Лекция 7. Сетевые протоколы и интерфейсы прикладных программ.

Чтобы передать по локальной сети (LAN) данные из пункта А в пункт **В,** необходимо не только оборудование, организованное в одну из конфигураций, **но и** сетевое программное обеспечение, которое должно собрать передаваемые данные в пакет. Сетевое программное обеспечение состоит из трех частей:

- редиректор (redirector), отсылающий запросы в сеть, а не на локальный жесткий диск;
- драйверы сетевых плат, обеспечивающие связь между операционной системой и сетевой платой;
- сетевые протоколы для отсылки и приема данных.

Такой способ организации передачи информации показан на рис. 7.1.

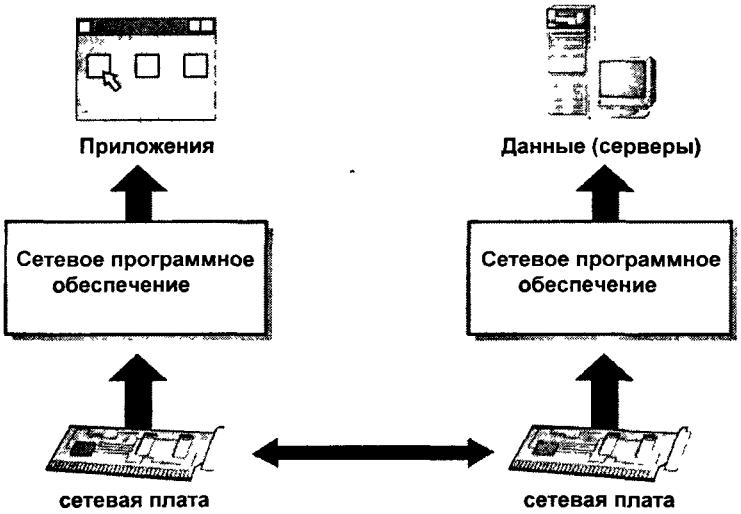


Рис. 7.1. Использование сети предоставляет пользователям возможность централизованного хранения данных

Так, например, в случае, когда необходимо сохранить файл со своего компьютера на общем жестком диске сетевого файлового сервера, происходит следующее.

1. Щелкают на кнопке **Save (Сохранить)** и выбирают опции сохранения документа на диске G:, который фактически является сетевым общим диском D: файлового сервера.
2. Редиректор проверяет этот запрос на сохранение, фиксирует локальную недоступность диска и направляет запрос в раздел операционной системы, называемый *драйвером сетевой файловой системы.*
3. Драйвер сетевой файловой системы передает запрос драйверу сетевой платы.
4. Драйвер сетевой платы передает запрос (вместе с данными) сетевой плате.
5. Сетевая плата создает пакет данных для передачи и отсылает его по сети.
6. Сетевая плата файлового сервера отмечает факт прибытия пакета и принимает его.

Далее процессы идут в обратном направлении: драйвер сетевой платы сервера передает запрос драйверу файловой системы операционной системы и записывает файл на локальный диск.

Если вы организуете сеть, то весьма вероятно, что рано или поздно вам понадобится соединить ее с другой сетью. Именно так и зародилась

Internet: локальные сети получили средства соединения друг с другом в единое целое. Большинство сетей растут постепенно, а не создаются сразу в окончательном виде. Часто сети состоят из оборудования, сетевых операционных систем и линий связи смешанных типов. Это, в свою очередь, означает, что в сетях необходимо использовать редиректоры разных типов и устанавливать несколько сетевых протоколов. Кроме того, сетевые платы могут использовать разные драйверы. Короче говоря, даже при самом тщательном планировании, рост и слияние сетей представляют собой достаточно сложный процесс. В этой главе рассматриваются различные сетевые компоненты. Ознакомившись с ними, вы сможете управлять сетевым программным обеспечением, когда это потребуется.

Прежде всего, мы обсудим роль редиректора — он должен обеспечить связи между приложениями и операционной системой, необходимые для получения приложением данных через сеть. Затем рассмотрим назначение драйвера сетевой платы. И, наконец, изучим три транспортных протокола, которые чаще всего используются в сетях персональных компьютеров.

Назначение и работа редиректора.

Как показано на рис. 7.2, с точки зрения компьютера, инициализирующего запрос по сети, редиректор играет главную роль в создании сетевого соединения. Его задача - "обманным" путем заставить приложение на локальной машине полагать, что оно получает данные с локального, а не сетевого диска. Суть применения редиректора: место хранения запрошенного файла не должно иметь значения, так как метод доступа должен быть единым.

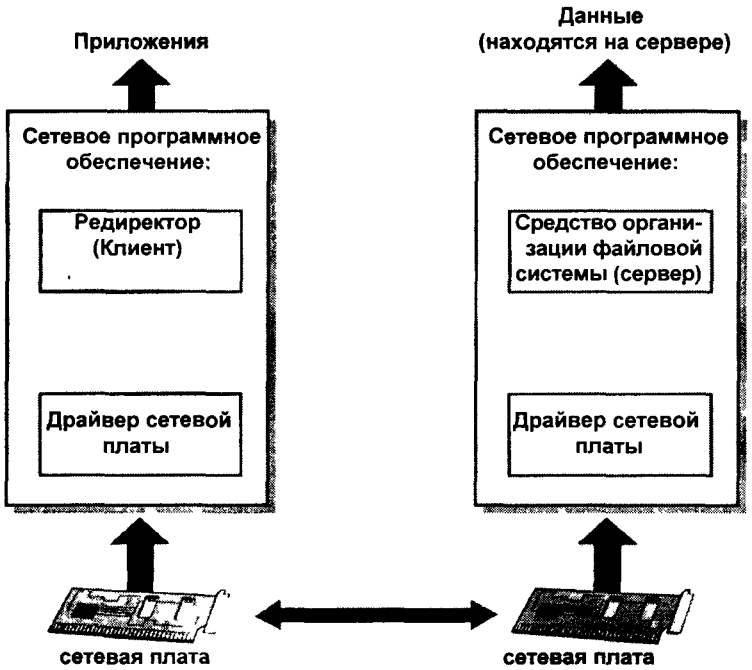


Рис. 7.2. Место редиректора в сети.

Например, что случится, если вы запустите программу WordCruncher и откроете файл, хранящийся на сетевом диске? С точки зрения WordCruncher сети *не существует.* Она знает только о существовании нескольких доступных дисков с именами, состоящими из букв с двоеточиями, например, F:, в:, с: и т.д. Как и любое другое приложение, WordCruncher способна использовать только локальные устройства хранения. Таким образом, необходимо задействовать тот уровень программного обеспечения (расположенный непосредственно под уровнем WordCruncher), задача которого заключается в предоставлении WordCruncher обычного интерфейса, основанного на буквенных обозначениях дисков (common drive-letter interface). Но делать это необходимо и в случае, когда приложению требуются данные, хранящиеся в сетевом устройстве. WordCruncher полагает, что она обращается к локальным дискам, однако ее запросы информации с сетевых дисков должны *перехватываться* и направляться по сети. Итак, если вы скомандуете WordCruncher получить данные из каталога DOCS, находящегося на сервере с именем BGDOG, программа редиректора передаст запрос так, как изложено в самом начале лекции.

Примечание

Редиректоры нередко называют *клиентами* (например, клиент Windows 98 для сетей NetWare), поскольку они обязательно входят в состав программного обеспечения клиентного компьютера.

Следовательно, прежде чем вы сможете подключиться к сети, вы должны установить редиректор, соответствующий типу сети. Отнюдь не во всех сетевых операционных системах используют одинаковые редиректоры. Поэтому вы должны установить редиректор, соответствующий операционной системе той сети, к которой подсоединяетесь. Различие между редиректорами может быть обнаружено на уровне представления данных модели OSI, (лекция 7. Так, в сетях Microsoft для передачи данных используют блоки сообщений сервера (Server Message Blocks — SMBs). Поэтому для них необходим редиректор, способный выразить запросы в терминах SMBs. В сетях Novell с этой же целью применяют программу NetWare Control Protocols (NCPs). Таким образом, чтобы запросить сервер NetWare, вы должны использовать редиректор, который может выразить запрос в терминах NCPs. Обратите внимание: недостаточно применить общий транспортный протокол, такой как TCP/IP — вы должны использовать редиректор, работающий с той операционной системой, к которой вы подсоединяетесь. К счастью, если на уровне представления данных задействован протокол, одинаковый для нескольких операционных систем, поддержка редиректора обеспечивается автоматически. Так, если вы работаете в Windows 98, то можете использовать Client for Microsoft Networks (Клиент сетей Microsoft), поддерживающий SMB, чтобы подключиться к любой операционной системе, поддерживающей SMBs.

