МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНА МЕТАЛУРГІЙНА АКАДЕМІЯ УКРАЇНИ

Г. Г. Швачич, О. В. Овсянніков, Л.М.Петречук

Комп'ютерні мережі та телекомунікації

для студентів спеціальностей 6.020105 «Документознвавство та інформаційна діяльність»

Конспект лекцій

Дніпропетровськ НМетАУ 2010

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНА МЕТАЛУРГІЙНА АКАДЕМІЯ УКРАЇНИ

Г. Г. Швачич, О. В. Овсянніков, Л.М.Петречук

Комп'ютерні мережі та телекомунікації

для студентів спеціальностей 6.020105 «Документознвавство та інформаційна діяльність»

Затверджено на засіданні Вченої ради академії
як конспект лекцій

Дніпропетровськ НМетАУ 2010

УДК 004 (075.8)
Г.Г. Швачич, О.В. Овсянніков, Л.М.Петречук. **Документознавство та інформаційна діяльність**. Конспект лекцій. – Дніпропетровськ: – НМетАУ, 2010. – 48 с.

Викладені основи ознайомлення з методами та алгоритмами обробки і візуалізації наукової та дослідницької інформації, та основи ознайомлення з сучасними засобами візуальної розробки Windows додатків.

Призначений для студентів спеціальностей «Документознавство та інформаційна діяльність».

Іл. 31. Бібліогр.: 5 найм.

Відповідальний за випуск Г.Г. Швачич, канд. техн. наук, проф.

Рецензенти: Б. И. Мороз д-р техн. наук, проф. (Академія митної Служби України)
Т. И. Пашова канд. техн. наук, доц. (Дніпропетровсьгий державний аграрний університет)

Классификация компьютерных сетей

Классификация — процесс группирования (отнесения к тому или иному типу) объектов изучения в соответствии с их общими признаками.

Классификация компьютерных сетей в технологическом аспекте

В качестве *критериев* классификации взяты различные *технологические* характеристики сетей: топология, метод коммутации, метод продвижения пакетов, тип среды передачи и другие. Такая классификация называется классификацией в технологическом аспекте.

По территории покрытия

Поскольку до недавнего времени выбор технологии, используемой для построения сети, был в первую очередь обусловлен ее *территориальным масштабом*, мы начнем нашу классификацию с технологических признаков компьютерной сети, обусловленных *территорией покрытия*. Все сети по этому критерию можно разделить на две группы:

- локальные сети (Local Area Network, LAN);
- глобальные сети (Wide Area Network, WAN).

Первые локальные и глобальные сети представляли собой два существенно отличающихся технологических направления. В частности, в локальных сетях обычно используются более качественные линии связи, которые не всегда доступны (из-за экономических ограничений) на больших расстояниях, свойственных глобальным сетям. Высокое качество линий связи в локальных сетях позволило упростить процедуры передачи данных за счет применения немодулированных сигналов и отказа от обязательного подтверждения получения пакета. Благодаря этому, скорость обмена данными между конечными узлами в локальных сетях, как правило, выше, чем в глобальных. Несмотря на то, что процесс сближения технологий локальных и глобальных сетей идет уже давно, различия между этими технологиями все еще достаточно отчетливы, что и дает основания относить соответствующие сети к различным технологическим типам.

Говоря в данном контексте «локальные сети» или «глобальные сети», прежде всего имеются в виду, различия *технологий* локальных и глобальных сетей, а не тот факт, что эти сети имеют разный территориальный масштаб.

Городские сети, или сети мегаполиса (Metropolitan Area Network, MAN) предназначены для обслуживания территории крупного города — мегаполиса, и сочетают в себе признаки как локальных, так и глобальных сетей. От первых они унаследовали плотность подключения конечных абонентов и высокосортные линии связи, а от последних — протяженность линий связи. В то же время появление городские сетей не привело к возникновению каких-нибудь качественно новых технологий, поэтому их не выделили в отдельный технологический тип сетей.

По среде передачи

В соответствии с технологическими признаками, обусловленными средой передачи, компьютерные сети подразделяют на два класса:

- проводные сети, то есть сети, каналы связи которых построены с использованием медных или оптических кабелей;
- беспроводные сети, то есть сети, в которых для связи используются беспроводные каналы связи, например, радио, СВЧ, инфракрасные или лазерные каналы.

Тип среды передачи влияет на технологию компьютерной сети, так как ее протоколы должны учитывать скорость и надежность соединения, обеспечиваемого каналом, а также частоту искажения в нем битов информации. Различие технологий локальных и глобальных сетей во многом определялось различием качества используемых в этих сетях каналов связи.

Качество канала связи зависит от многих факторов, но наиболее кардинально на него влияет выбор проводной или беспроводной среды.

Любая беспроводная среда — будь то радиоволны, инфракрасные лучи или СВЧ сигналы спутниковой связи — гораздо больше подвержена влиянию внешних помех, чем проводная. Роса, туман, солнечные бури, работающие в комнате микроволновые печи — вот только несколько примеров источников помех, которые могут привести к резкому ухудшению качества беспроводного канала. А значит, технологии беспроводных сетей должны учитывать типичность таких ситуаций и строиться таким образом, чтобы обеспечивать работоспособность сети несмотря на ухудшение вешних условий. Кроме того, существует ряд других специфических особенностей беспроводных сетей, которые служат основанием для выделения их в особый класс, например, естественное разделение радиосреды узлами сети, находящимися в радиусе действия всенаправленного передатчика; распределение диапазона радиочастот между сетями различного назначения, например, между телефонными и компьютерными.

По способу коммутации

В зависимости от способа коммутации, сети подразделяются на два класса:

- сети с коммутацией пакетов;
- сети с коммутацией каналов.

Хотя в компьютерных сетях преимущественно используется техника коммутации пакетов, принципиально допустимо и применение в них техники коммутации каналов.

В свое очередь, техника коммутации пакетов допускает несколько вариаций, отличающихся способом продвижения пакетов:

- дейтаграммные сети, например, Ethernet;
- сети, основанные на логических соединениях, например, IP-сети, использующие на транспортном уровне протокол TCP;
- сети, основанные на виртуальных каналах, например, MPLS-сети.

На основе топологии

- полносвязная топология;
- древовидная топология;
- топология звезда;
- топология кольцо;

- смешанная топология.

По признаку их первичности

Компьютерные сети разделяют также по признаку их первичности:

- первичные сети;
- наложенные сети.

Первичные сети занимают особое положение в мире телекоммуникационных сетей, это своего рода вспомогательные сети, которые нужны для того, чтобы гибко создавать постоянные физические двухточечные каналы для других компьютерных и телефонных сетей. В соответствии с семиуровневой моделью OSI первичные сети подобно простым кабелям выполняют функции физического уровня сетей. Однако в отличие от кабелей первичные сети включают дополнительное коммуникационное оборудование, которое путем соответствующего конфигурирования позволяет прокладывать новые физические каналы между конечными точками сети. Другими словами, первичная сеть — это гибкая среда для создания физических каналов связи.

Наложенные сети в этой классификации — это все остальные сети, которые предоставляют услуги конечным пользователям и строятся на основе каналов первичных сетей — «накладываются» поверх этих сетей. То есть и компьютерные, и телефонные, и телевизионные сети являются наложенными.

Другие аспекты классификации компьютерных сетей

Сети можно классифицировать в зависимости от того, кому предназначаются услуги этих сетей. Впервые мы имеем дело с критерием классификации, который относится к группе организационных, а не технических.

Итак, в зависимости от того, какому типу пользователей предназначаются услуги сети, сети делятся на два класса:

- сети операторов связи;
- корпоративные сети.

Сети операторов связи предоставляют публичные услуги, то есть клиентом сети может стать любой индивидуальный пользователь или любая организация, которая заключила соответствующий коммерческий договор на предоставление той или иной телекоммуникационной услуги. Традиционными услугами операторов связи являются услуги телефонии, а также предоставления каналов связи в аренду тем организациям, которые собираются строить на их основе собственные сети. С распространением компьютерных сетей операторы связи существенно расширили спектр своих услуг, добавив доступ в Интернет, услуги виртуальных частных сетей, веб-хостинг, электронную почту и IP-телефонию, а также широковещательную рассылку аудио- и видеосигналов. Ввиду того, что сеть оператора связи обслуживает, как правило, больше клиентов, чем корпоративная сеть (бывают, конечно, и исключения из этого правила), и кроме того, оператор несет прямую материальную ответственность за сбои в работе своей сети,

существует неформальное понятие «оборудование операторского класса», отражающее высокие показатели надежности, управляемости и производительности такого оборудования.

Корпоративные сети предоставляют услуги только сотрудникам предприятия, которое владеет этой сетью. Хотя формально корпоративная сеть может иметь любой размер, обычно под корпоративной понимают сеть крупного предприятия, которая состоит как из локальный сетей, так и из объединяющей их глобальной сети.

В зависимости от функциональной роли в составной сети делятся на три класса:

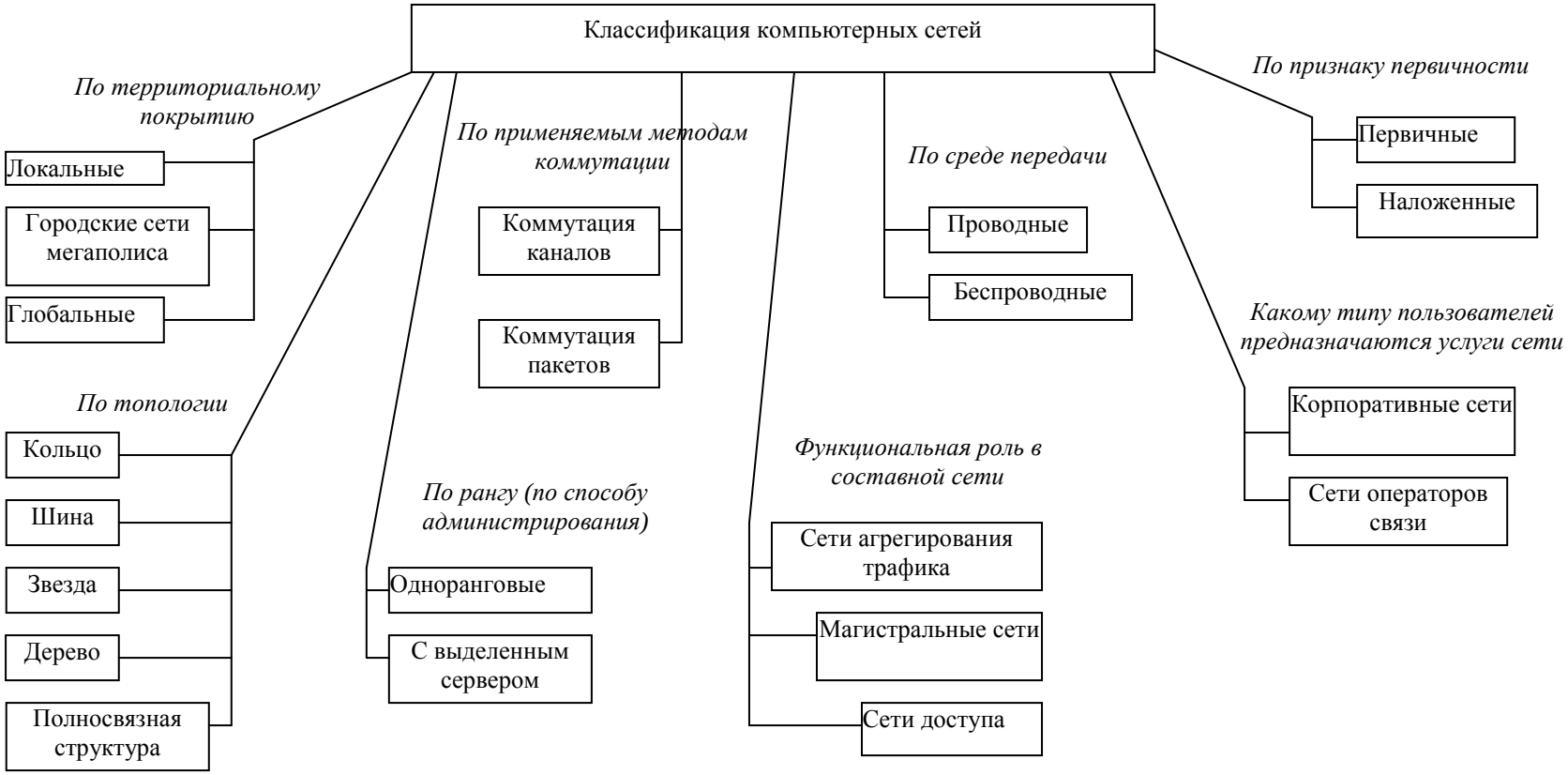
- сети доступа;
- магистральные сети;
- сети агрегирования трафика.

Сети доступа — это сети, предоставляющие доступ индивидуальным и корпоративным абонентам от их помещений (квартир, офисов) до первого помещения (пункта присутствия) оператора сети связи или оператора корпоративной сети. Другими словами, это сети, ответственные за расширение глобальной сети до помещений ее клиентов.

Магистральные сети — это сети, представляющие собой наиболее скоростную часть (ядро) глобальной сети, которая объединяет многочисленные сети доступа в единую сеть.

Сети агрегирования трафика — это сети, агрегирующие данные от многочисленных сетей доступа для компактной передачи их по небольшому числу каналов связи в магистраль. Сети агрегирования обычно используются только в крупных глобальных сетях, где они занимают промежуточную позицию, помогая магистральной сети обрабатывать трафик, поступающий от большого числа сетей доступа. В сетях среднего и небольшого размера сети агрегирования обычно отсутствуют.

