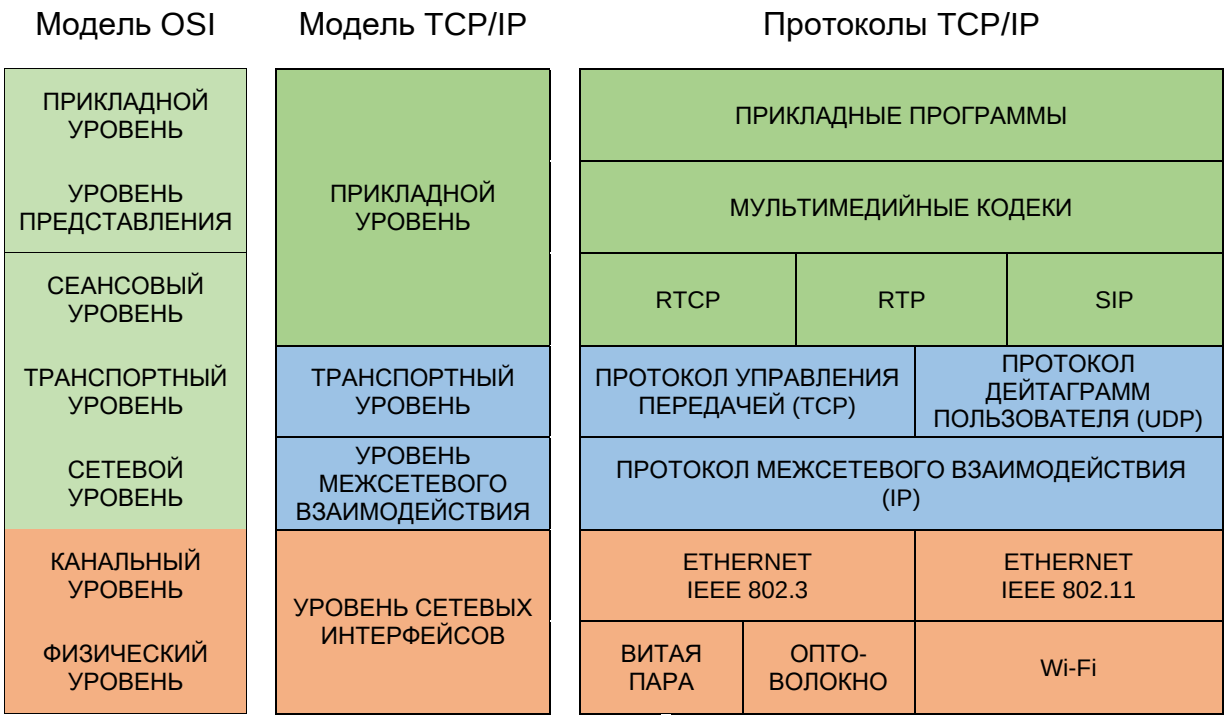
А.Харитонов, А.Джаманкулов

КОМПЬЮТЕРНЫЕ СЕТИ И ПРОТОКОЛЫ ПЕРЕДАЧИ ДАННЫХ



Санкт-Петербург
2023

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

УНИВЕРСИТЕТ ИТМО

А.Харитонов, А.Джаманкулов

КОМПЬЮТЕРНЫЕ СЕТИ И ПРОТОКОЛЫ ПЕРЕДАЧИ ДАННЫХ

УЧЕБНОЕ ПОСОБИЕ

РЕКОМЕНДОВАНО К ИСПОЛЬЗОВАНИЮ В УНИВЕРСИТЕТЕ ИТМО
по направлению подготовки 11.03.02 Инфокоммуникационные технологии и системы связи
в качестве учебного пособия для реализации основных профессиональных образовательных программ высшего образования бакалавриата

Санкт-Петербург
2023

Харитонов А., Джаманкулов А. Компьютерные сети и протоколы передачи данных – СПб: Университет ИТМО, 2023. – 136 с.

Рецензент(ы):
Береснев Артем Дмитриевич, старший преподаватель (квалификационная категория "старший преподаватель") факультета инфокоммуникационных технологий, Университета ИТМО.

Учебное пособие посвящено описанию теории компьютерных систем и изучению сетевого стека OSI. Рассмотрены основные протоколы передачи данных в компьютерных сетях на транспортном, прикладном и физическом уровнях сетевого стека. Дополнительно рассмотрены наиболее часто использующиеся вспомогательные сетевые службы.

Содержание

ВВЕДЕНИЕ.....5
1. ВВЕДЕНИЕ В КОМПЬЮТЕРНЫЕ СЕТИ.....7
1.1. История развития компьютерных сетей.....7
1.2. Модели передачи данных в компьютерных сетях.....12
1.3. Реализация моделей передачи данных в стеках протоколов TCP/IP, IPX/SPX.....16
2. СЕТЕВОЙ УРОВЕНЬ МОДЕЛИ OSI.....18
2.1. Адресация узлов в компьютерных сетях.....18
2.2. Разделение сетей на подсети (03-Network Level. IP-routing).....22
2.3. Специализированные адреса.....25
2.4. Маски переменной длины (VLSM).....26
2.5. Задача определения маршрутов между узлами.....27
2.6. Маршрутизация в глобальной сети.....35
2.7. Формат пакета в соответствии с протоколом IP.....49
2.8. Протокол ICMP.....54
2.9. Протокол IGMP.....55
3. ПРОТОКОЛЫ ТРАНСПОРТНОГОУРОВНЯ.....57
3.1. Протокол UDP (User Datagram Protocol).....57
3.2. Протокол TCP (Transfer Control Protocol).....59
4. ВСПОМОГАТЕЛЬНЫЕ СЕТЕВЫЕ СЛУЖБЫ.....72
4.1. Протокол DHCP.....72
4.2. Служба DNS.....74
5. ПРОТОКОЛЫ ПРИКЛАДНОГО УРОВНЯ.....80
5.1. Протокол HTTP.....80
5.2. Протокол FTP.....86
5.3. Почтовые протоколы.....90
5.4. Протоколы эмуляции удаленного терминала.....93
5.5. Прочие протоколы прикладного уровня.....94
6. ПЕРЕДАЧА ДАННЫХ НА ФИЗИЧЕСКОМ УРОВНЕ.....96

6.1. Основы передачи данных на физическом уровне.....96
6.2. Основные характеристики линий связи.....99
6.3. Физический уровень глобальных сетей.....102
6.4. Иерархия цифровых потоков.....106
ПРИЛОЖЕНИЕ 1. ПЕРЕЧЕНЬ СЕТЕВЫХ ПРОТОКОЛОВ.....108
ПРИЛОЖЕНИЕ 2. ПЕРЕЧЕНЬ ПОРТОВ С УКАЗАНИЕМ НАЗНАЧЕНИЯ.....125
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....134

ВВЕДЕНИЕ

Сегодня без компьютерных сетей невозможно представить ни одну отрасль человеческой деятельности. Это одна из самых главных и быстроразвивающихся технологий. Компьютерными сетями в разной степени объединены и суперкомпьютеры, и сервера, и персональная вычислительная техника, и даже оборудование на основе микроконтроллеров. Происходит плотная интеграция Интернета с мобильными и беспроводными технологиями. Стремительно растет "Интернет вещей", к Сети подключаются городские службы, транспорт, средства мониторинга. Развиваются как любительские системы передачи электронных данных, так и передовые способы на основе последних достижений технологий связи.

Глобальная сеть Интернет стала неотъемлемой частью культуры и, буквально, потребностью в повседневной жизни большинства людей, она представляет собой колоссальные запасы информации, созданные цивилизацией за тысячи лет своего существования и продолжающие пополняться со всё возрастающей скоростью.

Целью данного учебного пособия является формирование у студентов теоретических и практических знаний в проектировании и анализе компьютерных сетей на различных уровнях.

Этой цели соответствует изучение следующих пунктов:

- аппаратное и программное обеспечение компьютерных сетей;
- способы приема и передачи информации по сети;
- различия локальных и глобальных сетей;
- конфигурации сетевых средств;
- диагностика работы сетей

В первой части пособия представлена кратко ретроспектива развития компьютерных сетей, их модели и составляющие. В последующих частях составляющие разобраны более подробно. В частности – семиуровневая модель OSI (Open System Interconnection). Рассмотрены вспомогательные службы работы компьютерных сетей.

Задача пособия – дать студентам базовые сведения об основных понятиях, организации и структуре компьютерных сетей. Оно предназначено для реализации основных профессиональных образовательных программ высшего образования бакалавриата при изучении курса «Компьютерные сети». Однако может быть полезно и студентам других специальностей, магистрантам и аспирантам, а также студентам средних специальных учебных заведений.

После каждого раздела пособия студентам предложены вопросы для самопроверки. предназначено для студентов технических специальностей высших учебных заведений, получающих образование в области информационных технологий.

Учебное пособие помогает в реализации следующих компетенций обучающегося, формируемых в результате освоения дисциплины «Компьютерные сети»:

1. ОК-12 (общекультурные компетенции): способность понимать сущность и значение информации в развитии современного информационного общества, сознавать опасности и угрозы, возникающие в этом процессе, соблюдать основные требования информационной безопасности, в том числе защиты государственной тайны;
2. ОК-8 (общекультурные компетенции): готовностью использовать основные методы, способы и средства получения, хранения, переработки информации, готовностью работать с компьютером как средством управления информацией;
3. ОК-9 (общекультурные компетенции) способностью работать с информацией в глобальных компьютерных сетях.

Рекомендации преподавателям:

- учебное пособие наиболее целесообразно применять при подготовке и проведении лекций дисциплины «Компьютерные сети»;
- материалы пособия можно использовать при подготовке и проведении лабораторных занятий: «Разделение сетей на подсети настройке сетей», «Маршрутизация в глобальной сети», «Почтовые протоколы» и других лабораторных работ.

Рекомендации студентам:

- после каждой лекции во время самостоятельной работы для совершенствования знаний и компетенций предлагается составлять индивидуальный опорный конспект кратких ответов на вопросы самопроверки по соответствующему разделу пособия; в последующем этот конспект можно будет использовать при подготовке к экзамену по дисциплине;
- для самостоятельной работ при подготовке к лабораторным работам особое внимание предлагаем обратить на протоколы транспортного уровня TCP и UDP;
- для оформления реферативной части отчетов по лабораторным работам рекомендуется использовать материалы разделов 2, 3 и 5;
- при выполнении лабораторных работ будет целесообразным внимательно изучить и проанализировать сведения о развитии компьютерных сетей в контексте появления новых протоколов и стандартов.

Материалы пособия помогают педагогу провести занятия в том числе и в форме семинара, способствуют освоению студентами общих компетенций в соответствие с требованиями ФГОС.

Пособие может использоваться как обзорное для желающих познакомиться с основными принципами построения современных компьютерных сетей. В том числе оно может быть полезно для практикующих специалистов, работающих в сфере телекоммуникаций.

1. ВВЕДЕНИЕ В КОМПЬЮТЕРНЫЕ СЕТИ

1.1. История развития компьютерных сетей

Прежде чем начать изучение компьютерных сетей, осуществим краткий экскурс в историю их развития. Процесс развития компьютерных сетей тесно связан с процессом совершенствования и развития ЭВМ. Данный процесс можно условно разбить на следующие этапы:

Этап 1 (1950–1960 гг.). Данный этап – это период первых ЭВМ. Громоздкость, энергоемкость и дороговизна первых ЭВМ позволяли использовать их практически только в пакетном режиме. Данный режим подразумевал подготовку программы решения некоторой задачи на носителе, ввод информации с носителя и получение результата (рис. 1.1).

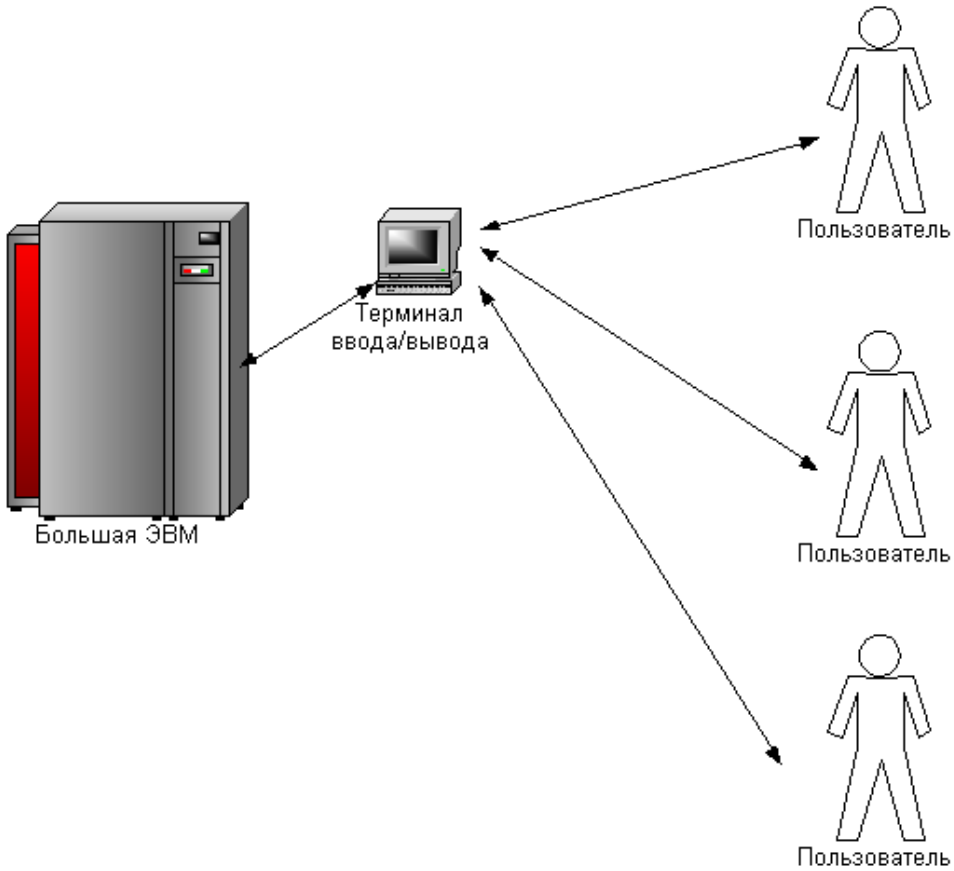


Рис. 1.1. Пакетный режим общения с первыми ЭВМ

При этом практически отсутствовала возможность интерактивного взаимодействия пользователя и ЭВМ. Т.е. пользователь не видел реакции ЭВМ на свои команды, а получал ответ в виде конечного результата работы программы.

Этап 2 (1960-1970 гг.). На данном этапе происходит удешевление ЭВМ и введение принципов интерактивной работы пользователей. Каждый пользователь получал в свое распоряжение диалоговое устройство – терминал, посредством которого мог общаться с ЭВМ. Вычислительные ресурсы, внутренняя и внешняя память центральной ЭВМ предоставлялась в использование всем пользователям. Терминалы в совокупности с центральной ЭВМ представляли собой территориально распределенную структуру, которую можно считать прообразом современной локальной сети (рис. 1.2).

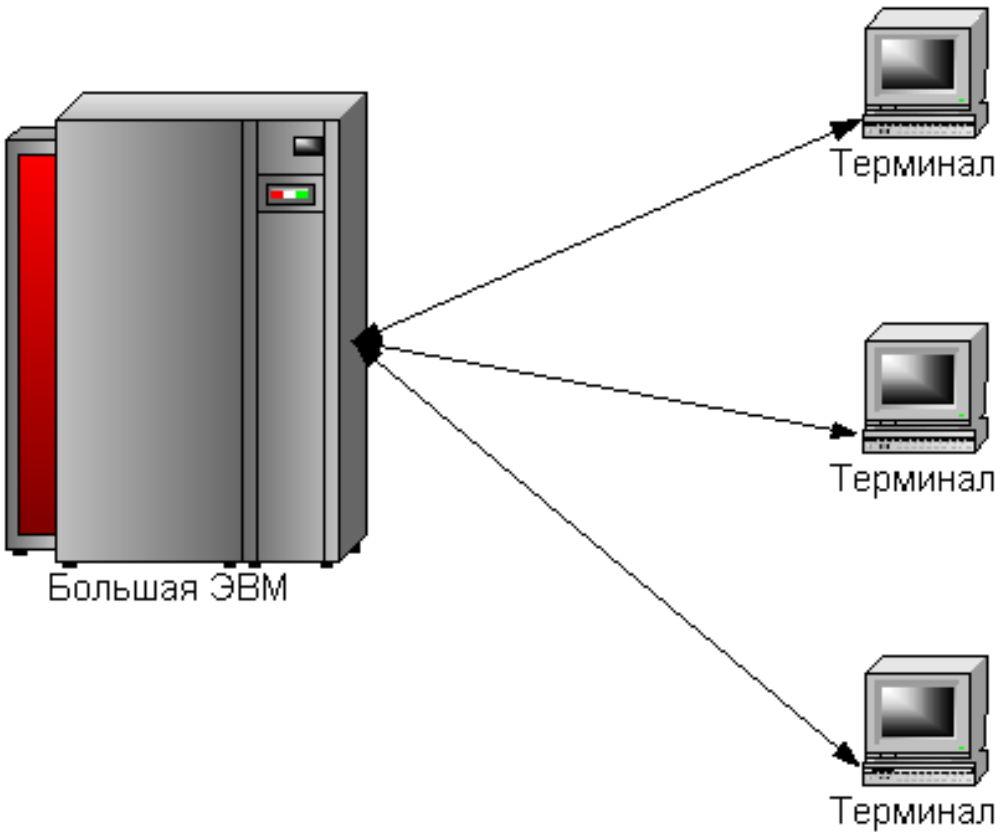


Рис. 1.2. Терминальный режим общения с большими ЭВМ

Однако есть существенная разница, т.к. в данном случае обработка информации оставалась централизованной, что противоречит современному пониманию компьютерных сетей.

Этап 3 (1960 гг. – наши дни). Данный этап – это этап объединения удаленных (возможно на сотни и тысячи км.) ЭВМ (рис. 1.3). Толчком к такому

объединению явилась необходимость подключатся к ЭВМ с терминалов, расположенных достаточно далеко.

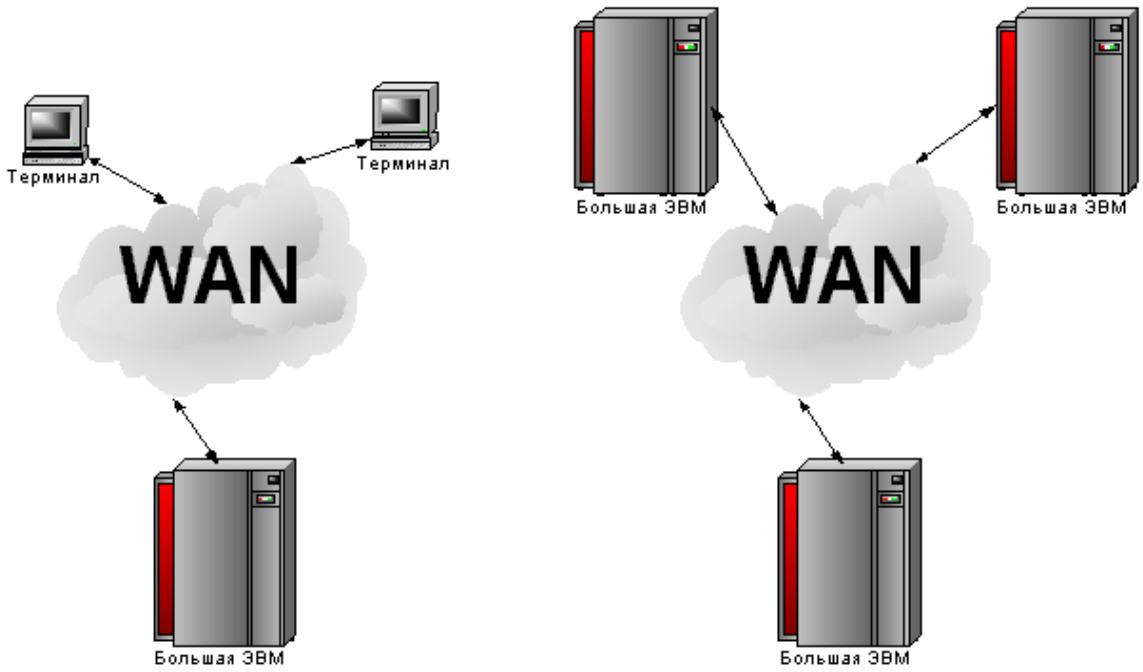


Рис. 1.3. Примеры образования глобальных сетей

Для решения подобных задач были использованы интенсивно развивающиеся в то время телефонные сети. Первоначально скорость передачи данных ограничивалась использованием аналоговых технологий передачи, однако уже с конца 60-х широко стали применяться цифровые каналы передачи данных. Именно это время можно считать основным этапом в становлении так называемых первичных сетей. Дальнейшее развитие цифровых каналов передачи данных, появление новых технологий и расширение пропускной способности продолжается до сих пор. Таким образом, исторически первыми появились именно составляющие глобальной сети WAN **(Wide Area Network)**. В современном мире широко применяется термин GAN (**Global Area Network**) для обозначения всемирной сети, которая объединяет территориально распределенные сети WAN [1].

Этап 4 (1970 гг. – наши дни). Данный этап – это время становления локальных сетей LAN **(Local Area Network),** которое происходит в 70-х годах одновременно с технологическим прорывом, который привел к появлению микропроцессоров, БИС, СБИС и резкому удешевлению ЭВМ, уменьшению их энергоемкости и габаритов. Появление на предприятиях и в организациях большого числа ЭВМ привело к необходимости объединения их между собой для решения общих задач, обмена данными и т.п. (рис. 1.4).

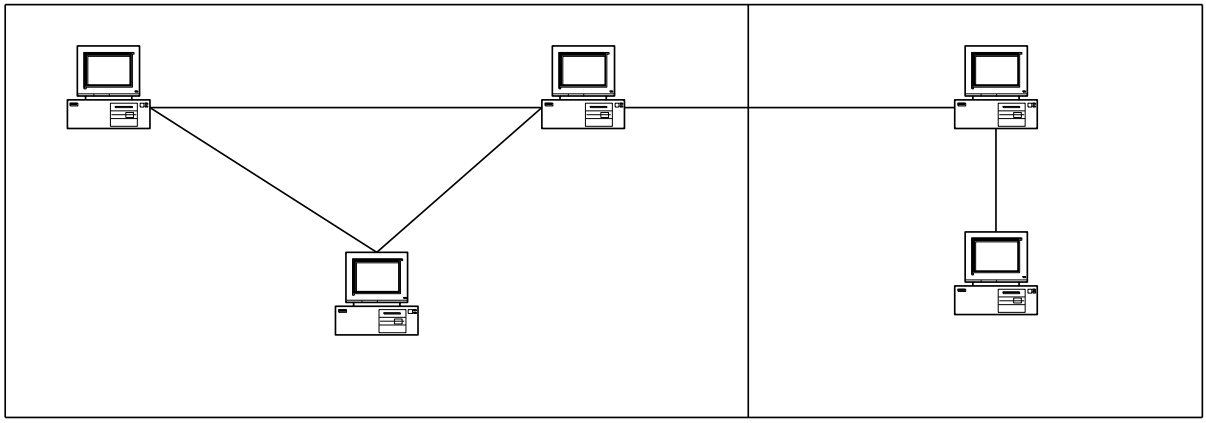


Рис. 1.4. Пример образования локальной сети

Последовательно к середине 1980-х были выработаны более-менее устоявшиеся стандарты на аппаратное обеспечение сетей и протоколы передачи данных (Ethernet, Arcnet, TockenRing, FDDI и т.п.). К концу 1990-х был выявлен явный лидер – технология Ethernet, которая является на данный момент наиболее распространенной технологией локальных сетей.

Последний этап следует рассмотреть более подробно. Первоначально возникла необходимость осуществлять обмен данными между компьютером и некоторыми периферийными устройствами, что в упрощенном виде можно считать прототипом сетевого взаимодействия. Далее возникла необходимость в связи компьютер-компьютер. Для этого использовались практически все имеющиеся на то время интерфейсы (Centronics, RS232, PS/2 и т.д.). При этом программные протоколы и аппаратные интерфейсы были часто несовместимы друг с другом, что приводило к необходимости использовать массу всевозможных устройств сопряжения. И только при возникновении необходимости объединять произвольное число компьютеров в единую среду стали зарождаться технологии Ethernet, TockenRing, Arcnet и т.п. Процесс становления локальных сетей наглядно представлен на рис. 1.5.

В дальнейшем время отбирало из массы разрабатываемых стандартов и технологий наиболее эффективные, удобные и надежные [1].

Отдельно следует обратить внимание на взаимосвязь между локальными и глобальными сетями. На ранних этапах глобальные сети отличались значительной территориальной распределенностью, использованием низкоскоростных линий связи и минимальным набором предоставляемых услуг, а локальные сети – малой территориальной распределенностью, использованием качественных высокоскоростных линий связи и богатым набором услуг. С течением времени данные различия сгладились, и в современном мире мы имеем ситуацию, в которой локальные сети предприятия могут быть представлены узлами, расположенными на расстоянии в тысячи километров и связанными высокоскоростными линиями связи, которые в свою очередь являются

составляющими глобальных сетей. Т.е. наблюдается конвергенция или взаимопроникновение сетей.

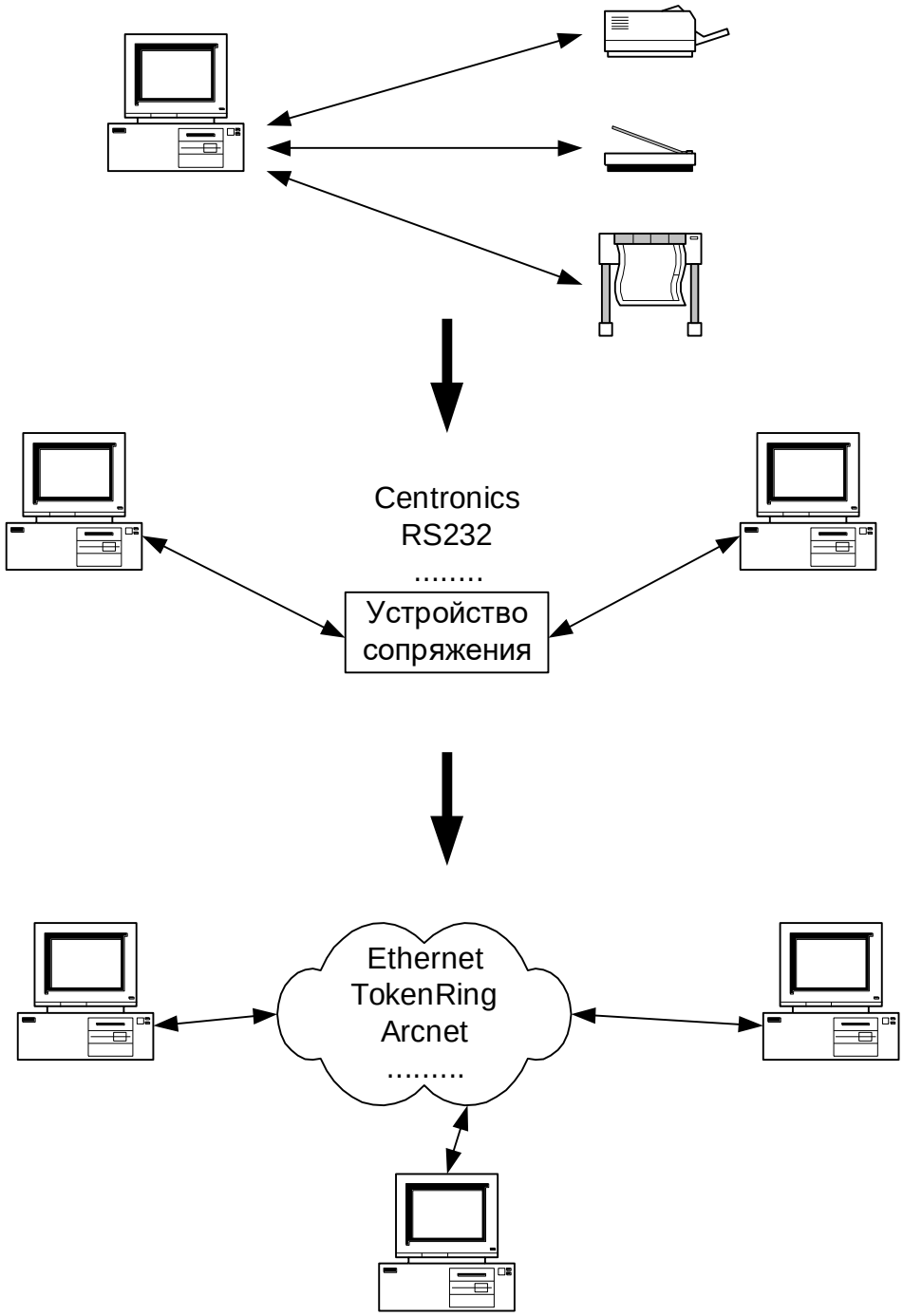


Рис. 1.5. Схема становления локальных сетей

Подобный процесс также происходит между компьютерными и телекоммуникационными сетями. Это связано с тем, что прокладка качественных, высокоскоростных линий связи большой протяженности – задача

трудоемкая и дорогостоящая. И поскольку на момент развития глобальных компьютерных сетей уже были заложены глобальные телекоммуникационные линии связи, наиболее выгодно было использовать уже имеющуюся инфраструктуру для передачи данных. Такое решение привело к тому, что первичные сети как основу глобальных компьютерных сетей составляют именно телекоммуникационные линии связи.

На данный момент к локальным сетям чаще всего относят сети, расположенные в пределах помещения, офиса, предприятия или объединяющие узлы, принадлежащие одной организации, предприятию и т.п. Локальные сети могут быть как подключены к глобальным, так и быть автономными.

Обобщая вышесказанное, классификацию сетей по территориальному признаку можно представить схемой на рис. 1.6. MAN (**Metropolitan Area Network**) – это термин, которым часто называют сети масштаба города [2]..

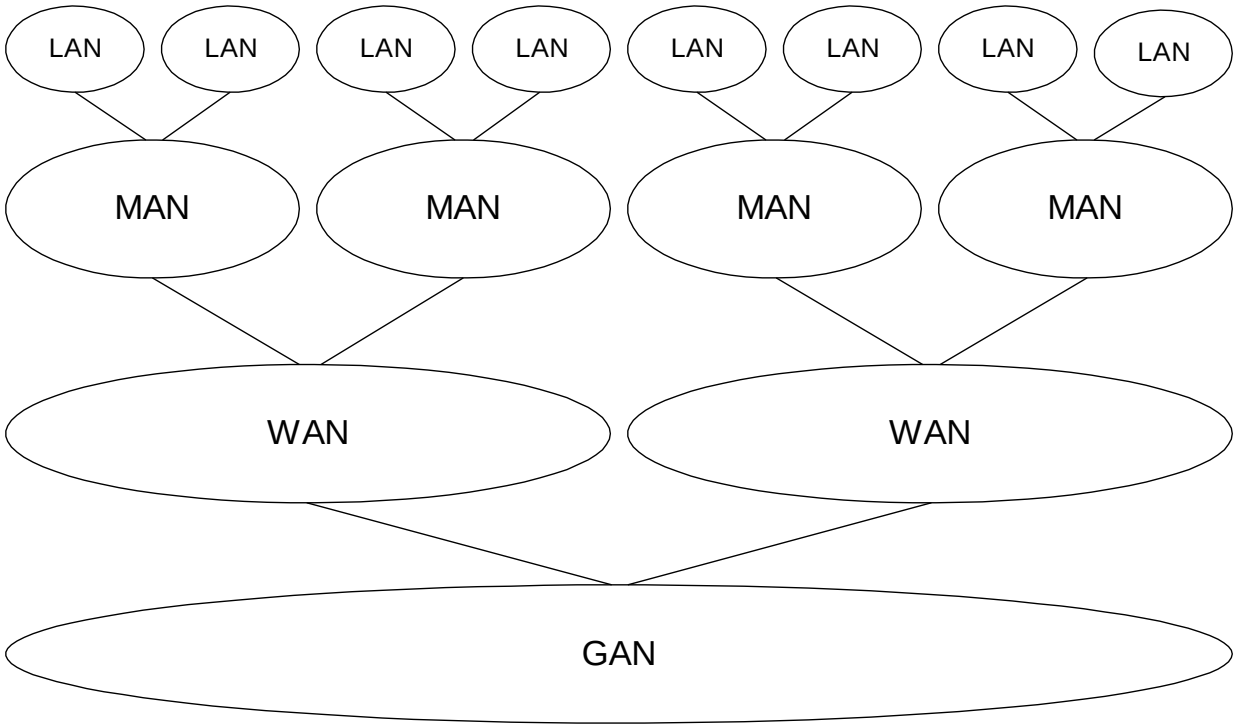


Рис. 1.6. Классификация сетей по территориальному признаку

1.2. Модели передачи данных в компьютерных сетях

Объединение компьютеров в сеть вызвано чаще всего следующими причинами:

- необходимость интенсивного обмена информацией;
- обеспечениедоступа к общим (разделяемым) ресурсам;

- выполнение распределенных приложений (приложений, части которых выполняются одновременно на нескольких ЭВМ, а сеть используется как средство обмена данными и синхронизации).

Объединение компьютеров в сеть осуществляется с помощью линий связи и соответствующего набора стандартного оборудования. Данное оборудование представляет собой аппаратное обеспечение сетей. Для реализации определенных стандартных процедур передачи данных посредством аппаратного обеспечения необходимо, чтобына каждом компьютере функционировали программные модули, отвечающие за работу сети. Данные модули представляют собой программное обеспечение сетей.

Теперь мы готовы дать соответствующее определение:

Компьютерная сеть – совокупность программных и аппаратных средств для передачи информации между узлами. Узлом компьютерной сети может являться компьютер или любое другое устройство, которое вырабатывает, преобразует или потребляет информацию.

Чтобы ясно представить себе сущность процесса передачи информации от одного узла сети к другому, существуют различные модели представления сетевого взаимодействия. Одной из наиболее известных является модель OSI (**Open System Interconnection**). Данная модель позволяет получить наиболее полное представление о процессах, происходящих при передаче информации между узлами.

Модель строится исходя из следующих соображений: существуют два узла А – передающий и В – принимающий. Из узла А в узел В передается некоторый блок информации (например, текстовое сообщение). Весь процесс передачи и приема разбивается на этапы (уровни), на каждом из которых решается свой набор задач. В модели OSI 7 уровней. Общий вид модели представлен на рис. 1.7.

Рассмотрим задачи, решаемые на каждом из уровней данной модели.

Прикладной (**Application layer**) – уровень прикладных программ, при работе которых возникает необходимость во взаимодействии с другими приложениями посредством передачи данных через сеть. Примером программ, работающих на данном уровне, могут служить броузер, почтовая программа, сетевая игра и т.д.

Представления данных (**Presentation layer**) – данный уровень решает задачу преобразования информации из представления, используемого на прикладном уровне, в представление, используемое для передачи данных через сеть. Такое преобразование может заключаться в кодировании, сжатии, фрагментации и т.п. действиях.

Рис. 1.7. Модель сетевого взаимодействия OSI

Сеансовый (**Session layer**) – данный уровень решает задачу синхронизации процессов, происходящих на узлах, участвующих в передачи данных. Синхронизация заключается в установлении моментов начала передачи, конца передачи, согласовании параметров передачи между узлами и т.п. Т.е. данный уровень фактически отвечает за установление сеанса связи с другим узлом сети.

Транспортный (**Transport layer**) – данный уровень обеспечивает доставку информации от одного узла к другому с выполнением требуемых условий по надежности, скорости доставки и др. параметрам.

Сетевой (**Network layer**) – данный уровень решает задачу адресации узлов в сети, задачу определения маршрута от одного узла к другому и задачу согласования параметров сетей, через которые проходят данные при взаимодействии узлов.

Канальный(**Data Link Layer**) – данный уровень обеспечивает согласование между сетевым и физическим уровнем. Т.е. преобразует данные в вид, удобный для передачи по каналам связи с использованием одной из известных технологий.

Физический (**Physical layer**) – на данном уровне происходит фактическая передача данных по каналам связи. Каналы связи могут быть представлены проводными линиями связи, радиоканалом, оптическими линиями связи и т.п. Соответственно данные могут передаваться в виде электрических, световых сигналов, в виде электромагнитных волн и т.п.

Данные, передаваемые через сеть, проходят вышеописанные уровни при передаче на узле А и, соответственно, в обратном порядке при приеме на узле В. Уровни пронумерованы от 1 до 7. Данная нумерация часто используется для указания, к какому уровню относится блок данных или на каком уровне работает то или иное оборудование.

До этого мы постоянно употребляли сочетание «блок данных». Однако для этого существуют более точные термины: пакет и кадр.

Пакет (кадр) – это блок данных, передаваемых через сеть как единое целое.

На уровнях 7-3 принято употреблять термин «пакет» (packet), а на уровнях 2-1 – «кадр» (frame).

Каждый пакет (кадр) состоит из тела и заголовков. Общий вид пакета можно представить структурой на рис. 1.8.

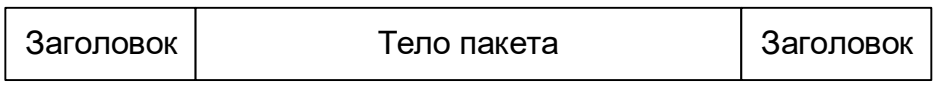


Рис. 1.8. Структура пакета данных

На каждом уровне пакет более высокого уровня является телом для пакета более низкого уровня. Т.е. фактически пакеты более высоких уровней упаковываются внутрь пакетов более низких уровней. Данный процесс можно представить схемой на рис. 1.9.

На физическом уровне происходит фактически не упаковка пакета, а его преобразование в последовательность электрических, световых или иных видов сигналов. Соответственно, при приеме происходит распаковывание пакетов в обратном порядке. Такая процедура обработки данных соответствует схеме LIFO (последний пришел – первый ушел).

Функции каждого из уровней в реальной жизни реализуются посредством протоколов передачи данных.

Протокол – совокупность форматов данных и правил их обработки, предназначенных для решения некоторой задачи при реализации взаимодействиями между узлами сети.

1.3. Реализация моделей передачи данных в стеках протоколов TCP/IP, IPX/SPX

Поскольку задачи, решаемые на разных уровнях многочисленны и разнообразны, то существует порядка тысячи протоколов, реализующих

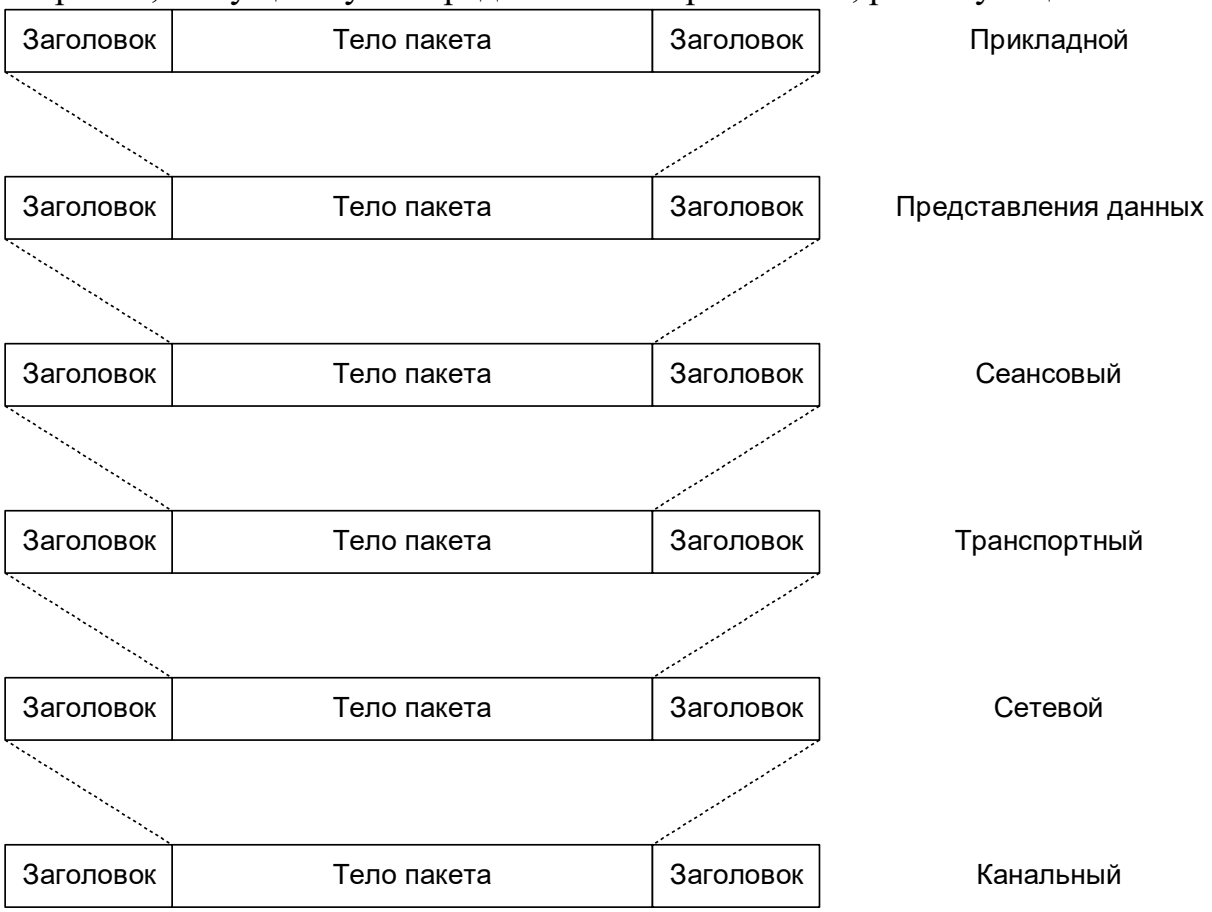


Рис. 1.9. Схема упаковывания пакетов при прохождении через уровни модели OSI

различные аспекты сетевого взаимодействия на различных уровнях с учетом различных технологий и типов оборудования. Последовательное рассмотрение протоколов начнем с сетевого уровня. Т.е. на первом этапе изучения компьютерных сетей мы абстрагируемся от нижних уровней, привязанных к технологиям и оборудованию. Уровни 3-7 можно выделить в отдельную группу, чаще всего реализуемую одной из нескольких групп протоколов. Группу протоколов, функционирующую на общих принципах и правилах, часто называют семейством или стеком. Второе название происходит от сходства процедуры обработки пакетов (LIFO) с процедурой работы стека.