Редиректоры и средства API.

Как указано во введении в эту главу, большинство пользовательских приложений "не подозревают" о наличии используемой сети (сетей). Однако

некоторые приложения, например, электронная почта или групповое программное обеспечение *должны* быть "осведомлены" о наличии сети, поскольку они вообще нужны *только для* работы в сети, должны быть способны "подключаться" к ней и устанавливать связь с другими программами, которые исполняются сетевыми машинами.

Программисты разрабатывают приложения, способные определять наличие сети таким образом, чтобы они воспринимали набор команд, предоставляемый сетью приложениям. Эти наборы команд называют средствами API или *интерфейсами прикладных программ.* API можно представить чем-то вроде комбинации органов управления и приборов автомобиля. Комбинация приборов вашего автомобиля ~ видимая часть интерфейса, предназначенного для управления автомобилем. Фактически, вы не представляете, что творится под капотом — просто нажимаете педаль газа, и автомобиль разгоняется.

Таким образом, вам нет необходимости детально знать устройство автомобиля, чтобы ездить на нем. Более того, если вы умеете управлять одним автомобилем, вы справитесь и с другим, поскольку его элементы управления — т.е. API — везде одинаковы.

(Однажды вечером, во время поездки на "Фольксвагене" моего друга, я узнала, что в разных автомобилях этой марки используются разные "API" для заднего хода. Это часто бывает в автомобилях с ручным переключением передач, но в целом наша аналогия верна).

Комбинация органов управления автомобилем позволяет передать только несколько простейших команд: торможение, ускорение, переключение скоростей и т.д. Например, нельзя дать команду "выехать задним ходом на дорогу". Вы должны отдать множество простейших команд, чтобы выполнить *фактическое* действие: выехать задним ходом на дорогу. В известном смысле необходимо создать программу с помощью стандартной комбинации органов управления, расположенных в некотором порядке.

Функции API вашего компьютера во многом работают точно так же. Сетевые программные средства, подобные редиректору, должны находиться на верхнем уровне различных транспортных протоколов. Если бы не было API, программистам сетевого программного обеспечения пришлось бы разработать одну программу-редиректор для подключения Windows NT (сетевая операционная система) к IPX/SPX, а другую программу-редиректор — для подключения Windows NT к TCP/IP. Структурно редиректор будет один и тот же. Просто он будет в состоянии "разговаривать" с разными транспортными протоколами. Чтобы избежать этого, следует создать общую комбинацию органов управления и приборов для всех сетевых служб. Таким образом, программа-редиректор встроена не в протокол, а в API (в нашем примере — в NetBIOS). NetBIOS может "сидеть" на верхнем уровне протоколов IPX/SPX, NetBEUI и TCP/IP. Это означает, Что транспортный протокол может изменяться, однако вам нет нужды переписывать сетевые утилиты, поскольку они написаны для API (NetBIOS).

Широко известным примером API служат *гнезда* (sockets) — временные каналы связи, установленные для передачи информации между клиентной и серверной программами. Данные программы могут работать как на одной машине, так и на разных машинах через сеть. Существуют три API, с

которыми вы, вероятно, уже сталкивались.

• Гнезда Novell.

• NetBIOS.

• Гнезда **TCP/IP.**

API перехватывают ваши сетевые запросы и выполняют поставленную задачу с помощью соответствующего транспортного протокола.

Драйверы файловых систем.

Редиректор расположен на том конце соединения, которое создает (генерирует) запрос. Другая часть, расположенная на том конце соединения, на котором *выполняется* запрос, — драйвер файловой системы.

Драйверы файловых систем используют для генерирования не только сетевых, но и любых других запросов на доступ к устройству хранения данных. Например, в Windows NT функции драйвера сетевой файловой системы может выполнять одна из поддерживаемых файловых систем:
FAT, NTFS, CDFS и сеть.

Примечание

FAT - это таблица размещения файлов, первоначально предназначавшаяся для работы с гибкими дисками (дискетами) и допускавшая работу с жесткими дисками объемом не более 4 Гбайт. NTFS (NT File System) - файловая система, использующаяся в Windows NT. В ней предусмотрена поддержка дисков больших объемов и средства защиты, отсутствующие в FAT. В приводах CD-ROM персональных компьютеров применяют файловую систему CDFS (Compact Disc File System).

В целом же роль *любого* драйвера файловой системы заключается в упорядочивании данных в устройстве хранения, которое он обслуживает. **Так,** в популярной дисковой файловой системе FAT, нумеруется каждый кластер и указывается, какой файл в нем сохраняется. Если для хранения данных, содержащихся в файле, необходимо несколько кластеров, в каждый кластер включают указатель на следующий кластер диска, хранящий данные из этого же файла. В последний кластер включается метка End of File (конец файла), которая позволяет файловой системе FAT "узнать" что файл закончился. А как файловая система "узнает", где находится кластер? Когда вы форматируете диск, то разбиваете его на кластеры с помощью специальной программы.

Когда вы предписываете программе найти файл, файловая система FAT предоставляет ей все необходимые данные и гарантирует их полное извлечение (поскольку весьма вероятно, что данные разбросаны по разным кластерам). Точно так же, когда вы пытаетесь сохранить файл на диске, файловая система FAT позволяет найти на нем первый свободный кластер и начать записывать данные, составляющие этот файл, в свободных кластерах, помечая каждый кластер так, чтобы он указывал на следующий занятый данным файлом кластер.

Примечание

В зависимости от структуры доступного дискового пространства свободные

кластеры могут и не составлять непрерывный блок. Файловая система не ищет первую группу свободных кластеров, общий размер которой достаточен для хранения всего файла в целом, а находит только первый свободный кластер. Если файл не помещается в него полностью, то остальные данные будут последовательно записываться в следующие свободные кластеры, в каком бы месте диска они не находились.

Геометрия диска.

Вы уже сбиты с толку этими разговорами о кластерах и сохранении в них данных? Суть файловой системы проста: в файловых системах Microsoft кластером называют наименьшую возможную логическую единицу, предназначенную для хранения данных. Поверхность диска разделена на концентрические круги, называемые *дорожками* (tracks), а они, в свою очередь, разделены на клиновидные участки (напоминающие нарезанный пирог), называемые *секторами.* Сектор — наименьшая единица хранения вместимостью 512 байт.

Кластерами называют логические группы секторов. Точное число секторов в кластере зависит от используемой файловой системы и размера диска. Чем больше емкость диска, тем больше общее количество кластеров в файловой системе, хотя в современных файловых системах и наблюдается тенденция к снижению количества кластеров. Использование более крупных кластеров позволяет снизить время доступа к диску, поскольку в крупном кластере можно сохранить больше данных, причем в одном месте на диске. С другой стороны, использование крупных кластеров при сохранении небольших файлов ведет к бесполезной трате дискового пространства, поскольку короткий файл все равно занимает весь кластер полностью. Так, если на диске хранится файл длиной 4 Кбайта, а размер кластера данного диска составляет 8 Кбайт, то оставшиеся незанятыми 4 Кбайта будут потеряны, поскольку они не могут использоваться для хранения других данных.

Сетевая файловая система представляет собой всего лишь еще один интерфейс для чтения дискового пространства. Единственное ее отличие от других файловых систем заключается в том, что она используется для сетевого, а не локального доступа. Таким образом, когда сервер получает **от** какого-нибудь клиента сети запрос на доступ к диску, он направляется в сетевую файловую систему, которая и выполняет то, что обязана — находит данные, сохраняет файл или что-либо еще. Достоинство ее состоит в том, что сетевая файловая система позволяет клиенту не беспокоиться о формате жесткого диска сервера. Даже если файловая система, в которой был отформатирован сетевой диск, не поддерживается клиентным программным обеспечением, это никак не сказывается на работе локального приложения, поскольку для выполнения сетевых запросов локальная файловая система не используется. Пока клиент "разговаривает" с сервером, тот будет интерпретировать для клиента свою файловую систему.

