Министерство образования Российской Федерации
Владимирский государственный университет
Муромский институт (филиал)

# ИНФОРМАЦИОННЫЕ СЕТИ

Учебное пособие
Составители: С.В. Кошелев, А.В. Яковлев

Муром 2004

УДК 681.3.06
К

Рецензенты:

Доктор технических наук, профессор Владимирского государственного педагогического университета
Н.Г. Наянзин

Кафедра
Информационных систем и информационного менеджмента
Владимирского государственного университета

Кошелев С.В., Яковлев А.В.
Информационные сети: Учеб. пособие. – Муром: Изд-полиграфический центр МИ ВлГУ, 2004. - с., ил., табл., библиогр. 41 назв.

ISBN

В учебном пособии изложены основные принципы построения и функционирования информационных сетей. Рассмотрены семиуровневая базовая модель связи открытых систем OSI, стандарты для физических компонентов сети спецификации IEEE802, стеки протоколов OSI и TCP/IP. Освящены вопросы выбора топологии сети, сетевой операционной системы и сетевого оборудования. Уделено внимание вопросам IP-маршрутизации в рамках ЛВС и глобальной вычислительной сети Internet.

Учебное пособие включает курс из пяти лабораторных работ и методические указания на курсовое проектирование по дисциплине «Информационные сети».

Настоящее учебное пособие предназначено для студентов высших учебных заведений специальности 071900 «Информационные системы и технологии» и студентов смежных специальностей. Оно также будет полезно всем, кто желает самостоятельно ознакомиться с основами информационных сетей.

УДК 681.3.06

ISBN

## Оглавление

**Введение ..... 7**
**1 Обзор и архитектура вычислительных сетей ..... 7**
1.1 Основные определения и термины ..... 7
1.2 Преимущества использования сетей ..... 10
1.3 Архитектура сетей ..... 11
Архитектура терминал – главный компьютер ..... 11
Одноранговая архитектура ..... 12
Архитектура клиент – сервер ..... 13
Выбор архитектуры сети ..... 16
Вопросы по теме ..... 16
**2 Семиуровневая модель OSI ..... 17**
2.1 Взаимодействие уровней модели OSI ..... 18
2.2 Прикладной уровень (Application layer) ..... 21
2.3 Уровень представления данных (Presentation layer) ..... 22
2.4 Сеансовый уровень (Session layer) ..... 23
2.5 Транспортный уровень (Transport Layer) ..... 24
2.6 Сетевой уровень (Network Layer) ..... 25
2.7 Канальный уровень (Data Link) ..... 27
2.8 Физический уровень (Physical Layer) ..... 29
2.9 Зависящие и независящие от технической реализации сети уровни ..... 31
2.10 Стеки коммуникационных протоколов ..... 32
Вопросы по теме ..... 32
**3 Стандарты и стеки протоколов ..... 33**
3.1 Спецификации стандартов ..... 33
Стандарт 802.1 ..... 34
Стандарт 802.2 ..... 34
Стандарт 802.3 ..... 34
Стандарт 802.4 ..... 35
Стандарт 802.5 ..... 35
Стандарт 802.6 ..... 36
Стандарт 802.7 ..... 36
Стандарт 802.8 ..... 36
Стандарт 802.9 ..... 36
Стандарт 802.10 ..... 36
Стандарт 802.11 ..... 36
Стандарт 802.12 ..... 37
3.2 Протоколы и стеки протоколов ..... 37
Сетевые протоколы ..... 37
Транспортные протоколы ..... 38
Прикладные протоколы ..... 38
3.3 Стек OSI ..... 39

3.4 Архитектура стека протоколов Microsoft TCP/IP ..... 40
Уровень приложения ..... 40
Уровень транспорта ..... 41
Межсетевой уровень ..... 42
Уровень сетевого интерфейса ..... 45
Вопросы по теме ..... 45
**4 Топология вычислительной сети и методы доступа ..... 46**
4.1 Топология вычислительной сети ..... 46
4.2 Виды топологий ..... 46
Топология «Общая шина» ..... 47
Топология «Кольцо» ..... 47
Топология «Звезда» ..... 48
Древовидная топология ..... 49
Ячеистая топология ..... 50
4.3 Методы доступа ..... 50
CSMA/CD ..... 51
TPMA ..... 52
TDMA ..... 53
FDMA ..... 54
Вопросы по теме ..... 55
**5 ЛВС и компоненты ЛВС ..... 56**
5.1 Локальная вычислительная сеть ..... 56
5.2 Основные компоненты вычислительной сети ..... 56
Рабочая станция ..... 57
Сервер ..... 57
Сетевое оборудование ..... 58
Сетевая операционная система ..... 58
Сетевое программное обеспечение ..... 59
Вопросы по теме ..... 59
**6 Физическая среда передачи данных ..... 60**
6.1 Кабели связи, линии связи, каналы связи ..... 61
6.2 Типы кабелей и структурированные кабельные системы ..... 61
6.3 Кабельные системы ..... 62
6.4 Типы кабелей ..... 62
Кабель типа «витая пара» (twisted pair) ..... 63
Коаксиальные кабели ..... 63
Оптоволоконный кабель ..... 64
6.5 Кабельные системы Ethernet ..... 65
10Base-T, 100Base-TX ..... 65
10Base2 ..... 66
10Base5 ..... 66
6.6 Беспроводные технологии ..... 66
Радиосвязь ..... 66

Связь в микроволновом диапазоне ..... 67
Инфракрасная связь ..... 67
Вопросы по теме ..... 67
**7 Сетевые операционные системы ..... 68**
7.1 Структура сетевой операционной системы ..... 68
7.2 Одноранговые сетевые ОС и ОС с выделенными серверами ..... 71
Вопросы по теме ..... 72
**8 IP-адресация и маршрутизация. DNS ..... 73**
8.1 Адреса IP ..... 73
8.2 Сопоставление сетевых адресов ..... 74
8.3 Маршрутизация в сетях IP ..... 74
Сети IP ..... 75
Подсети IP ..... 75
Шлюзы ..... 76
Таблица маршрутов ..... 77
8.4 Назначение IP-адресов узлам ..... 78
Динамическое назначение адресов при помощи DHCP ..... 79
Ограничения DHCP ..... 79
Использование WINS ..... 79
8.5 Служба разрешения доменных имен (DNS) ..... 80
Разрешение имен сетевых узлов ..... 80
Введение в DNS ..... 81
Поиск имени в системе DNS ..... 82
Вопросы по теме ..... 83
**9 Сетевое оборудование ..... 84**
9.1 Сетевые адаптеры ..... 84
Назначение ..... 84
Настройка сетевого адаптера и трансивера ..... 85
Функции сетевых адаптеров ..... 85
Типы сетевых адаптеров ..... 86
9.2 Повторители и концентраторы ..... 87
9.3 Мосты и коммутаторы ..... 89
Мосты ..... 89
Коммутаторы ..... 90
9.4 Маршрутизаторы ..... 91
9.5 Шлюзы ..... 92
Вопросы по теме ..... 93
**10 Требования, предъявляемые к сетям ..... 94**
10.1 Производительность ..... 94
10.2 Надежность и безопасность ..... 94
10.3 Прозрачность ..... 95
10.4 Поддержка разных видов трафика ..... 96
10.5 Управляемость ..... 97

Управление эффективностью ..... 97
Управление конфигурацией ..... 98
Управление учетом использования ресурсов ..... 98
Управление неисправностями ..... 98
Управление защитой данных ..... 98
10.6 Совместимость ..... 99
Вопросы по теме ..... 100
**Лабораторные работы по курсу «Информационные сети» ..... 100**
Лабораторная работа №1 «Знакомство с сетевым протоколом FTP» ..... 101
Цель работы ..... 101
Теоретические сведения ..... 101
Замечания по выполнению лабораторной работы ..... 101
Задание на лабораторную работу ..... 102
Лабораторная работа №2 «Сетевое программирование на базе Sockets» ..... 103
Цель работы ..... 103
Теоретические сведения ..... 103
Замечания по выполнению лабораторной работы ..... 109
Задание на лабораторную работу ..... 109
Лабораторная работа №3 «Знакомство с библиотекой DirectPlay» ..... 109
Цель работы ..... 109
Теоретические сведения ..... 109
Задание на лабораторную работу ..... 140
Лабораторная работа №4 «Программирование сетевых соединений, предоставляемых DirectPlay» ..... 140
Цель работы ..... 140
Теоретические сведения ..... 140
Задание на лабораторную работу ..... 140
Лабораторная работа №5 «Программирование сетевых соединений, предоставляемых DirectPlay, с использованием множества пользовательских сообщений» ..... 141
Цель работы ..... 141
Теоретические сведения ..... 141
Задание на лабораторную работу ..... 143
**Курсовое проектирование по дисциплине «Информационные сети» ... 144**
Введение ..... 144
Основные цели и задачи оптимизации локальных сетей ..... 144
Критерии эффективности работы сети ..... 147
Расчет показателей эффективности вычислительной системы ..... 156
Расчет работоспособности сети ..... 162
Тематика курсовых проектов ..... 164
Содержание курсового проекта ..... 164
Подготовка доклада и защита проекта ..... 165
**Литература ..... 166**

# Введение

Курс представляет собой введение в сетевую тематику и дает базовые знания по организации и функционированию сетей. В теоретической части даны общие понятия информационных сетей, их структуры, сетевых компонентов и т.д. Здесь приведены виды топологии, используемые для физического соединения компьютеров в сети, методы доступа к каналу связи, физические среды передачи данных. Передача данных в сети рассматривается на базе эталонной базовой модели, разработанной Международной организацией по стандартам взаимодействия открытых сетей. Описываются правила и процедуры передачи данных между информационными системами. Приводятся типы сетевого оборудования, их назначение и принципы работы. Описывается сетевое программное обеспечение, используемое для организации сетей и основные принципы построения сетевых операционных систем. Рассматриваются принципы межсетевого взаимодействия. Приводятся основные понятия из области сетевой безопасности.

Практическая часть курса содержит методические указания к пяти лабораторным работам, выполнение которых позволит студентам не только освоить базовые навыки работы в информационный сети, но и научиться разрабатывать сетевые приложения на базе двух современных технологий: виртуальных соединителей *Sockets* и интерфейса сетевых соединений *DirectPlay*, входящего в состав библиотеки *DirectX*.

Практическая часть курса также включает методические указания к курсовому проектированию по дисциплине «Информационные сети», в которых уделяется особое внимание вопросу проектирования информационно-вычислительной сети, а также расчетам ее работоспособности и эффективности функционирования.

Учебное пособие предназначено для студентов и преподавателей высших учебных заведений специальности 071900 «Информационные системы и технологии», также студентов смежных специальностей..

# 1 Обзор и архитектура вычислительных сетей

## 1.1 Основные определения и термины

*Сеть* – это совокупность объектов, образуемых устройствами передачи и обработки данных. Международная организация по стандартизации определила вычислительную сеть *как последовательную бит-ориентированную передачу информации между связанными друг с другом независимыми устройствами.*

Сети обычно находится в частном ведении пользователя и занимают некоторую территорию и по территориальному признаку разделяются на:

– локальные вычислительные сети (ЛВС) или Local Area Network (LAN), расположенные в одном или нескольких близко расположенных здани-

ях. ЛВС обычно размещаются в рамках какой-либо организации (корпорации, учреждения), поэтому их называют корпоративными;

– распределенные компьютерные сети, глобальные или Wide Area Network (WAN), расположенные в разных зданиях, городах и странах, которые бывают территориальными, смешанными и глобальными. В зависимости от этого глобальные сети бывают четырех основных видов: городские, региональные, национальные и транснациональные. В качестве примеров распределенных сетей очень большого масштаба можно назвать: Internet, EUNET, Relcom, FIDO;

В состав сети в общем случае включается следующие элементы:

– сетевые ЭВМ (оснащенные сетевым адаптером);
– каналы связи (кабельные, спутниковые, телефонные, цифровые, волоконно-оптические, радиоканалы и др.);
– различного рода преобразователи сигналов;
– сетевое оборудование.

Различают два понятия сети: *коммуникационная сеть* и *информационная сеть* (рис. 1.1).

*Коммуникационная сеть* предназначена для передачи данных, также она выполняет задачи, связанные с преобразованием данных. Коммуникационные сети различаются по типу используемых физических средств соединения.

*Информационная сеть* предназначена для хранения информации и состоит из *информационных систем*. На базе коммуникационной сети может быть построена группа информационных сетей:

Под *информационной системой* следует понимать систему, которая является поставщиком или потребителем информации.

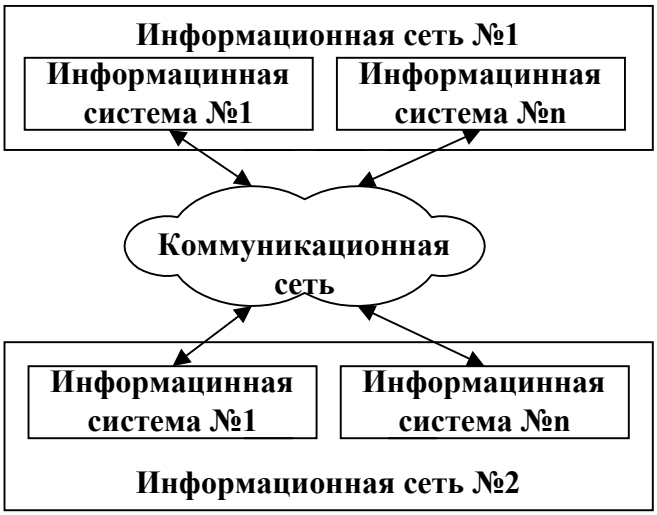


Рис. 1.1. Информационные и коммуникационные сети

Компьютерная сеть состоит из информационных систем и каналов связи.

Под *информационной системой* следует понимать объект, способный осуществлять хранение, обработку или передачу информация. В состав *информационной системы* входят: ЭВМ, программы, пользователи и другие составляющие, предназначенные для процесса обработки и передачи данных. В дальнейшем информационная система, предназначенная для решения задач пользователя, будет называться – *рабочая станция (client)*. Рабочая станция в сети отличается от обычного персонального компьютера (ПК) наличием *сетевой карты* (*сетевого адаптера*), канала для передачи данных и сетевого программного обеспечения.

Под *каналом связи* следует понимать путь или средство, по которому передаются сигналы. Средство передачи сигналов называют *абонентским,* или *физическим, каналом*.

*Каналы связи (data link)* создаются по линиям связи при помощи сетевого оборудования и физических средств связи. Физические средства связи построены на основе витых пар, коаксиальных кабелей, оптических каналов или эфира. Между взаимодействующими информационными системами через физические каналы коммуникационной сети и узлы коммутации устанавливаются *логические каналы.*

*Логический канал* – это путь для передачи данных от одной системы к другой. Логический канал прокладывается по маршруту в одном или нескольких физических каналах. *Логический канал* можно охарактеризовать, как маршрут, проложенный через физические каналы и узлы коммутации.

Информация в сети передается *блоками данных* по процедурам обмена между объектами. Эти процедуры называют *протоколами передачи данных.*

*Протокол* – это совокупность правил, устанавливающих формат и процедуры обмена информацией между двумя или несколькими устройствами.

Загрузка сети характеризуется параметром, называемым *трафиком*. *Трафик (traffic)* – это поток сообщений в сети передачи данных. Под ним понимают количественное измерение в выбранных точках сети числа проходящих *блоков данных* и их длины, выраженное в битах в секунду.