Существуют различные семейства протоколов: TCP/IP, IPX/SPX, OSI, NetBIOS/SMB и т.д. Наиболее широко известным и базовым для многих операционных систем и сетей является именно семейство TCP/IP (**Transmission**

Control Protocol / Internet Protocol). Более детальная информация о других семействах будет дана после изучения семейства TCP/IP.

Соответствие наиболее распространенных стеков протоколов модели OSI показано на рис. 1.10.

<table>
<tr><th>Модель OSI</th><th>TCP/IP</th><th>IPX/SPX</th><th>OSI</th><th>NetBIOS/SMB</th></tr>
<tr><td>Прикладной</td><td rowspan="2">HTTP,FTP,POP3,SMTP, SNMP, Telnet, SSH</td><td rowspan="3">NCP
SAP</td><td>X.400, X.500, FTAM</td><td rowspan="2">SMB</td></tr>
<tr><td>Представления данных</td><td rowspan="3">OSI</td></tr>
<tr><td>Сеансовый</td><td rowspan="2">TCP
UDP</td><td rowspan="2">NetBIOS</td></tr>
<tr><td>Транспортный</td><td>SPX</td></tr>
<tr><td>Сетевой</td><td>IP, ICMP, IGMP
OSPF, RIP, BGP</td><td>IPX, RIP</td><td>ES-ES
IS-IS</td><td></td></tr>
<tr><td>Канальный</td><td colspan="4">Ethernet, TokenRing, FDDI, SLIP, PPP</td></tr>
<tr><td>Физический</td><td colspan="4">Коаксиал, витая пара, оптика, радиоволны</td></tr>
</table>

Рис. 1.10. Соответствие модели OSI и популярных стеков протоколов

Конечно, рис. 1.10 дает лишь малое представление обо всей массе существующих протоколов. Более подробный перечень семейств и входящих в них протоколов можно найти в Приложении 1.

Вопросы для самопроверки

1. Перечислить этапы становления компьютерных сетей
2. Описать модель сетевого взаимодействия OSI
3. Рассмотреть задачи прикладного уровня
4. Рассмотреть задачи сетевого уровня
5. Рассмотреть задачи физического уровня

2. СЕТЕВОЙ УРОВЕНЬ МОДЕЛИ OSI

Как было сказано выше, изучение стека протоколов TCP/IP мы начнем с сетевого уровня. Вспомним основные задачи сетевого уровня:

- адресация узлов в сети;
- определение маршрутов продвижения пакетов между узлами;
- согласование параметров сетей, по которым проходят пакеты.

Базовыми для решения вышеперечисленных задач является протокол IP.

Прежде чем подробно рассмотреть формат пакета данного протокола, рассмотрим то, как протокол решает возложенные на него задачи.

2.1. Адресация узлов в компьютерных сетях

Для адресации узлов сети на сетевом уровне используется так называемый IP-адрес. IP-адрес – это числовой идентификатор, который присваивается каждому узлу (компьютеру), который подключен к сети. На данный момент существует два стандарта на протокол IP и IP-адресацию: IPv4 и IPv6. И по этим стандартам IP-адрес узла имеет формат, изображенный на рис. 2.1.

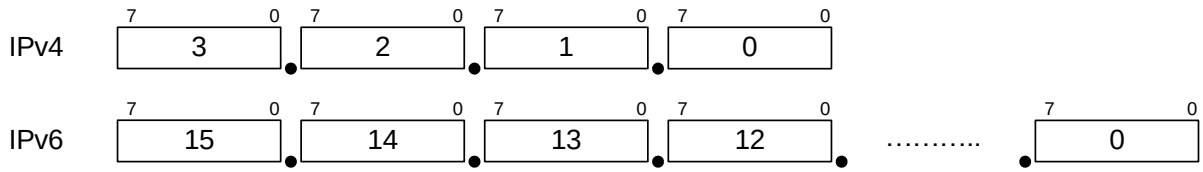


Рис. 2.1. Формат IP-адресов стандартов IPv4, IPv6

Как видно из рисунка, IP-адрес состоит из 4 или 16 байтов, которые при записи отделяются точками. Основное внимание мы сосредоточим на версии IPv4.

Простейшим методом адресации узлов в сети могла быть сквозная нумерация всех узлов, которые подключены к сети. Но такое решение задачи “в лоб” является неэффективным. Во-первых, организационно практически невозможно централизованно нумеровать все возможные узлы. Во-вторых, при сквозной нумерации усложняется поиск маршрутов от одного узла к другому, поскольку очень тяжело создать эффективную систему сквозной нумерации, которая бы позволяла по номеру узла однозначно определить его расположения и маршрут к нему. Поэтому адресацию решили построить таким образом. На рис. 2.2 приведен пример сети, где показано, что узлы всегда можно разделить на группы – сети. В этом случае для того, чтобы указать узел, необходимо указать номер сети, к которой относится узел, и номер узла внутри этой сети. Например, для узла, который отмечен на рис. 2.2, адрес будет выглядеть так: сеть №3, узел №1.

При таком методе адресации организационное управление нумерацией разбивается на уровни. На высших уровнях выполняется нумерация сетей, а на низших нумерация узлов в границах одной сети. При этом значительно облегчается задача маршрутизации, поскольку расположение узла легко определить по номеру сети, к которой он относится, а в пределах сети место узла определяется по его номеру. Конечно, предшествующее объяснение несколько упрощенное, но оно наглядно отображает принцип адресации с помощью IP-адресов.

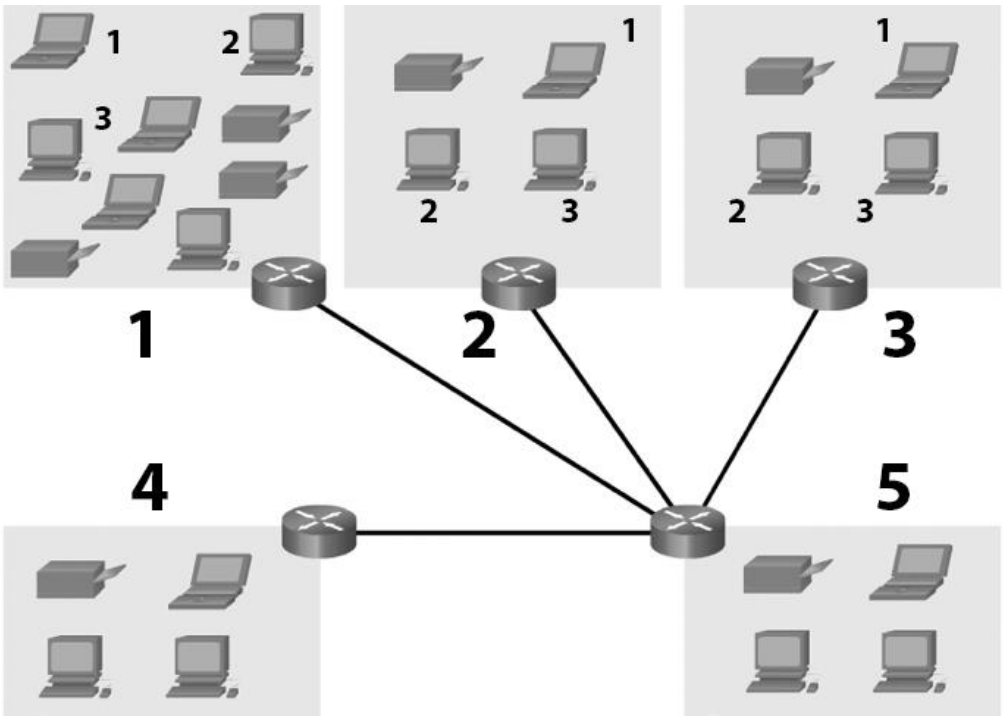


Рис. 2.2. Упрощенный вариант сети, которая состоит из нескольких подсетей

Таким образом мы определили, что адрес узла состоит из двух компонент: номера сети и номера узла. При этом возникает проблема, какое количество разрядов адреса отвести под номер сети, а какое – под номер узла. Если избрать какую-то единую схему (например, 2 байта – номер сети, 2 байта – номер узла), то возникнет ограничение – максимум 2^{16}сетей и 2^{16} узлов. Чтобы не создавать ограничений и сделать механизм адресации гибким, все IP-адреса поделили на классы, в зависимости от количества сетей и узлов, которые необходимо адресовать.

В соответствии со стандартом IPv4 существует 4 класса адресов: A, B, C, D. Распределение адресов на классы выполняется по схеме в табл. 2.1.

Использование такого строгого деления на классы стало тормозом при быстром темпе роста количества сетей и узлов. Поэтому для повышения гибкости механизма адресации были предложены некоторые усовершенствования.

Прежде чем рассмотреть современное состояние адресации по протоколу IPv4, рассмотрим понятия: сетевая маска, подсеть и широковещательный адрес.

*Сетевая маска (**network mask**) – это числовой параметр, который состоит, как и IP-адрес, из 4 байтов (для IPv4) и предназначен для определения адреса сети по заданному IP-адресу.*

Фактически маска – это IP-адрес, в котором в поле “номер сети” все разряды равняются единице, а все разряды в поле “номер узла” – нулю. Таким образом, для классов A, B, C маски будут иметь вид:

- A – 255.0.0.0 (число 255 в двоичной системе имеет вид 11111111)
- B – 255.255.0.0
- C – 255.255.255.0

Если известен IP-адрес узла, то для определения адреса сети, к которой относится данный узел, необходимо к IP-адресу и к маске применить операцию побитовой конъюнкции (логического И).

Например, задан IP-адрес 192.168.1.8 и маска 255.255.255.0. Определим адрес сети:

* Двойка отнимается от максимального количества двоичных комбинаций из-за того, что существует два специализированных

номера, которые будут рассмотрены ниже.

** Класс D содержит групповые адреса, то есть такие, по которым можно обращаться к нескольким узлам, объединенным в группу.

*** Значения 240...255 являются резервными.

IP-адрес:	$192.168.1.8_{10}$	=	$11000000\ 10101000\ 00000001\ 00001000_2$
Операция логического И:		&	
Сетевая маска:	$255.255.255.0_{10}$	=	$11111111\ 11111111\ 11111111\ 00000000_2$
Адрес сети:	$192.168.1.0_{10}$	=	$11000000\ 10101000\ 00000001\ 00000000_2$

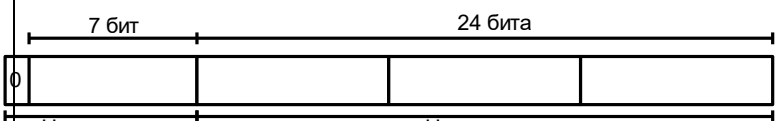
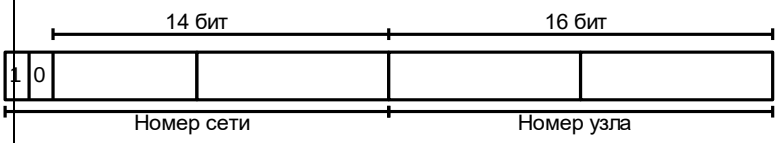
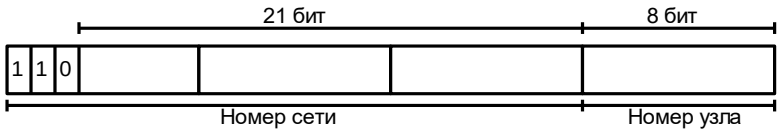
Как видим, адрес сети – это фактически IP-адрес, у которого в поле “номер узла” все разряды равняются нулю. Таким образом, имея IP-адрес и маску сети, всегда можно определить адрес сети. Этот адрес используется аппаратным и программным обеспечением для определения, к какой сети относится узел, IP-адрес которого указан в пакете, который передается по протоколу IP.

*Широковещательный адрес (**broadcast address**) – это специализированный адрес, который означает сразу все узлы, которые входят в состав данной сети.*

То есть пакет, который направлен по такому адресу, получат все узлы, которые относятся к данной сети. Используется такой адрес в случаях, если необходимо прислать пакет сразу всем узлам данной сети. Выглядит такой адрес так: все разряды поля «номер узла» равняются единице. Чтобы получить широковещательный адрес по заданному IP-адресу, необходимо к IP-адресу и

инверсии маски применить операцию побитовой дизъюнкции (логического ИЛИ).

Табл. 2.1. Классы IP-адресов

Класс	Формат IP-адреса	Значение старшего байта	Максимальное количество сетей	Максимальное количество узлов*	Маска сети
A		0 … 127	2^7 -1	2^{24} -2	255. 0. 0. 0
B		128 … 191	2^{14}	2^{16} -2	255. 255. 0. 0
C		192 … 223	2^{21}	2^8 -2	255. 255. 255. 0
D **		224 … 239 ***	–	–	–

Например, задан IP-адрес 192.168.1.8 и маска 255.255.255.0. Определим широковещательный адрес:

IP-адрес:	$192.168.1.8_{10}$	=	$11000000\ 10101000\ 00000001\ 00001000_2$
Операция логического ИЛИ:		&	
Инверсия сетевой маски:	$0.0.0.255_{10}$	=	$00000000\ 00000000\ 00000000\ 11111111_2$
Широковещ. адрес:	$192.168.1.255_{10}$	=	$11000000\ 10101000\ 00000001\ 11111111_2$

Как видим, среди номеров узлов существует два специализированных номера:

- адрес сети (номер узла содержит все нули);
- широковещательный адрес (номер узла содержит все единицы).

Именно из-за этих специализированных адресов максимальное количество узлов, которое определяется для каждого класса, уменьшается на 2 от максимального количества двоичных комбинаций [3].

2.2. Разделение сетей на подсети (03-Network Level. IP-routing)

Рассмотрим понятие подсеть и то, как они получаются. Простейшим средством построения сети предприятия, учреждения или другой организации любого уровня является объединение всех узлов (компьютеров) в одну сеть. Пример такого построения приведен на рис. 2.3. Сеть относится к классу А и имеет маску 255.0.0.0.

Следует отметить, что сети – это динамические объекты, которые имеют свойство изменять свою структуру, размеры и т.д. Поэтому в начале построения сети тяжело предусмотреть, какая адресация будет более эффективной. Однако при увеличении количества узлов, объединенных в одну сеть, возрастает нагрузка на сеть. Кроме того, возникают проблемы, связанные с ограничением доступа одних узлов к другому. Скажем, в примере, приведенном на рис. 2.3, возможно увеличение количества узлов в каждом отделе: «менеджеры», «технический отдел» и «бухгалтерия».

При этом может возникнуть необходимость отделить технический отдел от бухгалтерии, или менеджеров от технического отдела. Этого можно достичь разными путями, но наиболее эффективным и простым является структуризация сети, то есть ее разбиение на группы узлов (подсети).

В нашем примере используется сеть класса А с маской 255.0.0.0. Разбить сеть на подсети можно, если заменить маску класса А на маску, которая позволяет больше разрядов отвести под номер сети. Заменим маску класса А на маску меньшего класса – В. В этом случае мы получим следующий результат:

Сеть класса А.
Адрес сети – 10.0.0.0.
Маска сети – 255.0.0.0.

Разбиваем сеть на подсети класса В, используя маску 255.255.0.0. В этом случае под номер сети также отводится второй байт IP-адреса. Поскольку старший байт равняется 10 и определяет сеть, то следующий (второй) байт будет определять номер подсети. Пронумеруем подсети: «технический отдел» – 1,

«менеджеры» – 2, «бухгалтерия» – 3. В результате получим трех подсети класса B:

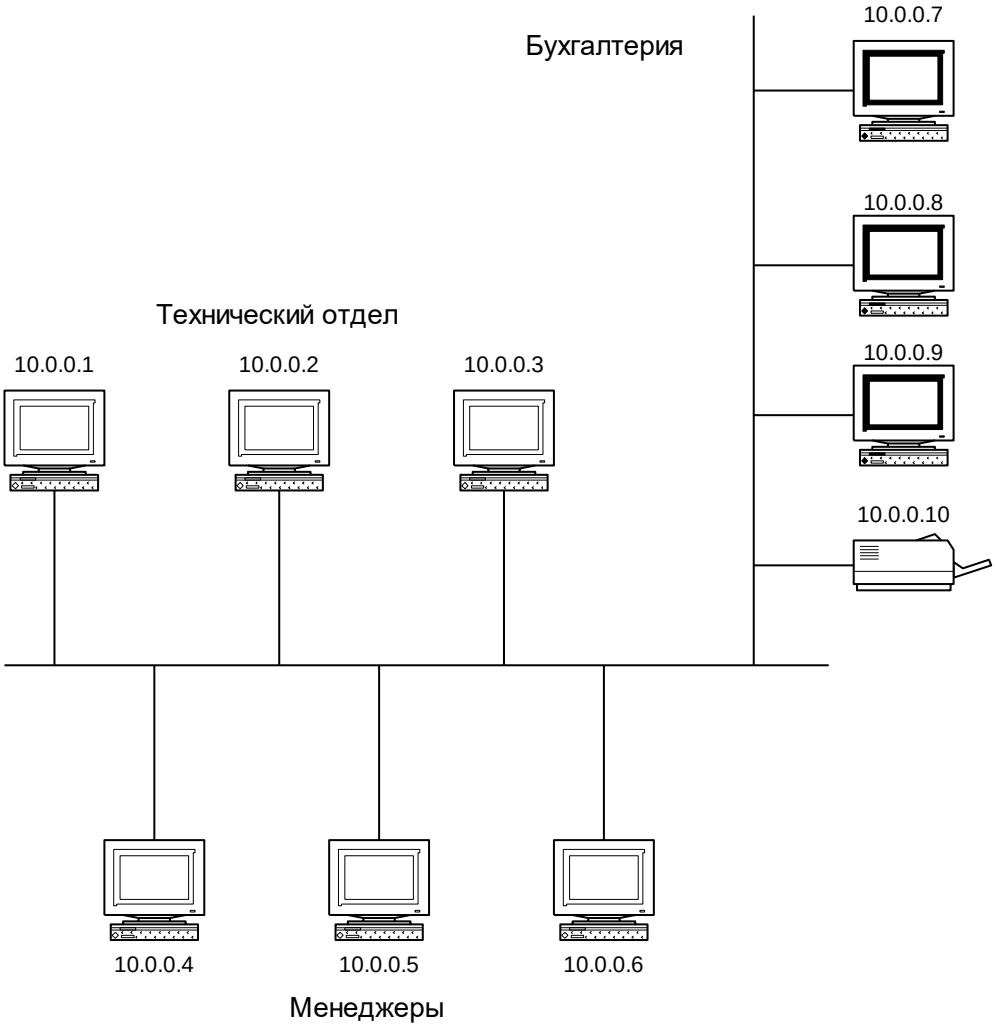


Рис. 2.3. Пример локальной сети предприятия

«Технический отдел»
Адрес подсети 10.1.0.0
Маска подсети 255.255.0.0
«Менеджеры»
Адрес подсети 10.2.0.0
Маска подсети 255.255.0.0
«Бухгалтерия»
Адрес подсети 10.3.0.0
Маска подсети 255.255.0.0

Соответственно, в каждой сети узлы нумеруются с использованием двух младших байтов IP-адреса. Результат распределения можно видеть на рис. 2.4. (M – это устройство, которое объединяет сети – так называемый маршрутизатор, который будет рассмотрен ниже).

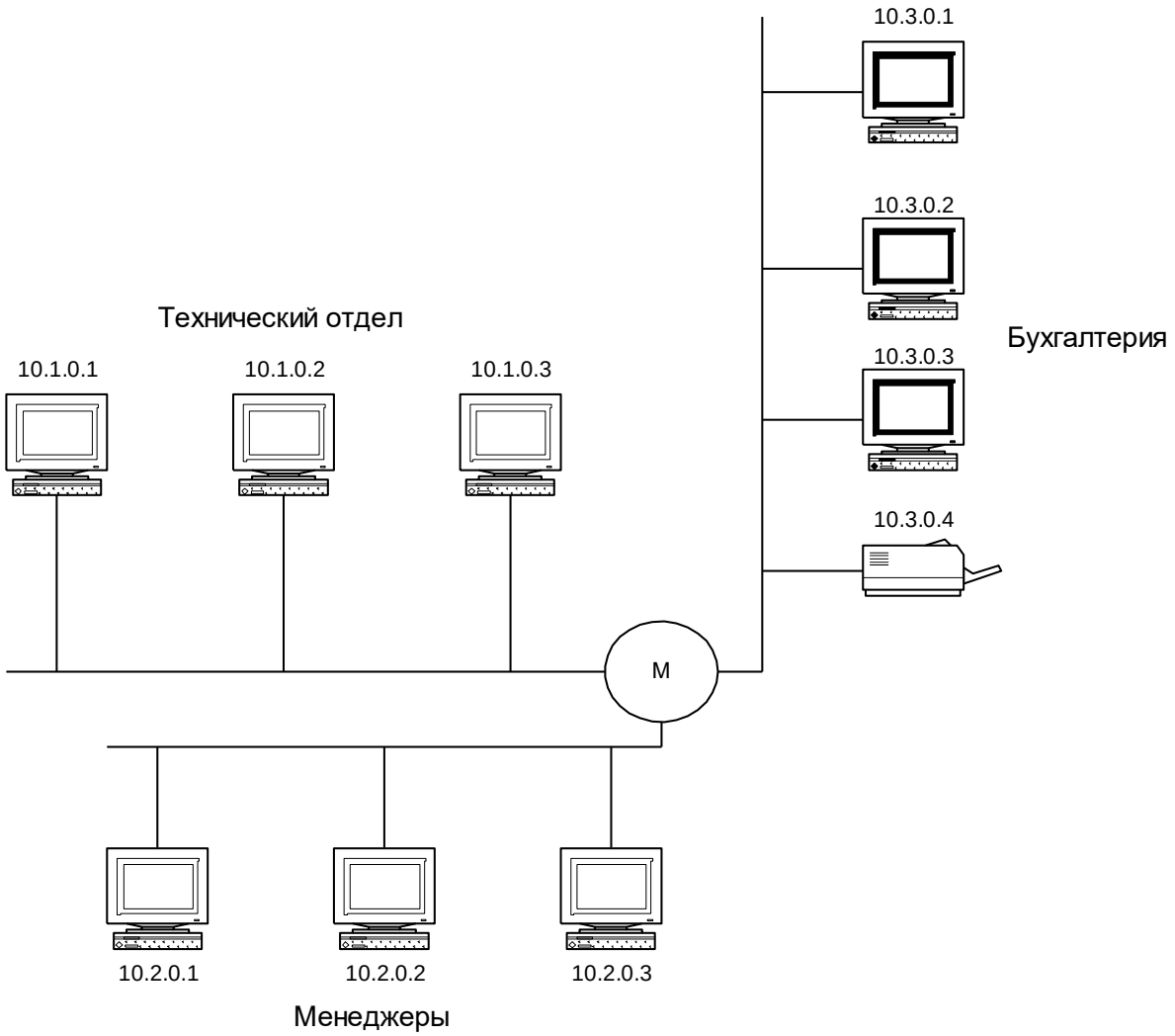


Рис. 2.4. Локальная сеть класса A разбитая на подсети класса B

Приведенный пример является очень простым. Имея сеть класса A, всегда можно разбить ее на подсети класса B или C. Также сеть класса B всегда можно разбить на подсети класса C. Вообще существует правило: если в сети заданного класса мы заменяем маску сети на маску, которая отводит больше разрядов под номер сети, то мы разбиваем сеть на подсети низшего класса.

Разбиение на подсети не только разрешает уменьшить нагрузку на сеть, отделить одни группы узлов от других, но еще и облегчает задачу администрирования сети.

Локальная сеть является преимущественно изолированной от других сетей, в том числе, глобальных сетей. Такое построение сетей позволяет в границах локальной сети использовать произвольные IP-адреса, за исключением специализированных. Вообще требование уникальности IP-адреса справедливо в пределах одного из типов сетей: глобальной или локальной. Фактически это

означает следующее: если Вы подключены к глобальной сети – Ваш узел должен иметь уникальный IP-адрес, если у Вас есть локальная сеть, то каждый узел в ней также должен иметь уникальный IP-адрес, но в разных локальных сетях IP-адреса могут совпадать [4].

2.3. Специализированные адреса

В стандарте IPv4 определены следующие специализированные адреса (табл. 2.2):

- Ограниченный адрес отправителя – это адрес, который узел имеет до того, как ему будет предоставлен IP-адрес (до загрузки ОС или указания адреса вручную).
- Ограниченный широковещательный адрес – это адрес всех узлов, которые подключены к данной сети.

Табл. 2.2 Специализированные IP-адреса

Адрес	Назначение
0.0.0.0	Ограниченный адрес отправителя
255.255.255.255	Ограниченный широковещательный адрес
IP-адрес, номер узла в котором содержит все единицы	Широковещательный сетевой (подсетевой) адрес
127.*.*.*	Программный интерфейс loopback
• Сети 192.168.0.0 – 192.168.255.0 с маской 255.255.255.0 • Сети 172.16.0.0 – 172.31.0.0 с маской 255.255.0.0 • Сеть 10.0.0.0 с маской 255.0.0.0	Адреса, которые рекомендованы для использования в локальных сетях

Программный интерфейс loopback – это адрес, который означает самого себя. То есть пакеты, которые отправляются на этот адрес, не выходят из узла, а сразу принимаются самим отправителем. Этот адрес используется для тестирования сети и программ.

Использование для локальных сетей адресов, которые указаны в последней строке табл. 2.2, не является обязательным. Как было сказано выше, в локальных сетях, которые изолированы от глобальной сети, можно применять произвольные адреса. Применение именно таких адресов просто является “правилом хорошего тона” администратора сети.

2.4. Маски переменной длины (VLSM)

При наличии большого количества сетей и подсетей, в особенности в классе С, возникает проблема ограниченности стандартного деления на классы, поскольку класс С невозможно поделить на подсети так, как было сделано в примере.

Преодолеть такое ограничение помогает использование метода VLSM (***Variable Length Subnet Mask***) – использование масок подсети переменной длины. Этот метод может быть применен для преодоления ограничений в сетях всех классов. Метод предусматривает, что маска может содержать произвольное количество единиц, а не только 8, 16 или 24. То есть маска не обязательно должна заканчиваться на границе байта.

Например: $11111111.11111111.11110000.00000000_2$=$255.255.240.0_{10}$.

Часто для указания сети с использованием VLSM используют запись вида: 192.168.128.0/20 (адрес сети / префикс), где 192.168.8.0 – адреса сети, а 20 – количество единиц в маске, считая с левой стороны IP-адреса. Такая форма записи называется CIDR (***Classless Interdomain Routing***).

Рассмотрим пример для сети, приведенной на рис. 2.3, считая, что адреса сети относится к классу С. Например, 192.168.8.0 с маской 255.255.255.0. Задача перед нами то же самая: разбить сеть на три подсети.

Согласно разбивке на классы, меньшего класса, чем С, не существует, то есть придется под номер подсети отвести часть младшего байта IP-адреса. Это возможно при использовании VLSM. Пусть под номер подсети отводится 4 разряда младшего байта. Нумерация подсетей будет такой же, как и в предшествующем примере.

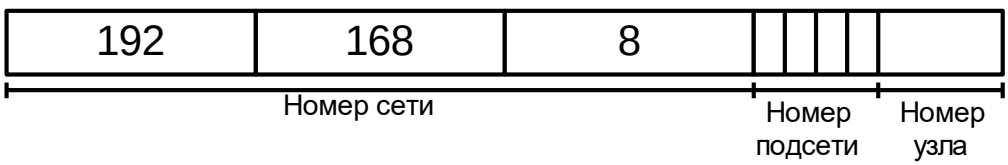


Рис. 2.5. Формат IP-адреса при использовании VLSM

Закодируем подсети в двоичной системе $1_{10} = 0001_2$, $2_{10} = 0010_2$, $3_{10} = 0011_2$. Тогда IP-адреса подсетей будут следующими:

“технический отдел”	11000000.10101000.00001000.00010000	192.168.8.16
“менеджеры”	11000000.10101000.00001000.00100000	192.168.8.32
“бухгалтерия”	11000000.10101000.00001000.00110000	192.168.8.48
маска подсетей	11111111.11111111.11111111.11110000	255.255.255.240

Для определения широковещательного адреса для любой подсети будем использовать стандартное правило: операцию логического ИЛИ с инверсией маски.

“технический отдел” 11000000.10101000.00001000.00010000
| 00000000.00000000.00000000.00001111
11000000.10101000.00001000.00011111 192.168.8.31

“менеджеры” 11000000.10101000.00001000.00100000
| 00000000.00000000.00000000.00001111
11000000.10101000.00001000.00101111 192.168.8.47
“бухгалтерия” 11000000.10101000.00001000.00110000
| 00000000.00000000.00000000.00001111
11000000.10101000.00001000.00111111 192.168.8.63

Таким образом нумеровать узлы в границах подсетей можно как в табл. 2.3.

Табл. 2.3. IP-адреса для подсетей

Подсеть*	Широковещательный адрес	Диапазон адресов
192.168.8.16/28	192.168.8.31	192.168.8.17-192.168.8.30
192.168.8.32/28	192.168.8.47	192.168.8.33-192.168.8.46
192.168.8.48/28	192.168.8.63	192.168.8.49-192.168.8.62

* Адрес записан в виде: адрес подсети/количество единиц в маске.

Таким образом, изменяя произвольно количество разрядов, которые отводятся под номер сети и номер узла, можно гибко управлять организацией сети.

Следует отметить, что в сетях с использованием VLSM существует отдельный набор специализированных адресов: 172.16.0.0/12 – рекомендованных для использования в локальных сетях.

Несмотря на то, что при использовании VLSM длину маски можно изменять произвольно, существует одно ограничение. Максимальная длина маски – 30 единиц. Данное ограничение связано с тем, что при длине маски 31 разряд под номер узла остается один разряд. С помощью одного разряда можно пронумеровать два узла. Но стандарт предусматривает, что два номера всегда заняты – это номер сети (подсети) и широковещательный адрес. Это значит, что при длине маски 31 разряд невозможно пронумеровать ни одного узла. Из-за этого длина маски ограничена 30 разрядами и минимальный размер сети – 2 узла (2^2-2).

2.5. Задача определения маршрутов между узлами.

Если все узлы объединены в одну сеть, то проблемы с доставкой пакетов каждому узлу не возникает. Поскольку сеть одна, то каждый узел получает пакет и, проверив IP-адрес назначения, может определить, принадлежит этот пакет ему или нет. В случае если в одну сеть объединено много подсетей, возникает

проблема, как передавать пакеты между сетями. То есть в тех узлах, в которых подсети объединяются, необходимо иметь возможность выбирать направление, в котором передавать пакеты в зависимости от того, к какой подсети относится узел–приемник пакета. Например, для сети, приведенной на рис. 2.6, существует одна точка, в которой объединяются подсети.

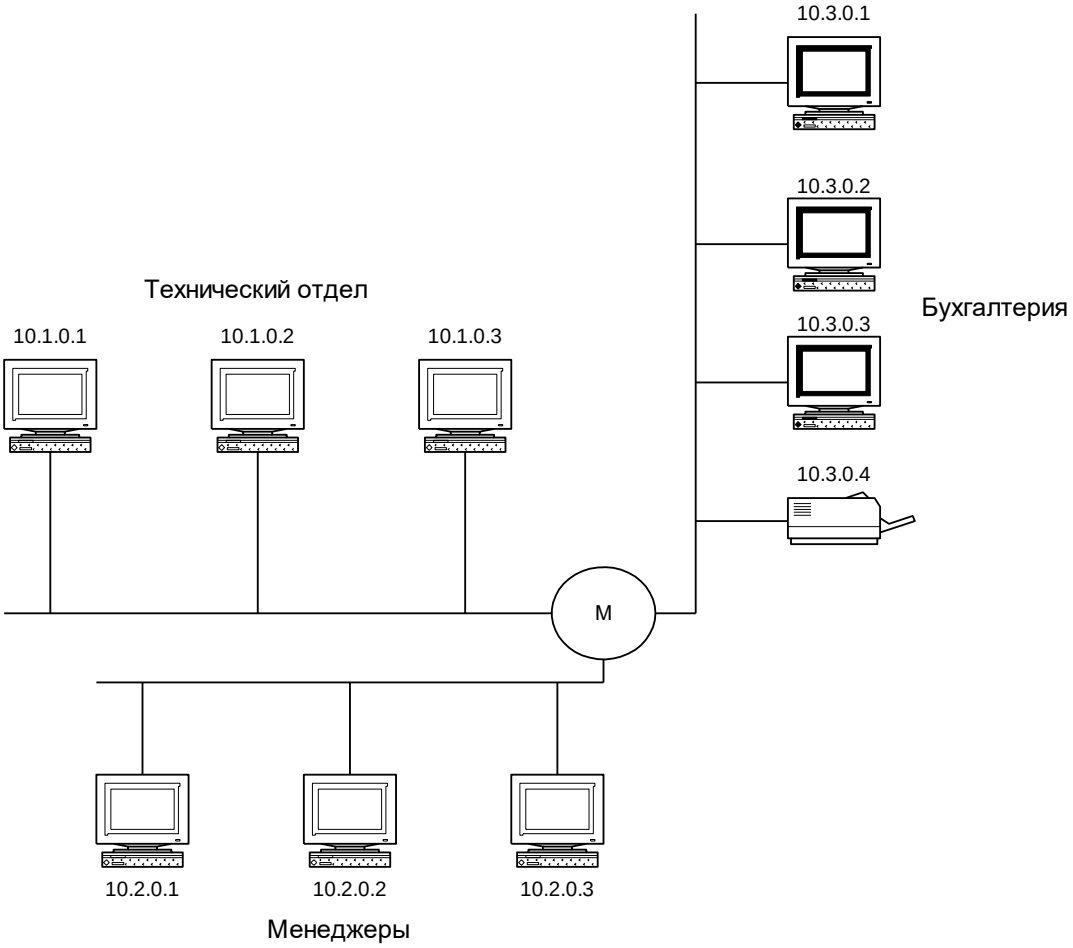


Рис. 2.6. Пример простого варианта маршрутизации

Задачу маршрутизации, то есть определение маршрута, по которому необходимо передать пакет, в зависимости от того, какому узлу он предназначен, решает аппаратное или программное устройство, которое называется маршрутизатором. Маршрутизатор – это устройство (компьютер), которое имеет несколько интерфейсов для подключения сетей. Он выполняет функции объединения подсетей в единую сеть и определение маршрутов передачи пакетов.

Рассмотрим пример. Пусть узел с адресом 10.2.0.1 передает пакет узлу 10.3.0.4. В этом случае непосредственная передача невозможна, поскольку эти узлы относятся к разным сетям 10.2.0.0 и 10.3.0.0. В сети установлен

маршрутизатор М. Каждый узел сети 10.2.0.0 должен быть сконфигурирован таким образом, чтобы пакеты, которые не относятся к данной сети, передавались на маршрутизатор. Получив такой пакет, маршрутизатор должен решить, в какую сеть его отправить и через какой интерфейс. В нашем случае маршрутизатор определяет, что пакет адресован в сеть 10.3.0.0. Это можно легко определить, применив к IP-адресу получателя маску подсети. Следующим шагом будет передача пакета на тот интерфейс, к которому подключена сеть 10.3.0.0. В этой сети пакет будет получен узлом 10.3.0.4.

Рассмотрев этот простой пример, можно сделать следующие выводы:

- каждый узел сети должен содержать информацию о том, куда адресовать пакеты, которые не принадлежат к данной сети (подсети);
- маршрутизатор, для того, чтобы ему можно было передавать пакеты, должен иметь свой IP-адрес. Причем в каждой сети (подсети) этот адрес будет различным;
- маршрутизатор должен содержать информацию обо всех сетях (подсетях), чтобы иметь возможность переслать пакет в соответствующем направлении.

Фактически маршрутизатор действительно представляет собой устройство (компьютер), в котором разрешена пересылка (forwarding) пакетов. Для этого маршрутизатор должен иметь несколько (не менее двух) интерфейсов для подключения к разным сетям и каждому интерфейсу можно назначать свой IP-адрес.

Информация, о которой говорится в первом и третьем выводах, действительно сохраняется на каждом узле сети и на каждом маршрутизаторе в виде таблицы маршрутизации.

Таблица маршрутизации содержит информацию, которой достаточно для определения по IP-адресу пакета, направления его пересылки. В общем случае формат таблицы маршрутизации зависит от типа аппаратного маршрутизатора или от типа ОС, которая используется в качестве программного маршрутизатора. На первом этапе мы будем использовать абстрактный формат таблицы, который подобный реальному. Формат таблицы приведен на рис. 2.7.

Адрес сети	Адрес шлюза	Маска сети	Номер интерфейса

Рис. 2.7. Формат таблицы маршрутизации

Объясним значения полей таблицы:

Адрес сети – адрес сети, для которой осуществлена запись в таблице маршрутизации. Если пакет, который поступает на маршрутизатор, адресован

узлу из данной сети, то направление его передачи будет определяться данной строкой таблицы.

Адрес шлюза – адрес узла, которому необходимо передать пакет, который адресован сети, указанной в первом столбце.

Маска сети – маска, которая применяется к IP-адресам для определения адреса сети для сравнения с адресом в первом столбце.

Номер интерфейса – номер порта (интерфейса), через который пакет выходит из устройства маршрутизации.

Рассмотрим, как в упрощенном виде отобразить алгоритм маршрутизации (рис. 2.8).

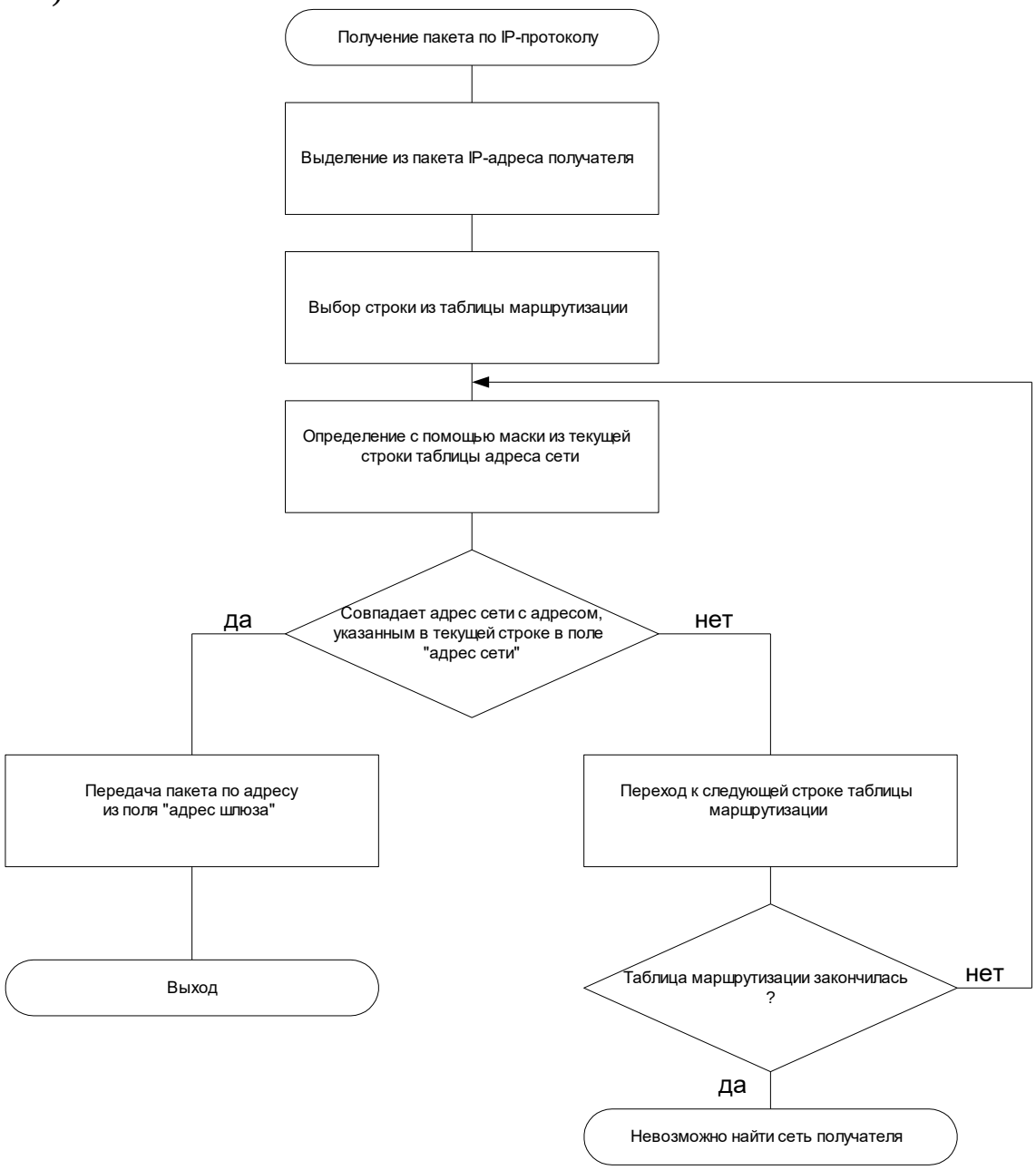


Рис. 2.8. Упрощенный алгоритм маршрутизации

Данный алгоритм отрабатывается узлами, которые выполняют маршрутизацию (компьютеры, аппаратные маршрутизаторы и т.п.).

