данная система может общаться только с узлами, расположенными в указанной сети. Ограниченные возможности этой таблицы маршрутизации легко увидеть при помощи команды `ping`. `ping` вынуждает удаленный узел вернуть пакет локальному узлу – при помощи ICMP-сообщения эхо (Echo Message). Прохождение пакетов до удаленного узла и обратно означает, что два узла могут успешно обмениваться данными.

Чтобы проверить возможности таблицы маршрутизации этой системы, прежде всего, выполните команду `ping` для другого узла локальной сети:

```
% ping -s crab
PING crab.wrotethebook.com: 56 data bytes
64 bytes from crab.wrotethebook.com (172.16.12.1): icmp_seq=0. time=11. ms
64 bytes from crab.wrotethebook.com (172.16.12.1): icmp_seq=1. time=10. ms
^C
----crab.wrotethebook.com PING Statistics----
2 packets transmitted, 2 packets received, 0% packet loss
round-trip (ms)  min/avg/max = 10/10/11
```

ping выводит по одной строке на каждый полученный ответ ICMP ECHO_RESPONSE.[1] Когда выполнение ping прерывается, программа отображает сводную статистику. Мы наблюдаем успешный обмен данными с узлом *crab*. Но если мы обратимся к узлу, который не принадлежит сети 172.16.12.0, скажем, к узлу издательства O'Reilly, результат будет другим.

```
 % ping 207.25.98.2
sendto: Network is unreachable
```

В данном случае сообщение «sendto: Network is unreachable» показывает, что локальный узел не знает, как отправить данные в сеть узла 207.25.98.2. В таблице маршрутизации системы – лишь три маршрута, и ни один из них не ведет в сеть 207.25.98.0.

Данная таблица маршрутизации не позволяет общаться даже с машинами другой подсети *books-net*. Подтверждает сказанное команда ping, выполненная для узла другой подсети:

```
% ping 172.16.1.2
sendto: Network is unreachable
```

Эти тесты показывают, что простейшая таблица маршрутизации, созданная в процессе настройки сетевых интерфейсов, позволяет обмениваться данными лишь с другими узлами локальной сети. Если сети не требуется доступ к другим сетям TCP/IP, вполне можно обойтись и таким вариантом. В противном случае следует добавить прочие маршруты в таблицу маршрутизации.

Создание статической таблицы маршрутизации

Как мы видели, простейшая таблица маршрутизации позволяет работать лишь с узлами, расположенными в напрямую подключенных физических сетях. Обращение к удаленным узлам требует добавления в таблицу маршрутизации маршрутов, пролегающих через внешние шлюзы. Одним из решений задачи является создание статической таблицы маршрутизации при помощи команд route.

---

[1] Команда ping системы Sun отобразит лишь сообщение «*crab* is alive», если не использовать ключ -s. Большинство реализаций ping не требуют наличия ключа -s.

С помощью команды Unix route можно вручную добавить или удалить записи таблицы маршрутизации. Так, чтобы добавить маршрут 207.25.98.0 в таблицу маршрутизации системы Solaris, наберите:

```
# route add 207.25.98.0 172.16.12.1 1
add net 207.25.98.0: gateway crab
```

Первый аргумент команды route – ключевое слово add. Первым ключевым словом в командной строке route может быть add либо delete, и оно предписывает route соответственно добавить новый маршрут либо удалить существующий. Значения по умолчанию нет – в отсутствие ключевых слов route просто отображает таблицу маршрутизации.

Следующее значение – адрес пункта назначения, то есть адрес, доступный по этому маршруту. Конечный адрес может быть представлен IP-адресом, именем сети из файла */etc/networks*, именем узла из файла */etc/hosts* либо ключевым словом default. Поскольку добавление большинства маршрутов происходит достаточно рано в процессе загрузки системы, численные IP-адреса используются чаще, чем имена. Такое положение снимает зависимость настройки маршрутизации от доступности и состояния серверов имен. Всегда используйте полные численные адреса, состоящие из четырех байтов. route выполняет расширение адреса, содержащего меньше четырех байтов, и полученный результат может достаточно сильно отличаться от желаемого.[1]

Если конечный адрес представлен ключевым словом default, route создает *маршрут по умолчанию*.[2] Маршрут по умолчанию используется в случаях, когда отсутствует явный маршрут в конечную сеть. Зачастую маршрут по умолчанию – единственный необходимый маршрут. Если сеть работает с одним шлюзом, используйте маршрут по умолчанию, чтобы передавать через этот шлюз весь трафик, предназначенный внешним сетям.

Далее в командной строке route следует адрес шлюза[3], а именно IP-адрес внешнего шлюза, через который передаются данные. Шлюз должен располагаться в сети с прямым подключением. Маршруты TCP/IP определяют следующий транзитный участок в пути к пункту назначения. Этот следующий транзитный участок должен быть напрямую доступен локальному узлу; следовательно, он должен быть в сети с прямым подключением.

И последний аргумент в командной строке – метрика маршрутизации. Аргумент метрики отсутствует при удалении маршрутов, но в некоторых более старых системах его присутствие обязательно для добавления маршрута. В Solaris 8 метрика является необязательной. Системы, требующие наличия аргумента метрики, используют его, только чтобы определить, пролегает

---

1 Некоторые реализации route преобразуют «26» в 0.0.0.26, хотя «26» вполне может означать сеть Milnet (26.0.0.0).

2 С маршрутом по умолчанию связан адрес сети 0.0.0.0.

3 В Linux значения командной строки route предваряются ключевыми словами. Например route add -net 207.25.98.0 netmask 255.255.255.0 gw 172.16.12.1. Сверьтесь с документацией своей системы.

маршрут через напрямую подключенный интерфейс или же через внешний шлюз. В случае нулевой метрики маршрут создается как проходящий через локальный интерфейс, и флаг G, который мы наблюдали в выводе команды `netstat -i`, отсутствует. Если значение метрики больше нуля, для маршрута устанавливается флаг G, и адрес считается адресом внешнего шлюза. В статической маршрутизации метрики не используются по назначению. Меняющиеся значения метрики имеют смысл только в динамической маршрутизации.

Создание статических маршрутов

В качестве примера займемся настройкой статической маршрутизации гипотетической рабочей станции *rodent*. На рис. 7.1 представлена подсеть 172.16.12.0. В подсети существует два шлюза, *crab* и *horseshoe*. *crab* – это шлюз в тысячи сетей Интернета; *horseshoe* обеспечивает доступ к другим подсетям *books-net*. Используем узел *crab* в качестве шлюза по умолчанию, поскольку через него пролегают тысячи маршрутов. Число маршрутов через узел *horseshoe* ограничено, их можно создать вручную. Выбор шлюза по умолчанию диктуется именно числом маршрутов, пролегающих через шлюз, а вовсе не объемом трафика. Даже если большая часть трафика с узла *rodent* уходит через *horseshoe* к другим узлам *books-net*, шлюзом по умолчанию должен быть *crab*.

