В помощь сетевым администраторам

Solaris 8

Руководство администратора

СИМВОЛ®

O'REILLY®

Пол Уоттерс

Solaris 8

Administrator's Guide

Paul Watters

O'REILLY®

Solaris 8

Руководство администратора

Пол Уоттерс

Санкт-Петербург—Москва
2003

Пол Уоттерс

Solaris 8. Руководство администратора

Перевод М. Зислиса

Главный редактор	*А. Галунов*
Зав. редакцией	*Н. Макарова*
Научный редактор	*С. Клименков*
Редактор	*А. Лосев*
Корректура	*С. Беляева*
Верстка	*Н. Гриценко*

Уоттерс П.

Solaris 8. Руководство администратора. – Пер. с англ. – СПб: Символ-Плюс, 2003. – 336 с., ил.

ISBN 5-93286-052-9

В книге «Solaris 8. Руководство администратора» подробно освещаются аспекты применения Solaris в качестве сервера, обеспечивающего работу сетевых служб всех уровней. Учитывая новую волну интереса к Solaris как к корпоративной сетевой ОС, автор уделяет основное внимание службам уровня предприятий и предлагает концептуальный, сложный материал, отсутствующий в прочих руководствах. Издание ориентировано на опытных сетевых администраторов, нуждающихся в предметном руководстве по сетевым функциям Solaris; рассмотрен процесс установки системы на платформах Intel и Sparc.

Материал книги, содержащий описание современных сетевых технологий, применяемых на практике, поможет освоить использование Solaris в качестве файлового сервера, сервера приложений и сервера баз данных и приобрести навыки, необходимые для понимания влияния новых служб и новых программных продуктов на существующие серверные системы.

ISBN 5-93286-052-9
ISBN 0-596-00073-1 (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 16.04.2003. Формат 70x100 1/16 . Печать офсетная.
Объем 21 печ. л. Тираж 2000 экз. Заказ N
Отпечатано с диапозитивов в Академической типографии «Наука» РАН
199034, Санкт-Петербург, 9 линия, 12.

Оглавление

Предисловие

Электронная коммерция навсегда изменила мир компьютеров. Приземленные задачи, связанные с созданием сетей, установкой систем, обеспечением удаленного доступа, в мировом масштабе вызывают гораздо больший интерес, чем проявляемый отдельными пользователями на местах. Клиент может получать доступ к приложению электронной коммерции, находясь на другом конце города и даже на другом конце света.

Пришло время, когда привычные витрины заменяются виртуальными, так что действия системных администраторов и архитекторов теперь подвергаются более тщательному анализу. Представьте себе, что реальный магазин вынужден закрыть двери для покупателей по причине аппаратного сбоя! Чтобы предоставить тот уровень обслуживания, который ожидается и даже требуется в эпоху сети Интернет, виртуальные магазины должны работать двадцать четыре часа в сутки, семь дней в неделю, по пятьдесят две недели в году. Меняются потребности покупателей, а значит, меняются потребности разработчиков и обслуживающего персонала систем: архитектурные решения ставятся под сомнение, а любое изменение бюджета должно быть оправдано в контексте существующих доходов. Вдумчивые системные администраторы системы Solaris™ теперь считают обязанностью вникать в деловую активность своих фирм, поскольку их действия при построении систем высокой доступности могут коренным образом повлиять на успешность любого предприятия.

Для кого эта книга

Эта книга является руководством для администраторов, перед которыми стоят задачи настройки сетевых служб Solaris, доступных далеко за пределами локальных сетей. Многие из принципов и примеров, приводимых в книге, применимы как в глобальных, так и в локальных сетях, хотя требования к безопасности и уровню обслуживания в приложениях электронной коммерции, как правило, оставляют в тени вопросы, связанные с локальными службами. Тем не менее в электронной коммерции особенно важна грамотная поддержка и сопровож-

дение локальных сетевых служб, от потенциала масштабирования которых может зависеть решение глобальных задач.

Методы, процедуры и техники, описанные в книге, не привязаны жестко к конкретному изданию системы Solaris; при этом отдельные команды, возможности и параметры могут варьироваться. Помимо прочего, некоторые из недавно появившихся служб доступны только в системе Solaris версии 8 и более поздних.

Чтение этой книги принесет пользу любому, кто имеет дело с сетевыми службами Solaris, будь то системный администратор, архитектор, инженер или разработчик.

Обзор

Настоящая книга рассказывает о построении сетей на основе служб системы Solaris. Сначала мы изучим основы сетевого взаимодействия и планирования подсетей и, в частности, рассмотрим процесс установки и настройки служб для отдельной вычислительной системы. От практических вопросов межсетевой маршрутизации мы перейдем к распространенным службам имен, реализующим управление различными ресурсами (узлами, пользователями и т. д.) в масштабах целых сетей. Речь пойдет об управлении этими ресурсами на уровне отдельных вычислительных систем, после чего будут подробно освещены две широко применяемые системы доступа к файлам (и связанные с ними протоколы). Дальше нас ждет обсуждение методик управления данными и вопросов, связанных с оценкой и измерением емкостных потребностей. Я представлю анализ способов обеспечения безопасности вычислительных систем и сетей посредством инструментов операционной системы Solaris и решений независимых разработчиков. И наконец, я расскажу о разработке системы электронной коммерции с применением уровней приложений и служб архитектуры Sun ONE™ (Open Network Environment, открытое сетевое окружение).

В главе 1 «Сеть – это компьютер» представлено будущее сетевых информационных систем глазами компании Sun – будущее, основанное на среде ONE и применении систем SunOS™ и Solaris в качестве базовых технологий, способных обеспечить солидный фундамент для деловой активности. На конкретном примере рассмотрен анализ требований и спецификаций для гипотетической системы электронной коммерции, построенной на базе Solaris, и того, каким образом различные уровни среды ONE могут настраиваться в целях эффективного решения поставленных задач.

В основе существования современных сетевых служб лежит семейство протоколов TCP/IP и сеть Интернет. Глава 2 «Создание сетей Solaris» посвящена краеугольным принципам работы сетей (в частности формату IP-адресов, разрешению имен узлов и адресов), а также вопросам построения сетей Solaris. Мы обсудим поддержку сетевых устройств, реа-

лизованную в системе Solaris, рассмотрим установку и настройку основных сетевых служб TCP/IP (включая демон интернет-служб *inetd*). Чтобы продемонстрировать основные принципы работы протокола TCP (Transmission Control Protocol) и родственных ему, мы разберем содержимое реальных пакетов, которыми обмениваются узлы под управлением Solaris.

В главе 3 «Установка системы Solaris» даны пошаговые инструкции по установке и выбору значений основных параметров сетевых настроек для платформ Solaris Intel и Solaris SPARC. Существует три метода установки системы Solaris: из командной строки (текстовый интерфейс), диалоговый (на основе меню) и веб-старт (на основе Java). Мы изучим последний. Кроме того, затронем задачи планирования, выполнение которых необходимо для выбора корректных параметров настройки сети. Вы узнаете о распространенных ошибках при установке системы Solaris, а также о методах разрешения возникающих проблем.

Учитывая сложность задач, связанных с настройкой единственной вычислительной системы, планирование и настройка целой сети могут выглядеть устрашающе. В главе 4 «Настройка сетей» рассказывается о настройке сети из машин под управлением Solaris, описываются настройка сетевых интерфейсов в различных конфигурациях, статическая и динамическая маршрутизация, а также способы диагностирования и разрешения сетевых проблем. В дополнение к этому материалу приводятся примеры инициализации устройств и проверки интерфейсов.