Классификации компьютерных сетей, носит условный характер. Например, предприятие, у которого имеется много небольших филиалов в разных городах. Сеть каждого филиала располагается в пределах одного здания, и по критерию территориального покрытия относится к классу локальных сетей. Данная организация обладает также сетью, которая связывает все локальные сети филиалов в единую сеть, покрывающую большую территорию, и по данному признаку относится к классу глобальных сетей. В то же время все сети рассматриваемой организации (и сети филиалов и связывающая их сеть) входят в один и тот же класс — класс корпоративных сетей.


Классификация компьютерных сетей
По территориальному покрытию
Локальные
Городские сети мегаполиса
Глобальные
По топологии
Кольцо
Шина
Звезда
Дерево
Полносвязная структура
По применяемым методам коммутации
Коммутация каналов
Коммутация пакетов
По рангу (по способу администрирования)
Одноранговые
С выделенным сервером
По среде передачи
Проводные
Беспроводные
Функциональная роль в составной сети
Сети агрегирования трафика
Магистральные сети
Сети доступа
По признаку первичности
Первичные
Наложенные
Какому типу пользователей предназначаются услуги сети
Корпоративные сети
Сети операторов связи

ЛОГИЧЕСКАЯ СТРУКТУРА СЕТИ

Различают физическую и логическую топологию *(шина, звезда…)*. **Физическая топология** - это геометрия построения сети, *схема соединения компьютеров в сети с использованием дополнительных технических средств*, а **логическая топология** определяет направления потоков данных между узлами сети и способы передачи данных, *программно - аппаратная структура, обеспечивающая перераспределение передаваемого трафика между различными физическими сегментами сети*.

Трафик – это объём информации, передаваемой по сети за определенный период времени (объем данных, которые проходят через сервер за какое-то определенное время). Измеряться он может в килобайтах, мегабайтах и гигабайтах, в зависимости от масштабов.

С точки зрения пользователя (создающего или имеющего собственный сайт) трафик – это объём принятых/отправленных данных его компьютером, посещаемость. Различают два вида: входящий (данные, которые принимает сервер) и исходящий трафики (данные, которые он отправляет).

Логическая и физическая топологии сети независимы друг от друга. Логическая топология не всегда будет соответствовать виду физической топологии т.к. в большинстве локальных сетей применяются соединительные устройства (такие как концентраторы (hubs) и маршрутизаторы (routers)), которые изменяют вид фактической схемы соединения. Особенности схемы соединения скрываются внутри соединительного устройства. Физическая топология выглядит звездообразной. Однако внутренняя проводка соединительного маршрутизатора обеспечивает реализацию логической шинной топологии.

Достаточно часто логические кольцевые или смешанные топологии реализуются в виде физических звездообразных топологий.

Наиболее важной проблемой, не решаемой путём физической структуризации, остаётся проблема перераспределения передаваемого трафика между различными физическими сегментами сети.

В большой сети естественным образом возникает неоднородность информационных потоков, так как сеть состоит из множества подсетей рабочих групп, отделов, филиалов предприятия и других административных образований. Очень часто наиболее интенсивный обмен данными наблюдается между компьютерами, принадлежащими к одной подсети, и только небольшая часть обращений происходит к ресурсам компьютеров, находящихся вне локальных рабочих групп. Также не редки ситуации, когда интенсивность внешних обращений выше интенсивности обмена между «соседними» машинами. Тем не менее, независимо от того, в какой пропорции распределяются внешний и внутренний трафик, для повышения

эффективности работы сети неоднородность информационных потоков необходимо учитывать.

Сеть с типовой топологией (шина, кольцо, звезда), в которой все физические сегменты развиваются в качестве одной распределяемой среды, оказывается неадекватной структуре информационных потоков в большой сети. Например, в сети с общей шиной взаимодействие любой пары компьютеров занимает её на всё время обмена, поэтому при увеличении числа компьютеров в сети шина становится узким местом.

Сети, построенные на основе концентраторов

Концентратор (concentrator), хаб (hub), повторитель (repeator). – это устройство, к которому подключаются сетевые объекты при звездообразной топологии. Концентратор играет роль разделителя сигналов, он принимает сигнал, поступивший в него из одного порта, и распределяет его по всем остальным своим портам. Т.е. *концентратор - устройства обмена информацией, не использующие аппаратные адреса компьютеров для локализации трафика внутри локальной сети*.

Некоторые концентраторы перед тем, как передать поступивший слабый сигнал, усиливают его.

На рисунке 1 показана сеть, построенная с использованием концентраторов. Пусть компьютер А, находящийся в одной подсети с компьютером В, посылает ему данные. Несмотря на разветвлённую *физическую* структуру сети, концентраторы распространяют любой кадр *по всем* её сегментам. Поэтому кадр, посылаемый компьютером А компьютеру В, хотя и не нужен отделам 2 и 3, в соответствии с *логикой* работы концентраторов поступает на эти сегменты тоже. И до тех пор, пока компьютер В не получит адресованный ему кадр, не один из компьютеров этой сети не сможет передавать данные.

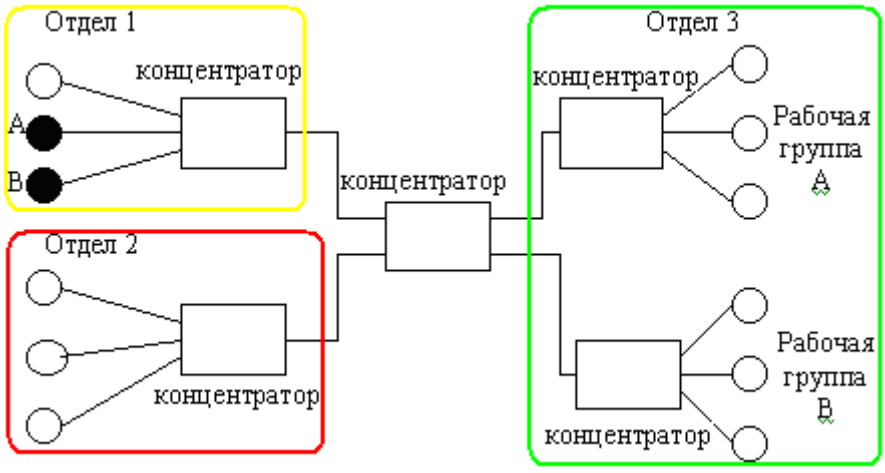


Рис. 1 Физическая структуризация с помощью концентраторов

В рассмотренном выше примере желательно бы было сделать так, чтобы кадры, которые передаёт компьютер одела 1, выходили бы за пределы этой сети только том случае, если эти кадры направлены какому-либо компьютеру из других отделов. С другой стороны, в сеть каждого из отделов должны попадать только те кадры, которые адресованы узлам этой сети.

Распространение трафика, предназначенного для компьютеров некоторого сегмента сети, только в пределах этого сегмента, называется локализацией трафика. Логическая структуризация сети – это процесс разбиения сети на сегменты с локализованным трафиком.

Для логической структуризации сети используются также коммуникационные устройства, как мосты, коммутаторы, маршрутизаторы и шлюзы.

Кадр(«frame») это блок данных, передаваемых по сети. Размер и структура кадра определяются используемым в сети протоколом *аппаратного* уровня. Между кадром и почтовым конвертом можно провести параллель - всем известно, что конверт стандарта #10 имеет физические размеры 4 1/8 дюйма на 9 1/2 дюйма. Но содержимое стандартных конвертов отличается по размеру, содержанию, срочности и т.д. Знание размеров конверта никоим образом не влияет на способ его доставки получателю. Механизм передачи кадров в сети называется протоколом. Протоколы фигурируют на третьем уровне эталонной модели OSI. Именно они формируют из кадров *пакеты* и обеспечивают их пересылку в локальной сети.

Для передачи данных в локальных и глобальных сетях устройство-отправитель должно знать адрес устройства-получателя. Поэтому каждый сетевой компьютер имеет уникальный адрес, и не один, а целых три адреса:

- физический или аппаратный (MAC-адрес);

- сетевой (IP-адрес);

- символьный (обычное имя компьютера или полное доменное имя).

Физический адрес компьютера

MAC-адрес (от англ. Media Access Control — управление доступом к среде, также Hardware Address) — это уникальный идентификатор, присваиваемый каждой единице оборудования компьютерных сетей.

MAC-адрес жестко “зашивается” в сетевую карту ее производителем и обычно записывается в виде 12 шестнадцатеричных цифр (например, 00-03-

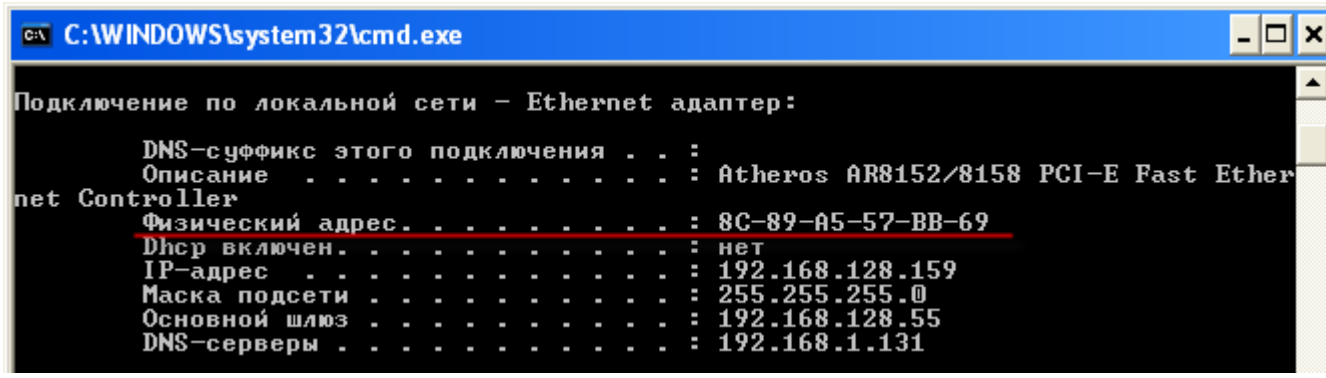
BC-12-5D-4E). Это гарантированно уникальный адрес: первые шесть символов идентифицируют фирму-производителя, которая следит, чтобы остальные шесть символов не повторялись на производственном конвейере. Когда у машины заменяется сетевой адаптер, то меняется и ее MAC-адрес.

Узнать MAC-адрес сетевой карты вашего компьютера можно следующим образом:

1. Зайдите в “Пуск” – “Выполнить” – введите с клавиатуры команду cmd – “ОК”.

2. Введите команду ipconfig /all и нажмите клавишу Enter.

Данная команда позволяет получить полную информацию обо всех сетевых картах ПК. Поэтому найдите в этом окошке строку Физический адрес – в ней будет обозначен MAC-адрес вашей сетевой карты. В моем случае это выглядит так:



Сетевой адрес компьютера

Сетевой адрес, или IP-адрес используется в сетях TCP/IP при обмене данными на сетевом уровне. IP расшифровывается как Internet Protocol – протокол интернета. IP-адрес компьютера имеет длину 32 бита и состоит из четырех частей, именуемых *октетами*. Каждый октет может принимать значения от 0 до 255 (например, 90.188.125.200). Октеты отделяются друг от друга точками.

IP-адрес компьютера, например 192.168.1.10, состоит из двух частей – номера сети (иногда называемого идентификатором сети) и номера сетевого компьютера (идентификатора хоста).

Номер сети должен быть одинаковым для всех компьютеров сети и в нашем примере номер сети будет равен 192.168.1. Номер компьютера должен быть уникален в данной сети, и компьютер в нашем примере имеет номер 10.

IP-адреса компьютеров в разных сетях могут иметь одинаковые номера. Например, компьютеры с IP-адресами 192.168.1.10 и 192.168.15.10 хоть и имеют одинаковые номера (10), но принадлежат к разным сетям (1 и 15). Поскольку адреса сетей различны, то компьютеры не могут быть спутаны друг с другом.

Чтобы отделить номер сети от номера компьютера, применяется маска подсети. Чисто внешне маска подсети представляет собой такой же набор из четырех октетов, разделенных между собой точками. Но, как правило, большинство цифр в ней – это 255 и 0.

255 указывает на биты, предназначенные для адреса сети, в остальных местах (которым соответствует значение 0) должен располагаться адрес компьютера. Чем меньше значение маски, тем больше компьютеров объединено в данную подсеть. Маска сети присваивается компьютеру одновременно с IP-адресом. Чтобы было понятно, приведем простой пример:

- сеть 192.168.0.0 с маской 255.255.255.0 может содержать в себе компьютеры с адресами от 192.168.0.1 до 192.168.0.254,
- а сеть 192.168.0.0 с маской 255.255.255.128 допускает адреса от 192.168.0.1 до 192.168.0.127.

Сети с большим количеством компьютеров делят на части, называемые подсетями. Деление на подсети применяется для обеспечения повышенной безопасности и разграничения доступа к ресурсам различных подсетей. Компьютеры разных подсетей не смогут передавать пакеты друг другу без специального устройства – *маршрутизатора*, а, следовательно, никто не сможет проникнуть в защищенную таким образом подсеть. Чтобы создать подсети, часть места в IP-адресе, отведенном для номера хоста, отдают под номера подсети.

Рассмотрим пример, когда у нас в локальной сети 30 компьютеров и требуется настроить их так, чтобы 20 компьютеров могли “общаться” между собой, но не смогли передавать и принимать данные от остальных 10 компьютеров, которые также должны общаться только между собой. Решение этой задачи довольно простое – делим нашу сеть на две подсети. В первой подсети “раздаем” компьютерам (их у нас 20) номера из диапазона 192.168.1.1 – 192.168.1.20, а во второй подсети для оставшихся 10 компьютеров раздаем номера из диапазона 192.168.2.1 – 192.168.2.10.