Драйверы сетевых плат.

Теперь вы знаете, каким образом запрос поступает от приложения в

операционную систему и выполняется ею. Как же этот запрос поступает в сеть? Эту задачу выполняет часть программного обеспечения, называемая *драйвером сетевой платы.*

Вообще говоря, любой драйвер устройства представляет собой часть программного обеспечения, позволяющего операционной системе и физическому устройству взаимодействовать друг с другом. Некоторые драйверы устройств входят в состав файлов операционной системы. Другие же можно загрузить с диска или Internet, однако они в любом случае остаются интерфейсом между сетевой платой и операционной системой.

Почему же нельзя просто встроить функциональные средства, необходимые сетевой плате, непосредственно в операционную систему, особенно, если учесть, что современные операционные системы создаются с учетом возможности работы в сети? Главным образом потому, что это непрактично. Десятки и даже сотни производителей предлагают тысячи моделей сетевых плат, причем в каждой модели используют собственный драйвер. По этой причине просто невозможно встраивать в операционную систему функции поддержки всех сетевых плат. И даже если бы эта схема имела практический смысл, она была бы нежелательна. В этом случае операционная система занимала бы слишком много места, намного больше *реально* необходимого. Если вы установили единственную сетевую плату, остальные драйверы вам не нужны. И более того: обновление драйверов для улучшения поддержки оборудования потребовало бы замены операционной системы. Разве вам понравится переустанавливать операционную систему при каждом обновлении драйвера сетевой платы? Не думаю.

Кстати сказать, в некоторых операционных системах используют модульную конструкцию, позволяющую в нее включить определенные функции драйвера. Подразумевается, что производителям оборудования нужно будет **только** дописать оставшуюся часть программы (stub portion) драйвера. Идея такого подхода такова (рис. 7.3): хотя каждая сетевая плата может управлять передачей данных между операционной системой и сетью методом, несколько отличным от используемых в других платах, основная функция сетевой платы неизменна в любом случае. Таким образом, эту функцию действительно можно встроить в операционную систему. С этой точки зрения создателю драйвера сетевой платы достаточно написать инструкции, специфичные для данной платы, которые позволят плате получать доступ к функциональным средствам, встроенным в операционную систему

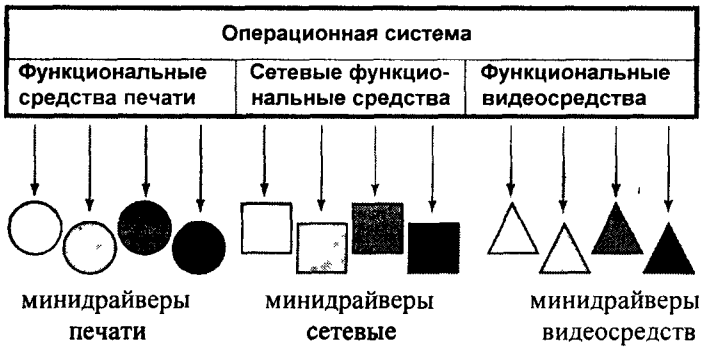


Рис. 7.3. Взаимодействие микродрайверов с операционной системой.

На этом мы завершим обсуждение общих вопросов, связанных с драйверами операционной системы, и приступим к основному — работе этих драйверов.

Что делают драйверы сетевых плат.

Драйверы сетевых плат (сетевые драйверы) отвечают за управление всеми внешними связями компьютера, в том числе и доступом в Internet. Для каждой используемой модели сетевой платы необходимо установить соответствующий сетевой драйвер. Так, если в состав персонального компьютера входит кабельный модем для подключения к Internet, а также сетевое оборудование, то ему необходимы два драйвера. Если в компьютере установлено несколько сетевых плат разного типа (например, он используется в качестве маршрутизатора), то для каждой платы также следует установить собственный драйвер. Однако если в компьютере есть две одинаковые платы, достаточно поставить единственный драйвер. По существу, драйверы обеспечивают сетевое соединение компьютеров на канальном уровне, позволяя, например, получать доступ к сетям Ethernet.

Интерфейсы сетевых компоновок.

В принципе, возможно создать сетевой драйвер, который сможет посылать данные как на сетевую плату, так и в сеть. По сути, он будет объединять собственно драйвер и средства поддержки сетевого протокола. Драйверы такого типа называют *монолитными драйверами устройств.*

Обратите внимание: я сказала *"возможно"*. Этот подход не рекомендуется во многом по тем же причинам, что и встраивание драйверов устройств в операционную систему. Гибкость обеспечивается только модульным подходом. Предположим, в драйвер вашей сетевой платы включены средства поддержки транспортного протокола. В этом случае, чтобы заменить или добавить еще один транспортный протокол, придется заменить драйвер. Такой подход неудобен, поскольку поддерживает только единственный транспортный протокол.

Поэтому вместо монолитных драйверов устройств в современных драйверах для "привязки" сетевых плат к транспортным протоколам используют другой инструмент — *интерфейс сетевых компоновок.* Как показано на рис. 7.4, — это интерфейс между драйвером сетевой платы и стеком транспортных протоколов (transport stack).

Примечание

Интерфейс сетевых компоновок- именно то средство, которое обеспечивает "увязку" протокола с сетевой платой. Это, в сущности, "свадьба" драйвера сетевой платы с сетевым транспортным протоколом.

Принцип его применения совершенно очевиден. Интерфейс сетевых компоновок связывает каждый установленный драйвер с каждым установленным транспортным протоколом. Названия этих связей (clump) зависят от типа сети. Так, в сетях NetWare их называют модулями. Обменом данных между модулями управляет программа, которую называют LAS в сетях NetWare и PROTMAN.SYS в Microsoft. Вся информация, необходимая ин-

терфейсу компоновок, сохраняется в текстовом файле, таком как PROTOCOL. INI в сетях Microsoft или NET.CFG в NetWare. Однако, если в драйверах используют стандартные значения параметров, то в этих файлах содержится совсем немного информации.

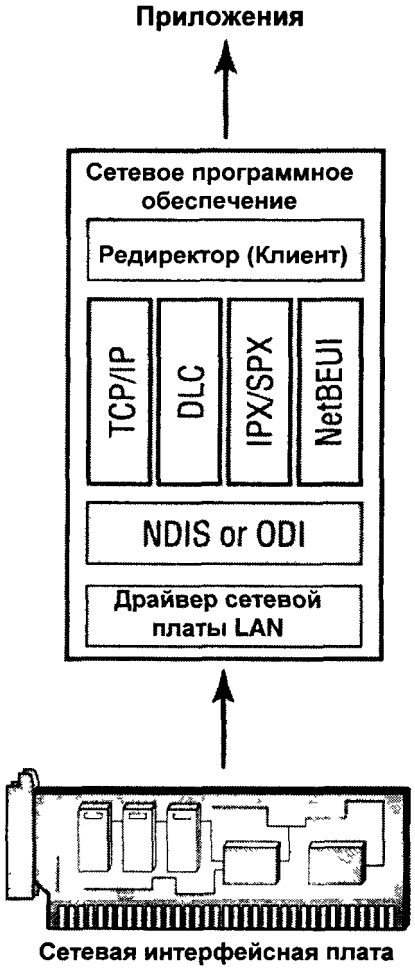


Рис. 7.**4**. Интерфейс сетевых компоновок позволяет нескольким протоколам связываться с единственным драйвером.

Связующие нити.

Как указано выше, основное преимущество использования интерфейса сетевых компоновок — возможность использования каждым сетевым драйвером нескольких транспортных протоколов. По умолчанию все сетевые транспортные протоколы "привязываются" ко всем установленным драйверам сетевых плат. Какой же протокол используется при отсылке данных по сети? Ответ зависит от порядка, в котором "привязывались" транспортные протоколы. Этот порядок можно изменять, поставив на первое место протокол, используемый чаще остальных. Это позволит уменьшить число повторных попыток его поиска, что необходимо для доступа к конкретному серверу.