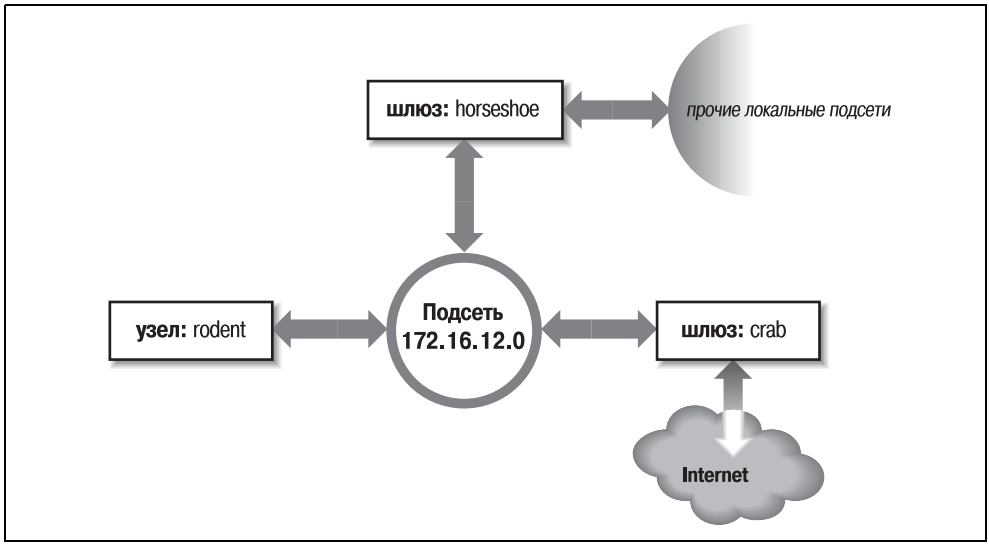

***Рис. 7.1.** Маршрутизация в подсети*

Чтобы создать маршрут по умолчанию на узле *rodent*, наберите:

```
# route add default gw 172.16.12.1
```

Конечный адрес представлен ключевым словом `default`, а в качестве адреса шлюза (172.16.12.1) выступает адрес узла *crab*. Теперь *crab* является шлю-

зом по умолчанию для узла *rodent*. Обратите внимание, что синтаксис команды отличается от описанного в предшествующем примере по route системы Solaris. Система *rodent* работает под управлением Linux. В Linux большинству аргументов команды route предшествуют ключевые слова. В данном случае адресу шлюза предшествует ключевое слово gw.

Создав маршрут по умолчанию, изучим таблицу маршрутизации, чтобы убедиться, что все в порядке:[1]

```
# route -n
Kernel IP routing table
Destination   Gateway       Genmask         Flags Metric Ref Use Iface
172.16.12.0   0.0.0.0       255.255.255.0   U     0      0     0 eth0
127.0.0.0     0.0.0.0       255.0.0.0       U     0      0     0 lo
0.0.0.0       172.16.12.1   0.0.0.0         UG    0      0     0 eth0
```

Снова воспользуемся командой ping, чтобы выяснить, может ли теперь *rodent* общаться с удаленными узлами. Если нам повезет[2], удаленный узел ответит, и мы увидим:

```
% ping 207.25.98.2
PING 207.25.98.2: 56 data bytes
64 bytes from ruby.ora.com (207.25.98.2): icmp_seq=0. time=110. ms
64 bytes from ruby.ora.com (207.25.98.2): icmp_seq=1. time=100. ms
^C
----207.25.98.2 PING Statistics----
2 packets transmitted, 2 packets received, 0% packet loss
round-trip (ms)  min/avg/max = 100/105/110
```

Такой вывод свидетельствует об успешном обмене данными с удаленным узлом, то есть у нас теперь есть действующий маршрут в сеть Интернет.

Однако мы еще не создали маршруты к другим сегментам сети *books-net*. Если мы выполним прозвонку для узла из другой подсети, нас ожидает нечто интересное:

```
% ping 172.16.1.2
PING 172.16.1.2: 56 data bytes
ICMP Host redirect from gateway crab.wrotethebook.com (172.16.12.1)
 to horseshoe.wrotethebook.com (172.16.12.3) for ora.wrotethebook.com (172.16.1.2)
64 bytes from ora.wrotethebook.com (172.16.1.2): icmp_seq=1. time=30. ms
^C
----172.16.1.2 PING Statistics----
1 packets transmitted, 1 packets received, 0% packet loss round-trip (ms)  min/avg/
max = 30/30/30
```

---

[1] В Solaris для изучения таблиц маршрутизации всегда применяется netstat. В Linux можно использовать netstat или route, но обычно используется route.

[2] Возможно, что удаленный узел неисправен или не работает. В таком случае ping не получит ответа. Не отчаивайтесь, повторите попытку с другим узлом.

rodent считает, что все пункты назначения достижимы через маршрут по умолчанию. Следовательно, даже данные, предназначенные другим подсетям, передаются через шлюз *crab*. Если *rodent* передает узлу *crab* данные, которые должны пройти через *horseshoe*, *crab* отвечает узлу *rodent* сообщением ICMP Redirect, которое предписывает обращаться к узлу *horseshoe*. (Сообщение ICMP Redirect описано в главе 1.) Команда `ping` показывает сообщение ICMP Redirect в действии. Перенаправление непосредственно влияет на таблицу маршрутизации:

```
# route -n
Kernel IP routing table
Destination   Gateway       Genmask         Flags Metric Ref Use Iface
172.16.12.0   0.0.0.0       255.255.255.0   U     0      0     0 eth0
127.0.0.0     0.0.0.0       255.0.0.0       U     0      0     0 lo
0.0.0.0       172.16.12.1   0.0.0.0         UG    0      0     0 eth0
172.16.1.2    172.16.12.3   255.255.255.0   UGHD  0      0   514 eth0
```

Маршрут с установленным флагом D был создан сообщением ICMP Redirect.

Некоторые администраторы при проектировании сетей пользуются сообщениями ICMP Redirect. Все узлы настраиваются на работу с маршрутом по умолчанию, даже в сетях с многочисленными шлюзами. Шлюзы обмениваются информацией маршрутизации посредством соответствующих протоколов и перенаправляют узлы к лучшим шлюзам для каждого конкретного маршрута. Такой вид маршрутизации, зависимой от сообщений ICMP Redirect, получил широкое распространение благодаря персональным компьютерам. Многие персональные компьютеры неспособны работать с протоколами маршрутизации; отдельные ранние модели не имели команды `route` и были ограничены маршрутом по умолчанию. Сообщения ICMP Redirect оказались подходящим способом поддержки таких клиентов. Кроме того, этот вид маршрутизации прост в настройке и может эффективно претворяться в жизнь при помощи сервера настройки, поскольку каждый узел использует один-единственный маршрут (по умолчанию). По этим причинам некоторые руководители сетей поощряют многочисленные перенаправления ICMP.

Прочие сетевые администраторы предпочитают избегать перенаправлений ICMP и управлять содержимым таблицы маршрутизации напрямую. Чтобы обойтись без перенаправлений, мы можем создать конкретные маршруты для каждой конкретной подсети при помощи команды `route`:

```
# route add -net 172.16.1.0 netmask 255.255.255.0 gw 172.16.12.3
# route add -net 172.16.6.0 netmask 255.255.255.0 gw 172.16.12.3
# route add -net 172.16.3.0 netmask 255.255.255.0 gw 172.16.12.3
# route add -net 172.16.9.0 netmask 255.255.255.0 gw 172.16.12.3
```