Существенное влияние на характеристику сети оказывает *метод доступа*. *Метод доступа* – это способ определения того, какая из рабочих станций сможет следующей использовать канал связи и как управлять доступом к каналу связи.

В сети все рабочие станции физически соединены между собою каналами связи по определенной структуре, называемой *топологией*. *Топология* – это описание физических соединений в сети, указывающее какие рабочие станции могут связываться между собой. Тип топологии определяет производительность, работоспособность и надежность эксплуатации рабочих станций, а также время обращения к файловому серверу. В зависимости от топологии сети используется тот или иной метод доступа.

Состав основных элементов в сети зависит от ее архитектуры. *Архитектура* – это концепция, определяющая взаимосвязь, структуру и функции взаимодействия рабочих станций в сети. Она предусматривает логическую, функциональную и физическую организацию технических и программных средств сети. Архитектура определяет принципы построения и функционирования аппаратного и программного обеспечения элементов сети.

В основном выделяют три вида архитектур: архитектура *терминал – главный компьютер*, архитектура *клиент – сервер* и *одноранговая* архитектура.

Современные сети можно классифицировать по различным признакам: по удаленности компьютеров, топологии, назначению, перечню предоставляемых услуг, принципам управления (централизованные и децентрализованные), методам коммутации, методам доступа, видам среды передачи, скоростям передачи данных и т. д. Все эти понятия будут рассмотрены более подробно при дальнейшем изучении курса.

## 1.2 Преимущества использования сетей

Компьютерные сети представляют собой вариант сотрудничества людей и компьютеров, обеспечивающего ускорение доставки и обработки информации. Объединять ЭВМ в сети начали более 30 лет назад. Когда возможности компьютеров выросли и ПК стали доступны каждому, развитие сетей значительно ускорилось.

Соединенные в сеть компьютеры обмениваются информацией и совместно используют периферийное оборудование и устройства хранения информации рис. 1.2.

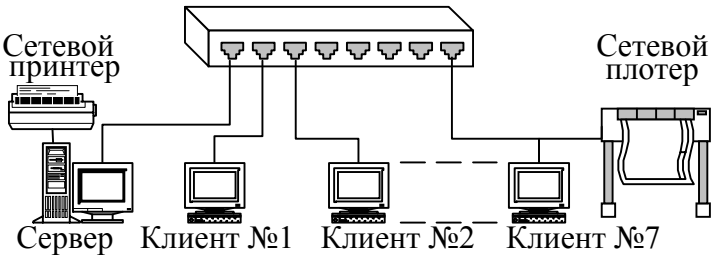


Рис. 1.2. Использование периферийного оборудования

С помощью сетей можно разделять ресурсы и информацию. Ниже перечислены основные задачи, решаемые с помощью сети, и которые трудно решить с помощью отдельного компьютера:

1. компьютерная сеть позволяет совместно использовать периферийные устройства, такие как принтеры, плоттеры, дисковые накопители, приводы CD-ROM, дисководы, стримеры, факс-модемы и т.д.
2. компьютерная сеть позволяет совместно использовать информационные ресурсы такие как каталоги, файлы, прикладные программы, базы данных и т.д.

3. компьютерная сеть позволяет работать с многопользовательскими программами, обеспечивающими одновременный доступ всех пользователей к общим базам данных с блокировкой файлов и записей, обеспечивающей целостность данных. Любые программы, разработанные для стандартных ЛВС, можно использовать в других сетях.

Перечисленные возможности информационной сети обеспечивают существенную экономию средств и времени.

## 1.3 Архитектура сетей

Архитектура сети определяет основные элементы сети, характеризует ее общую логическую организацию, техническое обеспечение, программное обеспечение, описывает методы кодирования. Архитектура также определяет принципы функционирования и интерфейс пользователя.

В данном курсе будет рассмотрено три вида архитектур:

– архитектура терминал – главный компьютер;
– одноранговая архитектура;
– архитектура клиент – сервер.

### Архитектура терминал – главный компьютер

Архитектура терминал – главный компьютер (terminal – host computer architecture) – это концепция информационной сети, в которой вся обработка данных осуществляется одним или группой главных компьютеров.

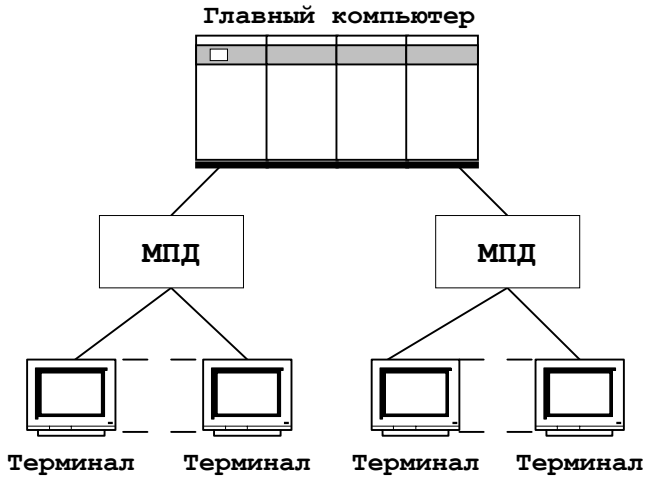


Рис. 1.3. Архитектура терминал – главный компьютер

Рассматриваемая архитектура предполагает два типа оборудования:

– главный компьютер, где осуществляется управление сетью, хранение и обработка данных.
– терминалы, предназначенные для передачи главному компьютеру команд на организацию сеансов и выполнения заданий, ввода данных для выполнения заданий и получения результатов.

*Главный компьютер* через мультиплексоры передачи данных (МПД) взаимодействуют с терминалами, как представлено на рис. 1.3.

Классический пример архитектуры сети с главными компьютерами – системная сетевая архитектура (System Network Architecture – SNA).

Архитектура терминал – главный компьютер применяется, главным образом, тогда, когда терминалы (рабочие станции) обладают очень низкой производительностью, чего в настоящее время практически не встречается.

Существенным недостатком такой архитектуры является наличие узкого места – главного компьютера, который должен обладать очень высокой производительностью для обеспечения эффективного функционирования сети, и высокой надежностью, поскольку выход его строя однозначно приводит к нарушению работоспособности всей информационной сети.

**Одноранговая архитектура**

Одноранговая архитектура (peer-to-peer architecture) – это концепция информационной сети, в которой ее ресурсы рассредоточены по всем системам. Данная архитектура характеризуется тем, что в ней все системы равноправны.

К *одноранговым* сетям относятся малые сети, где любая рабочая станция может выполнять одновременно функции файлового сервера и рабочей станции. В *одноранговых ЛВС* дисковое пространство и файлы на любом компьютере могут быть общими. Чтобы ресурс стал общим, его необходимо отдать в общее пользование, используя службы удаленного доступа сетевых одноранговых операционных систем. В зависимости от того, как будет установлена защита данных, другие пользователи смогут пользоваться файлами сразу же после их создания. *Одноранговые ЛВС* достаточно хороши только для небольших рабочих групп.

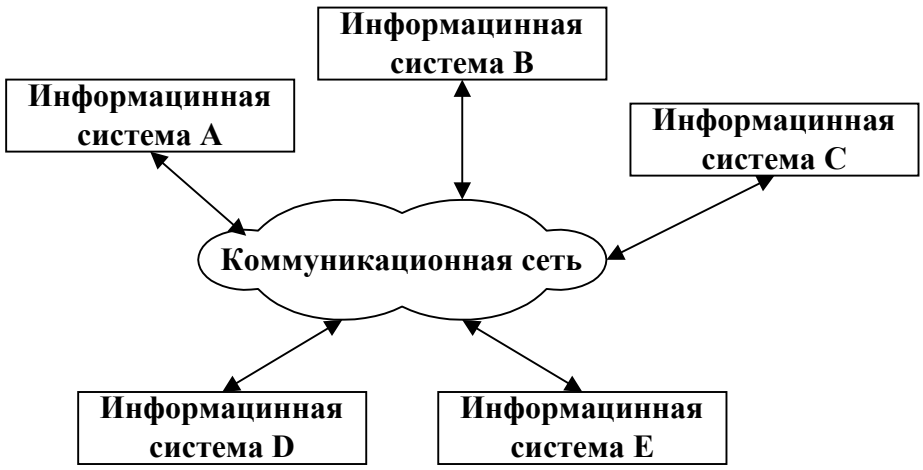


Рис. 1.4. Одноранговая архитектура

*Одноранговые ЛВС* являются наиболее легким и дешевым типом сетей. Создание таких сетей требует наличия на ЭВМ, кроме сетевой карты и сетевого носителя, лишь операционной системы класса *Windows 9x.*

Одноранговые сети имеют следующие преимущества:

- они легки в установке и настройке;
- отдельные ПК не зависят от выделенного сервера;
- пользователи в состоянии контролировать свои ресурсы;
- малая стоимость и легкая эксплуатация;
- минимум оборудования и программного обеспечения;
- нет необходимости в администраторе;
- хорошо подходят для сетей с количеством пользователей, не превышающим десяти.

К недостаткам одноранговой архитектуры можно отнести то, что:

- в случае отключения ПК от сети исчезают виды *сервиса*, которые они предоставляли;
- сетевую безопасность одновременно можно применить только к одному ресурсу, т.е. пользователь должен помнить столько паролей, сколько сетевых ресурсов;
- при получении доступа к разделяемому ресурсу ощущается падение производительности компьютера.

Существенным недостатком одноранговых сетей является отсутствие централизованного администрирования.

Использование одноранговой архитектуры не исключает одновременного применения в той же сети архитектуры «терминал – главный компьютер» или архитектуры «клиент – сервер».

**Архитектура клиент – сервер**


Рис. 1.5. Архитектура клиент – сервер

*Архитектура клиент – сервер* (client-server architecture) – это концепция информационной сети, в которой основная часть ее ресурсов сосредоточена в

серверах, обслуживающих своих клиентов (рис. 1.5). Рассматриваемая архитектура определяет два типа компонентов: *серверы и клиенты*.

*Сервер* - это объект, предоставляющий *сервис* другим объектам сети по их запросам. *Сервис* – это процесс обслуживания клиентов.

Сервер работает по заданиям клиентов и управляет выполнением их заданий. После выполнения каждого задания сервер посылает полученные результаты клиенту, предоставившему это задание.

Сервисная функция в архитектуре клиент – сервер описывается комплексом прикладных программ, в соответствии с которым выполняются разнообразные прикладные процессы.

Процесс, который вызывает сервисную функцию с помощью определенных операций, называется *клиентом*. Им может быть программа или пользователь. На рис. 1.6 приведен перечень сервисов в архитектуре клиент – сервер.

*Клиенты* – это рабочие станции, которые используют ресурсы сервера и предоставляют удобные *интерфейсы пользователя*. *Интерфейсы пользователя* это процедуры взаимодействия пользователя с системой или сетью.

Клиент является инициатором и использует электронную почту или другие сервисы сервера. В этом процессе клиент запрашивает вид обслуживания, устанавливает сеанс, получает нужные ему результаты и сообщает об окончании работы.

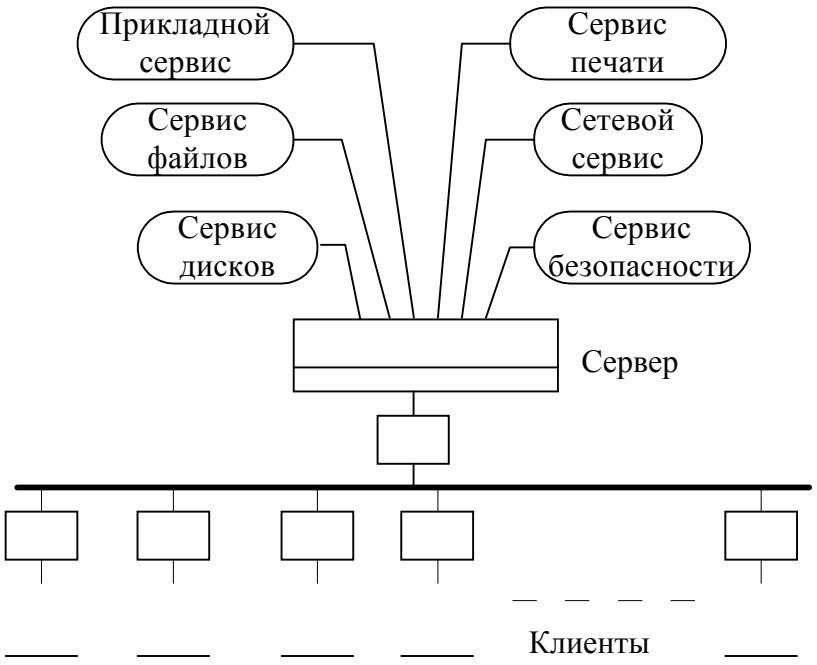


Рис. 1.6. Модель клиент-сервер

Если выполнение каких-либо серверных функций является основным назначением ЭВМ (например, предоставление файлов в общее пользование всем остальным пользователям сети, организация совместного использования факса, или предоставление всем пользователям сети возможности запус-

ка на данном компьютере своих приложений), то такая ЭВМ называется *выделенным сервером*. В зависимости от того, какой ресурс сервера является разделяемым, он называется файл-сервером, факс-сервером, принт-сервером, сервером приложений и т.д. На выделенных серверах желательно устанавливать сетевую операционную систему (ОС), специально оптимизированные для выполнения тех или иных серверных функций.

Рабочая станция чаще всего оснащается сетевой ОС общего назначения.

К наиболее распространенным сетевым ОС можно отнести:

– NetWare фирмы Novel;
– Windows NT фирмы Microsoft;
– UNIX;
– Linux.

Помимо сетевой операционной системы необходимы сетевые прикладные программы, реализующие преимущества, предоставляемые сетью.

Сети на базе серверов имеют лучшие характеристики и повышенную надежность. Сервер владеет главными ресурсами сети, к которым обращаются остальные рабочие станции.

В современной клиент – серверной архитектуре выделяется четыре группы объектов: клиенты, серверы, данные и сетевые службы. Клиенты располагаются в системах на рабочих местах пользователей. Данные в основном хранятся в серверах. Сетевые службы являются совместно используемыми серверами и данными. Кроме того службы управляют процедурами обработки данных.

Сети клиент – серверной архитектуры имеют следующие преимущества:

– обеспечивают централизованное управление учетными записями пользователей, безопасностью и доступом, что упрощает сетевое администрирование;
– позволяют организовывать сети с большим количеством рабочих станций;
– обеспечивают эффективный доступ к сетевым ресурсам;
– предоставляют доступ ко всем сетевым ресурсам, на основе учетной записи пользователя.

Наряду с преимуществами, сети клиент – серверной архитектуры имеют и ряд недостатков:

– неисправность сервера может как минимум привести к потере сетевых ресурсов, а в худшем случае вывести из строя всю сеть;
– требуют квалифицированного персонала для администрирования;
– имеют более высокую стоимость сетей и сетевого оборудования.

**Выбор архитектуры сети**

Выбор архитектуры сети зависит прежде всего от назначения сети и количества рабочих станций.

Следует выбрать одноранговую архитектуру, если:

- количество пользователей не превышает десяти;
- все машины находятся близко друг от друга;
- имеют место небольшие финансовые возможности;
- нет необходимости в специализированном сервере, таком как сервер БД, факс-сервер или каком-либо другом;
- нет возможности или необходимости в централизованном администрировании.