Если ваш компьютер подключен к локальной сети или интернет, вы можете узнать его IP-адрес и маску подсети уже знакомым нам способом:

1. Зайдите в “Пуск” – “Выполнить” – наберите cmd и нажмите “ОК”.

2. В открывшемся окне введите команду ipconfig /all и нажмите клавишу Enter.

Номер сети может быть выбран администратором произвольно, либо назначен по рекомендации специального подразделения Интернет (Network Information Center – NIC), если сеть должна работать как составная часть Интернет. Обычно интернет-провайдеры получают диапазоны адресов у подразделений NIC, а затем распределяют их между своими абонентами. Это внешние IP-адреса (доступные из интернета), например 90.188.125.200.

Для локальных сетей зарезервированы *внутренние* IP-адреса (к ним нельзя получить доступ через интернет без специального ПО) из диапазонов:

192.168.0.1 – 192.168.254.254

10.0.0.1 – 10.254.254.254

172.16.0.1 – 172.31.254.254

Из этих диапазонов вы, как системный администратор, и будете назначать адреса компьютерам в вашей локальной сети. Если вы “жестко” зафиксируете IP-адрес в настройках компьютера, то такой адрес будет называться *статическим* – это постоянный, неизменяемый IP-адрес ПК.

Существует и другой тип IP-адресов – динамические, которые изменяются при каждом входе компьютера в сеть. За управление процессом распределения динамических адресов отвечает служба DHCP.

DHCP (англ. Dynamic Host Configuration Protocol — протокол динамической настройки узла) — сетевой протокол, позволяющий компьютерам автоматически получать IP-адрес и другие параметры, необходимые для работы в сети TCP/IP. Данный протокол работает по модели «клиент-сервер».

Имя сетевого компьютера

Помимо *физического* и *сетевого* адресов компьютер может также иметь *символьный* адрес – **имя компьютера**. Имя компьютера – это более удобное и понятное для человека обозначение компьютера в сети. Различают:

- NetBIOS имена;
- и полные **доменные** имена компьютеров.

Имена NetBIOS используются в одноранговых локальных сетях, в которых компьютеры организованы в рабочие группы. **NetBIOS** – *протокол для взаимодействия программ через компьютерную сеть*. Протокол NetBIOS распознает обычные буквенные имена компьютеров и отвечает за передачу данных между ними. Проводник Windows для просмотра локальной сети предоставляет папку **Сетевое окружение**, автоматически отображающей имена NetBIOS компьютеров вашей локальной сети.

Имя NetBIOS может содержать не более 15 символов и должно быть на английском языке.

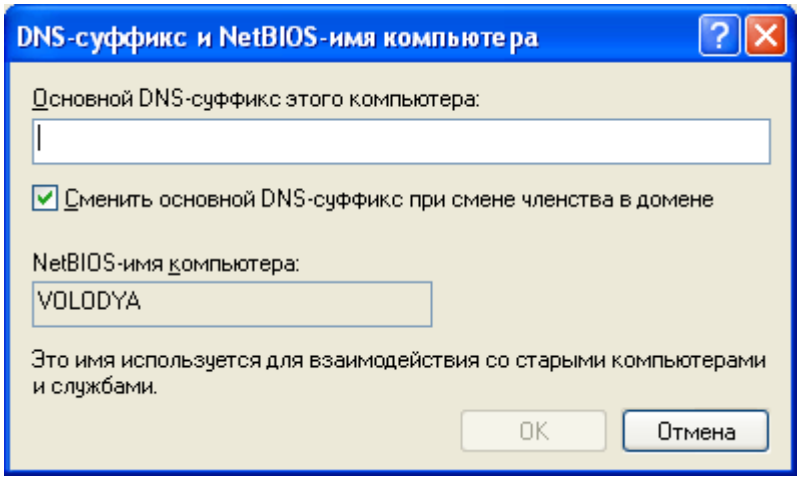
Чтобы узнать NetBIOS-имя вашего компьютера выполните следующие действия:

1. Щелкните правой кнопкой мыши по значку “Мой компьютер” на рабочем столе – выберите “Свойства”.

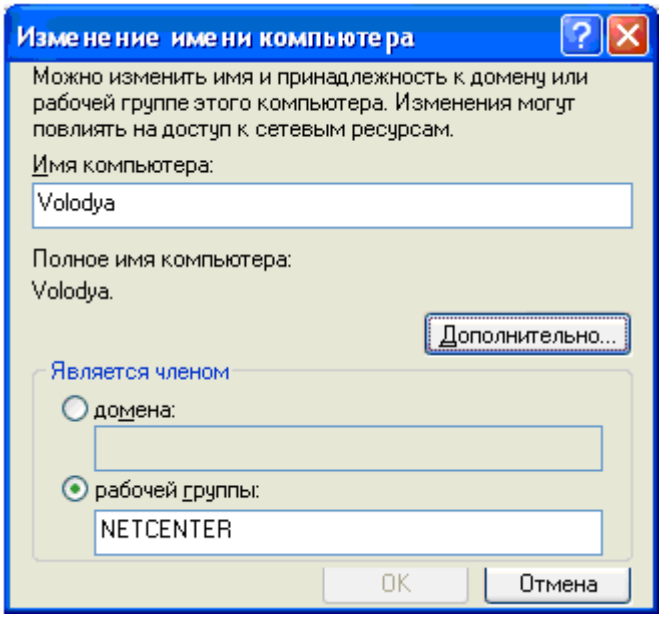
2. Перейдите на вкладку “Имя компьютера”.

3. Нажмите кнопку “Изменить” – затем “Дополнительно”.

4. Найдите строку “NetBIOS-имя компьютера”. Ниже и будет указано имя вашего ПК:



5. Нажмите кнопку “Отмена”. В окне “Изменение имени компьютера” вы можете изменить NetBIOS-имя в поле “Имя компьютера”:



В крупных иерархических сетях на базе домена используются полные **доменные имена** компьютеров, например, **webserver.ibm.com**.

Доменное Имя (англ. domain name) — уникальный идентификатор, который присваивается определенному IP-адресу (двух одинаковых быть не может). Доменное Имя это буквенный адрес компьютера. Доменное имя - это уникальное сочетание символов латинского алфавита, по которому можно идентифицировать ваш сайт среди множества других. Кроме букв, в домен могут входить цифры от 1 до 9 и символы дефиса «-», но дефис не может находиться в начале и в конце домена. Длина домена может быть от 2 до 63 символов.

Доменное имя компьютера состоит из трех составляющих:

- первая часть – имя хоста (webserver). Хост (host) - главный компьютер; ведущий узел (в сети); сервер;

- вторая – имя домена компании (ibm),

- третья – имя домена страны (например, ru – Россия) или имя одного из специальных доменов, обозначающих принадлежность домена организации к одному из профилей деятельности (com, gov, edu).

Доменное имя компьютера схематически выглядит:

ИМЯ КОМПЬЮТЕРА и **ДОМЕН, В КОТОРОМ ОНО НАХОДИТСЯ**.

Например, www.roga-kopyta.msk.ru:

имя www и находится в домене roga-kopyta.msk.ru. Точно так же домен roga-kopyta.msk.ru раскладывается дальше.

Для того, чтобы конкретному цифровому IP-адресу поставить в соответствие символьное доменное имя вашего сайта, существуют специальные DNS-серверы.

DNS-сервер - программа, осуществляющая преобразование доменного имени в цифровой IP-адрес и наоборот. В памяти этих серверов хранятся обширные таблицы, в которых каждому доменному имени ставится в соответствие IP-адрес. Помните, что одному IP-адресу могут соответствовать множество доменных имен!

У официального сайта Яндекс.Почта (компании Яндекс) Доменное Имя mail.yandex.ua соответствующее IP-адресу 213.180.204.25 (если IP-адрес написать в адресной строке, то браузер откроет сайт компании Яндекс).

Доменные Имена обслуживаются и централизованно администрируются набором серверов доменных имен DNS. DNS (Domain Name Service) — служба доменных имен. Наряду с цифровыми адресами, DNS позволяет использовать собственные имена компьютеров, так называемые Доменные Имена.

Вся информация о Доменных Именах хранится в центральной базе данных DNS, представляющей собой несколько мощных компьютеров, разбросанных по всему миру. В этой базе хранится информация о дате регистрации, о физическом или юридическом владельце Доменного Имени, а также путь к так называемому серверу имен — NAMESERVER, где содержится информация, на которую указывает Доменное Имя.

Единый каталог Internet, определяющий основу DNS, находится в государственной организации SRI International - Menlo Park, CA, US (Менло Парк, Калифорния, США).

Однако, все таки уникальным и полным адресом сайта является его URL (Uniform Resource Locator), который состоит из трех основных элементов: Протокол + Доменное имя + Путь/Файл . Как видно, что доменное имя сайта это только часть его полного адреса в Интернете.

Доменное Имя или буквенный адрес компьютера может быть:

доменное имя первого (верхнего) уровня — first level domain;

доменное имя второго уровня — second level domain;

доменное имя третьего уровня — third level domain.

Доменные Имена первого уровня подразделяются на:

- **домены общего пользования**, они могут устанавливают принадлежность сайта к определенной категории или виду деятельности:

.com - коммерческие ресурсы

.net - ресурсы и организации связанные с сетью

.org - некоммерческие организации

.info - информационные узлы

.biz - ресурсы для бизнеса

.arpa - домен используемый исключительно для инфраструктуры интернета

.edu - ВУЗ-ы (в частности вузы США)

.int - международные организации

.gov - правительственные организации США

.mil - военные ведомства США

.museum - музеи, учреждения и частные лица, имеющие отношение к музейному делу

.travel - индустрия путешествий, экскурсий и отдыха: турагентства, туроператоры.

- **национальные или географические домены**, они определяют принадлежность сайта к той или иной стране или географической территории:

Домен **.ru** принадлежит России - Russia (Россия)

.ca - Canada (Канада)

.de - Germany (Германия)

.fr - France (Франция)

.se - Sweden (Швеция)

.uk - United Kingdom (Великобритания)

.ua - Украина (Ukraine)

Украинские поисковые системы: http://meta.ua; http://www.ukr.net; http://www.google.com.ua

Доменное имя первого уровня выглядит так — www.ru (Российская зона интернета)

Доменное именя второго уровня - likbez-net.ru. Здесь .ru - это домен верхнего уровня. Собственное имя web-сайта – likbez-net находится на втором месте от окончания полного имени. Поэтому **likbez-net** является доменом второго уровня в зоне **.ru**.

Доменное имя третьего уровня - forum.likbez-net.ru. **forum** является доменом третьего уровня в зоне **likbez-net.ru**.

Логический сегмент сети

Широко практикуется разделение сети, основанной на протоколе IP, на логические сегменты, или логические подсети. Для этого каждому сегменту выделяется диапазон адресов, который задается адресом сети и сетевой маской. Например (в CIDR записи):

192.168.1.0/24, 192.168.2.0/24, 192.168.3.0/24 и т. д. — в каждом сегменте до 254 узлов

192.168.0.0/25, 192.168.128.0/26, 192.168.172.0/27 — в сегментах до 126, 62, 30 узлов соответственно

Логические подсети соединяются с помощью маршрутизаторов.

Мост (bridge), а также его быстродействующий функциональный аналог - коммутатор (switching hub), делит общую среду передачи данных на логические сегменты. Логический сегмент образуется путем объединения нескольких физических сегментов (отрезков кабеля) с помощью одного или нескольких концентраторов. Каждый логический сегмент подключается к отдельному порту моста/коммутатора. При поступлении кадра на какой-либо из портов мост/коммутатор повторяет этот кадр, но не на всех портах, как это делает концентратор, а только на том порту, к которому подключен сегмент, содержащий компьютер-адресат.

Разница между мостом и коммутатором состоит в том, что мост в каждый момент времени может осуществлять передачу кадров только между одной парой портов, а коммутатор одновременно поддерживает потоки данных между всеми своими портами. Другими словами, мост передает кадры последовательно, а коммутатор параллельно.

Мосты используются только для связи локальных сетей с глобальными, то есть как средства удаленного доступа,

При работе коммутатора среда передачи данных каждого логического сегмента остается общей только для тех компьютеров, которые подключены к этому сегменту непосредственно. Коммутатор осуществляет связь сред передачи данных различных логических сегментов. Он передает кадры между логическими сегментами только при необходимости, то есть только тогда, когда взаимодействующие компьютеры находятся в разных сегментах.

Разделение сети на логические сегменты дает выигрыш в производительности - трафик локализуется в пределах групп, и нагрузка на их разделяемые кабельные системы существенно уменьшается.

Физическое разделение

Как правило, физический сегмент сети ограничен сетевым устройством, обеспечивающим соединение узлов сегмента с остальной сетью:

Мосты или коммутаторы (2-й уровень в модели OSI)

Маршрутизаторы (3-й уровень в модели OSI)

Физический сегмент сети является доменом коллизий. Устройства, работающие на первом уровне модели OSI (повторители или концентраторы), домен коллизий не ограничивают.

ЛОГИЧЕСКАЯ СТРУКТУРА СЕТИ

Физическая структуризация сети полезна во многих отношениях, однако, в ряде случаев, обычно относящихся к сетям большого и среднего размера, невозможно обойтись без логической структуризации сети. Наиболее важной проблемой, не решаемой путём физической структуризации, остаётся проблема перераспределения передаваемого трафика между различными физическими сегментами сети.