Когда сеть начинает расти, и ответственность за решение различных задач делится между системами, особую важность приобретает установка по меньшей мере одной системы каталогов, которая позволит находить узлы сети. Для доменов сети Интернет могут потребоваться дополнительные службы имен. В главе 5 «Службы имен» рассказывается о важности распределенных служб имен, позволяющих находить машины, пользователей и различные виды сетевых ресурсов, и читатели знакомятся с процессом настройки службы доменных имен (DNS) системы Solaris. DNS позволяет преобразовывать IP-адреса в имена, удобные для использования людьми, и обратно. Сетевая информационная служба (Network Information System, NIS/NIS+), разработанная Sun, позволяет сделать шаг вперед от традиционной системы DNS в плане поддержки иерархических пространств имен. Система NIS/NIS+, которую мы рассмотрим в подробностях, является универсальным решением по управлению сетевыми ресурсами, позволяющим проводить как авторизацию доступа к ресурсам, так и идентификацию, то есть проверку подлинности, во взаимодействиях клиентов и серверов. И наконец, речь пойдет о более современном облегченном протоколе доступа к каталогам (Lightweight Directory Access Protocol, LDAP) и его реализации на платформе Solaris.

В главе 6 «Системное администрирование» рассказывается об управлении пользователями и ресурсами в масштабе одной вычислительной

системы, описывается применение графического интерфейса *admintool* для удаления, добавления и изменения учетных записей пользователей, пакетов программного обеспечения, последовательных портов и принтеров. Кроме того, упоминаются инструменты командной строки, позволяющие выполнять те же действия. По мере появления в системе новых пользователей и ресурсов задача их эффективного сопровождения и обеспечения непрерывности действия служб становится все более важной. Читатели узнают о том, как управлять доступом пользователей в диалоговом режиме с помощью инструментов пользовательских квот, входящих в состав Solaris. Наконец, будет рассмотрена установка и настройка таких сетевых служб, как почтовый агент *sendmail*.

В сетях довольно широко применяется клиент-серверная архитектура, в которой сервер обеспечивает клиентам надежный доступ к файлам. В главе 7 «Файловые серверы» система Solaris представлена в пользующейся спросом роли сетевого файлового сервера. Solaris в качестве кроссплатформенного сервера обладает рядом привлекательных особенностей, таких как встроенная поддержка очень длинных имен файлов и впечатляющая производительность системы ввода-вывода. Читатели научатся применять Solaris для организации файловых серверов с помощью протокола сетевой файловой системы NFS (Network File System) и программного пакета Samba. Несмотря на великолепные показатели производительности NFS, в гетерогенных сетях предпочтительно применение Samba, поскольку этот пакет позволяет организовать совместный доступ к дисковым разделам и принтерам всех клиентов, понимающих протокол SMB, в частности операционных систем Microsoft Windows и Mac OS. Применение Samba позволяет решениям на базе системы Solaris полностью заменить существующие серверы Windows NT: сервер под управлением Solaris может выступать в роли первичного контроллера домена (Primary Domain Controller, PDC) для существующих сетей NT.

Перечисленные главы посвящены разнообразным способам хранения данных и работы с ними на системах под управлением Solaris в случае пользовательских приложений и системных служб. Следующее утверждение может показаться вполне очевидным, но основной причиной создания многих компьютерных систем является необходимость *управления* данными. Так, главной задачей систем управления базами данных является запись изначально достоверных данных, их безопасное хранение и обеспечение надежного доступа. В конечном итоге хранилище достоверных данных бесполезно, если нет возможности получить к нему доступ, а безопасное хранилище данных, не гарантирующее корректность выполняемых для таблиц транзакций, и вовсе полная глупость. В главе 8 «Управление данными» рассказывается об основах грамотного управления данными и об инструментах, предоставляемых системой Solaris для воплощения этих идей на практике. Мы изучим реальные примеры, связанные с хранением данных, и в

частности создание резервных копий и восстановление из них. Потерю или повреждение информации можно предотвратить, если придерживаться некоторых решений (например, зеркалирование и расслоение), которые реализуются с помощью специальных аппаратных средств, программного обеспечения сторонних разработчиков либо набора простых сценариев и ленточного накопителя. Вы узнаете о распространенных вариантах хранения данных, таких как жесткие диски, приводы CD-ROM и Zip/Jaz.

Глава 9 «Сетевая безопасность» – это курс практической безопасности, который начинается с изучения средств анализа сетевой безопасности, пакетов SAINT и WebSAINT. Читатели научатся строить брандмауэры с фильтрацией пакетов на базе маршрутизаторов и применять проверку подлинности клиентов по протоколу Kerberos 5, реализация которого входит в состав системы Solaris 8. Глава завершается примерами управления безопасностью транспортных уровней сети с помощью протокола SSL и удаленным доступом с помощью интерпретатора SSH.

В главе 10 «Сетевые информационные системы» рассказывается о создании информационных систем с веб-интерфейсом на базе сети интранет с трехзвенной архитектурой. Отметим, что хотя задача обучить читателей программированию на языке Java™ и не ставилась, но в этой главе приводится полностью рабочий пример создания сервера приложений на базе Solaris (Apache/JServ), установки сервера баз данных (Postgres), а также установки и тестирования конкретного приложения. Для понимания данного примера обязательно понадобятся навыки, приобретенные при чтении предшествующих глав. В качестве архитектуры для сетевого распределения приложений мы рассмотрим посредник объектных запросов VisiBroker.

Соглашения, принятые в книге

В этой книге используются следующие типографские обозначения:

Курсивом

оформлены команды, имена файлов настройки, каталогов и демонов; кроме того, курсив применяется для смыслового выделения и при первом использовании термина.

`Моноширинным шрифтом`

оформлены MAC- и IP-адреса.

Комментарии и вопросы

Пожалуйста, направляйте вопросы и комментарии, связанные с книгой, издательству:

O'Reilly & Associates, Inc.

1005 Gravenstein Highway North

Sebastopol, CA 95472
(800) 998-9938 (в США или Канаде)
(707) 829-0515 (международный/местный телефон)
(707) 829-0104 (факс)

У данной книги существует веб-сайт, где представлены примеры, замеченные ошибки и другая дополнительная информация. Страница доступна по адресу:

http://www.oreilly.com/catalog/solaris8

Чтобы задать технический вопрос или прокомментировать содержание книги, воспользуйтесь адресом:

bookquestions@oreilly.com

Информация о других книгах, конференциях, центрах ресурсов (Resource Centers) и сети сайтов O'Reilly (O'Reilly Network) доступна на вебсайте O'Reilly по адресу:

http://www.oreilly.com

Благодарности

Мне хотелось бы упомянуть техническую поддержку, которую обеспечили замечательные ребята из Sun Microsystems, в частности Кейт Уотсон (Keith Watson), Равиндра Айер (Ravindra Iyer) и Дрю Ренн (Drew Wrenn). Многочисленные ценные предложения технических рецензентов также заслуживают благодарности. Спасибо, Экинвенд Зигмунд Уолтер-Джонсон третий (Akinwande Seigmund Walter-Johnson III), Эльза Ланкфорд (Elsa Lankford), Уильям Оссер (William Osser) и доктор Тим Гиббс (Tim Gibbs)! Хочется думать, что ваши усилия сделали книгу технически более выдержанной и упростили восприятие текста.

