Московский государственный университет им. М.В. Ломоносова

Факультет вычислительной математики и кибернетики

Устюгов М.Б.

Введение в **TCP/IP**

(открывая чёрный ящик **Internet**)

(учебное пособие для студентов)

под редакцией Машечкина И.В.

1996

Московский государственный университет им. М.В. Ломоносова

Факультет вычислительной математики и кибернетики

Устюгов М.Б.

Введение в **TCP/IP**

(открывая чёрный ящик **Internet**)

(учебное пособие для студентов)

под редакцией Машечкина И.В.

1996

УДК 681.3.06

Материал подготовил Устюгов М.Б. Научный руководитель и редактор Машечкин И.В.

Рецензенты:
Баула В.Г., кандидат ф.-м. наук,
Королёв Л.Н., чл.-корр. РАН.

Устюгов М.Б. **"Введение в TCP/IP (открывая чёрный ящик Internet)** (учебное пособие для студентов)" - М.: Изд-во факультета ВМиК МГУ (лицензия ЛР №040777 от 23.07.96) 97 с.

Печатается по решению Учёного Совета факультета вычислительной математики и кибернетики МГУ им. М.В. Ломоносова.

ISBN 5-89407-005-8

Содержание

1. Введение

1.1. Назначение этого документа, аудитория, соглашения

Смысл названия данного пособия можно объяснить двумя способами. Во-первых, с житейской точки зрения в черном ящике обязательно лежит что-то любопытное. Во-вторых, в науке (особенно в кибернетике) черный ящик поминают всякий раз, когда речь заходит о том, чем можно пользоваться, но нельзя (или не нужно) понять. Цель пособия состоит в том, чтобы объяснить, как этот ящик работает.

Автору хотелось бы, чтобы читатель обладал некоторыми знаниями в области сетей ЭВМ. Желательно, чтобы он хотя бы раз поработал в Internet. Но в то же время автор не советует данное пособие тем, кому нужно настроить ту или иную программу ОС UNIX, связанную с Internet - здесь нет таких подробностей. Кроме того, здесь не описаны (даже мельком) бесчисленные информационные ресурсы Internet (хотя, конечно, название может ввести в заблуждение, пообещав кому-нибудь Мир Internet в черном ящике). Но... это пособие написано для тех, кто хочет узнать, **как** работает Internet, **как** устроены его протоколы, **какие** в нём существуют потоки данных и почему это сделано так, а не иначе.

Для того, чтобы читатель мог лучше ориентироваться в англоязычных руководствах, автор, помимо перевода специальных терминов и всевозможных сокращений, приводит их оригинальный вариант (в скобках после перевода). Кроме того, одним из приложений к этому пособию является глоссарий, в котором объяснены некоторые термины, используемые в тексте.

Семейство протоколов TCP/IP, которое лежит в основе работы Internet, оказалось настолько удачным, что было реализовано на нескольких десятках различных архитектур ЭВМ. В некоторых из них используется нестандартная длина байта (отличная от 8 бит). Поэтому для совместимости реализаций протоколов TCP/IP и во избежание разночтений в стандартных документах Общества Internet (ISOC, Internet Society) используется термин *октет* (octet).

1* — 1232

Он означает объект, состоящий ровно из восьми бит. Чтобы не вводить читателя в заблуждение, мы оставим этот термин в стороне, но всегда, говоря "байт", будем иметь в виду байт, состоящий ровно из восьми бит.

1.2. История появления и развития Internet

Прежде чем рассказывать о том, как работает Internet, пожалуй, стоит сделать небольшой экскурс в её историю.

К счастью, сейчас ещё можно ограничиться несколькими небольшими параграфами, чтобы коротко рассказать об истории Internet. Но думали ли разработчики из американского агентства оборонных исследовательских проектов (DARPA, Defense Advanced Research Projects Agency), что через несколько десятилетий их детищу будет уделяться столько внимания? Итак, в 1969 году DARPA решило создать экспериментальную сеть коммутации пакетов ARPANET. Её целью было отработать технологию создания надёжных и совместимых средств коммуникации. Многие из идей, лежащих в основе этой сети, очень актуальны и сейчас. Например, связь в сети считается ненадёжной, то есть любой отрезок сети может в любой момент времени выйти из строя. Поэтому ARPANET была построена так, чтобы потребность в информации от узловых компьютеров сети (хостов) была минимальной. Для любой пересылки по сети хост должен лишь правильно адресовать пакет с данными. Ему не нужно знать, какой путь проделает этот пакет. Более того, сама сеть этого не знает. Ответственность за пересылку данных несут взаимодействующие между собой компьютеры. Основной принцип заключается в том, что каждый компьютер в сети может общаться с каждым, возможно, при помощи посредников, то есть промежуточных компьютеров.

ARPANET оказалась настолько успешной, что многие подключенные к ней организации стали пользоваться ею для ежедневной переписки. Поэтому в 1975 году ARPANET сменила статус экспериментальной сети на статус действующей. Но её усовершенствование продолжалось, и к началу 80-х годов был разработан базовый набор протоколов ARPANET - TCP/IP. В 1983 году эти протоколы стали военными стандартами, и всем компьютерам сети было предписано перейти на использование только этих протоколов. Для облегчения перехода была нанята компания Болт, Беранек, энд Ньюмен (BBN, Bolt, Beranek, and Newman, Inc.), которая встроила TCP/IP в операционную систему BSD (Berkeley)

UNIX. Тот факт, что довольно мощное и удобное семейство протоколов было теперь составной частью популярной в университетской среде операционной системы, и определил судьбу этого начинания.

С 1983 года ведет отсчёт собственно сама Internet. С этого времени к ARPANET стали подключаться многие организации, имеющие собственные сети, базирующиеся на TCP/IP. Сама ARPANET была разделена на две сети - MILNET и меньшую по размерам ARPANET. Термин Internet тогда означал сообщество этих двух сетей. Сейчас *Internet* - это всемирное объединение подключенных друг к другу сетей, использующих для связи протокол IP. Кроме того, существует термин *internet*, который означает множество различных физических сетей, использующих для связи общий протокол (не обязательно IP). Это множество образует единую логическую сеть.

В 1988 году ARPANET прекратила своё существование. Другая сеть, базирующаяся на TCP/IP и основанная Национальным научным фондом - NSFNET (National Science Foundation Network), заменила её. Рост Internet продолжался. Со временем стали использоваться более совершенные средства связи и узловые компьютеры. К примеру, если в 1988 году средняя скорость передачи между узлами NSFNET составляла 56 килобод, то к середине 1995 года она достигла отметки в 45 мегабод. Всё больше компаний подключалось к Internet. Количество компьютеров в сети росло по экспоненте. Некоторые данные: в августе 1981 года к Internet было подключено 213 компьютеров-хостов, в августе 1983 - 562, в ноябре 1986 - 5,089, а в январе 1992 - 727,000. По разным подсчётам, в декабре 1995 года услугами Internet пользовалось уже от 20 до 40 миллионов человек во всём мире.

Особенно бурно Internet стала развиваться после появления первых качественных коммерческих реализаций программ-серверов и программ-клиентов протокола HTTP (Hypertext Transfer Protocol) и языка HTML. Сейчас они вместе составляют то, что называется "Всемирной паутиной" (WWW, World Wide Web). Начавшееся в августе 1994 года полноценное коммерческое использование WWW (в это время появились первые реализации методов защиты передаваемой протоколом HTTP информации, что позволило обмениваться номерами кредитных карт и т.п.) привело к появлению и становлению рынка услуг в Internet. Его рост за год (август 1994 - август 1995 года) колоссален - с нуля центов до $100 миллионов. Создаёт-

ся такое впечатление, что только теперь Internet начинает жить полной жизнью.

Все эти данные говорят об успехе идеологии Internet и её неотъемлемой части - семейства протоколов TCP/IP, так как всё программное обеспечение, работающее в этой сети, базируется на этих протоколах. Такого не было бы без жёсткой политики в отношении к ним.

1.3. Протоколы и их стандарты

Прежде всего, что такое *протокол*? Под этим словом мы будем понимать формальное описание сообщений и правил, по которым два компьютера обмениваются этими сообщениями. Разумеется, протоколы могут быть разными по сложности, выполнять разные функции - один будет описывать правила общения сетевых карт посредством электрических сигналов, а другой - обмен данными между двумя программами (например, копирование файлов между машинами).

В Internet тоже имеются протоколы, выполняющие разные функции: от самых абстрактных - взаимодействия двух программ (например, передача информации об объектах виртуальной реальности), до самых "приземлённых" - обмена блоками данных, о назначении которых самим протоколам ничего не известно. Количество и разнообразие протоколов в Internet поражает воображение. Но представьте, что каждый разрабатывает себе протокол, который нужен только ему. Тогда количество протоколов станет просто невероятным, но польза от них будет минимальна, поскольку придется обеспечивать их совместимость. Чтобы не было хаоса, в Internet решили ввести очень жесткое ограничение на протоколы, используемые всеми, кто работает в этой сети. Такие протоколы называют стандартными протоколами, или просто *стандартами*.

Прежде чем стать стандартом, протокол должен пройти долгий путь. Сначала из кандидатов в стандарты он должен стать предложенным стандартом, т.е. сетевое сообщество должно постановить, что протокол в будущем, возможно, станет стандартом. После шести месяцев испытаний (как минимум) и при наличии двух независимых реализаций протокол может стать черновым стандартом. Через четыре месяца, после ещё более жестких испытаний и после подтверждения независимости протокола от разных платформ (если это требуется), он может стать стандартом. Тогда те, кому

нужен этот протокол, обязаны будут реализовывать его именно в таком виде, в каком он определен стандартом.

Но кто принимает решения об изменении статуса протокола? Для этого в Internet существует специальная Группа управления проектированием (IESG, Internet Engineering Steering Group), которая является, конечно же, виртуальным сообществом (своего рода кибер-НИИ) и принимает все решения по стандартизации протокола.

Не все стандартные протоколы в Internet прошли полный процесс стандартизации. Некоторые из них стали стандартами de-facto, поскольку были реализованы какой-нибудь крупной фирмой на очень большом количестве оборудования. Но, тем не менее, ни один протокол не может стать стандартом без рекомендации IESG. Кстати, на сегодняшний день стандартов насчитывается не более шести десятков, хотя среди черновых и предложенных есть немало протоколов, которые используются чрезвычайно широко.

Стандарты, принятые другими организациями, стандартными протоколами Internet не считаются. Например, семейство сетевых протоколов OSI (Open Systems Interconnection), стандарт Международной организации стандартизации (ISO, International Organisation for Standardisation), не является стандартом в Internet.

1.4. Краткий обзор сетевой модели OSI

ISO в своё время разработала стандарт Взаимодействия открытых систем (OSI). Неотъемлемой частью этого стандарта является модель сетевого взаимодействия OSI. На неё мы в дальнейшем будем опираться, описывая архитектуру сети, принятую в Internet. Поэтому рассмотрим поближе организацию сети с точки зрения этой модели.

Модель OSI содержит семь *уровней*, которые определяют функции протоколов передачи данных. Каждый *уровень* представляет собой набор действий, выполняющихся для передачи данных в сети между взаимодействующими программами. *Уровень* не определяет какой-то протокол, на любом из них может быть несколько протоколов. Но каждый протокол выполняет функции, присущие именно этому *уровню*. На рисунке 1.4.1 показаны все *уровни* и кратко описаны их функции.

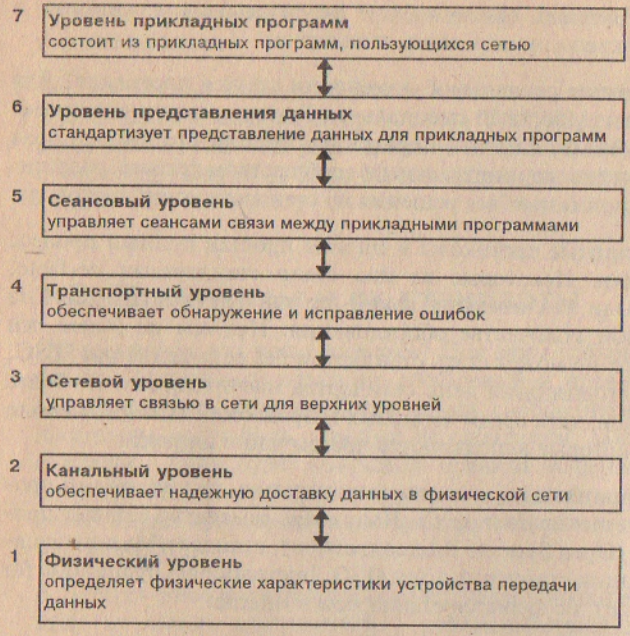


Рисунок 1.4.1: Семиуровневая модель OSI

Давайте немного подробнее посмотрим на задачи, выполняемые каждым из уровней этой модели, чтобы понять, почему было выбрано именно такое разбиение.

Уровень прикладных программ

На этом уровне располагаются все прикладные и системные программы, работающие с сетью.

Уровень представления данных

Протоколы этого уровня стандартизуют представление данных в сети для прикладных программ. Внутри разных машин одни и те же объекты (целые числа, строки и т.д.) могут быть представлены по-разному, но для передачи по сети нужны единые соглашения. Протоколы, реализующие эти соглашения, то есть перевод из внутреннего представления в сетевое и обратно, являются примером протоколов уровня представления данных.

Сеансовый уровень

На этом уровне осуществляется управление сеансами связи между взаимодействующими программами. Сюда входят функции управления паролями, синхронизации обмена данными, отмены сеанса в результате сбоя и т.д.

Транспортный уровень

Задачи корректной транспортировки данных решаются именно на этом уровне. Он отвечает за обнаружение и исправление ошибок и за сервис транспортировки данных.

Сетевой уровень

Он управляет связью в сети между двумя взаимодействующими машинами (не программами, а именно машинами). На этом уровне идёт работа с сетевыми адресами и маршрутизацией данных. Все подробности работы с низ лежащими уровнями также скрыты здесь. X.25 является самым известным протоколом этого уровня, не относящимся к TCP/IP.

Канальный уровень

На этом уровне идёт формирование данных для их передачи в сеть, а также их расшифровка после получения. Кроме того, он управляет доступом к физической среде передачи данных, здесь происходит синхронизация использования одной физической среды несколькими машинами, а также обнаружение и исправление ошибок.

Физический уровень

Все самые "приземлённые" параметры физической среды передачи данных (электрические, механические и др.) определены на этом уровне. Он отвечает за физическую связь между двумя устройствами передачи данных (сетевыми картами, модемами и т.д.). Самыми известными протоколами физического уровня являются ISDN (Integrated Services Digital Network), V.24, ATM.

Каждый протокол общается только с таким же протоколом на удаленной машине. Теоретически, он не заботится о протоколах других уровней (ниже или выше него). Ему есть дело только до его собеседника по сети.

Однако каждый уровень вовлечен в процесс общения двух программ друг с другом. Поэтому существуют соглашения о том, как передавать данные между соседними уровнями. Эти соглашения

называются *интерфейсом*, причём сколько разных уровней взаимодействует друг с другом, столько существует интерфейсов.

Данные от одной программы к другой проходят на передающей машине путь от уровня прикладных программ до физического уровня, а на принимающей - в обратном порядке. Уровни не знают, как работают их соседи, однако с помощью *интерфейсов* они умеют передавать и принимать от них данные. Разделение на уровни минимизирует затраты на модификацию протоколов каждого из уровней. Например, появление новой прикладной программы никак не затрагивает протоколы доступа к сети, а новая сетевая карта может быть установлена без переписывания прикладных программ.

Несмотря на то, что модель OSI очень хорошо подходит для теоретических исследований, на практике такое подробное деление уровней не всегда удобно. Примером хорошего компромисса между теорией и практикой является семейство протоколов TCP/IP. Отдельные его протоколы могут обслуживать сразу несколько уровней модели OSI. О том, какая модель сети используется в Internet и почему, пойдет речь дальше.

2. Семейство протоколов TCP/IP

Настало время пояснить, что же представляет из себя семейство протоколов TCP/IP. Во-первых, название TCP/IP расшифровывается как Transfer Control Protocol/Internet Protocol. Вообще говоря, из него можно понять, что это - пара протоколов (TCP и IP). Но это далеко не так. TCP и IP входят в это семейство и являются его главными, но не единственными протоколами. Скорее, TCP/IP - это идеология построения сетей, выраженная в протоколах. Они обладают следующим набором свойств:

- открытые (доступные для использования) стандарты протоколов, широко поддерживаемые разными вычислительными платформами и операционными системами. Они идеально подходят для интеграции разного аппаратного и программного обеспечения, даже если не нужен выход в Internet.
- независимость от аппаратного обеспечения сети передачи данных. TCP/IP может работать и объединять вместе сети, построенные на Ethernet, Token Ring, X.25, телефонных линиях связи и вообще на любых типах носителей, передающих данные.
- общая схема адресации, которая позволяет любому TCP/IP-устройству адресовать единственным образом любое другое устройство в сети. Даже если эта сеть - Internet.
- стандартизованные протоколы широко используемых пользовательских программ.

Итак, теперь стало понятнее, что же такое TCP/IP. Конечно, это семейство протоколов, как любое сложное и претендующее на целостность образование, не может обойтись без своей идеологии - модели передачи данных и архитектуры. Рассмотрим их, сравнивая с уже знакомым нам стандартом OSI.

2.1. Архитектура семейства, её отличия от модели OSI

Как уже говорилось, протоколы семейства TCP/IP не следуют строго модели OSI. Они разбиты на *четыре* уровня, в отличие от семиуровневой модели OSI. Ниже дано описание этой четырёхуровневой модели и её сопоставление со стандартной семиуровневой.

Уровень модели TCP/IP	Уровень модели OSI
Уровень прикладных программ Состоит из прикладных программ и процессов, использующих сеть и доступных пользователю. В отличие от модели OSI, прикладные программы сами стандартизуют представление данных.	**Уровень прикладных программ** **Уровень представления данных**
Транспортный уровень Обеспечивает доставку данных от компьютера к компьютеру. Кроме того, на этом уровне существуют средства для поддержки логических соединений между прикладными программами. В отличие от транспортного уровня модели OSI, в функции транспортного уровня TCP/IP не всегда входят контроль за ошибками и их коррекция. TCP/IP предоставляет два разных сервиса передачи данных на этом уровне. Протокол TCP обеспечивает все вышеперечисленные функции, а UDP - только передачу данных.	**Сеансовый уровень** **Транспортный уровень**
Межсетевой уровень Работает с дейтаграммами, адресами, выполняет маршрутизацию и "прикрывает" транспортный уровень от общения с физической сетью. Однако, в отличие от сетевого уровня модели OSI, этот уровень не устанавливает соединений с другими машинами. Межсетевой протокол (IP, Internet Protocol) выполняет все функции этого уровня в TCP/IP.	**Сетевой уровень**
Уровень доступа к сети Состоит из подпрограмм доступа к физической сети. Модель TCP/IP не разделяет два уровня модели OSI - канальный и физический, а рассматривает их как единое целое. На этом уровне редко создаются протоколы, входящие в семейство TCP/IP, потому что в документации протоколов канального уровня обычно говорится о том, как их может использовать IP.	**Канальный уровень** **Физический уровень**

Таблица 2.1.1: Соответствие уровней TCP/IP уровням OSI

Так же, как и в модели OSI, в TCP/IP данные проходят путь от верхнего уровня (прикладных программ) к нижнему (доступа к сети) при передаче данных и обратно при их получении. Каждый уровень TCP/IP добавляет к исходной порции данных свою контрольную информацию для обеспечения правильной доставки. Эта контрольная информация называется заголовком, потому что помещается перед посылаемыми данными. При передаче каждый следующий уровень рассматривает порцию данных, пришедшую от предыдущего, как единое целое и помещает перед нею свой заголовок. Добавление контрольной информации на каждом из уровней называется оформлением пакета.

Когда данные переданы по сети и получены самым нижним уровнем (доступа к сети), происходит обратный процесс. Каждый уровень вырезает свой заголовок из порции данных, а после этого передаёт оставшуюся часть следующему (более высокому) уровню. Таким образом, при прохождении порции данных снизу вверх (то есть при её расшифровке), она рассматривается уже не как единое целое, а как совокупность заголовка и некоторой информации.

Каждый уровень имеет свои собственные структуры данных. Они не зависят друг от друга и каждый уровень использует свой набор понятий для описания этой структуры. Кроме того, разные протоколы транспортного уровня (TCP и UDP) также по-разному называют полученные и переданные данные.

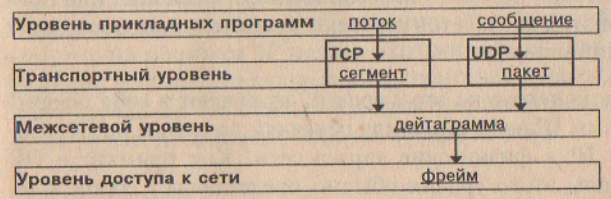


Рисунок 2.1.2: Терминология сетевой модели TCP/IP

На рисунке 2.1.2 показана терминология, используемая разными уровнями, относящаяся к передаваемым данным. Программы, использующие TCP, называют данные *потоком*, в то время как прог-

раммы, использующие UDP, называют их *сообщением*. TCP, передавая данные дальше, называет их *сегментом*, а UDP - *пакетом*. Межсетевой уровень разбивает все данные на блоки и называет их *дейтаграммами*. Несмотря на то, что TCP/IP использует множество различных типов физических сетей, каждая из которых имеет свою терминологию, мы будем называть данные, которые должны быть переданы, *фреймами*.

Давайте более подробно рассмотрим функции каждого из уровней протокола TCP/IP, начиная от уровня доступа к сети и заканчивая уровнем прикладных программ.

2.2. Уровень доступа к сети

Уровень доступа к сети является самым нижним уровнем в иерархии протокола TCP/IP. Протоколы на этом уровне обеспечивают систему средствами для передачи данных другим устройствам в сети. Они определяют, как использовать сеть для передачи дейтаграмм IP. В отличие от протоколов более высоких уровней, протоколы этого уровня должны знать детали физической сети (структуру пакетов, систему адресации и т.д.), чтобы правильно оформить передаваемые данные.

TCP/IP разработана таким образом, чтобы протоколы более высоких уровней не зависели от протоколов нижних уровней. Для уровня доступа к сети протоколы IP, TCP, UDP и т.д. являются протоколами более высокого уровня. Благодаря этому при появлении новых сетевых аппаратных средств необходима только разработка новых протоколов доступа к сети. Поэтому существует множество протоколов этого уровня - по одному на каждый стандарт сети (например, протокол передачи пакетов IP по сети Ethernet или протокол передачи этих же пакетов через последовательный порт, через модемное соединение).

Функции, выполняемые на этом уровне, включают в себя оформление дейтаграмм IP во фреймы для передачи по сети и преобразование адресов IP в физические адреса сети. Как принято в ОС UNIX, протоколы этого уровня обычно представляют собой комбинацию драйверов устройств и соответствующих программ.