В большой сети естественным образом возникает неоднородность информационных потоков, так как сеть состоит из множества подсетей рабочих групп, отделов, филиалов предприятия и других административных образований. Очень часто наиболее интенсивный обмен данными наблюдается между компьютерами, принадлежащими к одной подсети, и только небольшая часть обращений происходит к ресурсам компьютеров, находящихся вне локальных рабочих групп. (*До недавнего времени такое соотношение трафиков не подвергалось сомнению, и был даже сформулирован эмпирический закон «80/20», соответственно которому в каждой подсети 80% трафика является внутренним и только 20% - внешним*). Сейчас характер нагрузки сетей во многом изменился, широко внедряется технология *internet*, на многих предприятиях имеются централизованные хранилища корпоративных данных, активно используемые всеми сотрудниками предприятия. Всё это не могло не повлиять на распределение информационных потоков. В настоящее время не редки ситуации, когда интенсивность внешних обращений выше интенсивности обмена между «соседними» машинами. Тем не менее, независимо от того, в какой пропорции распределяются внешний и внутренний трафик, для повышения эффективности работы сети неоднородность информационных потоков необходимо учитывать.

Сеть с типовой топологией (шина, кольцо, звезда), в которой все физические сегменты развиваются в качестве одной распределяемой среды, оказывается неадекватной структуре информационных потоков в большой

сети. Например, в сети с общей шиной взаимодействие любой пары компьютеров занимает её на всё время обмена, поэтому при увеличении числа компьютеров в сети шина становится узким местом.

Сети, построенные на основе концентраторов

На рисунке 1 показана сеть, построенная с использованием концентраторов. Пусть компьютер А, находящийся в одной подсети с компьютером В, посылает ему данные. Несмотря на разветвлённую физическую структуру сети, концентраторы распространяют любой кадр по всем её сегментам. Поэтому кадр, посылаемый компьютером А компьютеру В, хотя и не нужен отделам 2 и 3, в соответствии с логикой работы концентраторов поступает на эти сегменты тоже. И до тех пор, пока компьютер В не получит адресованный ему кадр, не один из компьютеров этой сети не сможет передавать данные.

Такая ситуация возникает из-за того, что логическая структура данной сети осталась однородной (рис. 2) – она никак не учитывает увеличение интенсивности трафика внутри отдела и предоставляет всем парам компьютеров разные возможности по обмену информацией.

Приведенные на рисунках схемы показывают противоречие между логической структурой сети и структурой информационных потоков.

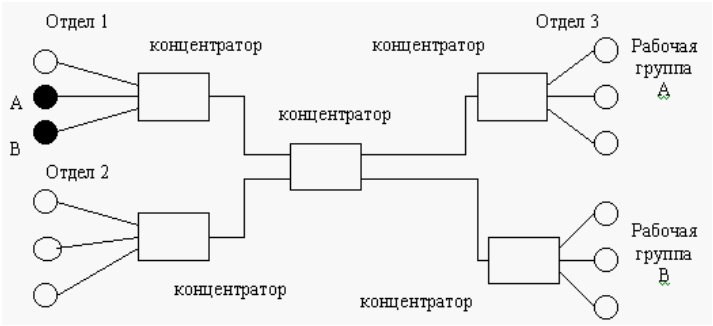


Рис. 1 Физическая структуризация с помощью концентраторов

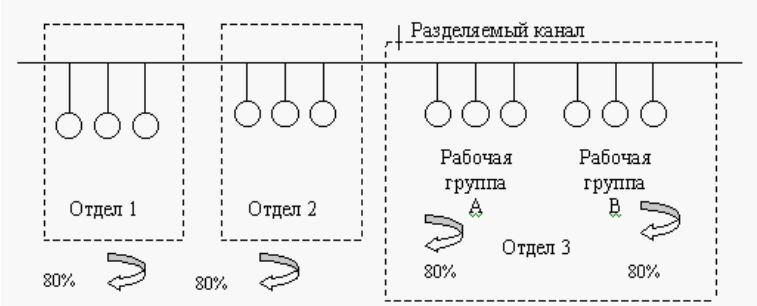


Рис. 2 Логическая структура сети

Решение проблемы состоит в отказе от идеи однородной единой разделяемой среды. Например, в рассмотренном выше примере желательно бы было сделать так, чтобы кадры, которые передаёт компьютер одела 1,

выходили бы за пределы этой сети только том случае, если эти кадры направлены какому-либо компьютеру из других отделов. С другой стороны, в сеть каждого из отделов должны попадать только те кадры, которые адресованы узлам этой сети. При такой организации работы сети её производительность существенно повысится, так как компьютеры одного отдела не будут простаивать в то время, когда обмениваются данными компьютеры других отделов.

Нетрудно заметить, что в предложенном решении мы отказались от идеи общей разделяемой среды в пределах всей сети, хотя и оставили её в пределах каждого отдела. Пропускная способность линий связи между отделами не должна совпадать с пропускной способностью среды внутри отделов. Если трафик между отделами составляет только 20% трафика внутри отдела (как уже отмечалось эта величина может быть другой), то и пропускная способность линий связи и коммуникационного оборудования, соединяющего отделы, может быть значительно ниже внутреннего трафика сети отдела.

Распространение трафика, предназначенного для компьютеров некоторого сегмента сети, только в пределах этого сегмента, называется локализацией трафика. Логическая структуризация сети – это процесс разбиения сети на сегменты с локализованным трафиком.

Для логической структуризации сети используются также коммуникационные устройства, как мосты, коммутаторы, маршрутизаторы и шлюзы.

Сети, построенные на основе мостов

Мост (*bridge*) делит разделяемую среду передачи сети на части (называемые логическими сегментами), передавая информацию из одного сегмента в другой, только в том случае, если такая передача действительно необходима, то есть если адрес компьютера назначения принадлежит другой сети. Локализация трафика не только экономит пропускную способность, но и уменьшает возможность несанкционированного доступа к данным, так как кадры не входят за пределы своего сегмента и естественно их сложнее перехватить злоумышленнику.

На рисунке 3 показана сеть, которая была получена из сети с центральным концентратором путём его замены на мост. Сети 1-го и 2-го отделов состоят из отдельных логических сегментов, а сеть отдела 3 – из двух логических сегментов. Каждый логический сегмент построен на базе концентратора и имеет простейшую физическую структуру, образованную отрезками кабеля, связывающими компьютеры с портами концентратора.

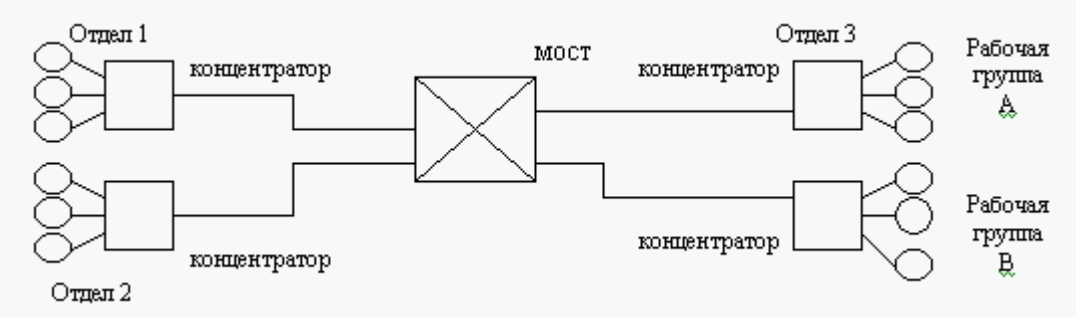


Рис. 3 Структуризация сети с помощью моста

Мосты используют для локализации трафика аппаратные адреса компьютеров. Это затрудняет распознавание принадлежности того или иного компьютера к определённому логическому сегменту – сам адрес не содержит никакой информации. Поэтому мост достаточно упрощённо представляет деление сети на сегменты – он запоминает, через какой порт на него поступил кадр данных от каждого компьютера сети, и в дальнейшем передаёт кадры, предназначенные для этого компьютера, на этот порт. Точной топологией связей между логическими сегментами мост не знает. Из-за этого применения мостов приводит к значительным ограничениям на конфигурацию связей сети, так как сегменты должны быть соединены таким образом, чтобы в сети не образовывались замкнутые контуры.

Сети, построенные на основе коммутаторов

Коммутатор (switch, switching hub) по принципу обработки кадров ничем не отличается от моста. Основное его отличие от моста состоит в том, что он является своего рода коммуникационным процессором, который обрабатывает кадры по алгоритму моста не зависимо от процессоров других портов. За счёт этого общая производительность коммутатора обычно на много выше производительности традиционного моста, имеющего один процессорный блок. Можно сказать, что коммутаторы – это мосты нового поколения, которые обрабатывают кадры в параллельном режиме.

Сети, построенные на основе маршрутизаторов

Ограничения, связанные с применением мостов и коммутаторов – по топологии связей, а также ряд других, - привели к тому, что в

ряду коммуникационных устройств появился ещё один тип оборудования – *маршрутизатор (router).* Маршрутизаторы более надёжно и более эффективно, чем мосты, изолируют трафик отдельных частей сети друг от друга. Маршрутизаторы образуют логические сегменты посредством явной адресации, поскольку используют не простые аппаратные, а составные числовые адреса. В этих адресах имеется поле номера сети, так что все компьютеры, у которых значение этого поля одинаково, принадлежат к одному сегменту, называемому в данном случае *подсетью (subnet).*

Кроме локализации трафика маршрутизаторы выполняют ещё много других полезных функций. Так, маршрутизаторы могут работать в сети с замкнутыми контурами, при этом они осуществляют выбор наиболее рационального маршрута из нескольких возможных. Сеть, представленная на рисунке 4 отличается от своей предшественницы тем, что между подсетями отделов 1 и 2 проложена дополнительная связь, которая может использоваться как для повышения производительности сети, так и для повышения её надёжности.

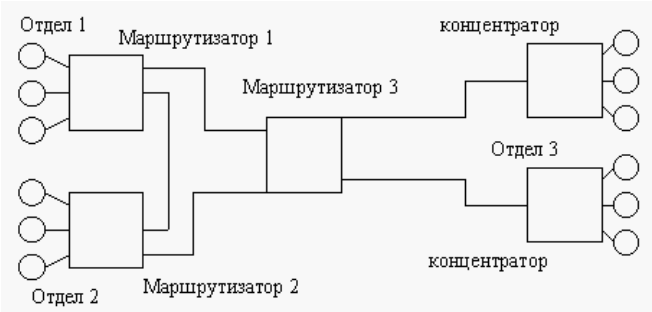


Рис. 4 Структуризация сети с помощью маршрутизаторов

Другой очень важной функцией маршрутизаторов является их способность связывать в единую сеть подсети, построенные с использованием разных сетевых технологий, например.

Сети, построенные на основе шлюзов

Кроме перечисленных устройств отдельные части сети может соединять *шлюз (gateway).* Обычно основной причиной, по которой в сети используют шлюз, является необходимость объединить сети с разными типами системного и прикладного программного обеспечения, а не желание локализовать трафик.

Тем не менее, шлюз обеспечивает и локализацию трафика в качестве некоторого побочного эффекта.

Крупные сети практически никогда не строятся без логической структуризации. Для отдельных сегментов и подсетей характерны типовые однородные топологии базовых технологий, и для их объединения всегда используется оборудование, обеспечивающее локализацию трафика, - мосты, коммутаторы, маршрутизаторы и шлюзы.

Сетевые службы

Для конечного пользователя сеть – это не компьютеры, кабели и концентраторы и даже не информационные потоки, для него сеть – это, прежде всего, тот набор служб, с помощью которых он получает возможность просмотреть список имеющихся в сети компьютеров, прочитать удалённый файл, распечатать документ на «чужом» принтере или послать почтовое сообщение. Именно совокупность предоставляемых возможностей – насколько широк их выбор, насколько они удобны, надёжны и безопасны – определяет для пользователя облик той или иной сети.

Кроме собственно обмена данными, сетевые службы должны решать и другие, более специфические задачи, например, задачи, порождаемые распределённой обработкой данных. К таким задачам относится обеспечение непротиворечивости нескольких копий данных, размещённых на разных машинах (служба репликации), или организация выполнения одной задачи параллельно на нескольких машинах сети (служба вызова удалённых процедур). Среди сетевых служб можно выделить административные, то есть такие, которые в основном ориентированы не на простого пользователя, а на администратора, и служат для организации правильной работы сети в целом. Служба администрирования учётных записей о пользователях, которая позволяет администратору вести общую базу данных о пользователях сети, система мониторинга сети, позволяющая захватывать и анализировать сетевой трафик, служба безопасности, в функции которой может входить среди прочего выполнение процедуры логического входа с

последующей проверкой пароля, - всё это примеры административных служб.

Реализация сетевых служб осуществляется программными средствами. Основными службами являются:

– файловая служба и служба печати, которые обычно предоставляются сетевой операционной системой
– служба баз данных, факса или передачи голоса являются системными сетевыми приложениями или утилитами, работающими в тесном контакте с сетевой ОС.

Вообще говоря, распределение служб между ОС и утилитами достаточно условно и меняется в конкретных реализациях ОС.

При разработке сетевых служб приходится решать проблемы, которые свойственны любым распределённым приложениям: определение протокола взаимодействия между клиентской и сервисной частями, распределение функций между ними, выбор схемы адресации приложений и др.

Одним из главных показателей качества сетевой службы является её удобство. Для одного и того же ресурса может быть разработано несколько служб, по-разному решающих одну и ту же задачу. Отличия могут заключаться в производительности или уровне удобства предоставляемых услуг. Например, файловая служба может быть основана на использовании команды передачи файла из одного компьютера в другой по имени файла, а это требует от пользователя знания имени нужного файла. Та же файловая служба может быть реализована и так, что пользователь монтирует удалённую файловую систему к локальному каталогу, а далее обращается к удалённым файлам как к своим собственным, что гораздо более удобно. Качество сетевой службы зависит и от качества пользовательского интерфейса – интуитивной понятности, наглядности, рациональности.