Что касается издательства O'Reilly, я хочу поблагодарить Джима Самзера (Jim Sumser), полного энтузиазма, терпеливого редактора, который помогал создавать эту книгу всеми доступными ему средствами. Спасибо, Джим, ты лучше всех! Кроме того, большое спасибо Робу Романо (Rob Romano) и Ханне Дайер (Hanna Dyer) за великолепные иллюстрации и Тиму О'Рейлли, парню, который является стержнем всего процесса.

В агентстве «Studio B» я говорю спасибо Нилу Залкинду (Neil Salkind) и Вики Хардинг (Vicki Harding), двум лучшим агентам в нашей части Техаса, а также Стэйси Бароун (Stacey Barone), Кристен Пикенс (Kristen Pickens), Крэйгу Уайли (Craig Wiley), Шерри и Дэвиду Рогельбергам (Sherry, David Rogelberg) за их отличную работу и терпение.

И, конечно же, спасибо моей супруге Майе и всей моей семье за то, что прощали мне ранние подъемы, работу допоздна и непрерывный стук клавиатуры.

4

Настройка сетей

Учитывая сложность задач, связанных с настройкой отдельного узла, планирование и настройка целой сети может выглядеть устрашающе. В этой главе вы узнаете о настройке сети из машин под управлением Solaris, о настройке сетевых интерфейсов в различных конфигурациях, о статической и динамической маршрутизации, а также о способах диагностирования и разрешения сетевых проблем. В дополнение к этому материалу приводятся примеры инициализации устройств и проверки интерфейсов.

Создание сетей и подсетей

Системы под управлением Solaris способны работать в изоляции от сетевой среды, но Solaris все-таки является операционной системой, ориентированной на работу в сети, и включает поддержку сетевого взаимодействия узлов локальной сети между собой и Интернетом, а именно:

- Поддержка устройств ethernet с одним, двумя или четырьмя интерфейсами
- Стандартизированная система именования сетевых устройств
- Поддержка широкого спектра сетевых устройств
- Настройка интерфейсов для работы по протоколам IPv4 и IPv6
- Маршрутизация по статическим и динамическим протоколам
- Диагностика и разрешение проблем, измерение производительности
- Фильтрация пакетов по всем TCP- и UDP-портам
- Передача данных по Ethernet и FDDI

- Поддержка сетей ATM (Asynchronous Transfer Mode), работающих в режиме асинхронной передачи данных

Произвольные сочетания этих возможностей упрощают создание сетей на основе Solaris и в особенности сетей, в которых системы под управлением Solaris играют ведущую роль в маршрутизации и фильтрации пакетов.

Типичная локальная сеть Solaris включает один или несколько серверов, которые предоставляют локальным клиентам доступ к сетевым службам. Клиентами могут быть другие системы Solaris, но с равной вероятностью – системы под управлением Linux, Microsoft Windows либо различных вариантов Unix. В некоторых сетях каждая из служб работает на специально отведенной под нее машине, что позволяет обеспечивать доступ к большинству служб в случаях, когда один из серверов прекращает работу. Такой вид разделения ролей серверов стал еще более эффективным с появлением системы E10000, которая позволяет создавать до 64 виртуальных серверов, работающих на одной машине. Если работа одного из доменов приостанавливается с целью выполнения задач техобслуживания, остальные домены продолжают работать.

Число и виды служб, которые могут работать в локальной сети Solaris, можно перечислять практически бесконечно, но в типичном варианте используются следующие:

Почтовый сервер	Сервер идентификации
Сервер USENET	Сервер управления ресурсами
UNIX-совместимый файловый сервер	Сервер удаленного доступа
PC-совместимый файловый сервер	Сервер RPC
Сервер резервного копирования	Сервер WWW
Сервер печати	Сервер каталогов

Эти виды служб в Solaris представлены следующими пакетами:

Sendmail	Kerberos
Inn	NIS+
NFS	Telnet и FTP
Samba	Демон RPC
Netbackup	Apache
Системы печати System V и BSD	LDAP

На рис. 4.1 приведен пример конфигурации сервера для сети класса C.

Когда функции сервера для локальной сети определены, необходимо решить ряд других вопросов, связанных, в частности, с выбором диапазонов IP-адресов для отдельных подсетей и IP-адресов отдельных узлов (это описано в главе 2). Современные сети, как правило, создают-

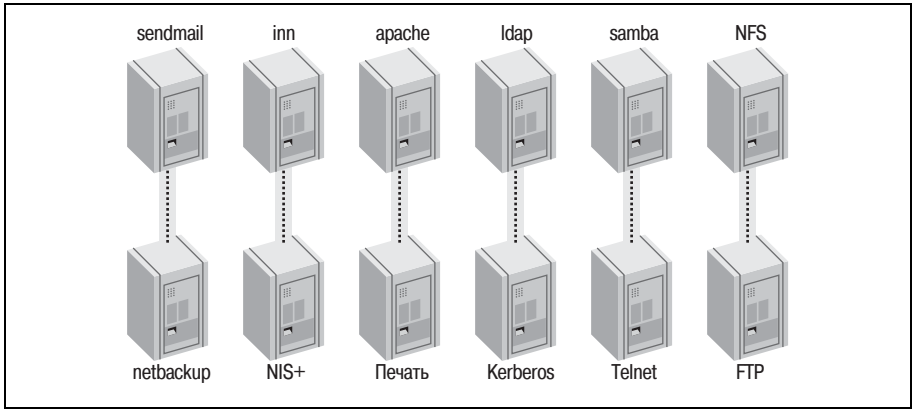

Рис. 4.1. *Пример конфигурации сервера Solaris для сети класса C*

ся на основе 10- и 100-мегабитных Ethernet-соединений; при этом узлы одной подсети связаны с единственным маршрутизатором посредством коммутатора или концентратора. На рис. 4.2 показана самостоятельная локальная сеть: узлы *chardon*, *blanc*, *riesling* и *semillon* соединены центральным коммутатором. Если все узлы подключены с помощью 100-мегабитного кабеля ethernet, весь обмен трафиком в этой сети происходит со скоростью 100 Мбит. Применение различных кабелей и скоростей передач может приводить к проблемам, и, поскольку в настоящее время большинство сетевых интерфейсов Solaris способны работать на скорости 100 Мбит, рекомендуется использовать эту скорость в качестве стандартной. В этой простой сети нет шлюзов, она не связана с другими сетями и с Интернетом и не нуждается в маршрутизаторе.

Если требуется подключение к другой сети, коммутатор может быть подключен к маршрутизатору, как показано на рис. 4.3.