Если же необходимо запретить доступ к серверу, использующему

конкретный протокол, достаточно просто удалить его из списка протоколом "привязанных" (bound) к сетевой плате. Запутались? Хорошо. Если вы запутались сейчас, но этот вопрос все же вас интересует, знайте, что вернусь к нему в лекции 18 "Защита сетей".

Звание "законодателя мод" в мире стандартов интерфейса сетевых компоновок оспаривают два конкурента: открытый интерфейс передачи данных (ODI) фирмы Novell и спецификация интерфейсов сетевых драйверов (NDIS) фирмы Microsoft. Принцип работы обоих интерфейсов во многом сходен. Основное различие в том, что драйверы ODI работают в реальном (незащищенном) режиме. Значит они должны использовать первые 640 Кбайт памяти, установленной на машине, и не могут "договариваться" с другими драйверами. Напротив, драйверы NDIS функционируют в защищенном режиме и поэтому могут работать в многозадачном режиме вместе с прочими драйверами. Кроме того, они *не используют* дефицитную обычную память.

Зачем же применяют драйверы ODI, если они не работают в защищенном режиме? Главным образом потому, что иногда они необходимы редиректору. Отнюдь не все редиректоры могут работать с драйверами NDIS. Чтобы рассеять сомнения, поднимите документацию на вашу операционную систему и ознакомьтесь с требованиями к редиректору. Вообще же говоря, как NDIS, так и ODI работают со всеми транспортными протоколами, а ограничения налагаются, в основном, со стороны драйвера.

Сетевые транспортные протоколы.

На протяжении всей главы я постоянно упоминала транспортные протоколы, фактически не пояснив, что же это такое. Во-первых, *что такое* протокол вообще? Проще всего представить протокол как стандарт или набор правил.

С протоколами часто приходится иметь дело в повседневной жизни. Например, если в вашей квартире зазвонил телефон, вы снимаете трубку и говорите: "Алло?". Почему *вы* сначала сказали "Алло"? Почему вызывающая сторона, прежде всего не идентифицировала себя? Да просто таков обычай. По традиции, в Соединенных Штатах первым отвечает вызываемое лицо. Иными словами, таков американский *протокол* ответа по телефону. Если бы вы жили в материковом Китае, вы подняли бы трубку и ожидали, когда вызывающая сторона скажет "Вэй" (Wei), чтобы обозначить себя. Что вы делаете, встретив знакомого: обнимаете, целуете, пожимаете руку? Опять же, ответ зависит от протокола.

Какое отношение все это имеет к сетям? Сетевые транспортные протоколы определяют метод передачи данных по сети, а также метод их пакетирования и адресации. Чтобы два сетевых компьютера могли установить связь, они должны использовать одинаковые транспортные протоколы, поскольку протокол определяет метод пакетирования и обмена данными.

Примечание

С технической точки зрения к протоколам можно отнести *любое* соглашение, задающее метод передачи данных через сеть, независимо от того, функционирует ли это соглашение на канальном или прикладном уровнях.

Однако нередко протоколами называют соглашения, которые задают метод пакетирования и передачи данных по сети. Именно это я буду иметь в виду в дальнейшем. Протоколы NetBEUI, IPX/SPX и TCP/IP работают на сетевом и транспортном уровнях модели OSI. Поскольку они работают на нескольких уровнях, их нередко называют *стеками протоколов, а* не просто протоколами.

Современные операционные системы могут одновременно поддерживать несколько протоколов, поэтому вы всегда можете использовать для установления связи протокол нужного типа. К сожалению, *не все* операционные системы поддерживают *все* транспортные протоколы. Фактически они "склонны" к специализации, и даже некоторые протоколы с одинаковыми именами невозможно использовать на всех платформах. Однако в настоящее время возможности операционных систем в достаточной мере перекрываются, что допускает связь и взаимодействие между ними.

Совет

Использование нескольких транспортных протоколов дополнительно загружает оперативную память (RAM). Поэтому, хотя современные операционные системы позволяют загружать несколько транспортных протоколов, рекомендуем выбрать только те, которые действительно необходимы.

Предупреждаю: последующие разделы следует рассматривать **как** введение в три основных протокола сетей персональных компьютеров. Протокол TCP/IP чрезвычайно сложен, в нем предусмотрены многочисленные параметры конфигурирования, достойные отдельной книги или даже нескольких. То, что приведено здесь, должно просто помочь вам получить общее представление о работе каждого протокола и всего, что с ним связано.

Протокол NetBEUI.

Давным-давно, когда IBM только вышла на рынок сетей персональных компьютеров, ей понадобился базовый сетевой протокол. Фирма не собиралась создавать крупные сети, а предусматривала организовывать небольшие рабочие группы, не более нескольких дюжин компьютеров.

Исходя из такой задачи и была разработана базовая сетевая система ввода-вывода — NetBIOS. Она представляет собой набор из 18 команд, которые позволяют создавать, поддерживать и использовать соединения между сетевыми компьютерами. Вскоре IBM дополнила NetBIOS, создав расширенный пользовательский интерфейс среды NetBIOS — NetBEUI. По существу NetBEUI — усовершенствованный вариант NetBIOS. Однако со временем имена NetBEUI и NetBIOS приобрели разный смысл. NetBEUI обозначает транспортный протокол, в то время как NetBIOS — набор программных команд, используемых системой для управления сетью (фактически это — API). Обращаясь к модели OSI, можно отметить, что NetBIOS работает на сеансовом уровне, в то время как NetBEUI — на сетевом и транспортном.

NetBEUI относится к числу наиболее скоростных протоколов (с точки зрения скорости пакетирования данных для передачи и извлечения их получателем из пакета). Кроме того, он прост в установке: вы устанавливаете протокол, "привязываете" его к сетевому драйверу и можете приступать к работе. Не требуется никакого конфигурирования, а в качестве адресов

компьютеров используются присвоенные вами же имена компьютеров, которые нетрудно запомнить.

Примечание

Имена компьютеров в сетях, использующих протокол NetBEUI, фактически представляют собой имена NetBIOS. Длина имен не должна превышать 16 символов, а регистр не имеет значения.

Применение NetBEUI ограничивает серьезная проблема: этот протокол не поддерживает маршрутизацию. Иными словами, если ваша сеть состоит из нескольких сегментов, то вы не сможете использовать его для передачи данных за пределы локального сегмента. Кроме того, он поддерживается только операционными системами Microsoft и OS/2, и если вы планируете связываться с файловым сервером, работающим под управлением UNIX или сервером печати NetWare, то потерпите фиаско.

В настоящее время использование NetBEUI ограничено. Быстродействие протокола TCP/IP возросло, так что NetBEUI потерял былое преимущество в производительности. В связи с повсеместным распространением Internet, вам непременно следует использовать TCP/IP — основной протокол этой сети. И даже несмотря на то, что некоторые приложения NetBIOS требуют поддержки протокола NetBEUI, в настоящее время с тем же успехом можно использовать NetBIOS "поверх" (over) протокола IP.

В целом же протокол NetBEUI нельзя назвать совсем бесполезным, но применимость его ограничена небольшими сетями, состоящими из единственного узла, которым не нужна связь с Internet и другими сетями.

Протокол IPX/SPX.

Транспортный протокол IPX/SPX — интеллектуальная собственность фирмы Novell. Фактически он состоит из двух протоколов: IPX и SPX. IPX (Internet Package Exchange — межсетевой обмен пакетами) — протокол сетевого уровня, не ориентированный на установление соединения (connectionless). Он отвечает за поиск наилучшего пути передачи пакетов к месту назначения и за отбор их по прибытии, управляет адресацией и маршрутизацией пакетов. Таким образом, на уровне IPX назначают логические сетевые адреса (в отличие от адресов аппаратных средств, которые используют на канальном уровне и "зашиты" изготовителями в сетевые платы). IPX-адрес состоит из четырехбайтового (32-битового) сетевого номера и шестибайтового (48-битового) узлового номера.