Следует выбрать клиент серверную архитектуру, если:

- количество пользователей превышает десяти;
- требуется централизованное управление, безопасность, управление ресурсами или резервное копирование;
- необходим специализированный сервер;
- нужен доступ к глобальной сети;
- требуется разделять ресурсы на уровне пользователей.

**Вопросы по теме**

1. Дать определение сети.
2. Чем отличается коммуникационная сеть от информационной сети?
3. Как разделяются сети по территориальному признаку?
4. Что такое информационная система?
5. Что такое каналы связи?
6. Дать определение физического канала связи.
7. Дать определение логического канала связи.
8. Как называется совокупность правил обмена информацией между двумя или несколькими устройствами?
9. Как называется объект, способный осуществлять хранение, обработку или передачу данных, в состав, которого входят компьютер, программное обеспечение, пользователи и др. составляющие, предназначенные для процесса обработки и передачи данных?
10. Каким параметром характеризуется загрузка сети?
11. Что такое метод доступа?
12. Что такое совокупность правил, устанавливающих процедуры и формат обмена информацией?

13. Чем отличается рабочая станция в сети от обычного персонального компьютера?
14. Какие элементы входят в состав сети?
15. Как называется описание физических соединений в сети?
16. Что такое архитектура сети?
17. Как назвать способ определения, какая из рабочих станций сможет следующей использовать канал связи?
18. Перечислить преимущества использования сетей.
19. Чем отличается одноранговая архитектура от клиент – серверной архитектуры?
20. Каковы преимущества крупномасштабной сети с выделенным сервером?
21. Какие сервисы предоставляет клиент серверная архитектура?
22. Преимущества и недостатки архитектуры терминал – главный компьютер.
23. В каком случае используется одноранговая архитектура?
24. Что характерно для сетей с выделенным сервером?
25. Как называются рабочие станции, которые используют ресурсы сервера?
26. Что такое сервер?

## 2 Семиуровневая модель OSI

Для единого представления данных в сетях с неоднородными устройствами и программным обеспечением международная организация по стандартам ISO (International Standardization Organization) разработала базовую модель связи открытых систем OSI (Open System Interconnection). Эта модель описывает правила и процедуры передачи данных в различных сетевых средах при организации сеанса связи. Основными элементами модели являются уровни, прикладные процессы и физические средства соединения. На рис. 2.1 представлена структура базовой модели. Каждый уровень модели OSI выполняет определенную задачу в процессе передачи данных по сети. Базовая модель является основой для разработки сетевых протоколов. OSI разделяет коммуникационные функции в сети на семь уровней, каждый из которых обслуживает различные части процесса области взаимодействия открытых систем.

Модель OSI описывает только системные средства взаимодействия, не касаясь приложений конечных пользователей. Приложения реализуют свои собственные протоколы взаимодействия, обращаясь к системным средствам. Если приложение может взять на себя функции некоторых верхних уровней модели OSI, то для обмена данными оно обращается напрямую к системным средствам, выполняющим функции оставшихся нижних уровней модели OSI.

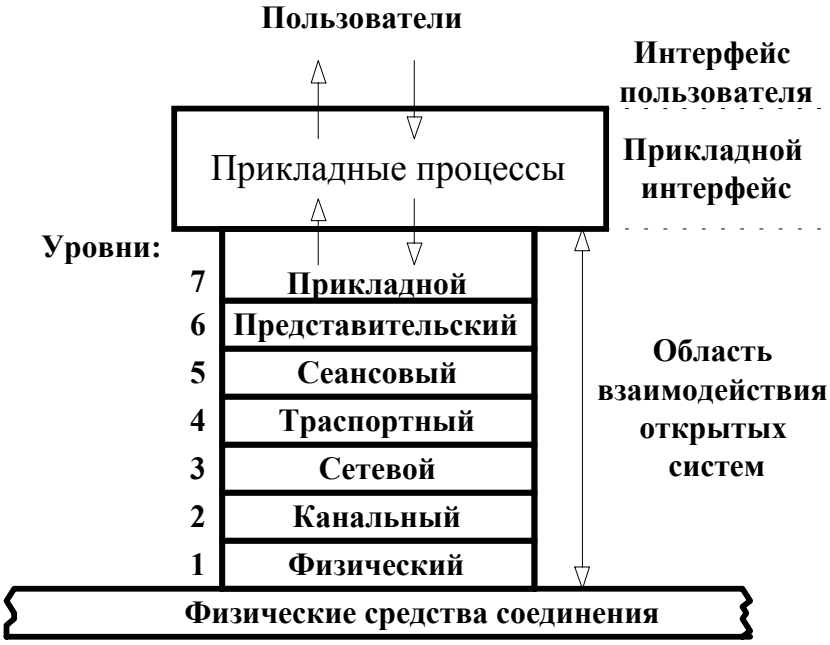


Рис. 2.1. Модель OSI

## 2.1 Взаимодействие уровней модели OSI

Модель OSI можно разделить на две различные модели, как показано на рис.2.2:

- горизонтальную модель на базе протоколов, обеспечивающую механизм взаимодействия программ и процессов на различных машинах;
- вертикальную модель на основе услуг, обеспечиваемых соседними уровнями друг другу на одной машине.

Каждый уровень компьютера–отправителя взаимодействует с таким же уровнем компьютера-получателя, как будто он связан напрямую. Такая связь называется логической или виртуальной связью. В действительности взаимодействие осуществляется между смежными уровнями одного компьютера.

Итак, информация на компьютере–отправителе должна пройти через все уровни. Затем она передается по физической среде до компьютера–получателя и опять проходит сквозь все уровни, пока не дойдет до того же уровня, с которого она была послана компьютером-отправителем.

В горизонтальной модели двум программам требуется общий протокол для обмена данными. В вертикальной модели соседние уровни обмениваются данными с использованием интерфейсов прикладных программ API (Application Programming Interface).

Перед подачей в сеть данные разбиваются на пакеты. Пакет (packet) – это единица информации, передаваемая между станциями сети. При отправке данных пакет проходит последовательно через все уровни программного обеспечения. На каждом уровне к пакету добавляется управляющая информация данного уровня (заголовок), которая необходима для успешной пере-

дачи данных по сети, как это показано на рис. 2.3, где *Заг* – заголовок пакета, *Кон* – конец пакета.

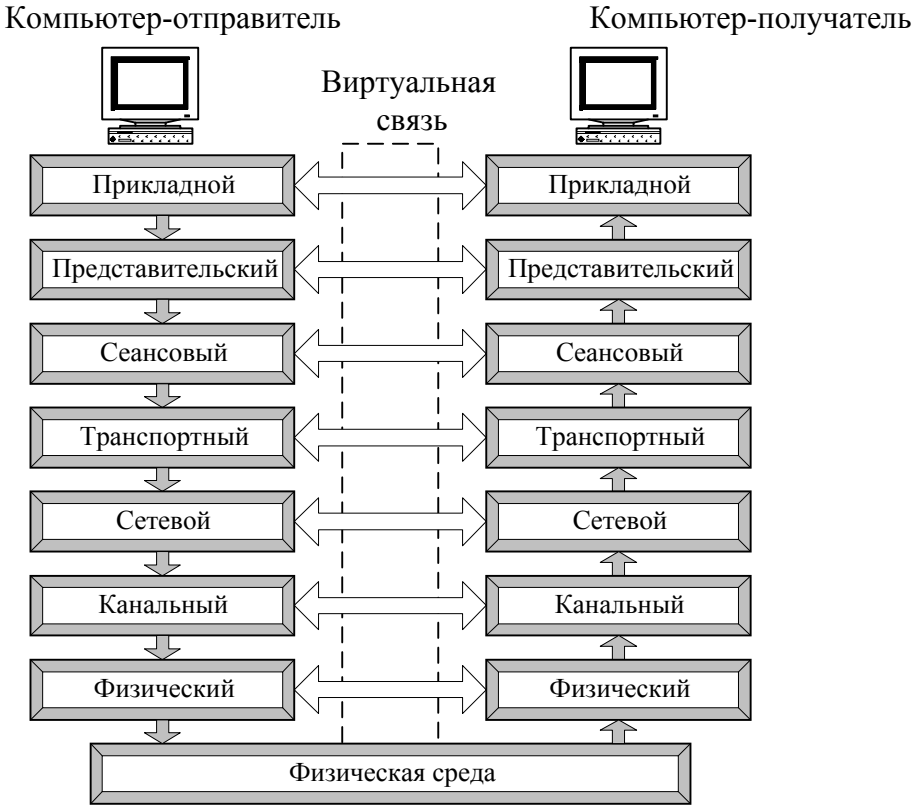


Рис. 2.2. Схема взаимодействия компьютеров в базовой эталонной модели OSI

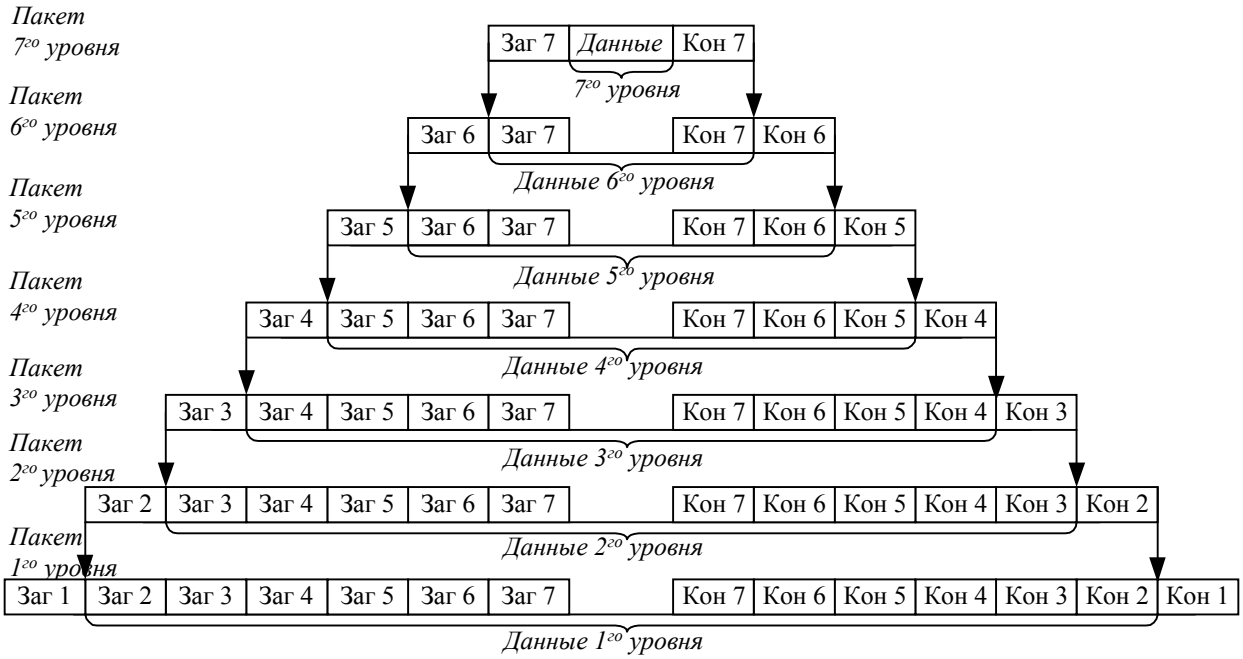


Рис. 2.3. Формирование пакета каждого уровня семиуровневой модели

На принимающей стороне пакет проходит через все уровни в обратном порядке. На каждом уровне протокол этого уровня читает информацию пакета, затем удаляет информацию, добавленную к пакету на этом же уровне от-

правляющей стороной, и передает пакет следующему уровню. Когда пакет дойдет до *Прикладного* уровня, вся управляющая информация будет удалена из пакета, и данные примут свой первоначальный вид.

Каждый уровень модели выполняет свою функцию. Чем выше уровень, тем более сложную задачу он решает.

| |
|---|
| **7. Прикладной**<br>представляет набор интерфейсов, позволяющий получить доступ к сетевым службам |
| **6. Представления**<br>преобразует данные в общий формат для передачи по сети |
| **5. Сеансовый**<br>поддержка взаимодействия (сеанса) между удаленными процессами |
| **4. Транспортный**<br>управляет передачей данных по сети, обеспечивает подтверждение передачи |
| **3. Сетевой**<br>маршрутизация, управление потоками данных, адресация сообщений для доставки, преобразование логические сетевые адреса и имена в соответствующие им физические |
| **2. Канальный**<br>**2.1. Контроль логической связи (LLC):**<br>формирование кадров<br>**2.2. Контроль доступа к среде (MAC):**<br>управление доступом к среде |
| **1. Физический:**<br>битовые протоколы передачи информации |

Рис. 2.4. Функции уровней

Каждый уровень обеспечивает сервис для вышестоящего уровня, запрашивая в свою очередь, сервис у нижестоящего уровня. Верхние уровни запрашивают сервис почти одинаково: как правило, это требование маршрутизации каких-то данных из одной сети в другую. Практическая реализация принципов адресации данных возложена на нижние уровни.

Рассматриваемая модель определяет взаимодействие открытых систем разных производителей в одной сети. Поэтому она выполняет для них координирующие действия по:

- взаимодействию прикладных процессов;
- формам представления данных;
- единообразному хранению данных;
- управлению сетевыми ресурсами;

– безопасности данных и защите информации;
– диагностике программ и технических средств.

На рис. 2.4 приведено краткое описание функций всех уровней.

## 2.2 Прикладной уровень (Application layer)

Прикладной уровень обеспечивает прикладным процессам средства доступа к области взаимодействия, является верхним (седьмым) уровнем и непосредственно примыкает к прикладным процессам. В действительности прикладной уровень – это набор разнообразных протоколов, с помощью которых пользователи сети получают доступ к разделяемым ресурсам, таким как файлы, принтеры или гипертекстовые Web-страницы, а также организуют свою совместную работу, например с помощью протокола электронной почты [30]. Специальные элементы прикладного сервиса обеспечивают сервис для конкретных прикладных программ, таких как программы пересылки файлов и эмуляции терминалов. Если, например программе необходимо переслать файлы, то обязательно будет использован *протокол передачи, доступа и управления файлами* FTAM (File Transfer, Access, and Management). В модели OSI *прикладная программа*, которой нужно выполнить конкретную задачу (например, обновить базу данных на компьютере), посылает конкретные данные в виде *Дейтаграммы* на *прикладной уровень*. Одна из основных задач этого уровня – определить, как следует обрабатывать запрос прикладной программы, другими словами, какой вид должен принять данный запрос.

Единица данных, которой оперирует прикладной уровень, обычно называется сообщением (message).

Прикладной уровень выполняет следующие функции:

1. описание форм и методов взаимодействия прикладных процессов;
2. выполнение различных видов работ: передача файлов, управление заданиями, управление системой и т.д.;
3. идентификация пользователей по их паролям, адресам, электронным подписям;
4. определение функционирующих абонентов и возможности доступа к новым прикладным процессам;
5. определение достаточности имеющихся ресурсов;
6. организация запросов на соединение с другими прикладными процессами;
7. передача заявок представительскому уровню на необходимые методы описания информации;
8. выбор процедур планируемого диалога процессов;
9. управление данными, которыми обмениваются прикладные процессы и синхронизация взаимодействия прикладных процессов;

10. определение качества обслуживания (время доставки блоков данных, допустимой частоты ошибок);
11. исправление ошибок и определение достоверности данных;
12. согласование ограничений, накладываемых на синтаксис (наборы символов, структура данных).