Как видно из блок-схемы, выбор сети, в которую передается пакет, осуществляется последовательным анализом строк таблицы маршрутизации. Если ни одна строка не позволяет определить адрес сети, то делается вывод, что маршрут не может быть найден, и соответствующее сообщение отправляется узлу, который пытался передать пакет.

Для того чтобы разобраться, как составляется таблица маршрутизации, рассмотрим более сложный пример маршрутизации на рис. 2.9.

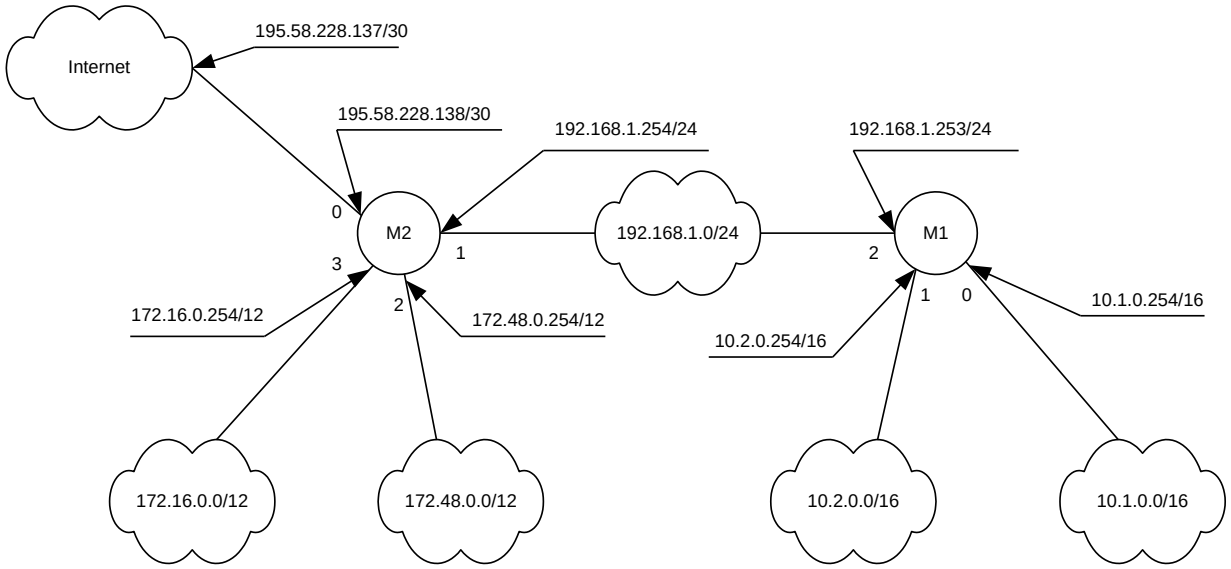


Рис. 2.9. Пример сети с использованием маршрутизаторов

В примере используются два маршрутизатора M1 и M2. Каждый маршрутизатор имеет набор портов (интерфейсов), к которым подключены сети. Следует отметить, что маршрутизатор – это устройство, которое присутствует одновременно в нескольких сетях. То есть он имеет несколько IP-адресов. Один из этих адресов связан с одним из интерфейсов маршрутизатора. Сеть, приведенная на рис. 2.9, подключена к глобальной сети. В нашем случае доступ к глобальной сети осуществляется через маршрутизатор высшего уровня (на рисунке не показанный), который имеет адрес 195.58.228.137/30. Маска сети в данном примере везде задана как количество единиц маски. Интерфейс маршрутизатора M2, через который подключена глобальная сеть, имеет адрес 195.58.228.138/30.

Таблица маршрутизации каждого маршрутизатора должна быть составлена так, чтобы обеспечить маршрут пересылки пакетов из одной произвольной точки сети к другой произвольной точке. Следует отметить, что в случае, если сеть имеет границы (не связана с глобальной), всегда можно перечислить в таблице все возможные маршруты. В случае если существует выход в глобальные сети,

предусмотреть все возможные маршруты невозможно. В этом случае используется понятие „маршрутизатор по умолчанию”. Такой маршрутизатор – это шлюз, на который поступают пакеты, для которых не удалось найти сеть, пересматривая таблицу маршрутизации. Как задавать такой маршрутизатор, мы рассмотрим ниже.

Сразу приведем состав таблицы маршрутизатора М1 в табл. 2.4.

Табл. 2.4. Таблица маршрутизации маршрутизатора М1

№ строки	Адрес сети	Адрес шлюза	Маска сети	Номер интерфейса
1	10.1.0.0	0.0.0.0	255.255.0.0	0
2	10.2.0.0	0.0.0.0	255.255.0.0	1
3	192.168.1.0	0.0.0.0	255.255.255.0	2
4	172.48.0.0	192.168.1.254	255.240.0.0	2
5	172.16.0.0	192.168.1.254	255.240.0.0	2
6	127.0.0.0	0.0.0.0	255.0.0.0	LO
7	0.0.0.0	192.168.1.254	0.0.0.0	2

Первые три строки таблицы описывают сети, которые непосредственно связаны с маршрутизатором М1. В столбце “адрес сети” указан адрес, взятый из рис. 2.9. В столбце “маска сети” указана маска, также взятая из рис. 2.9. В столбце “адрес шлюза” необходимо указать адрес узла, через который будут передаваться пакеты в соответствующую сеть. Вообще существует два варианта: в первом случае пакеты передаются в сеть, которая непосредственно не связана с данным маршрутизатором, тогда в этом столбике указывается адрес соответствующего маршрутизатора. Во втором случае пакеты передаются в сеть, непосредственно связанную с маршрутизатором, тогда указывать адресу шлюза нет необходимости, поскольку пакет будет передаваться не через другой маршрутизатор, а просто через один из интерфейсов данного маршрутизатора. Для такого случая существует специальный адрес шлюза 0.0.0.0 – то есть передача через данный маршрутизатор (через себя самого).

Поскольку первые три сети непосредственно подключены к М1, то в столбце “адреса шлюза” указывается 0.0.0.0. В последнем столбце указывается номер интерфейса, к которому подключена соответствующая сеть.

Четвертая и пятая строка описывают сети, путь к которым пролегает через промежуточный маршрутизатор М2. Поэтому в поле “адрес шлюза” указывается адрес маршрутизатора М2.

Шестая строка описывает путь для пакетов, которые идут к интерфейсу локальной петли (к самому себе). В этом случае указывается соответствующий шлюз 0.0.0.0 и специальный интерфейс LO (local). Седьмая строка описывает так

называемый маршрутизатор по умолчанию. Он используется для пакетов, для которых не обнаружилось сети во всех предшествующих строках.

В случае если сеть не имеет выхода к глобальной сети, в таблицах маршрутизации всегда можно прописать все возможные пути, тогда маршрутизатор по умолчанию не нужен. В нашем же случае сеть имеет выход к глобальной. В этом случае возможные пакеты с любыми адресами назначения. Для таких пакетов мы предусматриваем выход в глобальную сеть. Для нашей сети это можно осуществить через маршрутизатор М2, поэтому указываем его.

По аналогии заполняем таблицу маршрутизатора М2 (табл. 2.5).

Табл. 2.5. Таблица маршрутизации маршрутизатора М2

№ строки	Адрес сети	Адрес шлюза	Маска сети	Номер интерфейса
1	172.16.0.0	0.0.0.0	255.240.0.0	3
2	172.48.0.0	0.0.0.0	255.240.0.0	2
3	192.168.1.0	0.0.0.0	255.255.255.0	1
4	10.1.0.0	192.168.1.253	255.255.0.0	1
5	10.2.0.0	192.168.1.253	255.255.0.0	1
6	127.0.0.0	0.0.0.0	255.0.0.0	LO
7	0.0.0.0	195.58.228.137	0.0.0.0	0

В последней строке таблицы 195.58.228.137 – это адрес шлюза (маршрутизатора), который расположен в глобальной сети.

В реальных сетях возможны значительно более сложные варианты маршрутизации.

Как говорилось ранее, вид таблицы маршрутизации зависит от типа аппаратного или программного маршрутизатора. Ниже приведен пример трех таблиц с одним и тем же содержимым, но отличающихся форматом таблицы и представлением некоторых маршрутов.

Как видно из приведенных таблиц, в некоторых случаях обязательным является указание маршрутов ко всем портам маршрутизатора, отдельно для широковещательных адресов каждой сети и т.д. Номера интерфейсов могут указываться в виде символьных имен, IP-адресов или вовсе отсутствовать (в этом случае маршрутизатор определяет их по косвенным признакам). Дополнительно могут уточняться типы маршрутов (U – маршрут к непосредственно подключенной сети, UG – маршрут к сети через шлюз, UH, UGH – аналогично, но для маршрута к отдельному узлу). В общем случае для каждого маршрута может быть указан целый набор параметров, необходимых для работы различных протоколов маршрутизации.

Табл. 2.6. Таблица программного маршрутизатора под управлением ОС Windows

Network Address	**Netmask**	**Gateway Address**	**Interface**	**Metric**
127.0.0.0	255.0.0.0	127.0.0.1	127.0.0.1	1
0.0.0.0	0.0.0.0	198.21.17.7	198.21.17.5	1
56.0.0.0	255.0.0.0	213.34.12.4	213.34.12.3	15
116.0.0.0	255.0.0.0	213.34.12.4	213.34.12.3	13
129.13.0.0	255.255.0.0	198.21.17.6	198.21.17.5	2
198.21.17.0	255.255.255.0	198.21.17.5	198.21.17.5	1
198.21.17.5	255.255.255.255	127.0.0.1	127.0.0.1	1
198.21.17.255	255.255.255.255	198.21.17.5	198.21.17.5	1
213.34.12.0	255.255.255.0	213.34.12.3	213.34.12.3	1
213.34.12.3	255.255.255.255	127.0.0.1	127.0.0.1	1
213.34.12.255	255.255.255.255	213.34.12.3	213.34.12.3	1
224.0.0.0	224.0.0.0	198.21.17.5	198.21.17.5	1
224.0.0.0	224.0.0.0	213.34.12.3	213.34.12.3	1
255.255.255.255	255.255.255.255	198.21.17.5	198.21.17.5	1

Табл. 2.7. Таблица аппаратного маршрутизатора 3Com

Destination	**Mask**	**Gateway**	**Metric**	**Status**	**TTL**	**Source**
198.21.17.0	255.255.255.0	198.21.17.5	0	Up	–	Connected
213.34.12.0	255.255.255.0	213.34.12.3	0	Up	–	Connected
56.0.0.0	255.0.0.0	213.34.12.4	14	Up	–	Static
116.0.0.0	255.0.0.0	213.34.12.4	12	Up	–	Static
129.13.0.0	255.255.0.0	198.21.17.6	1	Up	160	RIP

Табл. 2.8. Таблица программного маршрутизатора под управлением ОС Linux

Destination	**Gateway**	**Flags**	**Interface**
127.0.0.0	127.0.0.1	UH	Lo
Default	198.21.17.7	UG	eth0
198.21.17.0	198.21.17.5	U	eth0
213.34.12.0	213.34.12.3	U	eth1
129.13.0.0	198.21.17.6	UG	eth0
56.0.0.0	213.34.12.4	UG	eth1
116.0.0.0	213.34.12.4	UG	eth1

В таблицах маршрутизации могут использоваться записи формата CIDR. С такой формой записи тесно связано понятие агрегирования маршрутов. Данная ситуация возникает в том случае, если к одному интерфейсу маршрутизатора подключены несколько сетей одновременно и эти сети имеют соседние адресные

пространства. В этом случае указание возможности пересылки пакетов к любой из этих сетей осуществляется с помощью агрегирования маршрутов. Агрегирование – процесс, обратный процессу разбивки на подсети. Как мы помним, если увеличить длину маски, то появляется возможность выделить подсети. Если же уменьшить длину маски, то можно сразу под одним адресом сети соединить несколько сетей меньшего уровня. Например, существует три сети 10.1.0.0/16, 10.2.0.0/16, 10.3.0.0/16. Если уменьшить маску до $11111111.00000000.00000000.00000000_2$=$255.0.0.0_{10}$, то запись вида 10.0.0.0/8 будет соответствовать любой из указанных подсетей. То есть сеть 10.0.0.0/8 является суперсетью для сетей 10.1.0.0/16, 10.2.0.0/16, 10.3.0.0/16. Фактически это означает, что, уменьшив длину маски, мы получаем сеть большего класса (суперсеть), которая включает в себя заданные подсети.

Следует также отметить, что при определении маршрутов используется правило «самой длинной маски». Это означает, что если при применении разных масок к заданному адресу назначения определяется несколько маршрутов, то выбирается всегда тот, который соответствует маске большей длины.

В таблице маршрутизации может быть несколько записей для одного и того же адреса сети. Это делается в случаях, если надо обеспечить несколько альтернативных путей, то есть продублировать их на случай выхода из строя одного из маршрутизаторов или одной из линий связи. В этом случае возникает проблема поиска лучшего маршрута. Кроме того, рассмотренные примеры маршрутизации отличает одна особенность: все таблицы создаются вручную и остаются неизменными в процессе работы. Такая маршрутизация и такие таблицы называются статическими. Преимущество такой маршрутизации – простота. Главный недостаток – это то, что маршрутизаторы не могут оперативно учитывать изменение структуры сети, исчезновение и появление новых маршрутизаторов и т.п. Для решения вышеуказанных проблем используется динамическая маршрутизация и динамические таблицы маршрутов. Для управления таблицами, передачи информации об изменении структуры сети и др. а также для поиска лучших маршрутов используются алгоритмы RIP (***Routing Information Protocol***), BGP (***Border Gateway Protocol***), OSPF (***Open Shortest Path First***) и др.

С алгоритмами RIP и OSPF можно столкнуться и в локальных, и в глобальных сетях, а алгоритм BGP свойственен именно глобальным сетям. В связи с этим перейдем от рассмотрения маршрутизации в локальных сетях к маршрутизации в глобальных сетях [5].

2.6. Маршрутизация в глобальной сети

Прежде чем рассматривать вопросы маршрутизации в глобальных сетях, кратко рассмотрим историю развития и становления глобальной сети Интернет.

То, что в современном мире мы называем Интернет, первоначально было экспериментальной сетью, созданной в 1969 году для исследовательских целей министерством обороны США. Данная сеть получила название ARPANET (***Advanced Research Project Agency Network***). Первоначально данная сеть объединяла 4 узла с помощью линий связи с пропускной способностью 56 Кбит/с.

Топология, с которой начиналась нынешняя глобальная сеть, приведена на рис. 2.10.

Рис. 2.10. Топология сети ARPANET (1969 г)

Потребность во взаимодействии удаленных друг от друга компьютеров в учебных, научных, коммерческих целях привела к росту масштабов сети ARPANET, и в 1976 году ее топология выглядела как на рис. 2.11. Видно, что сеть представляла собой сложное территориально распределенное образование, сочетающее в себе разнородные линии связи, спутниковые каналы и т.п.

Рост масштабов сети отрицательно сказывался на ее надежности и производительности. Поэтому в начале 1980-х была начата разработка альтернативной сети нового поколения, которая должна была заменить сеть ARPANET. Разработкой нового проекта занималась NSF (***National Scientific Foundation***). Проект получил название NSFNET и был запущен в эксплуатацию в 1986 году. Сеть ARPANET окончательно прекратила свое существование в 1989 году.

Сеть NSFNET представляла собой трехуровневую структуру. На нижнем уровне присутствовали сети образовательных, исследовательских учреждений, предприятий и отдельных пользователей. Данные сети объединялись в региональные узлы (***regional backbone***) по территориальному признаку. Региональные сети подключались к центральному узлу (***core backbone***), объединяющему 6 крупнейших суперкомпьютерных центров США (рис. 2.12). Главные магистральные линии связи сети NSFNET имели пропускную способность 1.544 Мбит/с (T1), а в 1991 были расширены до 45 Мбит/с (T3).

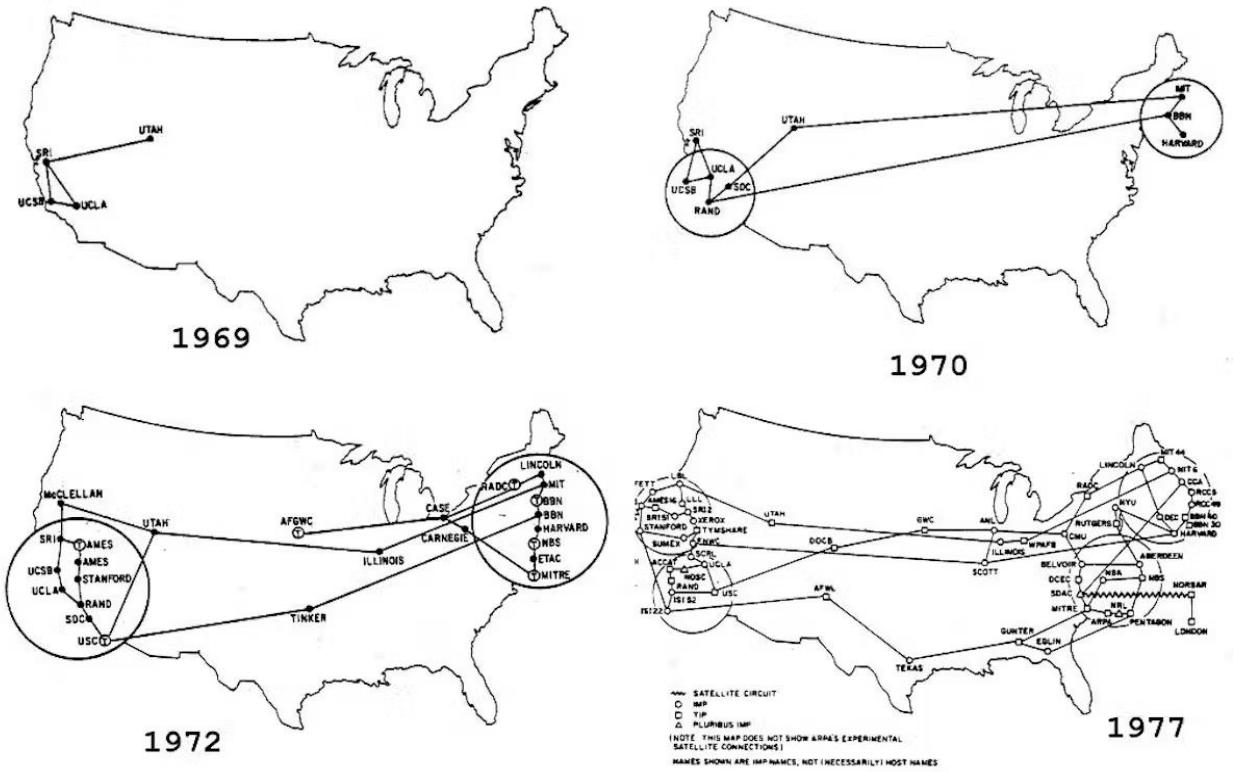


Рис. 2.11. Топология сети ARPANET (1969-1977 гг)

В 1990 г. была образована организация ANS (***Advanced Network and Service***). Данная организация занималась вопросами обслуживания сети, выработки единой политики маршрутизации, создания единой базы данных для решения задач маршрутизации и мониторинга сети.

В середине 1990-х практически исчезла возможность централизованно управлять столь сложной сетевой структурой. В результате началось интенсивное формирование различных коммерческих и правительственных групп по управлению и обслуживанию отдельных сегментов сети. На основе таких групп в дальнейшем появились организации, которые мы теперь называем ISP (Internet Service Provider).

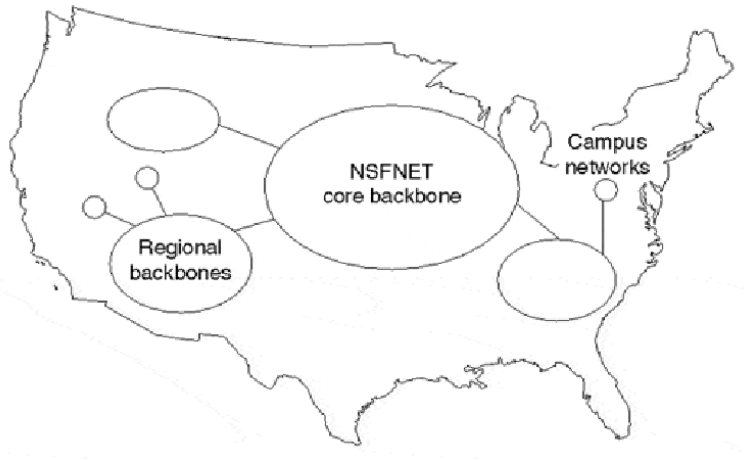


Рис. 2.12. Топология сети NSFNET (1986 г)

Каждая из таких организаций предоставляет доступ к глобальной сети и занимается обслуживанием своего сегмента. Именно в середине 1990-х происходит формирование современной структуры глобальной сети, вырабатываются принципы присоединения и взаимодействия сетей. За пределами США (в Азии, Европе) в это время также начинается структуризация сетевых инфраструктур.

NSFNET как отдельное образование была выведена из эксплуатации в 1995 году. На данный момент структуру глобальной сети (на примере США) можно представить схемой на рис. 2.13.

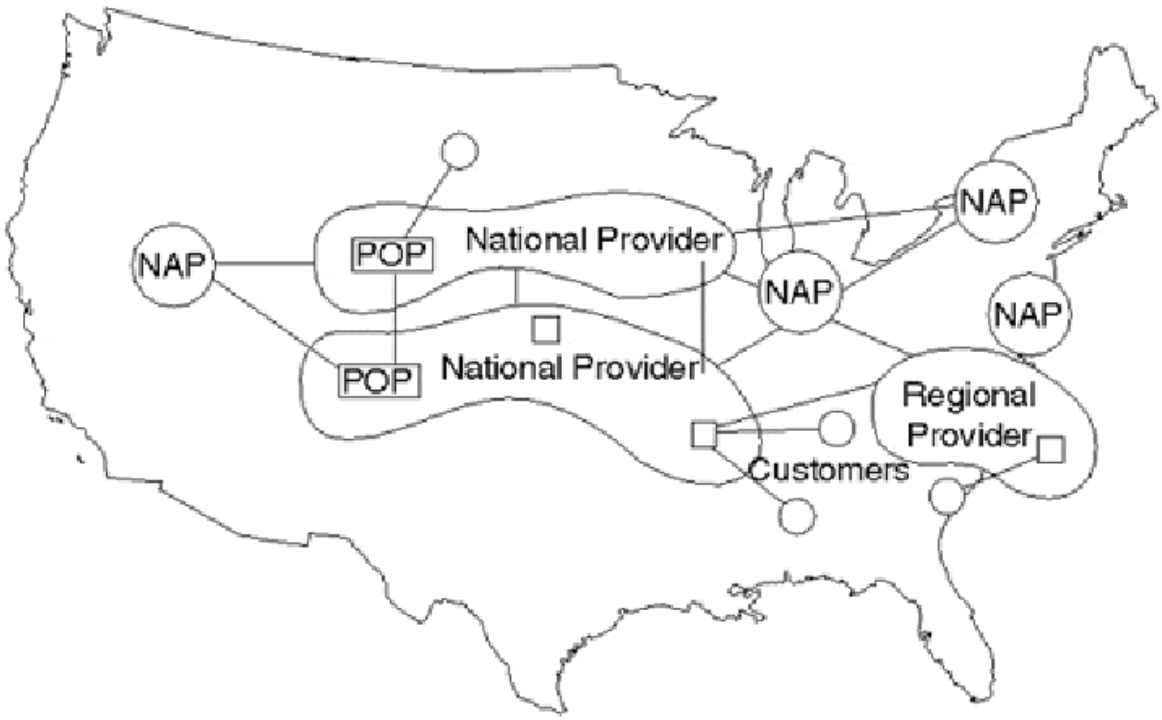


Рис. 2.13. Структура современной глобальной сети

В составе современной глобальной сети можно выделить следующие компоненты:

Customers – организации и частные лица, которым необходим доступ к глобальной сети (пользователи). Часто такие пользователи сами предоставляют доступ другим лицам к глобальной сети, т.е. являются так называемыми вторичными или субпровайдерами.

Provider – организация, предоставляющая доступ пользователям к глобальной сети. Инфраструктура провайдера образуется такими компонентами как POP (***Point Of Presens*** – *точки присутствия*) и коммуникациями, связывающими данные компоненты в единую сеть. Подключение пользователей производится в точках присутствия. Данные точки представляют собой технологические площадки, оснащенные оборудованием для подключения пользователей посредством коммутируемых или выделенных линий. Провайдеры, имеющие точки присутствия на территории определенного региона, называют региональными (***Regional Provider***), а на территории всей страны-национальными (***National Provider***) или в отечественной терминологии – первичными провайдерами.

Для того чтобы пользователи одного провайдера имели доступ к ресурсам других провайдеров, необходимо, чтобы сети провайдеров объединялись в единую сеть. Такое объединение происходит в специальных узлах NAP (***Network Access Point***) – сетевых точках доступа.

Чаще всего такие точки доступа образуются на технической площадке организаций – владельцев глобальных телекоммуникаций, на основе которых строится современная глобальная сеть.

Такие точки доступа часто представляют собой точки обмена трафиком IX (***Internet eXchange***). В этом случае на единой технической площадке размещается оборудование, которое соединяет сервера или маршрутизаторы отдельных провайдеров высокоскоростными линиями связи. Образно можно сказать, что оборудование отдельных провайдеров как бы включается в общую локальную сеть (рис. 2.14).

Как говорилось ранее, каждый узел, представленный в глобальной сети, должен иметь свой собственный уникальный в мировом масштабе IP-адрес. Разделение IP-адресов на классы, рассмотренное ранее, справедливо и в глобальных сетях. Как правило, каждый ISP имеет свой диапазон IP-адресов и при необходимости выдает узлам пользователей адреса из этого диапазона. Ввиду бурного роста масштабов глобальной сети для обеспечения гибкого выделения диапазонов широко используются VLSM (бесклассовые сети).

Наиболее дефицитным в данном случае является класс A, получить диапазон в данном классе пока еще возможно, но процедура его получения крайне сложна и чаще всего оправдана для крупных международных, национальных или региональных провайдеров.

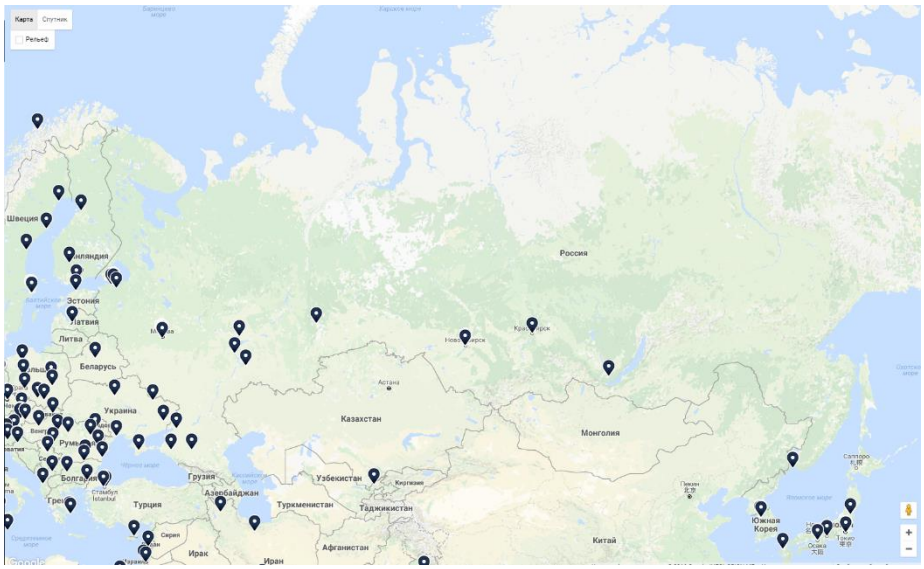


Рис. 2.14. Точки обмена IX в России (крупнейшие)

Вообще, диапазоны глобальных IP-адресов разделены по географическому признаку. Табл. 2.9 дает представление о данном распределении. Распределением диапазонов IP-адресов занимаются такие организации как: American Registry for Internet Numbers (ARIN) – http://www.arin
.net, The Internet Corporation for Assigned Names and Numbers (ICANN) – http:// www.icann.org, RIPE Network Coordination Centre (NCC) – http://www.ripe.net, Internet Assigned Numbers Authority (IANA) – http://www.iana.org.

Табл. 2.9. Распределение адресного пространства по географическому признаку

Адресное пространство	Регион размещения
61.0.0.0 to 61.255.255.255	APNIC—Pacific Rim
62.0.0.0 to 62.255.255.255	RIPE NCC—Europe
63.0.0.0 to 63.255.255.255	ARIN
64.0.0.0 to 64.255.255.255	ARIN
128.0.0.0 to 191.255.255.255	Various Registries
192.0.0.0 to 192.255.255.255	Multiregional
193.0.0.0 to 195.255.255.255	RIPE NCC—Europe
196.0.0.0 to 198.255.255.255	Various registries
199.0.0.0 to 199.255.255.255	ARIN—North America
200.0.0.0 to 200.255.255.255	ARIN—Central and South America
201.0.0.0 to 201.255.255.255	Reserved—Central and South America
202.0.0.0 to 203.255.255.255	APNIC—Pacific Rim
204.0.0.0 to 205.255.255.255	ARIN—North America
206.0.0.0 to 206.255.255.255	ARIN—North America
207.0.0.0 to 207.255.255.255	ARIN—North America

208.0.0.0 to 208.255.255.255	ARIN—North America
209.0.0.0 to 209.255.255.255	ARIN—North America
210.0.0.0 to 210.255.255.255	APNIC—Pacific Rim
211.0.0.0 to 211.255.255.255	APNIC—Pacific Rim
212.0.0.0 to 212.255.255.255	RIPE NCC—Europe
213.0.0.0 to 213.255.255.255	RIPE NCC—Europe
216.0.0.0 to 217.255.255.255	ARIN—North America

Первоначально процедура маршрутизации реализовывалась классическим методом маршрутизации, путем построения и хранения на каждом маршрутизаторе в глобальной сети таблицы маршрутов. Бурный рост масштабов Интернет привел к значительному увеличению количества маршрутов и, соответственно, к значительному росту размеров таблиц маршрутизации. Данный рост оказался настолько значительным, что реализация поиска маршрута стала задачей, практически не решаемой в реальном времени.

Простым способом сокращения размеров таблиц стало агрегирование маршрутов, т. е. применение технологии CIDR. Покажем на примере одной и той же структуры сети с использованием CIDR и без него – на рис. 2.15.

Как видно из рисунка, каждый провайдер должен передавать на следующий уровень информацию обо всех, подключенных к нему сетях. В данном случае – это маршруты к трем и двум сетям соответственно. Провайдер более высокого уровня также должен передавать выше маршруты ко всем сетям, расположенным ниже в иерархии. На рис. 2.15, при подключении к ближайшей NAP в таблице будут присутствовать 5 маршрутов. При использовании же CIDR маршруты можно агрегировать так, как показано на рисунке. В этом случае при подключении к NAP в таблице будет присутствовать всего лишь один маршрут.

Конечно, данный пример достаточно простой, но он отображает общую тенденцию к сокращению размеров таблиц маршрутизации при использовании CIDR. В реальных сетях не всегда удается достигнуть сокращения размеров таблиц маршрутизации только за счет использования агрегирования. Кроме того, для корректного определения маршрутов необходимо учитывать особенности подключения сетей провайдеров друг к другу и к NAP, чтобы избежать ложных маршрутов и отсутствия маршрутов к тем или иным сетям.

Как мы видим, маршрутизация – достаточно сложный процесс, как организационно, так и технически. Если представить себе глобальную сеть, работающую с классическими таблицами маршрутизации, то можно понять, что размеры таких таблиц будут просто гигантские, и совершенно невозможно будет обработать всю таблицу в поисках нужного маршрута за некоторое разумное время.

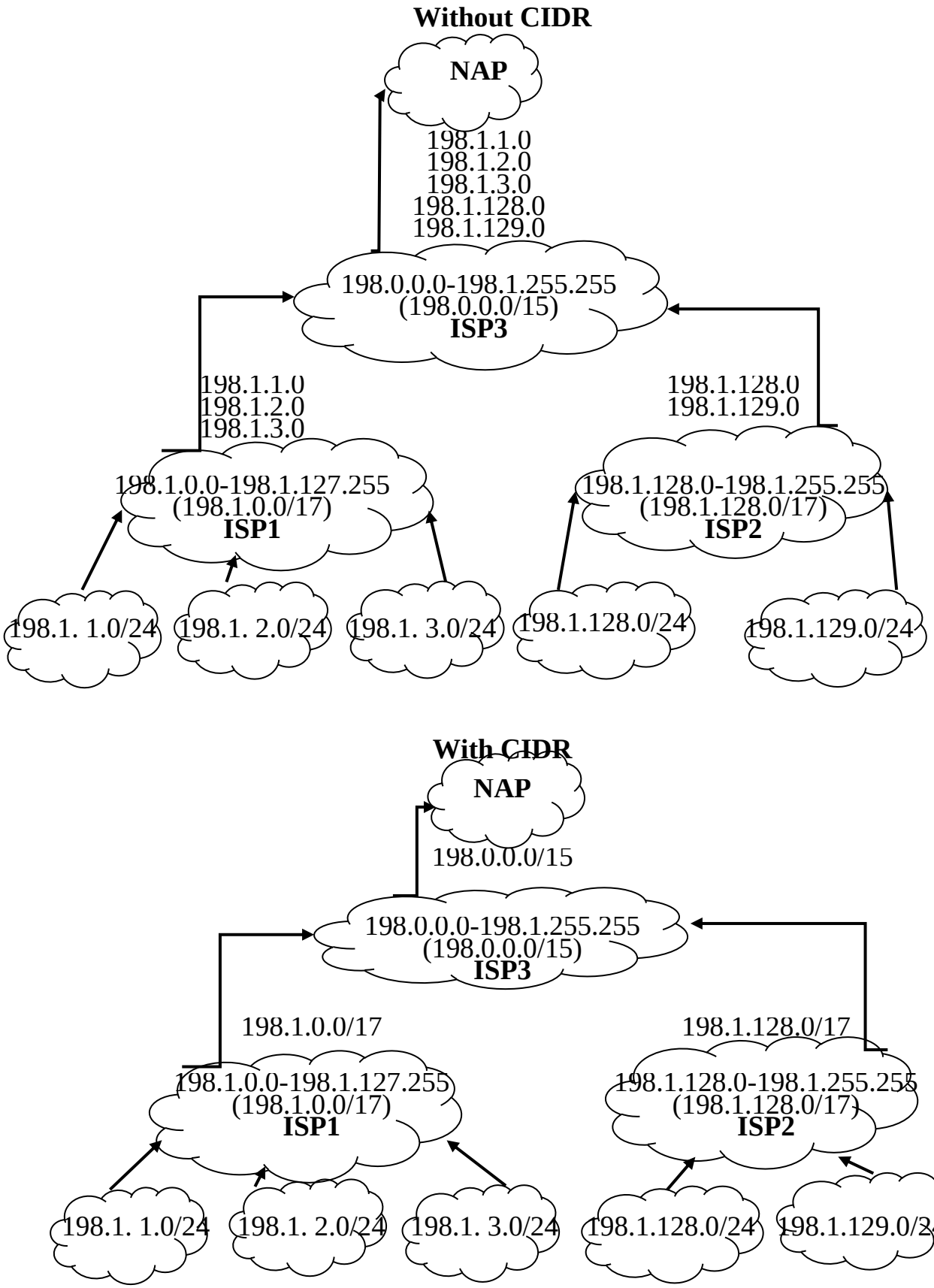


Рис. 2.15. Пример сокращения таблицы маршрутов при использовании CIDR

В связи с этим для упрощения маршрутизации в глобальных сетях и для повышения эффективности алгоритмов маршрутизации глобальную сеть разделили на автономные системы (***AS***). Автономная система – это совокупность сетей, которые находятся под единым администрированием и подчиняются единой политике маршрутизации.

Политика маршрутизации – это набор алгоритмов маршрутизации и правил их применения, действующих в пределах автономной системы. Схематично разделение на автономные системы показано на рис. 2.16.

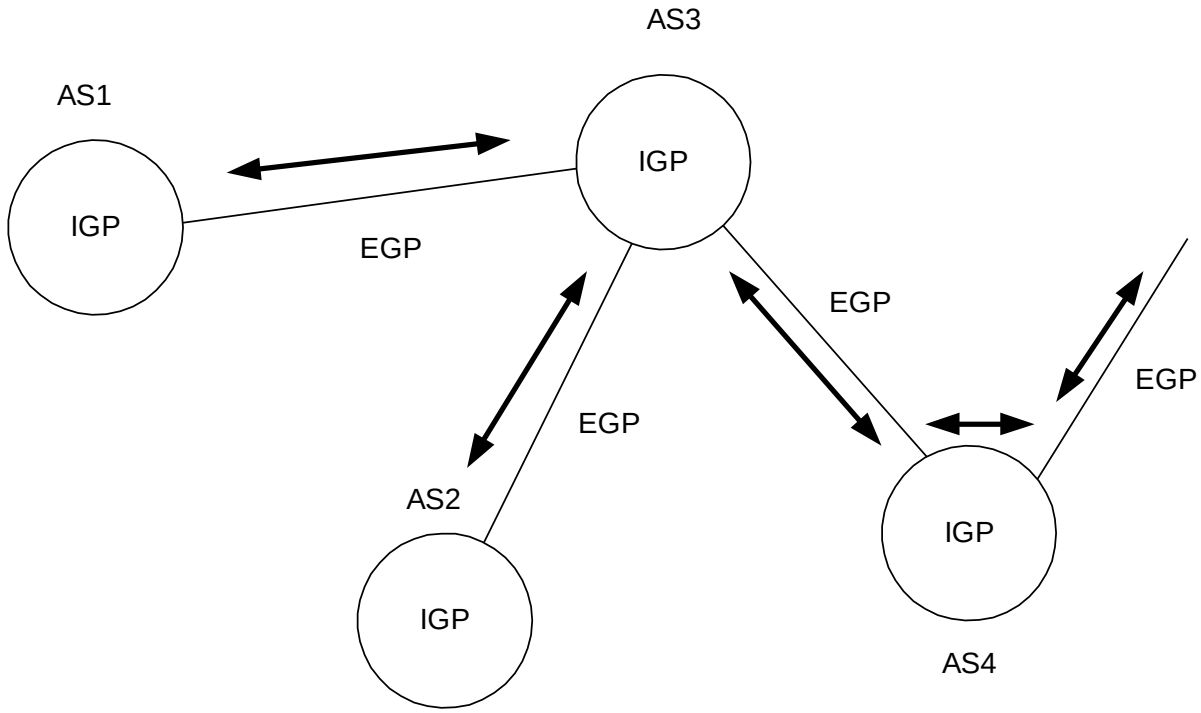


Рис. 2.16. Пример разделения на автономные системы

На рис. 2.16 мы имеем пример, показывающий все разновидности автономных систем.

AS1, AS2 – автономные системы с одной точкой выхода (***single-homed***);

AS3 – нетранзитная автономная система с несколькими точками выхода (***multi-homed nontransit***). Это означает, что система имеет связь с несколькими автономными системами, но не разрешает передачу информации о маршрутах через себя. Т.е. автономная система 3 обменивается информацией о маршрутах с системами 1, 2 и 4, но не пересылает информацию о маршрутах систем 1, 2 или 4 через себя.

AS4 – транзитная автономная система с несколькими точками выхода (***multi-homed transit***). Это означает, что система имеет связь с несколькими автономными системами и разрешает передачу информации о маршрутах через себя. Т.е. автономная система 4 обменивается информацией о маршрутах с

системой 3 и некоторой системой, не показанной на рисунке, и при этом пересылает информацию о маршрутах этих систем через себя.

При делении на автономные системы различают два класса протоколов маршрутизации: IGP (***Interior Gateway Protocol***) – класс протоколов, работающих внутри автономных систем, и EGP (***Exterior Gateway Protocol***) – класс протоколов, работающих между автономными системами.

К классу IGP относятся такие протоколы как RIP, OSPF, IS-IS (***Intermediate System to Intermediate System***), а к классу EGP – протокол BGP. Остановимся на вышеуказанных протоколах подробнее.

RIP – дистантно-векторный алгоритм маршрутизации. То есть он из нескольких альтернативных маршрутов всегда выбирает маршрут минимальной длины. Расстояние по протоколу RIP измеряется в метриках. Метрика – это количество промежуточных узлов (***hops***), которое должен пройти пакет, чтобы достичь сети назначения. В соответствии со стандартом, метрика может иметь значение от 0 до 15. Значение 15 означает бесконечно длинный маршрут. Если по прохождению 15 промежуточных узлов пакет не достигает сети назначения, то он считается потерянным. Метрика записывается в отдельном поле таблицы маршрутизации. Протокол RIP реализует динамическую маршрутизацию. Для этого через определенные промежутки времени маршрутизаторы рассылают своим соседям информацию об известных им маршрутах. В этом случае изменения в структуре сети со временем распространяются по всем маршрутизаторам. Для определения появления в сети новых маршрутизаторов используется специальный метод “определение маршрутизаторов”, который заключается в информировании всех соседей о своем появлении и обмене с ними установочной информацией. При этом маршрутизатор, который включается в сеть, не знает адреса своих соседей и производит рассылку на некоторый зарезервированный IP-адрес, чаще всего 224.0.0.9 (все RIP-маршрутизаторы данной сети).