Узел *rodent* имеет прямое подключение только к подсети 172.16.12.0, поэтому адреса всех шлюзов в таблице маршрутизации начинаются со значения 172.16.12. Вот так выглядит полученная таблица маршрутизации:

```
# route -n
Kernel IP routing table
```

```
Destination     Gateway         Genmask         Flags Metric Ref Use Iface
172.16.6.0      172.16.12.3     255.255.255.0   UG    0      0     0 eth0
172.16.3.0      172.16.12.3     255.255.255.0   UG    0      0     0 eth0
172.16.12.0     0.0.0.0         255.255.255.0   U     0      0     0 eth0
172.16.1.0      172.16.12.3     255.255.255.0   UG    0      0     0 eth0
172.16.9.0      172.16.12.3     255.255.255.0   UG    0      0     0 eth0
127.0.0.0       0.0.0.0         255.0.0.0       U     0      0     0 lo
0.0.0.0         172.16.12.1     0.0.0.0         UG    0      0     0 eth0
172.16.1.2      172.16.12.3     255.255.255.0   UGHD  0      0   514 eth0
```

Маршрут по умолчанию (через узел *crab*) позволяет работать с внешними сетями, а конкретные маршруты (через узел *horseshoe*) – достигать прочих подсетей в пределах *books-net*. Повторное выполнение тестов командой `ping` дает устойчивые положительные результаты. Однако при добавлении новых подсетей придется вручную добавить и соответствующие маршруты в таблицу маршрутизации. Кроме того, при перезагрузке системы все записи статической таблицы маршрутизации будут утрачены. Таким образом, применение статической маршрутизации требует создания маршрутов в процессе загрузки системы.

Создание статических маршрутов при загрузке

Приняв решение использовать статическую маршрутизацию, администратор должен внести два изменения в загрузочные файлы:

1. Добавить необходимые команды `route` в один из загрузочных сценариев.
2. Удалить из загрузочных файлов все команды, запускающие протоколы маршрутизации.

Чтобы добавить команды статической маршрутизации в загрузочный сценарий, следует, прежде всего, выбрать, в какой именно сценарий их поместить. В системах BSD и Linux под локальные дополнения процесса загрузки отведен файл *rc.local*. Он выполняется последним из загрузочных, а потому прекрасно подходит для внесения изменений в стандартный процесс загрузки. В системе Red Hat Linux, которую мы используем в примерах, полное имя файла *rc.local* – */etc/rc.d/rc.local*. В системе Solaris добавьте в файл */etc/init.d/inetinit* следующие команды:

```
route -n add default 172.16.12.1 > /dev/console
route -n add 172.16.1.0 172.16.12.3 > /dev/console
route -n add 172.16.6.0 172.16.12.3 > /dev/console
route -n add 172.16.3.0 172.16.12.3 > /dev/console
route -n add 172.16.9.0 172.16.12.3 > /dev/console
```

Ключ `-n` предписывает `route` отображать численные адреса в информационных сообщениях. Внося команды `route` в загрузочный файл системы Solaris, используйте ключ `-n`, чтобы запретить `route` тратить время, обращаясь к серверу имен, который в этот момент, вполне возможно, еще не функционирует. Ключ `-n` не требуется в случае системы Linux, поскольку Linux не сопровождает создание маршрутов информационными сообщениями.

Создав команды `route`, проверьте, не запускает ли сценарий модуль протокола маршрутизации, и при необходимости заблокируйте выполнение соответствующих строк сценария при помощи комментариев. Присутствие протокола маршрутизации ни к чему, если используется статическая маршрутизация. В системе Solaris из нашего примера программный модуль маршрутизации запускается только в случае, если в системе существует более одного сетевого интерфейса (то есть если она является маршрутизатором) либо если создан файл */etc/gateways*. (Об этом файле поговорим чуть позже.) Не выполнено ни одно из условий, значит, демон маршрутизации не будет запущен в процессе загрузки и от нас не требуются какие-либо дополнительные действия, помимо создания команд `route`.

Прежде чем вносить изменения в рабочую систему, сверьтесь с документацией. Возможно, следует изменить другой сценарий либо скорректировать полное имя демона маршрутизации. Только документация на конкретную систему содержит точные указания.

Несмотря на возможные отличия в именах загрузочных файлов, процедура в целом одинакова для всех систем. Описанные простые шаги – все, что нужно для настройки статической маршрутизации. Собственно говоря, проблема со статической маршрутизацией заключается не в ее настройке, но в сопровождении, если речь идет о меняющейся сетевой среде. Протоколы маршрутизации обладают достаточной гибкостью, чтобы справляться как с простыми, так и со сложными средами маршрутизации. Именно поэтому некоторые из последовательностей загрузки автоматически инициируют работу протоколов маршрутизации. Однако большинству систем Unix требуется лишь статический маршрут по умолчанию. Протоколы маршрутизации обычно нужны только маршрутизаторам.

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.[1] 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 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 маршруты), – критерий, по которому различаются протоколы маршрутизации. Протоколов внутренней маршрутизации существует несколько:

[1] Автономные системы описаны в главе 2.

- *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)[1], наберите такую команду:

```
# routed
```

Команда routed часто фигурирует без аргументов командной строки, но 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

[1] В некоторых системах демон маршрутизации имеет имя in.routed.

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 значение, используемое администратором для расстановки приоритетов маршрутов. (Системный администратор волен назначать маршруту любое значение 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 информа-

цию 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, приведенным 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 значения метрик – насущная необходимость, позволяющая сократить воздействие счета до бесконечности. Однако ограниченный размер сети – наименьший из недостатков протокола RIP. Настоящая задача по улучшению RIP связана с решением двух других проблем – медленной сходимости и классовой маршрутизации.

В 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. 7.2 отражена сеть, в которой может возникнуть такая проблема.

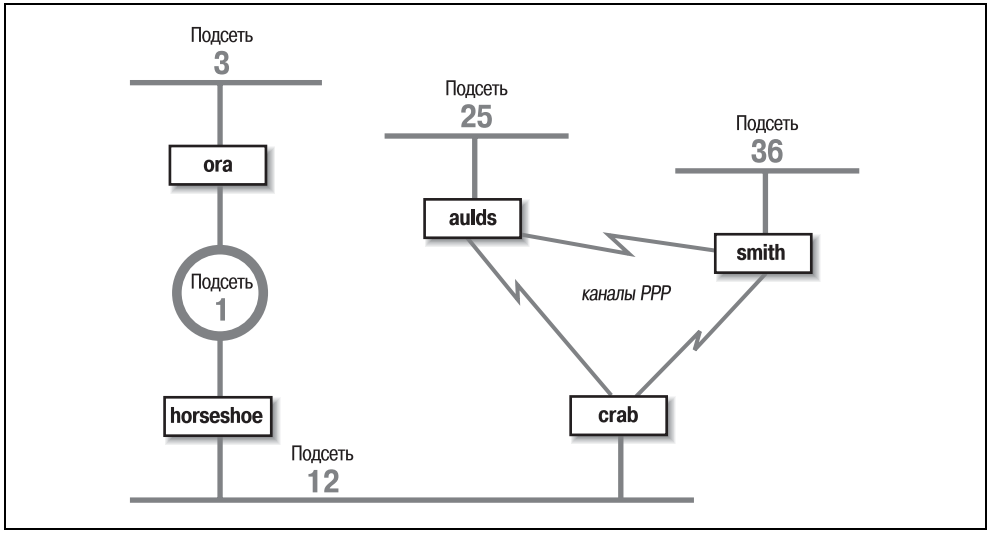

Рис. 7.2. *Пример сети*

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 *crab* 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 *horseshoe*, 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 3 нельзя получить через шлюз *crab*.

Эти две технологии решают описанную выше проблему. Но что будет, если произойдет сбой в работе узла *crab*? Взгляните на рис. 7.2. 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 заканчивается 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 сче-

те 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 TCP/IP, 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 надсетями 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 со-*

стояния 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:

Области (Areas)

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

Магистраль (Backbone)

Магистраль – 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)

Оконечная область 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 1 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 1 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. 7.2, 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. 7.3.