Указанные функции определяют виды сервиса, которые прикладной уровень предоставляет прикладным процессам. Кроме этого, прикладной уровень передает прикладным процессам сервис, предоставляемый нижестоящими уровнями.

Прикладной уровень отвечает за доступ приложений в сеть. Задачами этого уровня является перенос файлов, обмен почтовыми сообщениями и управление сетью.

К числу наиболее распространенных протоколов верхних трех уровней относятся:

– FTP (File Transfer Protocol) протокол передачи файлов;
– TFTP (Trivial File Transfer Protocol) простейший протокол пересылки файлов;
– X.400 электронная почта;
– Telnet работа с удаленным терминалом;
– SMTP (Simple Mail Transfer Protocol) простой протокол почтового обмена;
– CMIP (Common Management Information Protocol) общий протокол управления информацией;
– SLIP (Serial Line IP) IP для последовательных линий. Протокол последовательной посимвольной передачи данных;
– SNMP (Simple Network Management Protocol) простой протокол сетевого управления;
– FTAM (File Transfer, Access, and Management) протокол передачи, доступа и управления файлами.

## 2.3 Уровень представления данных (Presentation layer)

Уровень представления данных или представительский уровень представляет данные, передаваемые между прикладными процессами, в нужной форме.

Этот уровень обеспечивает то, что информация, передаваемая прикладным уровнем, будет понятна прикладному уровню в другой системе. По необходимости уровень представления в момент передачи информации выполняет преобразование форматов данных в некоторый общий формат представления, а в момент приема, соответственно, выполняет обратное преобразование. Таким образом, прикладные уровни могут преодолеть, например, син-

таксические различия в представлении данных. Такая ситуация может возникнуть в ЛВС с неоднотипными компьютерами (*IBM PC и Macintosh*), которым необходимо обмениваться данными.

В основу общего представления данных положена единая для всех уровней модели система ASN.1. Эта система служит для описания структуры файлов, а также позволяет решить проблему шифрования данных. На этом уровне может выполняться шифрование и дешифрование данных, благодаря которым секретность обмена данными обеспечивается сразу для всех прикладных сервисов. Примером такого протокола является протокол *Secure Socket Layer (SSL),* который обеспечивает секретный обмен сообщениями для протоколов прикладного уровня стека TCP/IP. Этот уровень обеспечивает преобразование данных (кодирование, компрессия и т.п.) прикладного уровня в поток информации для транспортного уровня.

Представительный уровень выполняет следующие основные функции:

1. генерация запросов на установление сеансов взаимодействия прикладных процессов.
2. согласование представления данных между прикладными процессами.
3. реализация форм представления данных.
4. представление графического материала (чертежей, рисунков, схем).
5. засекречивание данных.
6. передача запросов на прекращение сеансов.

Протоколы уровня представления данных обычно являются составной частью протоколов трех верхних уровней модели.

## 2.4 Сеансовый уровень (Session layer)

Сеансовый уровень – это уровень, определяющий процедуру проведения сеансов между пользователями или прикладными процессами.

Сеансовый уровень обеспечивает управление диалогом для того, чтобы фиксировать, какая из сторон является активной в настоящий момент, а также предоставляет средства синхронизации. Последние позволяют вставлять контрольные точки в длинные передачи, чтобы в случае отказа можно было вернуться назад к последней контрольной точке, вместо того чтобы начинать все сначала. На практике немногие приложения используют сеансовый уровень, и он редко реализуется.

Сеансовый уровень управляет передачей информации между прикладными процессами, координирует прием, передачу и выдачу одного сеанса связи. Кроме того, сеансовый уровень содержит дополнительно функции управления паролями, управления диалогом, синхронизации и отмены связи в сеансе передачи после сбоя вследствие ошибок в нижерасположенных уровнях. Функции этого уровня состоят *в координации связи* между двумя прикладными программами, работающими на разных рабочих станциях. Это

происходит в виде хорошо структурированного диалога. В число этих функций входит создание сеанса, управление передачей и приемом пакетов сообщений во время сеанса и завершение сеанса.

На сеансовом уровне определяется, какой будет передача между двумя прикладными процессами:

– *полудуплексной* (двусторонняя поочередная передача);
– *дуплексной* (двусторонняя одновременная передача).

В полудуплексном режиме сеансовый уровень выдает тому процессу, который начинает передачу, *маркер данных*. Когда второму процессу приходит время отвечать, маркер данных передается ему. Сеансовый уровень разрешает передачу только той стороне, которая обладает маркером данных.

Сеансовый уровень обеспечивает выполнение следующих функций:

1. установление и завершение на сеансовом уровне соединения между взаимодействующими системами;
2. выполнение нормального и срочного обмена данными между прикладными процессами;
3. управление взаимодействием прикладных процессов;
4. синхронизация сеансовых соединений;
5. извещение прикладных процессов об исключительных ситуациях;
6. установление в прикладном процессе меток, позволяющих после отказа либо ошибки восстановить его выполнение от ближайшей метки;
7. прерывание в нужных случаях прикладного процесса и его корректное возобновление;
8. прекращение сеанса без потери данных;
9. передача особых сообщений о ходе проведения сеанса;

Сеансовый уровень отвечает за организацию сеансов обмена данными между оконечными машинами. Протоколы сеансового уровня обычно являются составной частью протоколов трех верхних уровней модели.

## 2.5 Транспортный уровень (Transport Layer)

Транспортный уровень предназначен для передачи пакетов через коммуникационную сеть. На транспортном уровне пакеты разбиваются на блоки.

На пути от отправителя к получателю пакеты могут быть искажены или утеряны. Хотя некоторые приложения имеют собственные средства обработки ошибок, существуют и такие, которые предпочитают сразу иметь дело с надежным соединением. Работа транспортного уровня заключается в том, чтобы обеспечить приложениям или верхним уровням модели (прикладному и сеансовому) передачу данных с той степенью надежности, которая им требуется. Модель OSI определяет пять классов сервиса, предоставляемых транспортным уровнем. Эти виды сервиса отличаются качеством предоставляемых услуг: срочностью, возможностью восстановления прерванной связи,

наличием средств мультиплексирования нескольких соединений между различными прикладными протоколами через общий транспортный протокол, а главное способностью к обнаружению и исправлению ошибок передачи, таких как искажение, потеря и дублирование пакетов.

Транспортный уровень определяет адресацию физических устройств (систем, их частей) в сети. Этот уровень гарантирует доставку блоков информации адресатам и управляет этой доставкой. Его главной задачей является обеспечение эффективных, удобных и надежных форм передачи информации между системами. Когда в процессе обработки находится более одного пакета, транспортный уровень контролирует очередность прохождения пакетов. Если проходит дубликат принятого ранее сообщения, то данный уровень опознает это и игнорирует сообщение.

В функции транспортного уровня входит:

1. управление передачей по сети и обеспечение целостности блоков данных;
2. обнаружение ошибок, частичная их ликвидация и сообщение о неисправленных ошибках;
3. восстановление передачи после отказов и неисправностей;
4. укрупнение или разделение блоков данных;
5. предоставление приоритетов при передаче блоков (нормальная или срочная).
6. подтверждение передачи;
7. ликвидация блоков при тупиковых ситуациях в сети.

Начиная с транспортного уровня, все вышестоящие протоколы реализуются программными средствами, обычно включаемыми в состав сетевой ОС.

Наиболее распространенными протоколами транспортного уровня являются:

- TCP (Transmission Control Protocol) протокол управления передачей стека TCP/IP;
- UDP (User Datagram Protocol) пользовательский протокол дейтаграмм стека TCP/IP;
- NCP (NetWare Core Protocol) базовый протокол сетей NetWare;
- SPX (Sequenced Packet eXchange) упорядоченный обмен пакетами стека Novell;
- TP4 (Transmission Protocol) – протокол передачи класса 4.

## 2.6 Сетевой уровень (Network Layer)

Сетевой уровень обеспечивает прокладку каналов, соединяющих абонентские и административные системы через коммуникационную сеть, выбор маршрута наиболее быстрого и надежного пути.

Сетевой уровень устанавливает связь в вычислительной сети между двумя системами и обеспечивает прокладку виртуальных каналов между ними. *Виртуальный или логический канал* - это такое функционирование компонентов сети, которое создает взаимодействующим компонентам иллюзию прокладки между ними нужного тракта. Кроме этого, сетевой уровень сообщает транспортному уровню о появляющихся ошибках. Сообщения сетевого уровня принято называть *пакетами* (packet). В них помещаются фрагменты данных. Сетевой уровень отвечает за их адресацию и доставку.

Прокладка наилучшего пути для передачи данных называется *маршрутизацией*, и ее решение является главной задачей сетевого уровня. Эта проблема осложняется тем, что самый короткий путь не всегда самый лучший. Часто критерием при выборе маршрута является время передачи данных по этому маршруту; оно зависит от пропускной способности каналов связи и интенсивности трафика, которая может изменяться с течением времени. Некоторые алгоритмы маршрутизации пытаются приспособиться к изменению нагрузки, в то время как другие принимают решения на основе средних показателей за длительное время. Выбор маршрута может осуществляться и по другим критериям, например, надежности передачи.

Протокол канального уровня обеспечивает доставку данных между любыми узлами только в сети с соответствующей *типовой топологией*. Это очень жесткое ограничение, которое не позволяет строить сети с развитой структурой, например, сети, объединяющие несколько сетей предприятия в единую сеть, или высоконадежные сети, в которых существуют избыточные связи между узлами.

Таким образом, внутри сети доставка данных регулируется канальным уровнем, а вот доставкой данных между сетями занимается сетевой уровень. При организации доставки пакетов на сетевом уровне используется понятие *номер сети.* В этом случае *адрес* получателя состоит из *номера сети* и *номера компьютера* в этой сети.

Сети соединяются между собой специальными устройствами, называемыми маршрутизаторами. *Маршрутизатор* – это устройство, которое собирает информацию о топологии межсетевых соединений и на ее основании пересылает пакеты сетевого уровня в сеть назначения. Для того чтобы передать сообщение от отправителя, находящегося в одной сети, получателю, находящемуся в другой сети, нужно совершить некоторое количество транзитных передач (hops) между сетями, каждый раз, выбирая подходящий маршрут. Таким образом, маршрут представляет собой последовательность маршрутизаторов, по которым проходит пакет.

Сетевой уровень отвечает за деление пользователей на группы и маршрутизацию пакетов на основе преобразования MAC-адресов в сетевые адреса. Сетевой уровень обеспечивает также прозрачную передачу пакетов на транспортный уровень.

Сетевой уровень выполняет функции:

1. создание сетевых соединений и идентификация их портов;
2. обнаружение и исправление ошибок, возникающих при передаче через коммуникационную сеть;
3. управление потоками пакетов;
4. организация (упорядочение) последовательностей пакетов;
5. маршрутизация и коммутация;
6. сегментирование и объединение пакетов.

На сетевом уровне определяется два вида протоколов. Первый вид относится к определению *правил передачи пакетов* с данными конечных узлов от узла к маршрутизатору и между маршрутизаторами. Именно эти протоколы обычно имеют в виду, когда говорят о протоколах сетевого уровня. Однако часто к сетевому уровню относят и другой вид протоколов, называемых *протоколами обмена маршрутной информацией*. С помощью этих протоколов маршрутизаторы собирают информацию о топологии межсетевых соединений.

Протоколы сетевого уровня реализуются программными модулями операционной системы, а также программными и аппаратными средствами маршрутизаторов.

Наиболее часто на сетевом уровне используются протоколы:

– IP (Internet Protocol) протокол Internet, сетевой протокол стека TCP/IP, который предоставляет адресную и маршрутную информацию;
– IPX (Internetwork Packet Exchange) протокол межсетевого обмена пакетами, предназначенный для адресации и маршрутизации пакетов в сетях Novell;
– X.25 международный стандарт для глобальных коммуникаций с коммутацией пакетов (частично этот протокол реализован на уровне 2);
– CLNP (Connection Less Network Protocol) сетевой протокол без организации соединений.

## 2.7 Канальный уровень (Data Link)

Единицей информации канального уровня являются *кадры (frame)*. Кадры – это логически организованная структура, в которую можно помещать данные. Задача канального уровня передавать кадры от сетевого уровня к физическому уровню.

На физическом уровне просто пересылаются биты. При этом не учитывается, что в некоторых сетях, в которых линии связи используются попеременно несколькими парами взаимодействующих компьютеров, физическая среда передачи может быть занята. Поэтому одной из задач канального уровня является проверка доступности среды передачи. Другой задачей канального уровня является реализация механизмов обнаружения и коррекции ошибок.

Канальный уровень обеспечивает корректность передачи каждого кадра, помещая специальную последовательность бит, в его начало и конец, а также вычисляет контрольную сумму и добавляет её к кадру. Получатель производит повторное вычисление контрольной суммы и фиксирует ошибку в случае ее несовпадения.

Задача канального уровня - брать пакеты, поступающие с сетевого уровня и готовить их к передаче, «укладывая» в кадр соответствующего размера. Этот уровень обязан определить, где начинается и где заканчивается блок, а также обнаруживать ошибки передачи.

На этом же уровне определяются правила использования физического уровня узлами сети. Электрическое представление данных в ЛВС (биты данных, методы кодирования данных и маркеры) распознаются на этом и только на этом уровне. Здесь обнаруживаются и исправляются (путем требований повторной передачи данных) ошибки.

Канальный уровень обеспечивает создание, передачу и прием кадров данных. Этот уровень обслуживает запросы сетевого уровня и использует сервис физического уровня для приема и передачи пакетов. Спецификации IEEE 802.X делят канальный уровень на два подуровня:

– *LLC* (*Logical Link Control*) управление логическим каналом – осуществляет логический контроль связи. Подуровень LLC обеспечивает обслуживание сетевого уровня и связан с передачей и приемом пользовательских сообщений.
– *MAC* (*Media Assess Control)* контроль доступа к среде – регулирует доступ к разделяемой физической среде (передача маркера или обнаружение коллизий или столкновений) и управляет доступом к каналу связи. Подуровень *LLC* находится выше подуровня *MAC.*

Канальный уровень определяет доступ к среде и управление передачей посредством процедуры передачи данных по каналу. При больших размерах передаваемых блоков данных канальный уровень делит их на кадры и передает кадры в виде последовательностей. При получении кадров уровень формирует из них переданные блоки данных. Размер блока данных зависит от способа передачи, качества канала, по которому он передается.

В локальных сетях протоколы канального уровня используются компьютерами, мостами, коммутаторами и маршрутизаторами. В компьютерах функции канального уровня реализуются совместными усилиями сетевых адаптеров и их драйверов.

Канальный уровень может выполнять следующие виды функций:

1. организация (установление, управление, расторжение) канальных соединений и идентификация их портов;
2. организация и передача кадров;
3. обнаружение и исправление ошибок;
4. управление потоками данных;

5. обеспечение прозрачности логических каналов (передачи по ним данных, закодированных любым способом).