2.3. Межсетевой уровень

Над уровнем доступа к сети находится межсетевой уровень. Протокол Internet (Internet Protocol, IP, описан в RFC 791) является сердцем всего TCP/IP и самым важным протоколом этого уровня. Все протоколы на более высоких уровнях используют IP для доставки данных. Все данные TCP/IP проходят через IP, независимо от пункта назначения.

2.3.1. Протокол IP

Функции этого протокола включают в себя:

- определение *дейтаграмм*, базовых блоков данных, передаваемых по Internet
- определение схемы *адресации* Internet
- обмен данными между транспортным уровнем и уровнем доступа к сети
- *маршрутизацию* дейтаграмм
- *разбиение и обратную сборку* дейтаграмм

Прежде чем приступить к детальному описанию каждой функции протокола, рассмотрим некоторые его характеристики. Во-первых, IP является протоколом *без логического установления соединения*. Это значит, что он не обменивается контрольной информацией (называемой оповещением) для установления соединения, перед тем как передать данные. IP оставляет другим протоколам право устанавливать соединения - этим занимается либо протокол TCP, либо сами прикладные программы.

Протокол IP также *не занимается обнаружением и исправлением ошибок*. Этот протокол называют ненадежным, потому что он не содержит никакого кода для выполнения этих функций. Это не означает, что на протокол IP нельзя возложить выполнение каких-либо функций вообще. Он аккуратно доставит ваши данные в сеть и они будут переданы. Но отвечать за проверку полученных данных должные другие протоколы, если это вообще нужно.

Дейтаграммы

Протоколы TCP/IP были созданы для передачи данных через ARPANET, которая является сетью с коммутацией пакетов. Пакет - это блок данных, который передается вместе с информацией, необ-

ходимой для его корректной доставки. Каждый пакет перемещается по сети независимо от остальных.

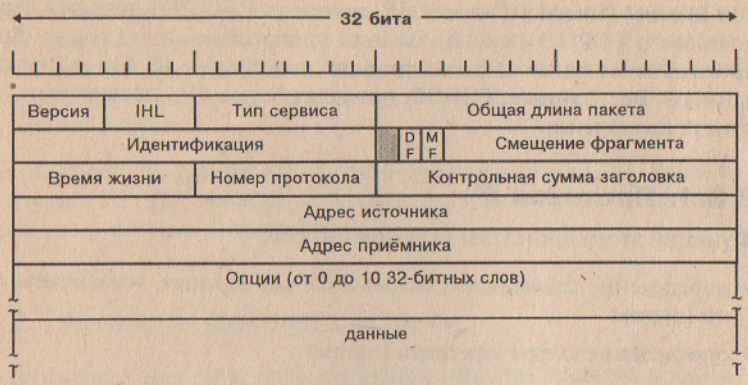


Рисунок 2.3.1.1: Формат дейтаграммы IP

Дейтаграмма - это пакет протокола IP. На рисунке 2.3.1.1 показан её формат. Контрольная информация занимает первые пять или шесть 32-битных слов дейтаграммы. Это её заголовок (header). По умолчанию его длина равна пяти словам, шестое является дополнительным. Для указания точной длины заголовка в нём есть специальное поле - *длина заголовка* (IHL, Internet Header Length).

В поле *версия* записана версия протокола IP. Сейчас стандартом является версия номер 4, поэтому практически во всех дейтаграммах, блуждающих в Internet, в этом поле стоит четвёрка. Это поле нужно для того, чтобы можно было отличать друг от друга дейтаграммы разных версий протокола IP.

Поле *тип сервиса* теоретически предоставляет хосту возможность выбирать, какой тип сервиса он хочет получить от протоколов сетевого уровня. В этом поле можно указать различные комбинации надёжности и скорости доставки. Быстрая доставка важна для оцифрованного голоса, в то время как для передачи файлов важнее передача без ошибок, чем скорость.

IP доставляет дейтаграммы по адресу, записанному в поле адрес назначения (Destination Address) в слове 5 заголовка. Адрес назначения - это стандартный 32-битный адрес протокола IP. Он определяет номер сети назначения и номер хоста в этой сети. Если хост находится в локальной сети, то пакет доставляется сразу по месту назначения. Если нет, то пакет сначала отправляется на межсетевой маршрутизатор. Маршрутизатор - это устройство, передающее пакеты между различными сетями. Процесс выбора маршрутизатора называется, как ни странно, *маршрутизацией*. Протокол IP отвечает за маршрутизацию каждого пакета.

Маршрутизация дейтаграмм

В современной сетевой терминологии различают маршрутизаторы и шлюзы. Первые перемещают данные между разными сетями, а вторые - между разными протоколами. Однако мы будем пользоваться обоими словами для обозначения одного и того же - маршрутизаторов.

TCP/IP оперирует только двумя типами сетевых устройств: маршрутизаторами и хостами. Маршрутизаторы перемещают данные между разными сетями, а хосты нет. Но если хост подключен больше чем к одной сети (это так называемый multihomed хост), он может передавать пакеты как в одну сеть, так и в другую. Однако он не будет делать этого для других компьютеров в сети, т.е. не будет заниматься маршрутизацией пакетов. В этом его отличие от маршрутизатора.

Компьютерные системы могут передавать данные только внутри той сети, к которой они подключены. Поэтому передача дейтаграмм из одной сети в другую (сети могут быть даже несовместимыми физически) идёт через шлюзы - от одного к другому. Внутри хоста (как передающего, так и принимающего) данные проходят путь от уровня прикладных программ до уровня доступа к сети (или наоборот). Дейтаграммы, которые переправляет шлюз, поднимаются только до межсетевого уровня. На этом уровне протокол IP, узнавая адрес получателя данных (на протяжении всего пути следования данных этот адрес не меняется - меняются промежуточные машины), принимает решение отправить дейтаграмму в одну из сетей, к которым подключен.

На рисунке 2.3.1.2 показано, как используются шлюзы для ретрансляции пакетов. Внутри хостов (будем называть их конечными системами) данные проходят все четыре уровня, а внутри шлюзов (промежуточных систем) они поднимаются только до межсетевого уровня, где протокол IP и принимает решение о маршрутизации.

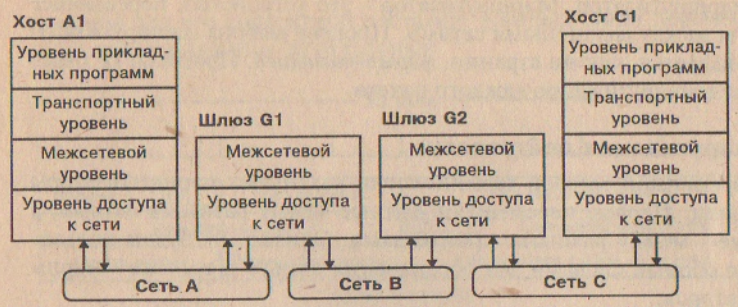


Рисунок 2.3.1.2: Маршрутизация с использованием шлюзов

Поскольку данные (дейтаграммы) могут передаваться непосредственно только между устройствами, объединёнными в одну физическую сеть, то в нашем примере передача пакетов от хоста A1 хосту C1 осуществляется в несколько приёмов, через шлюзы G1 и G2. Сначала хост A1 посылает данные шлюзу G1, с которым объединён сетью A. Шлюз G1 по сети B пересылает данные шлюзу G2. Шлюз G2, наконец, пересылает данные хосту C1, благодаря тому, что они оба подключены к сети C. Очень важно, что хост A1 ничего не знает о шлюзах, находящихся дальше шлюза G1 (то есть о шлюзах, с которыми не объединён в одну сеть). Он просто посылает данные, адресованные C1 (или вообще любому хосту в сети B или C), этому шлюзу и полностью полагается на его способность правильно ретранслировать пакеты в соответствующую сеть. Точно так же хост C1 не имеет ни малейшего понятия о шлюзе G1 и полностью полагается на свой локальный шлюз G2 при пересылке данных хостам в сети A или B.

"Но почему так много работы возложено на шлюзы?" - спросите вы. И окажетесь неправы. Дело в том, что единственное, что должен сделать шлюз - это выбрать соседний шлюз или хост (разделяющий с ним одну сеть) и передать ему данные. Что с ними произойдёт дальше - не его забота. Либо они уже дошли по назначению, либо ими займётся следующий шлюз. В выборе правильного соседа-получателя данных и состоит функция маршрутизации протокола IP.

С помощью поля *время жизни* (TTL, Time to live) можно указать, как долго дейтаграмма может прожить в сети. При прохождении через маршрутизаторы это поле уменьшается, как правило, на 1 (если дейтаграмма задержалась в маршрутизаторе больше, чем на 1 секунду, значение поля уменьшается больше чем на 1). Как только

время жизни уменьшится до 0, маршрутизатор обязан уничтожить дейтаграмму и не передавать её дальше.

Фрагментация дейтаграмм

Когда Internet Protocol готовит данные, полученные от транспортного уровня, он может обнаружить, что сегмент (или пакет) данных не может быть передан по сети целиком. Тогда он принимает решение разбить его на несколько частей - фрагментов. Этот процесс, естественно, называется *фрагментацией*.

Аналогичная проблема может возникнуть и у шлюза. Дело в том, что каждая физическая сеть имеет так называемый *максимальный передаваемый блок* (MTU, maximum transmission unit), больше которого она ничего передать не может. Разные сети имеют разные MTU, поэтому шлюз может принять решение о фрагментации и в этом случае. Когда промежуточная машина либо хост назначения получают несколько дейтаграмм, являющихся фрагментами одного и того же сегмента (пакета) данных, происходит обратный процесс - *сборка*. Транспортный уровень этой машины (или хоста) получает свою порцию данных в нетронутом виде - так, как её передал исходный хост.

Каждый фрагмент оформляется как дейтаграмма. Во время фрагментации IP разбивает порцию данных так, чтобы во всех фрагментах, кроме последнего, соблюдалось правило выравнивания данных по границе восьми октетов. В поле заголовка *смещение фрагментации* (Fragmentation Offset) записано, какой части дейтаграммы принадлежит данный фрагмент. Для того, чтобы показать, что эта дейтаграмма является фрагментом, используются еще два поля. Первое - это бит *More Fragments* (MF). Он устанавливается в 1, если данный фрагмент не является последним, и равен нулю в противном случае. Второе поле - *идентификация* (Identification). Оно служит для того, чтобы не перепутать фрагменты из *разных* дейтаграмм. Ведь при сборке пакета из разрозненных фрагментов межсетевому уровню надо определить, какие из них компоновать вместе. Поэтому в поле *идентификация* указывается уникальный номер дейтаграммы. Используя тройку Идентификация-Адрес отправителя-Адрес получателя, IP сможет собрать дейтаграмму из фрагментов. Сам протокол IP рекомендует в качестве Идентификации устанавливать номер, предоставленный транспортным уровнем (если есть такая возможность).

В специальных случаях можно указать, чтобы промежуточные машины не разбивали дейтаграмму на фрагменты. Это делается установкой в 1 поля *Don't Fragment* (DF).

Передача дейтаграмм транспортному уровню

Когда протокол IP какого-нибудь хоста получает дейтаграмму, он должен передать данные, содержащиеся в ней, соответствующему протоколу транспортного уровня. Для определения протокола используется поле *номер протокола* (Protocol Number) из третьего слова заголовка дейтаграммы. Каждый протокол транспортного уровня имеет свой уникальный номер, по которому его и отыскивает протокол IP.

2.3.2. Протокол ICMP

Неотъемлемой частью IP и межсетевого уровня является Протокол контрольных сообщений Internet (ICMP, Internet Control Message Protocol). Он пользуется сервисом передачи дейтаграмм протокола IP для приема и посылки собственных сообщений. Эти сообщения выполняют следующие функции:

Контроль за трафиком

Иногда возникает ситуация, когда количество получаемых дейтаграмм становится слишком большим, так что хост или шлюз не успевают их обработать. Тогда отправителю дейтаграмм посылается сообщение *Source Quench Message*, которое просит временно приостановить пересылку.

Обнаружение недостижимых получателей

Если система обнаруживает, что по какой-либо причине не может отправить дейтаграммы получателю, она оповещает отправителя. В этом случае передаётся *сообщение о недоступности получателя* (Destination Unreachable Message). Если недостижимый получатель - хост, то сообщение посылает промежуточный шлюз; если протокол транспортного уровня или прикладная программа - сам хост-получатель.

Изменение маршрута дейтаграмм

Шлюз может послать *сообщение о переназначении* (Redirect Message), чтобы попросить отправителя дейтаграмм использовать другой шлюз (возможно, из-за того, что он быстрее). При этом оба шлюза

и отправитель должны быть соединены напрямую. Это связано с идеологией маршрутизации дейтаграмм в Internet.

Проверка связи с хостом

С помощью ICMP можно узнать, есть ли связь с каким-нибудь устройством - шлюзом или хостом. Для этого достаточно послать сообщение Echo Message. Дело в том, что если хост или шлюз получают такое сообщение, они обязаны послать его обратно отправителю. Поэтому если ответ приходит - значит устройство работает и связь с ним есть. Если нет ответа - то с этим устройством лучше пока не связываться - всё равно не получится. Кстати, команда ping ОС UNIX работает используя именно это сообщение.

По сути, протокол ICMP является надстройкой над IP, которая выполняет информационные функции, контроль за работой, и диагностирование ошибок.

2.4. Транспортный уровень

Прямо над межсетевым уровнем находится транспортный уровень. Протоколы этого уровня пользуются для передачи данных протоколом IP и, как правило, сами используются прикладными программами. Два очень важных протокола транспортного уровня будут в центре внимания этой главы: *Протокол контроля передачи* (TCP, Transmission Control Protocol) и *Протокол пользовательских дейтаграмм* (UDP, User Datagram Protocol). TCP обеспечивает надежную доставку данных с автоматическим контролем и исправлением ошибок. UDP обеспечивает быструю доставку дейтаграмм, но без логического соединения.

2.4.1. Протокол UDP

Этот протокол даёт прикладным программам прямой доступ к сервису передачи дейтаграмм, похожему на сервис, предоставляемый IP. Это позволяет программам обмениваться сообщениями с минимальной загрузкой сети.

UDP - это ненадёжный (в том же смысле, в каком ненадёжен протокол IP, то есть UDP не занимается обнаружением и исправлением ошибок) протокол без логического соединения. То есть для того, чтобы отправить пакет, протокол не связывается с адресатом. Внутри компьютера протокол доставит данные, конечно, без ошибок. UDP использует 16-битные номера портов отправителя и по-

лучателя для идентификации соответствующего процесса (подробнее о том, что такое порты, читайте в параграфе 3.3.2). На рисунке 2.4.1.1 показан заголовок сообщения UDP.

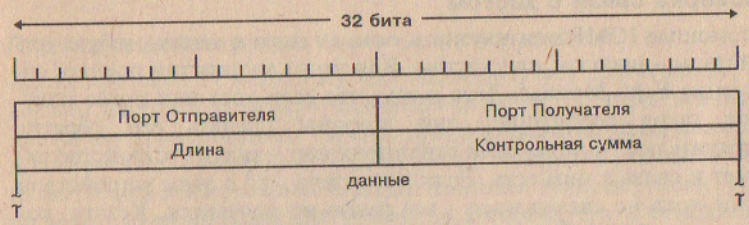


Рисунок 2.4.1.1: Формат сообщения UDP

Здесь длина - это длина (в октетах) всего пакета UDP - данных вместе с заголовком. Контрольная сумма считается как поразрядное дополнение от псевдо-заголовка и пакета (вместе с данными). Данные выравниваются по границе двух октетов (т.е., если надо, добавляется нулевой октет). Если при вычислении получился ноль, передаются единицы. Ноль передаётся в том случае, когда программа не считает контрольную сумму. Кстати, нужно сказать, что такое псевдо-заголовок. Он состоит из адреса отправителя, адреса получателя, номера протокола (для UDP это 17) и длины сообщения UDP (не выравненного по двум октетам). Его структура:

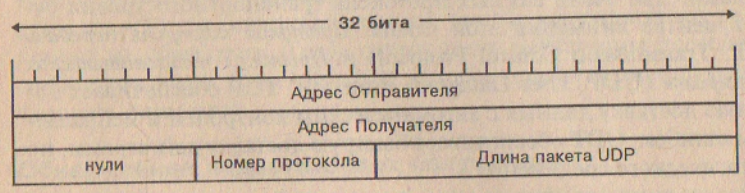


Рисунок 2.4.1.2: Формат псевдо-заголовка

2.4.2. Протокол TCP

Программы, которым нужна надёжная доставка данных, используют TCP. Этот протокол проверяет, что данные доставлены аккуратно и в правильной последовательности. TCP характеризуется тремя свойствами - это надёжный протокол, реализующий связь с логическим соединением, который рассматривает данные как непрерывный поток октет (байт). Давайте более подробно рассмотрим каждое из этих свойств: надёжность, связь с логическим соединением, данные как непрерывный поток октет.

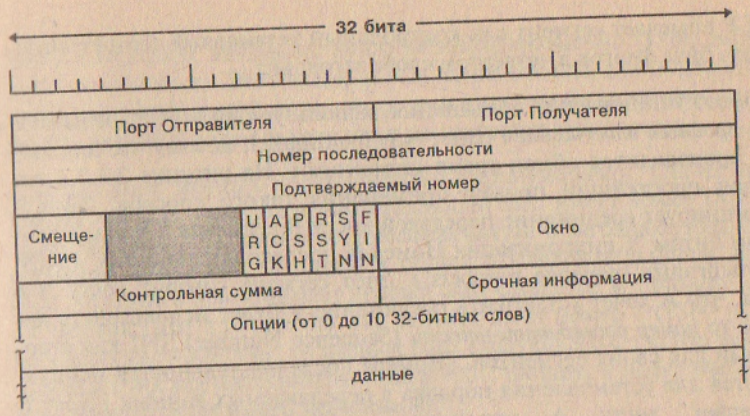


Рисунок 2.4.2.1: Формат сегмента TCP

TCP обеспечивает надёжность с помощью механизма с хитроумным названием - Положительное подтверждение с повторной передачей (PAR, Positive Acknowledgment with Retransmission). Проще говоря, система, пользующаяся PAR, передаёт данные снова и снова до тех пор, пока от удалённой системы не придёт подтверждение об их корректной доставке. Единица данных, которой обмениваются системы с протоколом TCP, называется сегментом. Каждый сегмент содержит контрольную сумму, по которой Получатель может проверить корректность доставки данных. Если данные дошли нормально, то он посылает отправителю положительное подтверждение. Если нет - данные просто игнорируются, а отправитель через некоторое время повторно посылает тот сегмент данных, о получении которого не было подтверждения.

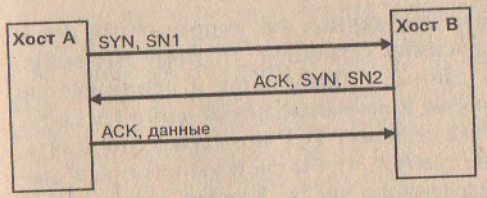


Рисунок 2.4.2.2: Пример тройного оповещения при установлении соединения протоколом TCP

TCP поддерживает связь с логическим соединением. Хосты обмениваются контрольной информацией, называемой *оповещением* (handshaking), чтобы установить контакт до передачи информации.

TCP помечает сегмент как контрольный установкой соответствующего бита флагов в четвёртом слове заголовка.

Способ установления соединения, используемый в TCP, называется *тройным оповещением* (three-way handshake), потому что для этого производится обмен тремя сегментами. На рисунке 2.4.2.2 показан простейший пример применения такого способа. Хост А инициирует соединение передачей хосту В сегмента с установленным битом "Синхронизация Номеров последовательностей" (SYN, "Synchronize sequence numbers"). Этот сегмент говорит хосту В о том, что А хочет установить соединение и будет использовать какой-то *номер последовательности* (Sequence Number) SN1 как стартовый для своих сегментов. (Номера последовательностей используются для установления порядка в передаваемых данных.) Хост В отвечает хосту А сегментом с установленными битами "Подтверждение" (ACK, "Acknowledgment") и SYN. Этим сегментом В подтверждает получение сегмента от А и информирует, с какого Номера последовательности SN2 будет начинать свою передачу. Наконец, хост А передаёт сегмент, которым подтверждает получение сегмента от В, и в котором содержатся первые реальные данные.

После такого обмена протокол TCP хоста А уверен, что удалённый протокол работает и готов к получению данных. Как только устанавливается это соединение, можно обмениваться данными. Когда протоколы решат завершить обмен, они опять обмениваются тремя сегментами с установленным битом "Нет больше данных для передачи" (FIN, "No more data from sender"). Таким образом они закрывают соединение. Весь этот обмен (до и после передачи данных) необходим для логического соединения между двумя системами.

TCP рассматривает передаваемые данные как непрерывный поток байт, а не как набор независимых сегментов. Поэтому ему важно знать порядок, в котором данные отправляются и принимаются. Поля *номер последовательности* и *подтверждаемый номер* (Acknowledgment number) в заголовке сегмента TCP позволяют это делать. Стандарт протокола TCP не требует, чтобы системы начинали отсчёт байт с какого-то определённого числа. Каждая система сама выбирает такое число. Чтобы работать с данными в правильном порядке, системы должны знать, с какого номера их собеседник начнёт передачу. Системы, устанавливающие соединение, синхронизируют свои начальные номера, передавая SYN-сегменты во время *тройного оповещения* (здесь речь идёт именно о том процессе, который описан выше). Поле *номер последовательности* этих сегментов содержит *начальный номер последовательности* (ISN, Initial Sequence Number). Этот номер и есть точка отсчёта для нумерации последовательностей байтов, которыми системы будут обмениваться. Обычно ISN равен 0, хотя это и не требуется стандартом.

Каждый байт данных последовательно пронумерован. Отсчёт начинается с ISN, поэтому первый реальный байт имеет номер ISN+1 (обычно просто 1). *Номер последовательности* в заголовке каждого сегмента определяет позицию в общей последовательности данных, которую в ней занимает первый байт сегмента. К примеру, если первый байт в общем потоке данных имел номер 1 (ISN=0), и передано уже 4000 байт, то первый байт следующего сегмента имеет в этой же последовательности номер 4001. Значит, и *номер последовательности* будет иметь значение 4001.

Сегмент ACK выполняет сразу две функции - положительное подтверждение и управление потоком данных. Подтверждение информирует отправителя о том, сколько байт данных было получено корректно. Управление потоком данных позволяет отправителю определить, сколько данных ещё получатель может принять. В поле *подтверждаемый номер* содержится *номер последовательности* последнего принятого байта. Стандарт протокола говорит, что не обязательно подтверждать получение каждого сегмента. Но если получено подтверждение какого-то байта, оно означает, что вся последовательность (все предыдущие сегменты), включая этот байт, принята корректно. К примеру, если первый байт имел номер 1, и было получено 2000 байт, Подтверждаемый номер должен иметь значение 2000.