Алгоритм Дейкстры создает карту следующим образом:

1. Локальная система устанавливается в качестве корня карты и получает нулевую стоимость.

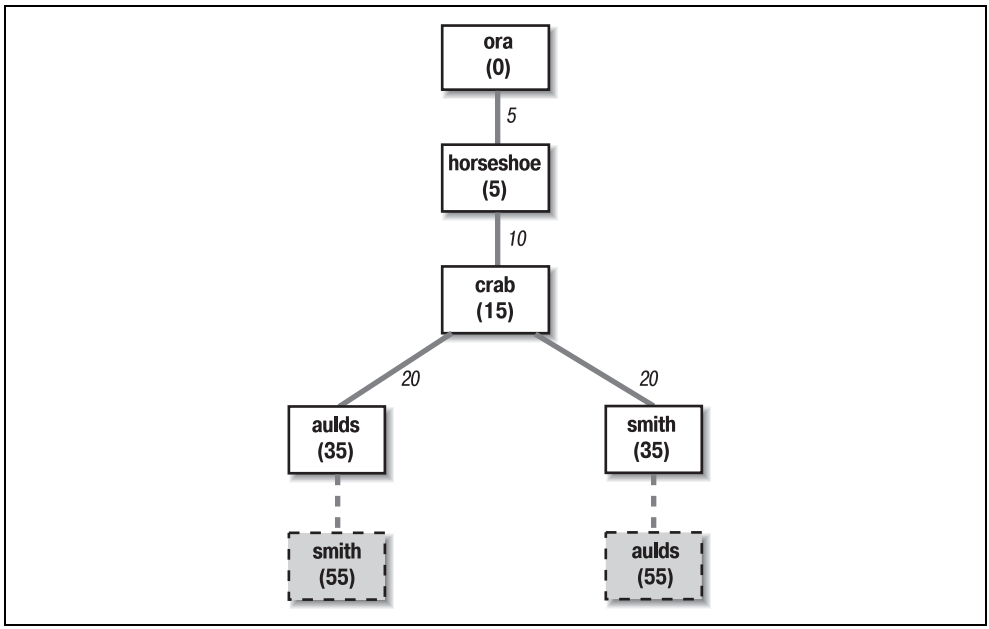

Рис. 7.3. *Граф сети*

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.

Информация базы данных состояния каналов накапливается и распространяется по простым и эффективным правилам. OSPF-маршрутизатор выполняет обнаружение своих соседей при помощи пакетов Hello.[1] 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.

Обратимся к рис. 7.2 за очередным примером. 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 подсетях 1 и 12. *ora* получает LSA и рассылает его в подсети 3. *crab* получает LSA и рас-

[1] Не путайте пакеты Hello с протоколом Hello. Здесь речь идет о пакетах OSPF Hello.

сылает 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 оказывается 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 конструкций...

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.

В последующих разделах мы изучим программный пакет `gated` (Gateway Routing Daemon, демон шлюзовой маршрутизации), сочетающий протоколы как внутренней, так и внешней маршрутизации, и рассмотрим примеры работы с RIP, RIPv2, OSPF и BGP.

Демон шлюзовой маршрутизации

Разработка ПО маршрутизации для систем Unix общего назначения выглядит весьма скромно. На большинстве площадок системы Unix используются для решения лишь самых простых задач маршрутизации, и в этих случаях обычно адекватен протокол RIP. В случаях сложной маршрутизации, требующей применения более совершенных протоколов, используются специализированные аппаратные маршрутизаторы, предназначенные для решения именно этой задачи. Многие из более совершенных протоколов маршрутизации доступны системам Unix только в пакете `gated`. `gated` сочетает несколько различных протоколов маршрутизации в одном программном пакете.

Кроме того, `gated` предоставляет и другие возможности, обычно присущие только специализированным маршрутизаторам:

- Одна система может работать с целым набором протоколов маршрутизации. `gated` объединяет информацию маршрутизации, полученную по различным протоколам, и выбирает «лучшие» маршруты.
- Маршруты, полученные посредством протоколов внутренней маршрутизации, могут распространяться через протокол внешней маршрутизации, что делает возможным динамическое изменение информации достижимости, распространяемой внешним образом, отражающее результаты изменения внутренних маршрутов.
- Правила маршрутизации позволяют создавать фильтры для приема и распространения маршрутов.
- Все протоколы настраиваются при помощи единственного файла (*/etc/gated.conf*). Для настройки применяется единый синтаксис команд.
- Пакет `gated` постоянно дорабатывается. Применение `gated` гарантирует, что ваш пакет маршрутизации шагает в ногу со временем.

Значение предпочтения в gated

Каждая реализация протокола маршрутизации имеет две стороны. Первая сторона – внешняя – реализует обмен информацией маршрутизации с удаленными системами. Вторая сторона – внутренняя – обновляет таблицу маршрутизации, используя сведения, полученные от удаленных систем. Например, когда протокол OSPF обменивается пакетами Hello в поисках соседей, то выполняет внешнюю функцию протокола. Когда протокол OSPF добавляет маршрут в таблицу маршрутизации, то выполняет внутреннюю функцию.

Внешние функции протоколов, реализованные в `gated`, соответствуют внешним функциям в других реализациях протоколов. Однако внутренняя сторона `gated` присуща только системам Unix. `gated` обрабатывает информацию маршрутизации от различных протоколов, каждый из которых имеет собственную метрику для определения лучшего маршрута, и объединяет эту информацию перед тем, как внести обновления в таблицу маршрутизации. До появления `gated` сосуществующие в системе Unix протоколы обновляли таб-

лицу маршрутизации совершенно независимо друг от друга. В итоге в таблице оказывался маршрут, внесенный последним, – не обязательно лучший.

В случае нескольких протоколов маршрутизации и нескольких сетевых интерфейсов система может получать маршруты одного направления по различным протоколам. `gated` сравнивает такие маршруты и пытается выбрать лучший. Однако метрики различных протоколов не подлежат прямому сравнению. Каждый протокол использует собственную метрику. Это может быть счетчик транзитных участков, задержка для маршрута либо произвольное значение, указанное администратором. Для выбора лучшего маршрута `gated` требуется больше, чем метрики протоколов. Демон использует собственную оценку, позволяющую выбирать между маршрутами, полученными от нескольких протоколов или интерфейсов. Данное значение известно как *значение предпочтения*.

Значения предпочтения позволяют `gated` объединить сведения, полученные из нескольких источников, в одной таблице маршрутизации. В табл. 7.1 перечислены источники маршрутов `gated` и значения предпочтения, присвоенные каждому из них по умолчанию. Диапазон значений предпочтения – от 0 до 255, причем наименьшее значение обозначает наиболее предпочтительный маршрут. По этой таблице можно понять, что `gated` предпочитает маршруты, полученные по OSPF, тем же маршрутам, полученным от BGP.

Таблица 7.1. Значения предпочтения по умолчанию

Тип маршрута	Предпочтение по умолчанию
Прямой маршрут	0
OSPF	10
IS-IS Level 1	15
IS-IS Level 2	18
Внутренний маршрут по умолчанию	20
Перенаправление ICMP	30
Маршрут, полученный из сокета маршрутов	40
Статический маршрут	60
Маршруты SLSP	70
RIP	100
Маршруты интерфейса точка-точка	110
Маршруты, пролегающие через неработающий интерфейс	120
Объединенные и порожденные маршруты	130
Маршруты OSPF ASE	150
BGP	170
EGP	200

Значение предпочтения может фигурировать в целом ряде операторов настройки. Оно может использоваться для расстановки приоритетов маршрутов, полученных через различные сетевые интерфейсы, от различных протоколов маршрутизации либо от различных внешних шлюзов. Значения предпочтения не передаются и не изменяются протоколами, они встречаются только в файле настройки. В следующем разделе мы изучим файл настройки `gated` (*/etc/gated.conf*) и хранимые в нем команды настройки.