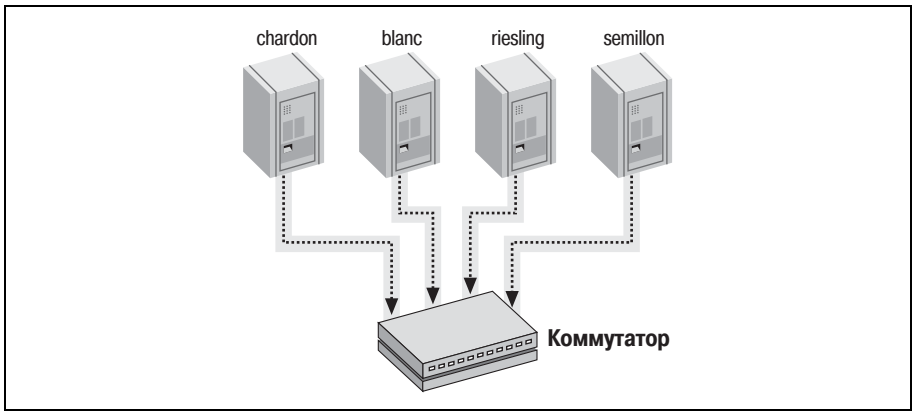

Рис. 4.2. *Простая сеть Solaris класса C, не связанная с сетью Интернет*

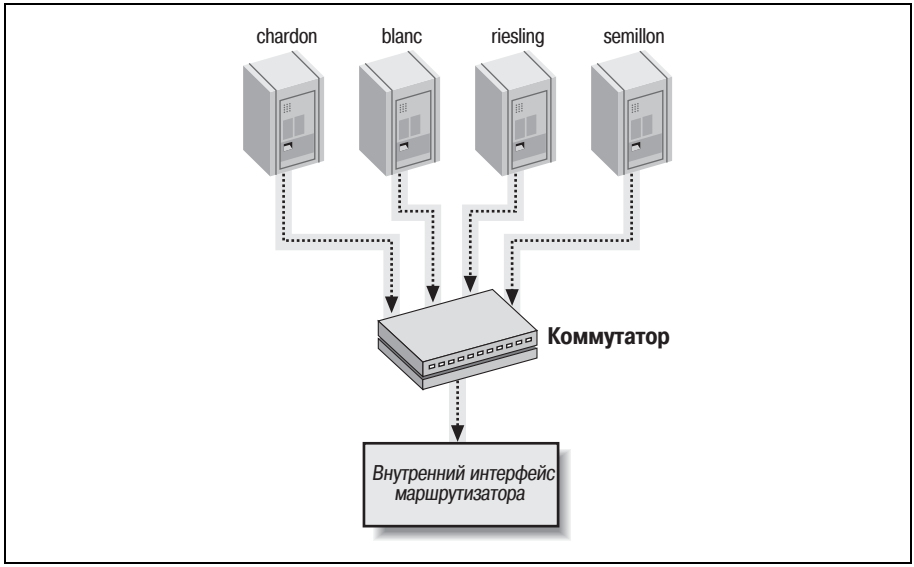

Рис. 4.3. *Простая сеть Solaris класса C, подключенная к другой сети*

В этом случае все пакеты от узлов *chardon*, *blanc*, *riesling* и *semillon* могут передаваться коммутатору и далее «внутреннему» интерфейсу маршрутизатора. Как вариант к одному из узлов, скажем, *chardon*, может быть подключен модем, с помощью которого устанавливается соединение с Интернетом. Узлы *blanc*, *riesling* и *semillon* могут получить прямой доступ к Интернету, не пользуясь доступом по протоколу Telnet на узел *chardon*. Для этого требуется зарегистрировать его в качестве шлюза. Коммутатор гарантирует доставку пакетов на нужный шлюз.

Кроме того, концентраторы и коммутаторы могут объединяться в цепочки с целью подключения отдаленных комнат, этажей и даже зданий к одной сети. Между маршрутизатором и наиболее удаленным клиентом должно быть не более трех транзитных участков, в противном случае количество конфликтов при передаче пакетов вырастет до неприемлемой величины.

Большинство сетевых площадок начинают существование с одной сети класса C. Постепенно число сетей класса C растет, и сети объединяются с помощью маршрутизаторов. Но прежде чем мы рассмотрим установку и настройку маршрутизатора, следует более пристально изучить настройку отдельных сетевых интерфейсов.

Настройка сетевых интерфейсов

Различные программы установки Solaris с радостью займутся настройкой существующих сетевых интерфейсов в процессе установки,

но существуют ситуации, когда требуется добавить новый интерфейс или изменить настройки существующих. А именно:

- Настройка существующего узла для работы в качестве маршрутизатора
- Перенос узла в другую подсеть
- Настройка распределения нагрузки между интерфейсами

Чтобы включить сетевой интерфейс в Solaris, может потребоваться выполнить несколько шагов:

- Установка драйверов устройств
- Перезагрузка системы с целью обновления настроек
- Назначение интерфейсу IP-адреса
- Выбор роли интерфейса (элемент маршрутизатора или элемент многосетевого узла)
- Создание для узла записи, отображающей IP-адрес в имя узла
- Настройка интерфейса на работу с проходящими через него данными

Драйверы устройств, как правило, хранятся в каталоге */kernel/drv* (или в каталоге, определяемым файлом */etc/system*) и перечислены в файле */etc/device_aliases*. Например, стандартному четырехпортовому интерфейсу от Sun соответствует драйвер */kernel/drv/qfe*, его псевдоним присутствует в файле */etc/device_aliases* (*qfe SUNW,qfe*). Следующая команда позволяет выполнить перезагрузку с целью обновления настроек:

```
bash-2.03# touch /reconfigure; init 6
```

Как вариант можно воспользоваться командой монитора OpenBoot PROM:

```
OK boot -r
```

IP-адрес назначается интерфейсом посредством создания hostname-файла, расположенного в каталоге */etc*. Для системы с единственным интерфейсом (скажем, */dev/eri0*), такой как Blade 100, файл носит имя *hostname.eri0*, где *eri0* – имя устройства, а цифра *0* определяет номер интерфейса.

Четырехпортовой Ethernet-карте (устройства */dev/qfe0*, */dev/qfe1*, */dev/qfe2* и */dev/qfe3*) соответствуют четыре файла с различными IP-адресами: *hostname.qfe0*, *hostname.qfe1*, *hostname.qfe2* и *hostname.qfe3*. Адреса могут идти по порядку и принадлежать одной сети (`192.64.18.1`, `192.64.18.2`, `192.64.18.3` и `192.64.18.4`), если узел является многоканальным (multi-homed), либо вразнобой (принадлежать разным сетям), если система является не многоканальным узлом, а маршрутизатором.

Многоканальный узел позволяет производить обмен данными только с локальной сетью (считая маршрутизатор этой сети), в то время как

маршрутизатор отвечает за передачу пакетов между сетями. Чтобы предотвратить маршрутизацию, следует создать на многоканальном узле пустой файл */etc/notrouter*. Кроме того, следует поместить IP-адрес маршрутизатора по умолчанию для локальной сети в файл */etc/defaultrouter*.

Можно создать записи для интерфейсов в файле */etc/hosts* либо записи в службе имен, использование которой определяется файлом */etc/nsswitch.conf*. Например, если необходимо создать отображение IP-адресов из файлов *hostname.qfe0*, *hostname.qfe1*, *hostname.qfe2* и *hostname.qfe3* в имена узлов *www1*, *www2*, *www3* и *www4*, файл */etc/hosts* будет содержать такой фрагмент:

```
bash-2.03# cat /etc/hosts
www1        192.64.18.1
www2        192.64.18.2
www3        192.64.18.3
www4        192.64.18.4
```

На рис. 4.4 показаны логические настройки четырехпортовой Ethernet-карты узла, на котором работают четыре независимых веб-сервера.