Поле *окно* (Window) содержит количество байт, которое получатель способен принять сразу, без подтверждения. Оно означает, что отправитель может передавать сегменты данных до тех пор, пока общее количество посланных байт не превысит значения этого поля. Получатель контролирует поток данных от отправителя, изменяя размер окна. Нулевое окно говорит о том, что отправителю нужно приостановить передачу до тех пор, пока не будет получен сегмент с ненулевым значением этого поля.

На рисунке 2.4.2.3 показан поток данных TCP, начинающийся с байта номер 1 (то есть ISN имеет значение 0). Получатель принял и подтвердил получение 2000 байт, поэтому *подтверждаемый номер* имеет значение 2000. У получателя также есть место для принятия 6000 байт, поэтому он устанавливает в поле *окно* значение 6000. Отправитель передаёт сегмент с 1000 байт и *номером последовательности* 4000. Отправитель не получил подтверждения о получении

данных, начиная с 2001 байта, но продолжает отправку до тех пор, пока не выйдет за пределы окна. Если он заполнит окно и не получит подтверждения, то через некоторое время начнёт снова передавать данные с первого неподтвержденного байта. На рисунке 2.4.2.3 повторная передача может начаться с 2001 байта, если не будет получено подтверждения. Эта процедура позволяет удостовериться, что данные корректно получены на другом конце соединения.

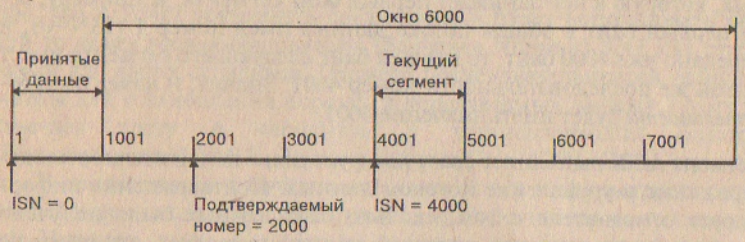


Рисунок 2.4.2.3: Поток данных протокола TCP

TCP также отвечает за доставку данных, которые он получил от IP, к соответствующей программе. Программа, к которой привязаны данные, опознаётся 16-битным номером порта. *Порт отправителя* и *порт получателя* содержатся в первом слове заголовка сегмента TCP. Корректная доставка данных прикладным программам является важной функцией транспортного уровня. Но о том, как это делается, мы поговорим дальше (параграф 3.3.1).

Другие поля заголовка сегмента TCP нас не интересуют, поэтому скажем о них вкратце. В поле *смещение* содержится размер заголовка в 32-битных словах (то есть длина заголовка всегда кратна четырём байтам). Это связано с тем, что в заголовке присутствует поле *опции* переменной длины. По значению поля *смещение* протокол определяет, откуда начинаются реальные данные. *Контрольная сумма* считается по тем же правилам, что и в протоколе UDP. Поле *срочная информация* совместно с битом URG в 4-ом слове заголовка используются для передачи срочных данных, обработка которых должна быть осуществлена в первую очередь. Что это за данные и как на них реагировать - дело прикладной программы, а не протокола TCP, просто он обрабатывает их в первую очередь, забывая про номера последовательностей, окна и т.д.

2.4.3. Сравнение протоколов TCP и UDP

Зачем в TCP/IP два базовых протокола транспортного уровня? Какой из них выбрать?

Дело в том, что есть довольно большая разница между этими двумя протоколами - TCP и UDP. TCP обеспечивает надёжную доставку данных с обнаружением и исправлением ошибок и с установлением логического соединения. UDP же ничего этого не делает, он просто отправляет пакеты с данными, не заботясь об их доставке. Зато загрузка сети становится меньше из-за отсутствия логического соединения. UDP стоит выбрать в следующих случаях:

- обмен данными невелик. Из-за этого загрузка сети, вызванная установлением логического соединения и обеспечением надёжности доставки, может превышать объём повторной посылки всех данных.
- программа осуществляет обмен по принципу "запрос-ответ" В этом случае ответ может считаться подтверждением получения запроса. Если ответа нет в течение какого-то промежутка времени, программа просто посылает запрос ещё раз.
- программа реализует свои собственные механизмы надёжной доставки данных и ей не нужен протокол, который будет заниматься тем же самым. Естественно, что ещё один уровень подтверждения получения данных и установления логического соединения будет снижать эффективность.

Разумеется, выбор протокола - процесс творческий. Программист должен сам решить (возможно, поэкспериментировав с каждым из этих протоколов), какой из них лучше для его программы.

2.5. Уровень прикладных программ

На самом верху архитектуры семейства протоколов TCP находится *уровень прикладных программ*. Все процессы этого уровня пользуются протоколами транспортного уровня для обмена данными по сети. Существует много протоколов прикладных программ. Большинство из них обеспечивают какой-то сервис для пользователя, хотя есть и "протоколы в себе", никакой непосредственной пользы не представляющие (для пользователя, конечно). Мы рассмотрим на примере простого шлюза архитектуру всего TCP/IP и, заодно, несколько протоколов уровня прикладных программ.

2.5.1. Примеры прикладных программ

Стоит рассказать о прикладных программах, наиболее часто используемых на этом уровне. Некоторые из них стали стандартами Internet, другие настолько популярны, что признаны стандартами de facto. Мы будем характеризовать программы их назначением и протоколами нижних уровней, на которые они опираются. Разумеется, программы являются всего лишь реализацией того или иного протокола уровня прикладных программ, поэтому будем перечислять только сами протоколы.

Протоколы, опирающиеся на TCP

TELNET (Network Terminal Protocol), Протокол сетевого терминала, разработан для удалённого доступа к компьютерам сети (remote login).

FTP (File Transfer Protocol), Протокол передачи файлов, используется для интерактивной передачи файлов между компьютерами сети.

SMTP (Simple Mail Transfer Protocol), Простой протокол передачи почты. Основной протокол для обмена почтой, использующийся в Internet.

Протоколы, работающие с помощью UDP

DNS (Domain Name Service), Служба имён доменов, или просто Служба именования. С помощью этого протокола устанавливается взаимно однозначное соответствие между IP-адресами сетевых устройств и их именами. Об этом протоколе и о системе именования вообще речь пойдёт дальше.

RIP (Routing Information Protocol), Протокол информации о маршрутизации. Маршрутизация данных (поиск путей их передачи от хоста-отправителя к хосту-получателю) в Internet является одной из важнейших функций семейства протоколов TCP/IP. RIP используется сетевыми устройствами для обмена информацией о маршрутизации. Маршрутизация тоже будет подробно рассмотрена дальше.

NFS (Network File System), Сетевая файловая система. С помощью этого протокола компьютеры могут совместно использовать файлы, разбросанные по сети.

Существуют также протоколы, по сути являющиеся протоколами уровня прикладных программ, но не использующие в своей работе

ни один из протоколов транспортного уровня. В качестве примера можно привести EGP (Exterior Gateway Protocol), Внешний протокол шлюзов. Это ещё один протокол маршрутизации. Но он не работает ни с TCP, ни с UDP, а напрямую с протоколом IP.

2.5.2. Иерархия протоколов на примере простого шлюза

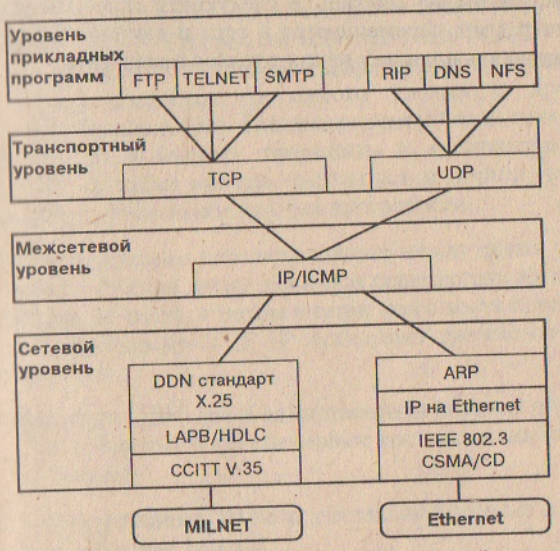


Рисунок 2.5.2.1: Иерархия протоколов в простом шлюзе

Чтобы лучше понять, как же всё-таки организована структура TCP/IP, рассмотрим иерархию протоколов в простом шлюзе, то есть в сетевом устройстве, способном передавать пакеты из одной физической сети в другую. Схема, показанная на рисунке 2.5.2.1, однако, является всего лишь наглядным упрощением такой структуры. Её цель - показать взаимоотношения протоколов в сетевом устройстве. На самом верху схемы расположены протоколы уровня прикладных программ типа FTP или TELNET. От них идут стрелки к протоколам более низких уровней, на которые они опираются. Под прикладными программами находятся протоколы транспортного уровня: TCP и UDP. Они работают непосредственно с IP. Все данные в системе идут через IP. Все протоколы ниже мы рассматриваем как протоколы уровня доступа к сети. Однако, как это видно, они могут находиться на разных уровнях модели OSI

(например, протоколы доступа к X.25 разбиты на три уровня - сетевой, канальный и физический). Но для TCP/IP это не играет никакого значения. В самом низу картинки находятся конкретные физические сети - Milnet и Ethernet. Работа шлюза заключается в том, что он принимает пакеты, расшифровывает их и отправляет в соответствующую физическую сеть.

В этой главе мы рассмотрели структуру TCP/IP, семейства протоколов, на котором построена Internet. В следующей главе мы увидим, как IP-дейтаграммы перемещаются в сети и как системы узнают, куда их отправлять и какому из протоколов передавать.

3. Адресация, маршрутизация и мультиплексирование

В предыдущей главе мы рассмотрели архитектуру семейства протоколов TCP/IP. Теперь нам известно, что TCP/IP состоит из четырех уровней. В этой главе мы более подробно рассмотрим, как данные перемещаются от одного уровня к другому и между компьютерами в сети. Главными темами этой главы будут: структура адресов в Internet, принципы и архитектура маршрутизации данных, а также номера протоколов и портов, используемые для правильной доставки данных программам.

Чтобы правильно передать данные между двумя сетевыми устройствами в Internet, нужно сначала переместить их по сети к соответствующему хосту, а потом внутри этого хоста передать соответствующей программе. TCP/IP использует три схемы для выполнения этих задач:

1. **Адресация.** IP-адреса единственным образом определяют каждый хост в Internet и обеспечивают доставку данных к соответствующему хосту.
2. **Маршрутизация.** Шлюзы доставляют данные к соответствующей сети в составе Internet.
3. **Мультиплексирование.** Номера протоколов и портов помогают доставлять данные соответствующим программам внутри хоста.

Все эти функции необходимы для общения двух программ в Internet. Давайте каждую из них рассмотрим более подробно.

3.1. Система адресации

Начнём с адресации. Основная её идея - чтобы каждый компьютер (точнее, каждый сетевой интерфейс) в Internet имел свой собственный уникальный адрес. Благодаря этому всё, что вам нужно знать для общения с этим устройством - его адрес.

3.1.1. IP-адреса

Протокол Internet перемещает данные по сети в виде дейтаграмм. Каждая из них доставляется по адресу, содержащемуся в поле Адрес назначения заголовка. Этот адрес является стандартным 32-битным адресом IP. Он содержит всю необходимую информацию для однозначной идентификации сети или хоста в этой сети.

Адрес IP состоит из двух частей - адреса сети и адреса хоста в этой сети. Но форматы этих частей могут различаться в разных IP-адресах. Точнее, может варьироваться количество бит, отведённое под эти части, но остаётся неизменной их интерпретация - это номера, уникальным образом определяющие сеть в Internet (адрес сети) или хост внутри этой сети (адрес хоста). Существует три класса IP-адресов, определяющих длину каждой части. Класс зашифровывается первыми одним-тремя битами IP-адреса. Используются следующие правила определения класса адреса и, соответственно, длины каждой его части:

- Первый бит адреса IP равен 0. Тогда это адрес сети класса A. Следующие 7 бит определяют адрес сети, а остальные 24 - адрес хоста в этой сети. Может существовать меньше чем 127 сетей класса A, но зато каждая такая сеть может состоять из миллионов хостов. (Почему меньше чем из 127? Ответ на этот вопрос дальше в этой же главе.)

Первые два бита адреса IP равны 1 0. Тогда это адрес сети класса B. Следующие 14 бит определяют адрес сети, остальные 16 - адрес хоста. Таких может быть уже больше, но в них содержится меньшее количество хостов.

- Первые три бита адреса IP - 1 1 0. Тогда это адрес сети класса C. Следующие 21 бит определяют адрес сети, оставшиеся 8 - адрес хоста. Количество сетей класса C может превышать миллион, но в каждой из таких сетей может быть не больше 254 хостов.

- Первые три бита адреса IP - 1 1 1. Тогда это адрес специального класса D. Эти адреса реально не ссылаются ни на одну из сетей. Они используются для идентификации групп компьютеров (multicasting), использующих общий протокол (в отличие от компьютеров, использующих общую сеть). Соответственно, адреса этого класса состоят только из адреса группы (не адреса сети и адреса хоста), который занимает 29 бит.

В Internet принята достаточно удобная система записи IP-адресов. Их обычно пишут в виде четырёх чисел, разделённых точкой. Каждое из таких чисел - это десятичная (или иногда 16-ричная запись) соответствующего байта в адресе. Конечно, в такой записи не видно явным образом, какому классу принадлежит этот адрес, но по первому номеру это можно определить. Первый номер адреса каждого класса лежит в определённом диапазоне:

- Меньше 128 - этот адрес принадлежит классу A. Первый байт адреса является адресом сети, остальные три - адресом хоста.

- От 128 до 191 - адрес класса B. Первые два байта определяют адрес сети, другие два - адрес хоста.

- От 192 до 223 - адрес класса C. Первые три байта - адрес сети, последний байт - адрес хоста.

- От 224 до 239 - адрес класса D, так называемый адрес группы (multicast address). Мы не будет рассматривать такие адреса.

Такая система записи называется "точечной" нотацией. Существуют и другие способы записи IP-адреса (например, последовательная запись всех байт адреса в 16-ричном представлении без разделителей), но мы будем пользоваться только "точечной" системой записи, потому что она очень наглядна и общепринята.

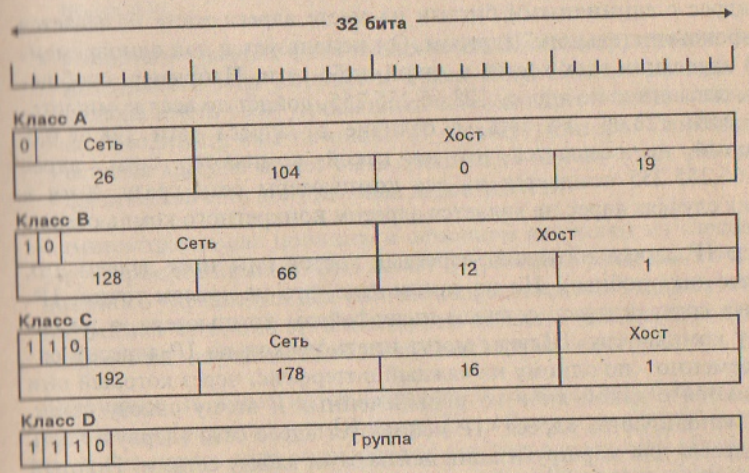


Рисунок 3.1.1.1: Структура IP-адреса

На рисунке 3.1.1.1 показано, как структура адреса изменяется в зависимости от его класса. Например, адресом класса A является 26.104.0.19. Первый бит этого адреса - 0, поэтому он интерпретиру-

ется как хост 104.0.19 в сети 26. Аналогично (соответственно своим классам) интерпретируются адреса других классов.

Не все адреса сетей или хостов могут использоваться по назначению. Некоторые из них являются зарезервированными. Например, есть два специальных адреса сетей класса A - это 0 (то есть сеть с номером 0) и 127 (сеть с номером 127). Первый из них обозначает *маршрут по умолчанию* (default route). Он используется протоколом IP для упрощения обработки информации о маршрутизации. Второй специальный адрес сети - 127 - называется *зацикленным адресом* (loopback address). Этот адрес позволяет сетевым программам обращаться к хосту, на котором они работают, точно так же, как и к любой другой сетевой машине.

Кроме специальных адресов сетей существуют и специальные адреса хостов. В сетях всех классов адреса хостов 0 и 255 (то есть со всеми нулевыми или единичными битами в адресе хоста) зарезервированы. Нулевой адрес хоста не адресует какую-нибудь конкретную машину - он адресует всю сеть целиком. Например, адрес 26.0.0.0 обозначает сеть номер 26, а адрес 128.66.0.0 - сеть номер 128.66. Такие адреса используются в таблице маршрутизации для ссылок на целые сети, а не на отдельные хосты.

IP адрес с единичными битами на месте адреса хоста называется "широковещательным" адресом. Он используется для одновременной адресации всех хостов в какой-либо сети. Например, сообщение, посланное по адресу 128.66.255.255, дойдёт до всех компьютеров сети 128.66. То есть, в отличие от адреса сети 128.66.0.0, который не ссылается ни на какой компьютер сети, адрес 128.66.255.255 ссылается на все компьютеры сети сразу. Хотя в обоих случаях адрес не является адресом конкретного компьютера.

Часто IP адреса называют адресами хостов (мы тоже делаем это, просто так удобнее). Но на самом деле, это не совсем точно. IP-адреса соответствуют сетевым интерфейсам компьютера, а не самому компьютеру. Шлюзы могут иметь несколько IP-адресов одновременно - по одному на каждый интерфейс, через который они общаются с какой-либо из подключенных к этому шлюзу сетей. Как используются адреса? IP использует адрес сети (первую часть IP-адреса) для маршрутизации дейтаграмм между сетями. Полный адрес, включая адрес хоста, используется для окончательной доставки дейтаграммы до компьютера, когда она уже достигла нужной сети.

3.1.2. Подсети

Стандартная структура адреса IP может быть модифицирована - биты адреса хоста можно использовать как дополнительные биты для адреса подсети. Правда, такая модификация может быть проведена только в пределах одной сети и не затронет весь Internet. Фактически, можно передвинуть границу между адресом сети и адресом хоста - в сторону адреса хоста, но не обратно. Таким образом можно увеличить количество сетей, уменьшив максимальное количество хостов в каждой из них. Биты, которые были отняты у адреса хоста, определяют сеть внутри исходной сети, называемую подсетью (subnet).

Когда может понадобиться разбиение сети на подсети? Существуют две главные группы причин - топологические и административные. Первые могут включать в себя:

- преодоление ограничения на длину сетей. Некоторые сети имеют ограничение по длине используемого кабеля. Например, максимальная длина "толстого" кабеля в сети Ethernet может составлять 500 метров. Если нужно организовать более длинную сеть, то приходится соединять отрезки кабелей с помощью маршрутизаторов. В этом случае каждый отрезок кабеля будет отдельной подсетью.

- соединение разных физических сетей. Например, нужно объединить вместе сети Ethernet и Token-Ring. Каждая из этих сетей должна иметь свой уникальный сетевой адрес, чтобы можно было объединить их с помощью IP-маршрутизатора. Уникальность сетевых адресов может быть обеспечена за счёт того, что каждая физическая сеть получит свой адрес подсети.

Административные причины в основном вытекают из следования структуре организации. Она может потребовать раздельного и независимого управления для разных подразделений. Тогда будет проще разбить сеть этой организации на подсети, чем каждому из подразделений получать свой собственный адрес сети.

Разбиение на подсети осуществляется наложением *битовой маски подсети* (subnet mask) на IP-адрес. Если бит в маске равен 1, значит этот бит адреса интерпретируется как бит адреса сети. Если бит в маске равен 0, соответствующий ему бит IP-адреса принадлежит адресу хоста. О разбиении на подсети знают только локальные машины сети (работающие в ней). Для всех остальных машин в

Internet эта сеть является единым целым, и адреса машин в ней интерпретируются как обычно.

Например, маска подсети, соответствующая сети класса B, выглядит как 255.255.0.0. Самая распространённая маска подсети расширяет сетевую часть адреса класса B на один байт. Эта маска - 255.255.255.0. При наложении этой маски на адрес класса B он интерпретируется так: первые два байта адреса определяют сеть класса B; третий байт определяет адрес подсети; последний байт определяет номер хоста в этой подсети.

Вовсе не обязательно использовать маски, выравненные на границу байта. Часто используются маски, которые проводят границу между адресом сети и хоста не по границе байта, а где-нибудь посередине. Например, можно определить маску 255.255.255.224. В этом случае три первых бита последнего байта адреса будут частью сетевого адреса, а последние пять бит - частью адреса хоста (напомню, что в двоичном представлении число 224 записывается как 11100000). Ниже приведены примеры наложения масок подсети на конкретные IP-адреса хостов и их расшифровка.

IP-адрес	Маска подсети	Расшифровка
128.66.12.1	255.255.255.0	Хост 1 в подсети 128.66.12.0
130.97.16.132	255.255.255.192	Хост 4 в подсети 130.97.16.128
132.90.132.5	255.255.240.0	Хост 4.5 в подсети 132.90.128.0
18.20.16.91	255.255.0.0	Хост 16.91 в подсети 18.20.0.0

Таблица 3.1.2.1: Примеры расшифровки IP-адресов при наложении маски подсети

3.1.3. Протоколы разрешения адресов на примере адресации в Ethernet (ARP, RARP)

В самом начале этой главы вскользь было упомянуто о том, что IP-адрес имеет сетевой интерфейс, посредством которого устройство обменивается IP-дейтаграммами с другими компьютерами. Однако об этом адресе знает межсетевой уровень, но не имеет представления сетевой уровень. Это происходит из-за того, что в каждой физической сети используется своя схема адресации, отличная от адресации в Internet. Причем сколько разных типов физических сред - столько типов адресации. Сразу возникает вопрос - как адресовать данные, если известен только IP-адрес их получателя? Для этого нужно решить задачу отождествления физических адресов сетевых устройств и их IP-адресов.

Мы рассмотрим один из способов решения этой задачи на примере Протокола разрешения адресов (ARP, Address Resolution Protocol). Этот протокол транслирует IP-адреса в адреса Ethernet. Эти адреса состоят из 6 байт и гарантируют уникальность, то есть в мире нт двух карт Ethernet с одинаковыми адресами. Для трансляции IP-адреса в Ethernet-адрес ARP внутри себя содержит таблицу соответствия одних адресов другим. Эта таблица строится автоматически. Когда ARP получает запрос на трансляцию, он сначала просматривает свою таблицу. Если IP-адрес в ней уже есть, то возвращается соответствующий ему адрес Ethernet. Если адреса в таблице ещё нет, то ARP выполняет следующие действия:

1) всем компьютерам сети Ethernet рассылается широковещательный запрос (в сетях Ethernet это возможно) с IP-адресом машины, Ethernet-адрес которой нужно узнать.