Настройка gated

Получить `gated` можно по адресу *http://www.gated.org*. В приложении В содержится информация о том, как скопировать и собрать пакет. В данном разделе мы будем работать с `gated` release 3.6, версией `gated`, доступной в настоящее время без каких-либо ограничений. Существуют и другие версии `gated`, доступные членам консорциума Gated. Если вы планируете создавать продукты, основанные на `gated`, либо проводить исследования протоколов при помощи `gated`, вам следует вступить в консорциум. Для целей этой книги отлично подойдет и версия 3.6.

`gated` читает свои настройки из файла */etc/gated.conf*. Команды настройки в этом файле напоминают код на языке C. Все операторы заканчиваются точкой с запятой, а связанные операторы группируются фигурными скобками. Такая структура позволяет быстрее разобраться во взаимосвязях фрагментов файла настройки, что очень важно, если выполняется настройка нескольких протоколов в одном файле. Помимо структуры языка, структура присуща и файлу */etc/gated.conf* в целом.

Различные операторы настройки, а также порядок, в котором они должны следовать, делят файл *gated.conf* на разделы: *операторы параметров*, *операторы интерфейсов*, *операторы определений*, *операторы индивидуальных* и *групповых протоколов*, *статические операторы*, *управляющие операторы* и *объединяющие операторы*. Нарушение порядка следования операторов приводит к возникновению ошибки в процессе разбора файла.

Есть еще два типа операторов, которые не входят в перечисленные категории. Это операторы инструкций и операторы трассировки. Такие операторы могут встречаться в любой точке файла *gated.conf* и не связаны напрямую с настройкой какого-либо протокола. Операторы инструкций предписывают определенное поведение модулю разбора файла настройки, а операторы трассировки позволяют управлять трассировкой из файла настройки.

В табл. 7.2 содержится сводка команд настройки `gated`. Для каждой команды указано имя, тип оператора и приведено краткое резюме ее функциональности. Полностью язык команд описан в приложении В.

Как видите, команд в языке настройки `gated` много. Язык содержит рычаги управления для ряда различных протоколов, плюс дополнительные команды настройки функциональности собственно демона `gated`. Читатели могут испытать легкое замешательство.

Таблица 7.2. Операторы настройки gated

Оператор	Тип	Назначение
%directory	инструкция	Указывает каталог, в котором расположены включаемые файлы
%include	инструкция	Включает содержимое указанного файла в *gated.conf*
Traceoptions	трассировка	Указывает, какие события необходимо отслеживать
Options	параметр	Задает параметры работы `gated`
Interfaces	интерфейс	Задает параметры интерфейсов
Autonomoussystem	определение	Задает номер АС
Routerid	определение	Задает исходный маршрутизатор для BGP или OSPF
Martians	определение	Задает недопустимые адреса пунктов назначения
Multicast	протокол	Задает параметры протокола групповой передачи
Snmp	протокол	Включает отчетность по SNMP
Rip	протокол	Включает RIP
Isis	протокол	Включает протокол IS-IS
kernel	протокол	Задает параметры взаимодействия с ядром системы
ospf	протокол	Включает протокол OSPF
redirect	протокол	Удаляет маршруты, созданные сообщениями ICMP
egp	протокол	Включает EGP
bgp	протокол	Включает BGP
icmp	протокол	Выполняет настройку обработки пакетов ICMP в целом
pim	протокол	Включает протокол групповой передачи PIM
dvmrp	протокол	Включает протокол групповой передачи DVMRP
msdp	протокол	Включает протокол групповой передачи MSDP
static	статический	Определяет статические маршруты
import	управление	Указывает, какие маршруты приемлемы
export	управление	Указывает, какие маршруты афишируются
aggregate	объединение	Управляет объединением маршрутов
generate	объединение	Управляет созданием маршрута по умолчанию

Во избежание этого не пытайтесь понять до конца и в деталях все возможности gated. В вашей среде маршрутизации едва ли будет востребована вся функциональность пакета. Даже при создании шлюза на границе двух автономных систем вам, скорее всего, придется использовать лишь два протокола маршрутизации: один для внутренней маршрутизации, а второй – для внешней. Файл настройки должен содержать только команды, которые непосредственно относятся к востребованным протоколам. В процессе чтения этого раздела пропускайте информацию, которая не понадобится. К примеру, если не будет использоваться протокол BGP, зачем изучать оператор bgp? В то же время, более подробная информация по операторам содержится в приложении B. Учитывая сказанное, обратимся к примерам настройки.

Примеры настройки, gated.conf

Подробности, приводимые в приложении B, могут создать впечатление, будто настройка gated сложнее, чем на самом деле. Богатый язык команд gated может ввести в заблуждение, поскольку предназначен для работы со многими протоколами, а кроме того, позволяет достичь одних и тех же результатов различными способами. Но, как мы увидим из реалистичных примеров, настройка в каждом конкретном случае совсем необязательно оказывается сложной.

Основой для примеров станет рис. 7.4. Мы установили новый маршрутизатор, который обеспечивает нашей магистрали прямой доступ к сети Интернет, и приняли решение установить новые протоколы маршрутизации. Один из узлов будет принимать обновления RIP-2, внутренний шлюз – работать с протоколами RIP-2 и OSPF, а внешний шлюз – с протоколами OSPF и BGP.

Шлюз *limulus* связывает подсети 172.16.9.0 и 172.16.1.0. Узлам подсети 9 он представляется в качестве шлюза по умолчанию, поскольку является шлюзом во внешний мир. Для передачи маршрутов в подсеть 9 *limulus* применяет RIP-2. Подсети 1 шлюз *limulus* представляется в качестве шлюза в подсеть 9 – посредством OSPF.

Шлюз *chill* обеспечивает подсеть 1 доступом к сети Интернет через автономную систему 164. Поскольку шлюз *chill* выходит в сеть Интернет, он представляется в качестве шлюза по умолчанию всем остальным системам подсети 1 – посредством OSPF. Внешней автономной системе, посредством BGP, он представляется путем ко внутренним сетям, о которых узнает при помощи OSPF.

Итак, рассмотрим настройку маршрутизации для узла *minasi*, шлюза *limulus* и шлюза *chill*.