Если используется DNS (см. главу 5), в соответствующий файл зоны потребуется добавить такие записи:

```
www1    IN    A    192.64.18.1    ;webserver
www2    IN    A    192.64.18.2    ;webserver
www3    IN    A    192.64.18.3    ;webserver
www4    IN    A    192.64.18.4    ;webserver
```

Прежде чем интерфейс сможет передавать или получать IP-трафик, он должен быть настроен с помощью команды *ifconfig*. Когда интерфейс задействован, можно воспользоваться следующей командой *ifconfig* для перечисления активных интерфейсов:

```
bash-2.03# /usr/sbin/ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
```

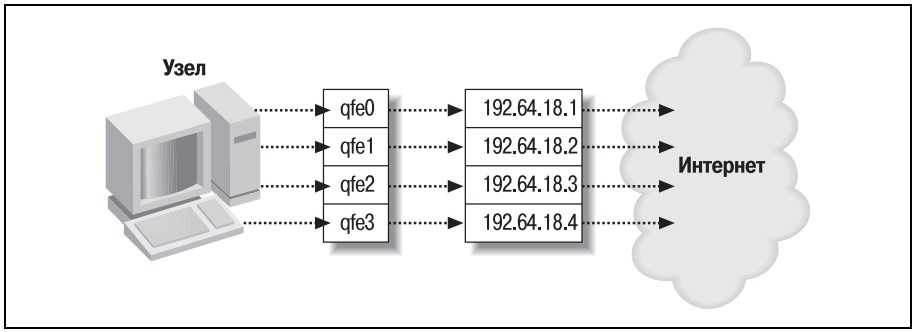

***Рис. 4.4.** Логические настройки четырехпортовой карты Ethernet*

```
eri0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 10.64.18.3 netmask ffffff00 broadcast 10.64.18.255
lo0: flags=2000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv6> mtu 8252 index 1
        inet6 ::1/128
eri0: flags=2000841<UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 2
        inet6 fe80::203:baff:fe04:a4e8/10
```

Если интерфейс настроен некорректно, при выполнении следующей команды для него будет отображено соответствующее сообщение:

```
bash-2.03# ifconfig eri0
ifconfig: status: SIOCGLIFFLAGS: eri0: no such interface
```

Исходя из предположения, что устройство *eri0* установлено корректно и для него загружены нужные драйверы, мы можем выполнить приведенную ниже команду и произвести настройку на аппаратном уровне.

```
bash-2.03# /usr/sbin/ifconfig eri0 plumb
```

После такой инициализации можно установить рабочие параметры для устройства, такие как IP-адрес (по-прежнему с помощью команды *ifconfig*):

```
bash-2.03# /usr/sbin/ifconfig eri0 10.64.18.3 broadcast 10.64.18.255 netmask
255.255.255.0
```

Чтобы включить интерфейс, следует воспользоваться ключевым словом *up*:

```
bash-2.03# /usr/sbin/ifconfig eri0 up
```

Приведенные команды можно скомбинировать в одну, единовременно выполнив низкоуровневую инициализацию, настройку параметров и включение интерфейса:

```
bash-2.03# /usr/sbin/ifconfig eri0 10.64.18.3 broadcast 10.64.18.255 netmask
255.255.255.0 plumb up
```

В зависимости от конфигурации локальной сети может потребоваться создание соединения вида «точка-точка», а не универсального, как в приведенном примере. Так, если необходимо ограничить доступ к безопасной системе баз данных, можно создать соединение «точка-точка», которое ограничит доступ к базе данных узлом, напрямую подключенным к машине (рис. 4.5). В таком варианте база данных связана с сервером, который, в свою очередь, связан с маршрутизатором; данные не могут передаваться от маршрутизатора к системе БД напрямую, не пройдя через сервер. Поэтому если взломщик собирается проникнуть в систему базы данных, ему придется преодолеть защиту сначала маршрутизатора, а затем и сервера.

Чтобы определить, корректно ли соответствие между IP- и ethernet-адресами с другими узлами локальной сети, воспользуйтесь командой

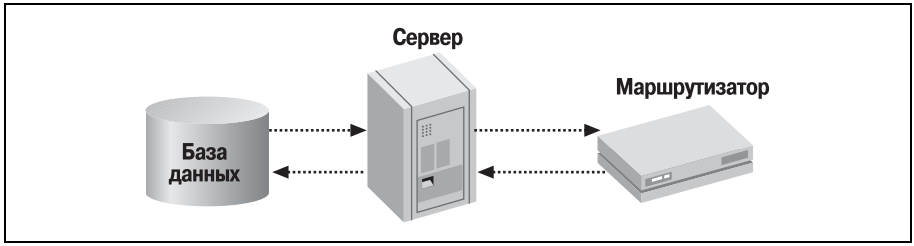

Рис. 4.5. *Повышение уровня безопасности системы баз данных с помощью соединения «точка-точка»*

arp, которая отображает все активные соединения между локальным узлом и другими узлами:

```
bash-2.03# /usr/sbin/arp -a

Net to Media Table: IPv4
Device   IP Address               Mask      Flags   Phys Addr
------ -------------------- --------------- ----- ---------------
eri0   hp                   255.255.255.255       00:50:ba:13:08:18
eri0   austin               255.255.255.255 SP    00:03:ba:04:a4:e8
eri0   224.0.0.0            240.0.0.0       SM    01:00:5e:00:00:00
```

Для локального узла выводится информация по отображению ethernet-адресов в IP-адреса. Флаги имеют следующие значения:

P	Опубликованный адрес
S	Статический адрес
U	Неопределенный адрес
M	Адрес для широковещательных (multicast) сообщений

И наконец, для достижения оптимальной производительности можно вручную установить некоторые из параметров протокола. Воспользуйтесь командой *ndd* для изменения параметров протоколов TCP, UDP, ARP и IP. *ndd* также может применяться для отображения существующих значений параметров, связанных с конкретным протоколом. Чтобы отобразить параметры, связанные с протоколом TCP, воспользуйтесь такой командой:

```
bash-2.03# ndd /dev/tcp \?
?                               (read only)
tcp_close_wait_interval         (read and write)
tcp_conn_req_max_q              (read and write)
tcp_conn_req_max_q0             (read and write)
tcp_conn_req_min                (read and write)
tcp_conn_grace_period           (read and write)
tcp_cwnd_max                    (read and write)
tcp_debug                       (read and write)
tcp_smallest_nonpriv_port       (read and write)
tcp_ip_abort_cinterval          (read and write)
```

```
tcp_ip_abort_linterval        (read and write)
tcp_ip_abort_interval         (read and write)
tcp_ip_notify_cinterval       (read and write)
tcp_ip_notify_interval        (read and write)
tcp_ip_ttl                    (read and write)
tcp_keepalive_interval        (read and write)
tcp_maxpsz_multiplier         (read and write)
tcp_mss_def                   (read and write)
tcp_mss_max                   (read and write)
tcp_mss_min                   (read and write)
tcp_naglim_def                (read and write)
tcp_rexmit_interval_initial   (read and write)
tcp_rexmit_interval_max       (read and write)
tcp_rexmit_interval_min       (read and write)
tcp_wroff_xtra                (read and write)
tcp_deferred_ack_interval     (read and write)
tcp_snd_lowat_fraction        (read and write)
tcp_sth_rcv_hiwat             (read and write)
tcp_sth_rcv_lowat             (read and write)
tcp_dupack_fast_retransmit    (read and write)
tcp_ignore_path_mtu           (read and write)
tcp_rcv_push_wait             (read and write)
tcp_smallest_anon_port        (read and write)
tcp_largest_anon_port         (read and write)
tcp_xmit_hiwat                (read and write)
tcp_xmit_lowat                (read and write)
tcp_recv_hiwat                (read and write)
tcp_recv_hiwat_minmss         (read and write)
tcp_fin_wait_2_flush_interval (read and write)
tcp_co_min                    (read and write)
tcp_max_buf                   (read and write)
tcp_zero_win_probesize        (read and write)
tcp_strong_iss                (read and write)
tcp_rtt_updates               (read and write)
tcp_wscale_always             (read and write)
tcp_tstamp_always             (read and write)
tcp_tstamp_if_wscale          (read and write)
tcp_rexmit_interval_extra     (read and write)
tcp_deferred_acks_max         (read and write)
tcp_slow_start_after_idle     (read and write)
tcp_slow_start_initial        (read and write)
tcp_co_timer_interval         (read and write)
tcp_extra_priv_ports          (read only)
tcp_extra_priv_ports_add      (write only)
tcp_extra_priv_ports_del      (write only)
tcp_status                    (read only)
tcp_bind_hash                 (read only)
tcp_listen_hash               (read only)
tcp_conn_hash                 (read only)
tcp_queue_hash                (read only)
tcp_host_param                (read and write)
tcp_1948_phrase               (write only)
```