2) машины получают запрос. Если одна из них в IP-адресе узнаёт свой собственный адрес, она отсылает обратно (по Ethernet-адресу спрашивающей машины) пакет со своим Ethernet-адресом.

3) ARP получает пакет с искомым Ethernet-адресом, заносит его в свою таблицу трансляции, и отвечает на исходный запрос.

Конечно, может случиться и так, что ни одна машина не опознала свой IP-адрес в широковещательном запросе. В таком случае ARP отвечает, что машина с таким адресом недоступна в этой сети.

Но каким образом обеспечивается уникальность адресов Ethernet? Для ответа на этот вопрос стоит поподробнее рассказать об адресах Ethernet и о том, откуда они берутся. Итак, адрес Ethernet состоит из 6 байт (это больше, чем адрес IP). Но структура адреса Ethernet гораздо проще структуры последнего. Первые три байта в нём - это номер производителя сетевого адаптера Ethernet, а последние 3 байта - внутренний номер, присвоенный этой карте производителем. Поэтому, по сути, адрес Ethernet просто представляет собой серийный номер сетевого адаптера. Причем ни один производитель не имеет права повторять номера сетевых карт и присваивать себе номер другого производителя. Вот почему все сетевые адаптеры Ethernet в мире имеют разные адреса. Для наглядности приведу пример адреса Ethernet и его расшифровки:

Сетевая карта Ethernet имеет адрес 8:0:20:b:4a:71 (числа записаны в шестнадцатеричной системе). Это карта номер b:4a:71, её изготовителем является фирма номер 8:0:20. Код соответствует фирме Sun Microsystems Inc.

Вообще говоря, существуют огромные таблицы, по которым можно определить производителя практически любой сетевой карты Ethernet. Но главное не это. Основное преимущество адреса Ethernet состоит в том, что он уникален. И заботиться об этом не нужно никому, кроме фирмы-производителя.

Протокол обратного разрешения адресов (RARP, Reverse Address Resolution Protocol) используется для перевода адресов Ethernet обратно в IP-адреса. Это нужно, например, когда хост загружает операционную систему с удалённой машины. Для этого он должен сначала узнать свой IP-адрес. Поэтому при загрузке он посылает широковещательный запрос вида "Скажите IP-адрес хоста с таким-то адресом Ethernet". RARP-сервер (если он вообще есть в этой сети и "слушает" запросы) должен в ответ послать IP-адрес.

В этой главе мы разобрались с адресом IP, то есть адресом сетевого интерфейса, посредством которого устройство в сети общается с другими устройствами; структурой этого адреса; разбиением логических сетей IP на подсети, а также тем, как IP-адрес превращается в физический адрес устройства, на примере протокола ARP и сетей Ethernet. В следующем разделе мы оторвёмся от нашей машины и проследим, какой путь проходят данные в сети Internet и как выбирается этот путь. То есть следующей темой будет...

3.2. Маршрутизация

Как только данные собираются покинуть машину, сразу встаёт вопрос - куда они попадут дальше? Самый простой ответ на него - в физическую сеть. Он является правильным, но явно нас не устроит, потому что из физической сети их должен кто-то забрать. Кто будет это делать? Кто будет отвечать за то, чтобы они дошли до адресата?

3.2.1. Ещё раз о маршрутизации как одной из функций протокола IP

Вспомним, что сетевой интерфейс устройства, работающего в Internet, имеет адрес IP. За этот адрес полностью отвечает межсетевой уровень. Почему бы на него не возложить функцию выбора адреса того устройства, которое получит данные следующим? "Велика проблема," - скажете вы. Посылай данные по тому адресу,

который указан в пакете TCP или UDP, и всё само дойдёт до нужного места. Как бы не так. Ведь это самое место может быть настолько удалено от нашей машины, что мы даже представить себе не сможем, сколько мытарств придётся испытать данным по дороге. Загвоздка состоит в том, что сетевые протоколы передачи данных, находящиеся на уровне ниже межсетевого, могут передавать их только в пределах прямого физического соединения. То есть приходится выбирать - кто следующий получит данные. Причём он должен иметь возможность не только их принять, но и переслать дальше.

Итак, одна из функций протокола IP - выбор машины (точнее, IP-адреса машины), находящейся в одной физической сети с нашей, с расчётом на то, что она сможет переслать данные дальше так, чтобы они в конце концов достигли адресата.

Проще говоря, протокол IP должен уметь выбрать следующего в цепочке передач данных, которая и составляет маршрут. Причём, что очень важно, на всех машинах действует один и тот же механизм выбора. Основан он на применении так называемой таблицы маршрутизации.

3.2.2. Таблица маршрутизации

Это на самом деле таблица, в которой содержатся знания о том, какую машину выбирать следующей в зависимости от адреса получателя. Точнее, интерес представляет только сетевая часть адреса, потому что с адресом хоста в сети назначения будут разбираться на месте.

Итак, предположим, что нам (мы - это протокол IP) нужно переслать некоторые данные по вполне определённому адресу. Причём не важно, откуда они у нас - или это наши данные, или же нас попросили их передать дальше (то есть мы работаем шлюзом). В первую очередь нужно выделить сетевую порцию адреса назначения. Если после этого обнаруживается, что адрес сети - локальный (то есть совпадающий с нашим сетевым адресом), то на весь адрес накладывается маска подсети. Если при сравнении полученного адреса подсети с нашим адресом подсети опять получаем совпадение - всё очень просто. Пересылаем данные хосту, работающему в одной физической сети с нами, и больше не заботимся ни о чём. Может возникнуть ситуация, когда всё совпало, но такого компьютера нет: он выключен или вообще не существовал. Тогда посыла-

ется ICMP-сообщение о недоступности хоста и пакет уничтожается.

Если на одном из этапов сравнения адресов получилось несовпадение, то нужно посмотреть в таблицу маршрутизации. Она может быть построена либо вручную, либо с помощью специальных протоколов маршрутизации, но в конечном счёте процесс выбора следующей машины в маршруте - это поиск по таблице маршрутизации.

Посмотрим, как она устроена. Для нас сейчас будут важны только четыре поля из неё (хотя в реальных системах таких полей больше). Пример таблицы маршрутизации приведён в таблице 3.2.2.1:

Адрес назначения	Шлюз	Флаги	Интерфейс
127.0.0.1	127.0.0.1	H	lo0
default	128.66.12.1	G	le0
128.66.12.0	128.66.12.3		le0
128.66.1.0	128.66.1.5		le1

Таблица 3.2.2.1: Пример таблицы маршрутизации

Структура таблицы очень простая: в первой колонке находится адрес назначения, который может быть либо адресом сети, либо подсети (если сетевая часть этого адреса совпадает с локальным адресом сети), либо хоста. Вы уже, наверное, заметили, что в первых двух строках таблицы на рисунке стоят специальные адреса. О них рассказывалось в разделе "IP-адреса", но стоит пояснить их смысл ещё раз. Адрес 127.0.0.1 означает адрес нашей машины. Это фиктивный адрес, не имеющий ничего общего с реальным IP-адресом, но, используя его, можно общаться с другими локальными программами как с сетевыми. Адрес default есть не что иное как 0.0.0.0, то есть маршрут по умолчанию. По его названию можно догадаться, что это тот маршрут, за который протокол IP хватается как утопающий за соломинку, если не найдёт в таблице ничего подходящего. Но не раньше. О том, что такое подходящий маршрут, будет рассказано ниже.

Вторая колонка состоит из адресов шлюзов. В третьей колонке стоят специальные флаги, характеризующие маршрут. Буква G (gateway) означает, что для достижения адреса назначения из первой колонки придётся воспользоваться услугами удалённого шлюза, чей адрес находится во второй колонке. Отсутствие буквы G говорит о том, что машина сама может доставить данные по адресу назначения. Это значит, что машина непосредственно подключена

к сети назначения. Примером могут служить третья и четвёртая строчки из таблицы маршрутизации в таблице 3.2.2.1: машина с IP-адресом 128.66.12.3 подключена к сети 128.66.12.0, поэтому она сама может переправлять пакеты любой из машин этой сети. Во-вторых, эта же машина подключена и к сети 128.66.1.0 и имеет второй адрес - 128.66.1.5, поэтому может отправлять пакеты в эту сеть. Флаг H информирует, что в первой колонке стоит адрес хоста, а не сети. В нашем примере адресом хоста является 127.0.0.1, т.е. локальный адрес машины.

Кроме флагов G и H в реальной таблице маршрутизации могут быть такие флаги как U (up) - флаг, показывающий, что данный маршрут действует; D - признак того, что маршрут был добавлен (или изменён) системой во время работы.

Четвертая колонка содержит названия интерфейсов - буквенно-цифровые обозначения интерфейсов ко внешним (в отдельных случаях - к внутренним) устройствам. Этими устройствами могут быть сетевые платы (если их много, то интерфейс к каждой из них имеет своё название), последовательный и параллельный интерфейсы и т.д.. В примере, приведённом в таблице 3.2.2.1, присутствуют три имени интерфейсов - lo0, le0, le1. Первый из них, как можно догадаться - внутренний, так как соответствует локальному адресу машины. Кстати, такой маршрут - не выходящий за пределы системы - называется loopback route. Отсюда и название интерфейса - lo0.

Следующие два интерфейса - le0 и le1 - соответствуют двум сетевым картам машины, то есть интерфейсам доступа к двум физическим сетям. Буква e в названии говорит о том, что сетевые карты - типа Ethernet.

У любопытных читателей, не испугавшихся структуры таблицы маршрутизации, может возникнуть вопрос: а откуда она берётся? Оказывается, эту таблицу можно менять. Самый простой способ - с помощью специальных команд (программ) добавлять, удалять или изменять маршруты. Но если изменения в топологии сети происходят достаточно часто или нужен автоматизм в изменении маршрутов? В этом случае применяют специальные протоколы маршрутизации. Они следят за изменениями маршрутов и могут менять таблицы маршрутизации на лету. Об этих протоколах будет рассказано в разделе 5.

3.2.3. Пример работы маршрутизации

На рисунке 3.2.3.1 показано, как работает маршрутизация в нашей учебной сети. Межсетевые уровни хостов и шлюза заменены небольшим фрагментом таблицы маршрутизации, в которой показаны только адреса сетей назначения и шлюзы, используемые для доступа к этим сетям. Кстати, в роли шлюза на этом рисунке выступает шлюз, чью таблицу маршрутизации мы только что рассмотрели.

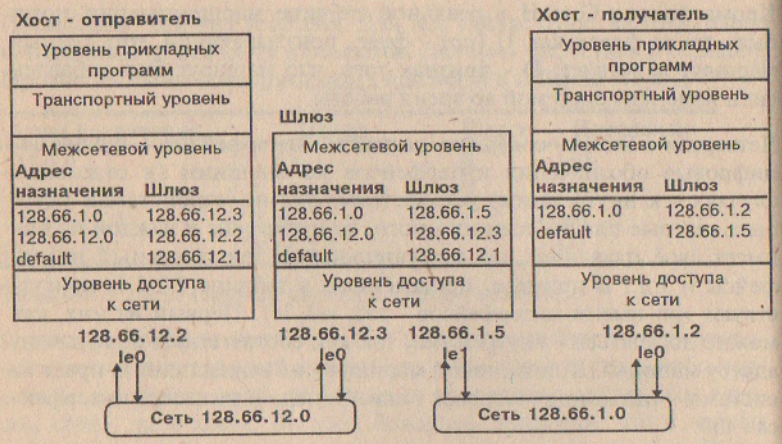


Рисунок 3.2.3.1: Пример использования таблиц маршрутизации

Когда хост-отправитель (128.66.12.2) передаёт данные хосту-получателю (128.66.1.2), то он сначала определяет, что сетевая часть адреса 128.66.1.2 (это адрес класса B) совпадает с сетевой частью его собственного адреса. Затем, накладывая маску подсети (в нашем случае это 255.255.255.0) на этот адрес, обнаруживает, что полный адрес сети получателя - 128.66.1.0. В таблице маршрутизации хоста-отправителя записано, что данные, получателем которых является хост с полным номером сети 128.66.1.0, должны быть переданы шлюзу 128.66.12.3. Этот шлюз, получив дейтаграммы от хоста-отправителя из сети 128.66.12.0, обнаружив, что они предназначены для хоста из сети 128.66.1.0, направляет их прямо хосту получателю, благо он и к этой сети имеет прямой доступ (правда, через другой интерфейс). Кстати, заметим, что в таблице маршрутизации любой машины шлюзами могут быть только те машины, которые имеют прямой доступ к сети данной машины.

Обратите внимание, что 128.66.12.1 является шлюзом по умолчанию (default) как для 128.66.12.2, так и для 128.66.12.3. Но из-за того, что 128.66.1.2 не имеет прямого доступа к сети 128.66.12.0, он имеет другой шлюз по умолчанию.

Подведём итоги рассмотрения таблицы маршрутизации. Итак,

- таблица маршрутизации не содержит маршрутов из конца в конец (полных маршрутов);
- маршруты - это адреса следующих шлюзов, находящихся на пути к сети назначения (вот что значит подходящий маршрут, о котором говорилось раньше);
- хост опирается на локальные шлюзы для доставки данных, а шлюзы - на другие шлюзы, с которыми они имеют прямое соединение. Исключение из этого правила - если хост-отправитель и хост-получатель находятся в одной логической сети (совпадают полные адреса сетей обоих хостов), то шлюз не нужен - данные отправляются напрямую;
- перемещаясь от шлюза к шлюзу, дейтаграммы в конце концов достигнут шлюза, имеющего доступ к сети назначения. Тогда именно этот шлюз переправит данные хосту-получателю.

3.3. Мультиплексирование

...наконец, после долгих мытарств и блужданий по шлюзам, дейтаграммы добрались до нужного хоста и попали в лапы протокола IP. А что дальше? Казалось бы, как просто - проверь контрольную сумму и передавай их наверх. Совершенно верно. Но кому наверх? Ведь там UDP, TCP, может ещё какой-нибудь EGP поживают. А ещё выше - море прикладных программ. Правда с ними разбираются перечисленные протоколы, но, всё-таки, как они это делают? Оказывается, выручает их метод с магическим названием "мультиплексирование", о котором дальше и пойдёт речь.

Вернёмся к хосту-отправителю. Он должен собрать данные от множества прикладных программ и передать их транспортным протоколам. Затем данные от нескольких транспортных протоколов должны попасть к единственному протоколу IP. Процесс объединения нескольких источников данных в один поток называется мультиплексированием (multiplexing). Как видим, в TCP/IP муль-

типлексирование происходит по меньшей мере дважды - сначала на уровне транспортных протоколов, затем на межсетевом уровне.

Данные доставлены хосту-получателю. Теперь их нужно распределить по соответствующим программам. Процесс разбиения единого источника данных на несколько независимых потоков называется демультиплексированием (demultiplexing). Каким образом система понимает, кому передавать данные (то есть как разбивать единый источник на потоки)? Для этого нужно всего лишь перенумеровать все протоколы и прикладные программы и в каждый пакет записывать информацию о том, для какой программы или протокола он предназначен. IP для идентификации транспортных протоколов использует номера протоколов (protocol numbers), а транспортные протоколы используют номера портов (port numbers) для идентификации прикладных программ.

Некоторые номера протоколов и портов зарезервированы для так называемых общеизвестных программ (well-known services). Это стандартные сетевые протоколы, такие как FTP и TELNET, которые используются повсеместно. То есть везде в Internet для доступа к ним используются одни и те же номера протоколов и портов. Эти номера описаны в RFC "Присвоенные номера" (Assigned Numbers).

3.3.1. Номера протоколов

Номер протокола формально - это байт, записываемый в третье слово заголовка дейтаграммы (пакета, формируемого протоколом IP). Его значение определяет протокол транспортного уровня, которому после расшифровки должны быть переданы данные из дейтаграммы.

В ОС UNIX номера протоколов записаны в специальном файле /etc/protocols. Он имеет очень простой формат:

официальное_название_протокола номер_протокола псевдоним_протокола

Этот файл устанавливает соответствия между номерами протоколов и официальными названиями, которые знает протокол IP. Приведём небольшой пример такого файла:


```
# Комментарии начинаются с символа '#'
#
# Internet (IP) protocols #
ip      0       IP      # псевдо-номер протокола IP
icmp    1       ICMP    # протокол ICMP
ggp     2       GGP     # протокол Gateway-Gateway Protocol
tcp     6       TCP     # протокол TCP
udp     17      UDP     # протокол UDP
```

Как используется эта таблица? Когда протокол IP получает дейтаграмму и она оказывается предназначенной именно этой машине, то IP должен доставить полученные данные уровню выше. Просмотрев поле Протокол в заголовке дейтаграммы, он ищет соответствующую запись в таблице номеров протоколов. Например, если номер протокола равен 6, IP отбрасывает заголовок от дейтаграммы (он делает это всегда перед передачей данных уровню выше) и передаёт оставшиеся данные протоколу TCP. Если номер протокола - 17, данные достаются протоколу UDP. Кроме того, на транспортном уровне, кроме TCP и UDP, могут быть другие протоколы, пользующиеся IP непосредственно. Одним из таких протоколов является Протокол шлюз-шлюз (GGP, Gateway-gateway protocol), протокол маршрутизации.

В документе Присвоенные номера перечислено гораздо больше номеров протоколов, однако обычному хосту требуются лишь немногие из них. Именно поэтому в таблице может быть записана лишь небольшая часть таких номеров.

3.3.2. Порты

После того, как транспортный протокол получил данные от IP, он должен передать их ещё выше, соответствующей прикладной программе. Прикладные программы идентифицируются с помощью 16-битных номеров портов. В первом слове заголовка каждого сегмента TCP или пакета UDP записаны два номера порта - Порт отправителя и Порт получателя. Первый соответствует программе, пославшей данные, второй - программе, которой эти данные предназначены.

В UNIX-е номера портов описаны в файле /etc/services. Порты с номерами меньше 256 зарезервированы для общеизвестных программ (well-known services), таких как FTP и TELNET; порты с номерами от 256 до 1024 используются утилитами, первоначально разработанными только для UNIX, например rlogin или talk. Прав-

да, сейчас многие из этих программ есть и в других операционных системах.

Номера портов не являются уникальными среди разных транспортных протоколов, но они не могут повторяться в одном из них. То есть для TCP и UDP возможно существование программ с одинаковыми номерами портов, причём эти программы могут быть разными. Уникальна комбинация номера протокола и номера порта, именно она единственным образом определяет ту программу, которой должны быть доставлены данные.

Ниже приведён небольшой кусочек реального файла /etc/services. Его формат очень похож на формат файла, описывающего номера протоколов /etc/protocols. В каждой строке сначала записано официальное название программы, потом идёт пара номер порта/протокол, соответствующая этой программе. Номера портов записаны вместе с протоколами из-за того, что разные протоколы могут использовать одинаковые номера портов. Затем может идти список псевдонимов официальных названий программ.

```
# Комментарии начинаются с символа '#'
#
# Network services, Internet style
#
echo    7/udp
echo    7/tcp
systat  11/tcp
ftp     21/tcp
telnet  23/tcp
smtp    25/tcp    mail     # Simple Mail Transfer Protocol
name    42/udp    nameserver
whois   43/tcp    nicname
#
# Host specific functions
#
tftp    69/udp              # Trivial FTP
pop-2   109/tcp             # Post Office Protocol-2
nntp    119/tcp   usenet    # Network News Transfer Protocol
#
# UNIX specific services
#
login   513/tcp
shell   514/tcp   cmd
who     513/udp   whod      # who daemon
syslog  514/udp
talk    517/udp
route   520/udp   router routed
```

Эта таблица плюс /etc/protocols - вся информация, необходимая для доставки данных соответствующим прикладным программам.

Дейтаграмма прибывает из сети по IP-адресу, записанному в пятом слове её заголовка. Протокол IP использует номер протокола из третьего слова заголовка дейтаграммы для передачи данных "правильному" транспортному протоколу. Наконец, этот транспортный протокол передаёт данные прикладной программе, соответствующей номеру порта из первого слова данных, полученных от IP. На рисунке 3.3.2.1 показан этот процесс.

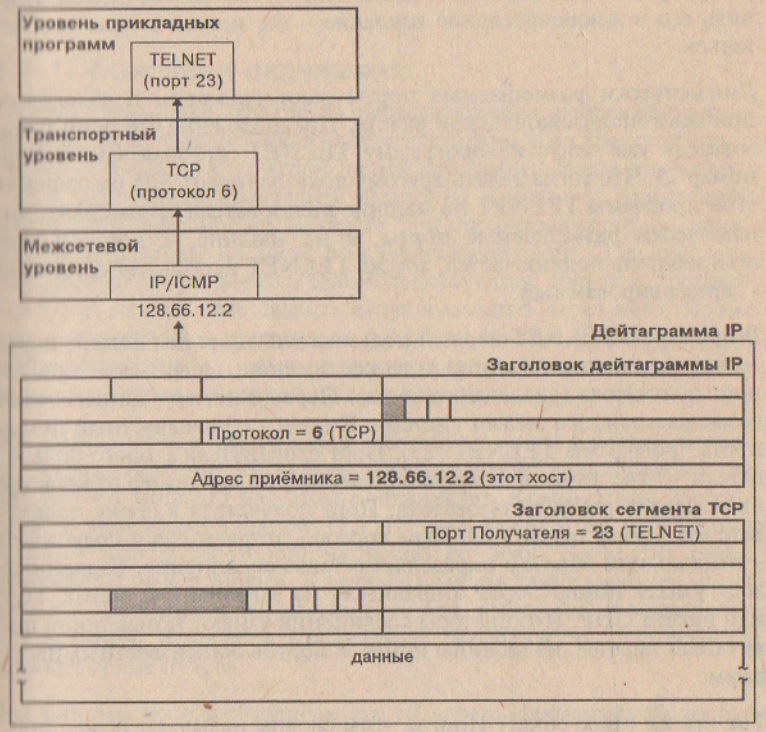


Рисунок 3.3.2.1: Пример процесса передачи данных с использованием номеров протоколов и портов

3.3.3. Сокеты

Номера портов общеизвестных программ позволяют хостам заранее знать, к какому порту на удалённой машине обращаться, если нужно вызвать ту или иную программу. К примеру, все машины предлагают обращаться к программе TELNET через порт 23.

Кроме зарезервированных портов есть ещё динамически размещаемые порты. Процессы могут запрашивать их во время работы. При получении такого запроса система должна выдать номер порта, который ещё не занят (то есть на данный момент не используемый каким-либо процессом для работы с одним из протоколов), и который не является зарезервированным. После окончания работы процесс должен освободить динамически размещённый порт. Зарезервированный порт освободить нельзя, так же как нельзя получить его в исключительное владение - им можно только пользоваться.

Динамически размещаемые порты дают гибкость, необходимую при многопользовательской работе. Предположим, что один пользователь уже запустил программу TELNET, которая заняла порт номер 23. Что тогда делать другому пользователю? Для разрешения этой проблемы TELNET на машине пользователя запрашивает динамически размещаемые порты, а на машине, к которой этот пользователь подключается, тот же TELNET использует один порт - зарезервированный.