При определении степени удобства разделяемого ресурса часто употребляют термин «прозрачность». *Прозрачный доступ* – это такой доступ, при котором пользователь не замечает, где расположен нужный ему ресурс – на его компьютере или на удалённом. После того, как он смонтировал удалённую файловую систему в своё дерево каталогов, доступ к удалённым файлам

становится для него совершенно прозрачным. Сама операция монтирования также может иметь разную степень прозрачности – в сетях с меньшей прозрачностью пользователь должен знать и задавать в команде имя компьютера, на котором хранится удалённая файловая система, в сетях с большей степенью прозрачности соответствующий программный компонент SET производит поиск разделяемых томов файлов безотносительно мест их хранения, а затем предоставляет их пользователю в удобном для него виде, например, в виде списка или набора пиктограмм.

Для обеспечения прозрачности важен способ адресации (именования) разделяемых сетевых ресурсов. Имена разделяемых сетевых ресурсов не должны зависеть от их физического расположения на том или ином компьютере. В идеале пользователь не должен ничего менять в своей работе, если администратор переместил том или каталог с одного компьютера на другой. Сам администратор и сетевая операционная система имеют информацию о расположении файловых систем, но от пользователя она скрыта. Такая степень прозрачности пока редко встречается в сетях, - обычно для получения доступа к ресурсам определённого компьютера сначала приходится устанавливать с ним логическое соединение. Такой подход применяется, например, в сетях Windows NT, 2000, XP.

Контрольные вопросы

№ п.п.	Вопрос	Ответ
1	Физическая структура (топология) сети – это:	Схема соединения компьютеров в сети с использованием дополнительных технических средств.
2	Логическая структура сети – это:	Программно - аппаратная структура, обеспечивающая перераспределение передаваемого трафика между различными физическими сегментами сети
3	Однородная структура сети – это:	Структура, обеспечивающая равные права доступа к ресурсам всех пользователей
4	Локализация трафика – это:	Распространение трафика, предназначенного для компьютеров некоторого сегмента сети, только в пределах этого сегмента
5	Логическая структуризация сети – это:	Процесс разбиения сети на сегменты с локализованным трафиком
6	Концентраторы – это:	Устройства обмена информацией, не использующие аппаратные адреса компьютеров для локализации трафика

		внутри локальной сети
7	Мосты – это:	Устройства обмена информацией, использующие аппаратные адреса - порты для локализации трафика внутри локальной сети
8	Коммутаторы – это:	Устройства обмена информацией, имеющие контроллер и микропрограмму идентификации адреса - порта для локализации трафика внутри локальной сети
9	Маршрутизаторы – это:	Интеллектуальные устройства обмена информацией, имеющие собственный процессор и программное обеспечение идентификации компьютера в локальной сети, а также обеспечивающие объединение локальных сетей в единую сеть
10	Шлюз – это:	Коммуникационный процессор, обеспечивающий объединение сетей с различными операционными системами и прикладным программным обеспечением в единую сеть
11	Непротиворечивость нескольких копий данных, размещённых на разных машинах обеспечивает	Служба репликации
12	Организацию выполнения одной задачи параллельно на нескольких машинах обеспечивает	Служба вызова удалённых процедур
13	Организацию правильной работы сети в целом, включая ее безопасность обеспечивает	Административная служба
14	Реализацию сетевых служб обеспечивают	Программные средства
15	Файловую службу и службу печати реализует	Сетевая операционная система
16	Доступ к информации, при котором пользователь не замечает, где расположен нужный ему ресурс – это:	Прозрачный доступ

Протоколы передачи данных

Протоколы представляют собой *набор условий (правил), которые регламентируют формат и процедуры обмена информацией* между двумя или несколькими независимыми устройствами или процессами. Протокол имеет три важнейших элемента: *синтаксис, семантику и синхронизацию* (timing). *Синтаксис* протокола определяет поля, например, 16-байтовое поле для адресов, 32-байтовое поле для контрольных сумм и 512 байт на пакет. *Семантика* протокола придает этим полям значение, например, если адресное поле состоит из всех адресов, это "широковещательный" пакет. *Синхронизация* - это количество битов в секунду, т.е. скорость передачи данных. Она важна не только на самых низких уровнях протокола, но и на высших.

На рисунке 1 показан типичный формат сообщения. В начале сообщения, передаваемого в сети, присваиваются символы синхронизации, с той целью,

чтобы другой узел в сети мог увидеть, что "приходит" сообщение и смог синхронизировать приемник с передатчиком. Заголовок сообщения содержит информацию об адресе (откуда и куда поступает сообщение). Текст сообщения - это сама информация, посылаемая по сети. Он имеет заголовок и иногда концевик, показывающий, где заканчивается сообщение. В конце сообщения также могут быть символы управления и синхронизации.

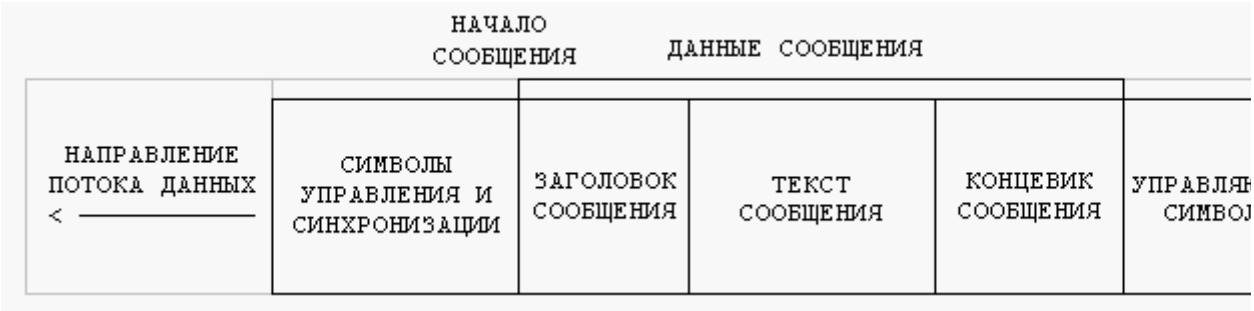


Рис. 1 Формат сообщения

Сложные системы передачи данных не используют один протокол для решения всех задач передачи. Вместо этого, им требуется набор взаимодействующих протоколов. Ярким примером такого взаимодействия служит стек протоколов TCP/IP.

Разделение протоколов на уровни

Для обоснования необходимости разделения протоколов на уровни можно привести ошибки, которые могут возникнуть, когда машины взаимодействуют в сети между собой.

Сбой оборудования характеризуется такими факторами, при которых, сервер или шлюз может прекратить работать либо из-за аварии оборудования, либо из-за краха операционной системы. Канал передачи данных может перестать работать или может быть неожиданно отключен. В связи с этим фактом протокольному ПО, необходимо распознавать такие сбои и восстанавливаться, если это возможно, после их возникновения.

Перегрузка сети возникает в связи с тем, что сеть имеют конечную пропускную способность. Поэтому протокольному ПО необходимо знать способы, которыми перегруженная машина может восстановить остальной трафик.

Иногда возникают случаи *задержки* или полной *потери передаваемых пакетов*. Поэтому протокольному ПО необходимо определять такие ошибки и адаптироваться к большим задержкам при передаче данных.

Ошибки, в передаваемых данных, могут возникать из - за электрических либо электромагнитных помех или сбоев в оборудовании. Поэтому протокольному ПО необходимо идентифицировать такие ошибки и восстанавливаться после их возникновения.

Дублирование данных (*пакетов*) или нарушение их последовательности могут возникать в случаях обмена данными, выполняемыми по нескольким путям. В связи с такими фактами протокольному ПО необходимо переупорядочивать пакеты и удалять их дубли.

Для устранения этих проблем протоколы подразделяют по уровням.

Семиуровневая справочная модель Взаимодействия Открытых Систем (ВОС)

В этой области доминируют две идеи относительно разделения протоколов на уровни. Первая, основывающаяся на работе, выполненной Международной Организацией по Стандартизации (МОС), известной как Справочная Модель ВОС (Взаимодействия Открытых Систем), часто называется моделью ВОС. Модель ВОС, содержит семь концептуальных уровней (рис. 2).

Уровень	*Возможности*
7	Прикладной
6	Представительный
5	Сеансовый
4	Транспортный
3	Сетевой
2	Канальный (аппаратный интерфейс)
1	Физический

Рис. 2 Семиуровневая справочная модель ВОС

Примечание*: Модель ВОС, созданная для описания протоколов одной сети, не содержит специального уровня для межсетевой маршрутизации, имеющегося в стеке протоколов TCP/IP.*

Протокол X25 и его связь с моделью ВОС

Схема разделения на уровни ВОС, хотя и разрабатывалась как концептуальная модель, а не руководство для реализаций, является основой для нескольких реализаций протоколов. Среди протоколов, связанных с моделью ВОС, существует набор протоколов, известный как X.25, который является самым популярным и широко используемым. Протокол X.25 был создан как рекомендация Международного Консультативного Комитета по Телефонии и Телеграфии (МККТТ), международной организации, вырабатывающей стандарты для международных телефонных служб. Протокол X.25 используется сетями передачи данных общего пользования в Европе и США.

С точки зрения протокола X.25, сеть представляется во многом аналогично телефонной системе. Сеть предполагается состоящей из сложных коммутаторов - пакетов, имеющих достаточный интеллект для реализации маршрутизации пакетов. Обслуживающие вычислительные

машины (серверы) не присоединены напрямую к каналам сети. Вместо этого каждый сервер присоединен к одному из коммутаторов пакетов, который использует последовательную линию взаимодействия с другими коммутаторами. Таким образом, соединение между пакетным коммутатором и сервером представляет миниатюрную сеть, состоящую из одной последовательной линии. Сервер реализует довольно сложную процедуру для передачи пакетов в сеть.

Физический уровень

Протокол X.25 определяет *стандарт для физического соединения между сервером и сетевыми коммутаторами пакетов, а также процедуры, используемые для передачи пакетов от одной машины к другой*. В справочной модели уровень 1 определяет физическое соединение, включая электрические параметры, такие как напряжение и ток. Соответствующий протокол, X.21, содержит его детальное описание, и используется сетями передачи данных общего пользования.

Канальный уровень

Уровень 2 в протоколе X.25 определяет, *каким образом данные передаются между сервером и пакетным коммутатором*. Протокол X.25 использует термин *кадр* для обозначения *элементарного блока данных*. Так как коммутационное оборудование доставляет только поток бит, протокол уровня 2 должен определить формат кадров и указать, как две ЭВМ будут определять границы кадров. Так как ошибки передачи могут разрушить данные, протокол уровня 2 включает процедуры обнаружения и обработки ошибок, используя контрольную сумму кадра. Для обеспечения надежности передачи кадра протокол уровня 2 реализует обмен подтверждениями, позволяющий двум машинам определять, когда кадр успешно передан. На втором уровне наиболее часто используется протокол *HDLC* (*Высокоуровневое Взаимодействие по Каналу Данных*).

Сетевой уровень

Справочная модель ВОС определяет, что третий уровень, *называемый сетевым уровнем или уровнем коммуникационной подсети*, обеспечивает возможности, описывающие завершение взаимодействия между сервером и сетью. Данный уровень определяет *базовый элемент, передающийся по сети, и включает понятия адресации назначения и маршрутизации к назначению*. В протоколе X.25 взаимодействие между сервером и коммутатором пакетов концептуально отделено от трафика. Поэтому, протоколы уровня 3 могут допускать обмен пакетами большего размера, чем пакеты уровня 2. Протокольное ПО уровня 3 собирает пакет в той форме, которую ожидает сеть, и использует уровень 2 для передачи пакета (возможно по частям) к

коммутатору пакетов. Уровень 3 также должен решать проблему переполнения сети.

Транспортный уровень

Уровень 4, протокола X.25 обеспечивает *сквозную надежность с помощью прямого взаимодействия ЭВМ-источника с ЭВМ-получателем*. Основная идея заключается в том, что хотя нижние уровни и обеспечивают надежность передачи каждого элемента, уровень *межконцевого* взаимодействия дублирует их. Этим достигается проверка работоспособности всех промежуточных машин, участвующих в трафике.

Сеансовый уровень

Более высокие уровни модели МОС описывают, *каким образом может быть организовано протокольное ПО для предоставления всех возможностей прикладной программе*. Комитет МОС считает проблему удаленных терминалов настолько важной, что выделил уровень 5 для ее решения. Фактически, главным средством, предоставляемым сетями передачи данных, является установление соединения между терминалом (ЭВМ - клиент) и удаленным сервером, через специальную программу – сервер или аппаратный сервер, которые называются СБОРЩИК-РАЗБОРЩИК ПАКЕТОВ (PAD). Таким образом, пользователи, имеющие свой собственный терминал и модем, устанавливают соединение посредством локального PAD - а с сервером.

Представительный уровень

Уровень 6 МОС разработан, чтобы включать в себя функции, требуемые многим прикладным программам, использующим сеть. Типичным примером являются стандартные процедуры, сжимающие текст или преобразующие графические образы для их передачи по сети. Хотя этот уровень еще не доработан, в последние годы проводились серьезные работы по его расширению, в результате которых появился Стандарт МОС, известный как Нотация Абстрактного Синтаксиса 1 (ASN1), обеспечивает представление данных, используемых прикладными программами.

Прикладной уровень

Самый верхний уровень МОС – уровень 7 включает прикладные программы (приложения), использующие сеть. Типичными примерами таких приложений являются утилиты электронной почты или передачи файлов. В частности, МККТТ рекомендовал протокол для электронной почты, называемый X.400 или X.400(1988). Фактически, МОС и МККТТ совместно разработали систему обработки сообщений, версия МОС которой, называется MOTIS.