Наиболее часто используемые протоколы на канальном уровне включают:

– HDLC (High Level Data Link Control) протокол управления каналом передачи данных высокого уровня, для последовательных соединений;
– IEEE 802.2 LLC (тип I и тип II) обеспечивают MAC для сред 802.x;
– Ethernet сетевая технология по стандарту IEEE 802.3 для сетей, использующая шинную топологию и коллективный доступ с прослушиванием несущей и обнаружением конфликтов;
– Token ring сетевая технология по стандарту IEEE 802.5, использующая кольцевую топологию и метод доступа к кольцу с передачей маркера;
– FDDI (Fiber Distributed Date Interface Station) сетевая технология по стандарту IEEE 802.6, использующая оптоволоконный носитель;
– X.25 международный стандарт для глобальных коммуникаций с коммутацией пакетов;
– Frame relay сеть, организованная из технологий X25 и ISDN.

## 2.8 Физический уровень (Physical Layer)

Физический уровень предназначен для сопряжения с *физическими средствами соединения*. *Физические средства соединения* – это совокупность *физической среды*, аппаратных и программных средств, обеспечивающая передачу сигналов между системами. *Физическая среда* – это материальная субстанция, через которую осуществляется передача сигналов. Физическая среда является основой, на которой строятся физические средства соединения. В качестве физической среды широко используются эфир, металлы, оптическое стекло и кварц.

Физический уровень состоит из подуровней стыковки со средой и преобразования передачи.

Первый из них обеспечивает сопряжение потока данных с используемым физическим каналом связи. Второй осуществляет преобразования, связанные с применяемыми протоколами. Физический уровень обеспечивает физический интерфейс с каналом передачи данных, а также описывает процедуры передачи сигналов в канал и получения их из канала. На этом уровне определяются электрические, механические, функциональные и процедурные параметры для физической связи в системах. Физический уровень получает пакеты данных от вышележащего канального уровня и преобразует их в оптические или электрические сигналы. Эти сигналы посылаются через среду передачи на приемный узел. Механические и электрические (оптические) свойства среды передачи, определяемые на физическом уровне, включают:

– тип кабелей и разъемов;

– разводку контактов в разъемах;
– схему кодирования сигналов для значений 0 и 1.

Физический уровень выполняет следующие функции:

1. установление и разъединение физических соединений;
2. передача сигналов в последовательном коде и прием;
3. прослушивание, в нужных случаях, каналов;
4. идентификация каналов;
5. оповещение о появлении неисправностей и отказов.

Оповещение о появлении неисправностей и отказов связано с тем, что на физическом уровне происходит обнаружение определенного класса событий, мешающих нормальной работе сети (столкновение кадров, посланных сразу несколькими системами, обрыв канала, отключение питания, потеря механического контакта и т. д.). Виды сервиса, предоставляемого канальному уровню, определяются протоколами физического уровня. Прослушивание канала необходимо в тех случаях, когда к одному каналу подключается группа систем, но одновременно передавать сигналы разрешается только одной из них. Поэтому прослушивание канала позволяет определить, свободен ли он для передачи. В ряде случаев для более четкого определения структуры физический уровень разбивается на несколько подуровней. Например, физический уровень беспроводной сети делится на три подуровня рис. 2.5.

| | |
|---|---|
| **1с** | **Подуровень, не зависимый от физических средств соединения** |
| **1б** | **Переходный подуровень,** |
| **1а** | **Подуровень, зависимый от физических средств соединения** |

Рис. 2.5. Физический уровень беспроводной локальной сети

Функции физического уровня реализуются во всех устройствах, подключенных к сети. Со стороны компьютера функции физического уровня выполняются сетевым адаптером. Повторители являются единственным типом оборудования, которое работает только на физическом уровне.

На физическом уровне выполняется преобразование данных, поступающих от более высокого уровня, в сигналы передаваемые по кабелю. В глобальных сетях на этом уровне могут использоваться модемы и интерфейс *RS-232C*. В локальных сетях для преобразования данных применяют сетевые адаптеры.

Физический уровень может обеспечивать как асинхронную так и синхронную передачу. На физическом уровне должна быть определена схема

кодирования для представления двоичных значений с целью их передачи по каналу связи. Во многих ЛВС используется манчестерское кодирование.

Примером протокола физического уровня может служить спецификация 10Base-T технологии Ethernet, которая определяет в качестве используемого кабеля неэкранированную витую пару категории 3 с волновым сопротивлением 100 Ом, разъем RJ-45, максимальную длину физического сегмента 100 метров, манчестерский код для представления данных на кабеле, и другие характеристики среды и электрических сигналов.

К числу наиболее распространенных спецификаций физического уровня относятся:

– EIA-RS-232-C, CCITT V.24/V.28 - механические/электрические характеристики несбалансированного последовательного интерфейса;
– EIA-RS-422/449, CCITT V.10 - механические, электрические и оптические характеристики сбалансированного последовательного интерфейса;
– Ethernet – сетевая технология по стандарту IEEE 802.3 для сетей, использующая шинную топологию и коллективный доступ с прослушиванием несущей и обнаружением конфликтов;
– Token ring – сетевая технология по стандарту IEEE 802.5, использующая кольцевую топологию и метод доступа к кольцу с передачей маркера;

## 2.9 Зависящие и независящие от технической реализации сети уровни

Функции всех уровней модели OSI могут быть отнесены к одной из двух групп: либо к функциям, зависящим от конкретной технической реализации сети, либо к функциям, ориентированным на работу с приложениями.

Три нижних уровня физический, канальный и сетевой являются зависящими реализации сети и протоколы этих уровней тесно связаны с технической реализацией сети, с используемым коммуникационным оборудованием. Т.е. переход на какое-либо другое оборудование означает смену протоколов этих уровня во всех узлах сети.

Три верхних уровня сеансовый, уровень представления и прикладной ориентированы на приложения и мало зависят от технических особенностей построения сети. На протоколы этих уровней не влияют никакие изменения в топологии сети, замена оборудования или переход на другую сетевую технологию.

Транспортный уровень является промежуточным, он скрывает все детали функционирования нижних уровней от верхних уровней. Это позволяет разрабатывать приложения, не зависящие от технических средств, непосредственно занимающихся транспортировкой сообщений.

Одна рабочая станция взаимодействует с другой рабочей станцией посредством протоколов всех семи уровней. Это взаимодействие станции осуществляют через различные коммуникационные устройства: концентраторы,

модемы, мосты, коммутаторы, маршрутизаторы, мультиплексоры. В зависимости от типа коммуникационное устройство может работать:

- либо только на физическом уровне (повторитель);
- либо на физическом и канальном уровнях (мост);
- либо на физическом, канальном и сетевом уровнях, иногда захватывая и транспортный уровень (маршрутизатор).

Модель OSI представляет собой хотя и очень важную, но только одну из многих моделей коммуникаций. Эти модели и связанные с ними стеки протоколов могут отличаться количеством уровней, их функциями, форматами сообщений, сервисами, предоставляемыми на верхних уровнях, и прочими параметрами.

### 2.10 Стеки коммуникационных протоколов

Иерархически организованная совокупность протоколов, решающих задачу взаимодействия узлов сети, называется *стеком коммуникационных протоколов*.

Протоколы соседних уровней, находящихся в одном узле, взаимодействуют друг с другом также в соответствии с четко определенными правилами и с помощью стандартизованных форматов сообщений. Эти правила принято называть *интерфейсом.* Интерфейс определяет набор услуг, которые нижележащий уровень предоставляет вышележащему уровню.

### Вопросы по теме

1. Что такое OSI?
2. Каково назначение базовой модели взаимодействия открытых систем?
3. На какие уровни разбита базовая модель OSI?
4. Какие функции несет уровень в модели взаимодействия открытых систем?
5. На какие единицы разбивается информация для передачи данных по сети?
6. Что обеспечивает горизонтальная составляющая модели взаимодействия открытых систем?
7. Какие элементы являются основными элементами для базовой модели взаимодействия открытых систем?
8. Какие функции выполняются на физическом уровне?
9. Какие вопросы решаются на физическом уровне?
10. Какой уровень модели OSI преобразует данные в общий формат для передачи по сети?
11. Какое оборудование используется на физическом уровне?
12. Какие известны спецификации физического уровня?

13. Перечислить функции канального уровня.
14. Какие функции канального уровня?
15. На какие подуровни разделяется канальный уровень и каковы их функции?
16. Функцией какого уровня является засекречивание и реализация форм представления данных?.
17. Какие протоколы используются на канальном уровне?
18. Какое оборудование используется на канальном уровне?
19. Какие функции выполняются и какие протоколы используются на сетевом уровне?
20. Какое оборудование используется на сетевом уровне?
21. Перечислить функции транспортного уровня.
22. Какие протоколы используются на транспортном уровне?
23. Перечислить оборудование транспортного уровня.
24. Дать определение сеансового уровня.
25. Какой уровень отвечает за доступ приложений в сеть?
26. Задачи уровня представления данных.
27. Перечислить функции прикладного уровня.
28. Перечислить протоколы верхних уровней.
29. Дать определение стандартных стеков коммуникационных протоколов.

# 3 Стандарты и стеки протоколов

## 3.1 Спецификации стандартов

Спецификации Institute of Electrical and Electronics Engineers IEEE802 определяют стандарты для физических компонентов сети. Эти компоненты – сетевая карта (Network Interface Card – NIC) и сетевой носитель (network media), которые относятся к физическому и канальному уровням модели OSI. Спецификации IEEE802 определяют механизм доступа адаптера к каналу связи и механизм передачи данных. Стандарты IEEE802 подразделяют канальный уровень на подуровни:

– Logical Link Control (LLC) – подуровень управления логической связью;
– Media Access Control (MAC) – подуровень управления доступом к устройствам.

Спецификации IEEE802 делятся на двенадцать стандартов.

**Стандарт 802.1**

Стандарт 802.1 (Internetworking – объединение сетей) задает механизмы управления сетью на MAC – уровне. В разделе 802.1 приводятся основные понятия и определения, общие характеристики и требования к локальным сетям, а также поведение маршрутизации на канальном уровне, где логические адреса должны быть преобразованы в их физические адреса и наоборот.

**Стандарт 802.2**

Стандарт 802.2 (Logical Link Control – управление логической связью) определяет функционирование подуровня LLC на канальном уровне модели OSI. LLC обеспечивает интерфейс между методами доступа к среде и сетевым уровнем.

**Стандарт 802.3**

Стандарт 802.3 (Ethernet Carrier Sense Multiple Access with Collision Detection – CSMA/CD LANs Ethernet – множественный доступ к сетям Ethernet с проверкой несущей и обнаружением конфликтов) описывает физический уровень и подуровень MAC для сетей, использующих шинную топологию и коллективный доступ с прослушиванием несущей и обнаружением конфликтов. Прототипом этого метода является метод доступа стандарта Ethernet (10BaseT, 10Base2, 10Base5). Метод доступа CSMA/CD. 802.3 также включает технологии Fast Ethernet (100BaseTx, 100BaseFx, 100BaseFl).

100Base-Tx – двухпарная витая пара. Использует метод MLT-3 для передачи сигналов 5-битовых порций кода 4B/5B по витой паре, а также имеется функция автопереговоров (Auto-negotiation) для выбора режима работы порта.

100Base-T4 – четырехпарная витая пара. Вместо кодирования 4B/5B в этом методе используется кодирование 8B/6T.

100BaseFx – многомодовое оптоволокно. Эта спецификация определяет работу протокола Fast Ethernet по многомодовому оптоволокну в полудуплексном и полнодуплексном режимах на основе хорошо проверенной схемы кодирования и передачи оптических сигналов, использующейся уже на протяжении ряда лет в стандарте FDDI. Как и в стандарте FDDI, каждый узел соединяется с сетью двумя оптическими волокнами, идущими от приемника (Rx) и от передатчика (Tx).

Этот метод доступа используется в сетях с общей шиной (к которым относятся и радиосети, породившие этот метод). Все компьютеры такой сети имеют непосредственный доступ к общей шине (т.е. общая шина работает в режиме *коллективного доступа* (multiply access – MA)), поэтому она может быть использована для передачи данных между любыми двумя узлами сети. Простота схемы подключения – это один из факторов, определивших успех стандарта Ethernet.

Метод доступа CSMA/CD определяет основные временные и логические соотношения, гарантирующие корректную работу всех станций в сети.

Все данные, передаваемые по сети, помещаются в кадры определенной структуры и снабжаются уникальным адресом станции назначения. Затем кадр передается по кабелю. Все станции, подключенные к кабелю, могут распознать факт передачи кадра, и та станция, которая узнает собственный адрес в заголовках кадра, записывает его содержимое в свой внутренний буфер, обрабатывает полученные данные и посылает по кабелю кадр-ответ. Адрес станции–источника также включен в исходный кадр, поэтому станция-получатель знает, кому нужно послать ответ.

**Стандарт 802.4**

Стандарт 802.4 (Token Bus LAN – локальные сети Token Bus) определяет метод доступа к шине с передачей маркера, прототип – ArcNet.

При подключении устройств в ArcNet применяют топологию «шина» или «звезда». Адаптеры ArcNet поддерживают метод доступа Token Bus (маркерная шина) и обеспечивают производительность 2,5 Мбит/с. Этот метод предусматривает следующие правила:

– все устройства, подключённые к сети, могут передавать данные, только получив разрешение на передачу (маркер);
– в любой момент времени только одна станция в сети обладает таким правом;
– кадр, передаваемый одной станцией, одновременно анализируется всеми остальными станциями сети.

В сетях ArcNet используется асинхронный метод передачи данных (в сетях Ethernet и Token Ring применяется синхронный метод), т. е. передача каждого байта в ArcNet выполняется посылкой ISU (Information Symbol Unit – единица передачи информации), состоящей из трёх служебных старт/стоповых битов и восьми битов данных.

**Стандарт 802.5**

Стандарт 802.5 (Token Ring LAN – локальные сети Token Ring) описывает метод доступа к кольцу с передачей маркера, прототип – Token Ring.

Сети стандарта Token Ring, так же как и сети Ethernet, используют разделяемую среду передачи данных, которая состоит из отрезков кабеля, соединяющих все станции сети в кольцо. Кольцо рассматривается как общий разделяемый ресурс, и для доступа к нему используется не случайный алгоритм, как в сетях Ethernet, а детерминированный, основанный на передаче станциями права на использование кольца в определенном порядке. Право на использование кольца передается с помощью кадра специального формата, называемого маркером, или токеном.

**Стандарт 802.6**

Стандарт 802.6 (Metropolitan Area Network – городские сети) описывает рекомендации для региональных сетей.

**Стандарт 802.7**

Стандарт 802.7 (Broadband Technical Advisory Group – техническая консультационная группа по широковещательной передаче) описывает рекомендации по широкополосным сетевым технологиям, носителям, интерфейсу и оборудованию.

**Стандарт 802.8**

Стандарт 802.8 (Fiber Technical Advisory Group – техническая консультационная группа по оптоволоконным сетям) содержит обсуждение использования оптических кабелей в сетях 802.3 – 802.6, а также рекомендации по оптоволоконным сетевым технологиям, носителям, интерфейсу и оборудованию, прототип – сеть FDDI (Fiber Distributed Data Interface).

Стандарт FDDI использует оптоволоконный кабель и доступ с применением маркера. Сеть FDDI строится на основе двух оптоволоконных колец, которые образуют основной и резервный пути передачи данных между узлами сети. Использование двух колец – это основной способ повышения отказоустойчивости в сети FDDI, и узлы, которые хотят им воспользоваться, должны быть подключены к обоим кольцам. Скорость сети до 100 Мб/с. Данная технология позволяет включать до 500 узлов на расстоянии 100 км.