В примере с TELNET первый пользователь знает два номера порта - порт отправителя (инициатора соединения, то есть его хоста) и порт получателя (удалённого хоста). Первый номер - динамически размещаемый, его выдаёт система. Второй - общеизвестный номер порта программы TELNET (23). Порт отправителя второго пользователя - тоже динамически размещаемый, но его номер отличается от номера первого пользователя. Порт получателя у обоих пользователей один и тот же - 23. Пара портов - отправителя и получателя - уникальным образом определяет сетевое соединение. Удалённый хост узнает номер порта отправителя из заголовка сегмента TCP или пакета UDP. Инициаторы соединений узнают номер порта получателя заранее, из таблицы номеров портов общеизвестных программ.

Так что же такое сокет? Иногда этим словом называют номер порта, но мы определим его как комбинацию адреса IP и номера порта (динамически размещаемого или зарезервированного - неважно). Сокет однозначно определяет сетевой процесс во всём Internet-е. Пара сокетов - хоста-получателя и хоста-отправителя - однозначно определяют сетевое соединение в рамках определённого протокола (например, TCP). Пример сокета - динамически размещённый порт с номером 3855 на хосте, имеющем адрес IP 128.66.12.2. Иногда такой сокет записывают как 128.66.12.2.3855.

3.4. Пример передачи данных

В этой главе мы подробно разберёмся, что же происходит во время передачи данных. Для этого вместе с ними мы попутешествуем между двумя общающимися прикладными программами на разных хостах.

3.4.1. Описание окружения

Давайте представим, что где-то существуют сети, изображённые на рисунке 3.4.1.1 (существование Internet можно и не представлять, а вот всё остальное будет игрой нашей фантазии). Наша задача состоит в том, чтобы проследить движение данных от прикладной программы, работающей на хосте host.ourcompany.com до прикладной программы на хосте host.aliencompany.com. Для простоты были выбраны программы, взаимодействующие только с протоколом UDP, поэтому все данные будут проходить через него. Кроме того, попросим эти программы посылать данные такого размера, чтобы дейтаграммы с ними не нужно было разбивать на отдельные фрагменты перед отправкой по сети Ethernet.

Все компьютеры в нашей схеме работают в сетях Ethernet. Внешние маршрутизаторы могут быть дополнительно подключены к сетям другого типа, но нас это не интересует. Компьютеры изображены в виде стека уровней TCP/IP, над каждым из них (кроме внешних маршрутизаторов) написано его имя. На хостах router изображены не все уровни. В реальной жизни такие машины могут быть как полноценными хостами, то есть иметь ещё и транспортный уровень, и уровень прикладных программ, но для нашего примера это не нужно.

Под каждым хостом подписан его IP-адрес. Некоторые из хостов подключены сразу к двум сетям, поэтому каждому сетевому адаптеру протокол IP ставит в соответствие отдельный адрес. Кроме IP-адреса, под каждым сетевым адаптером подписано название сетевого интерфейса, под которым он известен системе, и с которым реально будет общаться протокол IP. Как видно из рисунка, компьютеры в нашей сети (сверху) имеют IP-адреса класса C, в отличие от компьютеров в чужой сети (снизу), которые имеют адреса класса A. У нас используется маска подсети 255.255.255.0, а в чужой сети - 255.255.254.0 (число 254 в двоичном виде выглядит как 11111110), то есть чужой IP-адрес выглядит так: 8 первых бит - адрес сети, следующие 15 бит - адрес подсети, последние 9 бит - адрес

хоста. Разумеется, так он выглядит только для компьютеров, находящихся в IP-сети 120.0.0.0.

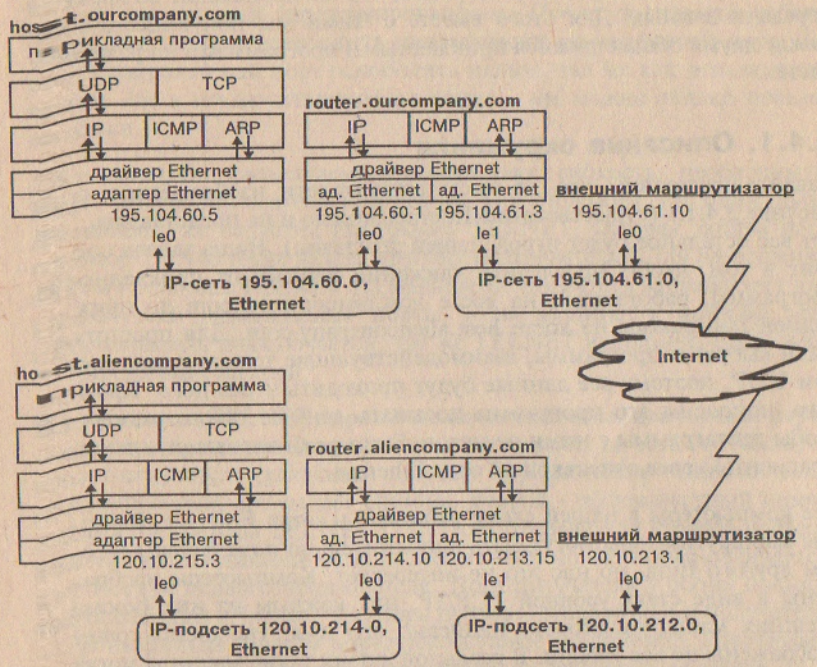


Рисунок 3.4.1.1: Схема, используемая в примере

Если внимательно посмотреть на IP-адреса сетей и хостов чужой сети, то может возникнуть вопрос - а почему они такие, почему третьи байты в адресах не совпадают? Всё очень просто. При наложении маски подсети 255.255.254.0 на IP-адреса хостов получаем такие адреса подсетей:

- 20.10.215.3 и 120.10.214.0 дают подсеть 120.10.214.0,
- 20.10.213.15 даёт подсеть 120.10.212.0.

3.4.2. Старт: от прикладной программы до сетевой карты

Итак, начнём. Прикладная программа на нашем хосте (host.ourcompany.com) хочет послать какие-то данные с помощью протокола UDP программе на хосте host.aliencompany.com. Протокол UDP выбран для простоты, хотя пример не сильно бы изменился, используй мы протокол TCP. Тогда перед посылкой пакета с данными произошёл бы процесс тройного оповещения (описанный в параграфе 2.4.2), а заголовок сегмента TCP имел бы много ненужной нам сейчас информации.

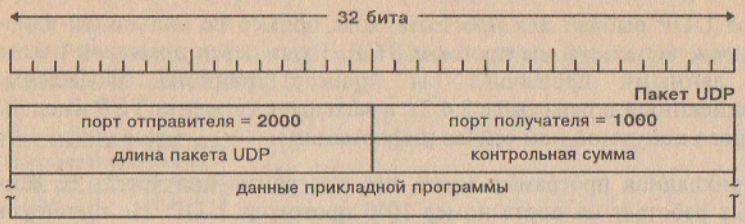
Прикладная программа знает, что программа-получатель её данных работает на порте номер 1000 протокола UDP. Не пытайтесь искать реальных программ, работающих на этом порте - даже если вы найдёте что-то, то это, как пишут в титрах некоторых фильмов, будет всего-лишь случайным совпадением. Теперь нашей программе нужно узнать, какой IP-адрес имеет её собеседник. Для этого она просит систему DNS узнать адрес хоста с именем host.aliencompany.com (о том, что такое имя, DNS, и как сопоставить имени IP-адрес, вы можете узнать из раздела IV). DNS через некоторое время отвечает, что IP-адрес этого хоста 120.10.215.3. Но это ещё не всё.

Перед тем, как начать передачу, программа должна "зарегистрироваться" у протокола UDP и получить динамически выделенный номер порта, то есть сокет. Это необходимо для того, чтобы удалённая программа могла нам ответить, а сделать она это может, только если знает IP-адрес нашей машины, используемый протокол и номер порта. Получив номер порта (пусть это будет порт 2000 протокола UDP), программа наконец-то начинает передачу данных.

Стоит заметить, что реально процесс подготовки передачи происходит несколько сложнее, но главная идея состоит в том, чтобы получить доступ к нужному протоколу (UDP или TCP) и номер порта.

Программа передаёт протоколу UDP IP-адрес удалённой машины, номер порта и данные. Протокол UDP заполняет заголовок пакета, заполняет псевдо-заголовок, считает и записывает в заголовок контрольную сумму, присоединяет к пакету данные и передаёт пакет и псевдо-заголовок протоколу IP так, как это показано на рисунке 3.4.2.1.

В псевдо-заголовке в поле Протокол мы поставили 17, потому что это номер протокола UDP. Чтобы лучше понять, почему в поля заголовка и псевдо-заголовка были занесены те или иные значения, советуем вам сравнить их с описанными в параграфе 2.4.1.



Псевдозаголовок

IP-адрес отправителя (195.104.60.5)		
IP-адрес получателя (120.10.215.3)		
нули	Протокол=17 (UDP)	длина пакета UDP

Рисунок 3.4.2.1: Информация, передаваемая протоколом UDP протоколу IP

Передав информацию, UDP о ней тут же забывает (в отличие от протокола TCP, который помнит о ней до тех пор, пока не удостоверится, что она целиком и без ошибок получена удалённой машиной).

Протокол IP должен произвести несколько действий, прежде чем отправить полученный от UDP пакет (теперь этот пакет рассматривается как данные целиком, потому что IP не разбирается, где в пакете заголовок, а где данные прикладной программы). IP из псевдо-заголовка узнаёт IP-адрес удалённой машины. Но кому отправить данные дальше? Ведь в общем случае удалённая машина может быть действительно удалённой, может находиться в локальной сети, а может быть самой машиной-отправителем (вспомните про адреса типа 127.0.0.1)!

Для решения этого вопроса IP выполняет два действия. Сначала он ищет в таблице маршрутизации IP-адрес следующего получателя. Потом он узнаёт Ethernet-адрес этой машины (она обязана быть в одной физической сети с нашим хостом, так работает схема маршрутизации протокола IP). Рассмотрим все эти действия по порядку.

Адрес назначения	Шлюз	Флаги	Интерфейс
127.0.0.1	127.0.0.1	H	lo0
default	195.104.60.1	G	le0
195.104.60.0	195.104.60.5		le0

Таблица 3.4.2.2: Таблица маршрутизации хоста host.ourcompany.com

Мы знаем IP-адрес получателя данных (это 120.10.215.3), поэтому давайте вместе с протоколом IP поищем нужный маршрут в его таблице маршрутизации, приведённой на рисунке 3.4.3.2.

Первый адрес назначения не подходит, последний - тоже. Чтобы выяснить это, мы сначала сравниваем адреса сетей нужного нам IP-адреса и адреса из таблицы. После двух неудач остаётся только default. Из второй колонки узнаём, что данные нужно отправить хосту 195.104.60.1 через интерфейс le0, причём этот хост будет промежуточным (флаг G), то есть он тоже переправит данные куда-то дальше.

Узнав IP-адрес следующего хоста, мы пока не можем отправить данные. Сетевой драйвер Ethernet не понимает IP-адреса, он работает только с адресами Ethernet. Поэтому теперь нужно выяснить Ethernet адрес хоста 195.104.60.1.

Для этого IP призывает на помощь протокол ARP. Тот сначала ищет физический адрес, соответствующий запрошенному IP-адресу, в своей таблице. Если его там не оказалось, то с помощью драйвера по всей сети Ethernet рассылается широковещательный запрос: "Владелец IP-адреса 195.104.60.1, назови свой Ethernet адрес." Будем оптимистами и будем считать, что интересующий нас хост работает и ответил на запрос, прислав свой Ethernet-адрес.

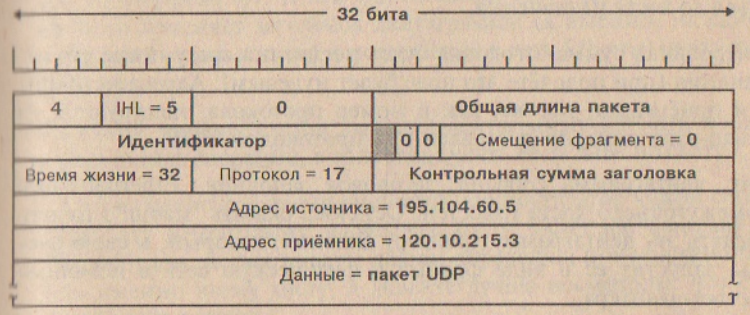


Рисунок 3.4.2.3: Дейтаграмма, передаваемая протоколом IP сетевому адаптеру

Теперь IP знает всё, что необходимо для посылки данных, а именно: какому хосту их посылать и Ethernet-адрес этого хоста. Дальше заполняется заголовок дейтаграммы, и к нему присоединяется полученный от UDP пакет. В конце концов дейтаграмма выглядит примерно так, как показано на рисунке 3.4.2.3.

Поясним, почему мы занесли в поля заголовка именно эти значения (чтобы понять дальнейшее, сравните этот заголовок с приведённым в параграфе 2.3.1).

Самая используемая сейчас в Internet версия протокола IP имеет номер 4, поэтому в поле Версия стоит четвёрка. Заголовок дейтаграммы не использует Опции, поэтому его длина равна 5, что и отображено в поле IHL. Кроме того, для простоты мы игнорируем поле Тип сервиса, поэтому в него записан 0. Общая длина пакета и идентификатор для каждой дейтаграммы могут быть свои, поэтому соответствующие поля остались нетронутыми.

В самом начале главы, при описании окружения, мы договорились, что все посылаемые данные будут такого размера, чтобы их можно было посылать по сети Internet одной дейтаграммой. Это условие определило значения флага MF и поля Смещение фрагмента - они установлены в 0. Но мы позволим промежуточным хостам разбивать дейтаграмму на фрагменты (это уже будет не нашей заботой) и поэтому установим флаг DF тоже в ноль.

Чтобы особо не мучить себя подсчётами и не беспокоиться за судьбу дейтаграммы, поставим ей время жизни, то есть значение одноимённого поля, равным 32 (это стандартное значение, используемое в Internet). Если дейтаграмма по дороге не заблудится среди маршрутизаторов и сетей, то ей должно этого хватить, чтобы добраться до хоста назначения.

Контрольная сумма заголовка будет посчитана сразу после его заполнения (при подсчёте это поле будет нулевым). Адреса источника и приёмника, так же, как и номер протокола, копируются из псевдо-заголовка, предоставленного протоколом UDP.

Итак, дейтаграмма собрана, заголовок заполнен, Ethernet-адрес промежуточного хоста известен. Осталось сказать "марш!", то есть передать эту дейтаграмму драйверу Ethernet, который, в свою очередь, запустит её в виде фрейма в физическую сеть с помощью Ethernet-адаптера.

3.4.3. Путь от отправителя к получателю

Недолго пробыла дейтаграмма в сети Ethernet. Увидев, что для него есть фрейм, сетевой адаптер хоста с IP-адресом 195.104.60.1 забрал её оттуда. Передав сетевому драйверу, он стал следить за сетью дальше, а дейтаграмма попала к протоколу IP этого же хоста.

Протокол IP сначала проверил контрольную сумму заголовка (предварительно прочитав это поле и записав в него 0), потом посмотрел на адрес назначения дейтаграммы и, сравнив со своими собственными, понял, что её нужно передать дальше. Опять начинается процесс поиска IP-, а потом Ethernet-адреса следующего хоста. Посмотрим на таблицу маршрутизации:

Адрес назначения	Шлюз	Флаги	Интерфейс
127.0.0.1	127.0.0.1	H	lo0
default	195.104.61.10	G	le1
195.104.61.0	195.104.61.3		le1
195.104.60.0	195.104.60.1		le0

Таблица 3.4.3.1: Таблица маршрутизации хоста router.ourcompany.com

Из неё опять видно, что в качестве маршрута для данных с адресом назначения 120.10.215.3 может быть выбран только default. Это значит, что следующим промежуточным хостом будет 195.104.61.10, внешний маршрутизатор, а интерфейсом, по которому отправятся данные, будет le1 (заметьте, что дейтаграмма прибыла на этот хост через интерфейс le0).

IP просит протокол ARP определить Ethernet-адрес хоста 195.104.61.10, но добавляет, что опрашивать нужно сеть, доступную через интерфейс le1. После определения этого адреса протокол IP немного изменяет заголовок дейтаграммы (а именно, на единицу уменьшает время жизни) и посылает её в сеть Ethernet через интерфейс le1.

Внешний маршрутизатор выловил дейтаграмму из сети, проделал аналогичные операции, и запустил гулять её дальше по свету...

...а через некоторое время внешний маршрутизатор чужой сети получил эту дейтаграмму и отослал хосту router.aliencompany.com. Мы не будем рассматривать, как он определил, что её нужно посылать именно этому хосту, а давайте лучше посмотрим, что с нею происходит дальше.

Хост router.aliencompany.com выбрал дейтаграмму из сети 120.10.212.0 через интерфейс le1. Теперь протоколу IP предстоит разобраться, куда её посылать дальше, поскольку предназначена она не ему. Таблица маршрутизации этого хоста приведена в таблице 3.4.3.2.

Адрес назначения	Шлюз	Флаги	Интерфейс
127.0.0.1	127.0.0.1	H	lo0
default	120.10.213.1	G	le1
120.10.212.0	120.10.213.15		le1
120.10.214.0	120.10.214.10		le0

Таблица 3.4.3.2: Таблица маршрутизации хоста router.aliencompany.com

Нужно найти по этой таблице маршрут к хосту 120.10.215.3. Первый адрес назначения не подходит, поэтому смотрим на третий. Сравнивая адреса сетей нужного нам адреса и адреса назначения из таблицы, убеждаемся, что они совпадают (то же самое верно и для четвёртого адреса). Дело в том, что IP-адреса с 0.0.0.0 по 127.255.255.255 являются адресами класса A, а в этом классе адрес сети занимает первый байт. Нужный нам адрес и адреса из третей и четвёртой строк - все они принадлежат сети 120.0.0.0.

Но теперь нужно вспомнить про маску подсети. В самом начале мы условились, что для сети 120.0.0.0 она равна 255.255.254.0. Накладываем маску на адрес 120.10.215.3 и на адреса из таблицы:

120.10.215.3 даёт подсеть 120.10.214.0, хост 1.3 (или 259, если записать одним числом)
120.10.212.0 и 120.10.214.0 дают подсеть с тем же адресом.

Уже видно, что подходит четвёртая строка таблицы, потому что она дала совпадение не только по адресу сети, но и по адресу подсети. Поэтому выбираем её, то есть сетевой интерфейс le0. В колонке Шлюз в четвёртой строке стоит наш собственный IP-адрес, а флаг G не установлен. Это означает, что дейтаграмму можно посылать прямо по тому адресу, который указан в её поле Адрес назначения.

Дальше - уже ставший привычным запрос протоколу ARP на поиск Ethernet-адреса хоста 120.10.215.3 в сети, доступной через интерфейс le0. После этого - очередное уменьшение Времени жизни в заголовке дейтаграммы и, наконец, её последнее путешествие по сети Ethernet. Провожатым становится сетевая карта, соответствующая интерфейсу le0.

3.4.4. Финиш: от сетевой карты до прикладной программы

А встречающим - карта, подключенная к одноимённому интерфейсу, но уже на машине 120.10.215.3. Теперь протокол IP, получив от

le0 дейтаграмму, уже сможет с облегчением вздохнуть - это для него.

После обычной проверки контрольной суммы заголовка IP должен передать данные выше. Для этого из поля-тёзки извлекается номер протокола, заголовок дейтаграммы отбрасывается, и данные в "чистом" виде передаются этому протоколу. Так как когда-то давно, на хосте-отправителе, мы в поле Номер протокола записали число 17, что соответствует протоколу UDP во всём Internet, то UDP и получает данные. Вместе с данными передаётся псевдо-заголовок, содержащий адреса отправителя, получателя, длину пакета, и контрольную сумму. Его заполняет IP по имеющейся в заголовке дейтаграммы информации. И пакет UDP (это именно он замаскировался под словом "данные"), и псевдо-заголовок должны совпадать с изображёнными на рис. 3.4.2.1.

Протокол UDP, в свою очередь, проверяет контрольную сумму всего пакета вместе с псевдо-заголовком. Если всё в порядке, то в поле Порт получателя заголовка пакета UDP читает номер порта. В нашем случае это 1000. Программа, зарегистрированная под этим номером у протокола UDP, в конце концов, получает долгожданные (или неожиданные) данные. Финиш.

Если эта программа решит ответить её далёкой собеседнице, то узнать номер порта (2000), IP-адрес (195.104.60.5) она может, исходя из полученной от UDP дополнительной информации. И тогда гонка начнётся заново, только уже в обратном направлении...

Сети, хосты, протоколы, программы - все могут быть обозначены различными числовыми значениями - адресами IP, номерами протоколов, номерами портов и сокетами. Но как не запутаться в этих числах? В следующей главе будет описана более простая система идентификации хостов - система, в которой вместо адресов IP используются имена.

4. Система именования

Стандарт протокола IP определяет имена, адреса и маршруты так:

Имя определяет то, что мы ищем. Адрес определяет где это находится. Маршрут определяет, как туда добраться.

Следуя такой терминологии, становится понятно, зачем в Internet появился ещё один тип объектов - имена. Человеку нужно имя компьютера для того, чтобы понять, что это за компьютер. Человеческая память хранит слова обычного языка гораздо лучше, чем наборы цифр. Поэтому использование имён компьютеров для человека гораздо проще и надёжнее использования адресов - они легче запоминаются и при их воспроизведении возникает гораздо меньше ошибок.

4.1. Имена и адреса

Любой сетевой интерфейс, подключенный к сети TCP/IP, имеет свой уникальный IP-адрес. Имя (будем дальше называть его именем хоста) может быть присвоено любому устройству, имеющему IP-адрес.

Однако имена хостов бесполезны, если, кроме самого хоста, больше никто не знает о его имени. Ведь практически всё программное обеспечение, работающее с Internet, оперирует адресами, не заботясь об именах. Хотелось бы, конечно, чтобы в большинстве случаев использование как имени, так и адреса было бы равносильно. Для этого все хосты в Internet должны уметь узнавать адреса друг друга по именам. Этот процесс - установления соответствия между именем хоста и его IP-адресом - называется переводом имён (name resolution). Причём, как мы уже сказали, любой хост в Internet должен быть способен перевести имя любого другого хоста. Кстати, человеку иногда может понадобиться узнать по адресу хоста его имя. Такой процесс называется обратным переводом.

Чтобы хосты были способны узнавать адреса друг друга по именам, необходима какая-то унифицированная система. Она должна быть

способна: распространять информацию об именах хостов по Internet; обеспечивать способы получения такой информации по запросу; уметь обновлять эту информацию. В Internet существует три подхода к построению таких систем. Это таблицы хостов, Сетевая информационная служба (NIS, Network Information Service фирмы Sun Microsystems) и Служба имён доменов (DNS, Domain Name System), стандарт Internet. Поговорим сначала о первой как об исторически более ранней.