Сбор сетевой статистики

Когда все сетевые интерфейсы настроены соответствующим образом, можно применить команду *netstat*, отвечающую за сбор сетевой статистики различного рода в целях проверки работоспособности интерфейсов. Сбор данных происходит для интерфейсов локального узла.

netstat умеет собирать статистику по следующим типам данных:

- Данные, сгруппированные по типу протокола
- Статистика по устройству, сгруппированная по типу адреса (IPv4, IPv6, адреса Unix)
- Данные DHCP
- Данные по интерфейсу, сгруппированные по адресам широковещательной передачи
- Информация из таблицы маршрутизации (включая широковещательные адреса)
- Данные по модулям STREAMS
- Состояние всех доступных IP-интерфейсов
- Состояние всех активных сокетов, маршрутов, физических и логических интерфейсов

В следующих разделах мы рассмотрим операции по сбору каждого из типов данных и обсудим их применение для диагностирования и разрешения проблем, снижающих производительность.

Статистика по протоколам

Статистику по протоколам можно разбить на несколько категорий:

Пакеты RAWIP (IP в чистом виде)	Пакеты TCP
Пакеты IPv4	Пакеты ICMPv4
Пакеты IPv6	Пакеты ICMPv6
Пакеты UDP	Пакеты IGMP

С каждым типом пакетов связан определенный набор средств. Скажем, для пакетов RAWIP существуют счетчики полученных (*rawipInDatagrams*) и отправленных (*rawipOutDatagrams*) после загрузки системы дейтаграмм. В UDP существует аналогичный набор счетчиков, измеряющих число полученных (*udpInDatagrams*) и отправленных (*udpOutDatagrams*) после загрузки системы дейтаграмм. Кроме счетчиков нормальных событий *netstat* отображает также информацию по ошибкам, такую как число UDP-ошибок получения (*udpInErrors*) и отправки (*udpOutErrors*). Эти значения следует проверять регулярно, убеждаясь, что отношение числа ошибок к числу нормальных событий со временем не растет. Например, в приведенном ниже фрагменте имеют место 293 события *tcpActiveOpens* против лишь одного *tcpAttempt-*

Fails. Если со временем отношение *tcpAttemptFails* к *tcpActiveOpens* увеличивается для трафика TCP, может потребоваться изменить определенные параметры TCP с помощью команды *ndd* либо заняться диагностированием сетевых ошибок. Вот типичный набор примеров, позволяющих разобраться с ошибками в работе протоколов для IPv6.

```
bash-2.03$ netstat -s

IPv6    ipv6Forwarding      =     2    ipv6DefaultHopLimit =   255
        ipv6InReceives      =     0    ipv6InHdrErrors     =     0
        ipv6InTooBigErrors  =     0    ipv6InNoRoutes      =     0
        ipv6InAddrErrors    =     0    ipv6InUnknownProtos =     0
        ipv6InTruncatedPkts =     0    ipv6InDiscards      =     0
        ipv6InDelivers      =    25    ipv6OutForwDatagrams=     0
        ipv6OutRequests     =    42    ipv6OutDiscards     =     2
        ipv6OutNoRoutes     =     0    ipv6OutFragOKs      =     0
        ipv6OutFragFails    =     0    ipv6OutFragCreates  =     0
        ipv6ReasmReqds      =     0    ipv6ReasmOKs        =     0
        ipv6ReasmFails      =     0    ipv6InMcastPkts     =     0
        ipv6OutMcastPkts    =    14    ipv6ReasmDuplicates =     0
        ipv6ReasmPartDups   =     0    ipv6ForwProhibits   =     0
        udpInCksumErrs      =     0    udpInOverflows      =     0
        rawipInOverflows    =     0    ipv6InIPv4          =     0
        ipv6OutIPv4         =     0    ipv6OutSwitchIPv4   =     0

ICMPv6  icmp6InMsgs         =     0    icmp6InErrors       =     0
        icmp6InDestUnreachs =     0    icmp6InAdminProhibs =     0
        icmp6InTimeExcds    =     0    icmp6InParmProblems =     0
        icmp6InPktTooBigs   =     0    icmp6InEchos        =     0
        icmp6InEchoReplies  =     0    icmp6InRouterSols   =     0
        icmp6InRouterAds    =     0    icmp6InNeighborSols =     0
        icmp6InNeighborAds  =     0    icmp6InRedirects    =     0
        icmp6InBadRedirects =     0    icmp6InGroupQueries =     0
        icmp6InGroupResps   =     0    icmp6InGroupReds    =     0
        icmp6InOverflows    =     0
        icmp6OutMsgs        =     8    icmp6OutErrors      =     0
        icmp6OutDestUnreachs=     0    icmp6OutAdminProhibs=     0
        icmp6OutTimeExcds   =     0    icmp6OutParmProblems=     0
        icmp6OutPktTooBigs  =     0    icmp6OutEchos       =     0
        icmp6OutEchoReplies =     0    icmp6OutRouterSols  =     3
        icmp6OutRouterAds   =     0    icmp6OutNeighborSols=     1
        icmp6OutNeighborAds =     0    icmp6OutRedirects   =     0
        icmp6OutGroupQueries=     0    icmp6OutGroupResps  =     4
        icmp6OutGroupReds   =     0
```

Статистика по типу адреса

Статистические данные для типов адресов делятся на три категории:

- inet (AF_INET)
- inet6 (AF_INET6)
- unix (AF_UNIX)

Взглянем на пример вывода для сокетов AF_UNIX:

```
bash-2.03$ netstat -f unix

Active UNIX domain sockets
Address     Type       Vnode       Conn     Local Addr      Remote Addr
30000d03738 stream-ord 30000d1eb78 00000000 /tmp/.X11-unix/X0
30000d038e0 stream-ord 00000000    00000000
30000d03a88 stream-ord 30000ce4a30 00000000 /tmp/jd_sockV6
30000d03c30 stream-ord 30000a62d78 00000000 /dev/kkcv
30000d03dd8 stream-ord 30000a62f50 00000000 /dev/ccv
```

Мы видим набор различных активных сокетов с Unix-адресацией, в частности сокет сервера X11 с адресом 30000d03738.