Настройка узла

Настройка маршрутизации узла очень проста. Оператор rip yes включает RIP, а для работы RIP, вообще говоря, больше ничего не требуется. В присутствии этого оператора протокол RIP должен работать в любой системе. Дополнительные операторы, заключенные в фигурные скобки, изменяют базо-

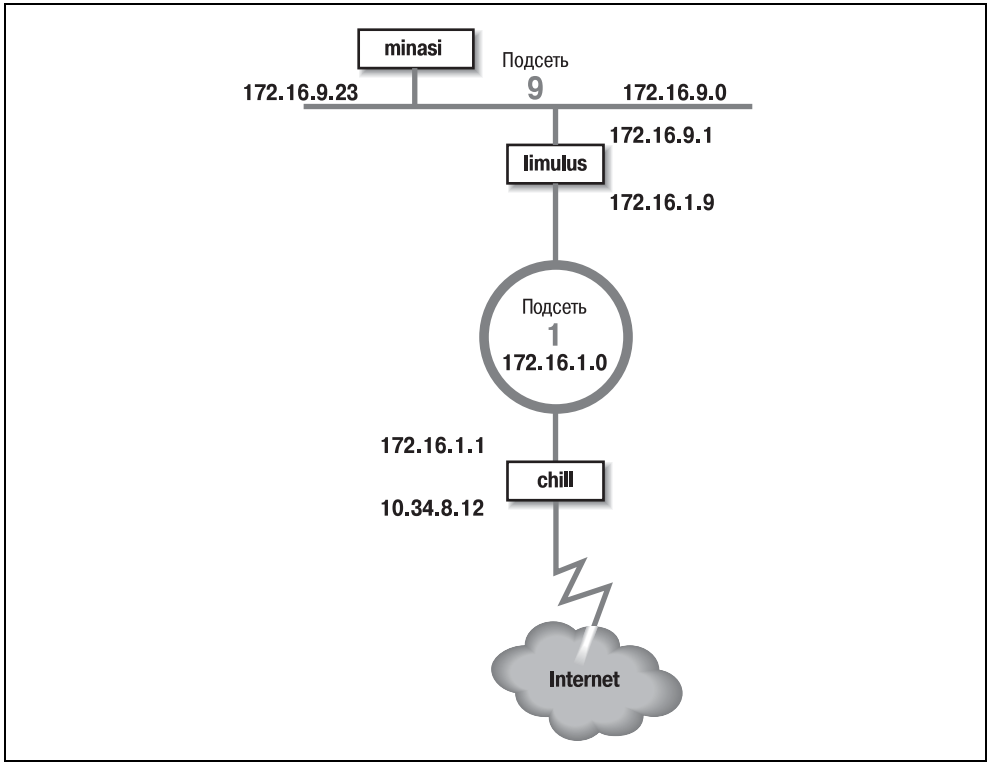

***Рис. 7.4.** Пример топологии маршрутизации*

вые настройки RIP. Мы используем ряд операторов, чтобы сделать приведенный ниже пример настройки RIP-2 для узла *minasi* чуть более интересным:

```
#
#  включить rip, не производить широковещательную
#  передачу обновлений,
#  принимать обновления RIP-2 на групповой адрес,
#  проверять подлинность обновлений.
#
rip yes {
        nobroadcast ;
        interface 172.16.9.23
             version 2
             multicast
             authentication simple "REAL stuff" ;
 } ;
```

Этот пример отражает общее строение операторов настройки, хранимых в файле *gated.conf*. Символ решетки (#) начинает строки комментариев.[1] Каж-

[1] Комментарии могут также заключаться между парами символов * и *\.

дый оператор заканчивается точкой с запятой. Связанные с оператором настройки уточнения могут быть многострочными и заключаются в фигурные скобки ({}). В этом примере уточнения nobroadcast и interface относятся непосредственно к оператору rip. Ключевые слова version, multicast и authentication входят в состав уточнения interface.

Ключевое слово nobroadcast запрещает узлу распространять собственные обновления RIP. По умолчанию принимается режим nobroadcast, если в системе всего один сетевой интерфейс, и broadcast, если интерфейсов несколько. Ключевое слово nobroadcast выполняет ту же функцию, что и ключ -q командной строки routed. gated, впрочем, способен на большее, чем routed, как видно из следующего уточнения.

Уточнение interface определяет параметры интерфейса для RIP. Параметры, связанные с уточнением в нашем примере, предписывают принимать обновления RIP-2 через групповой адрес RIP-2 интерфейса 172.16.9.23 и что подлинные обновления должны содержать пароль REAL^stuff. В RIP-2 механизм примитивной проверки подлинности построен на открытых текстовых паролях до 16 байтов длиной. Задача механизма – не защитить систему от злоумышленников, но предотвратить неожиданности в процессе настройки маршрутизаторов. Если какой-то пользователь по ошибке настроит свою систему на распространение данных RIP, маловероятно, что он также случайно укажет в своих настройках правильный пароль. Чтобы воспользоваться серьезным механизмом проверки подлинности – алгоритмом вычисления криптографической контрольной суммы MD5 (Message Digest 5) – следует указать значение md5 в уточнении authentication.

Настройки внутренних шлюзов

Настройки шлюзов сложнее, чем настройки рядового узла. Шлюзы всегда имеют несколько интерфейсов и время от времени работают с целыми наборами протоколов маршрутизации. Первый пример настройки связан с внутренним шлюзом, связующим подсеть 9 и несущую магистраль, подсеть 1. В подсети 9 шлюз использует RIP-2, чтобы сообщать о маршрутах Unix-системам. В подсети 1 – OSPF, чтобы обмениваться маршрутами с другими шлюзами. Вот настройки шлюза *limulus*:

```
#  Подсеть 9 не блокируется по времени
interfaces {
    interface 172.16.9.1 passive ;
} ;
# Идентификатор маршрутизатора OSPF
routerid 172.16.1.9 ;
# Включаем RIP-2; передаем OSPF-маршруты
# в подсеть 9, стоимость маршрутов - 5.
rip yes {
    broadcast ;
    defaultmetric 5 ;
    interface 172.16.9.1
         version 2
```

```
          multicast
          authentication simple "REAL stuff" ;
    } ;
    # Включаем OSPF; подсеть 1 является магистральной областью;
    # парольная проверка подлинности.
    ospf yes {
         backbone {
              interface 172.16.1.9 {
                  priority 5 ;
                  auth simple "It'sREAL" ;
                  } ;
              } ;
    } ;
```

Оператор `interfaces` определяет характеристики маршрутизации для сетевых интерфейсов. Ключевое слово `passive` в уточнении `interface` используется в данном случае (как и ранее) для создания статического маршрута, который не будет удален из таблицы маршрутизации. Постоянный маршрут в данной ситуации пролегает через напрямую подключенный сетевой интерфейс. Обычно, если по мнению `gated` работоспособность интерфейса нарушена, демон увеличивает стоимость интерфейса, присваивая ему высокое значение предпочтения (120), чтобы сократить вероятность маршрутизации данных через неработоспособный интерфейс. `gated` начинает считать интерфейс неработоспособным, если не получает обновлений маршрутизации через этот интерфейс. Мы не хотим, чтобы по вине `gated` интерфейс 172.16.9.1 был снижен в категории, даже если демон уверовал в существование проблем: наш маршрутизатор является единственным путем в подсеть 9. Именно поэтому настройки содержат уточнение `interface 172.16.9.1 passive`.

Оператор `routerid` определяет идентификатор маршрутизатора для протокола OSPF. Если идентификатор явным образом не задан в файле настройки, `gated` использует в качестве адреса-идентификатора маршрутизатора адрес первого из найденных интерфейсов. В данном случае мы указываем в качестве идентификатора адрес интерфейса, который действительно говорит на языке OSPF.

В предшествующем примере мы обсудили все уточнения оператора `rip`, за исключением одного `defaultmetric`. Уточнение `defaultmetric` определяет метрику RIP, используемую при распространении маршрутов, полученных от других протоколов маршрутизации. Данный шлюз работает как с OSPF, так и с RIP-2. Мы хотим передавать маршруты, полученные от OSPF, клиентам RIP, а для этого требуется метрика. Мы выбрали метрику RIP, равную 5. В отсутствие уточнения `defaultmetric` маршруты, полученные по протоколу OSPF, не передаются RIP-клиентам.[1] Данный оператор жизненно необходим для нашей сети.