Что означают эти номера? *Сетевой номер,* называемый также *внешним сетевым адресом,* идентифицирует физический сегмент, к которому подключен компьютер. Если в сегмент входят несколько серверов, они должны использовать единый внешний адрес.

Примечание

Внешние сетевые адреса присваивают только серверам NetWare. Клиентные машины сетей NetWare наследуют эти адреса от серверов, с которыми устанавливают связь.

Узловой номер, или *внутренний сетевой адрес* — это обычно адрес аппа-

ратных средств сетевой платы компьютера, что очень удобно, поскольку избавляет от необходимости транслировать имена, назначенные программными средствами (software-assigned names) в имена аппаратных средств. При установке сервера Novell NetWare вам предлагают принять или заменить внутренний номер IPX. Затем этот номер становится идентификационным номером (ID) данного сервера. Если на рабочей станции ввести команду slist, то вы увидите список ID для каждого перечисленного сервера.

На рис.7.5 показан формат полного IPS-адреса сети NetWare.

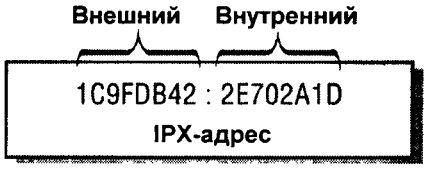


Рис. 7.5. Логический адрес NetWare.

Совет

Если вам необходимо поддерживать NetBIOS, вы можете задействовать поддержку NetBIOS "поверх" IPX/SPX.

Обратите внимание: протокол IPX управляет *маршрутизацией.* Иными словами, протокол IPX/SPX, в отличие от NetBEUI, относится к маршрутизируемым протоколам. Поэтому с его помощью можно соединять сегменты сети, разделенные маршрутизаторами. Чтобы найти наилучший путь, каждые 60 с в сети с протоколом IPX (средствами протокола RIP (Routing Information Protocol — обмен информации о маршрутизации) или NLSP (NetWare Link Service Protocol — обслуживание каналов NetWare)) передается широковещательный запрос (broadcast), т.е. посылается в предполагаемое место по всем известным каналам. После этого возвращается своеобразное "эхо". По времени задержки эхо-сигнала определяется кратчайший путь, ведущий в заданное место. Подобный метод имеет единственный недостаток: протокол IPX/SPX слишком "шумный" — постоянно обновляя текущие маршруты данных, оборудование генерирует значительный сетевой график. Подробнее маршрутизация рассматривается в лекции 12 "Дополнительное сетевое оборудование".

Поскольку IPX не поддерживает обработку ошибок, в некоторых случаях он дополняется средствами протокола SPX (Sequenced Package Exchange — последовательный обмен пакетами). Этот протокол транспортного уровня ориентирован на установление соединений. Он гарантирует установление надежного соединения перед отправлением данных по сети. Поскольку протокол SPX обеспечивает корректность работы соединения, то отвечает **за** обработку искаженных пакетов и других ошибок. Его основные функции заключаются в вычислениях в среде пар клиент/сервер, которым необходима связь, исключающая ошибки.

Протоколы IPX и SPX являются собственностью фирмы Novel, но благодаря протоколам, совместимым с IPX/SPX, их применение не ограничено сетями NetWare. Аналогичные протоколы отличаются от исходного,

но могут устанавливать с ним связь. Эти совместимые протоколы позволяют сетям Microsoft устанавливать связь с сетями NetWare, хотя в них и не используется общий транспортный протокола. Протоколы NetWare IP и TCP/IP фирмы Microsoft друг с другом не совместимы.

Протокол TCP/IP.

TCP/IP разработан ARPANET (Сеть перспективных исследований и разработок) по заказу министерства обороны США. Предназначен для соединения сетей с разнородным оборудованием, скажем, систем Sun с мэйнфреймами, а мэйнфреймов — с персональными компьютерами. Каждая половина имени протокола "TCP/IP" означает его ориентированность на решение собственной задачи.

Примечание

Строго говоря, TCP/IP состоит не из двух частей - фактически это набор из нескольких протоколов. Однако аббревиатуры IP и TCP известны лучше всех, так что здесь мы и остановимся на них.

Протокол Internet (IP — Internet Protocol) работает на сетевом уровне, предоставляя различным сетям стандартный набор правил и спецификаций для межсетевой пакетной маршрутизации с помощью IP-адресов. Протокол IP позволяет устанавливать связь как между локальными сетями, так и между отдельными компьютерами. С другой стороны, протокол управления передачей данных (TCP — Transmission Control Protocol) работает на транспортном уровне модели OSI. Он обеспечивает прием сетевой информации и трансляцию ее в форму, "понятную" сети, и таким образом организует взаимодействие процессов между двумя компьютерами или клиентами. IP можно представить себе как часть, задающую правила установления связи, а TCP — как часть, отвечающую за интерпретацию данных.

Вы можете сами оценить работу TCP/IP по тем задачам, которые он решает. Это — транспортный протокол Internet, системы, соединяющей тысячи отдельных компьютеров и сетей по всей планете. Хотя TCP/IP первоначально предназначался для использования университетами и армией, он стал самым популярным протоколом и офисных приложений, так как позволяет соединять локальные сети, UNIX-машины, миникомпьютеры DEC VAX, а также и множество компьютеров иных типов.

Что происходит, если удаленный компьютер Macintosh отсылает данные на персональный компьютер главного офиса? Во-первых, средства протокола TCP обеспечивают установление связи, гарантирующей дуплексный контроль ошибок (контроль ошибок данных в обоих направлениях) между обеими платформами. Во-вторых, IP задает правила установления связи и обеспечивает соединение портов компьютеров. К этому времени в соответствии с TCP подготавливаются данные. Соответствующая программа забирает их, разделяет на меньшие части, если их объем слишком велик, и вставляет в пакет новый заголовок ("адрес пересылки"), чтобы гарантировать правильную доставку пакета. Кроме того, в пакете указывается тип содержащихся данных и их объем. Затем пакет конвертируется в стандартный зашифрованный формат и передается на персональный компьютер главного

офиса. Наконец, программа, установленная на персональном компьютере в главном офисе, транслирует шифрованный пакет в собственный формат в соответствии с TCP. Этот процесс показан на рис. 9.6.

Во многих сетях, разбросанных по всему миру, протокол TCP/IP используется как стандартный. Это — единственное средство коммуникации, позволяющее связываться рабочим станциям всех типов — PC, Macintosh, UNIX. Кроме того, он необходим для выхода в Internet. TCP/IP работает несколько медленнее NetBEUI, однако издержки производительности компенсируются широкой поддержкой протокола. И действительно, лучше работать несколько медленнее, но иметь возможность связаться со всем миром, чем быстрее, но в пределах небольшой рабочей группы.

Требования к конфигурированию TCP/IP.

Один из недостатков протокола TCP/IP состоит в трудности его установки неопытными пользователями, поскольку необходимо задать множество адресов и серверов. Как указывалось ранее, установка адресов протоколов NetBEUI и IPX/SPX не встречает затруднений. При использовании NetBEUI вы назначаете имя компьютера, а при IPX/SPX — идентификатор сети (network identifier) и позволяете системе назначить собственный идентификатор узла (node identifier), основываясь на адресе аппаратных средств сетевой платы компьютера. Однако протокол TCP/IP требует указания множества адресов, а именно:

- локальный IP-адрес;
- IP-адрес сервера DNS (Domain Name Service — служба имен домена), который транслирует легко запоминаемые человеком адреса, например, computer.company.com, в IP-адреса;
- в сетях Windows NT, использующих имена **NetBIOS** для идентификации компьютеров, следует указать IP-адрес сервера WINS (Windows Internet Name Service — система присвоения имен Internet для Windows), который транслирует имена NetBIOS в IP-адреса;
- шлюз (gateway) по умолчанию (т.е. портал (главный вход) в следующий сегмент сети), который необходим также и для доступа в Internet;
- число (называемое *маской подсети),* идентифицирующее сетевой сегмент, в котором расположен данный компьютер;
- если задействовано динамическое присвоение IP-адресов, следует указать IP-адрес сервера, назначающего IP-адреса.

Если вы уже хоть немного работали в Internet, вероятно, вам знакома концепция IP-адресов. Говоря упрощенно, они представляют собой 32-битовые (четырехбайтовые) числа, которые идентифицируют сетевой компьютер.