**Стандарт 802.9**

Стандарт 802.9 (Integrated Voice and Data Network – интегрированные сети передачи голоса и данных) задает архитектуру и интерфейсы устройств одновременной передачи данных и голоса по одной линии, а также содержит рекомендации по гибридным сетям, в которых объединяют голосовой трафик и трафик данных в одной и той же сетевой среде.

**Стандарт 802.10**

В стандарте 802.10 (Network Security – сетевая безопасность) рассмотрены вопросы обмена данными, шифрования, управления сетями и безопасности в сетевых архитектурах, совместимых с моделью OSI.

**Стандарт 802.11**

Стандарт 802.11 (Wireless Network – беспроводные сети) описывает рекомендации по использованию беспроводных сетей.

**Стандарт 802.12**

Стандарт 802.12 описывает рекомендации по использованию сетей 100VG – AnyLAN со скоростью100Мб/с и методом доступа по очереди запросов и по приоритету (Demand Priority Queuing – DPQ, Demand Priority Access – DPA).

Технология 100VG – это комбинация *Ethernet* и *Token-Ring* со скоростью передачи 100 Мбит/с, работающая на неэкранированных витых парах. В проекте 100Base-VG усовершенствован метод доступа с учетом потребности мультимедийных приложений. В спецификации *100VG* предусматривается поддержка волоконно-оптических кабельных систем. Технология *100VG* использует метод доступа – обработка запросов по приоритету (demand priority access). В этом случае узлам сети предоставляется право равного доступа. Концентратор опрашивает каждый порт и проверяет наличие запроса на передачу, а затем разрешает этот запрос в соответствии с приоритетом. Имеется два уровня приоритетов – высокий и низкий.

## 3.2 Протоколы и стеки протоколов

Согласованный набор протоколов разных уровней, достаточный для организации межсетевого взаимодействия, называется *стеком протоколов*. Для каждого уровня определяется набор функций–запросов для взаимодействия с выше лежащим уровнем, который называется *интерфейсом*. Правила взаимодействия двух машин могут быть описаны в виде набора процедур для каждого из уровней, которые называются *протоколами*.

Существует достаточно много стеков протоколов, широко применяемых в сетях. Это и стеки, являющиеся международными и национальными стандартами, и фирменные стеки, получившие распространение благодаря распространенности оборудования той или иной фирмы. Примерами популярных стеков протоколов могут служить стек IPX/SPX фирмы Novell, стек TCP/IP, используемый в сети Internet и во многих сетях на основе операционной системы UNIX, стек OSI международной организации по стандартизации, стек DECnet корпорации Digital Equipment и некоторые другие.

Стеки протоколов разбиваются на три уровня:

– сетевые;
– транспортные;
– прикладные.

**Сетевые протоколы**

Сетевые протоколы предоставляют следующие услуги: адресацию и маршрутизацию информации, проверку на наличие ошибок, запрос повторной передачи и установление правил взаимодействия в конкретной сетевой среде. Ниже приведены наиболее популярные сетевые протоколы.

- **DDP** (Datagram Delivery Protocol – Протокол доставки дейтаграмм). Протокол передачи данных Apple, используемый в Apple Talk.
- **IP** (Internet Protocol – Протокол Internet). Протокол стека TCP/IP, обеспечивающий адресную информацию и информацию о маршрутизации.
- **IPX** (Internetwork Packet eXchange – Межсетевой обмен пакетами) в NWLink. Протокол Novel NetWare, используемый для маршрутизации и направления пакетов.
- **NetBEUI** (NetBIOS Extended User Interface – расширенный пользовательский интерфейс базовой сетевой системы ввода вывода)**.** Разработанный совместно IBM и Microsoft, этот протокол обеспечивает транспортные услуги для **NetBIOS.**

**Транспортные протоколы**

Транспортные протоколы предоставляют услуги надежной транспортировки данных между компьютерами. Ниже приведены наиболее популярные транспортные протоколы.

- **ATP** (Apple Talk Protocol – Транзакционный протокол Apple Talk) и **NBP** (Name Binding Protocol – Протокол связывания имен). Сеансовый и транспортный протоколы Apple Talk.
- **NetBIOS (**Базовая сетевая система ввода вывода)**.** NetBIOS Устанавливает соединение между компьютерами, а **NetBEUI** предоставляет услуги передачи данных для этого соединения.
- **SPX** (Sequenced Packet eXchange – Последовательный обмен пакетами) в NWLink. Протокол Novel NetWare, используемый для обеспечения доставки данных.
- **TCP** (Transmission Control Protocol – Протокол управления передачей). Протокол стека TCP/IP, отвечающий за надежную доставку данных.

**Прикладные протоколы**

Прикладные протоколы отвечают за взаимодействие приложений. Ниже приведены наиболее популярные прикладные протоколы.

- **AFP** (Apple Talk File Protocol – Файловый протокол Apple Talk). Протокол удаленного управления файлами Macintosh.
- **FTP** (File Transfer Protocol – Протокол передачи файлов). Протокол стека TCP/IP, используемый для обеспечения услуг по передачи файлов.
- **NCP** (NetWare Core Protocol – Базовый протокол NetWare). Оболочка и редиректоры клиента Novel NetWare.
- **SNMP** (Simple Network Management Protocol – Простой протокол управления сетью). Протокол стека TCP/IP, используемый для управления и наблюдения за сетевыми устройствами.

– **HTTP** (Hyper Text Transfer Protocol) – протокол передачи гипертекста и другие протоколы.

## 3.3 Стек OSI

Следует различать стек протоколов OSI и модель OSI рис.3.1. Стек OSI – это набор вполне конкретных спецификаций протоколов, образующих согласованный стек протоколов. Этот стек протоколов поддерживает правительство США в своей программе GOSIP. Стек OSI в отличие от других стандартных стеков полностью соответствует модели взаимодействия OSI и включает спецификации для всех семи уровней модели взаимодействия открытых систем

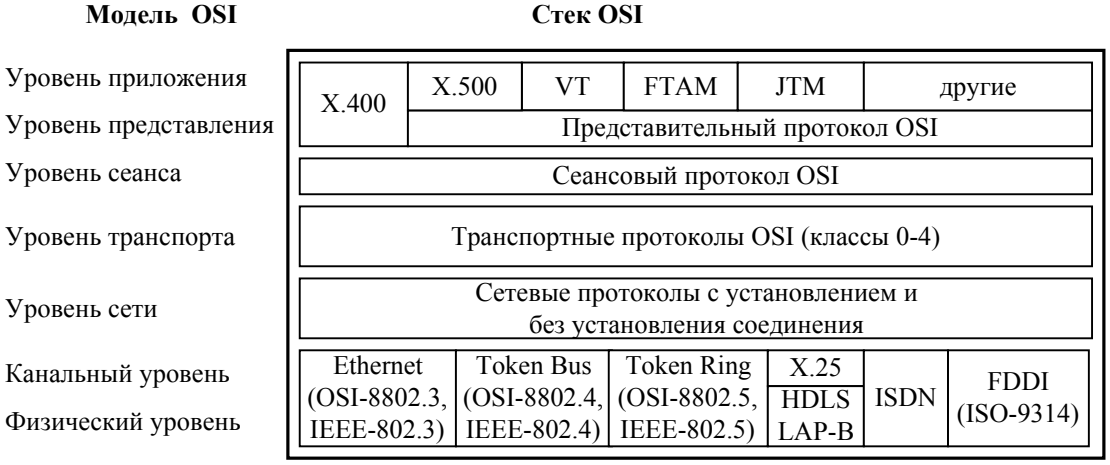


Рис. 3.1. Стек OSI

На *физическом и канальном уровнях* стек OSI поддерживает спецификации Ethernet, Token Ring, FDDI, а также протоколы LLC, X.25 и ISDN.

На *сетевом* уровне реализованы протоколы, как без установления соединений, так и с установлением соединений.

*Транспортный* протокол стека OSI скрывает различия между сетевыми сервисами с установлением соединения и без установления соединения, так что пользователи получают нужное качество обслуживания независимо от нижележащего сетевого уровня. Чтобы обеспечить это, транспортный уровень требует, чтобы пользователь задал нужное качество обслуживания. Определены 5 классов транспортного сервиса, от низшего класса 0 до высшего класса 4, которые отличаются степенью устойчивости к ошибкам и требованиями к восстановлению данных после ошибок.

Сервисы *прикладного уровня* включают передачу файлов, эмуляцию терминала, службу каталогов и почту. Из них наиболее перспективными являются служба каталогов (стандарт X.500), электронная почта (X.400), протокол виртуального терминала (VT), протокол передачи, доступа и управления файлами (FTAM), протокол пересылки и управления работами (JTM). В последнее время ISO сконцентрировала свои усилия именно на сервисах верхнего уровня.

## 3.4 Архитектура стека протоколов Microsoft TCP/IP

Набор многоуровневых протоколов, или как называют стек *TCP/IP*, предназначен для использования в различных вариантах сетевого окружения. Стек *TCP/IP* с точки зрения системной архитектуры соответствует эталонной модели *OSI* и позволяет обмениваться данными по сети приложениям и службам, работающим практически на любой платформе, включая Unix, Windows, Macintosh и другие.

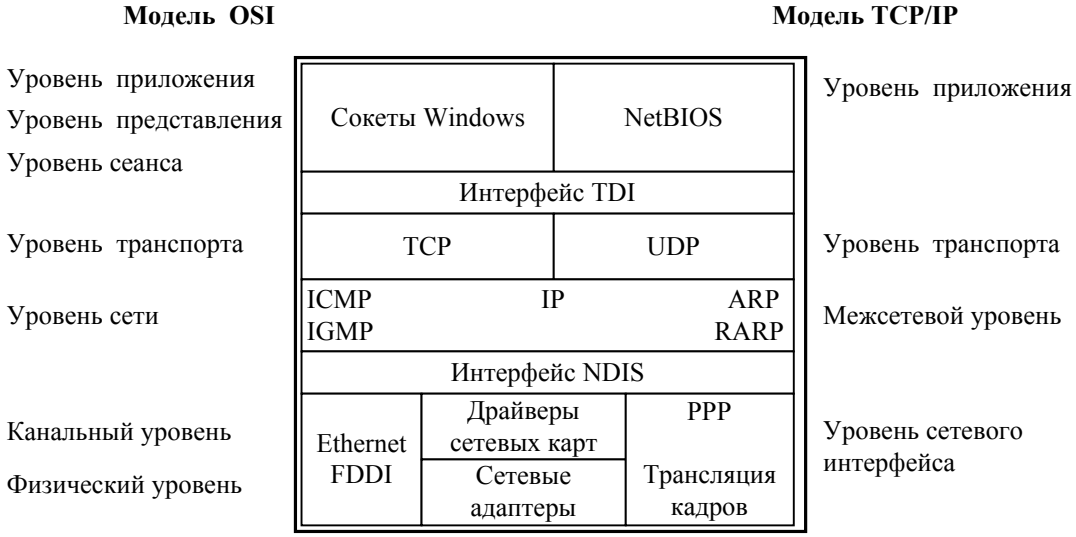


Рис. 3.2. Соответствие семиуровневой модели OSI и четырехуровневой модели TCP/IP

Реализация TCP/IP фирмы Microsoft [1] соответствует четырехуровневой модели вместо семиуровневой модели, как показано на рис. 3.2. Модель TCP/IP включает большее число функций на один уровень, что приводит к уменьшению числа уровней. В модели используются следующие уровни:

- уровень *Приложения* модели TCP/IP соответствует уровням *Приложения, Представления* и *Сеанса* модели OSI;
- уровень *Транспорта* модели TCP/IP соответствует аналогичному уровню *Транспорта* модели OSI;
- *межсетевой* уровень модели TCP/IP выполняет те же функции, что и уровень *Сети* модели OSI;
- уровень сетевого интерфейса модели TCP/IP соответствует *Канальному* и *Физическому* уровням модели OSI.

### Уровень приложения

Через уровень *Приложения* модели TCP/IP приложения и службы получают доступ к сети. Доступ к протоколам TCP/IP осуществляется посредством двух программных интерфейсов (API – Application Programming Interface):

- Сокеты Windows;
- NetBIOS.

Интерфейс *сокетов Windows,* или как его называют *WinSock*, является сетевым программным интерфейсом, предназначенным для облегчения взаимодействия между различными TCP/IP – приложениями и семействами протоколов.

Интерфейс *NetBIOS* используется для связи между процессами (IPC – Interposes Communications) служб и приложений ОС Windows. *NetBIOS* выполняет три основных функции:

- определение имен NetBIOS;
- служба дейтаграмм NetBIOS;
- служба сеанса NetBIOS.

В таблице 3.1 приведено семейство протоколов TCP/IP.

Таблица 3.1. Семейство протоколов TCP/IP.

| Протокол | Описание протокола |
|---|---|
| WinSock | Сетевой программный интерфейс |
| NetBIOS | Связь с приложениями ОС Windows |
| TDI | Интерфейс транспортного драйвера (Transport Driver Interface) позволяет создавать компоненты сеансового уровня. |
| TCP | Протокол управления передачей (Transmission Control Protocol) |
| UDP | Протокол пользовательских дейтаграмм (User Datagram Protocol) |
| ARP | Протокол разрешения адресов (Address Resolution Protocol) |
| RARP | Протокол обратного разрешения адресов (Reverse Address Resolution Protocol) |
| IP | Протокол Internet(Internet Protocol) |
| ICMP | Протокол управляющих сообщений Internet (Internet Control Message Protocol) |
| IGMP | Протокол управления группами Интернета (Internet Group Management Protocol), |
| NDIS | Интерфейс взаимодействия между драйверами транспортных протоколов |
| FTP | Протокол пересылки файлов (File Transfer Protocol) |
| TFTP | Простой протокол пересылки файлов (Trivial File Transfer Protocol) |

**Уровень транспорта**

Уровень транспорта TCP/IP отвечает за установления и поддержания соединения между двумя узлами. Основные функции уровня:

- подтверждение получения информации;
- управление потоком данных;
- упорядочение и ретрансляция пакетов.

В зависимости от типа службы могут быть использованы два протокола:

– TCP (Transmission Control Protocol – протокол управления передачей);
– UDP (User Datagram Protocol – пользовательский протокол дейтаграмм).

TCP обычно используют в тех случаях, когда приложению требуется передать большой объем информации и убедиться, что данные своевременно получены адресатом. Приложения и службы, отправляющие небольшие объемы данных и не нуждающиеся в получении подтверждения, используют протокол UDP, который является протоколом без установления соединения.

*Протокол управления передачей (TCP)*

Протокол TCP отвечает за надежную передачу данных от одного узла сети к другому. Он создает сеанс с установлением соединения, иначе говоря виртуальный канал между машинами. Установление соединения происходит в три этапа:

1. клиент, запрашивающий соединение, отправляет серверу пакет, указывающий номер порта, который клиент желает использовать, а также код (определенное число) ISN (Initial Sequence number).
2. сервер отвечает пакетом, содержащий ISN сервера, а также ISN клиента, увеличенный на 1.
3. клиент должен подтвердить установление соединения, вернув ISN сервера, увеличенный на 1.

Трехступенчатое открытие соединения устанавливает номер порта, а также ISN клиента и сервера. Каждый, отправляемый TCP – пакет содержит номера TCP – портов отправителя и получателя, номер фрагмента для сообщений, разбитых на меньшие части, а также контрольную сумму, позволяющую убедиться, что при передачи не произошло ошибок.