Протокол RIP имеет такие недостатки:

- интенсивная рассылка таблиц маршрутов, которая при их значительном размере вызывает значительную нагрузку на сеть;
- при использовании сильно разветвленных масштабных сетей использование метрики для определения расстояний становится неэффективным;
- протокол не учитывает физические характеристики линии связи и ее состояние. В результате выбранный маршрут не всегда оказывается лучшим;
- поскольку обновление информации на маршрутизаторах происходит с определенным периодом, то возможны моменты, в которые новые маршруты еще не известны, а устаревшие еще не удалены.

Протокол OSPF относится к классу протоколов состояния связей. Это означает, что выбор маршрута основывается на информации о свойствах отдельных маршрутов (активности, целостности, скорости передачи данных, загруженности и т.д.).

Основные особенности протокола OSPF:

- выбор лучшего маршрута основывается не на условной метрике, а на реальной характеристике маршрута;
- таблицы маршрутизации обновляются только при необходимости;
- разрешена работа с эквивалентными маршрутами с реализацией балансировки трафика;
- поддерживается маршрутизация групповых адресов.

Для сравнения алгоритмов RIP и OSPF обратимся к табл. 2.10.

Табл. 2.10. Сравнение характеристик протоколов RIP и OSPF.

1. Характеристика	RIP	OSPF
Принцип работы	Дистантно-векторный	Состояние связей
Диапазон метрики	0-15 (>15 - бесконечность)	1-65535
Политика обновления информации	Каждые 30 с.	При изменении состояния линии или каждые 30 мин.
Политика рассылки информации	Широковещательная	Групповая
Поддержка эквивалентных маршрутов	Нет	Да
Простота настройки	Да	Нет

При реализации протокола OSPF используется специальная структура сети. Типовой пример конфигурации сети при использовании алгоритма OSPF приведен на рис. 2.17.

В соответствии с протоколом OSPF, который работает только в пределах одной автономной системы, система может разделяться на зоны (area). Зона представляет собой группу сетей, в пределах которой работает своя копия алгоритма OSPF. В автономной системе выбирается одна главная зона (***backbone***), если зона только одна, то она и будет главной. Каждая зона должна иметь свой главный маршрутизатор, через который информация о сетях зоны передается в главную зону. Несколько зон могут соединяться вместе с помощью граничного маршрутизатора зон. Подключение к другим автономным системам происходит с помощью граничного маршрутизатора автономной системы.

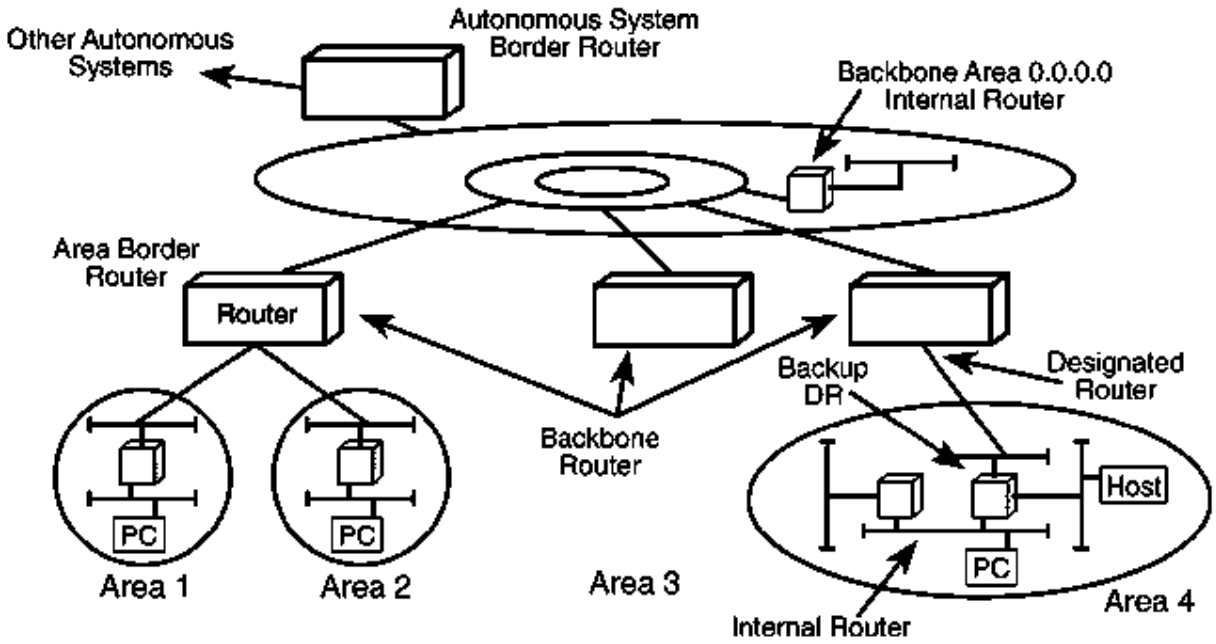


Рис. 2.17. Типовая структура сети при использовании протокола OSPF

Таким образом, на рис. 2.17 можновыделить 6 типов маршрутизаторов:

- ***backbone router (BR)*** – маршрутизатор, который имеет подключение к центральному узлу сети;
- ***area border router (ABR)*** – маршрутизатор, который находится в точке соприкосновения нескольких зон;
- ***autonomus system boundary router (ASBR)*** – маршрутизатор, который находится в точке соприкосновения данной автономной системы с другими автономными системами;
- ***internal router (IR)*** – который имеет подключения только в пределах одной зоны;
- ***designated router (DR)*** – главный маршрутизатор зоны, вся информация из зоны выходит через данный маршрутизатор;
- ***backup designated router (BDR)*** – резервный главный маршрутизатор зоны.

Каждая зона имеет свой идентификатор (***area ID***), который по формату напоминает IP-адрес, но не имеет ничего общего с IP-адресами обслуживаемых сетей. Главная зона всегда имеет идентификатор 0.0.0.0. Назначение идентификаторов остальным зонам определяется администратором и не подчиняется каким-либо особым правилам. Каждый маршрутизатор также имеет свой идентификатор (***router ID***). Идентификатор назначается независимо от зоны, в которой находится маршрутизатор, и имеет тот же формат, что и идентификатор зоны. Чаще всего в качестве идентификатора берут один из IP-адресов маршрутизатора. Идентификатор маршрутизатора должен быть уникальным в пределах автономной системы.

Общие принципы работы протокола OSPF следующие:

- каждый маршрутизатор при включении с помощью специального алгоритма HELLO оповещает своих соседей о своем появлении и собирает информацию о своих интерфейсах и подключенных к ним линиях связи;
- с интервалом в 30 мин. или при изменении состояния какой-либо линии связи, подключенной к маршрутизатору, выполняется формирование пакета данных LSA (***Link State Advertisement***). Такой пакет содержит список маршрутов, известных маршрутизатору, с указанием их метрик;
- пакет LSA отсылается соседним маршрутизаторам, а те, в свою очередь, пересылают их своим соседям. Т.е. происходит рассылка информации по всем маршрутизаторам зоны. За пределы зоны рассылка не выходит;
- каждый маршрутизатор на основании содержимого проходящих через него пакетов LSA строит базу данных;
- после каждого обновления базы данных по алгоритму Дейкстры строится дерево кратчайших путей с использованием метрик;
- полученное дерево используется для формирования таблицы маршрутизации.

Как говорилось ранее, для каждой линии связи определяется метрика. Метрика каждого маршрута – это сумма весов, вносимых каждой линией связи. Среди множества альтернативных маршрутов алгоритм Дейкстры выбирает путь с наименьшей метрикой. Т.е. чем меньше метрика – тем выгоднее данный маршрут для передачи данных. В табл. 2.11. приведены веса наиболее распространенных типов линий связи, рекомендуемые для назначения.

Табл. 2.11. Примеры назначения весов линиям связи

Тип линии связи	Вес
>= 100 Mbps	1
Ethernet/802.3	10
E1 (2.048 Mbps)	48
T1 (1.544 Mbps)	65
64 kbps	1562
56 kbps	1785
19.2 kbps	5208
9.6 kbps	10416

В общем случае назначение весов носит условный характер, но рекомендуется использовать следующую формулу для определения веса:

$$Metric = \frac{10^8}{speed}$$

Speed – скорость интерфейса.

В каждой зоне должен существовать главный маршрутизатор (DR), который накапливает информацию, получаемую от остальных маршрутизаторов зоны в виде LSA-уведомлений. Для повышения надежности в каждой зоне назначается резервный главный маршрутизатор (BDR). Информацию обо всех маршрутах сети главный маршрутизатор пересылает в виде обобщенного LSA-пакета на маршрутизатор главной зоны (0.0.0.0). Обмен информацией с другими зонами идет только через backbone area. В некоторых случаях, для обеспечения непосредственной передачи маршрутов между зонами используются граничные маршрутизаторы зон (ABR).

Для передачи информации о маршрутах в другие автономные системы используются граничные маршрутизаторы автономных систем (ASBR). Их задача – преобразовать информацию о маршрутах, расположенных внутри автономной системы в формат, распознаваемый протоколами маршрутизации класса EGP.

Как видно на рис. 2.17, многие маршрутизаторы выполняют сразу несколько функций, поэтому однозначно отнести маршрутизатор к одному из 6 типов не всегда возможно.

Таким образом, мы видим, что протокол OSPF значительно сложнее протокола RIP и по принципам работы, и по настройке. Данный протокол является результатом многолетней работы многих исследовательских центров и воплощает в себе лучшие идеи, направленные на повышение скорости, надежности и точности при решении задач маршрутизации внутри автономных систем.

Как говорилось ранее, кроме класса IGP, существует класс EGP – протоколы маршрутизации между автономными системами. Как пример такого протокола рассмотрим протокол BGP.

Протокол BGP представляет маршрут к заданной сети как последовательность номеров автономных систем, через который данный маршрут пролегает. Назначение номеров автономных систем и их регистрация, также как централизованное распределение диапазонов IP-адресов выполняется такими организациями, как ARIN, ICANN, RIPE, IANA.

В соответствии с протоколом BGP происходит обмен маршрутной информацией между парой маршрутизаторов, каждый из которых соответствует своей автономной системе. При инициализации маршрутизаторы обмениваются номерами автономных систем, своими конфигурационными параметрами и возможностями. После чего маршрутизаторы обмениваются информацией о маршрутах, которые известны данным маршрутизаторам. Последующие пересылки информации о маршруте выполняются только в случае некоторого события (исчезновение маршрута или появления нового маршрута).

При реализации протокола BGP повсеместно используется CIDR-представление IP-адресов сетей.

Подводя итог всему вышесказанному, следует отметить, что общая идея маршрутизации (посредством таблиц) сохраняется и в глобальных и локальных сетях. При этом в современных сетях повсеместно используется CIDR-представление IP-адресов. При этом определение маршрута передачи, т.е. через какой интерфейс и какой шлюз передавать пакет, сохраняется неизменным во всех видах сетей, меняется то, что подразумевается под шлюзом и интерфейсом), это может быть идентификатор маршрутизатора, идентификатор зоны, номер автономной системы и др.). Протоколы RIP, OSPF, BGP и им подобные решают задачу наполнения таблиц соответствующими записями для всех известных маршрутов в масштабах автономных систем, зон, отдельных сетей и др., т.е. фактически превращают глобальную сеть в связанную систему, со структурированной системой адресации и однозначным определением маршрута передачи данных [6].

2.7. Формат пакета в соответствии с протоколом IP

Рассмотрев то, как решаются задачи адресации и определения маршрутов на сетевом уровне, рассмотрим теперь, какой формат имеют пакеты, передаваемые по протоколу IP. Структура пакета приведена на рис. 2.18.

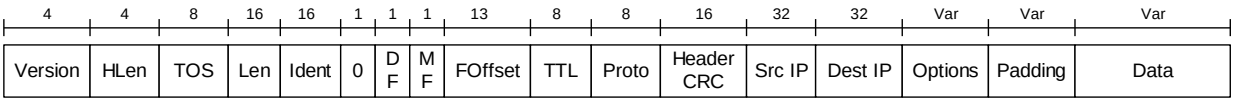


Рис. 2.18. Формат пакета протокола IP

Рассмотрим подробно поля пакета.

Version – номер версии IP-протокола;

HLen (header length) – длина заголовка в 32-разрядных блоках (без учета поля данных);

TOS (Type of Service) – поле, определяющее параметры качества обслуживания (QoS) данного пакета;

Len – длина пакета в байтах;

Ident – идентификатор фрагмента;

0 – нулевой бит;

DF (Disable fragmentation) -бит запрета фрагментации (0 – разрешена, 1 – запрещена);

MF (More fragment) – бит последнего фрагмента (0 – больше нет фрагментов, *1* – еще есть фрагментов последовательности);

FOffset (Fragment Offset) – смещение фрагмента от начала пакета;

TTL (Time To Live) – время жизни пакета, измеряется в количестве переходов через маршрутизатор. При каждом прохождении маршрутизатора поле уменьшается на 1. При достижении нулевого значения пакет уничтожается, что

предотвращает бесконечное существование в сети пакетов, которые не могут быть доставлены получателю.

Proto – числовой идентификатор, который определяет тип протокола, который используется для представления данных на более высоком уровне (т.е. упакован в данный IP-пакет).

Header CRC – контрольная сумма заголовка;

Src IP (Source IP) – IP-адрес источника;

Dest IP (Destination IP) – IP-адрес приемника;

Options – Необязательное поле опций.

Padding – Заполнение

Data – поле данных, содержит данные более высокого уровня.

Рассмотрим некоторые поля более подробно.

Поля Ident, DF, MF, FOffset предназначены для решения задачи фрагментации. Данная задача требуется для согласования параметров сетей, по которым проходят пакеты. Т.е. это третья задача протоколов сетевого уровня.

Задача фрагментации возникает по следующей причине. От узла-источника к узлу-приемнику пакеты могут проходить огромным количеством различных сетей, построенных с использованием различных технологий. На низшем, чем сетевой, канальном уровне любая сеть имеет такой параметр, как максимальный размер передаваемого блока данных (кадра). Такой параметр называется MTU (***Maximum Transfer Unit***). Процесс фрагментации рассмотрим на примере. Пусть пакет проходит через три сети, как показано на рис. 2.19.

Пусть из узла А в узел В передается IP-пакет размером 1500 байт. Узел А содержит информацию о значении MTU сетей, к которым он подключен. Поэтому фрагментация при необходимости будет выполнена на уровнях более высоких, чем сетевой, и на сетевом уровне все пакеты заведомо будут

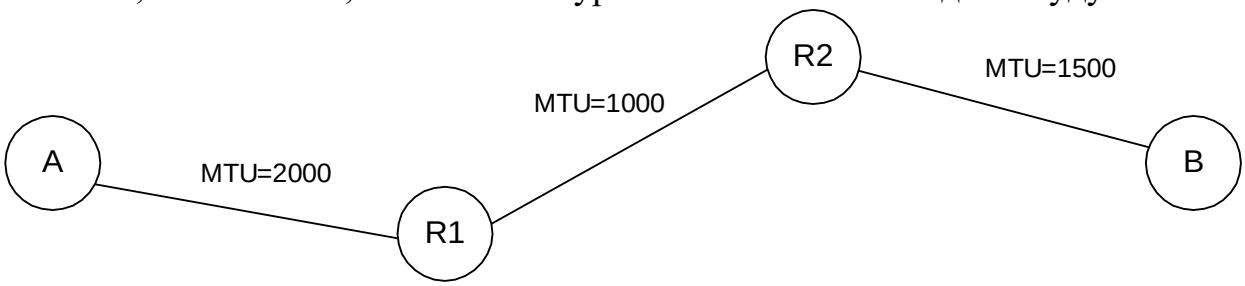


Рис. 2.19. Пример фрагментации на сетевом уровне

удовлетворять заданному MTU. В нашем случае так и есть, на сетевом уровне пакет имеет размер 1500 байт, что заведомо меньше MTU подключенной сети (2000).

На промежуточных узлах (чаще всего маршрутизаторах) пакеты распаковываются максимум до сетевого уровня, поэтому необходимо иметь механизм фрагментации в составе протокола IP.

В нашем случае маршрутизатор R1 принимает пакет и пытается передать его в следующую сеть. Однако значение MTU следующей сети 1000, тогда как наш пакет имеет размер 1500, т.е. предать его непосредственно невозможно. В этом случае запускается алгоритм фрагментации.

1. Анализируется бит DF исходного пакета. Если бит равен 1, то фрагментация запрещена и узлу отправителю передается сообщение об ошибке. Если бит DF равен 0, то осуществляется фрагментация.
2. Поле данных исходного пакета разбивается на части таким образом, чтобы каждая часть в сумме с длиной заголовка укладывались в параметр MTU.
3. Из одного исходного IP-пакета создается столько пакетов, на сколько частей разбивалось поле данных на шаге 2.
4. Практически все поля заголовка исходного пакета копируются в заголовки вновь созданных пакетов. Исключение составляют поля HLen, Len, Header CRC, которые пересчитываются для каждого фрагмента.
5. Для каждого фрагмента устанавливаются параметры фрагментации. Поле DF определяет, будет ли разрешена дальнейшая фрагментация при необходимости. Поле MF устанавливается во всех фрагментах, кроме последнего. Поле FOffsret определяет смещение фрагмента от начала исходного пакета (см. рис. 2.20).
6. Полученная последовательность фрагментов, оформленных как полноценные IP-пакеты, передается в следующую сеть.

Предположим, что в нашем случае поле данных разбилось на фрагменты размером 800 и 700 байт. В этом случае для схемы, показанной на рис. 2.19 маршрутизатор R2 получит эти фрагменты и без изменений отправит их в следующую сеть, т.к. параметр MTU данной сети (1000) превышает размер фрагментов. Далее узел B получит фрагменты и выполнит восстановление пакета.

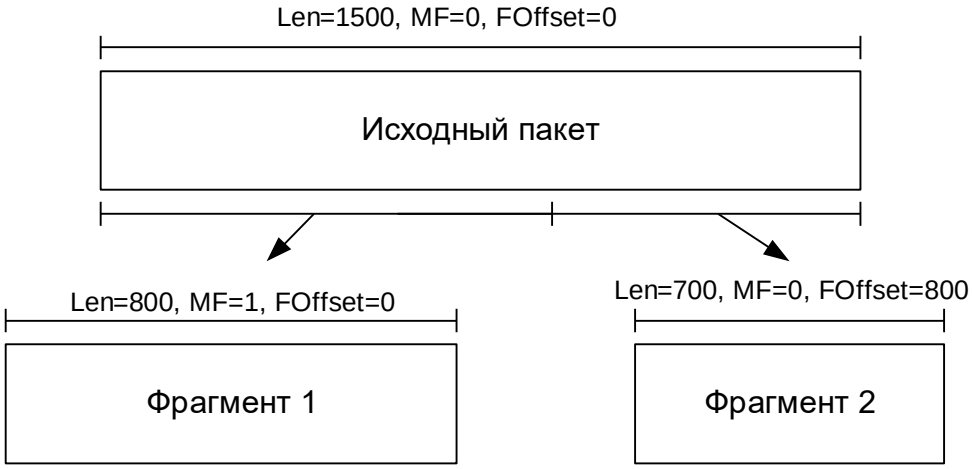


Рис. 2.20. Установка параметров фрагментов

Вернемся к вопросу фрагментации. На рис. 2.20 показано, как поле данных делится на части, и как устанавливаются поля MF и FOffset для каждой части.

Каждый из полученных фрагментов также может быть разбит на части неограниченное число раз.

Рассмотрим теперь, как восстанавливается исходный пакет на узле-получателе.

1. При получении какого-либо из фрагментов исходного пакета выделяется блок памяти, в котором фрагмент располагается в соответствии со значением поля FOffset, при этом запускается специальный таймер.
2. Каждый последующий пакет проверяется на то, не является ли он фрагментом интересующего нас пакета. Принадлежность фрагментов одному пакету определяется по уникальному, одинаковому для всех фрагментов значению поля Ident. Данное поле вычисляется на основании IP-адресов источника и приемника.
3. Все подходящие фрагменты, добавляются в буфер в соответствии со значением поля FOffset до тех пор, пока не будет получен фрагмент с признаком MF=0, и не будут заполнены все пропуски в буфере. После этого пакет считается восстановленным в первоначальном виде.

Если до восстановления пакета таймер досчитает до нуля, то пакет считается непринятым, и узлу-отправителю посылается сообщение об ошибке.

Следует отметить, что фрагменты одного пакета могут добираться в результате до узла-получателя разными маршрутами.

Рассмотрим некоторые оставшиеся поля.

Proto – номер протокола, который упакован в составе IP-пакета. Данные номера назначаются стандартным протоколам организацией IANA. Пример нумерации протоколов приведен в табл. 2.12.

Табл. 2.12. Нумерация стандартных протоколов

Номер	Протокол	Расшифровка
1	ICMP	Internet Control Message Protocol
2	IGMP	Internet Group Management Protocol
6	TCP	Transmission Control Protocol
8	EGP	Exterior Gateway Protocol
9	IGRP	Private Interior Gateway Protocol
17	UDP	User Datagram Protocol
46	RSVP	Reservation Protocol
50	ESP	Encapsulating Security Payload

51	AH	Authentication Header
88	EIGRP	Cisco's Enhanced Interior Gateway Routing Protocol
89	OSPF	Open Shortest Path First

Header CRC – контрольная сумма заголовка, предназначенная для проверки его корректности при передаче. Контрольная сумма CRC– это параметр, который вычисляется на основе содержимого пакета и используется для контроля правильности передачи. Вычисление может производиться как суммирование по некоторому модулю или как вычисление кода с коррекцией ошибок (коды Хэмминга и т.п.). Проверка правильности передачи происходит в соответствии со схемой на рис. 2.21.

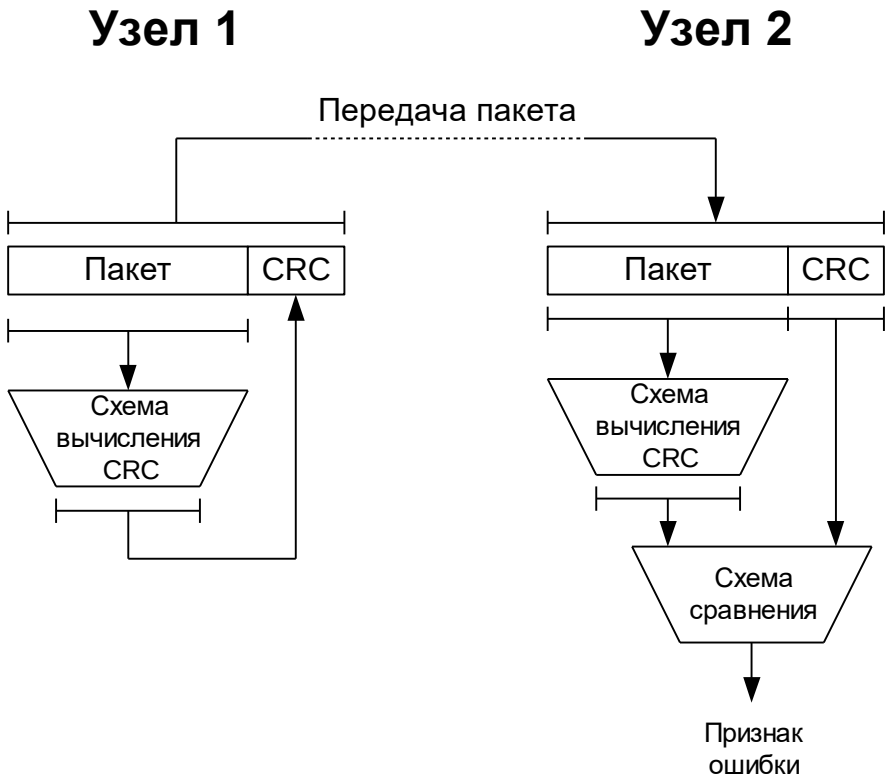


Рис. 2.21. Принцип проверки правильности передачи с помощью CRC

На узле-отправителе происходит вычисление контрольного кода по содержимому пакета, после чего код приформировывается к пакету и передается вместе с ним на узел-получатель. На узле получателе контрольный код вычисляется заново по содержимому полученного пакета, после чего сравнивается с кодом, полученным вместе с пакетом. Если коды совпадают, то считается, что пакет передан без ошибок. Если коды не совпадают, то при передаче произошла ошибка(и). В этом случае ошибка(и) исправляется, если это возможно, а если нет, то об ошибке уведомляется программное обеспечение и

принимается соответствующее решение (аварийное завершение программы, запрос на повторную передачу, игнорирование ошибки и т.д.)

Рассмотренное понятие «контрольная сумма» будет в дальнейшем неоднократно встречаться на различных уровнях модели OSI.

Поле Options при наличии сохраняет дополнительные параметры. Наличие данного поля можно легко определить, сравнив длину заголовка с минимально возможной (5 32-разрядных блоков). В данное поле могут записываться различные данные:

- временные метки;
- адреса промежуточных узлов;
- отладочная и диагностическая информация;
- и т.д.

Тип записанной информации и ее объем определяется по полям, на которые делится параметр Options в соответствии со стандартом.

Поле Padding представляет собой блок нулей, необходимый для дополнения размера заголовка до границы 32-разрядного блока при использовании параметров Options.

2.8. Протокол ICMP

Протокол ICMP (***Internet Control Message Protocol***) является вспомогательным протоколом сетевого уровня. При передаче данных по этому протоколу пакет упаковывается внутрь обычного IP-пакета так, как если бы он соответствовал более высокому уровню.

Формат пакета при использовании данного протокола представлен на рис. 2.22.

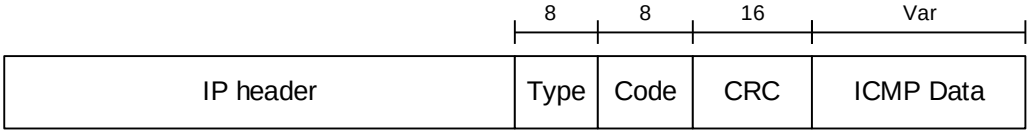


Рис. 2.22. Формат ICMP-пакета

Рассмотрим поля подробнее:

Type –тип управляющего сообщения;

Code – код, уточняющий смысл управляющего сообщения;

CRC – контрольная сумма;

ICMP Data – данные сообщения (размер и вид зависят от полей Type и Code)

Фактически сообщения ICMP используются для уведомления отправителя об ошибках (невозможности определить маршрут, об исчерпании промежутка TTL и т.д.), для диагностических целей (проверка доступности узла, трассировка маршрута и т.д.). В табл. 2.13 приведены наиболее распространенные типы управляющих сообщений.

Табл. 2.13. Типы управляющих сообщений

Код типа	Назначение	Расшифровка
0:	Echo reply	Ответ на запрос о доступности узла
3:	Destination unreachable	Невозможно распознать узел-получатель
4:	Source quench	Недостаточно места для приема пакета
5:	Redirect	Перенаправление на другой маршрутизатор
8:	Echo request	Запрос доступности узла
9:	Router advertisement	Обмен информацией между маршрутизаторами
10:	Router solicitation	
11:	Time exceeded	Промежуток TTL истек
12:	Parameter problem	Ошибка в параметрах пакета
13:	Timestamp request	Запрос временных меток
14:	Timestamp reply	Ответ на запрос временных меток
15:	Information request	Запрос информации
16:	Information reply	Ответ на запрос информации
17:	Address mask request	Запрос маски сети
18:	Address mask reply	Ответ на запрос маски сети
30:	Traceroute	Трассировка маршрута
31:	Datagram conversion error	Ошибка преобразования пакета
32:	Mobile host redirect	Перенаправление мобильного узла
33:	IPv6 Where-Are-You	Специфичны для IPv6
34:	IPv6 I-Am-Here	
35:	Mobile registration request	Запрос на регистрацию в IP-сети мобильного узла
36:	Mobile registration reply	Регистрация мобильного узла в IP-сети
37:	Domain name request	Запрос доменного имени
38:	Domain name reply	Ответ на запрос доменного имени

Для многих типов уточнение смысла сообщения (например, для Destination unreachable) производится полем Code.

2.9. Протокол IGMP

Протокол IGMP (***Internet Group Management Protocol***) также является вспомогательным протоколом сетевого уровня, а фактически – расширением ICMP при использовании групповой адресации. Данные этого протокола также упаковываются внутрь обычного IP-пакета.

Формат пакета при использовании данного протокола представлен на рис. 2.23.

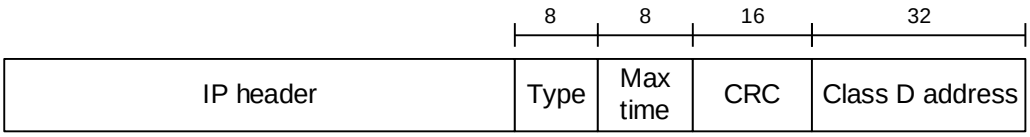


Рис. 2.23. Формат IGMP-пакета

Рассмотрим поля подробнее:

Type – тип группового сообщения;

Max time – максимальное время, в течение которого отправитель ожидает ответа на свой запрос;

CRC – контрольная сумма;

Class D address – групповой адрес в соответствии с правилами класса D, на который должен отправляться ответ на запрос.

Фактически данные сообщения используются узлами или маршрутизаторами для определения наличия в данной сети групп и их адресов или принадлежности узлов тем или иным группам.

Подробное рассмотрение вопросов групповой адресации выходит за рамки данного курса.

Вопросы для самопроверки

1. Описать процесс маршрутизации сетей
2. Перечислить некоторые стандартные протоколы
3. Описать процедуру фрагментации пакетов
4. Раскрыть принцип проверки правильности передачи с помощью CRC
5. Описать формат ICMP-пакета

3. ПРОТОКОЛЫ ТРАНСПОРТНОГО УРОВНЯ

В составе стека протоколов TCP/IP функции транспортного и сеансового уровня выполняются одной группой протоколов. Вспомним функции данных уровней:

- установка логического канала между узлами и синхронизация процессов;
- обеспечение надежной и гарантированной доставки пакетов;
- управление потоком данных, передаваемых через сеть.

Протоколы TCP и UDP – это протоколы стека TCP/IP, которые выполняют задачи транспортного и сеансового уровня. Их назначение – надежная передача пакетов от одного узла к другому с синхронизацией передачи-приема.

3.1. Протокол UDP (User Datagram Protocol)

Данный протокол является наиболее простым, он обеспечивает пересылку данных в том виде, в каком этого требует прикладной уровень. То есть количество и размер пакетов с информацией, переданных из прикладного уровня на узле-отправителе, будет совпадать с количеством и размером на узле-получателе. Это означает, что протокол передает данные “как есть”. При этом протокол UDP осуществляет передачу данных без установления логического канала. Это означает, что отправитель в любой момент может начать передачу без проверки готовности получателя. При передаче любого пакета отправитель не получает информацию о том, был ли действительно доставлен пакет или нет [8].

Можно сделать вывод, что данный протокол имеет такие преимущества:

- простой формат и алгоритм обработки;
- обеспечивает более скоростную связь;

и такие недостатки:

- низкую надежность передачи;
- осуществляет передачу без учета особенности сети (загрузка и т.п.);
- требует от прикладного уровня дополнительных мер по обеспечению надежности передачи.

Формат пакета протокола UDP на транспортном уровне имеет вид, представленный на рис. 3.1.

В варианте (а) перед пакетом идет так называемый псевдозаголовок. Как видно, информация в этом заголовке дублирует информацию из заголовка IP-протокола, поэтому чаще всего используется вариант (б).

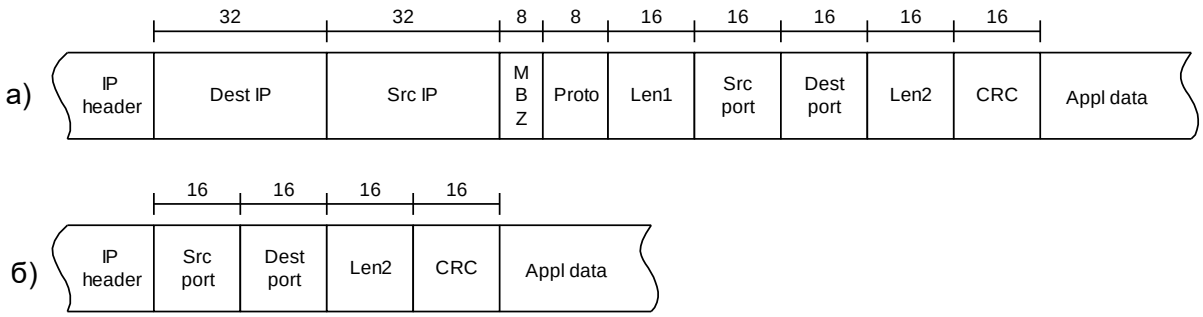


Рис. 3.1. Формат пакета UDP а) с использованием псевдозаголовка, б) без использования псевдозаголовка.

Рассмотрим назначения полей пакета:
Dest IP – IP-адрес получателя;
Src IP – IP-адрес отправителя;
MBZ(Must be zero) – поле, которое всегда содержит 0;
Proto – номер протокола UDP (17);
Len1 – длина пакета с учетом псевдозаголовка;
Src port –порт отправителя;
Dest port – порт получателя;
Len2 – длина пакета с учетом заголовка UDP;
CRC – контрольная сумма;
Appl data – пакет более высокого уровня модели OSI.

На верхних уровнях модели OSI, включая транспортный, широко используется понятие “порт”. Порт – это числовой идентификатор, который определяет, какая прикладная программа будет обрабатывать данный пакет. Фактически в системе каждой стандартной службе: почта, web и др. присвоен свой номер. Диапазон портов 0-65535. Обычно этот диапазон разбивается так, как представлено в табл. 3.1 (назначение наиболее распространенных стандартных портов приведено в приложении 2).

Табл. 3.1. Распределение портов

Диапазон	Назначение
0-1023	Специализированные стандартные порты
1024-65535	Порты, которые могут использоваться разными службами (пользовательские)

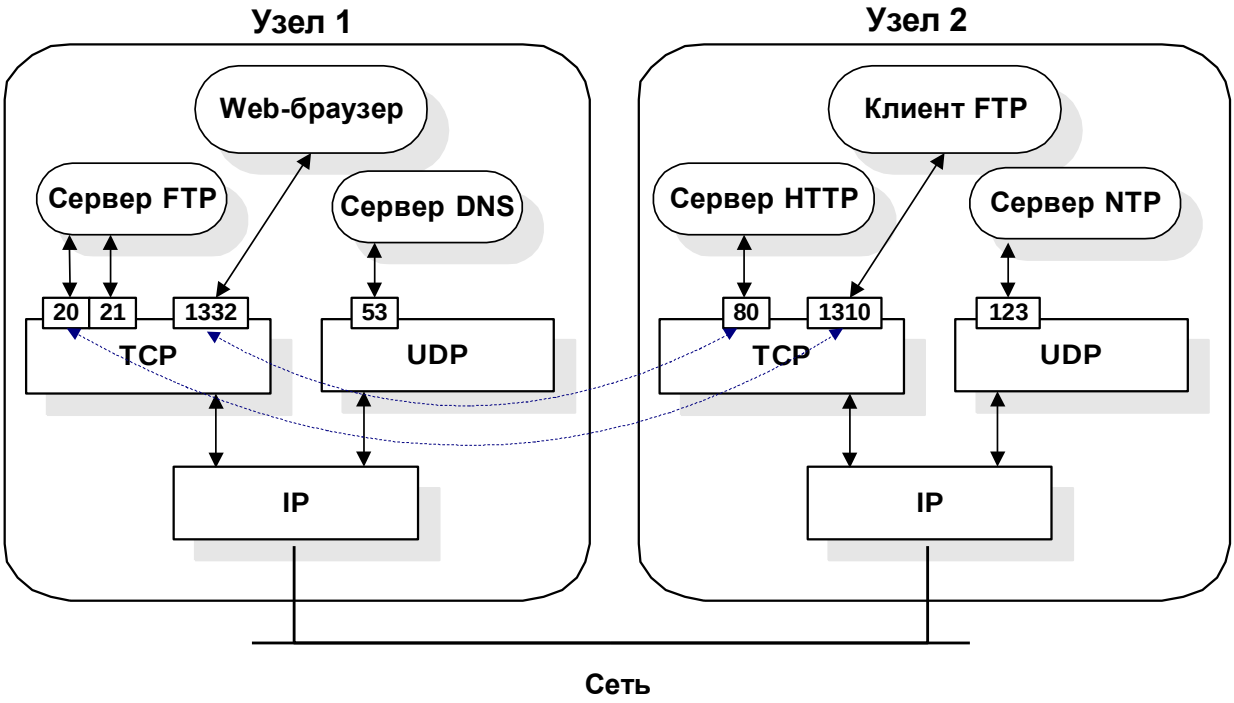


Рис. 3.2. Взаимодействие прикладных программ посредством портов

На рис. 3.2 показан пример взаимодействия прикладных программ с использованием понятия «порт».

Пользователь на узле 1 желает установить соединение с web-сервером на узле 2. Для этого он запускает браузер, который пытается установить соединение с web-сервером. Для этого броузер открывает один из пользовательских портов (1332) и устанавливает связь между данным портом узла 1 и стандартным портом web-сервера (80) на узле 2. Это будет означать, что все пакеты, которые придут на узел 2 с указанием порта-получателя 80, будут обрабатываться web-сервером, а все пакеты, которые придут на узел 1 с указанием порта-получателя 1332, будут обрабатываться web-броузером. Аналогично на рис. 3.2 показано установление соединения ftp-клиента с порта 1310 с ftp-сервером с указанием стандартных портов протокола FTP(20, 21). Также показана связь стандартных служб DNS и NTP с соответствующими портами 53 и 123. Указанные выше протоколы, будут подробно рассмотрены при изучении прикладного уровня модели OSI.

3.2. Протокол TCP (Transfer Control Protocol)

Данный протокол является значительно более сложным, чем UDP, но при этом он обеспечивает высокую надежность передачи данных, выполняет их фрагментацию,осуществляет синхронизированный обмен с установлением логического канала между узлами.

Формат TCP-пакета изображен на рис. 3.3.

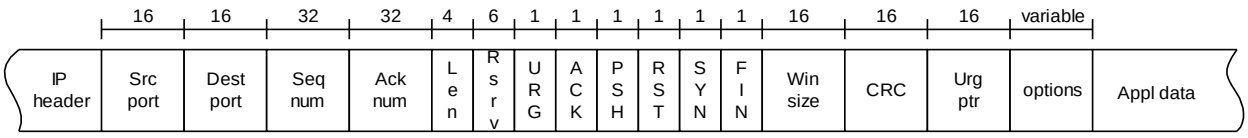


Рис. 3.3. Формат TCP-пакета

Рассмотрим поля пакета:

Src port – порт отправителя;

Dest port – порт получателя;

Seq num – номер первого байта пакета в потоке данных;

Acknum – следующий номер в потоке, который ожидает приемник;

Len – длина заголовка TCP в 32-разрядных блоках;

Rsrv – резервное поле;

URG – если бит установлен, то в поле Urg ptr присутствуют данные;

ACK – бит подтверждения, означает, что в поле Ack num – номер подтверждения;

PSH – бит проталкивания, если установлен, то пакет немедленно передается прикладному уровню;

RST – бит сбрасывания, если установлен, то соединение разрывается аварийно;

SYN – бит синхронизации, устанавливается в пакетах, которые участвуют в установлении или восстановлении соединения;

FIN – бит окончания соединения;

Win size – размер окна (указывается в пакетах с установленным битом ACK), означает количество байтов, начиная с номера, указанного в поле Ack num, которое получатель в состоянии обработать;

CRC – контрольная сумма;

Urg ptr – указатель, который указывает на первый байт порциисрочных данных;

options – специальным образом структурированное поле параметров;

Appl data – данные прикладного уровня.

Рассмотрим подробно поле options. Данное поле структурно разделяется на три части: тип параметра, длина, содержимое. В табл. 3.2 представлены все типы параметров, которые могут присутствовать в поле options.

Табл. 3.2 Типы параметров TCP-пакета

Тип	Длина	Значение
0	-	Конец списка параметров
1	-	Отсутствие операции
2	4	Максимальный размер сегмента, передается при

		установлении соединения
4	2	Sack-Permitted (разрешение режима выборочного подтверждения)
5	X	SACK (пакет, содержащий выборочное подтверждение)
8	10	Timestamps

Рассмотрим шаг за шагом, как происходит обмен данными по протоколу TCP.

Установление соединения

Для установления соединения используется механизм трехшагового квитирования. Отправитель, который желает установить соединение, отправляет пакет с установленным битом SYN и некоторым начальным номером последовательности A (начальный номер выбирается произвольным образом). После отправки отправитель запускает таймер и ожидает определенное время получения подтверждения, если за некоторый период времени подтверждение не приходит, пакет отправляется повторно и так столько раз, сколько задано параметрами системы.

Получатель получает пакет SYN и отправляет пакет-подтверждение. В этом пакете установлены биты SYN и ACK. Пол Ack num содержит номер последовательности A, увеличенный на единицу. Этим получатель сообщает, что он ждет следующий A+1 пакет. В поле Seq num находится начальный номер последовательности B, выбранный произвольным образом. То есть существует две нумерации A, A+1, ... для направления отправитель-получатель и B, B+1,... – для направления получатель-отправитель. После отправки получатель запускает таймер и ожидает определенное время на подтверждение, если за некоторый период времени подтверждение не приходит, пакет отправляется повторно и так столько раз, сколько задано параметрами системы.

Отправитель, получив пакет SYN с битом ACK, переходит в состояние установленного соединения и отправляет получателю подтверждение, т.е. пакет ACK с полем Ack num = B+1. После получения такого пакета получатель также переходит в состояние установленного соединения.

Данный процесс отображен на рис. 3.4.