4.2. Таблица хостов

В 70-х годах, в самом начале развития Internet, когда она состояла только из одной ARPANET и к ней было подключено несколько сотен хостов, вся информация о них могла храниться в одном файле. Такой файл назывался HOSTS.TXT и хранился в компьютере Стэнфордского исследовательского института (SRI, Stanford Research Institute). В нём, помимо имён хостов и их адресов, содержалась и другая информация - тип компьютера, операционной системы и т.д. Из HOSTS.TXT генерировался специальный файл операционной системы UNIX - /etc/hosts, который, собственно, и называется таблицей хостов.

Таблица хостов состоит из двух полей - адреса хоста и его имени. Просмотром этой таблицы система может определить адрес по имени, то есть перевести имя в адрес. Но чтобы каждый хост мог перевести имя другого хоста в адрес, в его таблице хостов должна содержаться соответствующая запись. Поэтому каждый хост должен иметь у себя таблицу хостов, содержащую информацию обо всех других хостах. Для небольшой и стабильной сети такой вариант подходит идеально. Но как только сеть начинает расти, эта система терпит крах. Во-первых, таблица хостов растёт пропорционально количеству хостов. Во-вторых - и это самое неприятное - каждое изменение в центральной таблице хостов (в нашем случае она хранилась на компьютере фирмы SRI) должно быть отражено во таблицах всех остальных хостов. А это увеличивает обмен (трафик) между хостами до недопустимых размеров. В итоге сеть становится просто неработоспособной.

Для решения этих проблем в 1984 году в Институте информатики университета Южной Каролины (USC Information Sciences Institute) под руководством Пола Мокапетриса (Paul Mockapetris) была разработана DNS. Эта система, в отличие от таблиц хостов,

масштабируема и позволяет децентрализовать управление именами хостов. О ней мы сейчас и поговорим.

4.3. Domain Name System

4.3.1. Свойства

Мы до сих пор сознательно ничего не говорили о структуре имён в Internet. Дело в том, что чёткие правила синтаксиса имён и их уникальности появились только в DNS. Вообще DNS можно охарактеризовать несколькими свойствами:

- DNS хорошо масштабируется. Она основана не на одной огромной таблице. По сути, это есть распределённая система баз данных, которая не приводит к коллапсу, когда сама сеть растёт. Сейчас DNS имеет информацию о десятках миллионах хостов.
- DNS гарантирует, что новая информация о хосте будет распространена по всему Internet. Но только тогда, когда она понадобится, и тем, кого она интересует.

Рассмотрим подробнее, что же такое DNS, какими понятиями она оперирует и как работает.

4.3.2. "Что в имени твоём?" Иерархия доменов

Видели ли вы когда-нибудь имя хоста в Internet? Если да, то оно наверняка выглядело как www.apple.com, wuarchive.wustl.edu или rector.msk.su? Что же означают эти странные названия и зачем между отдельными именами стоят точки? Этими и другими вопросами мы и займёмся дальше.

На протяжении всего параграфа будет прослеживаться очень хорошая аналогия - между именами файлов в MS-DOS или UNIX и именами хостов. Вспомним, что они выглядят примерно как

\windows\system\vgafixed.fon - в MS-DOS или как
/usr/bin/mail - в UNIX.

В этих примерах запись имён идёт от главного каталога, который обозначается куда-то наклонной чертой. Если мы не будем отступать от этого правила, и попытаемся пофантазировать и перебрать ВСЕ возможные имена файлов (включая пустое имя файла, которого, конечно, ни в UNIX, ни в MS-DOS быть не может), то получим множество всех имён файлов.

Представим теперь полные имена файлов в виде дерева. Поместим в его вершины имена каталогов, а в корень - главный каталог (самую первую наклонную черту). Разумеется, в самом низу, в листе, окажется имя файла (теперь уже короткое, без имён каталогов).

Оказывается, подобное можно проделать и с именами в Internet. На самый верх поместим точку " ". Кстати, она используется в одном единственном случае - когда нужно обозначить самый верхний уровень имён (подобно обозначениям "/" или "\" для имён главных каталогов в файловых системах). В вершины, лежащие непосредственно под точкой, поместим самые... правые части имён в Internet. Это значит, что самая правая часть имени - самая общая, а чем левее - тем специфичнее. Например, в имени www.apple.com самая правая часть - com - обозначает принадлежность к коммерческим организациям, следующая справа налево - apple - является названием этой коммерческой организации, а самая левая - www - обозначает сервис, предоставляемый фирмой apple. Как вы уже, наверное, догадались, разделителем частей имени в обычной, "плоской", записи служит всё та же точка. Так же, как и в именах файлов - наклонная черта.

Назовём теперь всё вышеперечисленное своими именами. Множество всех возможных имён в Internet (то есть дерево, состоящее из всех возможных вершин), называется пространством имён доменов (domain name space). Вершины этого дерева (части имени) называются узлами (nodes). Кстати, узлами также называются и листья этого дерева - дальше мы увидим, почему между ними не делается различия. Все узлы, лежащие ниже какого-то узла (вместе с этим узлом), образуют домен (domain). В терминах дерева домен - это поддерево. Всё дерево называют корневым доменом (root domain). На рисунке 4.3.2.1 приведён пример дерева имён доменов.

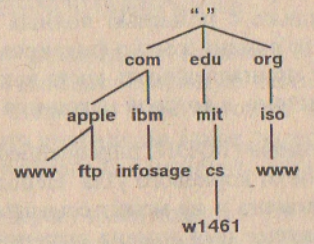


Рисунок 4.3.2.1: Пример дерева имён доменов

Нам осталось определить, пожалуй, самое главное понятие - имя домена (domain name). Как уже известно, домен - это поддерево, вершиной которого является какой-то узел, то есть узел однозначно определяет домен. Этот домен имеет своё имя. Оно состоит из имён узлов, расположенных на пути от корня (точки) до узла, порождающего домен, включая имя этого узла. Причём все эти имена отделены друг от друга точкой и расположены справа налево. То есть, идя от корня к узлу, мы добавляем имя очередного узла слева.

Приведём примеры. Пусть нам нужно построить имя домена, соответствующего узлу w1461 на рисунке 4.3.2.1. Идя сверху вниз, сначала получим имя edu (заметьте, что без точки - она используется только для обозначения корневого домена, то есть почти никогда). Затем, спускаясь ниже - mit.edu. Ещё ниже - cs.mit.edu. И, наконец, дойдя до узла w1461, получим имя домена w1461.cs.mit.edu. Аналогично, для узла infosage полным именем домена будет infosage.ibm.com.

Кстати, только что мы не случайно упомянули о полном имени домена. В файловых системах аналогом полного имени является абсолютный путь - путь от главного (корневого) каталога. Например, /usr/bin/mailx - абсолютный путь. Но файловые системы позволяют пользоваться относительным заданием пути к файлу. Если системе известно ваше текущее местонахождение (например, вы в каталоге /usr), то она сама сможет определить, что означает относительный путь (bin/mailx будет преобразован в /usr/bin/mailx). Почти то же самое действительно и для DNS. Зная, в каком домене вы находитесь, можно писать не полное имя домена, а только его часть. Например, если вы - в mit.edu, то задать имя домена, соответствующее узлу w1461, можно как w1461.cs. Всё, что должна уметь ваша система для получения полного имени домена - это определить имя текущего домена.

Правда, DNS имеет и существенные отличия от именования в файловых системах. Во-первых, на узлы за пределами текущего домена можно сослаться только с помощью полных имён доменов. Во-вторых, имя текущего домена жёстко фиксировано для каждого хоста. Оно определяет принадлежность хоста какому-то домену, а не точку в файловой системе, в которой находится пользователь.

Итак, полное имя домена (FQDN, fully qualified domain name) - это имя домена, заданное от корневого узла. Неполное имя домена задаётся от текущего домена и не может ссылаться на узлы в других доменах. То есть текущее имя домена ограничивает пространство имён тем, что лежит ниже узла, ему соответствующего.

Так же, как в файловых системах, имена непосредственных потомков какого-то узла (иначе, соседей) не могут совпадать. Как в файловой системе в одном каталоге не может быть двух файлов с одинаковыми именами, так и в DNS два разных домена не могут иметь одно и то же полное имя. Но это не означает, что два узла в разных доменах не могут иметь одно и то же имя. Например, узлы www в доменах apple.com и iso.org - нормальная ситуация. Но не может быть двух разных узлов с одним полным именем домена, например двух узлов infosage.ibm.com.

Теперь самая пора вспомнить о том, для чего DNS была построена. То есть об адресах IP и об их связи с именами.

Все хосты в Internet могут иметь свои имена. Некоторые хосты даже имеют по несколько имён. Но ведь хост может также иметь несколько IP-адресов? Это действительно так. Реально имя хоста привязано не к адресу, а к межсетевому уровню (экземпляру протокола IP). Поэтому и имя хоста принадлежит не какому-то конкретному адресу, а экземпляру протокола IP, который может иметь несколько адресов.

Узлы в дереве имён DNS соответствуют хостам. Причём все узлы, а не только самые нижние (листья, то есть вершины, у которых нет потомков). На первый взгляд это кажется несколько странным. Но такое соглашение оправдано - каждый узел, имеющий потомков, может быть ответственным за работу "своего" домена и почему не поставить ему в соответствие реальный хост? Это упрощает структуру и понимание DNS - за счёт слияния таких понятий как вершина и лист дерева.

Соответствие узлов в дереве имён хостам автоматически решает проблему имён хостов. На самом деле - в таком случае именем хоста просто является имя домена соответствующего узла. Примеры построения имён доменов мы уже приводили выше. В этих примерах мы фактически строили имена хостов.

В самом начале этой главы уже говорилось о том, что DNS - это распределённая база данных. Основную часть информации в ней составляют записи, ставящие в соответствие именам доменов реальные хосты или, что почти то же самое, адреса IP. Каждому имени поставлен в соответствие как минимум один адрес.

Давайте вернёмся к дереву имён доменов. На нём можно выделить узлы разных уровней. Самому верхнему уровню принадлежат все потомки корневого узла. Второму уровню - потомки узлов первого уровня, и так далее. Домены, соответствующие узлам верхнего

уровня, называются доменами верхнего уровня (top level domain). Теоретически эти домены могут называться как угодно, но на практике это не так. Дело в том, что домены верхнего уровня играют слишком большую роль в DNS и, как мы в дальнейшем увидим, отвечают практически за все существующие имена доменов. В Internet исторически сложилась так называемая организационная система имён доменов верхнего уровня. Она включает в себя восемь имён, каждому из которых соответствует тот или иной вид организации:

com Коммерческие организации, типа IBM (ibm.com), Microsoft (microsoft.com), Levi Strauss (levi.com), Citibank (citibank.com)

edu Образовательные учреждения, например, Стэнфордский университет (stanford.edu) или Университет Нью-Йорка (nyu.edu)

gov Правительственные организации, такие как NASA (nasa.gov), библиотека Конгресса США (Library of Congress - loc.gov) или Госдепартамент (state.gov)

mil Военные организации - армия США (army.mil), военно-морской флот США (navy.mil)

net Сетевые организации, например, сеть национального научного фонда США (nsf.net)

org Другие некоммерческие организации, типа Международной организации стандартизации (iso.org)

int Международные организации, например ООН (un.int) и НАТО (nato.int)

arpa Исторический домен верхнего уровня - то, что осталось от прародителя Internet, сети ARPANET

В этом списке явно преобладают организации из США. Это объяснимо - первоначально Internet развивалась там, и подавляющее большинство организаций, подключенных к этой сети, имели американскую прописку.

Когда Internet начала становиться международной сетью, появилась необходимость расширить список доменов верхнего уровня и включить в него имена, построенные по географическому принципу. Поэтому было решено использовать стандарт ISO 3166, определяющий двухбуквенные аббревиатуры для каждой страны мира. Эти сокращения дополнили список доменов верхнего уровня.

Дальше приведён короткий список стран и соответствующих имён доменов:

at Austria, Австрия
ca Canada, Канада
ch Switzerland, Швейцария
de Germany, Германия
il Israel, Израиль
jp Japan, Япония
ru (su) Russia (Soviet union), Россия (старый домен su - для СССР)
se Sweden, Швеция
uk (gb) United Kingdom (Great Britain), Великобритания (gb не используется)
us United States of America, США

Несмотря на то, что для США имеется отдельный домен верхнего уровня, он практически не используется - американцы без зазрения совести эксплуатируют перечисленные выше организационные домены (com, org, int и т.д.). Домен каждой страны имеет свою внутреннюю структуру - жестко фиксированные домены второго уровня. В Великобритании эта структура построена по организационному принципу, но имена этих доменов отличаются от аналогичных имён доменов в США. Так, коммерческие организации входят в домен co.uk, а образовательные учреждения - в ac.uk (от academic community).

В России ситуация с доменами второго уровня пока не стабилизировалась. Подавляющее их большинство - региональные (Москва - msk.ru, Тамбов - tambov.ru, Санкт-Петербург - spb.ru), но есть и перлы типа Московского государственного университета (msu.su), отдельных организаций (Демос - demos.su, КИАЕ им. Курчатова - kiae.su) и служб (система Россия Он-лайн фирмы Совам Телепорт - online.ru). Кроме того, появился первый организационный домен - для образовательных учреждений - edu.ru. Картина будет неполной, если не упомянуть об остатках доменов второго уровня в домене su. Всё это связано с отсутствием согласованной политики выделения новых имён доменов. Остаётся только надеяться на разумность и последовательность владельцев сети Relcom - основного в России поставщика связи с Internet (ISP, Internet service provider).

4.3.3. Структура DNS

Давайте вспомним, как была устроена ранняя система именования - с таблицами хостов. На каждой машине хранился огромный файл, в котором были записаны все имена и соответствующие им адреса. Все машины знали все имена. За таблицу отвечал один компьютер, откуда её обновлённую версию забирали все остальные. Как мы уже говорили, такая централизованная система перестала быть работоспособной при разрастании сети.

Одной из основных целей создания DNS была децентрализация управления. Она была достигнута за счёт применения *делегирования* (delegation), то есть передачи полномочий. Вся задача разбивается на составные части и ответственность за эти части возлагается на кого-то другого. Разбивать задачу на части можно по-разному, но мы разобьем её подоменно. Это означает, что мы делегируем ответственность за поддомены, а сами только запомним, кто теперь за них отвечает. Мы сами - это корневой домен, и теперь обязаны только помнить, кому передали ответственность за домены верхнего уровня.

Каждый домен может поступить точно так же - разбить свою зону ответственности на поддомены и делегировать полномочия кому-то ещё. И так далее, пока есть что делить на части.

Мы нигде не говорили о том, что ответственные за домены **должны** делить свои домены на части, **должны** делегировать ответственность за каждый из поддоменов - вовсе нет. Они **могут** делать это выборочно. Та часть пространства имён, за которую они несут ответственность, называется *зоной* (zone). Давайте посмотрим это на примере.

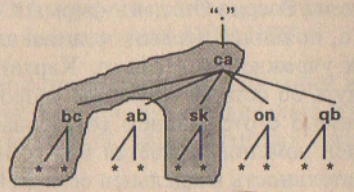


Рисунок 4.3.3.1: Пример зон ответственности (домены, составляющие одну зону, находятся в затенённой области)

На рисунке 4.3.3.1 домен верхнего уровня ca (соответствующий Канаде) имеет пять поддоменов - bc.ca (Британская Колумбия), ab.ca (Альберта), sk.ca (Саскачевань), on.ca (Онтарио), qb.ca

(Квебек), соответствующих пяти провинциям. Администрации трёх из них - Альберты, Онтарио и Квебека - готовы взять на себя управление и ответственность за все имена в соответствующих поддоменах. Но за оставшимися двумя поддоменами (Британской Колумбии и Саскичевани) будет следить администрация домена верхнего уровня ca. Поэтому зону ca будет составлять вся информация из *узла* ca и *поддоменов* bc.ca и sk.ca.

Программы, хранящие информацию об именах доменов, называются *серверами имён* (name servers). В общем случае они имеют полную информацию о некоторой части всего пространства имён доменов. Как уже говорилось, часть пространства имён доменов - это зона, поэтому именно серверы имён несут за неё ответственность. Каждый сервер может отвечать сразу за несколько зон.

DNS определяет несколько типов серверов имён. Первый из них - *ведущий* (primary master). Он получает всю информацию о зонах, за которые отвечает, из файлов того хоста, на котором работает. Другими словами - он является главным распорядителем информации о зонах и единственным, кто отвечает за её сохранность. На зону может приходиться только один ведущий сервер.

Второй тип - *ведомые* (secondary masters). Эти серверы имён являются помощниками ведущих. Они имеют доступ к информации о зонах ответственности, но только по чтению. В отличие от ведущего, они не имеют права изменять эту информацию.

Фактически в DNS было произведено разделение ответственности на *ответственность за достоверность информации* и *ответственность за её сохранность*. Первую несут и ведущие, и ведомые, вторую - только ведущие. Причём ведомых на одного ведущего может быть несколько - но не меньше одного.

Как же поддерживается достоверность информации на ведомых серверах? При запуске, а потом - через определённые промежутки времени эти серверы перекачивают всю информацию о зоне ответственности с ведущего. Этот процесс называется *пересылкой зоны* (zone transfer).

Существует ещё один тип серверов имён. Эти серверы не несут ответственности за достоверность информации, не отвечают за её сохранность. Единственное, что они умеют делать - это отвечать на запросы об именах доменов (о запросах - в следующем параграфе) и хранить всю узнанную информацию. Такие серверы называются *кэширующими* (caching-only). Информация, полученная при выполнении запроса, может быть использована повторно, но через

некоторое время перестаёт быть правильной. Тогда она уничтожается. Такая техника используется во всех типах серверов имён, но в кэширующих серверах это единственный способ работы.

Теперь настала пора поговорить о том, как работает DNS, откуда берёт информацию, как им пользуются другие программы. Об этом пойдёт речь в следующем параграфе.

4.3.4. Как DNS работает

Файлы, из которых ведущие серверы загружают информацию о своих зонах, называются *файлами данных* (data files). Ведомые в общем случае получают информацию от ведущего сервера, но иногда тоже загружают её из своих файлов данных. Это происходит в том случае, если ведомый настроен на резервное сохранение информации о зоне в файлах данных. Если ведомый сервер был перезапущен, то он проверяет, является ли информация в его файлах данных корректной. Если да, то происходит её загрузка из файлов данных, а в противном случае инициируется пересылка зоны с ведущего сервера.

Файлы данных содержат так называемые *записи о ресурсах* (resource records). Они описывают все хосты в этой зоне, а также делегирование поддоменов. Все записи формируют описание зоны.

Процесс перевода имён и переводчики

Информация о зонах сама по себе не имеет смысла. Мы должны уметь получать её от серверов имён. Этим занимаются так называемые *переводчики* (resolvers). В их функции входит:

- запрос информации у сервера имени
- обработка ответа на запрос (ответом может быть запись о ресурсе или ошибка)
- возврат результата программе, запросившей информацию

Переводчики обычно только и делают, что компонуют запрос, посылают его, ждут ответа от сервера, посылают запрос ещё раз, если ответа нет, возвращают информацию программе. Такой тип переводчиков называется в DNS *ограниченными переводчиками* (stub resolvers). Есть и более умные экземпляры, которые ещё и кэшируют полученные ранее ответы, но они встречаются реже.

Всю остальную работу по поиску нужной информации делают серверы имён. Они не только дают ответы по своим зонам ответственности, но также по всему пространству имён, если это необходимо. Как уже отмечалось в начале этой главы, такой процесс называется переводом имён или просто переводом.

Корневые серверы имён

Пространство имён доменов представлено в виде дерева, поэтому всё, что нужно знать серверу имён для поиска пути до любого узла в этом дереве - это имена и адреса корневых серверов имён. Наш сервер может спросить у корневого про любое имя домена в дереве, и тот укажет, что делать дальше.

Корневые серверы имён (root name servers) отвечают, конечно же, за корневой узел. Они знают, кто ответственен за домены верхнего уровня. Получив запрос на перевод какого-нибудь имени в адрес, они, по крайней мере, могут сказать адреса и имена серверов имён, ответственных за тот домен верхнего уровня, в пределах которого находится интересующее нас имя. Далее, серверы имён доменов верхнего уровня снабдят нас списком имён и адресов серверов, отвечающих за домен второго уровня нашего имени. Таким образом, каждый сервер имён будет давать нам всё более точную информацию о том, где спросить об интересующем нас имени. В конце концов мы доберёмся до сервера, знающего именно то, что нас интересует.

Но в самом начале работы мы всё-таки обращаемся к корневому серверу имён. Чтобы разгрузить его, в DNS есть специальные механизмы - типа кэширования информации (что это такое, мы уже говорили, более подробно - дальше в этом параграфе). Но при отсутствии информации в кэше всё равно приходится обращаться к корневому серверу. Из-за этого он является краеугольным камнем DNS - если не работает корневой сервер имён, то не работает и перевод имён в Internet. Чтобы избежать такой ситуации, существует не один, а целых семь корневых серверов (по состоянию на март 1993 года). Конечно, кто-то из них является ведущим, а остальные - ведомыми, но нагрузку удаётся распределить примерно поровну. Некоторые из корневых серверов находятся в MILNET, один в сети NASA, один в Европе, а остальные - вокруг хребта сети Национального научного фонда (NSFNET).

Несмотря на то, что корневых серверов целых семь, на них ложится огромная нагрузка. По данным на март 1993 года, в среднем каждый из них обрабатывал 20,000 запросов в час (около 6 запросов в

секунду). Но, тем не менее, перевод имён в Internet работает хорошо. На следующем рисунке показан пример этого процесса для реального имени хоста:

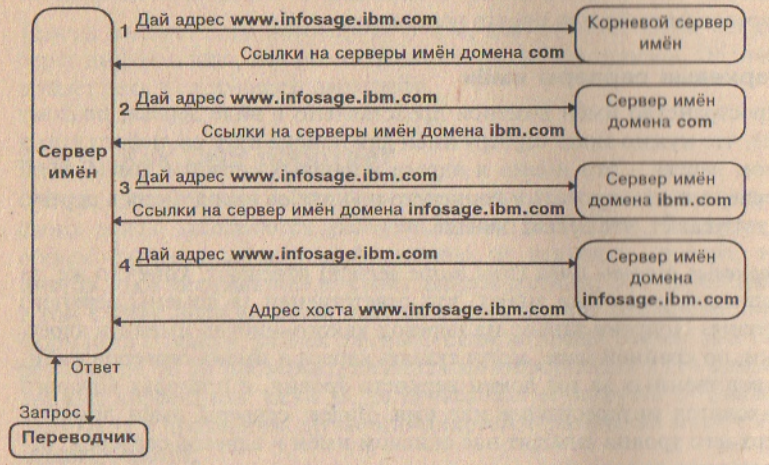


Рисунок 4.3.4.1: Процесс перевода имени www.infosage.ibm.com

Локальный сервер имён по просьбе переводчика запрашивает у корневого сервера адрес хоста www.infosage.ibm.com. Корневой сервер не знает адреса этого хоста, но знает адреса серверов имён домена верхнего уровня com. Он возвращает список этих серверов. Дальше локальный сервер обращается с таким же запросом к серверу имён домена com. Тот отвечает списком серверов домена ibm.com. Один из этих серверов, в свою очередь, выдаёт список серверов имён домена infosage.ibm.com, поскольку они отвечают за поддомены в его зоне ответственности. Запросив у сервера имён домена infosage.ibm.com, который уже известен, всё тот же адрес, получаем ответ - (реальный адрес).