*Пользовательский протокол дейтаграмм (UDP)*

В отличие от TCP UDP не устанавливает соединения. Протокол UDP предназначен для отправки небольших объемов данных без установки соединения и используется приложениями, которые не нуждаются в подтверждении адресатом их получения. UDP также использует номера портов для определения конкретного процесса по указанному IP адресу. Однако UDP порты отличаются от TCP портов и, следовательно, могут использовать те же номера портов, что и TCP, без конфликта между службами.

**Межсетевой уровень**

Межсетевой уровень отвечает за маршрутизацию данных внутри сети и между различными сетями. На этом уровне работают маршрутизаторы, которые зависят от используемого протокола и используются для отправки пакетов из одной сети (или ее сегмента) в другую (или другой сегмент сети). В стеке TCP/IP на этом уровне используется протокол IP.

*Протокол Интернета IP*

Протокол IP обеспечивает обмен дейтаграммами между узлами сети и является протоколом, не устанавливающим соединения и использующим дейтаграммы для отправки данных из одной сети в другую. Данный протокол не ожидает получение подтверждения (ASK, Acknowledgment) отправленных пакетов от узла адресата. Подтверждения, а также повторные отправки пакетов осуществляется протоколами и процессами, работающими на верхних уровнях модели.

К его функциям относится фрагментация дейтаграмм и межсетевая адресация. Протокол IP предоставляет управляющую информацию для сборки фрагментированных дейтаграмм. Главной функцией протокола является межсетевая и глобальная адресация. В зависимости от размера сети, по которой будет маршрутизироваться дейтаграмма или пакет, применяется одна из трех схем адресации.

*Адресация в IP-сетях*

Каждый компьютер в сетях TCP/IP имеет адреса трех уровней: физический (MAC-адрес), сетевой (IP-адрес) и символьный (DNS-имя).

*Физический, или локальный адрес узла,* определяемый технологией, с помощью которой построена сеть, в которую входит узел. Для узлов, входящих в локальные сети - это MAC–адрес сетевого адаптера или порта маршрутизатора, например, 11-A0-17-3D-BC-01. Эти адреса назначаются производителями оборудования и являются уникальными адресами, так как управляются централизовано. Для всех существующих технологий локальных сетей MAC – адрес имеет формат 6 байтов: старшие 3 байта - идентификатор фирмы производителя, а младшие 3 байта назначаются уникальным образом самим производителем.

*Сетевой, или IP-адрес*, состоящий из 4 байт, например, 109.26.17.100. Этот адрес используется на сетевом уровне. Он назначается администратором во время конфигурирования компьютеров и маршрутизаторов. IP-адрес состоит из двух частей: номера сети и номера узла. Номер сети может быть выбран администратором произвольно, либо назначен по рекомендации специального подразделения Internet (Network Information Center, NIC), если сеть должна работать как составная часть Internet. Обычно провайдеры услуг Internet получают диапазоны адресов у подразделений NIC, а затем распределяют их между своими абонентами. Номер узла в протоколе IP назначается независимо от локального адреса узла. Деление IP-адреса на поле номера сети и номера узла - гибкое, и граница между этими полями может устанавливаться произвольно. Узел может входить в несколько IP-сетей. В этом случае узел должен иметь несколько IP-адресов, по числу сетевых связей. IP-адрес характеризует не отдельный компьютер или маршрутизатор, а одно сетевое соединение.

*Символьный адрес, или DNS-имя,* например, SERV1.IBM.COM. Этот адрес назначается администратором и состоит из нескольких частей, например, имени машины, имени организации, имени домена. Такой адрес используется на прикладном уровне, например, в протоколах FTP или telnet.

*Протоколы сопоставления адреса ARP и RARP*

Для определения локального адреса по IP-адресу используется протокол разрешения адреса *Address Resolution Protocol (ARP)*. ARP работает различным образом в зависимости от того, какой протокол канального уровня работает в данной сети – протокол локальной сети (Ethernet, Token Ring, FDDI) с возможностью широковещательного доступа одновременно ко всем узлам сети, или же протокол глобальной сети (X.25, frame relay), как правило, не поддерживающий широковещательный доступ. Существует также протокол, решающий обратную задачу – нахождение IP-адреса по известному локальному адресу. Он называется реверсивный ARP – *RARP (Reverse Address Resolution Protocol)* и используется при старте бездисковых станций, не знающих в начальный момент своего IP-адреса, но знающих адрес своего сетевого адаптера.

В локальных сетях ARP использует широковещательные кадры протокола канального уровня для поиска в сети узла с заданным IP-адресом.

Узел, которому нужно выполнить отображение IP-адреса на локальный адрес, формирует ARP-запрос, вкладывает его в кадр протокола канального уровня, указывая в нем известный IP-адрес, и рассылает запрос широковещательно. Все узлы локальной сети получают ARP-запрос и сравнивают указанный там IP-адрес с собственным адресом. В случае их совпадения узел формирует ARP-ответ, в котором указывает свой IP-адрес и свой локальный адрес и отправляет его уже направленно, так как в ARP-запросе отправитель указывает свой локальный адрес. ARP-запросы и ответы используют один и тот же формат пакета.

*Протокол ICMP*

Протокол управления сообщениями Интернета (ICMP – Internet Control Message Protocol) используется IP и другими протоколами высокого уровня для отправки и получения отчетов о состоянии переданной информации. Этот протокол используется для контроля скорости передачи информации между двумя системами. Если маршрутизатор, соединяющий две системы, перегружен трафиком, он может отправить специальное сообщение ICMP – ошибку для уменьшения скорости отправления сообщений.

*Протокол IGMP*

Узлы локальной сети используют протокол управления группами Интернета (IGMP – Internet Group Management Protocol), чтобы зарегистрировать себя в группе. Информация о группах содержится на маршрутизаторах

локальной сети. Маршрутизаторы используют эту информацию для передачи групповых сообщений.

Групповое сообщение, как и широковещательное, используется для отправки данных сразу нескольким узлам.

*NDIS*

Network Device Interface Specification – спецификация интерфейса сетевого устройства, программный интерфейс, обеспечивающий взаимодействие между драйверами транспортных протоколов, и соответствующими драйверами сетевых интерфейсов. Позволяет использовать несколько протоколов, даже если установлена только одна сетевая карта.

**Уровень сетевого интерфейса**

Этот уровень модели TCP/IP отвечает за распределение IP-дейтаграмм. Он работает с ARP для определения информации, которая должна быть помещена в заголовок каждого кадра. Затем на этом уровне создается кадр, подходящий для используемого типа сети, такого как Ethernet, Token Ring или ATM, затем IP-дейтаграмма помещается в область данных этого кадра, и он отправляется в сеть.

## Вопросы по теме

1. Назначение спецификации стандартов IEEE802.
2. Какой стандарт описывает сетевую технологию Ethernet?
3. Какой стандарт определяет задачи управления логической связью?
4. Какой стандарт задает механизмы управления сетью?
5. Какой стандарт описывает сетевую технологию ArcNet?
6. Какой стандарт описывает сетевую технологию Token Ring?
7. Какой стандарт содержит рекомендации по оптоволоконным сетевым технологиям?
8. Что такое интерфейс уровня базовой модели OSI?
9. Что такое протокол уровня базовой модели OSI?
10. Дать определение стека протоколов.
11. На какие уровни разбиваются стеки протоколов?
12. Назвать наиболее популярные сетевые протоколы.
13. Назвать наиболее популярные транспортные протоколы.
14. Назвать наиболее популярные прикладные протоколы.
15. Перечислить наиболее популярные стеки протоколов.
16. Назначение программных интерфейсов сокетов Windows и NetBIOS.
17. Чем отличается протокол TCP от UDP?
18. Функции протокола IP.

19. Какие существуют виды адресации в IP-сетях?
20. Какой протокол необходим для определения локального адреса по IP-адресу?
21. Какой протокол необходим для определения IP-адреса по локальному адресу?
22. Какой протокол используется для управления сообщениями Inretent?
23. Назначение уровня сетевого интерфейса стека TCP/IP.

# 4 Топология вычислительной сети и методы доступа

## 4.1 Топология вычислительной сети

*Топология (конфигурация)* – это способ соединения компьютеров в сеть. Тип топологии определяет стоимость, защищенность, производительность и надежность эксплуатации рабочих станций, для которых имеет значение время обращения к файловому серверу.

Понятие топологии широко используется при создании сетей. Одним из подходов к классификации топологий ЛВС является выделение двух основных классов топологий*: широковещательные* и *последовательные*.

В *широковещательных топологиях* ПК передает сигналы, которые могут быть восприняты остальными ПК. К таким топологиям относятся топологии: *общая шина, дерево, звезда*.

В *последовательных топологиях* информация передается только одному ПК. Примерами таких топологий являются: *произвольная* (произвольное соединение ПК), *кольцо, цепочка*.

При выборе оптимальной топологии преследуются три основных цели:

- обеспечение альтернативной маршрутизации и максимальной надежности передачи данных;
- выбор оптимального маршрута передачи блоков данных;
- предоставление приемлемого времени ответа и нужной пропускной способности.

При выборе конкретного типа сети важно учитывать ее топологию. Основными сетевыми топологиями являются: шинная (линейная) топология, звездообразная, кольцевая и древовидная.

Например, в конфигурации сети ArcNet используется одновременно и линейная, и звездообразная топология. Сети Token Ring физически выглядят как звезда, но логически их пакеты передаются по кольцу. Передача данных в сети Ethernet происходит по линейной шине, так что все станции видят сигнал одновременно.

## 4.2 Виды топологий

Существуют пять основных топологий:

– общая шина (Bus);
– кольцо (Ring);
– звезда (Star);
– древовидная (Tree);
– ячеистая (Mesh).

**Топология «Общая шина»**

Общая шина это тип сетевой топологии, в которой рабочие станции расположены вдоль одного участка кабеля, называемого сегментом (рис. 4.1).

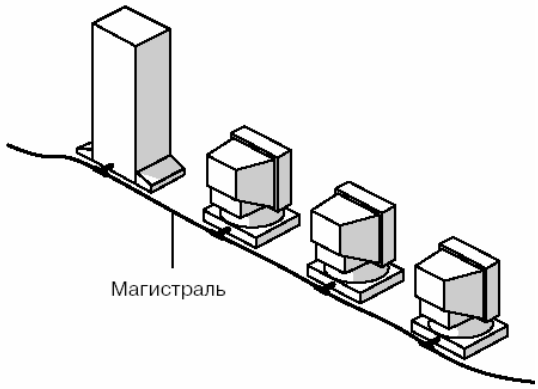


Рис. 4.1 Топология *Общая шина*

В данном случае кабель используется всеми станциями по очереди, поэтому принимаются специальные меры для того, чтобы при работе с общим кабелем компьютеры не мешали друг другу передавать и принимать данные. Все сообщения, посылаемые отдельными компьютерами, принимаются и прослушиваются всеми остальными компьютерами, подключенными к сети. *Рабочая станция* отбирает адресованные ей сообщения, пользуясь *адресной* информацией. Надежность при использовании такой топологии, средняя, поскольку выход из строя отдельных компьютеров не нарушает работоспособности сети в целом. Однако, так как используется только один кабель, в случае обрыва нарушается работа всей сети. Шинная топология – это наиболее простая и наиболее распространенная топология сети.

Примерами использования топологии общая шина является сеть 10Base–5 (соединение ПК толстым коаксиальным кабелем) и 10Base–2 (соединение ПК тонким коаксиальным кабелем).

**Топология «Кольцо»**

*Кольцо* – это топология ЛВС, в которой каждая станция соединена с двумя другими станциями, образуя кольцо (рис.4.2). Данные передаются от одной рабочей станции к другой в одном направлении (по кольцу). Каждый ПК работает как повторитель, ретранслируя сообщения к следующему ПК, т.е. данные, передаются от одного компьютера к другому как бы по эстафете. Если компьютер получает данные, предназначенные для другого компьюте-

ра, он передает их дальше по кольцу, в ином случае они дальше не передаются. Основная проблема при кольцевой топологии заключается в том, что каждая рабочая станция должна активно участвовать в пересылке информации, и в случае выхода из строя хотя бы одной из них, вся сеть парализуется. Топология *Кольцо* имеет хорошо предсказуемое время отклика, определяемое числом рабочих станций.

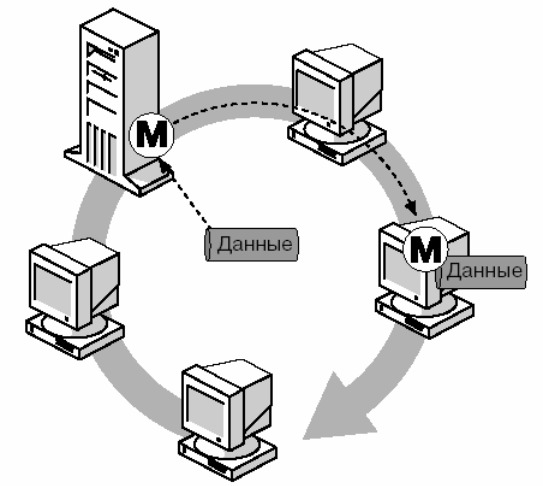


Рис. 4.2. Топология *Кольцо*

Чистая кольцевая топология используется редко. Вместо этого кольцевая топология играет транспортную роль в схеме метода доступа. Кольцо описывает логический маршрут, а пакет передается от одной станции к другой, совершая в итоге полный круг. В сетях Token Ring кабельная ветвь из центрального концентратора называется MAU (Multiple Access Unit). MAU имеет внутреннее кольцо, соединяющее все подключенные к нему станции, и используется как альтернативный путь, когда оборван или отсоединен кабель одной рабочей станции. Когда кабель рабочей станции подсоединен к MAU, он просто образует расширение кольца: сигналы поступают к рабочей станции, а затем возвращаются обратно во внутреннее кольцо

**Топология «Звезда»**

*Звезда* – это топология ЛВС (рис.4.3), в которой все *рабочие станции* присоединены к центральному узлу (например, к концентратору), который устанавливает, поддерживает и разрывает связи между рабочими станциями. Преимуществом такой топологии является возможность простого исключения неисправного *узла*. Однако, если неисправен центральный узел, вся сеть выходит из строя.

В этом случае каждый компьютер через специальный сетевой адаптер подключается отдельным кабелем к объединяющему устройству. При необходимости можно объединять вместе несколько сетей с топологией *Звезда,* при этом получаются разветвленные конфигурации сети. В каждой точке ветвления необходимо использовать специальные соединители (распределители, повторители или устройства доступа).

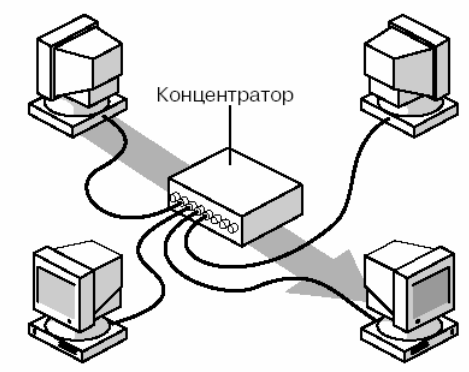


Рис. 4.3. Топология *Звезда*

Примером звездообразной топологии является топология Ethernet с кабелем типа *Витая пара* 10BASE-T, центром *Звезды* обычно является Hub.