Модель уровней Интернет TCP/IP

Вторая основная модель разделения протоколов на уровни не была разработана комитетом по стандартам, а появилась в результате исследований, приведших к появлению стека протоколов TCP/IP. После небольшой доработки модель МОС может быть приспособлена для описания схемы деления на уровни в протоколе TCP/IP. Тем не менее базовые предпосылки этих схем сильно различаются, что позволяет говорить об их различии.

На концептуальном уровне ПО протокола TCP/IP организовано в виде 4 уровней, опирающихся на пятый уровень оборудования. На рисунке 3 приведены концептуальные уровни, а также форма протокола, в которой передаются данные между уровнями.

Рис. 3 Четыре конептуальных уровня ПО TCP/1P и форма объектов, передаваемых между ними

Прикладной уровень

На самом верхнем уровне пользователи вызывают прикладные программы, которые обращаются к сервисам, доступным в среде Интернет. Приложение взаимодействует с протоколами транспортного уровня для передачи или приема данных. Каждая прикладная программа выбирает тип транспортировки, который ей требуется, либо последовательность отдельных сообщений, или непрерывный поток байт. Прикладная программа передает данные транспортному уровню в требуемой форме для доставки.

Транспортный уровень

Основной задачей транспортного уровня является обеспечение взаимодействия между прикладными программами. Такое взаимодействие часто называется межконцевое (*end-to-end*). Транспортный уровень может управлять потоком информации. Он может также обеспечивать надежную передачу, гарантируя, что данные поступили без ошибок и в порядке их передачи. Для этого протокол предусматривает обмен подтверждениями между передающей и принимающей сторонами, что позволяет повторно передавать потерянные пакеты. Транспортное ПО делит передаваемый поток данных на небольшие части, называемые пакетами. Транспортное ПО передает каждый пакет вместе с адресом назначения следующему уровню. Транспортный уровень обеспечивает возможность принимать данные от нескольких прикладных программ и посылать их более нижнему уровню. С этой целью протокол добавляет дополнительную информацию к каждому

пакету, включая коды, идентифицирующие прикладную программу, пославшую его, и прикладную программу - получатель, а также контрольную сумму пакета. Принимающая машина использует контрольную сумму для проверки целостности принятого пакета, а код назначения - для идентификации прикладной программы, которой он должен быть передан.

Межсетевой уровень

Межсетевой уровень управляет взаимодействием между машинами. Он принимает запрос на посылку пакета от транспортного уровня вместе с указанием машины, на которую этот пакет должен быть послан. Он инкапсулирует пакет в IP-дейтаграмме, заполняет заголовок дейтаграммы, использует алгоритмы маршрутизации для определения того, можно ли послать дейтаграмму напрямую, или следует послать ее шлюзу, и передает дейтаграмму соответствующему интерфейсу с сетью. Межсетевой уровень также обрабатывает приходящие дейтаграммы, проверяет их корректность, и использует алгоритм маршрутизации для того, чтобы решить, нужно ли обработать дейтаграмму локально или ее следует переправить дальше. Для дейтаграмм, адресованных локальной машине, ПО межсетевого уровня удаляет заголовок дейтаграммы и определяет, какой из транспортных протоколов будет обрабатывать пакет. Наконец, межсетевой уровень посылает сообщения об ошибках (*ICMP)* по мере необходимости и обрабатывает все приходящие сообщения *ICMP*.

Уровень интерфейса с сетью

ПО протокола TCP/IP самого низкого уровня состоит из уровня интерфейса с сетью, ответственного за прием IP-дейтаграмм и передачу их по конкретной сети. Интерфейс с сетью может состоять из драйвера устройства в случае локальной сети или сложной подсистемы, использующей свой протокол канального уровня в случае когда сеть состоит из коммутаторов пакетов, взаимодействующих с сервером, используя *HDLC*.

Различия между схемами X.25 и TCP/IP

Существует два важных различия между схемой разделения на уровни протоколов TCP/IP и X.25. Первое различие связано с реализацией надежности, а второе различие связано с местонахождением интеллектуальных функций в системе.

Надежность на канальном уровне и межконцевая надежность

Одно из основных различий между протоколами TCP/IP и X.25 состоит в их подходах к обеспечению сервиса надежной передачи данных. В модели X.25 протокольное ПО обнаруживает и обрабатывает ошибки на всех уровнях. На канальном уровне сложные протоколы гарантируют, что передача между сервером и пакетным коммутатором, к которому он присоединен, будет корректной. К каждому передаваемому элементу данных присоединяется

контрольная сумма, и получатель подтверждает каждый принятый фрагмент данных. Протокол канального уровня включает таймаут и алгоритм повторной передачи данных. Данные методы защищают данные от потерь и обеспечивают автоматическое восстановление данных после сбоев или рестартов оборудования.

В отличие от описанной схемы X.25 протокол TCP/IP основывает свое разделение на уровни на том, что надежность - это межконцевая проблема. Суть архитектуры заключается в том, чтобы сеть могла управляться с ожидаемой загрузкой, и позволить отдельным каналам или машинам терять данные или искажать их, не пытаясь исправлять ошибки. Фактически, большая часть TCP/IP уровня интерфейса с сетью не обеспечивает надежности. Вместо этого, большинство ошибок обрабатывает транспортный уровень.

Освобождение уровня интерфейса от верификации делает TCP/IP более легким для реализации. Промежуточные шлюзы могут отбрасывать дейтаграммы, ставшие испорченными из-за ошибок передачи. Они могут отбрасывать любые дейтаграммы, которые не могут быть доставлены. Они могут отбрасывать дейтаграммы, когда скорость их поступления превышает пропускную способность машины. Они могут посылать дейтаграммы по другим путям с меньшей или большей задержкой, не информируя об этом источник или назначение.

Наличие ненадежных каналов означает, что некоторые дейтаграммы не дойдут до назначения. Обнаружение и восстановление после потери дейтаграмм выполняется между источником и назначением, и поэтому называется *межконцевой верификацией*. Протоколы, находящиеся на транспортном уровне, использует контрольные суммы, подтверждения и таймауты для управления передачей.

Поэтому, в отличие от разделения на уровни в X.25, ориентированного на соединения, TCP/IP помещает большую часть управления надежностью на один уровень.

Местонахождение средств управления

Другое различие между моделью X.25 и TCP/IP появляется при определении местонахождения средств управления работой. Как правило, в сетях, использующих X.25, подразумевается, что сеть - это утилита, обеспечивающая транспортное средство. Производитель, предоставляющий средство, управляет доступом к сети и следит за трафиком для учета работы пользователей. Поставщик сетевого сервиса решает такие проблемы, как маршрутизация и управление потоком внутренним образом, делая процесс передачи данных надежным. При таком подходе на долю сервера мало что остается. Сеть - это сложная, независимая система, к которой могут присоединяться относительно простые ЭМВ, которые мало участвуют в работе сети.

В отличие от X.25, протокол TCP/IP требует от ЭВМ участия почти во всех сетевых протоколах. ЭВМ принимают участие в обнаружении ошибок и восстановлении после них маршрутизации, так как они должны выбрать шлюз при посылке дейтаграммы, они участвуют в управлении сетью, так как они должны обрабатывать управляющие сообщения ICMP. Поэтому, при сравнении с сетью X.25, Интернет TCP/IP может рассматриваться как относительно простая система доставки пакетов, к которой присоединены интеллектуальные ЭВМ, а не просто терминалы.

Примечание. *Протоколы Internet, история их создания и стек TCP/IP подробно рассматриваются в дисциплине Интернет технологии и ресурсы*.

Контрольные вопросы

№ п.п.	Вопрос	Ответ
1	Протоколы представляют собой	Набор условий (правил), которые регламентируют формат и процедуры обмена информацией между двумя или несколькими независимыми устройствами или процессами
2	Синтаксис протокола определяет	Поля данных, адресов и их контрольных сумм
3	Семантика протокола определяет	Значения полей
4	Синхронизация протокола обеспечивает	Скорость передачи данных
5	Первый уровень модели ВОС – это:	Физический уровень
6	Второй уровень модели ВОС – это:	Канальный уровень
7	Третий уровень модели ВОС – это:	Сетевой уровень
8	Четвертый уровень модели ВОС – это:	Транспортный уровень
9	Пятый уровень модели ВОС – это:	Сеансовый уровень
10	Шестой уровень модели ВОС – это:	Представительский уровень
11	Седьмой уровень модели ВОС – это:	Прикладной уровень
12	Физический уровень протокола X.25 определяет	Стандарт для физического соединения между сервером и сетевыми коммутаторами пакетов, а также процедуры, используемые для передачи пакетов от одной машины к другой
13	Канальный уровень протокола X.25 определяет	Каким образом данные передаются между сервером и пакетным коммутатором
14	Сетевой уровень протокола X.25 определяет	Базовый элемент, передающийся по сети, который включает адресацию назначения и маршрутизацию к назначению. Обеспечивает возможности, описывающие завершение взаимодействия между сервером и сетью.
15	*Транспортный уровень протокола X.25 обеспечивает*	Сквозную надежность с помощью прямого взаимодействия ЭВМ-источника с ЭВМ-получателем
16	*Сеансовый уровень протокола X.25 определяет*	Организацию протокольного программного обеспечения для предоставления всех возможностей прикладной программе
17	Представительский уровень протокола	Сжатие и распаковку передаваемой информации

	X.25 обеспечивает	
18	Прикладной уровнь протокола X.25 обеспечивает	Функционирование приложений, использующих сеть, таких как утилиты электронной почты и утилиты передаваемых файлов
19	Первый уровень протокола TCP/IP реализует	Интерфейс с сетью обеспечивающий передачу кадров между уровнями сети
20	Второй уровень протокола TCP/IP обеспечивает	Формирование и обмен дейтаграммами между уровнями сети
21	Третий уровень протокола TCP/IP обеспечивает	Формирование и обмен пакетами между уровнями сети
22	Четвертый уровень протокола TCP/IP обеспечивает	Объмен данными или сообщениями между уровнями сети

ОБЪЕКТ QUERY, ОСНОВЫ ЯЗЫКА SQL

Объект Query один из наиболее мощных и гибких компонентов, среды Delphi, предназначенных для управления данными. С его помощью создаются приложения, работающие с промышленными SQL серверами, такими, как: InrterBase, Oracle или Sybase и другими

Доступ к данным в объекте Query осуществляется посредством *SQL утверждений*. Сокращение *SQL* означает *Structured Query Language* - Язык Структурированных Запросов. *SQL* - это мощный язык управления базами данных, который легко доступен в среде *Delphi*, но который отличается от “родного” языка среды *Delphi*. В среде *Delphi SQL* утверждения используются для просмотра таблиц, объединения таблиц, создания отношений вида один – ко - многим, или выполнения практически любых действий, аналогичных действиям других инструментов доступа и управления данными.

Состав языка SQL

Язык *SQL* предназначен для манипулирования данными в реляционных базах данных, определения структуры баз данных и для управления правами доступа к данным в многопользовательской среде. В язык *SQL* в качестве составных частей входят:

- язык манипулирования данными (*Data Manipulation Language, DML*)
- язык определения данных (*Data Definition Language, DDL*)
- язык управления данными (*Data Control Language, DCL*).

Подчеркнем, что приведенные названия языков не являются самостоятельными языками, а представляют собой различные команды одного языка. Такое деление проведено только с целью различия функционального назначения команд.

Язык манипулирования данными используется, для манипулирования данными в таблицах баз данных. Он содержит 4 базовые команды:

SELECT – выбрать; ***INSERT*** – вставить; ***UPDATE*** – обновить; ***DELETE*** – удалить.

Язык определения данных используется для создания и изменения структур данных и их составных частей: таблиц, индексов, представлений (виртуальных таблиц), а также триггеров и сохраненных процедур. Основными его командами являются:

CREATE DATABASE - создать базу данных; ***CREATE TABLE*** - создать таблицу; ***CREATE VIEW*** - создать виртуальную таблицу; ***CREATE INDEX*** - создать индекс; ***CREATE TRIGGER*** - создать триггер; ***CREATE PROCEDURE*** - создать процедуру; ***ALTER DATABASE*** - модифицировать базу данных; ***ALTER TABLE*** - модифицировать таблицу; ***ALTER VIEW*** - модифицировать виртуальную таблицу; ***ALTER INDEX*** - модифицировать индекс; ***ALTER TRIGGER*** - модифицировать триггер; ***ALTER PROCEDURE*** - модифицировать процедуру; ***DROP DATABASE*** - удалить базу данных; ***DROP TABLE*** - удалить таблицу; ***DROP VIEW*** - удалить виртуальную таблицу; ***DROP INDEX*** - удалить индекс; ***DROP TRIGGER*** - удалить триггер; ***DROP PROCEDURE*** - удалить процедуру.

Язык управления данными используется для управления правами доступа к данным и выполнением процедур в многопользовательской среде. Более точно его можно назвать "*язык управления доступом*". Он состоит из двух основных команд: ***GRANT*** - дать права и ***REVOKE*** - забрать права.

С точки зрения прикладного интерфейса существуют две разновидности команд *SQL*: ***интерактивный SQL*** и ***встроенный SQL***.

Интерактивный SQL используется в специальных утилитах типа ***WISQL*** или ***DBD***. Интерактивный *SQL* позволяет в диалоговом режиме вводить запросы, посылать запросы для выполнения на сервер и получать результаты в предназначенном для этого окне.

Встроенный SQL используется в прикладных программах, позволяя им посылать запросы к серверу и обрабатывать полученные результаты, в том числе, комбинируя ***set***-ориентированный и ***record***-ориентированный подходы.