Оператор `ospf yes` включает OSPF. Первое уточнение в этом операторе – `backbone`. Оно показывает, что маршрутизатор входит в состав магистральной

[1] Строго говоря, не совсем так. Маршруты афишируются со стоимостью 16, которая обозначает недостижимые пункты назначения.

области OSPF. Каждый оператор `ospf yes` должен включать, по меньшей мере, одно уточнение `area`. Уточнение `area` определяет конкретную область, например `area 2`, но, по меньшей мере, один маршрутизатор должен существовать в магистральной области. И хотя магистраль OSPF – это область 0, ее нельзя указать как `area 0`; для этой цели следует воспользоваться ключевым словом `backbone`. В нашей сети магистралью является подсеть 1, и все связанные с ней маршрутизаторы принадлежат магистральной области. Один маршрутизатор может принадлежать нескольким областям, и для каждой области могут быть определены собственные параметры. Обратите внимание на группировку уточнений вложенными парами фигурных скобок. Оставшиеся уточнения файла настройки связаны непосредственно с ключевым словом `backbone`.

Уточнение `interface` определяет интерфейс, связывающий маршрутизатор с магистральной областью. Здесь мы наблюдаем два дополнительных уточнения, `priority` и `auth`.

Уточнение `priority 5 ;` указывает приоритет данного маршрутизатора в процессе выбора магистралью отмеченного маршрутизатора. Чем больше номер приоритета, тем менее вероятно, что данный маршрутизатор будет выбран в качестве отмеченного. Расстановка приоритетов способствует выбору наиболее подходящего маршрутизатора.

Уточнение `auth simple "It'sREAL" ;` предписывает использовать простую парольную проверку подлинности в магистральной зоне и содержит текстовый пароль. В Gated 3.6 проверка подлинности сводится к трем вариантам: `none`, `simple` и `md5`. `none` означает отсутствие проверки подлинности. `simple` означает, что должен быть указан верный пароль из восьми символов, в противном случае обновление будет отвергнуто. Парольная проверка подлинности защищает лишь от случайностей, она не предназначена для защиты от злоумышленников. `md5` предписывает выполнять проверку подлинности по алгоритму MD5.

Настройка внешнего шлюза

Самое сложное – настройка шлюза *chill*, работающего с OSPF и BGP. Вот файл настройки шлюза *chill*:

```
# Определяем номер нашей AC для BGP
autonomoussystem 249;

# Идентификатор маршрутизатора OSPF
routerid 172.16.1.1;

# Отключаем RIP
rip no;

# Включаем BGP
bgp yes {
   group type external peeras 164 {
        peer 10.6.0.103 ;
        peer 10.20.0.72 ;
```

```
        };
};

# Включаем OSPF; подсеть 1 является магистральной областью;
# парольная проверка подлинности.
ospf yes {
     backbone {
         interface 172.16.1.1 {
              priority 10 ;
              auth simple "It'sREAL" ;
              } ;
          } ;
};

# Распространять маршруты, полученные от OSPF,
# выполнять маршрутизацию в сеть с прямым доступом
# через BGP в AS 164
export proto bgp as 164 {
     proto direct ;
     proto ospf ;
};

# Распространять маршруты, полученные от BGP из
# AS 164, в области OSPF.
export proto ospfase type 2  {
     proto bgp autonomoussystem 164  {
          all ;
          };
};
```

Такие настройки включают BGP и OSPF и устанавливают определенные параметры того и другого протокола. BGP для работы необходим номер АС, который равен 249 для *books-net*. OSPF необходим адрес-идентификатор маршрутизатора. Последний мы установили в адрес интерфейса маршрутизатора, который работает с OSPF. Номер АС и идентификатор маршрутизатора встречаются раньше прочих настроек, поскольку `autonomoussystem` и `routerid` – операторы определений, которые должны предшествовать первому оператору протокола. Различные типы операторов мы рассматривали выше в табл. 7.2.

Первый оператор протокола – тот, что отключает RIP. Мы не собираемся работать с RIP, но в `gated` RIP по умолчанию включается. Следовательно, его необходимо отключить явным образом, при помощи оператора `rip no;`.

Включение BGP выполняется оператором `bgp yes`, который содержит несколько дополнительных параметров BGP. Уточнение `group` устанавливает параметры для всех BGP-равных узлов группы и определяет тип создаваемого соединения BGP. Пример ориентирован на классическое соединение внешнего протокола маршрутизации, а внешняя автономная система, к которой мы подключаемся, имеет номер АС 164. `gated` позволяет создавать сеансы BGP пяти различных типов, но лишь один из них, тип `external`, используется для прямого обмена данными со внешней автономной системой. Еще четыре

типа групп используются для внутреннего BGP (*internal* BGP, IBGP).[1] IBGP – это просто обозначение протокола BGP, когда он применяется для распространения информации маршрутизации в пределах автономной системы. В нашем примере протокол применяется для передачи информации между автономными системами.

Соседи BGP, от которых принимаются обновления, обозначены уточнениями `peer`. Каждый *равный* (*peer*) является членом группы. Все параметры группы, такие как номер АС, применяются ко всем системам группы. Чтобы разрешить прием обновлений от любой системы АС 164, воспользуйтесь ключевым словом `allow` вместо списка равных.

Протокол OSPF включается оператором `ospf yes`. Настройка OSPF на этом маршрутизаторе ничем не отличается от настройки любого из маршрутизаторов магистральной области. Единственное, что изменилось относительно предыдущего примера, – число приоритета. Поскольку на данный маршрут ложится особенно большая нагрузка, мы решили сделать его маршрутизатор менее интересным для процесса выбора обозначенного маршрутизатора.

Оператор `export` управляет маршрутами, которые `gated` передает другим маршрутизаторам. Первый оператор `export` предписывает `gated` использовать BGP (`proto bgp`) для передачи сведений в автономную систему 164 (`as 164`) обо всех напрямую подключенных сетях (`proto direct`) и маршрутах, полученных от OSPF (`proto ospf`). Обратите внимание, что в этом операторе указан отнюдь не номер АС для *books-net*, а номер АС внешней системы. Первая строка оператора `export` определяет адресатов информации. Уточнения `proto` в фигурных скобках определяют состав передаваемой информации.

Второй оператор `export` распространяет маршруты, полученные от внешней автономной системы. Маршруты принимаются по протоколу BGP и передаются по протоколу OSPF. Поскольку эти маршруты получены от внешней автономной системы, они афишируются в качестве ASE-маршрутов (*autonomous system external*, внешние для автономной системы). Поэтому оператор `export` предписывает использовать для распространения маршрутов протокол `ospfase`. Параметр `type 2` задает тип распространяемых внешних маршрутов. `gated` поддерживает два типа. Маршруты типа 2 – это маршруты, полученные по протоколу внешних шлюзов, не имеющего метрики маршрутизации, совместимой с метрикой OSPF. Эти маршруты афишируются со стоимостью сообщения с пограничным маршрутизатором. В данном случае маршруты афишируются с OSPF-стоимостью сообщения со шлюзом *chill*. Маршруты типа 1 – это маршруты, полученные от внешнего протокола, имеющего метрику, сравнимую с метрикой OSPF. В этом случае при распространении маршрутов метрика внешнего протокола прибавляется к стоимости сообщения с граничным маршрутизатором.