Статистика по широковещательным адресам

Статистика по широковещательным адресам позволяет выявить интерфейсы, которые принимают широковещательные сообщения по адресу 224.0.0.1 (ALL_HOSTS). Это нужно, чтобы пакеты соответствующим образом направлялись с помощью демона обнаружения маршрутизаторов (*in.rdisc*), речь о котором пойдет далее в разделе «Маршрутизация». В следующем примере отображается информация для протоколов IPv4 и IPv6 :

```
bash-2.03$ netstat -g

Group Memberships: IPv4
Interface Group               RefCnt
--------- ------------------- ------
lo0       224.0.0.1                1
eri0      224.0.0.1                1

Group Memberships: IPv6
If    Group                   RefCnt
----- ----------------------- ------
lo0   ff02::1:ff00:1               1
lo0   ff02::1                     1
eri0  ff02::202                    1
eri0  ff02::1:ff04:a4e8            1
eri0  ff02::1                      2
```

Статистика по маршрутизации

Ядро системы ведет таблицу маршрутизации, которая создается демоном маршрутизации *in.routed*. Маршруты, которые были предварительно созданы, всегда доступны для отображения с помощью команды:

```
bash-2.03$ netstat -r

Routing Table: IPv4
Destination          Gateway              Flags Ref   Use    Interface
-------------------- -------------------- ----- ----- ------ ---------
10.64.18.0           austin                U        1      5  eri0
```

```
224.0.0.0          austin              U        1      0  eri0
localhost          localhost           UH      25 215051  lo0
```

Мы видим, что основной Ethernet-интерфейс *eri0* предоставляет два сетевых маршрута для пакетов: в сеть 10.64.18.0 и в сеть широковещательной передачи 224.0.0.0. Кроме того, интерфейс loopback-вызовов (*lo0*) связан с интерфейсом локального узла (*localhost*), который часто используется для отладки и тестирования. Все маршруты относятся к стандарту IPv4, однако информация по маршрутизации доступна и для версии IPv6:

```
Routing Table: IPv6
Destination/Mask           Gateway                   Flags Ref Use    If
-------------------------- ------------------------- ----- --- ------ -----
fe80::/10                  fe80::203:baff:fe04:a4e8    U       1      0 eri0
ff00::/8                   fe80::203:baff:fe04:a4e8    U       1      0 eri0
default                    fe80::203:baff:fe04:a4e8    U       1      0 eri0
localhost                  localhost                   UH      5     28 lo0
```

Статистика по модулям STREAMS

STREAMS представляет собой пакет System V, обеспечивающий доступ к системным вызовам, стандартным библиотекам и ядру с целью разработки сетевых приложений. Любое приложение, использующее STREAMS, обладает определенным набором свойств, по которым можно собрать статистику, поскольку операции ввода–вывода отличаются от аналогичных операций в других сетевых API (скажем, в интерфейсе прикладного программирования для BSD-сокетов). *netstat* позволяет получить эту статистику, включая и информацию по очередям, реализующим характерные для потока операции чтения–записи:

```
bash-2.03$ netstat -m
streams allocation:
                                          cumulative  allocation
                     current   maximum       total    failures
streams                326       340        7634           0
queues                 938       962       18662           0
mblk                  1144      1651        7773           0
dblk                  1140      1729     2349590           0
linkblk                 11       169          18           0
strevent                 9       169      121739           0
syncq                   25        50         101           0
qband                    0         0           0           0

1646 Kbytes allocated for streams data
```

Более подробная информация доступна в системной справке по *streamio*.

Статистика по IP-интерфейсам

netstat позволяет получать статистику, собранную на уровне протокола IP. В частности доступно число полученных и отправленных паке-

тов, число ошибок при отправке и получении, а также число конфликтов пакетов. Как и ранее, информация для версий IPv4 и IPv6 отображается в двух разделах:

```
bash-2.03$ netstat -i
Name  Mtu  Net/Dest      Address        Ipkts  Ierrs Opkts  Oerrs Collis Queue
lo0   8232 loopback      localhost      227695 0     227695 0     0      0
eri0  1500 austin        austin         2573   0     2130   0     0      0

Name  Mtu  Net/Dest      Address          Ipkts  Ierrs Opkts  Oerrs Collis
lo0   8252 localhost     localhost        227705 0     227705 0     0
eri0  1500 fe80::203:baff:fe04:a4e8/10 fe80::203:baff:fe04:a4e8    2573   0
2130   0     0
```

Комбинированная статистика по сокетам, маршрутам и интерфейсам

Большинство администраторов предпочитают получать с помощью команды *netstat* информацию, скомбинированную в один отчет. Пример 4.1 содержит отчет с комбинированной статистикой по маршрутам, сокетам и интерфейсам.

Пример 4.1. Вывод команды netstat -a

```
bash-2.03$ netstat --a
UDP: IPv4
   Local Address         Remote Address     State
-------------------- -------------------- -------
      *.route                             Idle
      *.*                                 Unbound
      *.*                                 Unbound
      *.sunrpc                            Idle
      *.*                                 Unbound
      *.32771                             Idle
      *.sunrpc                            Idle
      *.*                                 Unbound
      *.32775                             Idle
      *.32779                             Idle
      *.32780                             Idle
Routing
*.*                                  Unbound
      *.32821                             Idle
      *.32822                             Idle
      *.32823                             Idle
      *.name                              Idle
      *.biff                              Idle
      *.talk                              Idle
      *.time                              Idle
      *.echo                              Idle

UDP: IPv6
   Local Address                      Remote Address                  State
```

```
If
---------------------------- ---------------------------- ------- ------
      *.*                                                         Unbound
      *.sunrpc                                                    Idle
      *.*                                                         Unbound
      *.32771                                                     Idle
      *.32779                                                     Idle
      *.*                                                         Unbound
      *.32821                                                     Idle
      *.time                                                      Idle

TCP: IPv4
   Local Address        Remote Address    Swind Send-Q Rwind Recv-Q  State
-------------------- -------------------- ----- ------ ----- ------ -------
      *.*                  *.*                0      0 24576      0 IDLE
      *.sunrpc             *.*                0      0 24576      0 LISTEN
      *.*                  *.*                0      0 24576      0 IDLE
      *.sunrpc             *.*                0      0 24576      0 LISTEN
      *.*                  *.*                0      0 24576      0 IDLE
      *.32775              *.*                0      0 24576      0 LISTEN
      *.32776              *.*                0      0 24576      0 LISTEN
      *.32782              *.*                0      0 24576      0 LISTEN
      *.32783              *.*                0      0 24576      0 LISTEN

TCP: IPv6
Local Address   Remote Address  Swind  Send-Q  Rwind  Recv-Q  State  If
*.*             *.*             0              24576  0       IDLE
*.sunrpc        *.*             0      0       24576  0       LISTEN
*.*             *.*             0      0       24576  0       IDLE
*.32775         *.*             0      0       24576  0       LISTEN
localhost.32780 localhost.32775 32768  0       32768  0       CLOSE_WAIT
*.32782         *.*             0      0       24576  0       LISTEN
*.32791         *.*             0      0       24576  0       LISTEN
*.ftp           *.*             0      0       24576  0       LISTEN
*.telnet        *.*             0      0       24576  0       LISTEN

Active UNIX domain sockets
Address  Type          Vnode     Conn  Local Addr      Remote Addr
30000d03738 stream-ord 30000d1eb78 00000000 /tmp/.X11-unix/X0
30000d038e0 stream-ord 00000000 00000000
30000d03a88 stream-ord 30000ce4a30 00000000 /tmp/jd_sockV6
30000d03c30 stream-ord 30000a62d78 00000000 /dev/kkcv
30000d03dd8 stream-ord 30000a62f50 00000000 /dev/ccv
```

Некоторые из типов сообщений TCP, содержащихся в отчете, могут быть читателю незнакомы; их интерпретация приведена в табл. 4.1.