Рекурсивные и итерационные запросы

Из этого примера видно, что основная часть работы по переводу имени ложится на плечи нашего, локального сервера имён. Все остальные по запросу просто отсылают наиболее подходящую информацию, которая у них есть. В большинстве случаев это ссылки на другие серверы имён.

Таким образом, локальный сервер вынужден сам пробегаться по ссылкам и искать окончательный ответ - IP-адрес хоста. В этом состоит суть так называемого *рекурсивного запроса* (recursive query).

Сервер обязан вернуть либо найденный адрес, либо сообщение об ошибке. Он не может вернуть спросившему его переводчику ссылку на другой сервер имён - переводчик ничего не сможет сделать с этой информацией. Поэтому тот посылает рекурсивный запрос.

Есть ещё один тип запросов - *итерационный* (iterative query). В этом случае сервер, получивший такой запрос, ищет информацию только в своих файлах данных или в своём кэше. Не найдя точного ответа, он может отослать ссылки на серверы имён самых подходящих доменов. Таким доменом может оказаться и корневой.

Теоретически, рекурсивный поиск информации (соответствующий рекурсивному запросу) эффективнее. Сервер имён, запросивший адрес, ждёт, пока запрос спустится по дереву имён доменов до самого знающего сервера, который выдаст адрес. Потом полученный адрес пройдёт обратный путь и попадёт к серверу имён. Равномерно распределяется загрузка сервером имён - каждый из них (кроме самого последнего, "знающего") формирует рекурсивный запрос и ждёт; равномерно распределяется нагрузка на сеть - вместо того, чтобы запросы и ответы на них проходили каждый раз через локальные сетевые каналы, они блуждают по Internet. Но за всё надо платить. Представьте, что корневой сервер имён решил отвечать на все рекурсивные запросы. Тогда при получении каждого он сначала пытается найти ответ в собственной информации, при неудаче формирует новый запрос, ждёт ответа на него, выдаёт новые запросы, если на старый до сих пор не получен ответ, обрабатывает ответ и отсылает адрес или сообщение об ошибке. И так - 6 раз в секунду. Это резко повышает нагрузку на программу-сервер имён, на линии связи, повышает требования к аппаратуре и, в конечном счёте, обходится много дороже. Есть ещё один отрицательный момент - неизвестно, сколько придётся ждать локальному серверу имён ответа на свой рекурсивный запрос. А ведь его нужно повторить, если ответа нет, то есть запросить корневой сервер ещё раз.

Поскольку плата за альтруизм, то есть за рекурсивные запросы, оказывается слишком высокой, предпочитают пользоваться итерационными, то есть возлагать всю работу на локальный сервер имён. В одном-единственном случае отсылается рекурсивный запрос - от переводчика серверу имён.

На следующем рисунке схематично показан процесс поиска информации в DNS с использованием обоих типов запросов.

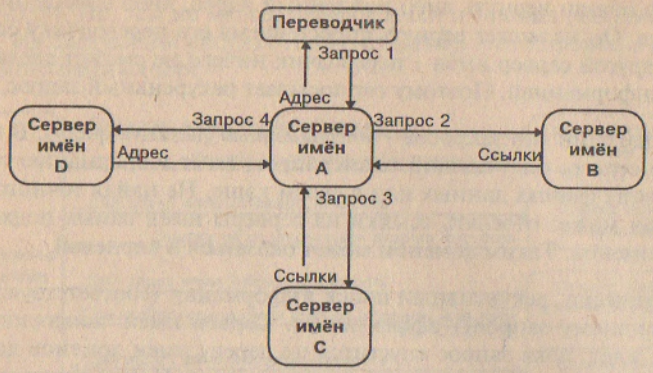


Рисунок 4.3.4.2: Схема поиска информации в DNS

Поясним, что происходит на этом рисунке. Сначала переводчик посылает рекурсивный запрос 1 серверу имён A. Этот сервер формирует и посылает итерационный запрос 2 серверу имён B. Получив от B ссылки на самые подходящие серверы имён, A формирует и посылает итеративный запрос 3 одному из этих серверов - C. Дальше C тоже присылает список ссылок на серверы имён. Среди них есть и D, который получает итерационный запрос 4 от сервера A. Сервер D знает точный ответ на запрос, поэтому посылает адрес. Сервер A получает этот адрес и отсылает его переводчику. Каждый из серверов в этом примере выполнил свой запрос - сервер A - рекурсивный, остальные - итерационные.

Кэширование информации

В процессе поиска информации в DNS сервер имён, выполняющий рекурсивный запрос, получает ещё и множество дополнительной информации - ссылки на ведущие и ведомые серверы имён каких-нибудь доменов с описанием их зон ответственности. Такая информация могла бы быть потеряна, но она сохраняется для повторного использования (кэшируется). Кстати, полученный окончательный ответ - IP-адрес, соответствующий какому-нибудь домену, также может быть сохранён.

Использование уже полученной информации - об адресах серверов имён, их зонах и IP-адресах - существенно ускоряет выполнение запроса и работу DNS в целом. Когда сервер получает новый запрос, он сначала просматривает информацию у себя в файлах данных (если это ведущий или ведомый сервер имён), потом - в кэше. Там он может обнаружить адреса серверов имён, чьи домены содержат поддомен запрошенного имени. Тогда он обращается сразу

к этим серверам, не беспокоя корневой, (при рекурсивном запросе), либо возвращает в ответ адреса этих серверов (при итерационном).

Конечно, информация в кэше не может и не должна храниться вечно. Если бы так было, то её обновление на ведущем и ведомых серверах никогда бы не распространилось по сети - оно бы просто никому не понадобилось, вся информация бралась бы из кэшей. По этой причине любая запись о ресурсах в файлах данных DNS имеет своё время жизни (TTL, Time to Live). Ровно столько времени эта запись может храниться в кэше. Потом она должна быть уничтожена. Значение этого параметра является результатом компромисса между эффективностью и достоверностью информации. Чем больше TTL, тем меньше запросов на получение этой записи. Но, в то же время, тем больше времени понадобится, чтобы изменение записи распространилось по сети.

Перевод адресов в имена

Вся эта глава была посвящена одной проблеме - переводу имени домена в адрес IP. Но иногда необходимо осуществить обратное - перевести IP адрес в имя домена. На первый взгляд, решив первую проблему, мы автоматически избавляемся и от второй - имея записи о ресурсах, где указаны имена доменов и их адреса, можно найти имя по адресу. Не всё так просто. Представьте себе, что вы попросили программу-переводчик найти имя домена с адресом 15.16.192.152. Что дальше? Обычная схема работы DNS здесь уже не срабатывает. Ведущие и ведомые серверы отвечают за имена доменов, а не за IP адреса хостов.

Но давайте посмотрим на адрес IP не как на адрес, а как на имя. По внешнему виду оно очень напоминает обычное имя домена - несколько подимён разделены точками. Правда, важность подимён слева направо убывает (самые левые числа являются адресом сети, правые - адресом хоста). В DNS всё наоборот - имя домена верхнего уровня стоит правее остальных.

А теперь представьте, что мы переставили части адреса IP в обратном порядке, добавили справа имя какого-нибудь домена и сказали, что всё полученное - также имя домена. Теперь в соответствие ему можно поставить какую-либо информацию, например - имя домена, соответствующее такому зашифрованному адресу. Например, соответствие "домен (хост) winnie.corp.hp.com имеет IP адрес 15.16.192.152" превратилась бы тогда в "домен 152.192.16.15.in-addr.arpa соответствует домену (хосту) winnie.corp.hp.com".

Теперь всё стало на свои места. Поскольку у нас есть полное имя домена, пусть даже построенное из адреса, то оно входит в пространство имён доменов. У этого имени есть несколько уровней. Самый верхний (из интересующих нас) - это уровень имён доменов, соответствующих самой **левой** части адреса IP. Всего таких доменов может быть 256. Дальше, у каждого из этих доменов может быть 256 поддоменов, имена которых соответствуют второму слева числу в адресе IP. И так далее - до самой правой части адреса. Домены, соответствующие самой правой части адреса, своих поддоменов иметь не могут, но каждому из них может быть поставлена в соответствие та самая запись "домену x1.x2.x3.x4.in-addr.arpa соответствует домен abc.def.ghi". За эти записи могут отвечать обычные серверы имён. Поясним это на примере, изображённом на рисунке 4.3.4.3.

На нём нарисована часть поддерева, соответствующая в пространстве имён доменов адресам IP, записанным в обратном порядке. Нетрудно увидеть, что зашифрованный адрес - 15.16.192.152. Он соответствует хосту winnie.corp.hp.com.

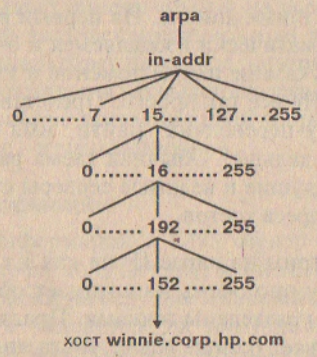


Рисунок 4.3.4.3: Пример имени домена при обратном переводе

В сети Internet существует единая политика выделения новых адресов организациям. Этой работой занимаются поставщики услуг в Internet. Каждой организации, которая получает доступ к этой сети, выделяется не один адрес, не разрозненная группа адресов, а все адреса какой-либо сети или подсети. Сама организация вольна распоряжаться полученными адресами по своему усмотрению. Но факт остаётся фактом - ответственность за адреса хостов, принадлежащих сети или подсети, несёт единое административное образование. Поэтому, в принципе, существует возможность собрать

его адреса хостов, построить из них имена доменов и поставить им в соответствие реальные имена этих хостов. Все полученные записи сложить на какой-нибудь сервер имён и назначить его ответственным (ведущим) за домен, соответствующий **адресу** сети или подсети.

В нашем примере мы можем это сделать для домена 15.in-addr.arpa. Собрав всю информацию о реально используемых адресах, имеющих адрес сети 15 (то есть вида 15.x1.x2.x3) и именах, соответствующих этим адресам, мы сложим всё это на какой-нибудь сервер имён и назначим его ответственным за домен 15.in-addr.arpa. Теперь, как только кто-нибудь захочет узнать имя домена по адресу вида 15.x1.x2.x3, он вынужден будет обращаться к этому серверу.

Мы сознательно уходили от объяснения, почему именно x1.x2.x3.x4.in-addr.arpa. Всё довольно просто. Об истории домена arpa уже немного говорилось. Внутри него есть поддомен in-addr, который просто расшифровывается как *инвертированный адрес* (inverted address). Дальше (левее) - всё уже знакомо - обратная (инвертированная) запись адреса IP. Если записать адрес в обычном виде, то вряд ли нам удалось бы назначить ответственного за домен 15.in-addr.arpa, поскольку он соответствовал бы всем хостам, чей адрес заканчивается числом 15. Проще узнать всех своих соседей, чем тёзок.

5. Протоколы маршрутизации

Давайте вспомним основные принципы маршрутизации в TCP/IP.

- маршрутизацией занимается протокол IP
- выбор маршрута - это поиск в таблице маршрутизации адреса соседнего (с которым имеется прямое соединение) шлюза, отвечающего за определённую сеть или хост (кроме default, который отвечает за "все остальные" сети и хосты)
- таблица маршрутизации может составляться и изменяться как вручную, так и автоматически. Для последнего созданы специальные протоколы маршрутизации, о которых пойдёт речь в этой главе.

5.1. Функции протоколов маршрутизации

Все протоколы маршрутизации имеют одни и те же функции. Они определяют "лучший" маршрут до места назначения и распространяют информацию о маршрутизации в сети. То, как они это делают, особенно как определяют лучший маршрут, отличает протоколы маршрутизации друг от друга.

5.2. Внутренняя и внешняя маршрутизация

Давайте немного заглянем в историю Internet. Когда эта сеть создавалась, ARPANET была "хребтом" (backbone) - по ней перемещался основной объём информации (подобно тому как человеческий позвоночник выдерживает основные нагрузки). Эта система была названа ядром (core), а шлюзы, составляющие её - центральными шлюзами (core gateways). Все остальные системы и сети подключались к ядру и передавали информацию через центральные шлюзы.

В такой иерархической системе вся информация о маршрутизации во всех составляющих Internet сетях собиралась центральными шлюзами. Они эту информацию обрабатывали, а потом обменивались друг с другом при помощи протокола От шлюза к шлюзу (GGP, Gateway to Gateway Protocol). Обработанная информация также передавалась внешним шлюзам сетей, подключенных к ядру.

Вне ядра Internet существовали группы независимых сетей, называемых автономными системами (AS, autonomous systems). Этот термин имеет формальное определение в маршрутизации в TCP/IP. Он означает не просто независимую сеть, а группу сетей и шлюзов со своим собственным механизмом сбора информации о маршрутизации и обмена ею с другими сетями. Информация о маршрутизации, передаваемая другой сети, называется информацией о доступности (reachability information). Это просто-напросто сведения о том, к каким сетям можно получить доступ с помощью данной автономной системы.

Обмен информацией о маршрутизации между автономными системами (то есть информацией о доступности) получил название внешней маршрутизации. Внутренняя маршрутизация осуществляется *внутри* автономных систем. Существуют специальные протоколы внутренней маршрутизации. О них и пойдёт речь в следующем разделе.

5.3. Подробнее о внутренней маршрутизации

Как уже говорилось выше, протоколы внутренней маршрутизации (впрочем, как и любые другие протоколы маршрутизации) используются для сбора и обмена информацией о маршрутизации, а также для выбора лучшего маршрута. Их особенность в том, что делают они это внутри автономных систем. Существует несколько разных протоколов внутренней маршрутизации, их отличие - в способе определения, какой маршрут лучший.

5.3.1. Метрики и разнообразие протоколов

Разные протоколы по-разному определяют лучший маршрут. Для этого они пользуются разными метриками - единицами измерения качества маршрута. Чем меньше метрика маршрута, тем он лучше.

Самый распространённый протокол маршрутизации - Протокол информации о маршрутизации (RIP, Routing Information Protocol). В качестве метрики он использует так называемый *счётчик ретрансляций* (hop count). Это число определяет "длину" маршрута и отражает количество шлюзов, через которые он пролегает, а также скорость и стоимость связи между ними (правда, эти параметры задаются только статически). Чем "короче" маршрут, то есть чем меньшее значение имеет счётчик ретрансляций, тем он лучше. Такой подход к определению лучшего маршрута называют дистанционно-векторным алгоритмом (distance-vector algorithm). Более подробно о том, откуда берутся значения счётчика ретрансляций и как RIP собирает и распространяет информацию, будет рассказано в следующем параграфе.

Самый длинный маршрут, с которым может иметь дело RIP, имеет длину 15 ретрансляций. Более длинные маршруты он рассматривает как недостижимые (разумеется, с его помощью) и отбрасывает. При этом не происходит ничего страшного, ведь маршрут - это адрес соседнего шлюза, с помощью которого можно переслать данные в указанную сеть. Если маршрут для какой-нибудь сети отсутствует в таблице маршрутизации, то данные будут посланы по маршруту по умолчанию, *default* (этот механизм был описан раньше, в главе 3.2).

Из-за такого ограничения RIP не может работать в больших автономных системах, где "длина" маршрута может превысить 15 ретрансляций. Кроме того, RIP пользуется статической метрикой, поэтому не может реагировать на изменения таких параметров, как скорость связи и загрузка сети. Поэтому были разработаны более "чуткие" протоколы, одним из которых является Hello.

Этот протокол использует в качестве метрики *задержку* (delay) - время, за которое дейтаграммы доходят до места назначения. При посылке пакета Hello записывает так называемый *временной штамп* (time stamp). Получающая система вычитает его из времени получения и определяет задержку. Лучший маршрут должен иметь наименьшую задержку.

Hello не получил широкого распространения, видимо, из-за необходимости постоянного измерения задержки, которое приводило к перегрузкам сети и задержке отправления дейтаграмм. В настоящее время быстро развивается новый протокол маршрутизации, "Выдай кратчайший маршрут первым" (OSPF, Open Shortest Path First). Его особенностями являются отслеживание *состояния связи* (link-state) с соседними маршрутизаторами, способность работать в очень

больших автономных системах, а также поддержка *маршрутизации с использованием равноценных маршрутов* (equal cost multipath routing). Эта тирада означает, что OSPF может одновременно работать с несколькими маршрутами для одного и того же места назначения. При этом с течением времени одни маршруты могут "ухудшаться" (становиться "дороже"), другие, наоборот, "хорошеть", а OSPF будет следить за ними и выбирать кратчайший (самый "дешёвый", то есть лучший). К сожалению, широкому распространению этого протокола пока препятствует реализованная и растиражированная на множестве систем логика протокола IP, в которой для каждого пункта назначения выбирается тот маршрут, который в таблице маршрутизации окажется первым. Самым массовым протоколом маршрутизации остаётся пока RIP, поэтому в следующем параграфе остановимся на нём более подробно.

5.3.2. RIP и как он работает

Давайте попристальнее взглянем на метрику протокола RIP - *счётчик трансляций*. Во-первых, она используется для выбора более выгодного маршрута. Во-вторых, не имеет единицы измерения - только в простых системах, где все сети похожи, счётчик трансляций может отражать количество шлюзов, через которые проходят пакеты. В-третьих, она может иметь только неотрицательное значение. В дистанционно-векторном алгоритме принято измерять нулевым расстоянием (то есть нулевым значением счётчика трансляций) маршрут, пролегающий внутри машины. То есть если данные не выходят за пределы системы, а передаются между локальными программами - стоимость их передачи равна нулю.

Базовым расстоянием, которое можно и нужно "измерять" (оценивать) в сетях TCP/IP, является "стоимость" соединения соседних шлюзов. Это расстояние по-другому ещё называют ценой передачи через сеть (ту, которая соединяет эти две машины). Измерив цену передачи через все сети, составляющие маршрут, а затем сложив их, можно узнать цену маршрута, то есть его метрику. "Но," - спросите вы, - "ведь таблица маршрутизации содержит указания только о том, какому из соседних шлюзов пересылать дейтаграммы. Как же посчитать цену всего маршрута?"

Ответ очень простой: нужно узнать цену, которую "платят" соседние шлюзы за пересылку дейтаграмм в интересующий пункт назначения (если они вообще имеют с ним дело), а потом сложить её с ценой передачи до каждого из этих шлюзов. Тот шлюз, цена маршрута через который окажется наименьшей, определит наилучший

маршрут (который состоит из адреса назначения, адреса шлюза, имени интерфейса, цены и другой менее интересной для нас информации).

Теперь о том, как все вышеописанные механизмы работают в RIP. Во-первых, сразу после запуска он опрашивает своих соседей о том, какие маршруты они могут предложить, то есть какие записи содержатся в их таблицах маршрутизаций. Получив ответ, он производит сложение полученных метрик маршрутов с метриками самих шлюзов и обновляет свою таблицу маршрутизации. Об обновлении чуть позже, а сейчас о нормальной работе RIP. Через некоторое время после первого обновления своей таблицы маршрутизации протокол начинает рассылку информации о маршрутах из этой самой таблицы. Рассылка периодически повторяется. Тем самым он даёт знать соседним шлюзам, что через него можно осуществлять пересылку по определённым маршрутам, и сообщает их стоимость.

Как же обновляется таблица маршрутизации? RIP имеет все полномочия для работы с ней - он может добавлять, обновлять и удалять маршруты. Первое происходит только в одном случае, если, получив очередную посылку от RIP соседнего шлюза и обработав её, он обнаружит маршрут, которого нет в таблице (то есть нет маршрута до некоторого адреса назначения) и цена которого не превышает 15 счётчиков трансляций. Попросту говоря, если какой-нибудь сосед окажется в состоянии обеспечить пересылку данных в доселе неведомый пункт назначения, да к тому же недорого (не дороже 15 счётчиков трансляции), то такой новый маршрут нужно добавить в таблицу маршрутизации.

Обновление и удаление маршрутов производится немного сложнее. Во-первых, получив очередную посылку от RIP соседнего шлюза, протокол может обнаружить, что тот предложил альтернативу уже существующему маршруту. Тогда, если новый маршрут окажется дешевле (его метрика будет меньше), то он добавляется в таблицу, а старый удаляется. Но как быть, если альтернативу предложил тот же самый шлюз, через который пролегает маршрут, содержащийся в таблице? Тогда этот маршрут не обновляется только в двух случаях - если его цена не изменилась (то есть, на самом деле, никакая это не альтернатива) и если его цена зашкалила за 15 счётчиков трансляций. В этом случае маршрут просто удаляется. Во всех остальных случаях маршрут обновляется, потому что таблица должна содержать свежую информацию.

Иногда от какого-нибудь шлюза перестают приходить посылки (машина перестала работать, связь оборвалась, выключили протокол RIP - мало ли что случилось). Тогда через некоторое время (обычно - около 3 минут) из таблицы удаляются все маршруты, пролегающие через забарахливший шлюз.

5.4. Внешняя маршрутизация

Как уже говорилось в начале этой главы, внешняя маршрутизация есть не что иное как обмен информацией о доступности между автономными системами. А функция протоколов маршрутизации, как внешних, так и внутренних, состоит в выборе "лучшего" маршрута и распространении информации о маршрутизации (в данном случае - информации о доступности). Давайте немного более подробно рассмотрим, что же такое автономные системы и информация о доступности.

В документе RFC 1163 (он описывает один из протоколов внешней маршрутизации, BGP) сказано так: "Классически автономную систему (АС) определяют как набор шлюзов под единым техническим контролем, использующих единый протокол внутренней маршрутизации и единые метрики для маршрутизации пакетов внутри АС и некоторый протокол внешней маршрутизации для передачи пакетов другим АС. ...другие автономные системы считают, что данная АС имеет единый согласованный план маршрутизации и цельную картину того, какие сети доступны в ней. С точки зрения внешней маршрутизации АС может рассматриваться как монолитная система..."