По сути, это программный адрес компьютера, в отличие от аппаратного адреса, "зашитого" в сетевую плату. В двоичном формате IP-адрес выглядит примерно так:

11000000 01101010 01111110 11000001

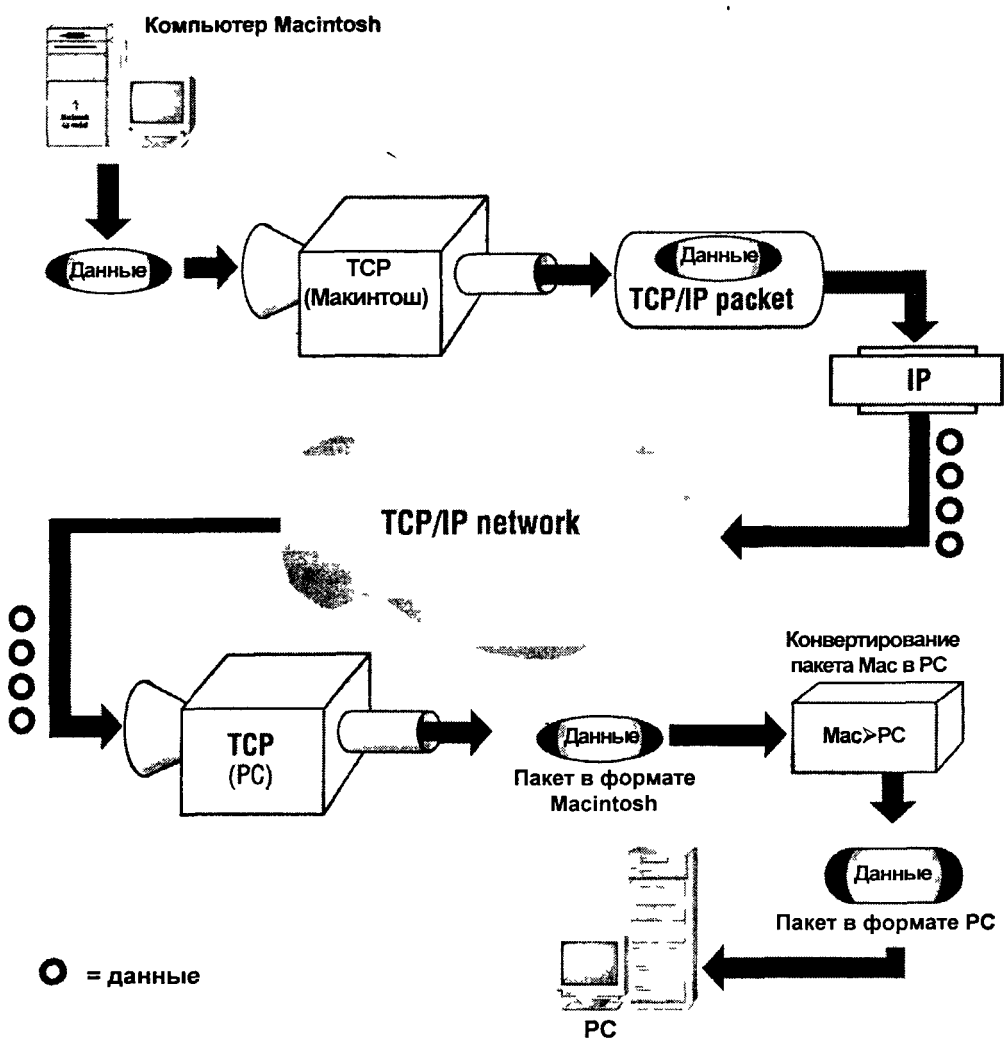


Рис. 7.6. Пересылка пакета с компьютера Macintosh на PC по протоколу TCP/IP.

В таком виде его нелегко понимать пользователям, за исключением разве что программистов (и компьютеров). Поэтому, исключительно для удобства, IP-адреса обычно записывают в формате октетов, разделенных точками. В этом формате каждый байт из 32 битов номера преобразуется в десятичное число.

192.106.126.193

Каждой сетевой плате, работающей в сети TCP/IP, присваивается уникальный IP-адрес, идентифицирующий ее *во всей сети,* а не только в локальном сегменте.

Откуда берутся эти IP-адреса? Где узнать, какие числа следует включать в них? Ответ зависит от "области охвата" вашей сети. Если вы создаете IP-адреса для локальной сети TCP/IP, которая *никогда* не будет подключена к Internet, то можете назначать их до некоторой степени произвольно (достаточно помнить, что двум сетевым платам *нельзя* назначать одинаковый адрес). Если же вы собираетесь подключиться к Internet, необходимо

получить уникальные IP-адреса, обратившись в международную организацию InterNIC.

InterNIC — единственная организация, которая уполномочена выделять IP-адреса заинтересованным фирмам и организациям. В первом приближении можно считать, что она выделяет группы IP-адресов, основываясь на размерах организаций. С этой целью InterNIC предоставляет организации конкретные числа для первого байта (первых двух, или трех байтов), а для назначения остальных адресов позволяет организации использовать оставшиеся номера по собственному усмотрению. Так, например, если вы запросили у InterNIC набор адресов Internet, вам могут предоставить набор, скажем, 192.106.X.X. Это означало бы, что все ваши IP-адреса должны начинаться с префикса 192.106, но вы можете назначить номера (вплоть до 255) по собственному усмотрению остальным двум октетам. Часть, назначенную InterNIC, называют *полем сети* адреса, а ту, что назначена вами, — *полем узла* (host portion).

Совет

Если вас еще не испугала идея самостоятельно назначать IP-адреса каждому компьютеру вашей организации, ужаснитесь сейчас. Вам предстоит не только назначить номера, но также и создать файл, сопоставляющий имена компьютеров с IP-адресами. Этот файл должен поддерживаться и постоянно обновляться на каждом сетевом компьютере. Чтобы упростить задачу, вы можете запустить сервер DHCP (протокол динамического конфигурирования компьютера), чтобы взять IP-адреса из специального буфера, а затем использовать серверы WINS или DNS для преобразования имен в адреса. Серверы WINS работают с именами NetBIOS, а DNS - с доменными именами.

Краеугольный камень адресации Internet заключается в идентификации не собственно компьютера, а *подсети,* т.е. той части сети, в которую компьютер входит Это достигается не с помощью внешнего сетевого адреса, как в адресах IPX/SPX, а с *масками подсети.* Маска подсети — число, которое можно "наложить" на IP-адрес. Если сетевая часть IP-адресов компьютеров совпадает с ней, значит, машины находятся в одной подсети. В противном случае, два IP-адреса относятся к разным подсетям.

Установить связь между двумя компьютерами одной подсети несложно. Оборудование передает данные (в соответствии с требованиями протокола TCP/IP) с помощью широковещательной передачи, а компьютер, адрес которого совпадает с указанным в пакете IP, принимает данные. Если компьютеру одной подсети необходимо связаться с компьютером в другой, — запрос должен поступить на маршрутизатор, соединяющий подсети. Маршрутизатор просматривает сетевой адрес места назначения, определяет, находится ли он в данной подсети или нет, а затем направляет пакет в следующую подсеть. Затем этот маршрутизатор проверяет IP-адрес места назначения, определяет, находится ли он в *данной* подсети, и вслед за этим либо передает сообщение с помощью широковещательной передачи, либо снова направляет пакет в следующую подсеть. Эта процедура продолжается до обнаружения нужной подсети Задача маршрутизации весьма сложна (мы вернемся к ней в лекции 9 "Дополнительное сетевое оборудование").

Когда пакет поступает в место назначения, протокол определения адреса (ARP) преобразует IP-адрес в аппаратный адрес сетевой платы. Помимо того,

протокол ARP отвечает за трансляцию адресов исходящих данных.

Куда мы двинемся дальше? Часть 1 - протокол IP версии 6.

Internet воплотил предсказания авторов фантастических романов о создании мировой компьютерной сети. Поскольку же эта сеть работает по протоколу TCP/IP, его изменения соответственно отражают и потребности этой глобальной сети.