Реляционные операции и команды манипулирования данными

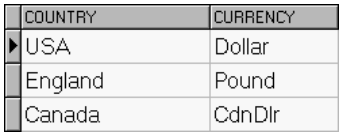
Наиболее важной командой языка манипулирования данными является команда ***SELECT***. За кажущейся простотой ее синтаксиса скрывается ее большая мощность.

В качестве информационной основы для примеров будем использовать базу данных "Служащие предприятия" (employee.gdb), входящую в поставку *Delphi* и находящуюся, по умолчанию, в папке \IBLOCAL\EXAMPLES. В рассматриваемых примерах предполагается, что все команды языка **SQL** вводятся интерактивным способом.

Базовыми операциями реляционных баз данных являются: ***Restriction*** - выборка; ***Projection*** – проекция; ***Join*** – соединение; ***Union*** - объединение.

Операция выборки позволяет получить все строки (записи) либо часть строк одной таблицы.

Пример 1 демонстрирует, каким образом можно получить все строки таблицы Country.

SELECT * FROM country	

COUNTRY	CURRENCY
USA	Dollar
England	Pound
Canada	CdnDlr

Пример 2 демонстрирует, каким образом можно получить подмножество строк таблицы Country, удовлетворяющее условию Currency = “Dollar”.

SELECT * FROM country WHERE currency = “Dollar”	

COUNTRY	CURRENCY
USA	Dollar

Операция проекции позволяет выделить подмножество столбцов таблицы.

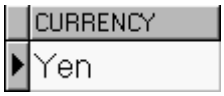
Пример 3 демонстрирует, каким образом можно получить список денежных единиц.

SELECT currency FROM country	

CURRENCY
Dollar
Pound

В практической работе часто требуется получить некое подмножество столбцов и строк таблицы, т.е. выполнить комбинацию **Restriction** и **Projection**. Для этого достаточно перечислить столбцы таблицы и наложить ограничения на строки.

Пример 4 демонстрирует, каким образом определить денежную единицу Японии.

SELECT currency FROM country WHERE country = “Japan”	

CURRENCY
Yen

Пример 5 демонстрирует, каким образом получить фамилии работников, которых зовут “Roger”.

SELECT first_name, last_name FROM employee WHERE first_name = "Roger"	

FIRST_NAME	LAST_NAME
Roger	De Souza
Roger	Reeves

Примеры 1 – 5 иллюстрируют общую форму команды *SELECT* в языке *SQL* для одной таблицы:

- *SELECT* (выбрать) специфицированные поля.
- *FROM* (из) специфицированной таблицы.
- *WHERE* (где) некоторое специфицированное условие является истинным.

Операция соединения позволяет соединять строки нескольких таблиц для образования новых строк данных.

Пример 6 демонстрирует, каким образом можно получить список руководителей проектов.

```
SELECT first_name, last_name,
proj_name
FROM employee, project
WHERE emp_no = team_leader
```

FIRST_NAME	LAST_NAME	PROJ_NAME
Ashok	Ramanathan	Video Database
Pete	Fisher	DigiPizza

Операция объединения позволяет объединять результаты отдельных запросов по нескольким таблицам в единую результирующую таблицу. Таким образом, предложение *UNION* объединяет вывод двух или более *SQL*-запросов в единый набор строк и столбцов.

Пример 7 демонстрирует, каким образом можно получить список работников и заказчиков, проживающих во Франции.

```
SELECT first_name, last_name, job_country
FROM employee
WHERE job_country = "France"
UNION
SELECT contact_first, contact_last, country
FROM customer
WHERE country = "France"
```

FIRST_NAME	LAST_NAME	JOB_COUNTRY
Jacques	Glon	France
Michelle	Roche	France

Рассмотрим общую форму команды SELECT, учитывающую возможность соединения нескольких таблиц и объединения результатов.

SELECT [DISTINCT] список_выбираемых_элементов (полей)

FROM список_таблиц (или представлений)

[WHERE предикат]

[GROUP BY поле (или поля) [HAVING предикат]]

[UNION другое_выражение_Select]

[ORDER BY поле (или поля) или номер (номера)].

Отметим, что под предикатом понимается некоторое специфицированное условие отбора, значение которого имеет булевский тип. Квадратные скобки означают необязательность использования дополнительных конструкций команды. Точка с запятой является стандартным терминатором команды. Отметим, что в *WISQL* и в компоненте *Query* ставить конечный терминатор не обязательно. При этом там, где допустим один пробел между элементами, разрешено ставить любое

количество пробелов и пустых строк - выполняя желаемое форматирование для большей наглядности.

Основные конструкции команды SELECT

Рассмотрим простейшие конструкции языка **SQL**, которые позволяют:

- определять поля, которые должны быть выбраны;
- назначать к выборке все поля;
- управлять вертикальным и горизонтальным порядком выбираемых полей;
- подставлять собственные заголовки полей в результирующей таблице;
- производить вычисления в списке выбираемых элементов;
- использовать литералы в списке выбираемых элементов;
- ограничивать число возвращаемых строк;
- формировать сложные условия поиска, используя, реляционные и логические операторы;
- устранять одинаковые строки из результата.

Список выбираемых элементов может содержать: имена полей, вычисления, литералы, функции и агрегирующие конструкции.

Содержимое списков полей.

Пример 8 демонстрирует, каким образом можно получить список имен, фамилий и служебных телефонов всех работников предприятия.

```
SELECT first_name, last_name,
phone_no
FROM phone_list
```

FIRST_NAME	LAST_NAME	PHONE_NO
Terri	Lee	(408) 555-1234
Oliver H.	Bender	(408) 555-1234

Отметим, что *PHONE_LIST* - это виртуальная таблица (представление), созданная в ***InterBase*** и основанная на информации из двух таблиц - *EMPLOYEE* и *DEPARTMENT*. Однако, как уже упоминалось в общей структуре команды *SELECT*, к ней можно обращаться так же, как и к действительной таблице.

Содержимое всех полей.

Пример 9 демонстрирует, каким образом можно получить список служебных телефонов всех работников предприятия, включая дополнительную информацию.

```
SELECT * FROM
phone_list
```

EMP_NO	FIRST_NAME	LAST_NAME	PHONE_EXT	LOCATION	PHONE_NO
12	Terri	Lee	256	Monterey	(408) 555-1234
105	Oliver H.	Bender	255	Monterey	(408) 555-1234

Содержимое всех полей в произвольном порядке.

Пример 10 демонстрирует, каким образом можно получить список служебных телефонов всех работников предприятия, включая дополнительную информацию, расположив их в требуемом порядке.

```
SELECT first_name, last_name,
phone_no, location, phone_ext,
emp_no FROM phone_list
```

FIRST_NAME	LAST_NAME	PHONE_NO	LOCATION	PHONE_EXT	EMP_NO
Terri	Lee	(408) 555-1234	Monterey	256	12
Oliver H.	Bender	(408) 555-1234	Monterey	255	105

Вычисления.

Пример 11 демонстрирует, каким образом можно получить список номеров служащих и их зарплату, в том числе увеличенную на 15%.

```
SELECT emp_no, salary, salary * 1.15 FROM
employee
```

EMP_NO	SALARY	COLUMN3
2	105900	121785
4	97500	112125

Порядок вычисления выражений подчиняется общепринятым правилам: сначала выполняется умножение и деление, а затем - сложение и вычитание. Операции одного уровня выполняются слева направо. Разрешено применять скобки для изменения порядка вычислений.

Литералы.

Для придания большей наглядности получаемому результату можно использовать литералы. Литералы - это строковые константы, которые применяются наряду с наименованиями столбцов и, таким образом, выступают в роли псевдостолбцов. Строка символов, представляющая собой литерал, должна быть заключена в одинарные или двойные скобки.

Пример 12 демонстрирует, каким образом можно получить список сотрудников и их зарплату.

```
SELECT first_name, "получает",
salary, "долларов в год" FROM
employee
```

FIRST_NAME	COLUMN2	SALARY	COLUMN4
Robert	получает	105900	долларов в год
Bruce	получает	97500	долларов в год

Конкатенация.

Команда **Select** обладает возможностью соединять два или более столбца, имеющие строковый тип, друг с другом, а также соединять их с литералами. Для этого используется операция конкатенации (||).

Пример 13 демонстрирует, каким образом можно получить список всех сотрудников предприятия.

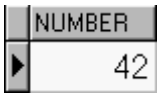
```
SELECT "сотрудник " || first_name || " " ||
last_name FROM employee
```

COLUMN1
сотрудник Robert Nelson
сотрудник Bruce Young

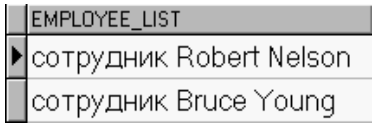
Использование квалификатора AS

Для придания наглядности выводимым результатам наряду с литералами в списке выбираемых элементов можно использовать квалификатор ***AS***. Данный квалификатор заменяет в результирующей таблице существующее название столбца на заданное название. Это наиболее эффективный и простой способ создания заголовков.

Пример 14 демонстрирует, каким образом можно подсчитать количество служащих.

SELECT count(*) AS number FROM employee	

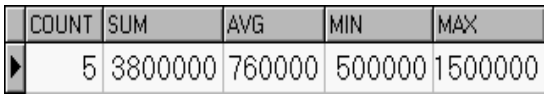
Пример 15 демонстрирует, каким образом можно получить список всех сотрудников.

SELECT "сотрудник " || first_name || " " || last_name AS employee_list FROM employee	

Агрегатные функции.

К агрегирующим функциям относятся функции вычисления суммы (SUM), максимального (MAX) и минимального (MIN) значений столбцов, арифметического среднего (AVG), а также количества строк, удовлетворяющих заданному условию (COUNT).

Пример 16 демонстрирует, каким образом можно вычислить: количество секторов, являющихся подразделениями отдела 100 (маркетинга и продажи), их суммарный, средний, минимальный и максимальный бюджеты.

SELECT count(*), sum (budget), avg (budget), min (budget), max (budget) FROM department WHERE head_dept = 100	

Предложение FROM команды SELECT

В предложении **FROM** перечисляются все объекты, из которых производится выборка данных. Каждая таблица или представление, которые описываются в запросе, должны быть перечислены в предложении **FROM**.

Ограничения на число выводимых строк

Число возвращаемых в результате запроса строк может быть ограничено путем использования предложения **WHERE**, содержащего условия отбора. Условие отбора для отдельных строк может принимать значения **true**, **false** или **unnown**. При этом запрос возвращает в качестве

результата только те строки (записи), для которых предикат имеет значение **true**.

Типы предикатов, используемых в предложении **WHERE**: *сравнение с использованием реляционных операторов*: (= равно); (<> не равно); (!= не равно); (> больше); (< меньше); (>= больше или равно); (<= меньше или равно), а также: BETWEEN, IN, LIKE, CONTAINING, IS NULL, EXIST, ANY, ALL.

Операции сравнения

Реляционные операторы могут использоваться с различными элементами. При этом важно соблюдать следующее правило: *элементы должны иметь сравнимые типы*. Если в базе данных определены домены, то сравниваемые элементы должны относиться к одному домену. Элементом сравнения могут выступать: (Значение поля); (Литерал); (Арифметическое выражение); (Агрегирующая функция); (Встроенная функция); (Значение, возвращаемые подзапросом).

При сравнении литералов конечные пробелы игнорируются.

Пример 17 демонстрирует, каким образом можно получить список сотрудников занимающих должность администраторов и номера отделов, в которых они работают.

```
SELECT first_name, last_name, dept_no
FROM employee
WHERE job_code = 'Admin'
```

FIRST_NAME	LAST_NAME	DEPT_NO
Terri	Lee	000
Ann	Bennet	120

Пример 18 демонстрирует, каким образом можно получить список сотрудников работающих вне США, а также номера их отделов и страну.

```
SELECT first_name, last_name, dept_no,
job_country
FROM employee
WHERE job_country <> «USA»
```

FIRST_NAME	LAST_NAME	DEPT_NO	JOB_COUNTRY
Ann	Bennet	120	England
Roger	Reeves	120	England

Предикат BETWEEN

Предикат BETWEEN задает диапазон значений, для которого выражение принимает значение **true**. Разрешено также использовать конструкцию NOT BETWEEN.

Пример 19 демонстрирует, каким образом можно получить список сотрудников, годовая зарплата которых больше 20000 и меньше 30000.

```
SELECT first_name, last_name, salary FROM
employee
WHERE salary BETWEEN 20000 AND 30000
```

FIRST_NAME	LAST_NAME	SALARY
Ann	Bennet	22935
Kelly	Brown	27000

Предикат BETWEEN с отрицанием NOT (NOT BETWEEN) позволяет получить выборку записей, указанные поля которых имеют значения меньше нижней границы и больше верхней границы.

Пример 20 демонстрирует, каким образом можно получить список самых “старых” и самых “молодых” (по времени поступления на работу) сотрудников.

```
SELECT first_name, last_name,
hire_date FROM employee
WHERE hire_date NOT BETWEEN
"1-JAN-1989" AND
"31-DEC-2004"
```

FIRST_NAME	LAST_NAME	SALARY
Ann	Bennet	22935
Kelly	Brown	27000

Предикат IN

Предикат IN проверяет, входит ли заданное значение, предшествующее ключевому слову “IN” (например, значение столбца или функци). Если заданное проверяемое значение равно какому-либо элементу в списке, то предикат принимает значение **true**. Разрешено также использовать конструкцию NOT IN.