Таблица 4.1. Константы TCP, используемые командой netstat

Сообщение	Описание
BOUND	Сокет связан
CLOSED	Сокет закрыт
CLOSING	Сокет закрывается

Таблица 4.1 (продолжение)

Сообщение	Описание
CLOSE_WAIT	Сокет в ожидании закрытия
ESTABLISHED	Сокет успешно подключен
FIN_WAIT_1	Сокет закрывается (локальный)
FIN_WAIT_2	Сокет закрывается (удаленный)
IDLE	Сокет бездействует
LAST_ACK	Сокет закроется, получив последнее подтверждение
LISTEN	Сокет активен, ведется прием сообщений
SYN_RECEIVED	Сокет синхронизируется
SYN_SENT	Сокет создает соединение
TIME_WAIT	Сокет в ожидании закрытия

Маршрутизация

Представьте себя курьером, путь которого всегда начинается у локального склада. Вам выдают перечень адресов, связанных с набором пакетов. Задача – доставить пакеты за минимальное время, учитывая следующие ограничения:

- Число дорог, пройденных для доставки конкретного пакета, должно быть минимальным.
- Следует избегать тупиков и несчастных случаев.
- Определить, какой дорогой следовать, можно только с помощью каталога улиц и сопоставления найденных имен с теми, что в списке.

Если вам кажется, что для курьера задачка тривиальна, прикиньте, насколько сложной может стать его работа, если учесть следующие факторы:

- Число дорог и их комбинаций растет экспоненциально с каждым годом. Курьера могут попросить проследовать дорогами, о которых он раньше даже не слышал!
- Заранее невозможно предугадать, где произойдет несчастный случай или окажется тупик.
- Каталог улиц, имеющийся в распоряжении, давно устарел, потому что число дорог растет экспоненциально с каждым годом.

Таково описание сложностей, с которыми было связано рождение Интернета и объединение узлов и сетей в глобальных масштабах. Чтобы передать пакет от узла A узлу B, необходимо определить физический путь, которым проследует пакет.

Не существует централизованной службы поиска, позволяющей определить путь маршрута пакета для всех возможных пар узлов Интерне-

та (то есть путь между отправителем и получателем). Это означает, что маршруты должны создаваться динамически. (Единственным исключением из этого правила являются определенные случаи, когда стабильно предсказуемые маршруты определяются статически.)

При передаче данных через Интернет или между подсетями необходимо наличие промежуточных узлов, ответственных за такую передачу. Такие узлы называются маршрутизаторами и отвечают за маршрутизацию пакетов между узлами, которые могут разделять отдельные подсети или целые континенты. Чтобы оценить, через какое число маршрутизаторов может пройти пакет, воспользуемся командой *traceroute* и взглянем на транзитные участки, по которым проходят пакеты при подключении узла в Сиднее (Австралия) к веб-серверу компании Sun Microsystems:

```
bash-2.03$ traceroute wwwwseast.usec.sun.com/
Tracing route to wwwseast.usec.sun.com [192.9.49.30]
over a maximum of 30 hops:
  1   184 ms   142 ms   138 ms  202.10.4.131
  2   147 ms   144 ms   138 ms  202.10.4.129
  3   150 ms   142 ms   144 ms  202.10.1.73
  4   150 ms   144 ms   141 ms  atm11-0-0-11.ia4.optus.net.au [202.139.32.17]
  5   148 ms   143 ms   139 ms  202.139.1.197
  6   490 ms   489 ms   474 ms  hssi9-0-0.sf1.optus.net.au [192.65.89.246]
  7   526 ms   480 ms   485 ms  g-sfd-br-02-f12-0.gn.cwix.net [207.124.109.57]
  8   494 ms   482 ms   485 ms  core7-hssi6-0-0.SanFrancisco.cw.net [204.70.10.9]
  9   483 ms   489 ms   484 ms  corerouter2.SanFrancisco.cw.net [204.70.9.132]
 10   557 ms   552 ms   561 ms  xcore3.Boston.cw.net [204.70.150.81]
 11   566 ms   572 ms   554 ms  sun-micro-system.Boston.cw.net [204.70.179.102]
 12   577 ms   574 ms   558 ms  wwwwseast.usec.sun.com [192.9.49.30]
Trace complete.
```

Можно видеть, что для передачи пакета получателю потребовалось двенадцать узлов. Кроме того, отображаемые интервалы времени, затрачиваемые на получение ответов, довольно велики – до полусекунды. Возможно, что соединение не будет установлено по причине окончания интервала ожидания. Такая диагностика может быть полезна при попытке определить, какой из промежуточных узлов (или какая из промежуточных сетей) является причиной того, что ваше соединение с узлом на другом конце света внезапно оборвалось.

В этом разделе мы рассмотрим способы, которыми в Solaris решаются классические проблемы маршрутизации.

Статическая маршрутизация, как правило, идет рука об руку с прямым физическим соединением пары узлов, то есть со случаем, когда реализация динамической маршрутизации является неэффективной либо связана с понижением уровня безопасности. Например, если в локальной сети есть три подсети, между которыми необходимо организовать обмен данными, для каждого маршрутизатора можно создать статические маршруты к двум другим маршрутизаторам. Число

конкретных маршрутов, необходимых для беспрепятственной циркуляции данных между сетями, прямо пропорционально квадрату числа маршрутизаторов в сети. После каждой модификации топологии сети все эти маршруты придется редактировать вручную. Может показаться, что речь идет о чрезмерных объемах трудной работы, но могут возникать ситуации, когда эта работа оправдана: представьте безопасный сервер баз данных, к которому можно получить доступ, только воспользовавшись конкретным маршрутом, информация о котором широко не распространяется. В этом случае именно статическая маршрутизация (а не динамическое обнаружение маршрута) является подходящим решением. Такую конфигурацию можно реализовать, создав соединение «точка-точка» на дополнительном интерфейсе с помощью команды *ifconfig* (как это сделать, рассказано в разделе, посвященном настройке сетевых интерфейсов).

Альтернативой статической маршрутизации является маршрутизация динамическая, работа которой связана с парой демонов: собственно демона маршрутизации (*in.routed*) и демона обнаружения маршрутизаторов (*in.rdisc*). Демон *in.routed* реализует протокол информации маршрутизации (Routing Information Protocol) и отвечает за обновление и сопровождение записей в таблицах маршрутизации ядра. Для выполнения операций маршрутизации демон использует протокол UDP (порт 520) и работает на всех сетевых интерфейсах, которые были инициализированы на аппаратном уровне (*plumb*) и являются активными (*up*).

В случае отсутствия файла */etc/notrouter* и при наличии двух или более рабочих интерфейсов узел начинает действовать в качестве маршрутизатора. Таким образом, данные могут поступать через один интерфейс и тут же передаваться дальше через другой. В локальной сети интерфейс, связанный со всеми локальными узлами, обычно называется *внутренним* (*internal*) интерфейсом, тогда как интерфейс, доступный другой подсети или интернет-провайдеру, называется *внешним* (*external*). Применив фильтрацию сетевых пакетов, можно создать набор правил, определяющих, какого рода TCP- или UDP-пакеты разрешено передавать через конкретные интерфейсы и порты. Такого рода фильтры, несомненно, важны для защиты локальных сетей и сокрытия служб, предназначенных только локальным узлам, от внешнего доступа.

Демон обнаружения маршрутизаторов, *in.rdisc*, использует протокол управляющих сообщений Интернета (Internet Control Message Protocol, ICMP). Что касается обнаружения маршрутов, демон *in.rdisc*, работающий на обычной машине, принимает широковещательные сообщения по адресу 224.0.0.1 (ALL_HOSTS). Сообщениям назначаются приоритеты, после чего маршрутизатор по умолчанию выбирается на основе его близости к узлу. *in.rdisc* на маршрутизаторе оповещает о своем существовании с помощью групповых сообщений на адрес 224.0.0.1 и принимает запросы по адресу 224.0.0.2 (ALL_ROUTERS). Узлы могут напрямую получить адрес маршрутизатора по адресу 224.0.0.2.