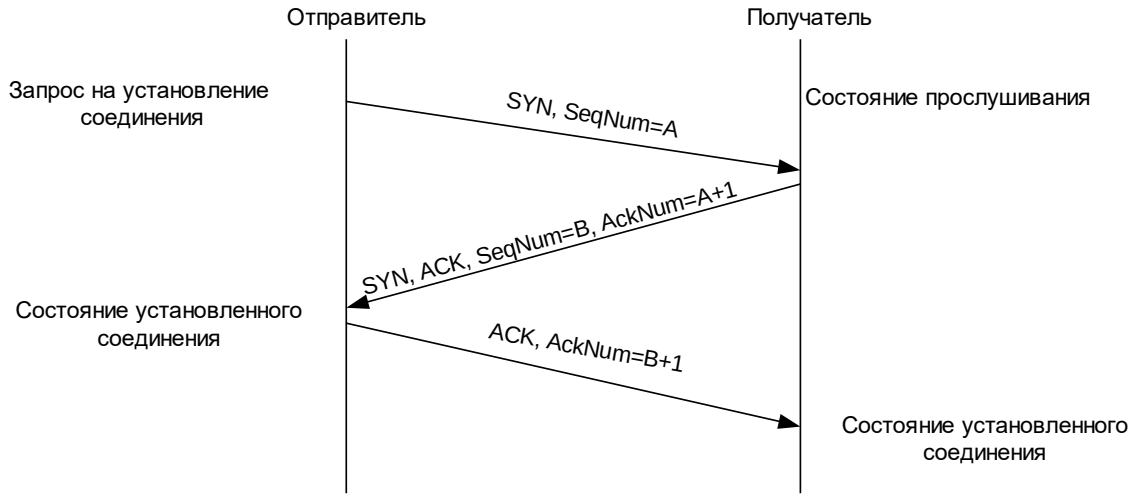


Рис. 3.4. Трехшаговое квитирование при установлении соединения

Окончание соединения

Если при осуществлении обмена данными возникает аварийная ситуация, то один узел всегда может прислать другому пакет RST, что приводит к немедленному разрыву соединения.

В случае нормального закрытия соединения выполняется процесс четырехшагового квитирования (рис. 3.5).

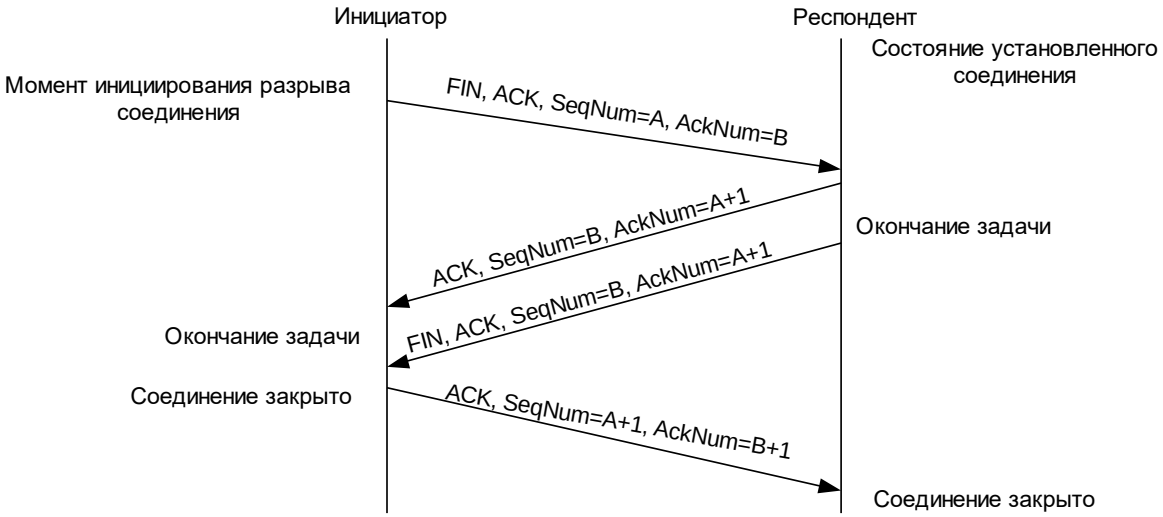


Рис. 3.5. Четырехшаговое квитирование при закрытии соединения

В этом случае сторона, которая решила инициировать окончание соединения, присылает пакет с установленными битами FIN и ACK, поля SeqNum и AckNum имеют значения, полученные в результате обмена данными (пусть это будут значения A и B). Сторона, которая получила такой пакет, возвращает подтверждение его получения, и, в свою очередь, присылает свой

пакет FIN. Последний шаг – это отправка инициатором пакета-подтверждения на пакет FIN. После выполнения вышеуказанных действий соединения считается разорванным.

Поддержание соединения

Процесс передачи пакетов после установления соединение состоит из поочередных отправки пакета и ожидания его подтверждения. При этом контроль над потерей пакетов и сохранением последовательности осуществляется с помощью полей Seq numи Ack num.

На рис. 3.6 и рис. 3.7 приведены варианты простой передачи данных с использованием механизма подтверждений.

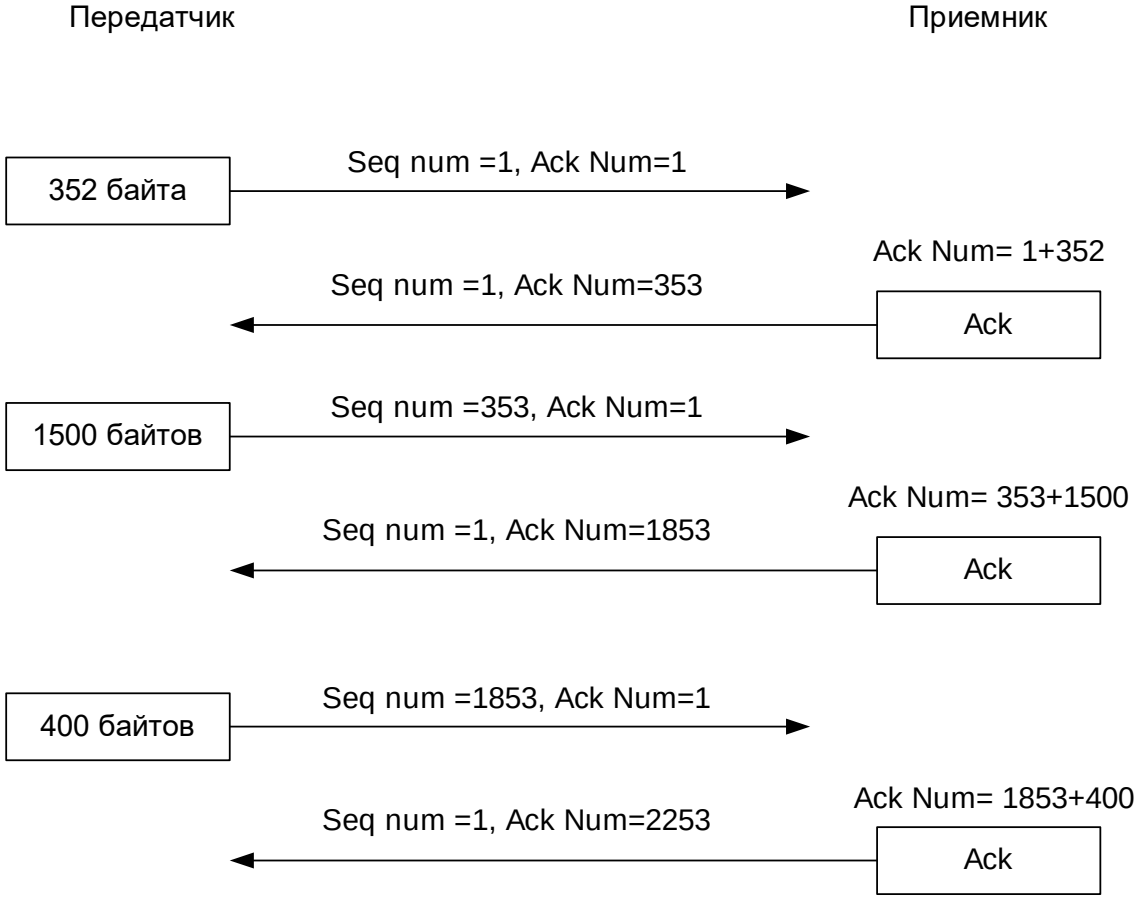


Рис. 3.6. Передача данных в одном направлении

Будем рассматривать вариант, когда начальные номера последовательностей были выбраны нулевыми. В этом случае, в соответствии с рис. 3.5, после установления соединения поля Seq Num и Ack Num равны единице.

При передаче данных протокол TCP разбивает поток на сегменты таким образом, чтобы уложиться в параметр «максимальный размер сегмента», указанный при установлении соединения. Номер последовательности (Seq Num)- это, фактически, номер первого байта передаваемого сегмента в общем потоке

байтов. Соответственно, Ack Num – это номер первого байта сегмента, прием которого подтверждается данной квитанцией.

На рис. 3.6 видно, что передача первого сегмента размером 352 байта сопровождается полем Seq Num=1 (передается сегмент, у которого первый байт является первым в потоке данных) и полем Ack Num=1 (с момента установления соединения получатель не посылал данных, поэтому мы ничего не подтверждаем). Фактически номер подтверждения N – говорит отправителю, что получатель получил N-1 байт потока и ожидает от отправителя сегмент, начинающийся с N-го байта.

Таким образом, на рис. 3.6 получатель, получив сегмент из 352 байт, определяет, что 352 байта потока получены и следует ожидать от отправителя сегмент, начинающийся с 353-го байта. Об этом он и уведомляет отправителя пакетом-подтверждением с номером Ack Num=353.

Отправитель, получив подтверждение приема первых 352-х байтов, отправляет следующий сегмент размером 1500 байтов. В этом пакете устанавливается поле Seq Num=1+352=353 (передается сегмент, у которого первый байт является 353-м в потоке) и поле Ack Num=1 (подтверждатьпо-прежнему нечего).

Получатель, ожидающий сегмент, начинающийся с 353-го байта, получает такой сегмент и определяет, что получено всего 352+1500 (1852) байтов потока и следующим должен быть сегмент, начинающийся с 1853-го байта. Об этом получатель уведомляет отправителя пакетом-подтверждением (с полем Ack Num=1853).

Отправитель, получив подтверждение приема первых 1852-х байтов, отправляет следующий сегмент размером 400 байтов. В этом пакете устанавливается поле Seq Num=1+352+1500 (передается сегмент, у которого первый байт является 1853-м в потоке) и поле Ack Num=1 (подтверждатьпо-прежнему нечего).

Получатель, ожидающий сегмент, начинающийся с 1853-го байта, получает такой сегмент и определяет, что получено всего 352+1500+400 (2252) байт потока и следующим должен быть сегмент, начинающийся с 2253-го байта. Об этом получатель уведомляет отправителя пакетом-подтверждением (с полем Ack Num=2253).

Вышеописанный процесс повторяется до момента завершения соединения. Как видно на рис. 3.6, параметр Ack Num в пакетах, передаваемых отправителем, и параметрSeq Numв пакетах, отправляемых получателем, не изменяются с момента установления соединения. Это связано с тем, что данные (поток байтов) передаются в данном примере только в одном направлении и за все время соединения ни одного байта данных не передано от получателя отправителю.

На рис. 3.7. показан пример передачи данных в обоих направлениях.

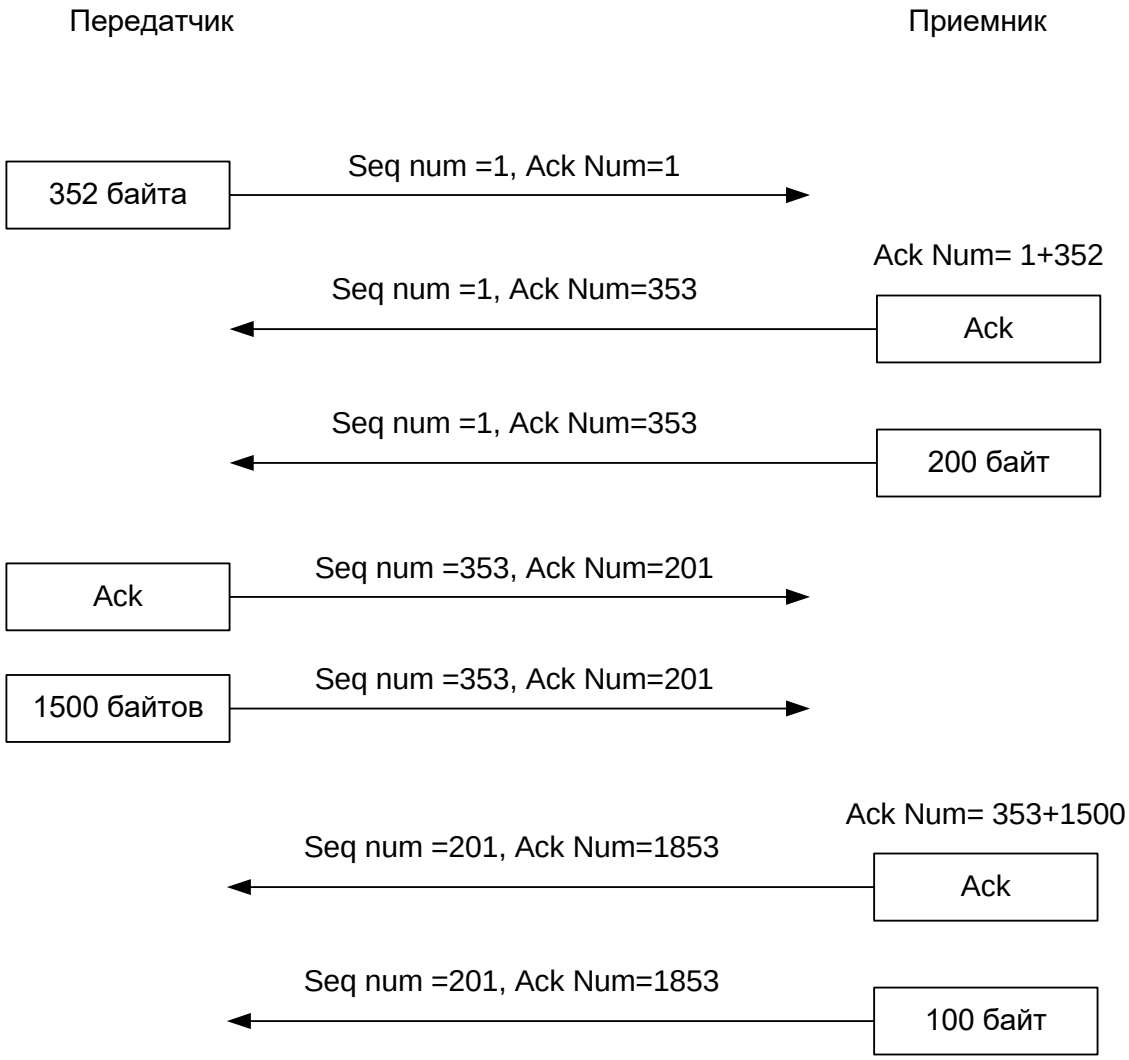


Рис. 3.7. Передача данных в двух направлениях

Как видно на рис. 3.7, передача пакетов-подтверждений (служебных) не оказывает влияния на нумерацию байтов в потоке данных. Фактически для потока данных все служебные пакеты имеют нулевую длину сегмента.

Примеры, приведенные на рис. 3.6 и рис. 3.7 являются достаточно простыми и не отражают большинства ситуаций, возникающих при реальной передаче данных. Во-первых, при передаче как данных, так и служебных пакетов-подтверждений возможна их потеря в линиях связи. В этом случае реализуется перепосылка пакета. Перепосылка заключается в повторной передаче пакета при отсутствии подтверждения в течение заданного промежутка времени. Количество повторных передач определяется настройками системы [9].

Если бы в реальных системах был реализован вышеописанный механизм передачи с перепосылкой, то это был бы очень надежный механизм передачи, который строился бы по схеме: ***передача*** – ***ожидание подтверждения*** –

передача – **….** Однако данный механизм имел бы ограниченную скорость передачи данных из-за постоянных ожиданий подтверждения.

Для устранения вышеуказанного недостатка используется механизм скользящего окна.

В каждом пакете протокола TCP есть возможность указывать такое значение, как размер окна. Данное значение определяет, какое количество свободного места для пакетов данного соединения имеется в приемном буфере узла. Т.е. сколько идущих подряд пакетов на данный момент может принять получатель. В этом случае соответствующий объем данных в пакетах отправитель отправляет непрерывно, не ожидая подтверждения. Заполнив окно сегментами, отправитель останавливается и следующий сегмент отправляет только после получения подтверждения о получении получателем первого сегмента из переданной группы. Соответственно, еще один сегмент будет передан только после получения подтверждения о приеме второго сегмента группы и т.д. Т.е. на каждом шаге окно сдвигается на один сегмент (скользит по потоку данных). Данный процесс наглядно показан на рис. 3.8.

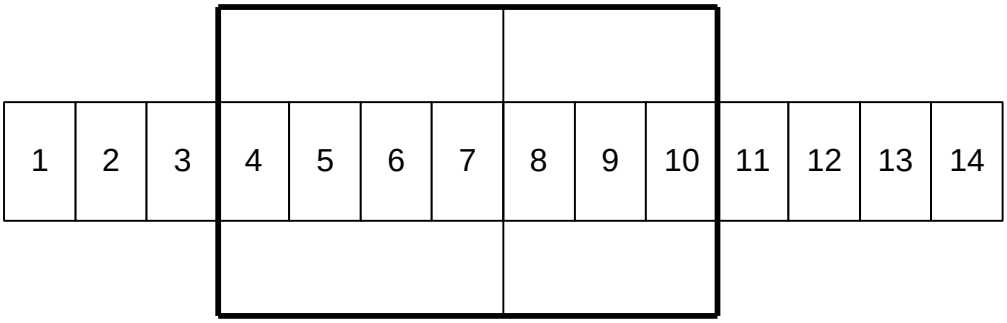


Рис. 3.8. Управление потоком данных с помощью скользящего окна

На рис. 3.8 можно выделить следующие категории сегментов:

1,2,3 – сегменты переданы, и на данный момент на них уже получены подтверждения;

4,5,6,7 – сегменты переданы, но подтверждения еще не получены;

8,9,10 – сегменты могут быть переданы без ожидания подтверждения;

11,12,13,14 – сегменты пока еще не могут быть переданы.

Если бы получение подтверждения о получении сегмента 4 (рис. 3.8) происходило после передачи сегмента 10, и так каждый раз подтверждение о получении первого сегмента окна происходило бы после передачи последнего сегмента окна, то процесс передачи ничем не отличался бы от описанного ранее (без использования окна). Однако на практике, чаще всего, подтверждения о получении сегментов окна приходят до того, как окно будет заполнено. Соответственно, смещение окна в этом случае устраняет задержку (ожидание) и, образно говоря, окно непрерывно скользит по потоку данных, устраняя задержки.

При реализации метода скользящего окна возникает проблема перепосылки. Дело в том, что при обнаружении в течение некоторого промежутка времени отсутствия подтверждения, пакет считается непринятым. Однако при использовании окна за время ожидания подтверждения могли быть переданы и другие пакеты в составе окна. Отправитель никак не может распознать, были действительно получены получателем пакеты, переданные в промежуток между передачей потерянного пакета и установление факта потери. Поэтому в простейшем случае отправитель повторяет пересылку всех пакетов, о судьбе которых ему ничего неизвестно. Решение данной проблемы возможно с применением технологии выборочного подтверждения SACK (Selected ACK), которая будет описана ниже.

Метод скользящего окна позволяет получателю ограничивать или увеличивать количество сегментов, принимаемых непрерывным потоком. Получатель в каждом пакете-подтверждении может указывать меньший размер окна (если свободное место в приемном буфере узла уменьшилось) или больший размер (если свободное место в приемном буфере узла увеличилось). Т.е. фактически данный метод – управление потоком данных со стороны получателя.

При использовании скользящего окна механизм перепосылки реализуется следующим образом. При передаче любого сегмента его копия помещается в очередь потерянных пакетов (на всякий случай), и при этом запускается специальный таймер. Если сегмент потерян, то получатель, получая следующие сегменты, будет в поле Ack Num каждого подтверждения указывать один и тот же номер (номер в потоке данных первого байта того сегмента, который ожидался, но так и не был получен). Если в течение того времени, пока работает таймер, будут приходить такие подтверждения и в момент, когда таймер отсчитает отпущенный для ожидания интервал, так и не придет подтверждение о приеме соответствующего сегмента, то сегмент будетсчитаться потерянным и будет произведена его повторная передача.

Для повышения надежности и эффективности протокола TCP реализуется ряд алгоритмов, учитывающих состояние промежуточных сетей и узлов при передаче. Основными алгоритмами являются:

- медленный старт (slow start);
- устранение перегрузок (congestion avoidance);
- быстрая перепосылка (fast retransmit).

Механизм медленного старта – это управление потоком со стороны отправителя. Данный метод заключается в том, что при старте соединения отправитель устанавливает сам себе минимальный размер окна (обычно равный максимальному размеру сегмента). Т.е. один сегмент заполняет окно. После отправки сегмента и успешного получения подтверждения отправитель увеличивает размер окна на один сегмент. После успешной передачи в окне двух сегментов окно увеличивается еще на один сегмент и т.д. Т.е. отправитель

исходит из худших предположений. Если же передача удалась, то он позволяет себе ускорить передачу на некоторую величину и т.д. Данный механизм продемонстрирован на рис. 3.9.

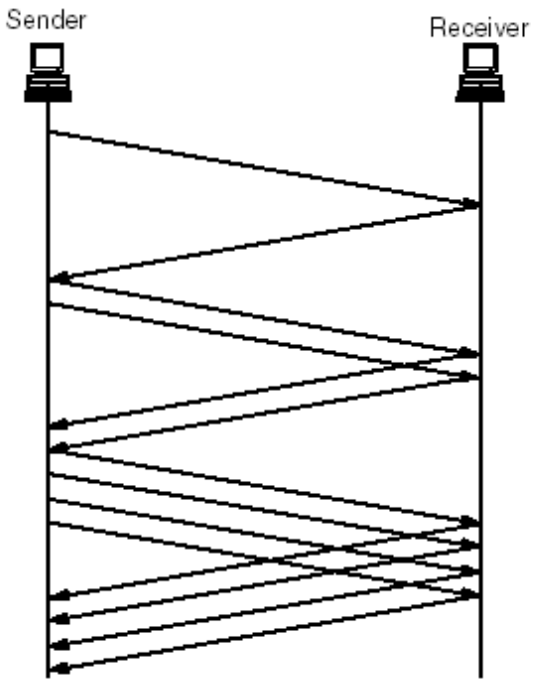


Рис. 3.9. Механизм медленного старта

Чаще всего (как на рис. 3.9) реализуется такая зависимость: на каждое одно успешное подтверждение окно увеличиваетсяна два сегмента.

Механизм устранения перегрузок в сети применяется в сочетании с механизмом медленного старта. Наличие при обмене данными данного механизма определяется специальным битом в пакете протокола TCP (поле Reserv). При работе данного метода вводится два специальных значения: окно переполнения (cwnd) ипорог медленного старта (ssthresh). Алгоритм управления потоком в этом случае заключается в следующем:

1. При установлении соединения параметр cwnd устанавливается равным 1 сегменту, а sstresh – равным 65535 (максимально возможный размер сегмента).
2. После установления соединения реализуется механизм медленного старта. При этом на каждом шаге для передачи сегментов используется минимальное из двух значений: размер окна переполнения и размер окна, объявленный получателем.
3. Если в какой-то момент обнаружена перегрузка по истечению времени ожидания или по факту наличия дублирующихся подтверждений, то порог ssthresh устанавливается равным ½ текущего размера окна. После чего

передача продолжается или, если истекло время ожидания, то окно переполнения снова сбрасывается до размера в 1 сегмент.

4. При продолжении передачи после факта перегрузки возможны два варианта. Если текущее значение окна меньше ssthresh, то реализуется стандартный медленный старт, иначе увеличение размера окна происходит линейно: на одно успешное подтверждение – на один сегмент (рис. 3.10).

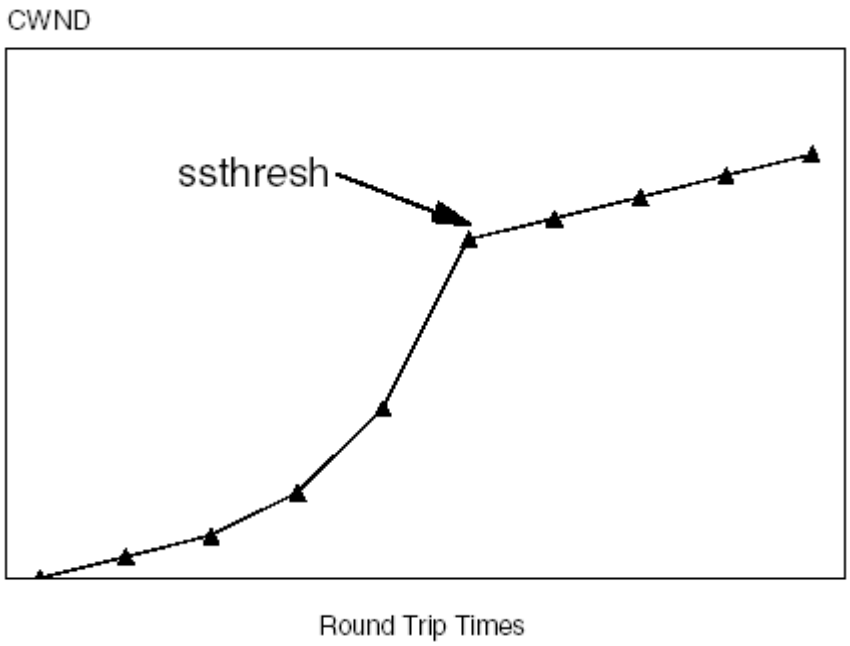


Рис. 3.10. Связь между параметрами cwnd и ssthresh

Фактически, вышеописанный алгоритм наращивает частоту отправки сегментов до возникновения перегрузки. После возникновения перегрузки частота посылки замедляется или сбрасывается до минимальной и затем снова постепенно наращивается.

Кроме вышеуказанных параметров при управлении передачей широко используется величина RTT (***Round Trip Time***) – время двойного оборота. RTT – это время от момента отправки пакета до момента получения подтверждения. Данный параметр вычисляется для каждого передаваемого пакета. Усредняясь определенным образом, данный параметр используется для автоматического расчета времени ожидания подтверждения.

Механизм быстрой перепосылки основывается на том предположении, что если отправитель начинает получать повторяющиеся подтверждения, то это сигнализирует либо о потере пакета, либо о перестановке пакетов местами при передаче. Поскольку при получении пакета подтверждение отправляется всегда немедленно, то при перестановке пакетов подтверждение все равно придет достаточно быстро (за это время успеют прийти 1-2 повторяющихся подтверждений), а при потере пакета это количество резко возрастет. В связи с

этим было предложено считать, что если получено больше чем 3 дублирующихся подтверждения, то нужно немедленно перепосылать пакет (не ожидая истечения времени ожидания). Данное решение позволяет устранить задержки, вносимые ожиданием истечения времени. Реализация данного механизма представлена на рис. 3.11.

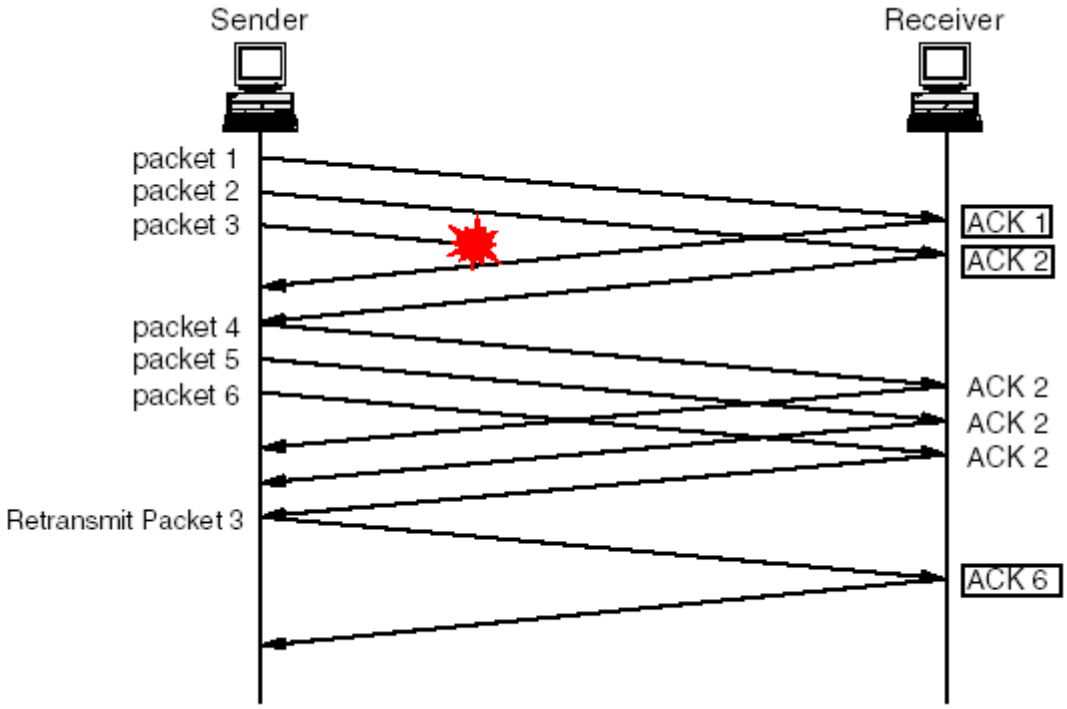


Рис. 3.11. Механизм быстрой перепосылки

Конечно, рассмотренные методы обеспечения эффективной работы протокола TCP не единственные, однако их более глубокое рассмотрение выходит за рамки данного курса.

Отдельно следует рассмотреть технологию выборочного подтверждения. Для того чтобы избежать повторной передачи целой серии пакетов, которые, возможно, и не были утеряны, получатель периодически отправляет отправителю специальный пакет-подтверждение. У такого пакета в поле Options указывается наличие выборочного подтверждения SACK. Данное подтверждение содержит указатель на начало и конец блока байтов, успешно принятых получателем. Таких блоков может быть указано сразу несколько. Чтобы избежать значительного объема данных, передаваемого в пакете SACK в случае, если теряется много сегментов, количество блоков ограничено четырьмя. Отправитель получает информацию о блоках и, обнаружив пропуски в потоке байтов, перепосылает только те пакеты, которые не были получены получателем.

На основе перечисленных принципов работы протокола TCP можно сделать выводы о его следующих преимуществах:

- высокая надежность при доставке данных;

- возможность приспособления протокола к особенностям сети;
- отсутствие необходимости в сегментации пакетов на прикладном уровне;
- выявление и корректная обработка аварийных ситуаций;

и следующие недостатки:

- сложный формат и алгоритм обработки протокола;
- наличие большого количества параметров конфигурации, которые влияют на эффективность работы протокола.

Подводя итог описанию протоколов TCP и UDP, можно сделать вывод о том, что применение протокола TCP оправдано в тех случаях, когда потеря данных при передаче является критической и необходимо обеспечить максимально надежную связь. В тех же случаях, когда потеря данных не критична и требуется обеспечить максимум скорости без требований к надежности, наиболее подходящим является протокол UDP.

Вопросы для самопроверки

1. Описать процесс работы сетей на транспортном уровне
2. Перечислить протоколы транспортного уровня
3. Описать процедуру передачи данных в двух направлениях
4. Раскрыть связь между параметрами cwnd и ssthresh
5. Описать механизм быстрой перепосылки

4. ВСПОМОГАТЕЛЬНЫЕ СЕТЕВЫЕ СЛУЖБЫ

4.1. Протокол DHCP

При подключении к сети узел (компьютер) должен быть определенным образом сконфигурирован, т.е. для узла должны быть заданы определенные параметры:

- IP-адрес;
- маска;
- адрес шлюза, через который узел выходит во внешние сети;
- адрес DNS-сервера;
- и т.п.

В большинстве случаев такая настройка для конечного пользователя является нетривиальной задачей, и для решения проблемы администратору приходится выполнять конфигурирование каждого узла обслуживаемой сети.

Для того чтобы упростить и упорядочить процедуру конфигурирования узлов сети, были предложены протоколы автоматической конфигурации. Такие протоколы должны были при загрузке компьютера выдавать ему набор параметров, необходимых для функционирования сети. Таким протоколом стали протокол BOOTP и его расширение – протокол DHCP (***Dynamic Host Configuration Protocol***). На последнем мы остановимся подробнее.

При работе протокола DHCP предполагается, что в сети функционирует специальный узел – DHCP-сервер, который занимается выдачей настроек, а остальные узлы являются DHCP-клиентами, кроме того, в сети могут присутствовать DHCP-ретрансляторы. Для передачи данных протокола DHCP в качестве транспортного используется протокол UDP, порт 68.

Рассмотрим по шагам процесс получения настроек с использованием протокола DHCP.

При загрузке узлу, для которого указано автоматическое получение параметров, присваивается IP-адрес 0.0.0.0.

В момент установления сетевых параметров узел-клиент посылает запрос DHCPDISCOVER, который отправляется по адресу 255.255.255.255 (всем узлам данной сети).

Сервер, получив данный запрос, отвечает на него ответом DHCPOFFER, в котором передает свой IP-адрес и IP-адрес клиента. После этого узел-клиент запрашивает дополнительные настройки запросом DHCPREQUEST. Сервер возвращает требуемые параметры в ответе DHCPACK.

В общем случае существует несколько подходов к назначению IP-адресов (рис. 4.1).

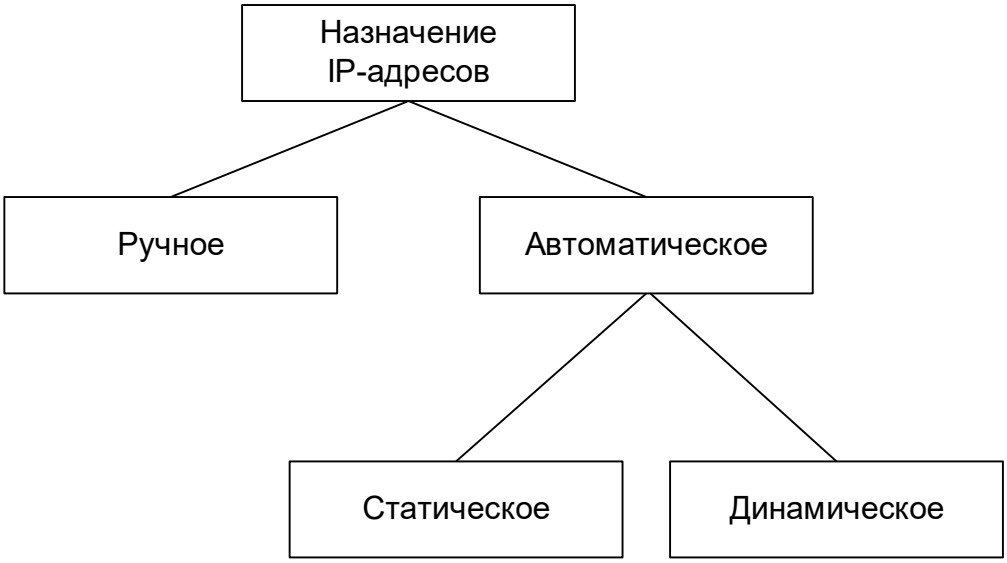


Рис. 4.1. Способы назначения IP-адресов

Статическое назначение предполагает, что каждому узлу всегда назначается один и тот же IP-адрес. В этом случае возможна привязка выдаваемого адреса к какому–либо признаку узла. Такое назначение параметров позволяет администратору управлять назначением адресов конкретным узлам и, кроме того, при изменении параметров настройки сети эти параметры достаточно поменять на сервере, и каждый узел получит новые значения автоматически.

В тех случаях, когда безразлично, какому узлу какой адрес назначен, или диапазон доступных адресов достаточно мал и его не хватает на все узлы сети, применяют динамическое назначение. В том случае, когда диапазон доступных адресов мал, можно воспользоваться тем фактом, что не все узлы одновременно будут работать в сети. Если количество одновременно работающих узлов не превысит размер диапазона, то динамическое присвоение решит проблему нехватки адресов. В этом случае на сервере указывается диапазон доступных адресов, и каждому узлу по запросу выдается один из адресов из диапазона. Выданный адрес помечается как занятый. Если в диапазоне не остается свободных адресов, то последующим узлам отказывают в получении адреса. Если узел выключается из сети, то IP-адрес освобождается и может быть выдан другому узлу.

Для реализации такого механизма используется понятие аренды адреса. При выдаче адреса в своем ответе сервер указывает, на какое время выдается адрес. На узле-клиенте запускается соответствующий таймер, и время аренды делится на три промежутка T1, T2, T3.

По истечении промежутка T1 узел отправляет серверу запрос DHCPRELEASE с просьбой продлить аренду. Если сервер продлевает аренду, он уведомляет узел сообщением DHCPACK. Если по каким-либо причинам DHCP-сервер не отвечает, по истечении промежутка T2 узел начинает повторную

процедуру поиска нового DHCP-сервера (DHCPDISCOVER). Если найти новый сервер не удалось, то по истечении промежутка T3 узел прекращает свою деятельность в сети и освобождает арендуемый IP-адрес.

В табл. 4.1 приведен полный перечень запросов и сообщений, передаваемых между DHCP-сервером и клиентом.

Табл. 4.1. Перечень DHCP-сообщений

DHCPDISCOVER	Широковещательный запрос на поиск DHCP-сервера в сети
DHCPOFFER	Ответ сервера, возвращающий IP-адрес сервера
DHCPREQUEST	Запрос клиентом конфигурационных параметров
DHCPACK	Подтверждение выбранного IP и передача дополнительных параметров конфигурации
DHCPNACK	Отклонение выбранного IP
DHCPDECLINE	Сообщение клиента о невозможности использовать выданный IP-адрес
DHCPRELEASE	Запрос на продление аренды IP-адреса
DHCPINFORM	Информирование сервера клиентом об уже имеющихся настройках

4.2. Служба DNS

На ранних этапах развития компьютерных сетей обращение к ресурсам (серверам) происходило путем указания соответствующего IP-адреса. С бурным ростом числа ресурсов в глобальной сети стало невозможным запоминать такое большое количество бессмысленных числовых идентификаторов. Для человека значительно более удобным оказалось запоминание символьных осмысленных имен ресурсов. Именование ресурсов символьными именами очень удобно, но порождает массу технических проблем.

Первоначально на каждом компьютере сети хранился файл с таблицей соответствия имен и IP-адресов. Однако такой подход был приемлемым только при малых масштабах сети. В глобальном масштабе невозможно на каждом узле хранить огромную таблицу и непрерывно отслеживать достоверность информации в ней.

Решением данной проблемы стала разработка иерархической структуры символьных имен с реализацией централизованного управления данной структурой в мировом масштабе. Служба управления данной структурой получила название DNS (***Domain Name Service***)

В соответствии с иерархической структурой в общем виде символьное имя ресурса представляет собой доменное имя, состоящее из отдельных элементов – доменов, разделяемых при записи точками (рис. 4.2).

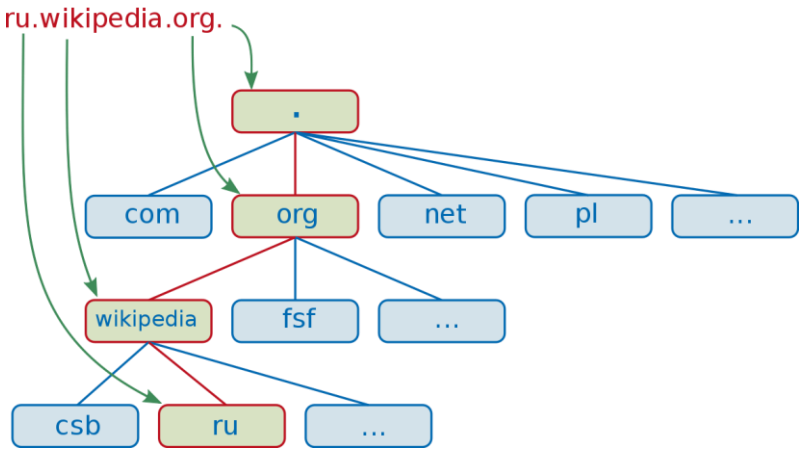


Рис. 4.2. Структура доменного имени

Главным отправным элементом имени является корневой домен, который в явном виде при записи имени не указывается, но наличие которого всегда подразумевается. Далее идут отдельные домены. Чем левее в записи располагается домен, тем бо лее низшим считается его уровень. Первым после корневого идет TLD (***Top Level Domain***) – домен верхнего уровня, а за ним домены второго, третьего и т.д. уровней.

Группа доменов, считая от корневого, называется зоной. Каждая доменная зона является зоной ответственности одного или нескольких DNS-серверов.

Иерархическая структура DNS-серверов примера приведена на рис. 4.3.