Пример 21 демонстрирует, каким образом можно получить список сотрудников, занимающих должности вице-президент, администратор и финансовый директор.

```
SELECT first_name, last_name, job_code FROM
employee
WHERE job_code IN ("VP", "Admin", "Finan")
```

FIRST_NAME	LAST_NAME	JOB_CODE
Robert	Nelson	VP
Terri	Lee	Admin

Пример 22 демонстрирует, каким образом можно получить список сотрудников, работающих не в США, не в Японии и не в Великобритании.

```
SELECT first_name, last_name, job_country
FROM employee
WHERE job_country NOT IN ("USA",
"Japan", "England")
```

FIRST_NAME	LAST_NAME	JOB_COUNTRY
Claudia	Sutherland	Canada
Roberto	Ferrari	Italy

Предикат LIKE.

Предикат LIKE используется только с символьными данными. Он проверяет, соответствует ли данное символьное значение строке с указанной маской. В качестве маски используются все разрешенные символы с учетом верхнего и нижнего регистров, а также специальные символы:

% - замещает любое количество символов (в том числе и 0);

(_) - замещает только один символ. (*Разрешено также использовать конструкцию NOT LIKE*).

Пример 23 демонстрирует, каким образом можно получить список сотрудников, фамилии которых начинаются с буквы “F”.

```
SELECT first_name, last_name FROM
employee
WHERE last_name LIKE "F%"
```

FIRST_NAME	LAST_NAME
Phil	Forest
Pete	Fisher

Пример 24 демонстрирует, каким образом можно получить список сотрудников, имена которых заканчиваются буквами “er”.

```
SELECT first_name, last_name FROM
employee
WHERE first_name LIKE "%er"
```

FIRST_NAME	LAST_NAME
Roger	De Souza
Roger	Reeves

Предикат CONTAINING.

Предикат CONTAINING аналогичен предикату LIKE, за исключением того, что он не чувствителен к регистру букв. Разрешено также использовать конструкцию NOT CONTAINING.

Пример 25 демонстрирует, каким образом можно получить список сотрудников, фамилии которых содержат буквы “ne”, “Ne”, “NE”, “nE”.

```
SELECT first_name, last_name  FROM
employee
WHERE last_name CONTAINING "ne"
```

FIRST_NAME	LAST_NAME
Robert	Nelson
Ann	Bennet

Предикат IS NULL.

В SQL-запросах **NULL** означает, что значение столбца неизвестно. Поисковые условия, в которых значение столбца сравнивается с NULL, всегда принимают значение **unknown** и, соответственно, приводят к ошибке, в противоположность true или false, т.е.

WHERE dept_no = NULL или даже WHERE NULL = NULL.

Предикат **IS NULL** принимает значение true только тогда, когда выражение слева от ключевых слов “IS NULL” имеет значение null (пусто, не определено). Разрешено также использовать конструкцию IS NOT NULL, которая означает “не пусто”, “имеет какое-либо значение”.

Пример 27 демонстрирует, каким образом можно получить список отделов, в которых еще не назначены руководители.

```
SELECT department, mngr_no FROM
department
WHERE mngr_no IS NULL
```

DEPARTMENT	MNGR_NO
Marketing	
Software Products Div.	

Логические операторы

К логическим операторам относятся операторы **AND**, **OR**, **NOT**, позволяющие выполнять различные логические действия: логическое умножение AND (пересечение условий), логическое сложение **OR** (объединение условий), логическое отрицание **NOT** (отрицание условий).

Оператор **AND** означает, что общий предикат будет истинным только тогда, когда условия, связанные по **AND**, будут истинны.

Оператор **OR** означает, что общий предикат будет истинным, когда хотя бы одно из условий, связанных по **OR**, будет истинным.

Оператор **NOT** означает, что общий предикат будет истинным, когда условие, перед которым стоит этот оператор, будет ложным.

В одном предикате логические операторы выполняются в следующем порядке: сначала выполняется оператор **NOT**, затем - **AND** и только после этого выполняется оператор **OR**. Для изменения порядка выполнения операторов разрешается использовать скобки.

Пример 27 демонстрирует, каким образом можно получить список служащих, работающих в отделе 622 или в должности инженер с зарплатой не выше 40000$ в год.

```
SELECT first_name, last_name,
dept_no, job_code,
salary FROM employee WHERE
dept_no = 622 OR
 job_code = "Eng" AND salary <=
40000
ORDER BY last_name
```

FIRST_NAME	LAST_NAME	DEPT_NO	JOB_CODE	SALARY
Jennifer M.	Burbank	622	Eng	53167,5
Phil	Forest	622	Mngr	75060

Преобразование типов CAST

В **SQL** имеется возможность преобразовать значение столбца или функции к другому типу для более гибкого использования операций сравнения. Для этого используется функция **CAST**.

Пример 28 демонстрирует, каким образом можно получить список сотрудников, работающих в отделах, номера которых содержат “00”.

```
SELECT first_name, last_name, dept_no
FROM employee
WHERE CAST(dept_no AS char(20))
CONTAINING "00"
```

FIRST_NAME	LAST_NAME	DEPT_NO
Robert	Nelson	600
Terri	Lee	000

Изменение порядка выводимых строк ORDER BY

Порядок выводимых строк может быть изменен с помощью опционального (дополнительного) предложения **ORDER BY**, расположенного в конце SQL-запроса. Это предложение имеет вид: ORDER BY <порядок строк> [ASC | DESC]. Порядок строк может задаваться одним из двух способов: ***именами*** столбцов и ***номерами*** столбцов.

Способ упорядочивания определяется дополнительными зарезервированными словами **ASC** и **DESC**. Основным способом, по умолчанию, является упорядочивание по возрастанию **ASC**. Если же указано слово **DESC**, то упорядочивание будет производиться по убыванию.

Обратите внимание на тот факт, что предложение **ORDER BY** должно указываться в самом конце запроса.

Упорядочивание с использованием имен столбцов.

Пример 29 демонстрирует, каким образом можно получить список сотрудников, упорядоченный по фамилиям в алфавитном порядке.

```
SELECT first_name, last_name,
dept_no, job_code,
salary FROM employee ORDER BY
last_name
```

FIRST_NAME	LAST_NAME	DEPT_NO	JOB_CODE	SALARY
Janet	Baldwin	110	Sales	61637,8125
Oliver H.	Bender	000	CEO	212850

Пример 30 демонстрирует, каким образом можно получить список сотрудников, упорядоченный по фамилиям в порядке, обратном алфавитному порядку.

```
SELECT first_name, last_name, dept_no,
job_code, salary FROM employee ORDER
BY last_name DESC
```

FIRST_NAME	LAST_NAME	DEPT_NO	JOB_CODE	SALARY
Katherine	Young	623	Mngr	67241,25
Bruce	Young	621	Eng	97500

Упорядочивание с использованием номеров столбцов

Пример 31 демонстрирует, каким образом можно получить список сотрудников, упорядоченный по их зарплате с 10% надбавкой.

```
SELECT first_name, last_name,
dept_no, job_code, salary * 1.1 FROM
employee ORDER BY 5
```

FIRST_NAME	LAST_NAME	DEPT_NO	JOB_CODE	COLUMN5
Ann	Bennet	120	Admin	25228,5
Kelly	Brown	600	Admin	29700

Устранение дублирования (модификатор DISTINCT)

Дублированными являются такие строки в результирующей таблице, в которых идентичен каждый столбец. Иногда, в зависимости от поставленной задачи, бывает необходимо устранить все повторы строк из результирующего набора. Для этой цели служит модификатор **DISTINCT**.

Данный модификатор может быть указан только один раз в списке выбираемых элементов и действует на весь список.

Операции сравнения

Пример 32 демонстрирует, каким образом можно получить список служащих, имена которых – Roger.

```
SELECT first_name, last_name FROM
employee
WHERE first_name = "Roger"
```

FIRST_NAME	LAST_NAME
Roger	De Souza
Roger	Reeves

Соединение JOIN

Операция соединения используется в языке **SQL** для вывода *связанной* информации, хранящейся в нескольких таблицах, в одном запросе. В этом проявляется одна из наиболее важных особенностей запросов **SQL** - способность определять связи между многочисленными таблицами и выводить информацию из них в рамках этих связей. Именно эта операция придает гибкость и легкость языку **SQL**. Применяя эту операцию, появится возможность:

- Соединять данные из нескольких таблиц в единую результирующую таблицу.
- Задавать имена столбцов двумя способами.
- Записывать внешние соединения.
- Создавать внутренние соединения в таблице.

Операции соединения подразделяются на два вида - *внутренние* и *внешние*. Оба вида соединений задаются в предложении **WHERE** запроса **SELECT** с помощью специального *условия соединения*. Внешние соединения поддерживаются стандартом **ANSI-92** и содержат зарезервированное слово **JOIN**, в то время как внутренние соединения могут задаваться как с использованием слова **JOIN**, так и без использования слова **JOIN**.

Связывание производится, как правило, по первичному ключу одной таблицы и внешнему ключу другой таблицы (для каждой пары таблиц). При этом важно учитывать все поля внешнего ключа, иначе результат будет искажен. Соединяемые поля могут присутствовать в списке выбираемых элементов. Предложение **WHERE** может содержать множественные условия соединений. Условие соединения может также комбинироваться с другими предикатами в предложении **WHERE**.

Внутренние соединения.

Внутреннее соединение возвращает только те строки, для которых условие соединения принимает значение **true**.

Пример 33 демонстрирует, каким образом можно получить список сотрудников, состоящих в должности “вице-президент”, а также названия их отделов.

```
SELECT first_name, last_name,
department
FROM employee, department WHERE
job_code = "VP"
```

FIRST_NAME	LAST_NAME	DEPARTMENT
Robert	Nelson	Corporate Headquarters
Mary S.	MacDonald	Corporate Headquarters

В примере 33 запрос (без соединения) возвращает неверный результат, так как имеющиеся между таблицами связи не задействованы. Исходя из этого, появляется дублирование информации в результирующей таблице. Правильный результат дает запрос с использованием операции соединения.

Пример 34 демонстрирует, каким образом можно получить аналогичную информацию без дублирования.

```
SELECT first_name, last_name,
department
FROM employee, department WHERE
job_code = "VP"
AND employee.dept_no =
department.dept_no
```

FIRST_NAME	LAST_NAME	DEPARTMENT
Robert	Nelson	Engineering
Mary S.	MacDonald	Sales and Marketing

В приведенном выше примере 34 запросе использовался способ непосредственного указания таблиц с помощью их имен. Возможен, а иногда и просто необходим, также способ указания таблиц с помощью алиасов (псевдонимов). При этом алиасы определяются в предложении FROM запроса SELECT и представляют собой любой допустимый идентификатор, написание которого подчиняется таким же правилам, что и написание имен таблиц. Потребность в алиасах таблиц возникает тогда, когда названия столбцов, используемых в условиях соединения двух или более таблиц, совпадают, а названия таблиц имеют длинные названия.

Внутренние соединения. В некоторых задачах необходимо получить информацию, выбранную особым образом только из одной таблицы. Для этого используются так называемые *самосоединения*, или рефлексивные соединения. Это не отдельный вид соединения, а просто соединение таблицы с собой с помощью алиасов. Самосоединения полезны в случаях, когда нужно получить пары аналогичных элементов из одной и той же таблицы.

Пример 35 демонстрирует, каким образом можно получить список пар отделов с одинаковыми годовыми бюджетами.

```
SELECT d1.department, d2.department, d1.budget
FROM department d1, department d2
WHERE d1.budget = d2.budget AND d1.dept_no
< d2.dept_no
```

DEPARTMENT	DEPARTMENT_1	BUDGET
Software Development	Finance	400000
Field Office: East Coast	Field Office: Canada	500000

Внешние соединения

Внешнее соединение возвращает все строки из одной таблицы, и только те строки из другой таблицы, для которых условие соединения принимает значение **true**. Строки второй таблицы, не удовлетворяющие условию соединения (т.е. имеющие значение **false**), получают значение **null** в результирующем наборе. Существует два вида внешнего соединения: **LEFT JOIN** и **RIGHT JOIN**. В левом соединении (**LEFT JOIN**) запрос возвращает *все строки из левой таблицы* (т.е. таблицы, стоящей *слева* от зарезервированного словосочетания **LEFT JOIN**) и только те из правой таблицы, которые удовлетворяют условию соединения. Если же в правой таблице не найдется строк, удовлетворяющих заданному условию, то в результате они замещаются значениями **null**. Для правого соединения результат будет обратным.

Пример 36 демонстрирует, каким образом можно получить список сотрудников и название их отделов, включая сотрудников, еще не назначенных ни в какой отдел.

```
SELECT first_name, last_name,
department
FROM employee e LEFT JOIN
department d
ON e.dept_no = d.dept_no
```

FIRST_NAME	LAST_NAME	DEPARTMENT
Robert	Nelson	Engineering
Bruce	Young	Software Development

Пример 38 демонстрирует, каким образом можно получить аналогичную информацию путем правого соединения.

```
SELECT first_name, last_name,
department
FROM employee e RIGHT JOIN
department d
ON e.dept_no = d.dept_no
```

FIRST_NAME	LAST_NAME	DEPARTMENT
Terri	Lee	Corporate Headquarters
Oliver H.	Bender	Corporate Headquarters

Гибкость и мощь языка SQL состоит в том, что он позволяет объединить все операции реляционной алгебры в одной конструкции, извлекая, таким образом, любую требуемую информацию.

ЛИТЕРАТУРА

1. Олифер В.Г. Олифер Н. А. Компьютерные сети. Принципы, технологии протоколы. Учебник. Санкт-Петербург. 2000. – 672с.
2. Кулаков Ю. А. Луцкий Г. М. Компьютерные сети. Кулаков Ю. А. Луцкий Г. М. – 384с.
3. Кулаков Ю. А. Луцкий Г. М. Локальные сети. Кулаков Ю. А. Луцкий Г. М. – 336с.
4. Р. Боас, М. Фервай, Х. Гюнтер, Delphi 4 Повне керівництво, Київ, видавництво BHV, 1998р.