Примечание

В настоящее время используется версия 4 протокола IP Версии 5 не существует.

В конце 1998 г. протокол IP — часть набора протоколов TCP/IP, отвечающая за маршрутизацию пакетов по сети, — начали адаптировать к изменениям типов передаваемых данных и для улучшения управления возрастающего трафика Internet. Изменения в протоколе позволяют:

- улучшить адресацию, поддерживающую более длинные (до 128 бит) адреса, а также *кластерные адреса* (cluster address) или *групповые адреса* (anycast address), идентифицирующие группы узлов сети TCP/IP;
- упростить форматы заголовков, позволяющие компенсировать влияние на сеть укрупненных пакетов IP;
- улучшить поддержку расширений и параметров, включая пробел для пустых расширений с целью облегчения изменения формата пакета (если это понадобится в дальнейшем);
- ввести метки потоков, идентифицирующих потоки пакетов, поступающих с конкретного узла;
- ввести дополнительные расширения, повышающие возможность контроля ошибок и идентификации пользователей, а также (при необходимости) защиту данных.

Подробные сведения о версии 6 протокола IP (IPv6) описаны в RFC 1883. Здесь они приведены частично, в основном, для тех, кто ни за что в жизни не желает конструировать маршрутизаторы.

Повышенная гибкость маршрутизации. Логически IP-адреса совершенно несложны: их число не превышает того, что можно "выжать" из 32 бит, т.е. всего-навсего 4 294 967 296 (около 4 миллиардов — если это вам интересно), а каждое устройство в Internet "требует" собственного IP-адреса. Если принять во внимание, что не все адреса доступны, то это число дополнительно ограничивается следующими причинами:

- Десятичное значение каждого октета 32-битового адреса не превышает 255.
- Многие адреса резервируются для целей, отличных от тех, для которых предназначены обычные IP-адреса. Например, адреса, начинающиеся с 10 в первом октете используются только локально.
- Фирмам и организациям выдают *группы* адресов, которыми они распоряжаются самостоятельно, независимо от того, нужны они им или нет. Например, все адреса, начинающиеся с 192.233-x.x принадлежат фирме Novell. И даже если адрес 192.233.54.5 в ней не используется, *никто,* кроме Novell, не сможет им воспользоваться.

Для уменьшения числа необходимых IP-адресов предпринималось

несколько попыток (серверы DHCP для временного выделения адресов, CIDR и т.п.). Однако число пользователей Internet растет, и вскоре потребуются 128-битовые адреса. Я не прорицатель, и это нравится мне точно так же, как необходимость звонить по телефонному номеру из 10 цифр, но, тем не менее, 128-битовые IP-адреса необходимы точно так же, как телефонные номера из 10 цифр — более краткие идентификаторы уже непригодны.

Для облегчения адресации пакетов группам пользователей (не обязательно подсетям или сетям) предусматривается использование групповых адресов (anycast address). Вместо отсылки пакетов индивидуально каждому члену группы, вы должны будете отсылать их кластеру, который представляет собой логическую, а не физическую группу.

Примечание

Групповые адреса (anycast addresses) заменят широковещательные адреса (broadcast address), использование которых предусмотрено протоколом IPv4.

Метки потоков (Flow labelling). Удлинение адресов может причинить неудобства пользователям, которым придется их вводить, однако упростит идентификацию компьютеров Internet. Другая проблема Internet, вызывающая беспокойство, — график.

На заре своего появления Internet поддерживала небольшой трафик. Большую часть данных, передаваемых по сети, составляли файлы и сообщения электронной почты. Однако со временем характер графика изменился. Теперь он состоит из поддержки групп новостей и досок объявлений, позволяющих посылать сообщения на всеобщее обозрение. Появились комнаты для переговоров (chat rooms) и Web. Кроме того, в настоящее время стали возможны и телефонные переговоры по Internet. Загрузка данными продолжает расти, и это вызвано ростом числа служб и пользователей — они сами по себе загружают трафик значительно больше, чем передача файлов.

Однако представьте себе на мгновение, что Internet — это множество сетей, соединенных маршрутизаторами. Маршрутизаторы рассматриваются в лекции 12 "Дополнительное сетевое оборудование". Каждый маршрутизатор отвечает за идентификацию наилучшего пути передачи данных по месту назначения. С этой целью он должен идентифицировать место назначения каждого принятого пакета, т.е. открыть и исследовать множество пакетов. Так вот, в пакете IPv6 есть поле, где можно указать конкретный поток, к которому относится пакет. Идея такова: если маршрутизатор установит, что пакет является частью потока пакетов, следующих в одно и то же место, ему фактически нет нужды определять, где находится это место, после того, как он исследует первый же пакет в проходящей группе (flow group).

Примечание

По умолчанию маршрутизатор должен "помнить" метку потока в течение шести секунд, однако это время можно увеличить вручную.

Приоритет пакета. Иногда трафик Internet может стать настолько тяжелым, что пакет может "погибнуть". Как правило, пакеты отбрасываются без учета их важности. Однако пакетам IPv6 можно присваивать приоритеты в соответствии с назначением.

Значения приоритетов разделены на два диапазона: 0—7 и 8—15. При перегрузке сети пакеты с низшим приоритетом (номером) в пределах данного диапазона отбрасываются в первую очередь, причем каждый диапазон рассматривается отдельно. Иными словами, пакет с приоритетом 6 не обязательно отбрасывается ранее пакета с приоритетом 8, поскольку пакет с приоритетом 6 в пределах своего диапазона имеет высший приоритет.

Значения приоритетов 0—7 используют для указания приоритета трафика, для которого источник обеспечивает контроль перегрузки (congestion control), т.е. трафик, использующий протокол высшего уровня (например, TCP) для отслеживания способности системы управлять потоком данных, при перегрузке системы прерывается. Ниже приведены значения приоритетов графика с контролем перегрузки, предусмотренные спецификацией IPv6.

О — трафик без приоритета.

1 — трафик -"заполнитель" ("Filler" traffic) (сетевые новости).

2 — необслуживаемая передача данных (электронная почта).

4 — обслуживаемая передача большого объема данных **(FTP,** NFS).

6 — интерактивный трафик (telnet, протокол дисплея).

7 — управляющий трафик Internet (маршрутизирующие протоколы, SNMP).

Примечание

Приоритеты 3 и 5 зарезервированы на будущие категории.

Значения 8—15 используют для указания приоритета графика, который не прерывается в ответ на перегрузку. К нему относятся пакеты речевой и видеоинформации, отсылаемые с постоянной скоростью. Они не отмечены в спецификации, но, как правило, более важная информация (скажем, слабо различимый голос) должна иметь приоритет выше, чем информация, которая при передаче имела бы превосходное качество, но малосущественна (скажем, высококачественная видеоинформация).

Поддержка крупнейших пакетов. Среди прочих расширений (extensions), предназначенных для улучшения "отклика" IPv6 на условия работы в Internet, можно отметить те, которые позволяют увеличить размер пакетов **IP,** т.е. нести больший объем полезных данных по сравнению с IPv4. Это весьма полезная возможность, поскольку применение крупных пакетов позволяет передать данные с помощью меньшего числа пакетов, что, в свою очередь, уменьшает задержку при маршрутизации пакетов.

Куда мы двинемся дальше? Часть II - миграция к IPv6.

Как вы заметили, текущая реализация протокола IP значительно отличается от той, что маячит на горизонте. Без использования новых параметров, таких как приоритеты и управление потоками, адресация пакетов была бы значительно сложней. Если же в пакете этих установок не существует, новые возможности игнорируются. Переход на новый протокол не происходит автоматически. Не думайте, что одним прекрасным утром вы обнаружите, что все узлы вашей сети обновились до IPv6.

Задача перехода от IPv4 к IPv6 отнюдь не проста. В настоящее время открылись курсы, на которых обучают решению этой проблемы и, кроме того, можно использовать некоторые сетевые ресурсы. Лучшие из них находятся по адресу

http://www.emos.be/coexist/etg071/gintrod.htmttINICIO.

Не вдаваясь в детали, укажу четыре метода перехода сети на протокол IPv6.

• Поддержка обоих протоколов.