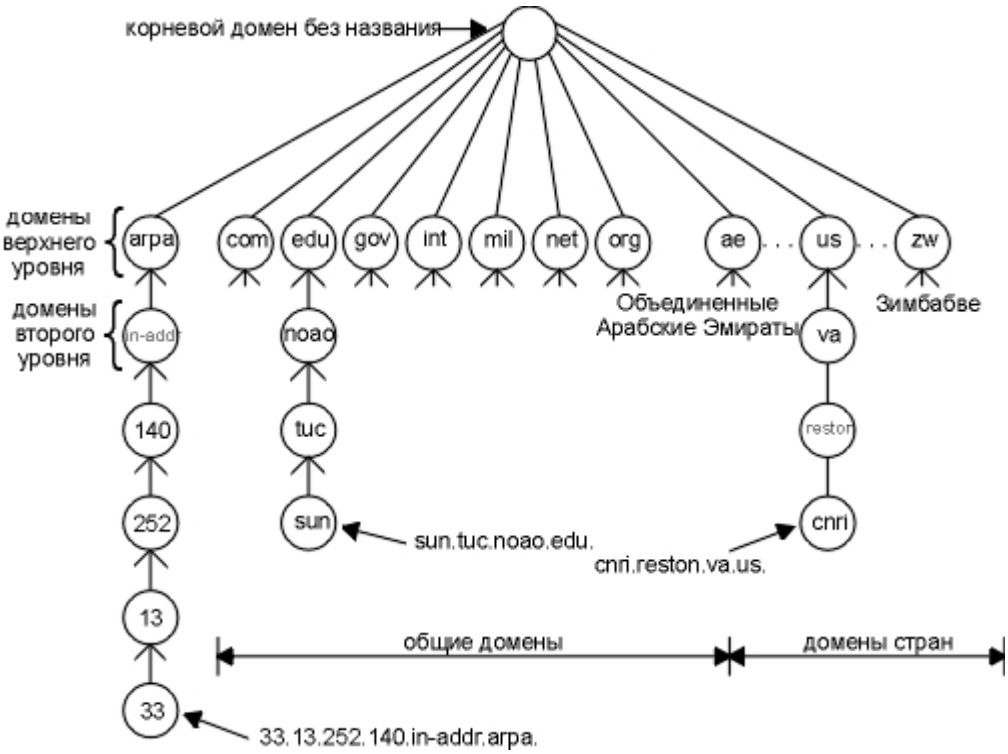


Рис. 4.3. Поиск по иерархической структура DNS

Корневую зону (корневой домен) обслуживает группа серверов, которые называются «корневые сервера Интернет». Данные сервера равномерно распределены по всему миру и обеспечивают функционирование системы DNS. База данных каждого корневого сервера хранит таблицу соответствия доменов верхнего уровня и IP-адресов соответствующих DNS-серверов.

К доменам верхнего уровня относятся все географические домены, присвоенные каждой стране, а также домены: com, net, org, mil, gov, edu, info, biz и т.д. Вопросами регистрации доменов верхнего уровня занимается организация IANA.

Каждую зону (com, ru, info и т.д.) обслуживает свой DNS-сервер (сервера). В задачи сервера входит поддержание БД и выдача по запросу информации о соответствии IP-адреса имени, и наоборот.

Процедура преобразования символьного имени в IP-адрес может выполняться двумя способами:

1) нерекурсивный поиск;
2) рекурсивный поиск.

Нерекурсивный поиск. Если на узле возникает необходимость преобразования символьного имени в IP-адрес, то распознавание (разрешение) IP-адреса производится путем анализа доменного имени, начиная с самого старшего домена. Например, при распознавании доменного имени **www.domen.spb.ru** будут выполняться следующие действия.Узел обратится с DNS-запросом (порт 53) к одному из корневых серверов с целью определить IP-адрес DNS-сервера, обслуживающего зону **ru**. Корневой сервер находит в своей базе соответствующую запись и возвращает IP-адрес соответствующего DNS-сервера. Далее узел посылает новый запрос DNS-серверу, обслуживающему зону **ru**, с целью определить IP-адрес DNS-сервера, обслуживающего зону **spb.ru** . Результат поиска возвращается узлу. Далее процесс повторяется для зоны **domen.spb.ru**. В конце концов, DNS-сервер зоны **domen.spb.ru** содержит запись об IP-адресе узла **www.domen.spb.ru**. Это и есть конечный результат поиска. Данный процесс графически изображен на рис. 4.4.

Из рисунка можно сделать вывод, что данная процедура имеет следующие недостатки:

- сложность реализации;
- необходимость оперирования списком корневых серверов, его обновления и т.п.;
- значительные временные затраты на поиск;
- значительная нагрузка на сеть множественными запросами.

Рекурсивный поиск. Данный процесс заключается в том, что каждый узел (клиент), подключенный к глобальной сети, обслуживается некоторым провайдером, который имеет свой собственный DNS-сервер, реализующий вышеприведенную процедуру нерекурсивного поиска. Обращения к таким

серверам могут передаваться по цепочке от низших провайдеров к высшим. В результате на каком-то уровне будет запущена процедура нерекурсивного поиска. Таким образом, удается избавиться от необходимости на каждом узле реализовывать модуль нерекурсивного поиска. Достаточно, чтобы узел пользователя мог послать запрос вышестоящему DNS-серверу и обработать ответ. Схема подобного поиска приведена на рис. 4.5.

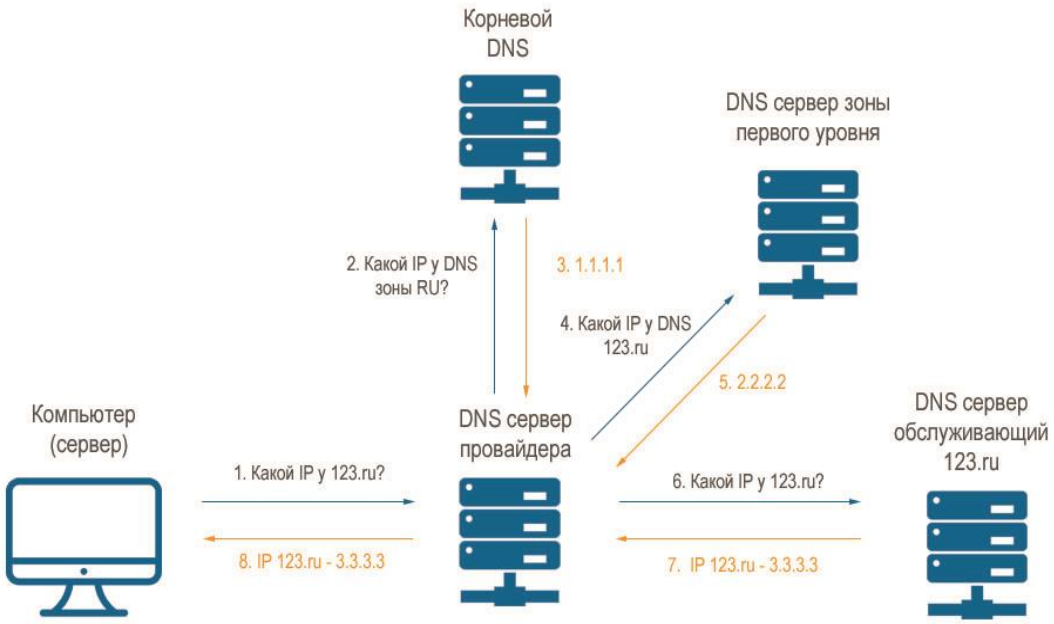


Рис. 4.4. Нерекурсивная процедура поиска

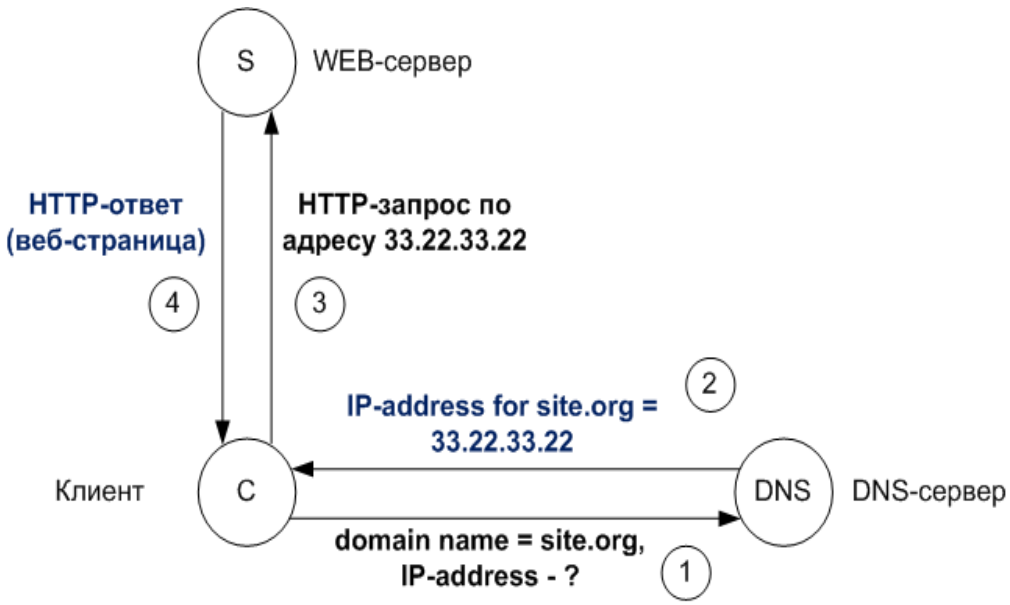


Рис. 4.5. Рекурсивная процедура поиска

Рекурсивная процедура поиска позволяет упростить процедуру поиска для конечных узлов и разгрузить сеть от множественных запросов. Фактически все сложные действия возлагаются на DNS-сервера вышестоящих провайдеров. Кроме того, большинство DNS-серверов являются кэширующими. Т.е. сервера сохраняют в течение некоторого времени определенный объем обработанных запросов в кэше. В случае повторного запроса процедура поиска не запускается, а ответ извлекается из кэша [10].

Исходя из вышесказанного, можно описать, в чем состоит процедура регистрации доменного имени.

Пользователь должен зарегистрировать доменное имя в той или иной зоне в специально уполномоченных для этого организациях. Кроме этого, пользователь должен иметь узел, подключенный к глобальной сети с уникальным IP-адресом, который будет использоваться в качестве DNS-сервера. В общем случае провайдер может предоставлять услуги хостинга, регистрации имен и размещения DNS–серверов на своей технической площадке.

Регистратор имени заполняет специальную форму, данные которой размещаются в базе данных DNS-сервера соответствующей зоны. Например, для домена **domen** в зоне **spb.ru** примерный вид записи представлен в табл. 4.2. Как видно из записи, зону может обслуживать не один, а несколько серверов (NS, NS1, NS2). Чаще всего это является обязательным условием регистрации имени для повышения надежности системы DNS. Поле MX определяет адрес обработчика почты, т.е. того узла, на который будет перенаправляться почта для адресов вида **…@…..domen. spb.ru (**подробнее о функционировании электронной почты в лекции №5, часть 2**)**.

*Табл. 4.2. Пример записи в БД DNS-сервера зоны **spb.ru** для домена **domen***

Поле	Значение
NAME	domen
NS	195.58.238.15
NS1	195.58.224.40
NS2	193.14.23.1
MX	195.58.224.42
Register date	10.10.2003
Last update	01.03.2004
Expired	10.10.2006

Пример реального файла конфигурации для DNS-сервера зоны domen.spb.ru под управление ОС Linux будет следующим:

```
//                          Имя DNS-сервера          e-mail ответственного лица
@        IN    SOA         ns.domen.spb.ru.         hostmaster.domen.spb.ru. (
                           5;                       //серийный номер записи
                           8H;                      //период обновления информации
                                                    //другими серверами
                           2H;                      //тайм-аут при неудачном обновлении
```

		1W;	//максимальное TTL
		1D);	//минимальное TTL
	A	**195.58.238.15**	//IP-адрес данного сервера
www	**A**	**195.58.238.16**	//IP-адрес узла www. **domen.spb.ru**
ftp	**A**	**195.58.238.17**	//IP-адрес узла ftp. **domen.spb.ru**
mail	**A**	**195.58.238.18**	//IP-адрес узла mail. **domen.spb.ru**
	MX	**10 mail.domen.spb.ru**	//указатель на первичный обработчик //почты
	MX	**20mail2.domen.spb.ru**	//указатель на вторичный обработчик //почты

В данном случае первая часть IN SOA описывает общие параметры DNS-сервера. В параметре A указывается IP-адрес данного DNS-сервера. Далее для каждого поддомена, зарегистрированного в данной зоне, указывается IP-адрес и может быть указан обработчик почты. В данном случае обработчики указаны в конце, общие для всех поддоменов (параметр MX). Обработчиков два, если один из них недоступен, то почта будет передана другому. Приоритетность выбора обработчика определяется величиной числового параметра перед его адресом.

Все процедуры поиска и все структуры данных, о которых мы только что говорили, относится к поиску в прямой зоне. Прямой зоной (прямым поиском) называется иерархия, преобразующая доменное имя в IP-адрес. Обратное преобразование осуществляется подобными методами в обратной зоне – иерархии, преобразующей IP-адрес в доменное имя. Структура данной иерархии представлена на рис. 4.6.

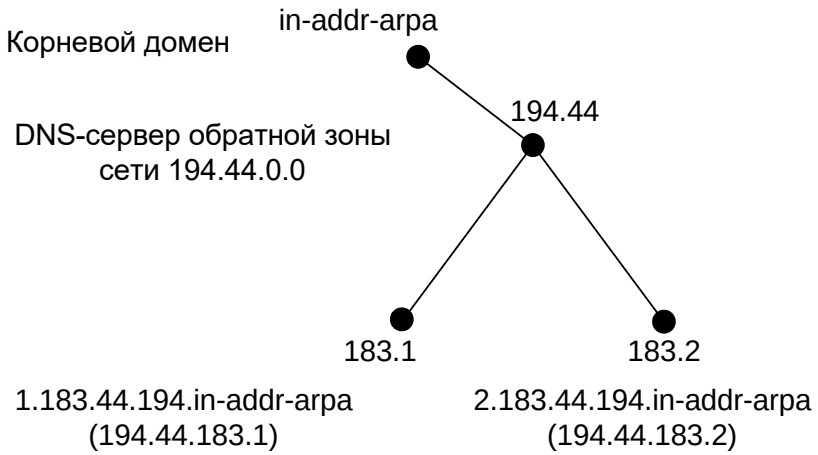


Рис. 4.6. Иерархия обратной зоны DNS

Процедуры рекурсивного и нерекурсивного поиска в обратной зоне аналогичны таким же процедурам в прямой зоне.

Недостатком классической схемы построения обратной зоны является то, что она привязана к стандартному делению на классы A, B и C. В настоящее же время широкое распространение получили бесклассовые сети. Для решения задачи поиска в обратной зоне при использовании бесклассовых сетей существует несколько решений, которые пока жестко не стандартизованы и рассмотрение которых выходит за рамки данного курса.

5. ПРОТОКОЛЫ ПРИКЛАДНОГО УРОВНЯ

На прикладном уровне и уровне представления данных модели передачи данных OSI работает множество протоколов, каждый из которых решает свою прикладную задачу. Примерами таких протоколов являются:

- HTTP (***Hyper Text Transfer Protocol***) – протокол, по которому передаются пакеты, содержащие HTML-страницы и прочие WEB-ресурсы;
- FTP (***File Transfer Protocol***) – протокол пересылки файлов;
- SMTP (***Simple Message Transfer Protocol***) -протокол пересылки почтовых сообщений;
- POP3 (***Post Office Protocol***) – почтовый протокол взаимодействия почтового клиента и сервера;
- Telnet и SSH – протоколы для удаленной работы с компьютером;
- SNMP (***Simple Network Manage Protocol***) – протокол управления сетевыми устройствами;
- и т.д.

5.1. Протокол HTTP

Hyper Text Transfer Protocol (HTTP) – это протокол прикладного уровня, который обеспечивает пересылку, модификацию и прочие действия с разнообразными ресурсами (данными). Для указания ресурсов, к которым применяются упомянутые действия, используется механизм ссылок. Ссылка – это идентификатор ресурса, который определяет его расположение. Структура такого идентификатора (URL – Universal Resource Locator) следующая:

http://domain:port/directory/file,

где

http – тип протокола;

domain – доменное имя;

port – номер порта (стандартным для HTTP является порт 80);

directory – каталог, в котором расположен ресурс;

file – файл, в котором расположен ресурс.

В качестве ресурса могут выступать любые данные: текст, изображение и т.д. Принцип работы протокола HTTP основывается на системе “запрос-ответ”. При обмене данными по протоколу HTTP участники обмена обычно делятся на клиент и сервер. Клиент – это программа, которая посылает запрос, а сервер – это программа, которая отсылает ответ на запрос.

В общем видеоформат запроса изображен на рис. 5.1.

Строка статуса	Метод
	SP
	URL
	SP
	Версия HTTP
	CRLF
Заголовок запроса	
CRLF	
Содержимое запроса	

Рис. 5.1. Формат HTTP-запроса

Следует отметить, что формат, приведенный на рис. 5.1 является обобщенным и в конкретном случае при применении расширений протокола HTTP может иметь другую структуру. В данном случае первым идет строка статуса. К строке статуса относятся следующие поля:

- метод – вид действия, которое будет выполняться над ресурсом, может принимать значение: GET, HEAD, PUT, POST,DELETEи др.;
- URL – идентификатор ресурса;
- версия HTTP – номер версии протокола (может принимать значение HTTP 0.9, 1.0, 1.1).
- SP и CRLF – это специальные обозначения для пробела (SP) и символов переводастроки и каретки (CRLF).

В табл. 5.1 рассмотрены основные методы протокола HTTP.

Табл. 5.1. Назначение основных методов протокола HTTP

Метод	Назначение
GET	Метод служит для получения произвольной информации, которая указана идентификатором. Если URL ссылается на некоторый процесс, то в качестве ответа будут выступать результаты работы процесса. Возможна ситуация, когда данный метод заменяется на условный GET. В таком запросе в заголовке запроса должно быть поле “If-Modified-Since”. Если ресурс не изменялся с указанной даты, то будет возвращено специальное сообщение, в противоположном случае ответ будет аналогичен ответу на обычный запрос GET.
HEAD	Метод, аналогичный методу GET, но при ответе возвращаются лишь служебные поля ответа, а содержание (сам ресурс) не возвращается. Данный метод может быть использован для получения информации о ресурсе без передачи самого ресурса.

PUT	Данный метод дает указание серверу создать новый ресурс или сохранить информацию из запроса в виде ресурса, на который указывает URL.
POST	Метод используется для того, чтобы сервер принял информацию из поля “Содержание запроса” как входную. Данная информация может быть использованная как входная информация процессов, программ (скриптов) и как новая информация для добавления в базы данных и т.п. информационные структуры. Реальная функция метода определяется сервером и типом ресурса, на который указывает идентификатор.
DELETE	Метод используется для уничтожения ресурса, указанного с помощью URL.
И т.д.	

Поле “Заголовок запроса” может содержать разнообразнейшие параметры, примеры которых приведен в табл. 5.2. Обычно параметры указываются в виде: **название параметра: значение,** для разделения параметров используется символ CRLF.

Табл. 5.2. Назначение основных параметров заголовка запроса

Параметр	Назначение
Accept	Уведомляет о типах данных, которые могут быть распознаны и обработаны клиентом
Accept-Charset	Указывает кодовую таблицу символов, которая используется для данного ресурса.
Accept-Encoding	Указывает тип кодирования, которое используется для данного ресурса
Accept-Language	Указывает язык, который используется для данного ресурса
Authorization	Содержит параметры авторизации
From	Указывает электронный адрес отправителя запроса
If-Modified-Since	Параметры, определяющие работу метода “условный GET”
If-Unmodified-Since	
Range	Диапазон данных в случае, если сервер поддерживает закачку ресурса по частям
Referer	Идентификатор ресурса, из которого выполняется запрос
User-Agent	Указывает номер версии и тип программы-клиента

Extension-Header	Указывает возможные расширения стандартных параметров.
Max-Forwards	Максимальное число пересылок запроса через промежуточные узлы (прокси-сервера)
Proxy-Authorization	Авторизация средствами прокси-сервера
Pragma	Управление кэшированием запрашиваемого ресурса
и т.д.	

Более полный перечень параметров запроса можно найти в дополнительной литературе.

Непосредственно содержание запроса, если оно присутствует, содержит информацию в соответствующем формате. Пример HTTP-запроса, полученного с помощью средств отладки браузера Opera, приведен на рис. 5.2:

```
Hypertext Transfer Protocol
GET /contacts.html HTTP/1.1
Host: design.originweb.info
User-Agent: Opera/9.80 (Windows NT 6.2; Win64; x64; YB/5.0.3) Presto/2.12.388 Version/12.18
Accept: text/html, application/xml;q=0.9, application/xhtml xml, image/png, image/jpeg, image/gif, image/x-xbitmap, */*;q=0.1
Accept-Language: en
Accept-Charset: iso-8859-1, utf-8, utf-16, *;q=0.1
Accept-Encoding: deflate, gzip, x-gzip, identity, *;q=0
Connection: Keep-Alive, TE
TE: deflate, gzip, chunked, identity, trailers
```

Рис. 5.2. Пример HTTP-запроса

Общий формат HTTP-ответа приведен на рис. 5.3.

Строка статуса	Версия HTTP
	SP
	Код статуса
	SP
	Пояснительная надпись
	CRLF
Заголовок ответа	
CRLF	
Заголовок содержимого	
CRLF	
Содержимое ответа	

Рис. 5.3. Формат HTTP-ответа

Первым идет поле “Строка статуса”, оно содержит такие элементы, как:

- версия протокола HTTP;
- код статуса – цифровой параметр, который сообщает о результате обработки запроса;
- пояснительная надпись – это текстовая строка, которая содержит фразу, объясняющую пользователю код статуса в доступном виде.

Цифровой код статуса состоит из трех цифр и имеет такой формат (вместе с кодом указана пояснительная надпись):

Информационный (1xx)

- 100 Continue
- 101 Switching Protocols

Успешного завершения (2xx)

- 200 OK
- 201 Created
- 202 Accepted
- 203 Non-Authoritative Information
- 204 No Content
- 205 Reset Content
- 206 Partial Content

Перенаправление (3xx)

- 300 Multiple Choices
- 301 Moved Permanently
- 302 Moved Temporarily
- 303 See Other
- 304 Not Modified
- 305 Use Proxy

Ошибка клиента (4xx)

- 400 Bad Request
- 401 Unauthorized
- 402 Payment Required
- 403 Forbidden
- 404 Not Found
- 405 Method Not Allowed
- 406 Not Acceptable
- 407 Proxy Authentication Required
- 408 Request Timeout
- 409 Conflict
- 410 Gone
- 411 Length Required
- 412 Precondition Failed
- 413 Request Entity Too Large

- 414 Request-URI Too Long
- 415 Unsupported Media Type

Ошибка сервера (5xx)

- 500 Internal Server Error
- 501 Not Implemented
- 502 Bad Gateway
- 503 Service Unavailable
- 504 Gateway Timeout
- 505 HTTP Version Not Supported

Поле “Заголовок ответа” может содержать дополнительную информацию об ответе на запрос, о типе сервера, текущую дату и др. Примеры параметров заголовка приведены в табл. 5.3.

Табл. 5.3. Параметры заголовка ответа

Параметр	Назначение
Retry-After	Указывает на необходимость повторить запрос через некоторый промежуток времени
Server	Указывает тип сервера
WWW-Authenticate	Указывает на необходимость авторизации
и т.д.	

Поле “Заголовок содержимого” содержит информацию непосредственно о содержимом ответа (о ресурсе). Примеры параметров данного заголовка приведены в табл. 5.4.

Табл. 5.4. Параметры заголовку содержания ответа

Параметр	Назначение
Allow	Перечень методов, который поддерживает ресурс
Content-Encoding	Кодирование содержания ответа
Content-Language	Язык содержания ответа
Content-Length	Длина содержания ответа (ресурса)
Content-Type	Тип содержания ответа (текст, изображение и т.п.)
Last-Modified	Дата последней модификации ресурса
и т.д.	

Непосредственно содержание ответа, если оно присутствует, содержит информацию в соответствующем формате. Пример HTTP-ответа, полученного с помощью программы Ethereal, приведен на рис. 5.4.

```
Hypertext Transfer Protocol
HTTP/1.0 200 OK\r\n
Date: Mon, 12 Apr 2004 18:34:21 GMT\r\n
Server: Apache/1.3.23 (Unix)(ASP/Linux) mod_ssl/2.8.7 OpenSSL/0.9.6b DAV/1.0.3 PHP/4.1.2 mod_perl/1.26\r\n
Last-Modified: Mon, 05 Apr 2004 17:16:17 GMT\r\n
ETag: "502d5-115b-40719461"\r\n
Accept-Ranges: bytes\r\n
Content-Length: 4443\r\n
Content-Type: text/html\r\n
X-Cache: MISS from tranzit.spb.ru \r\n
Proxy-Connection: keep-alive\r\n
\r\n
```

Рис. 5.4. Пример HTTP-ответа

Вышеприведенное описание HTTP-протокола носит описательный характер, более подробное рассмотрение данного вопроса выходит за рамки курса. Более подробную информацию можно найти в дополнительной литературе.

5.2. Протокол FTP

Протокол FTP представляет собой простой, но эффективный протокол передачи файлов с одного узла на другой с реализацией дружественного пользователю интерфейса. Принцип работы протокола основывается на том, что он обеспечивает доступ к совокупности файлов, расположенных на удаленном узле (сервере) так же, как и к файлам, расположенным на локальном компьютере.

Модель работы протокола FTP представлена на рис. 5.5.

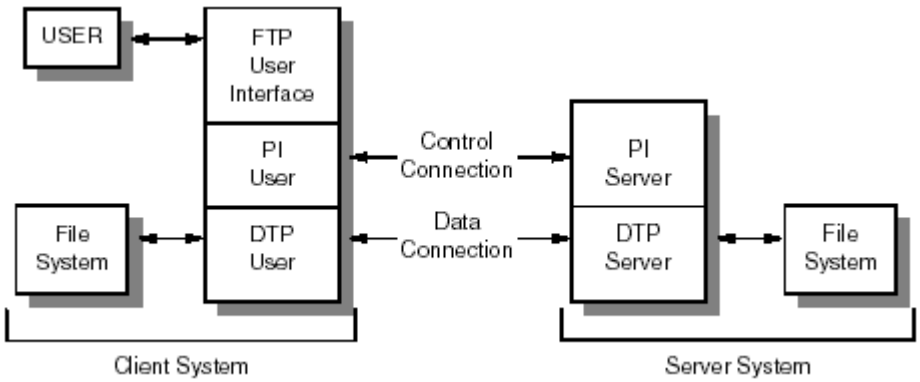


Рис. 5.5. Модель передачи данных по протоколу FTP

В составе FTP-клиента можно выделить три компоненты: интерфейс пользователя, интерпретатор протокола (PI), модуль передачи данных (DTP). Протокол использует два TCP-соединения с портами 21 и 20. Первое (порт 21) используется для установления соединения между интерпретаторами клиента и

сервера, передачи-приема команд и ответов. Процесс общения клиента и сервера напоминает протокол удаленного управления TELNET, т.е. это простой набор команд по управлению пересылкой файлов и работе с файловой системой удаленного сервера.

По командам интерпретатора модуль пересылки данных обращается к файловым системам клиента или сервера и выполняет обмен данными (порт 20).

Установление соединения в соответствии с протоколом FTP происходит путем установления соединения на транспортном уровне средствами протокола TCP. Происходит эта операция, если прикладная программа требует установки соединения и при этом указано, что это должен быть именно протокол FTP (путем указания в составе URL или стандартного порта данного протокола).

Формат команды и ответа одинаковый и изображен на рис. 5.6.

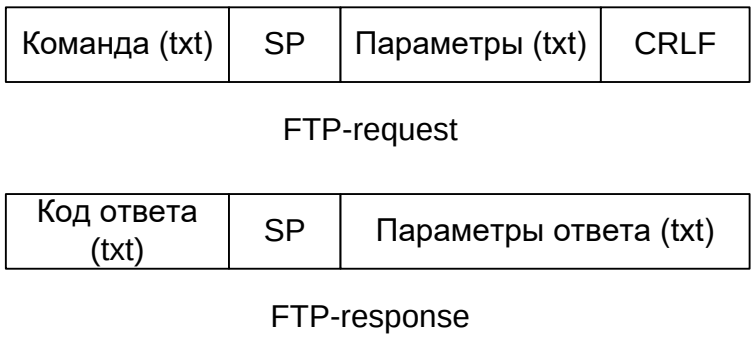


Рис. 5.6. Формат команды и ответа протокола FTP

Для объяснения работы протокола рассмотрим сценарий входа на FTP-сервер, выполненный с помощью программы Ethereal. Выполняется вход на FTP-сервер 192.168.2.1.

```
//Трехшаговое квитирование при установлении соединения
Transmission Control Protocol, Src Port: 1026 (1026), Dst Port: ftp (21), Seq: 88759, Ack: 0, Len: 0
Transmission Control Protocol, Src Port: ftp (21), Dst Port: 1026 (1026), Seq: 3862062230, Ack: 88760, Len: 0
Transmission Control Protocol, Src Port: 1026 (1026), Dst Port: ftp (21), Seq: 88760, Ack: 3862062231, Len: 0
//Сервер готов для входа пользователя
Response code: Service ready for new user (220)
Response arg: Unact WWW Server
//Команда USER с указанием имени пользователя
Request command: USER
Request arg: victor
//Имя верное, необходим пароль
Response code: User name okay, need password (331)
Response arg: Password required for victor.
//Передача пароля
Request command: PASS
Request arg: *********
//Сервер сообщает, что пользователь успешно зашел на сервер
Response code: User logged in, proceed (230)
Response arg: User victor logged in.
//Команда, которая требует от сервера ответа о действиях, которые клиент//может выполнять с файлами
```

Request command: REST
Request arg: 0
//Сервер сообщает, что клиент должен применять команды STORE или//RETRIEVE
Response code: Requested file action pending further information (350)
Response arg: Restarting at 0. Send STORE or RETRIEVE to initiate transfer
//Клиент передает команду, чтобы определить текущий каталог на сервере
Request command: PWD
//Сервер сообщает текущий каталог
Response code: PATHNAME created (257)
Response arg: "/" is current directory.
//Клиент спрашивает информацию о системе
Request command: SYST
//Сервер сообщает информацию о системе
Response code: NAME system type (215)
Response arg: UNIX Type: L8
//Клиент дает команду установить соединение из узла с адресом
//192.168.2.200
Request command: PORT
Request arg: 192,168,2,200,4,3
//Сервер отвечает, что соединения установлено
Response code: Command okay (200)
Response arg: PORT command successful
//Клиент запрашивает перечень файлов в текущем каталоге
Request command: LIST
//Сервер отвечает, что может выполнить эту операцию и для этого откроет //передачу данных
Response code: File status okay; about to open data connection (150)
Response arg: Opening ASCII mode data connection for file list
//Передача данных происходит через порт 20 через TCP-соединение
Transmission Control Protocol, Src Port: ftp-data (20), Dst Port: 1027 (1027), Seq: 3856793957, Ack: 88973, Len: 315
FTP Data
315 Bytes
//Сервер сообщает, что закрывает передачу данных
Response code: Closing data connection (226)
Response arg: Transfer complete.

Далее аналогично осуществляется переход по каталогам, выбор файлов, их пересылка и др. Общий перечень команд протокола FTP представлен в табл. 5.5.

Табл. 5.5. Перечень основных команд протокола FTP

Команда	Назначение
OPEN	Установление соединения с удаленным узлом
USER	Передача имени пользователя
PASS	Передача пароля пользователя
SITE	Передача удаленному узлу информации о сервисах, поддерживаемых данным узлом
CD	Выбор каталога на удаленном узле
LCD	Выбор каталога на локальном узле
LIST	Запрос перечня файлов в текущем каталоге
PWD	Запрос текущего каталога
PORT	Установление соединения для передачи данных

CWD	Изменение текущего каталога
STORE	Сохранение файла на сервере
RETREIVE	Получение файла от сервера
и т.д.	

Существует два вида соединений: активное и пассивное. При активном установление соединения на передачу данных осуществляет клиент, а при пассивном – соединение с клиентом устанавливает сам сервер.Принцип установления соединения в этих случаях следующий.

Active connection – в этом случае клиент посылает серверу команду PASV. Сервер выбирает порт, через который будет происходить соединение и уведомляет об этом клиента. После чего сервер переходит в состояние прослушивания данного порта. Клиент инициирует соединение с указанным портом.

Passive connection – клиент посылает серверу команду PORT, в которой указывает свой IP-адрес и номер порта, через который будет устанавливаться соединение. Сервер, получив команду, инициирует установление соединения с клиентом.

С точки зрения безопасности, особенно, если клиент находится за межсетевым экраном или шлюзом, наиболее удобен первый вариант. В этом случае нет необходимости разрешать инициацию входящих соединений, что является одним из основных правил безопасности локальных сетей.

Рассмотренные ранее протоколы HTTP и FTP поддерживают работу через прокси-серверы. Прокси-сервер (proxy) – это узел, который получает запрос от клиента и транслирует его далее либо конечному узлу (серверу), либо следующему прокси в цепочке.

Схематически этот процесс изображен на рис. 5.7.

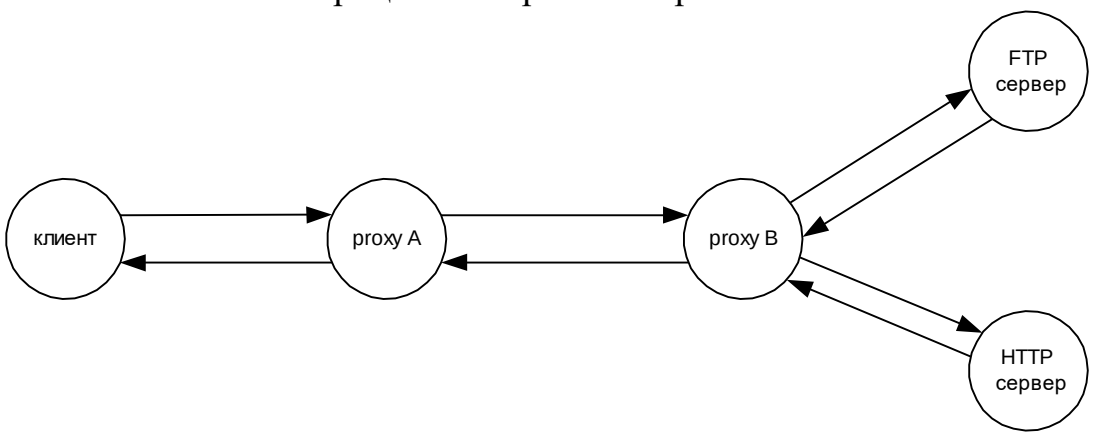


Рис. 5.7. Пример работы через прокси-сервера

Как видно из рисунка, клиент посылает свой запрос не напрямую, а промежуточному узлу A. Узел A перенаправляет свой запрос узлу B, а тот, в свою очередь, требуемому серверу. Аналогично ответ на запрос возвращается

через такую же цепочку. Использование такой сложной схемы может быть оправдано в следующих случаях:

- непосредственное соединение между клиентом и сервером осуществляется по медленной линии связи, а прокси-сервера предоставляют пакетам путь через быстрые линии связи. В этом случае наблюдается выигрыш в скорости доступа к серверу;
- клиенту необходимо обеспечить анонимность, т.е. скрыть свой адрес от сервера. В этом случае сервер увидит, что к нему обращаются с узла В, и может вообще не узнать о существовании клиента. Удлинение цепочки прокси-серверов уменьшает шанс определить истинного инициатора запроса;
- большинство прокси-серверов выполняют функцию кэширования, т.е. временного сохранения проходящей через них информации. В этом случае повторный запрос просто извлечет уже имеющиеся данные из кэша, и они не будут повторно запрашиваться с сервера. Все это повышает скорость доступа к ресурсам и снижает трафик (объем передаваемых данных). (Протокол HTTP имеет множественные встроенные средства для поддержки прокси-серверов и кэширования).

Проксирование применимо, в принципе, к любым протоколам, но чаще всего реализуется только для протоколов HTTP и FTP.

5.3. Почтовые протоколы

Электронная почта – это средство пересылки сообщений, содержащих произвольные данные от отправителя заданному адресату посредством глобальной компьютерной сети.

Идентификация адресата и отправителя осуществляется путем указания электронного адреса (e-mail). В общем случае формат электронного адреса следующий: **name@domain**.

name – имя пользователя;

domain – доменное имя, определяющее зону, в которой зарегистрирован данный пользователь.

Как видно из вышесказанного, функционирование электронной почты тесно связано с системой доменных имен.

При регистрации доменов в записи БД необходимо указывать поле MX, которое определяет обработчик почты, т.е. узел, на который будет перенаправляться почта для пользователей данной зоны.

Функционирование электронной почты в мировом масштабе обеспечивается сетью узлов, которые называются MTA (***Mail Transfer Agent***). MTA – это и есть те обработчики почты каждой зоны, указываемые в полях MX.

Существуют различные почтовые протоколы, используемые для пересылки данных между MTA, но одним из наиболее известных является протокол SMTP (***Simple Message Transfer Protocol***). Данный протокол обеспечивает установление

связи между двумя MTA, передачу данных (сообщений) MTA, обслуживающему требуемую зону, обработку ошибок и т.п.

Пересылку почтовых сообщений рассмотрим на примере. Пусть пользователь локальной сети некоторого предприятия отправляет письму адресату user@domen.spb.ru (рис. 5.8).

Как видно на рис. 5.8, отправитель формирует электронное письмо и передает его некоторому узлу MTA для дальнейшей пересылки. Для отправки пользователь должен быть зарегистрирован на данном MTA и ему должна быть разрешена отправка сообщений через данный MTA. В случае если сообщение принято MTA (в нашем примере это MTA предприятия), то дальнейшая судьба сообщения определяется с использованием службы DNS.

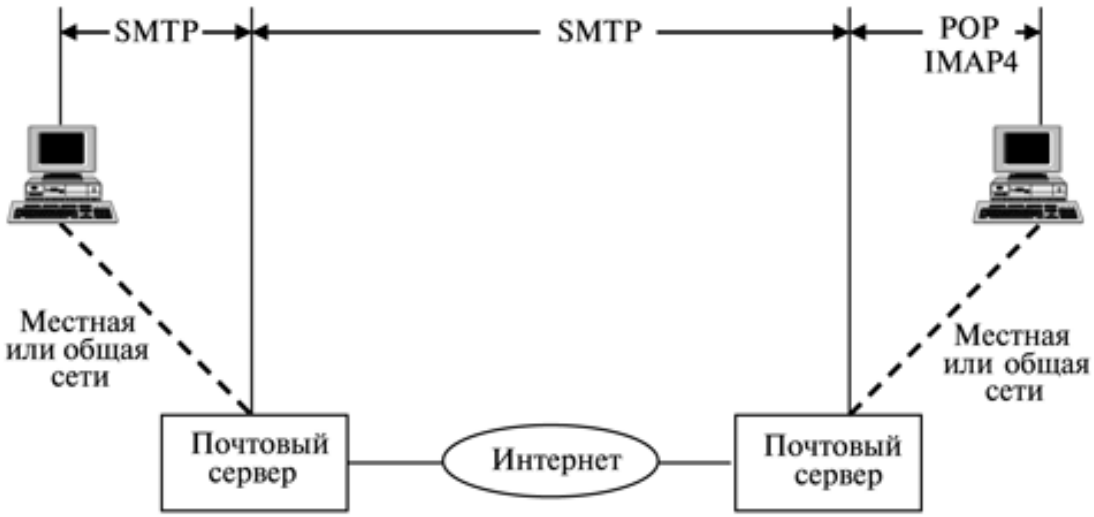


Рис. 5.8. Пример передачи электронного письма

MTA определяет адрес обработчика почты для доменной зоны, указанной в адресе получателя (domen.spb.ru). Процесс определения идет так же, как и при разрешении доменного имени. Поскольку старшие зоны **ru и spb.ru** своих обработчиков почты не имеют, то первым найденным MTA будет обработчик почты зоны domen.spb.ru. MTA отправителя установит соединение с MTA **mail.domen.spb.ru** и передаст ему сообщение.

Таким образом, можно сделать вывод, что протокол SMTP – это протокол надежной доставки сообщений между парой узлов MTA. Для доставки используется стандартный порт 25.

Само почтовое сообщение имеет обобщенный формат, показанный на рис. 5.9.

Рис. 5.9. Обобщенный формат почтового сообщения

Заголовок содержит различную служебную информацию, необходимую для доставки сообщения и возникающую в процессе прохождения сообщения через промежуточные узлы. Пример заголовка электронного письма приведен на рис. 5.10.

```
Return-Path: <user@pmi.habr.com>
Received: from domen.spb.ru ([unix socket])
by domen.spb.ru (Cyrus v2.1.16) with LMTP; Sat, 16 Apr 2015 08:16:47 +0300
Received: from elefant.habr.com (elefant.habr.com
[194.44.183.18])
by domen.spb.ru (8.12.10/8.12.10) with ESMTP id j3G5GkDT014963
for <user@domen.spb.ru>; Sat, 16 Apr 2015 08:16:46 +0300 (EEST)
 (envelope-from user@pmi.habr.com)
Received: from USER ([10.80.128.21])
by elefant.habr.com (8.12.9/8.11.3) with SMTP id j3G52MNt043068
for <user@domen.spb.ru>; Sat, 16 Apr 2015 08:02:24 +0300 (EEST)
 (envelope-from user@pmi.habr.com)
Message-ID: <001a01c54241$df422a50$1580500a@USER>
From: "USER" <user@pmi.habr.com>
To: "USER" <user@domen.spb.ru>
Subject: Hello
Date: Sat, 16 Apr 2015 08:05:12 +0300
MIME-Version: 1.0
Content-Type: text/plain;
charset="koi8-r"
```

Рис. 5.10. Пример заголовка электронного письма

Поле **Return-Path** содержит почтовый адрес, на который будет отправлено ответное сообщение, если пользователь выберет опцию «Ответить» в почтовом клиенте.

Блок параметров **Received** определяет, от кого было получено исходное сообщение, кто его переслал дальше и для кого. Данный блок добавляется каждым промежуточным MTA при пересылке сообщения.

В данном сообщении мы можем видеть, что непосредственно нами письмо получено от MTA **domen.spb.ru**. Данный MTA, в свою очередь, получил письмо от MTA **elefant.habr.com** для адресата **user@domen.spb.ru**.

В свою очередь MTA **elefant.habr.com** получил письмо от узла-отправителя USER (10.80.128.21) для адресата **user@domen.spb.ru**. Параметр **Message-ID** – это уникальный идентификатор сообщения.