Источником маршрутов, распространяемых по второму оператору `export`, является соединение BGP (`proto bgp`) с автономной системой 164 (`autonomous-`

[1] Все типы групп описаны в приложении В.

system 164). Уточнение proto может дополняться *фильтром маршрутов*. Фильтр маршрутов используется для отбора маршрутов из конкретного источника. Фильтр может перечислять сети и связанные с ними маски для отбора конкретных направлений. В нашем примере ключевое слово all предписывает выбирать все маршруты, полученные по протоколу BGP, что в действительности является умолчанием. Таким образом, мы могли бы опустить ключевое слово all. Однако вреда в его присутствии нет, зато оно четко документирует наши намерения.

Все маршруты, полученные от внешних автономных систем, могут привести к созданию очень большой таблицы маршрутизации. Отдельные маршруты полезны, если существует несколько пограничных маршрутизаторов, реализующих сообщение с внешним миром. Однако если пограничный маршрутизатор всего один, может оказаться, что достаточно маршрута по умолчанию. Чтобы экспортировать маршрут по умолчанию, добавьте оператор options gendefault ; в начало файла настройки.[1] Такой оператор предписывает gated генерировать маршрут по умолчанию, если происходит сообщение с равными BGP-соседями. Кроме того, замените второй оператор export в файле примера на следующий:

```
# Афишировать маршрут по умолчанию при сообщении
# с BGP-равными.
export proto ospfase type 2  {
     proto default ;
};
```

Данный оператор export предписывает gated афишировать пограничный маршрутизатор в качестве шлюза по умолчанию, но только при наличии активных соединений с внешней системой.

Как можно видеть из приведенных примеров, файлы *gated.conf* имеют небольшой размер и легко читаются. Если вам необходимо работать с протоколом маршрутизации на своем компьютере, воспользуйтесь gated. gated — это единое программное обеспечение и единый язык настройки для всех ваших узлов, внутренних и внешних шлюзов.

Тестирование настройки

Проверьте файл настройки перед использованием; синтаксис настройки gated очень сложен, в нем легко ошибиться. Сохраните настройки в тестовом файле, проверьте их, а затем перенесите в файл */etc/gated.conf*. Ниже мы приведем пример того, как это можно сделать.

Предположим, файл настройки называется *test.conf*, и мы его уже создали. Для проверки воспользуемся ключами командной строки -f и -c:

```
% gated -c -f test.conf trace.test
```

[1] Альтернативным способом создания маршрута по умолчанию является оператор generate. За подробностями обращайтесь к приложению B.

Ключ `-f` предписывает `gated` читать настройки из указанного файла (в данном случае – *test.conf*), а не из файла */etc/gated.conf*. Ключ `-c` предписывает `gated` проверить файл настройки на наличие ошибок синтаксиса. Завершив чтение файла, `gated` прекращает работу: таблица маршрутизации при этом не изменяется. Ключ `-c` включает трассировку, поэтому указывайте имя файла для данных трассировки, если не желаете, чтобы они отображались на терминале. В данном случае мы указали имя файла трассировки – *trace.test*. Кроме того, ключ `-c` создает снимок состояния демона после чтения файла настройки и записывает образ в файл */usr/tmp/gated_dump*.[1] Для выполнения `gated` с ключом `-c` нет необходимости в статусе администратора системы, равно как и в принудительном завершении процесса `gated`.

Файлы образа и трассировки (*trace.test*) могут содержать дополнительные данные, включая сведения об ошибках. Убедившись, что настройки верны, поместите содержимое нового файла настройки (*test.conf*) в файл */etc/gated.conf* (для этого необходимы полномочия администратора).

Существует альтернативная программа – `gdc`, которая должна выполняться пользователем `root` либо с полномочиями суперпользователя. Она предоставляет функциональность для проверки и ввода в эксплуатацию новых настроек. `gdc` использует три различных файла настройки. Текущие настройки хранятся в */etc/gated.conf*. Предыдущий вариант настройки хранится в файле */etc/gated.conf–*, а «следующий» вариант настройки хранится в файле */etc/gated.conf+*, для которого как раз и проводится тестирование. Вот так `gdc` проверяет настройки:

```
# cp test.conf /etc/gated.conf+
# gdc checknew
configuration file /etc/gated.conf+ checks out okay
# gdc newconf
# gdc restart
gated not currently running
gdc: /etc/gated was started
```

В данном примере настройки для проверки копируются в */etc/gated.conf+* и проверяются посредством команды `gdc checknew`. Если в файле найдены ошибки синтаксиса, отображается предупреждение, а подробная диагностика записывается в файл */usr/tmp/gated_parse*. В данном примере ошибок нет, так что мы делаем файл-претендент текущим файлом настройки при помощи команды `gdc newconf`. Данная команда перемещает текущие настройки в файл *gated.conf–*, а новые настройки (*gated.conf+*) – в файл текущих. Команда `gdc restart` – ее не было в нашем примере – принудительно завершает `gated`, если демон запущен, и запускает новый экземпляр `gated` с новыми настройками.

---

[1] */usr/tmp* – каталог по умолчанию для записи этого файла и файла *gated_parse*, описанного ниже; однако некоторые системы сохраняют эти файлы в каталоге */var/tmp*.

Запуск gated при загрузке

Как и многие другие программы маршрутизации, демон gated следует включить в процесс загрузки системы. В некоторых системах код для запуска gated уже существует в загрузочных файлах. Если у вас не такая система, код придется создать вручную. Если в загрузочном файле уже существует код для запуска routed, замените его кодом для запуска gated. gated и routed не должны работать одновременно.

Наш воображаемый шлюз *crab* работает под управлением Solaris, и в файле */etc/init.d/inetinit* содержится код для запуска routed. Закомментируем соответствующие строки и добавим такие:

```
if [ -f /usr/sbin/gated -a -f /etc/gated.conf ]; then
     /usr/sbin/gated;       echo -n 'gated' > /dev/console
fi
```

Данный фрагмент кода предполагает, что gated установлен в каталог */usr/sbin* и что файл настройки называется */etc/gated.conf*. Код проверяет присутствие gated и существование файла настройки */etc/gated.conf*. Если существуют оба файла, выполняется запуск gated.

Наличие файла настройки проверяется потому, что демон gated обычно работает с файлом настройки. Если запустить gated и не указать файл настройки, демон проверит наличие в таблице маршрутизации маршрута по умолчанию. Будет использован маршрут по умолчанию, если он существует, и запущен протокол RIP в противном случае. Создавайте файл */etc/gated.conf*, даже если собираетесь работать только с RIP. Файл настройки документирует конфигурацию подсистемы маршрутизации и защищает от возможных изменений стандартных настроек демона в будущем.

Резюме

Маршрутизация – это связующее звено, объединяющее различные сети в сеть Интернет. Без маршрутизации сети не смогут обмениваться данными. Настройка маршрутизации – важная задача сетевого администратора.

Чтобы обмениваться данными по сетевому интерфейсу с напрямую подключенной сетью, необходима примитивная маршрутизация. Такие маршруты можно встретить в таблице маршрутизации – им соответствуют записи без флага G (*gateway*, шлюз). В некоторых системах минимально необходимые маршруты создаются посредством команды ifconfig в процессе настройки интерфейса. В системах Linux маршрут через интерфейс должен быть явно создан командой route.

Команда route используется для создания статических таблиц маршрутизации. Статическая маршрутизация – это маршрутизация, которая требует внимания и сопровождения со стороны сетевого администратора. Маршруты добавляются и удаляются из таблицы маршрутизации при помощи команды route. Чаще всего статическая маршрутизация применяется для создания маршрутов по умолчанию.