• Включение в один пакет адресов для обоих протоколов.

• Создание туннеля IPv6 посредством протокола IPv4.

• Трансляция заголовков с тем, чтобы узлы IPv6 могли связываться с узлами IPv4.

Если же вы решитесь пройти весь этот путь, вам придется обновить всю сеть в следующем порядке.

1. Обновить сервер DNS для поддержки новых адресов.
2. Обновить узлы для поддержки как IPv4, так и IPv6.
3. Развернуть обновленные узлы.
4. Обновить область (сегмент сети) для полного перехода на протокол IPv6, причем оба протокола должны поддерживать только граничные маршрутизаторы.
5. Обновить маршрутизаторы для полного перехода на протокол IPv6.
6. Развернуть новые маршрутизаторы.

Итак, вы обновляете систему определения имен, а затем, работая "изнутри", распространяете IPv6 по всей сети, причем по ходу процесса обеспечиваете совместимость протоколов.

Поддержка нескольких стеков транспортных протоколов.

Теперь, по прочтении главы, вам должны быть очевидны два момента. Во-первых, идеального сетевого протокола не существует. Во-вторых, весьма вероятно, вам понадобится использовать все три протокола (NetBEUI, IPX/SPX и TCP/IP), причем каждый по особой причине - к счастью, это возможно. Поддержка множества транспортных протоколов является немалой ценностью современной сетевой модели (рис. 7.7).

Можно заметить, что в клиентный компьютер загружено четыре транспортных протокола, в то время как на сервере загружен единственный. Это бывает в том случае, если клиентный компьютер одновременно связан с несколькими серверами. Стек протоколов IPX может обеспечивать связь с серверами Novell, стек TCP/IP — с сервером Internet, а протокол NetBEUI используют для локального доступа к компьютерам с операционными системами Microsoft. Каждый транспортный протокол привязан к драйверам и редиректору, установленным на каждом компьютере. Как указывалось ранее, в разделе "Интерфейсы сетевых компоновок", порядок компоновки различных протоколов можно изменять. Таким образом, при соединении с серверами NetWare в первую очередь будет использован протокол IPX/SPX (или совместимый с ним); а при соединении с Internet — протокол TCP/IP.

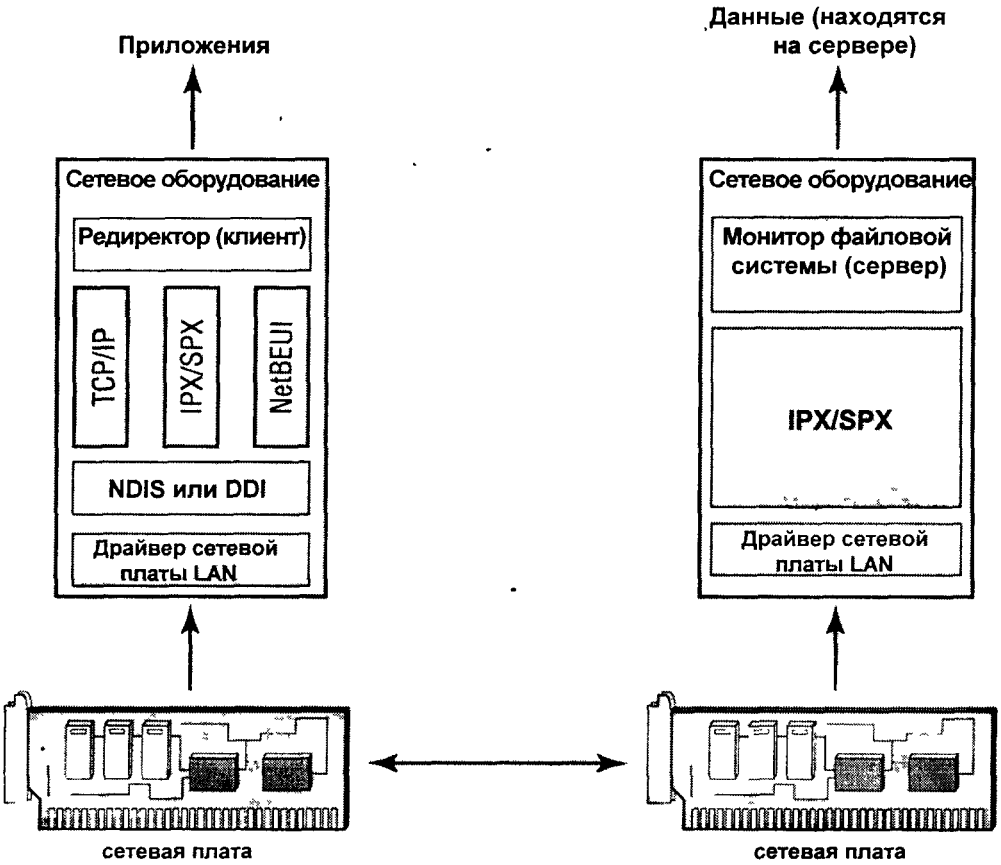


Рис.7.7. Сеть, допускающая использование нескольких транспортных протоколов.

Выводы.

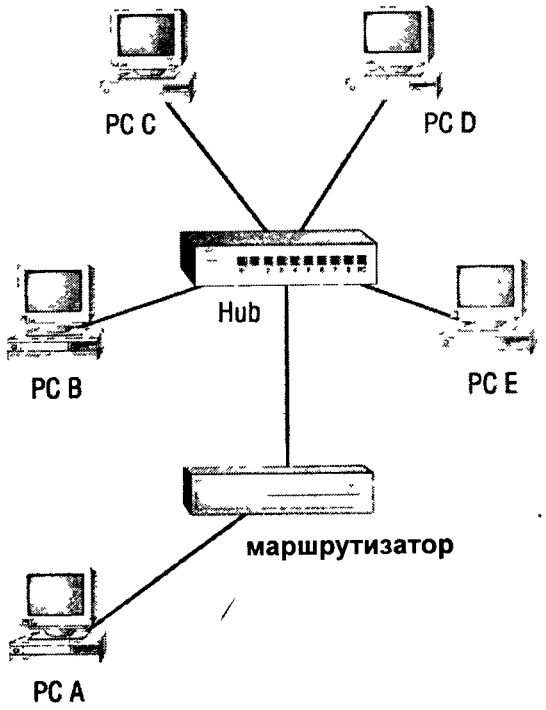
Чтобы передать данные по компьютерной сети, вам необходимо не только оборудование, но и сетевое программное обеспечение.

- Редиректор, гарантирующий отправление запроса драйверу соответствующей файловой системы для получения с ее помощью доступа через сеть.
- Сетевая файловая система, предназначенная в данном случае для отображения дискового пространства.
- Сетевой драйвер, позволяющий операционной системе установить связь с сетевой платой.
- Сетевой транспортный протокол, обеспечивающий пакетирование и отправление информации по адресу назначения через сеть.

Каждая часть сетевого программного обеспечения обособлена, что придает системе гибкость и упрощает при необходимости замену (обновление) отдельных компонент. Тем не менее, все части жизненно важны для установления надежной связи между компьютерами. В предыдущих главах большое внимание уделялось подробностям сетевых соединений. Вы получили начальные сведения о тех из них, которые понадобятся в дальнейшем. В следующей главе я расскажу, как собрать все это вместе, чтобы создать локальную сеть.

Упражнение к лекции 7.

1. Посмотрите на рисунок, приведенный ниже. Какие транспортные протоколы, из числа обсуждавшихся в этой лекции, можно использовать для отсылки данных от PC A к PC D? Если есть ограничения, то какие?



2. Допустим, один компьютер работает под управлением Windows 98, а второй — DOS. Следует ли использовать на *каждом* компьютере сетевые драйверы ODI или NDIS. Или это не имеет значения?
3. В каком транспортном протоколе, упомянутом в этой главе, используют шлюз по умолчанию (default gateway)?
4. Адреса **IPv4** имеют длину ____ **бит, а адреса IPv6 _____ бит.**
5. Какая часть сетевого программного обеспечения управляет запросами данных с сетевых дисков с приложениями?
6. Средство API, используемое для установки связи между клиентной и серверной программами, называют _____
7. Что такое монолитный драйвер устройства?