From, **To** и **Subject** – это поля, которые будут отображаться в строках почтового клиента «От кого», «Кому», «Тема».

Date – дата отправки исходного сообщения.

MIME-Version -версия кодировки тела сообщения.

Content-Type – тип содержимого письма.

Для кодирования информации, передаваемой в теле письма, используется специальный формат MIME (***Multipurpose Internet Mail Extensions***), который стандартизирован для представления практически всех известных в мире форматов данных (текст, графика, аудио, видео, исполнимые файла и т.д.).

До этого мы говорили о пересылке почтового сообщения от MTA отправителя к MTA получателя. Кроме этого существует задача получения сообщения почтовым клиентом пользователя от узла, на котором расположен почтовый ящик.

Наиболее известными протоколами для решения данной задачи являются протоколы POP3 (***Post Office Protocol***) и IMAP (***Internet Mail Access Protocol***). Данные протоколы реализуют набор команд, которые позволяют почтовому клиенту принимать сообщения, анализировать содержимое почтового ящика, удалять сообщения и т.д.

Стандартными для работы протоколов POP3 и IMAP являются порты 110 и 143, соответственно.

5.4. Протоколы эмуляции удаленного терминала

При администрировании и управлении сетями часто возникает необходимость в удаленной работе с некоторым узлом (сервером). Т.е. пользователь желает работать с компьютером через сеть так, как если бы он находился за монитором компьютера. Традиционно устройство для интерактивной работы с ЭВМ называют терминалами, а устройство взаимодействия, которое удалено от компьютера и подключено к нему через сеть, называют удаленным терминалом.

Рассматриваемые в данном разделе протоколы реализуют эмуляцию удаленного терминала, т.е. создают перед пользователем копию интерфейса (содержимого экрана) удаленного сервера и позволяют вводить команды, передавать их на сервер и отображать результат их выполнения. Т.е. у пользователя возникает иллюзия, что он работает непосредственно за компьютером, от которого он удален на значительное расстояние.

Типовыми представителями данного класса протоколов являются протоколы: TELNET, SSH, X11, Rlogin, RAW и т.д.

На рис. 5.11 приведен пример окна эмуляции удаленного терминала при работе с сервером под ОС Linux. На рисунке видно, как в окне на экране компьютера создается копия содержимого экрана удаленного компьютера (видна оболочка **midnight commander**).

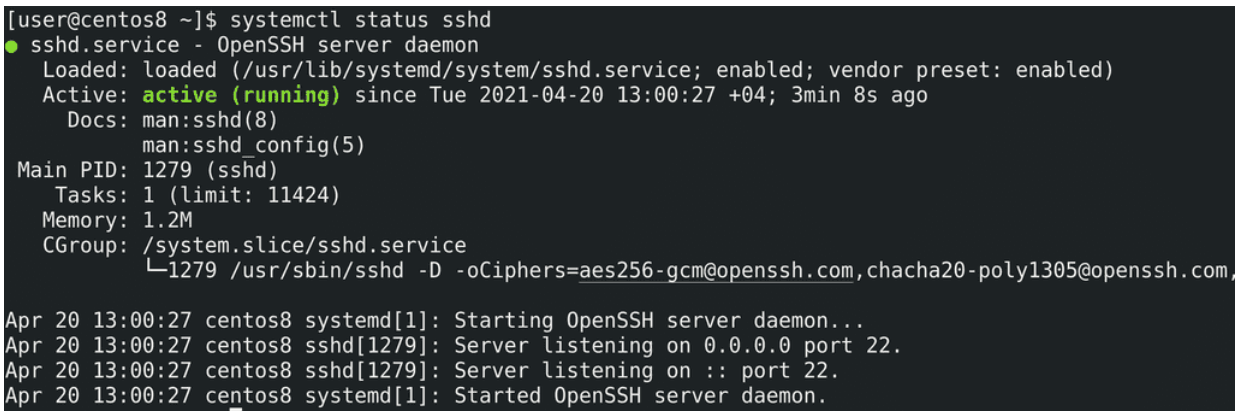


Рис. 5.11. Пример эмуляции удаленного терминала

Протокол TELNET (порт 23) – простейший протокол передачи управляющих команд и реакции на их выполнение между TELNET-клиентоми TELNET-сервером. Данные, в соответствии с этим протоколом, передаются без какого либо кодирования (шифрования). В результате протокол обладает такими преимуществами, как простота реализации и высокая скорость работы, но в то же время таким важным недостатком, как низкая защищенность. Данное свойство особенно важно для протоколов этого класса, так как они предназначены для удаленного управления, и перехват данных позволит выполнять злоумышленнику несанкционированные действия с удаленным узлом.

Более совершенным является протокол SSH (***Secure SHell***) – порт 22. Данный протокол интенсивно использует шифрование данных, создание шифрованных туннелей для передачи данных, развитые средства аутентификации и другие средства повышения безопасности.

Детальное изучение протоколов данного класса выходит за рамки данного курса и может быть выполнено с использованием дополнительной литературы.

5.5. Прочие протоколы прикладного уровня

Как уже говорилось ранее, количество протоколов прикладного уровня огромно и соответствует числу разнообразнейших прикладных задач, которые требуют сетевого взаимодействия.

Ниже приведен краткий перечень наиболее известных протоколов прикладного уровня. Более полную информацию можно получить в приложении 1 или в дополнительной литературе.

SNMP (***Simple Network Management Protocol***) – протокол удаленного управления сетевым оборудованием;

NFS (***Network File System***), AFS (***Andrew File System***) – протоколы реализации распределенных сетевых файловых систем;

SMB (***Server Message Block***) – протокол пересылки серверных сообщений, используется для реализации общего сетевого доступа к файлам и принтерам;

RPC (***Remote Procedure Call***) – протокол удаленного вызова процедур;

RTSP (***Real Time Stream Protocol***) – протокол передачи потоковых данных в реальном времени;

NTP (***Network Time Protocol***) – протокол сетевой синхронизации времени;

RADIUS, KERBEROS – протоколы аутентификации и безопасного доступа к данным;

FINGER – протокол определения данных о пользователях в сети;

ICQ, EDONKEY – протоколы прикладных программ (интернет-пейджера ICQ и файлообменной сети eDonkey);

и т.д.

Вопросы для самопроверки

1. Описать прикладной уровень и уровень представления данных модели передачи данных OSI
2. Перечислить протоколы прикладного уровня
3. Привести пример передачи электронного письма
4. Раскрыть связь между параметрами работы протоколов POP3 и IMAP
5. Привести пример эмуляции удаленного терминала

6. ПЕРЕДАЧА ДАННЫХ НА ФИЗИЧЕСКОМ УРОВНЕ

6.1. Основы передачи данных на физическом уровне

На физическом уровне модели OSI передаваемые данные представлены в виде электрических или иных сигналов. В простейшем случае представление информации сводится к тому, что единичному значению двоичной цифры в соответствие ставится один уровень напряжения (тока), а нулевому значению – другой уровень. Любая информация, передаваемая через линию связи, может быть представлена как поток битов (двоичных цифр) или как последовательность различных уровней напряжения (тока). Например, пусть лог. 1 соответствует напряжение +5В, а лог. 0 – напряжение –5В, тогда передача последовательности 110101001 может быть представлена в виде графика на рис. 6.1.

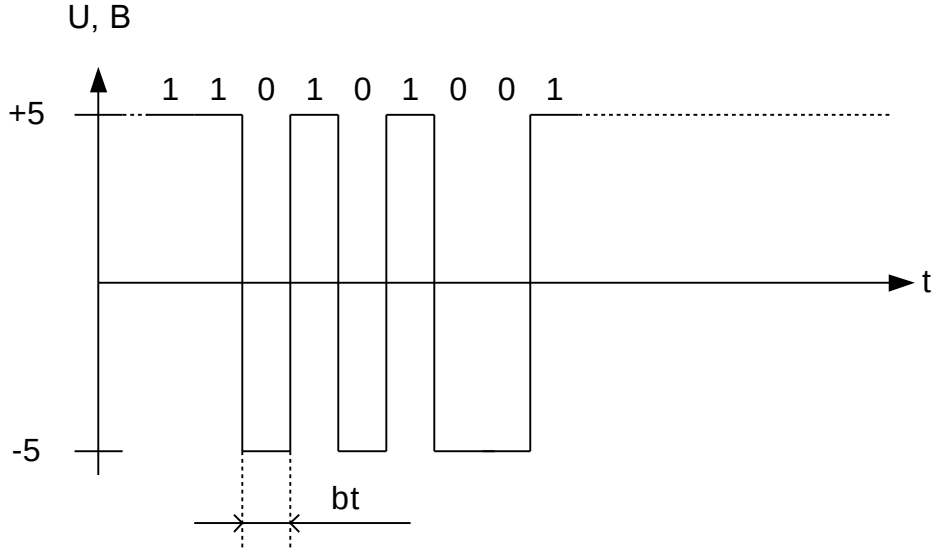


Рис. 6.1. Пример передачи данных в виде электрического сигнала

На рисунке видно, что битовая последовательность представлена как последовательность сменяющихся значений напряжений. Вообще передача данных от одного устройства другому может происходить параллельным или последовательным способом. При параллельном способе одновременно по нескольким параллельным проводникам передается несколько бит информации (байт, слово и т.д.). При последовательном способе проводник один и по нему последовательно (один за другим) передаются информационные биты [11].

Достоинство параллельного способа – высокая скорость передачи данных, недостатки – большое количество проводников и невозможность передавать данные на значительные расстояния. Последнее связано с тем, что при увеличении длины проводника искажения, вносимые им в передаваемые данные,

увеличиваются, причем для каждого проводника по-разному. В результате на дальних расстояниях происходит рассогласование линий связи, и данные искажаются до неузнаваемости.

Поэтому в компьютерных сетях (глобальных и локальных) практически всегда применяется только последовательный способ передачи данных. В свою очередь, последовательным способом передача может идти асинхронно или синхронно. Наличие двух вариантов передачи обусловлено существованием проблемы выделения отдельных битов на приемнике при их получении. Т.е. как узнать, что передача одного бита закончилась, и началась передача другого. При асинхронном методе используется схема, представленная на рис. 6.2.

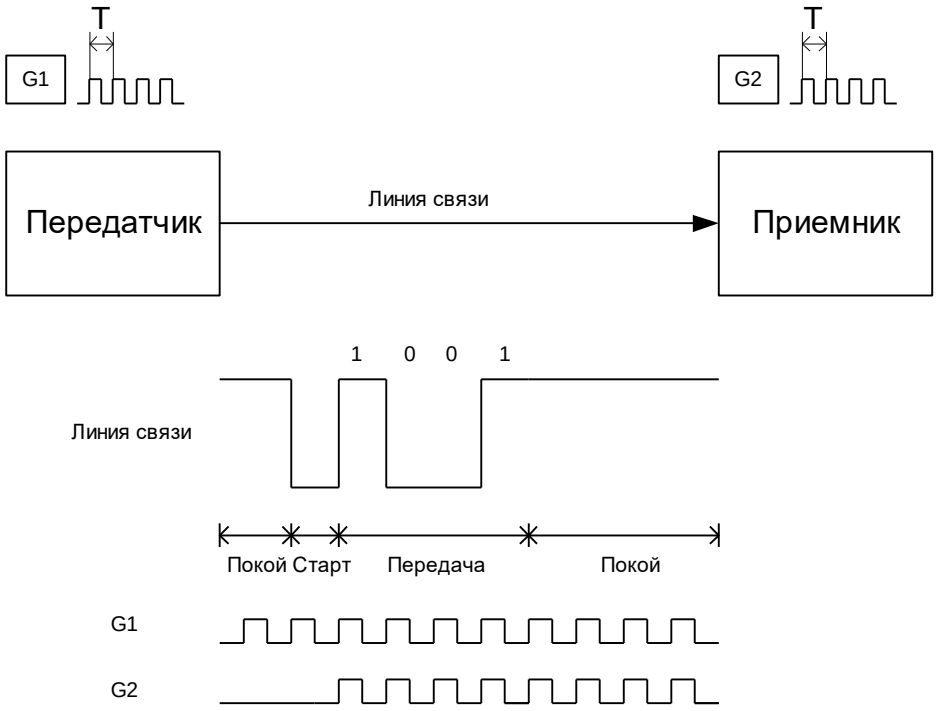


Рис. 6.2. Асинхронный метод передачи данных

Поскольку при асинхронном методе передатчик и приемник связаны только одной линией связи, то необходим механизм, который бы позволял выявить начало передачи, определить моменты передачи каждого бита и выявить конец передачи. Для этого одно из состояний линий связи (на рисунке единичное) выбирается как состояние покоя.

При начале передачи передатчик запускает генератор синхронизации G1. Каждый такт данного генератора определяет время передачи одного бита T. Первый же переход линии связи из состояния покоя в противоположное является признаком начала передачи (так называемым стартовым битом). При появлении стартового бита приемник запускает свой генератор G2, параметры которого идентичны генератору передатчика. Далее, каждый интервал T будет означать интервал передачи одного бита и для передатчика и для приемника. Факт

окончания передачи чаще всего определяется по заранее известному размеру передаваемого пакета битов. После чего генераторы останавливаются, и линия связи переводится в состояние покоя.

Данный метод имеет такое преимущество, как минимальное число линий связи, и такие недостатки, как усложненный алгоритм работы и низкую максимально достижимую скорость передачи из-за влияния на больших скоростях несогласованности автономных генераторов передатчика и приемника.

При синхронном методе используется схема, представленная на рис. 6.3. На рисунке видно, что появляется вторая линия связи – для передачи сигналов синхронизации. В этом случае передатчик непосредственно управляет приемом каждого бита на приемнике через сигнал синхронизации. Соответственно снимаются вопросы выявления факта начала и конца передачи и вопрос рассогласования параметров генераторов. При синхронном методе передача длится столько, сколько присутствует сигнал синхронизации. Рассогласование же невозможно, так как генератор один и для приемника и для передатчика.

Соответственно, данный метод позволяет обеспечить более высокую скорость передачи, чем асинхронный метод и упростить алгоритм работы.

Главный недостаток метода – наличие дополнительной линии связи. Это очень серьезный недостаток для сетей, в которых используются сотни линий связи и наличие для каждой линии дополнительного проводника значительно усложняет создание и обслуживание линии связи, а также увеличивает ее стоимость. Кроме того, на значительных расстояниях две линии (данные и синхронизация) могут значительно рассогласовываться, что вносит дополнительные проблемы в построение синхронных линий связи значительной протяженности.

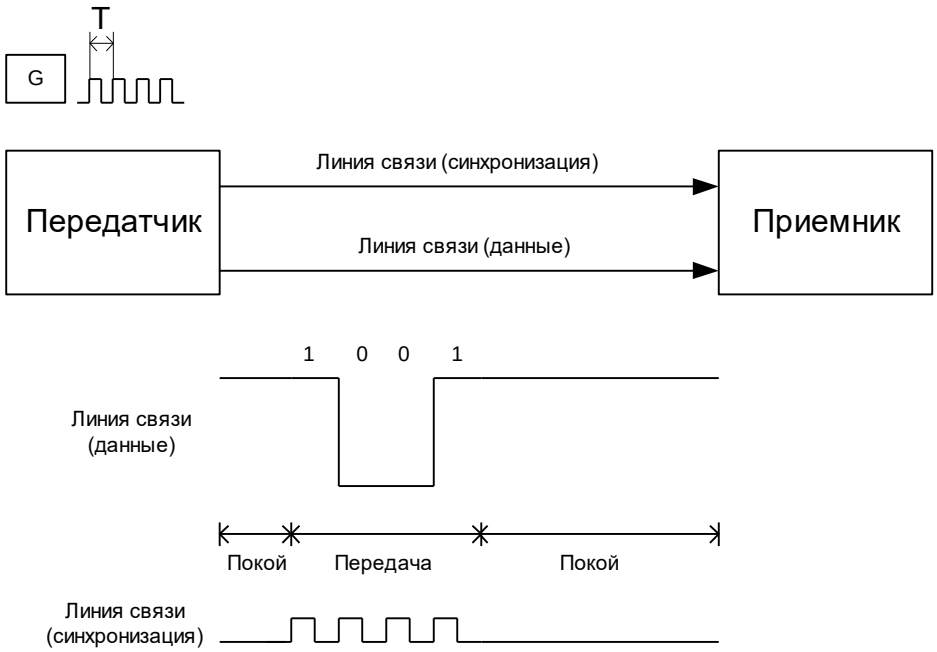


Рис. 6.3. Синхронный метод передачи данных

Выход из этой ситуации – создание самосинхронизирующихся кодов, т.е. таких способов передачи данных, при которых сигнал синхронизации и данные смешиваются определенным образом и передаются по одной линии связи. Такое решение усложняет алгоритм работы устройств, но позволяет реализовать достоинства синхронного метода передачи с использованием только одной линии связи.

Наиболее известным самосинхронизирующимся кодом в сетях является манчестерский код.

В представленных выше примерах предполагалось, что данные по линии связи передаются «как есть», т.е. без дополнительного кодирования (кроме манчестерского кода). Такое простое представление данных при передаче по линии связи имеет множество недостатков: длинные последовательности одинаковых битов вызывают появление так называемой постоянной составляющей, которая отрицательно влияет на работу приемо-передающих устройств, непосредственное кодирование имеет также низкую помехоустойчивость и чувствительно к ошибкам при передаче, а также не обладает свойством самосинхронизации.

Поэтому на практике применяют множество способов кодирования информации перед ее преобразованием в электрический сигнал. Вид кодирования (физическое, логическое) зависти от используемой физической среды передачи данных, от технологии канального уровня и других факторов. Ниже приведен краткий перечень наиболее известных методов кодирования:

NRZ (***Non Return Zero***);

NRZI (***Non Return Zero Inverse***);

AMI (***Alternate Mark Inversion***);

2B1Q (два бита передаются за один такт сигналом, который имеет четыре состояния);

4B/5B (исходные символы длиной 4 бита заменяются избыточным кодом длиной 5 бит).

6.2. Основные характеристики линий связи

Каждая линия связи, по которой передаются данные, характеризуется некоторым набором параметров. Для уяснения сущности этих параметров рассмотрим несколько теоретических вопросов.

Как известно, любой сигнал можно представить в виде суммы синусоид. Каждая такая синусоида называется гармоникой. В зависимости от амплитуды гармоники она вносит больший или меньший вклад в общий сигнал.

Если бы проводник сигнала был идеальным, то сигнал бы распространялся по нему без каких-либо искажений. Но на практике каждый проводник обладает сопротивлением, индуктивностью и емкостью, которые распределены по всей его

длине. Значения данных величин определяются типом материала проводника, его геометрическими размерами и т.д.

Так как индуктивность, емкость и сопротивление оказывают различное влияние на сигналы различных частот, то для каждой линии связи определяется апмлитудно-частотная характеристика (АЧХ). АЧХ показывает, как ослабляются сигналы различных частот в данном проводнике. При этом определяется некоторый порог ослабления. Непрерывный диапазон частот, для которых ослабление не превышает заданный порог, называется полосой пропускания линии связи.

С шириной полосы пропускания тесно связано понятие пропускной способности линии связи. Пропускная способность – это количество единиц информации, передаваемых за единицу времени. Чаще всего измеряется в бит/с, Кбит/с, Мбит/с и т.д.

Если через линию связи передавать периодический сигнал (синусоиду, меандр и др.), то такая передача будет иметь нулевую информативность, т.к. изменение сигнала является легко предсказуемым и бесконечно повторяющимся. Такой сигнал часто называют несущим.

Информативность присуща непредсказуемому изменению несущего сигнала. Если сигнал имеет два фиксированных значения, то любое изменение передает один бит информации, если более двух – то больше.

Скорость передачи данных будет тем выше, чем чаще происходят изменения состояния линии связи, т.е. чем более высокую частоту имеет несущий сигнал. Так как сигналы в цифровых линиях связи представлены преимущественно прямоугольными импульсами, то для их передачи необходимо, чтобы линия связи пропускала все гармоники от высших до низших. Это связано с тем, что высшие и низшие гармоники влияют на качество фронтов импульса, а соответственно и на его распознаваемость на противоположном конце линии связи. При этом увеличение частоты вызывает увеличение ширины требуемой полосы пропускания. Соответственно, при ограниченной (фиксированной) полосе пропускания имеется ограничение на максимальную частоту несущего сигнала и частоту изменения состояния линии связи.

Соотношение между вышеуказанными величинами описывается формулой Шеннона:

$$C = F * \log_2(1 + \frac{P_c}{P_{Ш}})$$

где

C – максимальная пропускная способность линии связи;

F – полоса пропускания линии связи;

Рс – мощность источника сигнала;

Рш – мощность шума в линии связи.

Обобщение всего вышесказанного можно увидеть на рис. 6.4. Пусть даны АЧХ двух линий связи. Эти линии имеют полосы пропускания F1 и F2 соответственно. Пусть по линии передается дискретный сигнал произвольной формы. Спектральное разложение данного сигнала на гармоники показано на рисунке. Можно видеть, что полоса пропускания F1 позволяет передавать без искажений все гармоники сигнала, поэтому на другом конце линии связи мы получим практически неискаженный сигнал (а). В свою очередь, полоса пропускания F2 не пропускает («срезает») верхние гармоники сигнала. Это приводит к тому, что на выходе мы имеем искаженный сигнал (б).

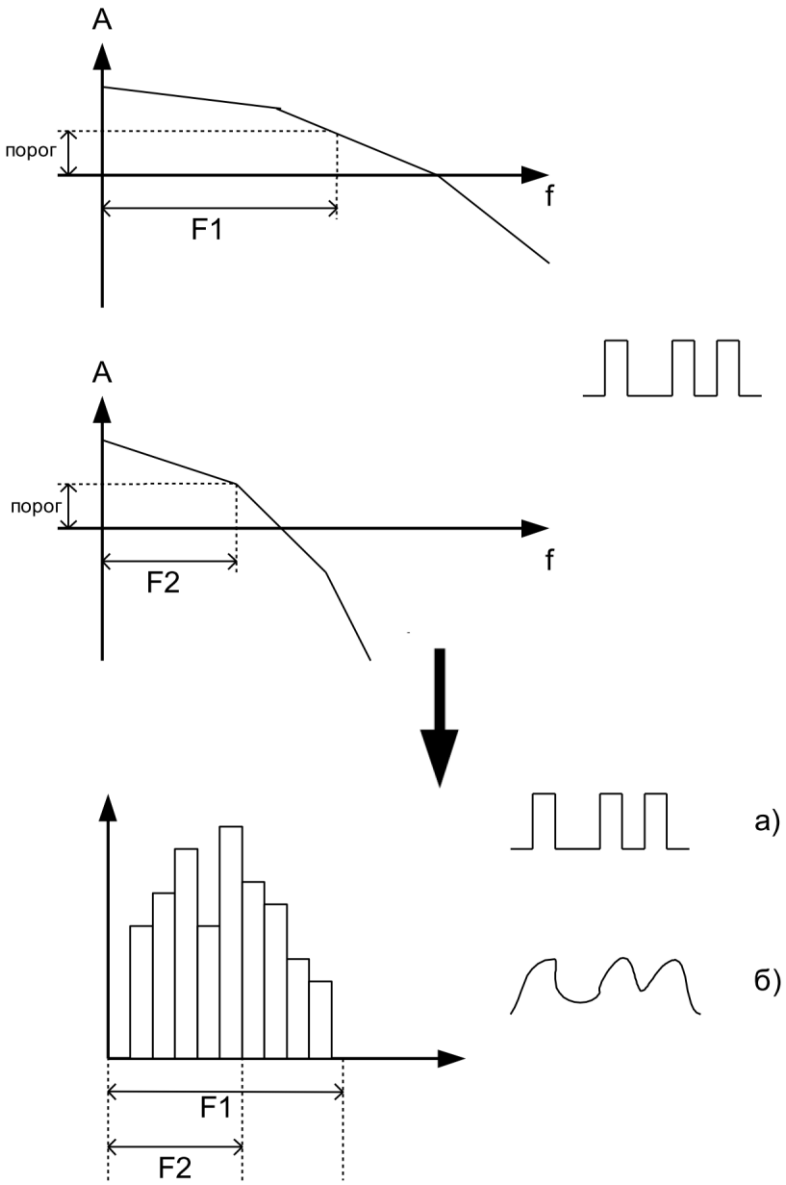


Рис. 6.4. Иллюстрация к вопросу о связи полосы пропускания и пропускной способности

Максимальная пропускная способность также зависит от способа кодирования. Если сигнал может иметь два устойчивых состояния, то пропускная способность в два раза превышает полосу пропускания, если четыре состояния – то в четыре раза. Данное соотношение описывает формула Найквиста:

$$C = 2F * \log_2 M$$

где
C – пропускная способность;
F – ширина полосы пропускания;
M – количество состояний линии связи.

Следует помнить, что при последовательном соединении линий связи с различной пропускной способностью максимальная пропускная способность всей линии будет определяться пропускной способностью самого медленного участка. В то же время на отдельных участках пропускная способность может быть выше.

6.3. Физический уровень глобальных сетей

Как говорилось в лекции 1, основу построения глобальных компьютерных сетей составляют телекоммуникационные линии связи (телефонные сети и т.д.). При реализации передачи данных между любыми двумя узлами возникает следующая проблема. Каждое соединение требует существования между узлами автономной линии связи (проводника или пары проводников). Увеличение числа одновременно устанавливаемых соединений пропорционально увеличивает число проводников в линиях связи. Уже для 1000 одновременных соединений потребуется линия связи в несколько тысяч проводников. Естественно, на практике такое решение реализовать невозможно. Поэтому в свое время были разработаны различные способы мультиплексирования сигнала (одновременной передачи по одной линии связи сигналов различных соединений).

Общий принцип мультиплексирования показан на рис. 6.5.

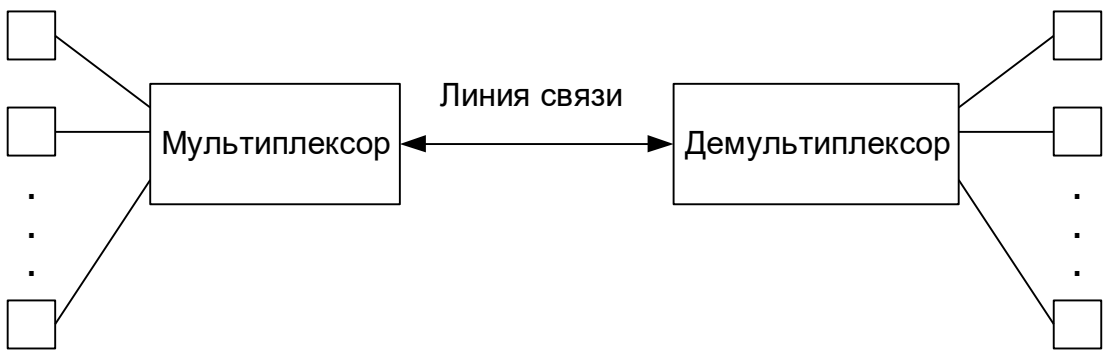


Рис. 6.5. Общий принцип мультиплексирования данных

Фактически данные смешиваются некоторым образом и передаются по одной линии связи, а на другой стороне происходит разделение данных на столько потоков, сколько было на противоположной стороне до мультиплексирования.

Существует три основных метода мультиплексирования: FDM, TDM и WDM.

Метод FDM (***Frequency Division Modulation***) – передача с разделением по частоте. Данный метод широко использовался для мультиплексирования аналоговых сигналов в аналоговых линиях связи. Его принцип схож с классической модуляцией в радиоприеме. Схема данного метода приведена на рис. 6.6.

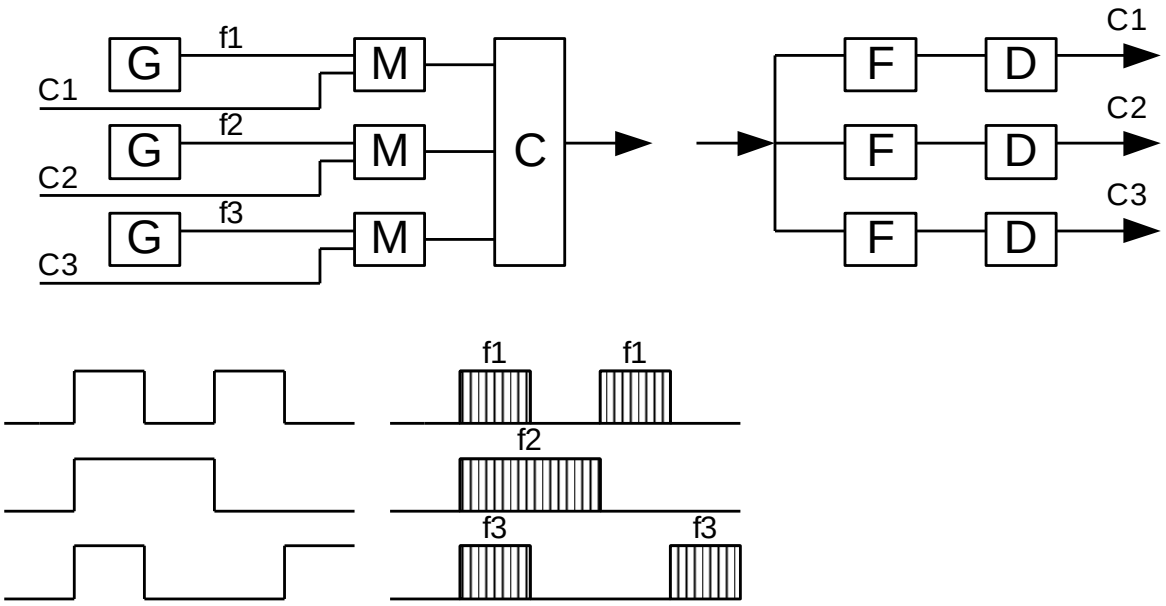


Рис.6.6. Принцип разделения по частоте

Мультиплексор в этом случае состоит из генераторов опорной частоты (G), модуляторов (M), смесителя (C). Количество генераторов соответствует количеству информационных сигналов C1-C3. Каждый генератор вырабатывает свой сигнал несущей частоты f1-f3. На модуляторах происходит модуляция каждого опорного сигнала своим информационным сигналом. Информационные сигналы и результат модуляции показан на нижней части рисунка. Далее полученные сигналы смешиваются смесителем и передаются в линию связи. Проблемы с одновременной передачей не возникает, т.к. каждый сигнал имеет свою частоту.

Демультиплексор состоит из фильтров (F) и детекторов (D). Каждый фильтр настроен на свою несущую частоту и выполняет функцию выделения одного сигнала из общего потока. Детектор выполняет выделение из сигнала огибающей, т.е. первоначальной формы информационного сигнала.

Данный метод легко применим к аналоговым сигналам, но при использовании дискретных сигналов возникают определенные трудности. Кроме того, метод требует сложной аппаратуры для реализации, обеспечивает малую степень уплотнения данных и предъявляет высокие требования к точностным параметрам составных элементов.

Поскольку все современные телекоммуникационные линии связи предназначены для передачи дискретных (цифровых) сигналов, то наиболее распространенным является другой метод – TDM (***Time Division Modulation***) – разделение по времени. Принцип работы данного метода показан на рис. 6.7.

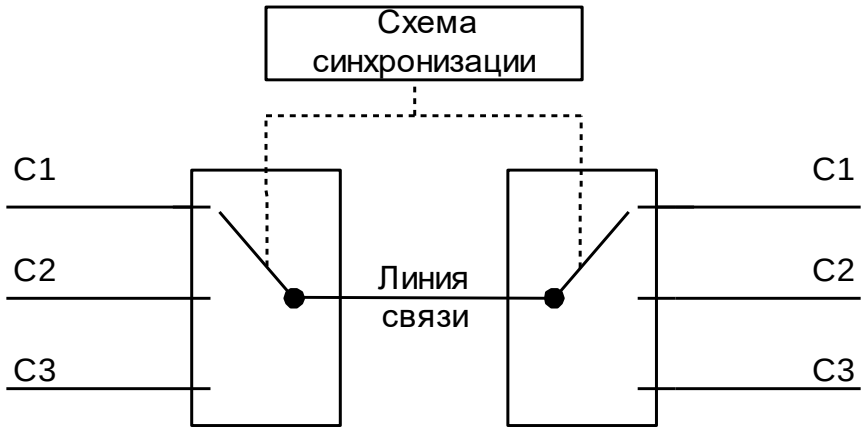


Рис. 6.7. Принцип разделения по времени

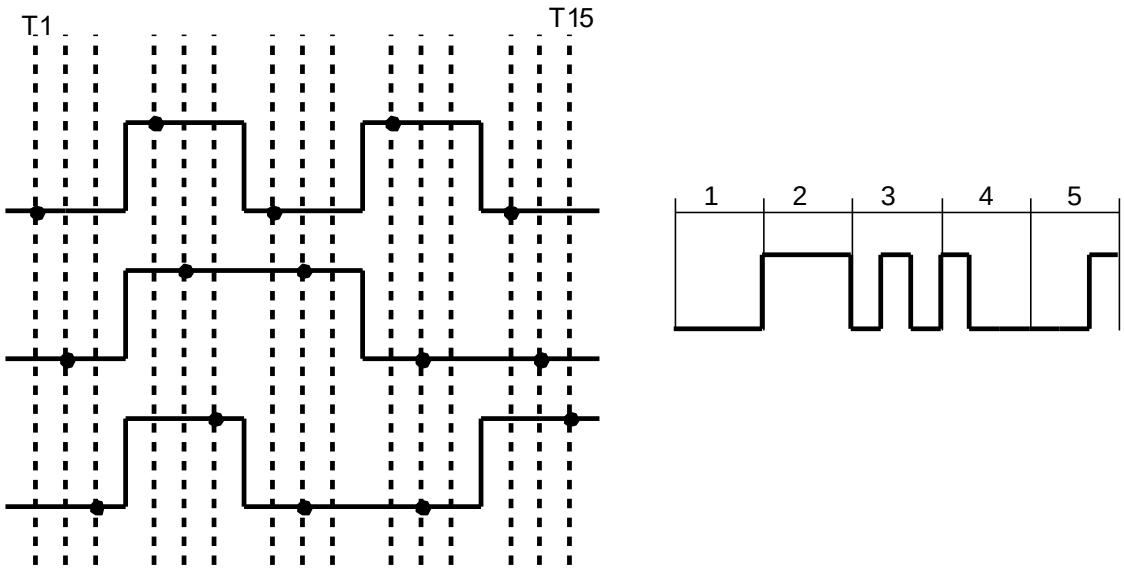


Рис. 6.8. Диаграмма разделения по времени

На рис. 6.8 видно, что фактически несколько линий данный поочередно коммутируются на одну линию связи с достаточно высокой частотой. В этом

случае на линии связи периодически появляются значения то сигнала С1, то сигнал С2, то сигнала С3. Такие значения называются выборками. В нашем случае на диаграмме видно 5 групп по 3 выборки в каждой. Если на стороне демультиплексора коммутация линии связи на приемники будет осуществляться синхронно с мультиплексором, то через линию связи циклически поочередно будут передаваться значения всех сигналов в виде выборок. Если частота коммутации достаточно высокая (выше, чем максимальная частота информационных сигналов), то для приемника и передатчика процесс коммутации становится «прозрачным», т.е. мы получаем возможность передать через одну линию связи несколько сигналов одновременно.

Данный метод обеспечивает высокую степень уплотнения, легко применим к цифровым сигналам, прост в реализации. Недостатки метода – необходимость стабильной и надежной синхронизации процесса мультиплексирования и демультиплексирования, а также необходимость применения быстродействующих коммутаторов для мультиплексирования сигналов высоких частот.

Третий метод WDM (***Wave Division Modulation***) – с разделением по волне специфичен для оптоволоконных линий связи.

Оптическое волокно представляет собой проводник-световод, изготовленный из специального полимера. Носителем информации в световоде является световой луч (рис. 6.9). Длина волны излучения может быть различной и находиться как в видимой, так и в невидимой части диапазона. Особенностью оптоволоконных линий связи является высокая пропускная способность, обусловленная малым затуханием и искажением сигнала при распространении по проводнику, а также нечувствительность к электрическим помехам.

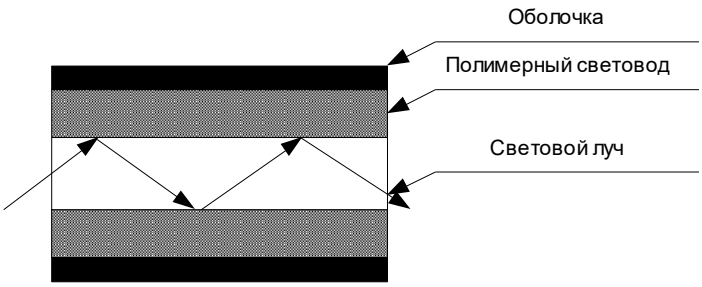


Рис. 6.9. Передача информации по оптоволокну

Каждый луч в световоде характеризуется длиной волны излучения λ. Посылая в световод лучи с разной длиной волны, можно одновременно без помех друг другу передавать несколько сигналов в одной линии связи. На этом и основано мультиплексирование по длине волны.

6.4. Иерархия цифровых потоков

В зависимости от вида передаваемых данных к пропускной способности линий связи предъявляются различные требования. В связи с интенсивным ростом объема передаваемых данных и увеличением числа абонентов глобальных телекоммуникационных сетей возникла необходимость в стандартизации типов цифровых потоков по скоростям обмена и построении универсальной схемы слияния и разделения таких потоков. Для решения возникшей проблемы в 1970-х годах была разработана первая иерархия цифровых потоков – PDH (***Plesiochronous Data Hierarchy***). Данная иерархия называлась плезиохронной, т.е. псевдосинхронной. В соответствии с данной иерархией цифровые потоки можно было разделять между пользователями в зависимости от их потребностей, сливать воедино, выделять на демультиплексорах и обрабатывать произвольным образом. По цифровым потокам можно было передавать любую информацию (данные, звук, видео и т.д.).

Существует два стандарта PDH: американский ANSI и международный (европейский) CCITT. Иерархия цифровых потоков PDH приведена в табл. 6.1.

Для обозначения наиболее часто используемых потоков применяют обозначения T1-T3 в американском стандарте и E1-E3 – в европейском. Как видно из таблицы, минимальным цифровым потоком является поток 64Кбит/с. 24 минимальных потока образуют поток T1, 4 потока T1 образуют T2, 7 потоков T2 образуют поток T3 и т.д.

Табл. 6.1. Иерархия потоков PDH

Обозначение	ANSI			CCITT		
	Кол-во голосовых каналов	Кол-во каналов пред. уровня	Скорость, Мбит/с	Кол-во голосовых каналов	Кол-во каналов пред. уровня	Скорость, Мбит/с
DS0	1	1	64 Кбит/с	1	1	64 Кбит/с
DS1 (T1/E1)	24	24	1.544	30	30	2.048
DS2 (T2/E2)	96	4	6.312	120	4	8.488
DS3 (T3/E3)	672	7	44.736	480	4	34.368
DS4	4032	6	274.176	1920	4	139.264

Недостатки иерархии PDH:

- невозможность выделения низкоскоростного потока из высокоскоростного без его полного демультиплексирования. Например, чтобы

выделить поток DS0 из T3 необходимо демультиплексировать поток до уровня T2, затем до T1 и только потом выделить требуемый поток;

- отсутствие развитых средств управления сетью, использующей данную иерархию, отсутствие средств обеспечения надежности и т.д.
- низкие по современным меркам максимальные пропускные способности, поддерживаемые иерархией.

Для преодоления вышеуказанных недостатков была разработана иерархия SDH (***Synchronous Data Hierarchy***). Первоначально американский стандарт данного вида иерархии получил название SONET (***SynchronousOptical NET***). Основные особенности иерархии SDH следующие:

- гибкая схема мультиплексирования, которая обеспечивает возможность выделения произвольного стандартного потока данных без демультиплексированиявсего информационного потока;
- встроенные средства мониторинга, управления и обеспечения отказоустойчивости;
- высокое качество транспортного обслуживания любого вида трафика с обеспечением гарантированной пропускной способности и минимальных задержек.

В табл. 6.2 приведена иерархия цифровых потоков SDH.