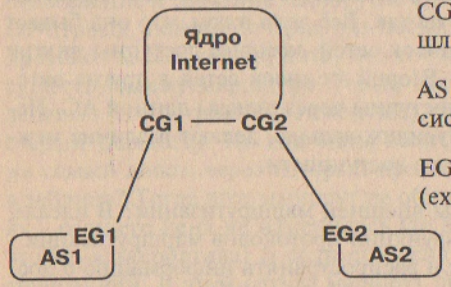
Вы спросите - ну а что же с информацией о доступности? Вроде бы с этим всё понятно? Как бы не так. Всё дело в том, что она бывает двух типов. Первый тип - список сетей, которые доступны внутри самой автономной системы. Второй - список сетей в других автономных системах, которые доступны через шлюзы данной АС. Некоторые протоколы, как мы увидим дальше, делают различие между этими типами информации о доступности.

Так что же делают протоколы внешней маршрутизации? В идеале, они должны выполнять все функции протоколов маршрутизации - выбирать "лучший" маршрут и распространять информацию о доступности. Но, оказывается, не все протоколы умеют делать первое. Один из них - протокол EGP. Но, несмотря на этот недостаток, он

до сих пор широко используется в Internet. Им мы и займёмся в следующем параграфе.

5.4.1. "Старый" протокол EGP

Протокол внешних шлюзов (EGP, Exterior Gateway Protocol) - один из ранних протоколов внешней маршрутизации в Internet. Он используется с 1984 года, когда эта сеть состояла из некоторого количества автономных систем, объединённых центральными шлюзами (рисунок 5.4.1.1). Эти шлюзы выполняли всю работу по маршрутизации между автономными системами. Для обмена информацией между центральными шлюзами и автономными системами использовался (и до сих пор продолжает использоваться) протокол EGP. Как было сказано выше, протокол внешней маршрутизации заведует обменом информацией о доступности. Но EGP обращает внимание на то, кто с кем при этом общается - либо шлюз автономной системы и один из центральных шлюзов, либо два центральных шлюза между собой. В первом случае шлюз автономной системы посылает только информацию о том, к каким сетям можно получить доступ внутри этой системы. Центральный шлюз в ответ шлёт информацию о том, к каким сетям можно получить доступ через "автономную систему" центральных шлюзов (как говорилось в начале этой главы, центральные шлюзы составляют так называемый "хребет" (backbone)). А когда происходит обмен между центральными шлюзами, то в ход идёт информация о том, до каких сетей можно добраться в автономных системах, за которые несут ответственность эти шлюзы. Таким образом, шлюзы автономных систем и центральные шлюзы выполняют неравнозначную работу.



CG1, CG2 - центральные шлюзы (core gateways)

AS1, AS2 - автономные системы (autonomous systems)

EG1, EG2 - внешние шлюзы (external gateways)

Рисунок 5.4.1.1: Иерархия шлюзов

Помимо собственно информации о доступности, EGP производит обмен метриками. Но дальше обмена дело не идёт, поскольку разные автономные системы по-разному их определяют. Из-за такой несовместимости метрик EGP не в состоянии сам выбрать "лучший" маршрут. В то же время, он и не разрабатывался для этого, ведь в старой, централизованной структуре не было выбора, потому что маршрут между автономными системами был единственный - через "хребет"

Со временем Internet разрослась и на поверхность выплыли недостатки EGP: неспособность выбора "лучшего" пути и плохая масштабируемость. Про первое мы уже упомянули, а второе является следствием первого, потому что чем сложнее связи между автономными системами, тем больше разных маршрутов, между которыми приходится делать выбор. Кстати, EGP по-прежнему применяют в централизованной структуре сети Milnet, но его также приспособили к новой модели маршрутизации, основанной на доменах маршрутизации (routing domains).

Домены маршрутизации - это всё те же самые автономные системы. Но теперь они обмениваются данными без посредников, напрямую, то есть теперь автономные системы общаются непосредственно друг с другом, а не через ядро Internet. Для этого им снова нужна внешняя маршрутизация, но EGP не всегда хорошо справляется с такой работой, потому что автономные системы могут быть связаны не одной, а несколькими парами шлюзов. И тут уже не обойтись без выбора "лучшего" маршрута, то есть шлюза. Для этой цели был придуман новый протокол внешней маршрутизации BGP. О нём речь идёт в следующем параграфе.

5.4.2. Восходящая звезда BGP

Ключевым отличием Протокола пограничных шлюзов (BGP, Border Gateway Protocol) от EGP является способность использовать информацию, которой обмениваются автономные системы, для выбора "лучшего" пути. Но почему EGP не мог этого делать? Потому что он воспринимал только информацию о доступности и разнородные, несовместимые метрики. BGP использует больше информации о каждом маршруте, поэтому способен их сравнивать. Эта дополнительная информация называется *атрибутами маршрута* (path attributes).

Атрибуты маршрута могут включать в себя информацию, необходимую для выбора маршрута по административному признаку. Та-

кая маршрутизация называется *стратегической* (policy based routing) и основывается на нетехнических причинах (например, политических, организационных, связанных с защитой) для выбора маршрута.

Благодаря использованию *атрибутов маршрута* BGP расширяет возможности для выбора "лучшего" маршрута и реализации разных *стратегий маршрутизации* (routing policies). Это очень важно для автономных систем, не использующих центральные шлюзы для внешней маршрутизации. Часто для этого нужны именно нетехнические правила, например, отправка пакетов от какого-то хоста из автономной системы через шлюз, не используемый больше никем (с самой высокой, и поэтому дорогой скоростью передачи). Кроме того, возможности BGP необходимы для эффективной реализации новой структуры сетей - сообщества примерно одинаковых автономных систем. Такая структура лучше расширяема (масштабируема), чем старая иерархическая.

6. Заключение

В этом пособии были рассмотрены основы организации семейства протоколов TCP/IP в том виде, в каком оно существует сейчас. Кроме того, читатель мог получить представление об устройстве протоколов TCP, IP, UDP, службе адресов DNS, и технологиях маршрутизации, включая протокол RIP.

Конечно, это только небольшая часть тех протоколов, которые используются в современном Internet. Здесь даже не были затронуты такие темы, как сетевое управление, защита информации, ограничение и авторизация доступа. Очень многое можно почерпнуть из документов RFC, в которых описано практически всё, что используется в настоящее время в Internet.

Разумеется, Internet не стоит на месте. В последние годы произошло невероятное увеличение интереса к этой сети сетей, а вместе с этим - взрыв в области разработки новых технологий. Сейчас уже никого не удивишь живым звуком или видеоизображением, передаваемым по Internet, очень скоро можно будет поселиться в виртуальной реальности и общаться с людьми в виде аватаров (объектов виртуальной реальности) со всей планеты.

Всё это многообразие возможностей и растущий интерес породили несколько проблем, которые только предстоит решить:

- **Нехватка IP-адресов.** Сейчас IP-адрес ограничен 32 битами. Это около миллиарда различных адресов. На самом деле не все IP-адреса используются эффективно, да это и невозможно по разным причинам. При дальнейшем росте числа подключенных к Internet пользователей и организаций свободных адресов будет оставаться всё меньше. Сейчас уже разрабатываются новые версии протокола IP, призванные решить эту проблему (кстати, в настоящий момент используется IP 4-ой версии). Версия 6 (называемая иногда IPng - IP new generation, новое поколение протокола IP) расширяет адрес до 128 бит. Этого достаточно, чтобы каждому кубическому микрону объёма земного шара дать 100000 IP-адресов. Но уже есть проекты, предполагающие подключение в Internet бытовых приборов, от телевизоров и специализированных приставок, до микроволновых печей и стираль-

ных машин, поэтому такое огромное количество IP-адресов скоро может стать реально необходимым.

- **Необходимость новых технологий маршрутизации.** В разделе "Протоколы маршрутизации" уже говорилось о том, что реализованная сейчас в IP техника маршрутизации не даёт возможность выбирать из нескольких альтернатив. Новые версии IP снимают эту проблему. Кроме того, есть реальная потребность в использовании для выбора маршрута не только технической, но и административной информации. Уже есть протоколы, дающие такую возможность, например BGP, но их внедрение потребует ещё очень много времени и средств.

- **Учёт.** Internet вырос из чисто научной сети ARPANet. Но сейчас в нём заправляют коммерческие компании, и это совершенно нормально, потому что без их денег не было бы такого роста. Кроме того, в силу своей природы Internet - открытая сеть, и пользователю нет никакого дела до технических деталей получаемой им услуги, зато он готов платить деньги за эту услугу. Поэтому существует необходимость учёта: использованных ресурсов, объёмов полученной и переданной информации, и так далее. Такая бухгалтерия позволила бы пользователям быть уверенными, что они не переплачивают, а компаниям понимать, откуда берутся дыры в их бюджетах, и как увеличить доход.

- **Управление сетями.** Это одна из самых больших проблем. По подсчётам Американского национального института стандартов и технологий, основанного на обзорах нескольких аналитических фирм, в 1994 году американские компании потратили около 200 миллиардов долларов на обслуживание сетей и их управление. Поэтому сейчас разрабатываются новые технологии автоматического и автоматизированного управления, которые позволили бы уменьшить эту цифру.

Мы только вступаем в Информационную эру, но уже ясно, что сети (компьютерные, телекоммуникационные и т.д.) будут играть в ней одну из главных ролей. Поэтому тем, кто в будущем станет заправлять компьютерной индустрией, то есть нынешним студентам, уже сейчас стоит обратить внимание на пополнение своих знаний о сетях данных. Автор скромно надеется, что это пособие в некоторой мере поможет их изучению, хотя бы в стенах родного Московского университета.

Приложение I. Рекомендуемая литература

Список литературы, рекомендуемой для более глубокого изучения семейства протоколов TCP/IP.

I.1. RFC (Request for Comments)

Это официальные документы Общества Internet. Здесь дан очень короткий список тех документов, которые автор рекомендует прочитать в первую очередь. Всего таких документов на октябрь 1996 года было около 2000. В списке ниже сначала стоит номер документа, потом его автор и название:

ICMP:
792 *Дж. Постел,* Протокол контрольных сообщений Internet (*Postel, J.* Internet Control Message Protocol)
IP:
791 *Дж. Постел,* Протокол Internet (*Postel, J.* Internet Protocol)
ARP:
826 *Д. Пламмер,* Протокол разрешения адресов Ethernet, или Конвертация адресов сетевых протоколов в 48-битный адрес Ethernet для передачи с помощью аппаратуры Ethernet (*Plummer, D.* Ethernet Address Resolution Protocol: Or converting network protocol addresses to 48.bit Ethernet address for transmission on Ethernet hardware)
TCP:
793, *Дж. Постел,* Протокол контроля передачи (*Postel, J.* Transmission Control Protocol)
UDP:
768 *Дж. Постел,* Протокол пользовательских дейтаграмм (*Postel, J.* User Datagram Protocol)
DNS:
1035 *П. Мокапетрис,* Имена доменов - реализация и спецификация (*Mockapetris, P.* Domain names - implementation and specification)
1034 *Мокапетрис П.,* Имена доменов - концепции и средства реализации (*Mockapetris, P.* Domain names - concepts and facilities)

RIP:
1058 *К. Хедрик,* Протокол информации о маршрутизации (*Hedrick, C.* Routing Information Protocol)
BGP:
1163 *К. Лаухид, Й. Рехтер,* Протокол пограничных шлюзов (*Lougheed, K.; Rekhter, Y.* Border Gateway Protocol (BGP))
Интересующие RFC можно переписать на сервере gatekeeper.dec.com с помощью программы FTP. Ниже приведена последовательность команд, необходимая для получения этих документов (наклонным шрифтом выделено то, что вы должны вводить с клавиатуры):

```
ftp gatekeeper.dec.com
user: anonymous
password: ваш адрес электронной почты
cd /pub/net/info/rfc
bin (если файл в сжатом виде или в формате PostScript) или
   ascii (если документ в виде текстового файла)
get имя-файла
```

Имена файлов сконструированы из аббревиатуры rfc, номера документа и расширения. Например, файл *rfc1058.txt* содержит описание протокола RIP в текстовом виде. Перед номером, состоящим из трёх цифр, ставится ноль, например, *rfc0768.txt* (правда, из этого правила есть исключения). Расширения могут быть разными: помимо файлов в текстовом виде (.txt) вы можете найти файлы в формате PostScript (расширение .ps), или в сжатом виде (расширение .Z). PostScript-файлы можно распечатать на PostScript-принтере, или просмотреть с помощью специальных программ (например, Ghostview). Сжатые файлы нужно распаковать программой compress:

```
compress -d имя-сжатого-файла.
```

I.2. Рекомендуемые книги

Andrew Tanenbaum, Computer Networks: 2nd edition. Prentice Hall, Englewood Cliffs, NJ, 1989
Andrew Tanenbaum, Computer Networks: 3rd edition. Prentice Hall, Englewood Cliffs, NJ, 1996
Великолепное введение в компьютерные сети, с достаточным количеством деталей. Рассказывает не только о TCP/IP, но и о протоколах более низкого уровня (Ethernet, Token Ring) и об устройстве компьютерных сетей. Третья редакция содержит более современную информацию, но более труднодоступна.

Douglas Comer, Internetworking with TCP/IP, Volume I: Principles, Protocols & Architectures. 2nd edition, Prentice Hall, Englewood Cliffs, NJ
Книга, очень подробно рассказывающая о TCP/IP, его принципах и специфике реализаций протоколов. На сегодняшний день является одной из лучших и авторитетнейших книг по TCP/IP.

Craig Hunt, TCP/IP Network Administration, O'Reilly & Associates, 1993
В книге Крэга Ханта изложены основы архитектуры TCP/IP, базовые протоколы, а также процедуры инсталляции и настройки этих протоколов для различных версий ОС UNIX. Эта книга являлась основной при подготовке данного пособия.

Paul Albitz, Cricket Liu, DNS and BIND in a Nutshell, O'Reilly & Associates, 1993
Рассмотрена архитектура, принципы работы, а также процедуры инсталляции и настройки службы имён DNS и программы BIND.

Эд Крол, Всё об Internet, Торгово-издательское бюро BHV, 1995
(Оригинал: *Ed Krol,* The Whole Internet, O'Reilly & Associates, 1994)
Отличная книга для тех, кто видел Internet один раз в жизни и даже не знает, как им пользоваться. Во введении есть немного информации об устройстве Internet, достаточно для понимания основ обычным пользователем.

Книги на английском языке можно получить в Библиотеке иностранной литературы или Государственной публичной научно-технической библиотеке, а также заказать через Internet (например, в виртуальном книжном магазине amazon.com).

Книгу Эда Крола на русском языке можно купить, например, в Доме книги на Новом Арбате или в магазине Библио-глобус на Мясницкой.

Приложение II. Глоссарий

ARP/ RARP
Address Resolution Protocol/ Reverse Address Resolution Protocol
Протокол разрешения адресов/ Протокол обратного разрешения адресов

ARP - протокол для определения низкоуровневого сетевого адреса по соответствующему адресу IP. RARP выполняет обратную функцию - по данному физическому адресу определяет соответствующий ему адрес IP. Оба этих протокола работают только в сетях с возможностью широковещательной передачи (см. broadcast).

ARPA
Advanced Research Projects Agency
Агентство передовых исследовательских проектов

Агентство при министерстве обороны США, заказавшее и профинансировавшее создание сети ARPANET (см. ARPANET).

ARPANET

Экспериментальная сеть, созданная в конце 60-х годов и послужившая полигоном для проверки концепций и протоколов, на которых построена сеть Internet. Уже не существует.

AS
autonomous system
автономная система

Сообщество сетей под единым административным контролем, использующих общий протокол внутренней маршрутизации.

ATM
Asynchronous Transfer Mode
Асинхронный режим передачи

Стандарт, который определяет эффективную, высокоскоростную (от 1,544 мегабод до 1,2 гигабод) технологию передачи данных с коммутацией пакетов фиксированного размера с динамическим распределением ширины полосы пропускания.

backbone
"хребет"

Система центральных шлюзов, через которые проходит основной поток данных.

BBN, Inc.
Bolt, Beranek, and Newman, Inc.
Болт, Беранек, энд Ньюмэн, Инк.

Одна из ведущих исследовательских фирм США в области

сетевых технологий. В ней была создана реализация семейства протоколов TCP/IP для операционной системы BSD (Berkeley) UNIX.

BGP
Border Gateway Protocol
Протокол пограничных шлюзов

Протокол внешней маршрутизации, определённый в документе RFC1771. Его особенностями являются так называемая стратегическая маршрутизация и возможность выбора "лучшего" маршрута в зависимости от значений различных параметров.

broadcast
широковещательная передача

Способ адресации, при котором все компьютеры локальной сети получают посланные данные.

datagram
дейтаграмма

Пакет информации, передаваемый по сети без уведомления о передаче и без подтверждения приёма.

default route
маршрут по умолчанию

Элемент таблицы маршрутизации, используемый для передачи пакетов сетям, не перечисленным в этой таблице.

demultiplexing
демультиплексирование

Процесс разбиения одного потока данных на несколько, в зависимости от их получателя.

DNS
Domain Name System
Служба имён доменов

Распределённая база данных, обеспечивающая преобразование имён компьютеров (типа www.cs.msu.su) в IP-адреса (типа 125.5.202.15).

EGP
Exterior Gateway Protocol
Протокол внешних шлюзов

Протокол внешней маршрутизации, который распространяет информацию среди внешних маршрутизаторов, подключенных к данной автономной системе (см. AS).

FTP
File Transfer Protocol
Протокол передачи файлов

Протокол, позволяющий пользователям на одном хосте получать доступ и копировать файлы с/на другой хост через сеть. Также FTP - это программа, позволяющая пользователю работать с этим протоколом.

Gateway
шлюз

Система, пересылающая данные между двумя разнородными логическими сетями (например, Internet и DECnet) или программами (например,

несовместимыми программами электронной почты).

Hello

Один из ранних протоколов внутренней маршрутизации, использовавший в качестве метрики при выборе маршрута задержку передачи пакета.

host
хост

Компьютер с доступом к сети, на котором работают прикладные программы.

HTML
Hypertext Markup Language
Язык разметки гипертекста

Язык, предназначенный для описания страниц и связей между ними в World Wide Web (см. WWW).

HTTP
Hypertext Transfer Protocol
Протокол передачи гипертекста

Специализированный протокол передачи файлов, лежащий в основе World Wide Web (см. WWW).

ICMP
Internet Control Message Protocol
Протокол контрольных сообщений Internet

Расширение протокола IP, позволяющее передавать информационные сообщения и сообщения об ошибках, а также тестовые пакеты.

IESG
Internet Engineering Steering Group
Группа управления проектированием Internet

Организация, принимающая решения о стандартизации протоколов в Internet. Подчиняется Обществу Internet (см. ISOC).

internet

Сообщество физически разнородных сетей, использующих для связи единый протокол.

Internet

Всемирное сообщество сетей, использующих для связи протокол IP.

intranet

Корпоративная сеть какой-либо организации, используемая для нужд этой организации, с ограниченным доступом извне.

IP
Internet Protocol
Протокол Internet

Протокол межсетевого уровня семейства протоколов TCP/IP. Это протокол коммутации пакетов без установления соединения. Различают 4-ю версию протокола IP (стандартную) и 6-ю версию (экспериментальную), которую также называют IPng (IP new generation - новое поколение IP).

ISDN
Integrated Services Digital Network

Цифровая сеть с интеграцией услуг

Цифровая телефонная служба и одновременно технология передачи цифровой информации по медной витой паре. Отличие от обычной телефонной сети в том, что голос передаётся в цифровом виде и одновременно с данными.

ISO
International Organisation for Standardisation
Международная организация стандартизации

Организация, в рамках которой разрабатываются международные стандарты. Среди прочих этой организацией был разработан стандарт OSI (см. OSI).

ISOC
Internet Society
Общество Internet

Некоммерческая организация, координирующая работу Internet и внедрение технологий, используемых в этой сети.

ISP
Internet Service Provider
Поставщик связи с Internet

Организация (как правило, коммерческая), предоставляющая доступ к Internet.

loopback address
зацикленный адрес

IP-адрес 127.0.0.1, который соответствует адресу локального хоста. Данные, посланные хостом по этому адресу, придут на этот же хост, не покидая его.

multiplexing
мультиплексирование

Процесс объединения пакетов из нескольких потоков данных в один поток, без объединения пакетов.

NFS
Network File System
Сетевая файловая система

Протокол доступа к файлам на другом хосте как к файлам на локальной машине.

NSFNET
National Science Foundation Network
Сеть национального научного фонда США

Одна из составляющих, одна из частей "хребта" (см. backbone) Internet.

octet
октет

Байт, состоящий ровно из восьми бит.

OSI
Open Systems Interconnect
Взаимодействие открытых систем

Стандарт Международной организации стандартизации, система сетевых протоколов.

OSPF
Open Shortest Path First
Протокол "Выдай кратчайший маршрут первым"

Протокол внутренней маршрутизации, основанный на так называемом *состоянии соединения* в пику протоколам, осно-

ванным на дистанционно-векторном алгоритме.

protocol
протокол

Формальное описание сообщений и правил, по которым два субъекта обмениваются этими сообщениями.

RFC
Request for Comments

Серия документов, начатая в 1969 году, описывающая семейство протоколов, используемых в Internet, и эксперименты с ними.

RIP
Routing Information Protocol
Протокол информации о маршрутизации

Протокол внутренней маршрутизации, основанный на дистанционно-векторном алгоритме.

router
маршрутизатор

Система, пересылающая данные между двумя разнородными физическими сетями.

SMTP
Simple Mail Transfer Protocol
Простой протокол передачи почты

Протокол для передачи электронной почты. Используется обычно между двумя почтовыми серверами, в то время как другие протоколы используются для получения пользователем доступа к электронной почте.

SNMP
Simple Network Management Protocol
Простой протокол сетевого управления

Протокол, использующийся для управления узлами в Internet. Узлами могут быть не только компьютеры пользователей, но также маршрутизаторы, шлюзы, и вообще всё, что подключено к Internet.

TCP
Transmission Control Protocol
Протокол контроля передачи

Один из основных протоколов TCP/IP (см. TCP/IP), протокол надёжной передачи данных транспортного уровня с установлением логического соединения.

TCP/IP

Семейство протоколов, которое лежит в основе Internet.

TTL
Time to Live
время жизни

1. Поле в заголовке IP-пакета, указывающее, как долго этот пакет может путешествовать по сети, прежде чем его удалят. Обычно измеряется в счётчиках ретрансляций (количестве хостов, через которые пройдёт этот пакет).

2. Параметр записи о ресурсе в файлах данных DNS, определяющий, через какое время эта запись перестанет быть правильной и должна быть удалена.

UDP
User Datagram Protocol
Протокол пользовательских дейтаграмм

Один из основных протоколов TCP/IP (см. TCP/IP), протокол передачи дейтаграмм, не устанавливающий предварительно соединения и не обеспечивающий надёжность.

WWW
World Wide Web
Всемирная паутина

Гипертекстовая и мультимедийная система поиска ресурсов Internet и доступа к ним.

Типография ордена "Знак Почета" издательства МГУ
119899, Москва, Воробьевы горы.
Заказ N 1232 Тираж 300 экз.