Табл. 6.2. Иерархия потоков SDH

SDH	SONET	Скорость
	STS-1, OC-1	51.84 Мбит/с
STM-1	STS-3, OC-3	155.52 Мбит/с
STM-3	OC-9	466.56 Мбит/с
STM-4	OC-12	622.08 Мбит/с
STM-6	OC-18	933.120 Мбит/с
STM-8	OC-24	1.244 Гбит/с
STM-12	OC-36	1.866 Гбит/с
STM-16	OC-48	2.488 Гбит/с
STM-64	OC-192	9.953 Гбит/с
STM-256	OC-768	39.81 Гбит/с

Вопросы для самопроверки

1. Описать процесс передачи информации по оптоволокну
2. Перечислить способы передачи данных на физическом уровне
3. Привести пример диаграммы разделения по времени
4. Раскрыть недостатки иерархии цифровых потоков

ПРИЛОЖЕНИЕ 1. ПЕРЕЧЕНЬ СЕТЕВЫХ ПРОТОКОЛОВ

Наименование	Расшифровка
3GPP2 A11	3GPP2 A11
802.11 MGT	IEEE 802.11 wireless LAN management frame
802.11 Radiotap	IEEE 802.11 Radiotap Capture header
AAL1	ATM AAL1
AAL3/4	ATM AAL3/4
AARP	Appletalk Address Resolution Protocol
ACAP	Application Configuration Access Protocol
ACN	ACN
ACSE	OSI ISO/IEC 10035-1 ACSE Protocol
AFP	AppleTalk Filing Protocol
AFS (RX)	Andrew File System (AFS)
AH	Authentication Header
AIM	AOL Instant Messenger
AIM Administration	AIM Administrative
AIM Advertisements	AIM Advertisements
AIM BOS	AIM Privacy Management Service
AIM Buddylist	AIM Buddylist Service
AIM Chat	AIM Chat Service
AIM ChatNav	AIM Chat Navigation
AIM Directory	AIM Directory Search
AIM Email	AIM E-mail
AIM Generic	AIM Generic Service
AIM ICQ	AIM ICQ
AIM Invitation	AIM Invitation Service
AIM Location	AIM Location
AIM Messaging	AIM Messaging
AIM OFT	AIM OFT
AIM Popup	AIM Popup
AIM Signon	AIM Signon
AIM SSI	AIM Server Side Info
AIM SST	AIM Server Side Themes
AIM Stats	AIM Statistics
AIM Translate	AIM Translate
AIM User Lookup	AIM User Lookup
AJP13	Apache JServ Protocol v1.3
ALC	Asynchronous Layered Coding

ALCAP	AAL type 2 signalling protocol – Capability set 1 (Q.2630.1)
AMR	Adaptive Multi-Rate
ANS	Intel ANS probe
ANSI BSMAP	ANSI A-I/F BSMAP
ANSI DTAP	ANSI A-I/F DTAP
ANSI IS-637-A Teleservice	ANSI IS-637-A (SMS) Teleservice Layer
ANSI IS-637-A Transport	ANSI IS-637-A (SMS) Transport Layer
ANSI IS-683-A (OTA (Mobile))	ANSI IS-683-A (OTA (Mobile))
ANSI IS-801 (Location Services (PLD))	ANSI IS-801 (Location Services (PLD))
ANSI MAP	ANSI Mobile Application Part
AODV	Ad hoc On-demand Distance Vector Routing Protocol
AOE	ATAoverEthernet
ARCNET	ARCNET
ARP/RARP	Address Resolution Protocol
ARTNET	Art-Net
ASAP	Aggregate Server Access Protocol
ASF	Alert Standard Forum
ASN1	ASN.1 decoding
ASP	AppleTalk Session Protocol
ATM	ATM
ATM LANE	ATM LAN Emulation
ATP	AppleTalk Transaction Protocol packet
ATSVC	Microsoft Task Scheduler Service
Auto-RP	Cisco Auto-RP
AVS WLANCAP	AVS WLAN Capture header
AX4000	AX/4000 Test Block
BACapp	Building Automation and Control Network APDU
BACnet	Building Automation and Control Network NPDU
BEEP	Blocks Extensible Exchange Protocol
BER	Basic Encoding Rules (ASN.1 X.690)
BFD Control	Bi-directional Fault Detection Control Message
BGP	Border Gateway Protocol
BICC	Bearer Independent Call Control
BitTorrent	BitTorrent

Boardwalk	Boardwalk
BOFL	Wellfleet Breath of Life
BOOTP/DHCP	Bootstrap Protocol
BOOTPARAMS	Boot Parameters
BOSSVR	DCE/RPC BOS Server
BROWSER	Microsoft Windows Browser Protocol
BSSAP	BSSAP/BSAP
BSSGP	Base Station Subsystem GPRS Protocol
BUDB	DCE/RPC BUDB
BUTC	DCE/RPC BUTC
BVLC	BACnet Virtual Link Control
CAST	Cast Client Control Protocol
CBAPDev	CBAPhysicalDevice
CCSDS	CCSDS
CDP	Cisco Discovery Protocol
CDS_CLERK	CDS Clerk Server Calls
CFLOW	Cisco NetFlow
CGMP	Cisco Group Management Protocol
CHDLC	Cisco HDLC
CIP	Common Industrial Protocol
CLDAP	Connectionless Lightweight Directory Access Protocol
CLEARCASE	Clearcase NFS
CLNP	ISO 8473 CLNP ConnectionLess Network Protocol
CLTP	ISO 8602 CLTP ConnectionLess Transport Protocol
CMIP	X711 CMIP
CMP	Certificate Management Protocol
CMS	Cryptographic Message Syntax
CONV	DCE/RPC Conversation Manager
COPS	Common Open Policy Service
COSEVENTCOMM	Coseventcomm Dissector Using GIOP API
CoSine	CoSine IPNOS L2 debug output
COSNAMING	Cosnaming Dissector Using GIOP API
COTP	ISO 8073 COTP Connection-Oriented Transport Protocol
CPFI	Cross Point Frame Injector
CPHA	Check Point High Availability Protocol
CRMF	Certificate Request Message Format
CUPS	Common Unix Printing System (CUPS) Browsing Protocol

DAAP	Digital Audio Access Protocol
Data	Data
DCCP	Distributed Checksum Clearinghouse Protocol
DCE_DFS	DFS Calls
DCERPC	DCE RPC
DCOM	DCOM
DDP	Datagram Delivery Protocol
DDTP	Dynamic DNS Tools Protocol
DEC_STP	DEC Spanning Tree Protocol
DFS	Microsoft Distributed File System
DHCPFO	DHCP Failover
DHCPv6	DHCPv6
Diameter	Diameter Protocol
dicom	DICOM
DISTCC	Distcc Distributed Compiler
DLSw	Data Link SWitching
DNP 3.0	Distributed Network Protocol 3.0
DNS	Domain Name Service
DNSSERVER	Windows 2000 DNS
DOCSIS	DOCSIS 1.1
DOCSIS BPKM-ATTR	DOCSIS Baseline Privacy Key Management Attributes
DOCSIS BPKM-REQ	DOCSIS Baseline Privacy Key Management Request
DOCSIS BPKM-RSP	DOCSIS Baseline Privacy Key Management Response
DOCSIS DSA-ACK	DOCSIS Dynamic Service Addition Acknowledge
DOCSIS DSA-REQ	DOCSIS Dynamic Service Addition Request
DOCSIS DSA-RSP	DOCSIS Dynamic Service Addition Response
DOCSIS DSC-ACK	DOCSIS Dynamic Service Change Acknowledgement
DOCSIS DSC-REQ	DOCSIS Dynamic Service Change Request
DOCSIS DSC-RSP	DOCSIS Dynamic Service Change Response
DOCSIS DSD-REQ	DOCSIS Dynamic Service Delete Request
DOCSIS DSD-RSP	DOCSIS Dynamic Service Delete Response
DOCSIS INT-RNG-REQ	DOCSIS Initial Ranging Message
DOCSIS MAC MGMT	DOCSIS Mac Management
DOCSIS MAP	DOCSIS Upstream Bandwidth Allocation
DOCSIS REG-ACK	DOCSIS Registration Acknowledge

DOCSIS REG-REQ	DOCSIS Registration Requests
DOCSIS REG-RSP	DOCSIS Registration Responses
DOCSIS RNG-REQ	DOCSIS Range Request Message
DOCSIS RNG-RSP	DOCSIS Ranging Response
DOCSIS TLVs	DOCSIS Appendix C TLV's
DOCSIS type29ucd	DOCSIS Upstream Channel Descriptor Type 29
DOCSIS UCC-REQ	DOCSIS Upstream Channel Change Request
DOCSIS UCC-RSP	DOCSIS Upstream Channel Change Response
DOCSIS UCD	DOCSIS Upstream Channel Descriptor
DOCSIS VSIF	DOCSIS Vendor Specific Endodings
DRSUAPI	Microsoft Directory Replication Service
DSI	Data Stream
DTSPROVIDER	DCE Distributed Time Service Provider
DTSSTIME_REQ	DCE Distributed Time Service Local Server
DVMRP	Distance Vector Multicast Routing Protocol
E.164	ITU-T E.164 number
EAP	Extensible Authentication Protocol
EAPOL	802.1x Authentication
ECHO	Echo
EDONKEY	eDonkey Protocol
EFSRPC	Microsoft Encrypted File System Service
EIGRP	Enhanced Interior Gateway Routing Protocol
ENC	OpenBSD Encapsulating device
ENIP	EtherNet/IP (Industrial Protocol)
ENRP	Endpoint Name Resolution Protocol
ENTTEC	ENTTEC
EPM	DCE/RPC Endpoint Mapper
EPMv4	DCE/RPC Endpoint Mapper v4
ESIS	ISO 9542 ESIS Routeing Information Exchange Protocol
ESP	Encapsulating Security Payload
ESS	Extended Security Services
ETHERIC	Etheric
ETHERIP	Ethernet over IP
Ethernet	Ethernet
EVENTLOG	Microsoft Eventlog Service
FC	Fibre Channel
FC ELS	FC Extended Link Svc
FC FZS	Fibre Channel Fabric Zone Server
FC-dNS	Fibre Channel Name Server

FC-FCS	FC Fabric Configuration Server
FC-SB3	Fibre Channel Single Byte Command
FC-SP	Fibre Channel Security
FC-SWILS	Fibre Channel SW_ILS
FC_CT	Fibre Channel Common Transport
FCIP	FCIP
FCP	Fibre Channel Protocol for SCSI
FDDI	Fiber Distributed Data Interface
FIX	Financial Information eXchange Protocol
FLDB	DCE/RPC FLDB
FR	Frame Relay
Frame	Frame
FRSAPI	Microsoft File Replication Service API
FRSRPC	Microsoft File Replication Service
FTAM	OSI ISO 8571 FTAM Protocol
FTP	File Transfer Protocol (FTP)
FTP-DATA	FTP Data
FTSERVER	FTServer Operations
FW-1	Checkpoint FW-1
GIF image	Compuserve GIF
giFT	giFT Internet File Transfer
GIOP	General Inter-ORB Protocol
GMRP	GARP Multicast Registration Protocol
GNUTELLA	Gnutella Protocol
GPRS NS	GPRS Network service
GPRS-LLC	Logical Link Control GPRS
GRE	Generic Routing Encapsulation
Gryphon	DG Gryphon Protocol
GSM BSSMAP	GSM A-I/F BSSMAP
GSM DTAP	GSM A-I/F DTAP
GSM MAP	GSM Mobile Application Part
GSM RP	GSM A-I/F RP
GSM SMS	GSM SMS TPDU (GSM 03.40)
GSM SMS UD	GSM Short Message Service User Data
GSS-API	Generic Security Service Application Program Interface
GTP	GPRS Tunneling Protocol
GVRP	GARP VLAN Registration Protocol
H.261	ITU-T Recommendation H.261
H.263	ITU-T Recommendation H.263 RTP Payload header

	(RFC2190)
H1	Sinec H1 Protocol
H225	h225
H235	H235-SECURITY-MESSAGES
h245	h245
H248	H.248 MEGACO
h450	h450
HCLNFSD	Hummingbird NFS Daemon
HPEXT	HP Extended Local-Link Control
HSRP	Cisco Hot Standby Router Protocol
HTTP	Hypertext Transfer Protocol
HyperSCSI	HyperSCSI
IAP	Information Access Protocol
IAPP	Inter-Access-Point Protocol
IAX2	Inter-Asterisk eXchange v2
IB	Interbase
ICAP	Internet Content Adaptation Protocol
ICBAAccoCB	ICBAAccoCallback
ICBAAccoCB2	ICBAAccoCallback2
ICBAAccoMgt	ICBAAccoMgt
ICBAAccoMgt2	ICBAAccoMgt2
ICBAAccoServ	ICBAAccoServer
ICBAAccoServ2	ICBAAccoServer2
ICBAAccoServSR	ICBAAccoServerSRT
ICBAAccoSync	ICBAAccoSync
ICBABrowse	ICBABrowse
ICBABrowse2	ICBABrowse2
ICBAGErr	ICBAGroupError
ICBAGErrEvent	ICBAGroupErrorEvent
ICBALDev	ICBALogicalDevice
ICBALDev2	ICBALogicalDevice2
ICBAPDev	ICBAPhysicalDevice
ICBAPDev2	ICBAPhysicalDevice2
ICBAPDevPC	ICBAPhysicalDevicePC
ICBAPDevPCEvent	ICBAPhysicalDevicePCEvent
ICBAPersist	ICBAPersist
ICBAPersist2	ICBAPersist2
ICBARTAuto	ICBARTAuto
ICBARTAuto2	ICBARTAuto2
ICBAState	ICBAState

ICBAStateEvent	ICBAStateEvent
ICBASysProp	ICBASystemProperties
ICBATime	ICBATime
ICEP	Internet Communications Engine Protocol
ICL_RPC	DCE/RPC ICL RPC
ICMP	Internet Control Message Protocol
ICMPv6	Internet Control Message Protocol v6
ICP	Internet Cache Protocol
ICQ	ICQ Protocol
IDispatch	DCOM IDispatch
IEEE 802.11	IEEE 802.11 wireless LAN
IEEE802a	IEEE802a OUI Extended Ethertype
IGAP	Internet Group membership Authentication Protocol
IGMP	Internet Group Management Protocol
IGRP	Cisco Interior Gateway Routing Protocol
ILMI	ILMI
IMAP	Internet Message Access Protocol
INITSHUTDOWN	Remote Shutdown
IOXIDResolver	DCOM OXID Resolver
IP	Internet Protocol
IP/IEEE1394	Apple IP-over-IEEE 1394
IPComp	IP Payload Compression
IPDC	IP Device Control (SS7 over IP)
IPFC	IP Over FC
IPMI	Intelligent Platform Management Interface
IPP	Internet Printing Protocol
IPv6	Internet Protocol Version 6
IPVS	IP Virtual Services Sync Daemon
IPX	Internetwork Packet eXchange
IPX MSG	IPX Message
IPX RIP	IPX Routing Information Protocol
IPX SAP	Service Advertisement Protocol
IPX WAN	IPX WAN
IRC	Internet Relay Chat
IrCOMM	IrCOMM Protocol
IRemUnknown	IRemUnknown
IRemUnknown2	IRemUnknown2
IrLAP	IrDA Link Access Protocol
IrLMP	IrDA Link Management Protocol

ISAKMP	Internet Security Association and Key Management Protocol
iSCSI	iSCSI
ISDN	ISDN
ISIS	ISO 10589 ISIS InTRA Domain Routeing Information Exchange
ISL	Cisco ISL
ISMP	InterSwitch Message Protocol
iSNS	iSNS
ISUP	ISDN User Part
ISystemActivator	ISystemActivator ISystemActivator Resolver
IUA	ISDN Q.921-User Adaptation Layer
Jabber	Jabber XML Messaging
JFIF (JPEG) image	JPEG File Interchange Format
JXTA	JXTA P2P
KADM5	Kerberos Administration
KLM	Kernel Lock Manager
Kpasswd	MS Kpasswd
KRB4	Kerberos v4
KRB5	Kerberos
KRB5RPC	DCE/RPC Kerberos V
L2TP	Layer 2 Tunneling Protocol
LACP	Link Aggregation Control Protocol
LANMAN	Microsoft Windows Lanman Remote API Protocol
LAPB	Link Access Procedure Balanced (LAPB)
LAPBETHER	Link Access Procedure Balanced Ethernet (LAPBETHER)
LAPD	Link Access Procedure, Channel D (LAPD)
B Laplink	Laplink
LDAP	Lightweight Directory Access Protocol
LDP	Label Distribution Protocol
Line-based text data	Line-based text data
LLAP	LocalTalk Link Access Protocol
LLB	DCE/RPC NCS 1.5.1 Local Location Broker
LLC	Logical-Link Control
LMI	Local Management Interface
LMP	Link Management Protocol (LMP)
Log	Log Message
LogotypeCertExtn	Logotype Certificate Extensions
LOOP	Configuration Test Protocol (loopback)

LPD	Line Printer Daemon Protocol
LSA	Microsoft Local Security Architecture
LSA_DS	Microsoft Local Security Architecture (Directory Services)
Lucent/Ascend	Lucent/Ascend debug output
LWAPP	LWAPP Encapsulated Packet
LWAPP-CNTL	LWAP Control Message
LWAPP-L3	LWAPP Layer 3 Packet
LWRES	Light Weight DNS RESolver (BIND9)
M2PA	MTP2 Peer Adaptation Layer
M2TP	MTP 2 Transparent Proxy
M2UA	MTP 2 User Adaptation Layer
M3UA	MTP 3 User Adaptation Layer
MACC	MAC Control
Malformed packet	Malformed Packet
Manolito	Blubster/Piolet MANOLITO Protocol
MAP_DialoguePDU	MAP_DialoguePDU
MAPI	Microsoft Exchange MAPI
MDS Header	MDS Header
Media	Media Type
MEGACO	MEGACO
Messenger	Microsoft Messenger Service
MGCP	Media Gateway Control Protocol
MGMT	DCE/RPC Remote Management
MIME multipart	MIME Multipart Media Encapsulation
MIPv6	Mobile IPv6
MMSE	MMS Message Encapsulation
Mobile IP	Mobile IP
Modbus/TCP	Modbus/TCP
MOUNT	Mount Service
MPEG1	RFC 2250 MPEG1
MPLS	MultiProtocol Label Switching Header
MPLS Echo	Multiprotocol Label Switching Echo
MQ	WebSphere MQ
MQ PCF	WebSphere MQ Programmable Command Formats
MRDISC	Multicast Router DISCovery protocol
MS Proxy	MS Proxy Protocol
MSDP	Multicast Source Discovery Protocol
MSNIP	MSNIP: Multicast Source Notification of Interest Protocol

MSNMS	MSN Messenger Service
MTP2	Message Transfer Part Level 2
MTP3	Message Transfer Part Level 3
MTP3MG	Message Transfer Part Level 3 Management
MySQL	MySQL Protocol
NBDS	NetBIOS Datagram Service
NBIPX	NetBIOS over IPX
NBNS	NetBIOS Name Service
NBP	Name Binding Protocol
NBSS	NetBIOS Session Service
NCP	NetWare Core Protocol
NDMP	Network Data Management Protocol
NDPS	Novell Distributed Print System
NetBIOS	NetBIOS
NETLOGON	Microsoft Windows Logon Protocol
NFS	Network File System
NFSACL	NFSACL
NFSAUTH	NFSAUTH
NIS+	NIS+
NIS+ CB	NIS+ Callback
NLM	Network Lock Manager Protocol
NLSP	NetWare Link Services Protocol
NMAS	Novell Modular Authentication Service
NMPI	Name Management Protocol over IPX
NNTP	Network News Transfer Protocol
NORM	Negative-acknowledgment Oriented Reliable Multicast
NS_CERT_EXTS	NetScape Certificate Extensions
NSIP	Network Service Over IP
NSPI	NSPI
NTLMSSP	NTLM Secure Service Provider
NTP	Network Time Protocol
Null	Null/Loopback
NW_SERIAL	NetWare Serialization Protocol
OAM AAL	ATM OAM AAL
OCSP	Online Certificate Status Protocol
OLSR	Optimized Link State Routing Protocol
OPSI	Open Policy Service Interface
OSPF	Open Shortest Path First
PAGP	Port Aggregation Protocol

PCLI	Packet Cable Lawful Intercept
PCNFSD	PC NFS
PER	Packed Encoding Rules (ASN.1 X.691)
PFLOG	OpenBSD Packet Filter log file
PFLOG-OLD	OpenBSD Packet Filter log file, pre 3.4
PGM	Pragmatic General Multicast
PGSQL	PostgreSQL
PIM	Protocol Independent Multicast
PKCS-1	PKCS#1
PKInit	PKINIT
PKIX Certificate	PKIX CERT File Format
PKIX1EXPLICIT	PKIX1Explitit
PKIX1IMPLICIT	PKIX1Implitit
PKIXPROXY	PKIXProxy (RFC3820)
PKIXQUALIFIED	PKIX Qualified
PKIXTSP	PKIX Time Stamp Protocol
PKTC	PacketCable
PN-DCP	PROFINET DCP
PN-RT	PROFINET Real-Time Protocol
PNIO	PROFINET IO
POP	Post Office Protocol
Portmap	Portmap
PPP	Point-to-Point Protocol
PPP BACP	PPP Bandwidth Allocation Control Protocol
PPP BAP	PPP Bandwidth Allocation Protocol
PPP CBCP	PPP Callback Control Protocol
PPP CCP	PPP Compression Control Protocol
PPP CDPCP	PPP CDP Control Protocol
PPP CHAP	PPP Challenge Handshake Authentication Protocol
PPP Comp	PPP Compressed Datagram
PPP IPCP	PPP IP Control Protocol
PPP IPV6CP	PPP IPv6 Control Protocol
PPP LCP	PPP Link Control Protocol
PPP MP	PPP Multilink Protocol
PPP MPLSCP	PPP MPLS Control Protocol
PPP OSICP	PPP OSI Control Protocol
PPP PAP	PPP Password Authentication Protocol
PPP PPPMux	PPP Multiplexing
PPP PPPMuxCP	PPPMux Control Protocol
PPP VJ	PPP VJ Compression

PPP-HDLC	PPP In HDLC-Like Framing
PPPoED	PPP-over-Ethernet Discovery
PPPoES	PPP-over-Ethernet Session
PPTP	Point-to-Point Tunnelling Protocol
PRES	ISO 8823 OSI Presentation Protocol
Prism	Prism
PTP	Precision Time Protocol (IEEE1588)
Q.2931	Q.2931
Q.931	Q.931
Q.933	Q.933
QLLC	Qualified Logical Link Control
QUAKE	Quake Network Protocol
QUAKE2	Quake II Network Protocol
QUAKE3	Quake III Arena Network Protocol
QUAKEWORLD	QuakeWorld Network Protocol
RADIUS	Radius Protocol
RANAP	Radio Access Network Application Part
Raw	Raw packet data
Raw_SigComp	Decompressed SigComp message as raw text
Raw_SIP	Session Initiation Protocol (SIP as raw text)
rdaclif	DCE/RPC Directory Acl Interface
RDM	RDM
REMACT	DCOM IRemoteActivation
REP_PROC	AFS (4.0) Replication Server call declarations
RIP	Routing Information Protocol
RIPng	RIPng
RLM	Redundant Link Management Protocol
Rlogin	Rlogin Protocol
RMCP	Remote Management Control Protocol
RMI	Java RMI
RMP	HP Remote Maintenance Protocol
roverride	Remote Override interface
RPC	Remote Procedure Call
RPC_BROWSER	RPC Browser
RPC_NETLOGON	Microsoft Network Logon
RPL	Remote Program Load
RQUOTA	Remote Quota
RS_ACCT	DCE/RPC RS_ACCT
RS_ATTR	Registry Server Attributes Manipulation Interface
RS_BIND	DCE/RPC RS_BIND

RS_PGO	DCE Name Service
RS_PLCY	RS Interface properties
RS_REPADM	Registry server administration operations.
RS_REPLIST	DCE/RPC Repserver Calls
RS_UNIX	DCE/RPC RS_UNIX
RSH	Remote Shell
RSTAT	RSTAT
RSVP	Resource ReserVation Protocol (RSVP)
RSYNC	RSYNC File Synchroniser
RTcfg	RTcfg
RTCP	Real-time Transport Control Protocol
RTmac	Real-Time Media Access Control
RTMP	Routing Table Maintenance Protocol
RTP	Real-Time Transport Protocol
RTP Event	RFC 2833 RTP Event
RTPS	Real-Time Publish-Subscribe Wire Protocol
RTSP	Real Time Streaming Protocol
RUDP	Reliable UDP
RWALL	Remote Wall protocol
RX	RX Protocol
SADMIND	SADMIND
SAMR	Microsoft Security Account Manager
SAP	Session Announcement Protocol
SCCP	Signalling Connection Control Part
SCCPMG	Signalling Connection Control Part Management
SCSI	SCSI
SCTP	Stream Control Transmission Protocol
SDLC	Synchronous Data Link Control (SDLC)
SDP	Session Description Protocol
SEBEK	SEBEK – Kernel Data Capture
SECIDMAP	DCE Security ID Mapper
Serialization	Java Serialization
SES	ISO 8327-1 OSI Session Protocol
sFlow	InMon sFlow
SGI MOUNT	SGI Mount Service
Short frame	Short Frame
SIGCOMP	Signaling Compression
SIP	Session Initiation Protocol
SIPFRAG	Sipfrag
SIR	Serial Infrared

SKINNY	Skinny Client Control Protocol
SLARP	Cisco SLARP
SliMP3	SliMP3 Communication Protocol
SLL	Linux cooked-mode capture
SM	Cisco Session Management
SMB	SMB (Server Message Block Protocol)
SMB Mailslot	SMB MailSlot Protocol
SMB Pipe	SMB Pipe Protocol
SMPP	Short Message Peer to Peer
SMRSE	Short Message Relaying Service
SMTP	Simple Mail Transfer Protocol
SMUX	SNMP Multiplex Protocol
SNA	Systems Network Architecture
SNA XID	Systems Network Architecture XID
SNAETH	SNA-over-Ethernet
SNDCP	Subnetwork Dependent Convergence Protocol
SNMP	Simple Network Management Protocol
Socks	Socks Protocol
SONMP	Nortel SONMP
SoulSeek	SoulSeek Protocol
Spnego	Spnego
SPNEGO-KRB5	SPNEGO-KRB5
SPOOLSS	Microsoft Spool Subsystem
SPRAY	SPRAY
SPX	Sequenced Packet eXchange
SRVLOC	Service Location Protocol
SRVSVC	Microsoft Server Service
SSCF-NNI	SSCF-NNI
SSCOP	SSCOP
SSH	SSH
SSL	Secure Socket Layer
STAT	Network Status Monitor Protocol
STAT-CB	Network Status Monitor CallBack Protocol
STP	Spanning Tree Protocol
STUN	Simple Traversal of UDP Through NAT
SUA	SS7 SCCP-User Adaptation Layer
SVCCTL	Microsoft Service Control
Symantec	Symantec Enterprise Firewall
Syslog	Syslog message
T.38	T.38

TACACS	TACACS
TACACS+	TACACS+
TALI	Transport Adapter Layer Interface v1.0, RFC 3094
TAPI	Microsoft Telephony API Service
TCAP	Transaction Capabilities Application Part
TCP	Transmission Control Protocol
TDMA	TDMA RTmac Discipline
TDS	Tabular Data Stream
TELNET	Telnet
Teredo	Teredo IPv6 over UDP tunneling
TFTP	Trivial File Transfer Protocol
TIME	Time Protocol
TKN4Int	DCE/RPC TokenServer Calls
TNS	Transparent Network Substrate Protocol
Token-Ring	Token-Ring
TPCP	Alteon – Transparent Proxy Cache Protocol
TPKT	TPKT
TR MAC	Token-Ring Media Access Control
TRKSVR	Microsoft Distributed Link Tracking Server Service
TSP	Time Synchronization Protocol
TTP	Tiny Transport Protocol
TUXEDO	BEA Tuxedo
TZSP	Tazmen Sniffer Protocol
UBIKDISK	DCE/RPC FLDB UBIK TRANSFER
UBIKVOTE	DCE/RPC FLDB UBIKVOTE
UCP	Universal Computer Protocol
UDP	User Datagram Protocol
UDPENCAP	UDP Encapsulation of IPsec Packets
V.120	Async data over ISDN (V.120)
V5UA	V5.2-User Adaptation Layer
Vines ARP	Banyan Vines ARP
Vines Echo	Banyan Vines Echo
Vines FRP	Banyan Vines Fragmentation Protocol
Vines ICP	Banyan Vines ICP
Vines IP	Banyan Vines IP
Vines IPC	Banyan Vines IPC
Vines LLC	Banyan Vines LLC
Vines RTP	Banyan Vines RTP
Vines SPP	Banyan Vines SPP
VLAN	802.1q Virtual LAN

VRRP	Virtual Router Redundancy Protocol
VTP	Virtual Trunking Protocol
WAP SIR	WAP Session Initiation Request
WBXML	WAP Binary XML
WCCP	Web Cache Coordination Protocol
WCP	Wellfleet Compression
WHDLC	Wellfleet HDLC
WHO	Who
WINREG	Microsoft Registry
WKSSVC	Microsoft Workstation Service
WSP	Wireless Session Protocol
WTLS	Wireless Transport Layer Security
WTP	Wireless Transaction Protocol
X.25	X.25
X.29	X.29
X11	X11
X509AF	X.509 Authentication Framework
X509CE	X.509 Certificate Extensions
X509IF	X.509 Information Framework
X509SAT	X.509 Selected Attribute Types
XDMCP	X Display Manager Control Protocol
XOT	X.25 over TCP
XYPLEX	Xyplex
YHOO	Yahoo Messenger Protocol
YMSG	Yahoo YMSG Messenger Protocol
YPBIND	Yellow Pages Bind
YPPASSWD	Yellow Pages Passwd
YPSERV	Yellow Pages Service
YPXFR	Yellow Pages Transfer
ZEBRA	Zebra Protocol
ZIP	Zone Information Protocol

ПРИЛОЖЕНИЕ 2. ПЕРЕЧЕНЬ ПОРТОВ С УКАЗАНИЕМ НАЗНАЧЕНИЯ

Название сервиса или службы	Номер порта / транспортный протокол
TCPMUX	1/TCP
TCPMUX	1/UDP
RJE	5/TCP
RJE	5/UDP
ECHO	7/TCP
ECHO	7/UDP
DISCARD	9/TCP
DISCARD	9/UDP
SYSTAT	11/TCP
SYSTAT	11/UDP
DAYTIME	13/TCP
DAYTIME	13/UDP
QOTD	17/TCP
QOTD	17/UDP
MSP	18/TCP
MSP	18/UDP
CHARGEN	19/TCP
CHARGEN	19/UDP
FTP-DATA	20/TCP
FTP-DATA	20/UDP
FTP	21/TCP
FTP	21/UDP
SSH	22/TCP
SSH	22/UDP
TELNET	23/TCP
TELNET	23/UDP
SMTP	25/TCP
SMTP	25/UDP
TIME	37/TCP
TIME	37/UDP
RLP	39/TCP
RLP	39/UDP
NAMESERVER	42/TCP
NAMESERVER	42/UDP
NICNAME	43/TCP
NICNAME	43/UDP

TACACS	49/TCP
TACACS	49/UDP
RE-MAIL-CK	50/TCP
RE-MAIL-CK	50/UDP
DOMAIN	53/TCP
DOMAIN	53/UDP
WHOIS++	63/TCP
WHOIS++	63/UDP
BOOTPS	67/TCP
BOOTPS	67/UDP
BOOTPC	68/TCP
BOOTPC	68/UDP
TFTP	69/UDP
GOPHER	70/TCP
GOPHER	70/UDP
NETRJS-1	71/TCP
NETRJS-1	71/UDP
NETRJS-2	72/TCP
NETRJS-2	72/UDP
NETRJS-3	73/TCP
NETRJS-3	73/UDP
NETRJS-4	74/TCP
NETRJS-4	74/UDP
FINGER	79/TCP
FINGER	79/UDP
HTTP	80/TCP
HTTP	80/UDP
KERBEROS	88/TCP
KERBEROS	88/UDP
SUPDUP	95/TCP
SUPDUP	95/UDP
HOSTNAME	101/TCP
HOSTNAME	101/UDP
ISO-TSAP	102/TCP
CSNET-NS	105/TCP
CSNET-NS	105/UDP
RTELNET	107/TCP
RTELNET	107/UDP
POP2	109/TCP
POP2	109/UDP

POP3	110/TCP
POP3	110/UDP
SUNRPC	111/TCP
SUNRPC	111/UDP
AUTH	113/TCP
AUTH	113/UDP
SFTP	115/TCP
SFTP	115/UDP
UUCP-PATH	117/TCP
UUCP-PATH	117/UDP
NNTP	119/TCP
NNTP	119/UDP
NTP	123/TCP
NTP	123/UDP
NETBIOS-NS	137/TCP
NETBIOS-NS	137/UDP
NETBIOS-DGM	138/TCP
NETBIOS-DGM	138/UDP
NETBIOS-SSN	139/TCP
NETBIOS-SSN	139/UDP
IMAP	143/TCP
IMAP	143/UDP
SNMP	161/TCP
SNMP	161/UDP
SNMPTRAP	162/UDP
CMIP-MAN	163/TCP
CMIP-MAN	163/UDP
CMIP-AGENT	164/TCP
CMIP-AGENT	164/UDP
MAILQ	174/TCP
MAILQ	174/UDP
XDMCP	177/TCP
XDMCP	177/UDP
NEXTSTEP	178/TCP
NEXTSTEP	178/UDP
BGP	179/TCP
BGP	179/UDP
PROSPERO	191/TCP
PROSPERO	191/UDP
IRC	194/TCP

IRC	194/UDP
SMUX	199/TCP
SMUX	199/UDP
AT-RTMP	201/TCP
AT-RTMP	201/UDP
AT-NBP	202/TCP
AT-NBP	202/UDP
AT-ECHO	204/TCP
AT-ECHO	204/UDP
AT-ZIS	206/TCP
AT-ZIS	206/UDP
QMTP	209/TCP
QMTP	209/UDP
Z39.50	210/TCP
Z39.50	210/UDP
IPX	213/TCP
IPX	213/UDP
IMAP3	220/TCP
IMAP3	220/UDP
LINK	245/TCP
LINK	245/UCP
FATSERV	347/TCP
FATSERV	347/UDP
RSVP_TUNNEL	363/TCP
RSVP_TUNNEL	363/UDP
RPC2PORTMAP	369/TCP
RPC2PORTMAP	369/UDP
CODAAUTH2	370/TCP
CODAAUTH2	370/UDP
ULISTPROC	372/TCP
ULISTPROC	372/UDP
LDAP	389/TCP
LDAP	389/UDP
SVRLOC	427/TCP
SVRLOC	427/UDP
MOBILEIP-AGENT	434/TCP
MOBILEIP-AGENT	434/UDP
MOBILIP-MN	435/TCP
MOBILIP-MN	435/UDP
HTTPS	443/TCP

HTTPS	443/UDP
SNPP	444/TCP
SNPP	444/UDP
KPASSWD	464/TCP
KPASSWD	464/UDP
PHOTURIS	468/TCP
PHOTURIS	468/UDP
SAFT	487/TCP
SAFT	487/UDP
GSS-HTTP	488/TCP
GSS-HTTP	488/UDP
PIM-RP-DISC	496/TCP
PIM-RP-DISC	496/UDP
ISAKMP	500/TCP
ISAKMP	500/UDP
GDOMAP	538/TCP
GDOMAP	538/UDP
IIOP	535/TCP
IIOP	535/UDP
DHCPV6-CLIENT	546/TCP
DHCPV6-CLIENT	546/UDP
DHCPV6-SERVER	547/TCP
DHCPV6-SERVER	547/UDP
RTSP	554/TCP
RTSP	554/UDP
NNTPS	563/TCP
NNTPS	563/UDP
WHOAMI	565/TCP
WHOAMI	565/UDP
SUBMISSION	587/TCP
SUBMISSION	587/UDP
NPMP-LOCAL	610/TCP
NPMP-LOCAL	610/UDP
NPMP-GUI	611/TCP
NPMP-GUI	611/UDP
HMMP-IND	612/TCP
HMMP-IND	612/UDP
IPP	631/TCP
IPP	631/UCP
LDAPS	636/TCP

LDAPS	636/UDP
ACAP	674/TCP
ACAP	674/UDP
HA-CLUSTER	694/TCP
HA-CLUSTER	694/UDP
KERBEROS-IV	750/UDP
KERBEROS-IV	750/TCP
WEBSTER	765/TCP
WEBSTER	765/UDP
PHONEBOOK	767/TCP
PHONEBOOK	767/UDP
RSYNC	873/TCP
RSYNC	873/UDP
TELNETS	992/TCP
TELNETS	992/UDP
IMAPS	993/TCP
IMAPS	993/UDP
IRCS	994/TCP
IRCS	994/UDP
POP3S	995/TCP
POP3S	995/UDP
SOCKS	1080/TCP
SOCKS	1080/UDP
MS-SQL-S	1433/TCP
MS-SQL-S	1433/UDP
MS-SQL-M	1434/TCP
MS-SQL-M	1434/UDP
ICA	1494/TCP
ICA	1494/UDP
WINS	1512/TCP
WINS	1512/UDP
INGRESLOCK	1524/UDP
PROSPERO-NP	1525/TCP
PROSPERO-NP	1525/UDP
DATAMETRICS	1645/TCP
DATAMETRICS	1645/UDP
SA-MSG-PORT	1646/TCP
SA-MSG-PORT	1646/UDP
KERMIT	1649/TCP
KERMIT	1649/UDP

L2TP	1701/TCP
L2TP	1701/UDP
TFTP-MCAST	1758/TCP
TFTP-MCAST	1758/UDP
HELLO	1789/TCP
HELLO	1789/UDP
RADIUS	1812/TCP
RADIUS	1812/UDP
RADIUS-ACCT	1813/TCP
RADIUS-ACCT	1813/UDP
MTP	1911/TCP
MTP	1911/UDP
HSRP	1985/TCP
HSRP	1985/UDP
GDP-PORT	1997/TCP
GDP-PORT	1997/UDP
NFS	2049/TCP
NFS	2049/UDP
ZEPHYR-SRV	2102/TCP
ZEPHYR-SRV	2102/UDP
ZEPHYR-CLT	2103/TCP
ZEPHYR-CLT	2103/UDP
ZEPHYR-HM	2104/TCP
ZEPHYR-HM	2104/UDP
CVSPSERVER	2401/TCP
CVSPSERVER	2401/UDP
VENUS	2430/TCP
VENUS	2430/UDP
VENUS-SE	2431/TCP
VENUS-SE	2431/UDP
CODASRV	2432/TCP
CODASRV	2432/UDP
CODASRV-SE	2433/TCP
CODASRV-SE	2433/UDP
HPSTGMGR	2600/TCP
HPSTGMGR	2600/UDP
DISCP-CLIENT	2601/TCP
DISCP-CLIENT	2601/UDP
DISCP-SERVER	2602/TCP
DISCP-SERVER	2602/UDP

SERVICEMETER	2603/TCP
SERVICEMETER	2603/UDP
NSC-CCS	2604/TCP
NSC-CCS	2604/UDP
NSC-POSA	2605/TCP
NSC-POSA	2605/UDP
NETMON	2606/TCP
NETMON	2606/UDP
CORBALOC	2809/TCP
ICPV2	3130/TCP
ICPV2	3130/UDP
MYSQL	3306/TCP
MYSQL	3306/UDP
RWHOIS	4321/TCP
RWHOIS	4321/UDP
KRB524	4444/TCP
KRB524	4444/UDP
RFE	5002/TCP
RFE	5002/UDP
CFENGINE	5308/TCP
CFENGINE	5308/UDP
CVSUP	5999/TCP
CVSUP	5999/UDP
X11	6000/TCP
AFS3-VOLSER	7005/TCP
AFS3-VOLSER	7005/UDP
AFS3-ERRORS	7006/TCP
AFS3-ERRORS	7006/UDP
AFS3-BOS	7007/TCP
AFS3-BOS	7007/UDP
AFS3-UPDATE	7008/TCP
AFS3-UPDATE	7008/UDP
AFS3-RMTSYS	7009/TCP
AFS3-RMTSYS	7009/UDP
SD	9876/TCP
SD	9876/UDP
AMANDA	10080/TCP
AMANDA	10080/UDP
BPRD	13720/TCP
BPRD	13720/UDP

BPDBM	13721/TCP
BPDBM	13721/UDP
BPJAVA-MSVC	13722/TCP
BPJAVA-MSVC	13722/UDP
VNETD	13724/TCP
VNETD	13724/UDP
BPCD	13782/TCP
BPCD	13782/UDP
VOPIED	13783/TCP
VOPIED	13783/UDP
QUAKE	26000/TCP
QUAKE	26000/UDP
WNN6-DS	26208/TCP
WNN6-DS	26208/UDP
TRACEROUTE	33434/TCP
TRACEROUTE	33434/UDP

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Олифер В.Г., Олифер Н.А, Компьютерные сети. Принципы, технологии, протоколы, 4-е изд., СПб: изд. Питер, – 2010 г., 944 с.
2. Таненбаум Э., Уэзеролл Д. Компьютерные сети. 5-е изд. – СПб.: Питер, 2012. – 960 с.: ил
3. Столлингс В., Компьютерные сети, протоколы и технологии Интернета, Пер с англ., СПб: изд. БХВ, – 2010 г., 832 с.
4. Дибров, М. В. Сети и телекоммуникации. Маршрутизация в ip-сетях в 2 ч. Часть 1 : учебник и практикум для академического бакалавриата / М. В. Дибров. – М. : Издательство Юрайт, 2018. – 333 с.
5. Сети и телекоммуникации : учебник и практикум для академического бакалавриата / К. Е. Самуйлов [и др.] ; под ред. К. Е. Самуйлова, И. А. Шалимова, Д. С. Кулябова. – М. : Издательство Юрайт, 2019. – 363 с.
6. Баринов, В.В. Компьютерные сети: Учебник / В.В. Баринов, И.В. Баринов, А.В. Пролетарский. – М.: Academia, 2018. – 192 с.
7. Лэммл Т, Cisco Certified Network Associate Учебное руководство. Экзамен 640-507, 2-е изд., Пер. с англ. М: изд. «Лори», – 2002 г., 576 с.
8. Найк Д., Стандарты и протоколы Интернета, Пер. с англ., М: Издательский отдел «Русская редакция» ТОО Channel Trading Ltd, – 1999 г., 384 с.
9. Колисниченко Д.Н. Linux-сервер своими руками, СПб: Наука и техника, – 2002 г, 576 с.
10. Кульгин М.Компьютерные сети. Практика построения., 2-е изд. СПб: изд. Питер, – 2003 г., 459 с.
11. Семенов А.Б. Проектирование и расчет структурированных кабельных систем и их компонентов, М: изд. АйТи, – 2003 г., 416 с.
12. Стахнов А. Сетевое администрирование Linux., СПб: изд. «БХВ», – 2004 г., 452 с.

Харитонов Антон
Джаманкулов Амантур

Компьютерные сети и протоколы передачи данных

Учебное пособие

В авторской редакции
Редакционно-издательский отдел Университета ИТМО
Зав. РИО Н.Ф. Гусарова
Подписано к печати
Заказ №
Тираж
Отпечатано на ризографе

Редакционно-издательский отдел
Университета ИТМО
197101, Санкт-Петербург, Кронверкский пр., 49, литер А