Звездообразная топология обеспечивает защиту от разрыва кабеля. Если кабель рабочей станции будет поврежден, это не приведет к выходу из строя всего сегмента сети. Она позволяет также легко диагностировать проблемы подключения, так как каждая рабочая станция имеет свой собственный кабельный сегмент, подключенный к концентратору. Для диагностики достаточно найти разрыв кабеля, который ведет к неработающей станции. Остальная часть сети продолжает нормально работать.

Звездообразная топология имеет и недостатки: во-первых, она не экономична с точки зрения использования кабеля, во-вторых, концентраторы довольно дороги и, в-третьих, кабельные концентраторы при большом количестве кабеля трудно обслуживать. Однако в большинстве случаев в такой топологии используется недорогой кабель типа *витая пара*. В некоторых случаях можно даже использовать существующие телефонные кабели.

**Древовидная топология**

*Древовидная* топология (рис. 4.4.) – достигается из звездообразной путем каскадирования концентраторов. Такая топология широко используется в современных высокоскоростных локальных компьютерных сетях. В качестве узлов коммутации чаще всего выступают высокоскоростные коммутаторы. Наиболее характерным представителем сетей с подобной структурой является сеть 100VG AnyLan. И кроме того, высокоскоростной вариант магистральной сети Ethernet – Fast Ethernet также имеет древовидную структуру.

По сравнению с шинными и кольцевыми сетями древовидные локальные сети обладают более высокой надежностью. Отключение или выход из строя одной из линий или коммутатора, как правило, не оказывает существенного влияния на работоспособность оставшейся части локальной сети.

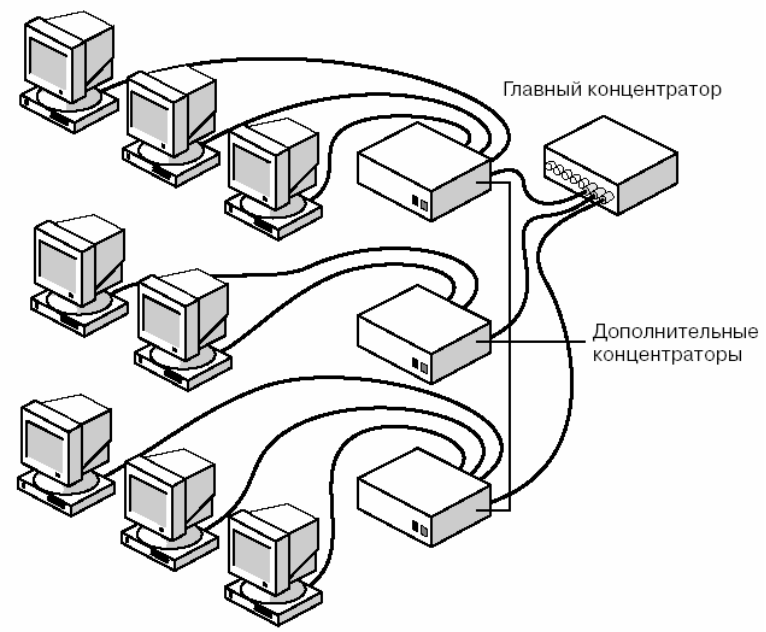


Рис. 4.4. *Древовидная* топология

**Ячеистая топология**

*Ячеистая* топология (рис. 4.5.) – это топология, при которой все *рабочие станции* соединены со всеми (полносвязанная топология). Ячеистая топология нашла применение в последние несколько лет. Ее привлекательность заключается в относительной устойчивости к перегрузкам и отказам. Благодаря множественности путей из устройств, включенных в сеть, трафик может быть направлен в обход отказавших или занятых узлов. Даже несмотря на то, что данный подход отмечается сложностью и дороговизной (протоколы ячеистых сетей могут быть достаточно сложными с точки зрения логики, чтобы обеспечить эти характеристики), некоторые пользователи предпочитают ячеистые сети сетям других типов вследствие их высокой надежности.

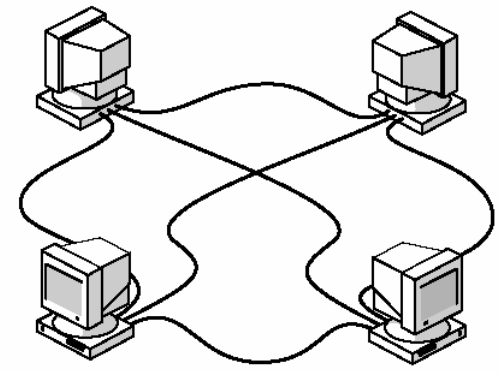

Рис. 4.5. *Ячеистая* топология

## 4.3 Методы доступа

Метод доступа – это способ определения того, какая из рабочих станций сможет следующей использовать ЛВС. То, как сеть управляет доступом к ка-

налу связи (кабелю), существенно влияет на ее характеристики. Примерами методов доступа являются:

- множественный доступ с прослушиванием несущей и разрешением коллизий (Carrier Sense Multiple Access with Collision Detection – CSMA/CD);
- множественный доступ с передачей полномочия (Token Passing Multiple Access – TPMA) или метод с передачей маркера;
- множественный доступ с разделением во времени (Time Division Multiple Access – TDMA);
- множественный доступ с разделением частоты (Frequency Division Multiple Access – FDMA) или множественный доступ с разделением длины волны (Wavelength Division Multiple Access – WDMA).

**CSMA/CD**

Алгоритм множественного доступа с прослушиванием несущей и разрешением коллизий приведен на рис. 4.6.

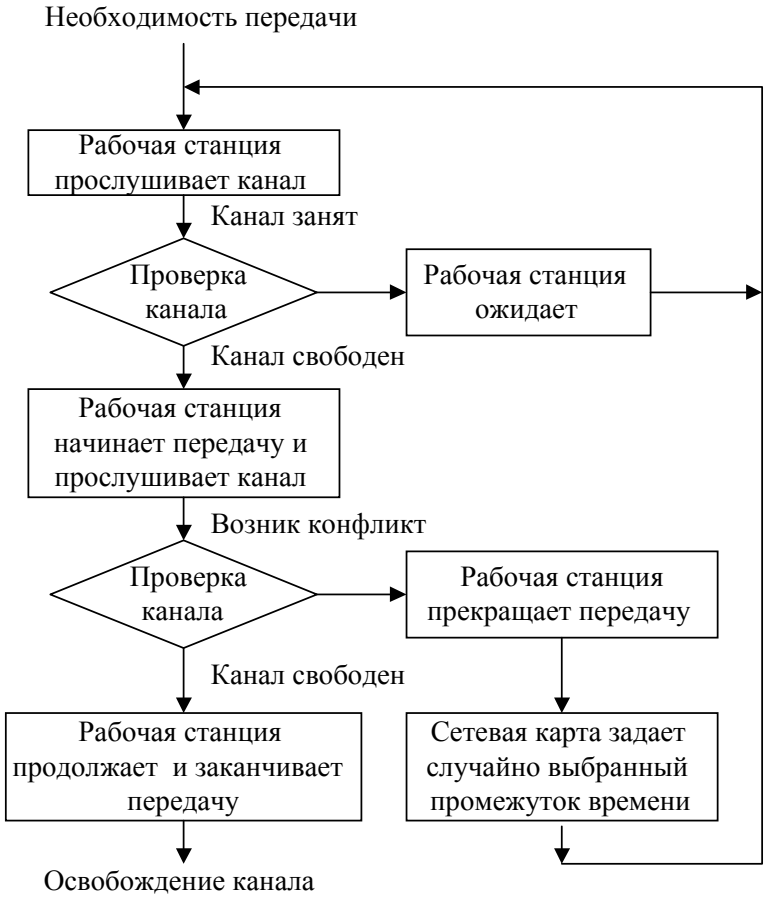


Рис. 4.6. Алгоритм *CSMA/CD*

*Метод множественного доступа с прослушиванием несущей и разрешением коллизий (CSMA/CD)* устанавливает следующий порядок: если рабочая станция хочет воспользоваться сетью для передачи данных, она сначала должна проверить состояние канала (начинать передачу станция может, если

канал свободен). В процессе передачи станция продолжает прослушивание сети для обнаружения возможных конфликтов. Если возникает конфликт из-за того, что два узла попытаются занять канал, то обнаружившая конфликт интерфейсная плата, выдает в сеть специальный сигнал, и обе станции одновременно прекращают передачу. Принимающая станция отбрасывает частично принятое сообщение, а все рабочие станции, желающие передать сообщение, в течение некоторого, случайно выбранного промежутка времени выжидают, прежде чем начать сообщение.

Все сетевые интерфейсные платы запрограммированы на разные псевдослучайные промежутки времени. Если конфликт возникнет во время повторной передачи сообщения, этот промежуток времени будет увеличен. Стандарт типа *Ethernet* определяет сеть с конкуренцией, в которой несколько рабочих станций должны конкурировать друг с другом за право доступа к сети.

### ТРМА

Алгоритм множественного доступа с передачей полномочия, или маркера, приведен на рис. 4.7.


Рис.4.7. Алгоритм ТРМА

*Метод с передачей маркера* – это метод доступа к среде, в котором от рабочей станции к рабочей станции передается маркер, дающий разрешение

на передачу сообщения. При получении маркера рабочая станция может передавать сообщение, присоединяя его к маркеру, который переносит это сообщение по сети. Каждая станция между передающей станцией и принимающей видит это сообщение, но только станция – адресат принимает его. При этом она создает новый маркер.

Маркер (token), или полномочие, – уникальная комбинация битов, позволяющая начать передачу данных.

Каждый узел принимает пакет от предыдущего, восстанавливает уровни сигналов до номинального уровня и передает дальше. Передаваемый пакет может содержать данные или являться маркером. Когда рабочей станции необходимо передать пакет, ее адаптер дожидается поступления маркера, а затем преобразует его в пакет, содержащий данные, отформатированные по протоколу соответствующего уровня, и передает результат далее по *ЛВС*.

Пакет распространяется по *ЛВС* от адаптера к адаптеру, пока не найдет своего адресата, который установит в нем определенные биты для подтверждения того, что данные достигли адресата, и ретранслирует его вновь в *ЛВС*. После чего пакет возвращается в узел из которого был отправлен. Здесь после проверки безошибочной передачи пакета, узел освобождает *ЛВС*, выпуская новый маркер. Таким образом, в *ЛВС* с передачей маркера невозможны коллизии (конфликты). Метод с передачей маркера в основном используется в кольцевой топологии.

Данный метод характеризуется следующими достоинствами:

- гарантирует определенное время доставки блоков данных в сети;
- дает возможность предоставления различных приоритетов передачи данных.

Вместе с тем он имеет существенные недостатки:

- в сети возможны потеря маркера, а также появление нескольких маркеров, при этом сеть прекращает работу;
- включение новой рабочей станции и отключение связаны с изменением адресов всей системы.

**TDMA**

Множественный доступ с разделением во времени основан на распределении времени работы канала между системами (рис.4.8).

Доступ *TDMA* основан на использовании специального устройства, называемого тактовым генератором. Этот генератор делит время канала на повторяющиеся циклы. Каждый из циклов начинается сигналом *Разграничителем*. Цикл включает *n* пронумерованных временных интервалов, называемых ячейками. Интервалы предоставляются для загрузки в них блоков данных.

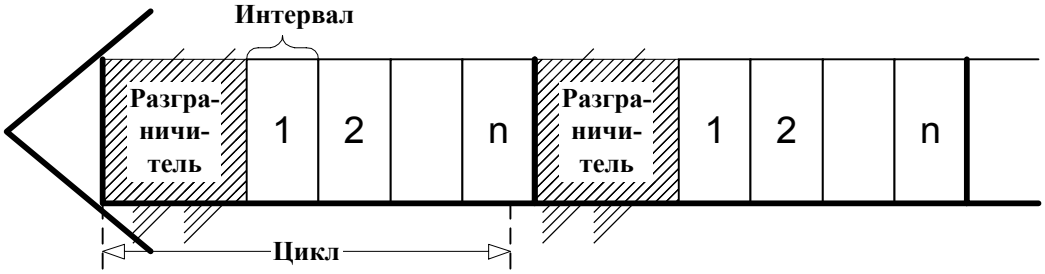


Рис.4.8. Структура множественного доступа с разделением во времени

Данный способ позволяет организовать передачу данных с коммутацией пакетов и с коммутацией каналов.

Первый (простейший) вариант использования интервалов заключается в том, что их число ($n$) делается равным количеству абонентских систем, подключенных к рассматриваемому каналу. Тогда во время цикла каждой системе предоставляется один интервал, в течение которого она может передавать данные. При использовании рассмотренного метода доступа часто оказывается, что в одном и том же цикле одним системам нечего передавать, а другим не хватает выделенного времени. В результате – неэффективное использование пропускной способности канала.

Второй, более сложный, но высокоэкономичный вариант заключается в том, что система получает интервал только тогда, когда у нее возникает необходимость в передаче данных, например при асинхронном способе передачи. Для передачи данных система может в каждом цикле получать интервал с одним и тем же номером. В этом случае передаваемые системой блоки данных появляются через одинаковые промежутки времени и приходят с одним и тем же временем запаздывания. Это режим передачи данных с имитацией коммутации каналов. Способ особенно удобен при передаче речи.

**FDMA**

Доступ *FDMA* основан на разделении полосы пропускания канала на группу полос частот (Рис. 4.9), образующих *логические каналы*.

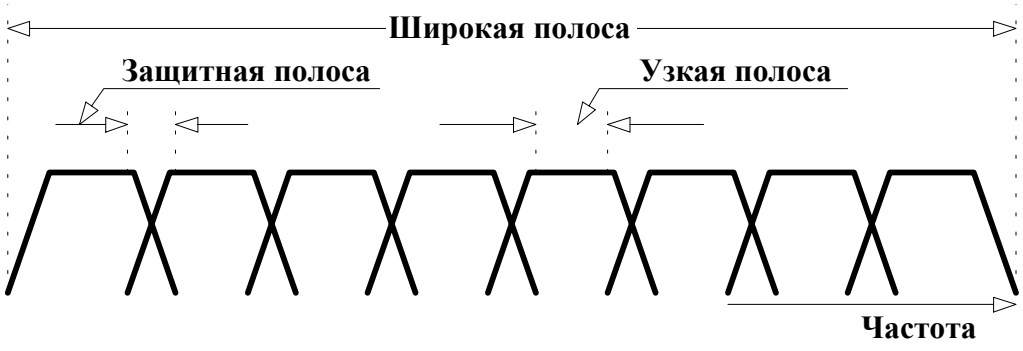


Рис. 4.9. Схема выделения логических каналов

Широкая полоса пропускания канала делится на ряд узких полос, разделенных защитными полосами. Размеры узких полос могут быть различными. В каждой узкой полосе создается логический канал. Передаваемые по логи-

ческим каналам сигналы накладываются на разные несущие и поэтому в частотной области не должны пересекаться. Вместе с этим, иногда, несмотря на наличие защитных полос, спектральные составляющие сигнала могут выходить за границы логического канала и вызывать шум в соседнем логическом канале.

В оптических каналах разделение частоты осуществляется направлением в каждый из них лучей света с различными частотами. Благодаря этому пропускная способность физического канала увеличивается в несколько раз. При осуществлении этого мультиплексирования в один световод излучает свет большое число лазеров (на различных частотах). Через световод излучение каждого из них проходит независимо от другого. На приемном конце разделение частот сигналов, прошедших физический канал, осуществляется путем фильтрации выходных сигналов.

Метод доступа FDMA относительно прост, но для его реализации необходимы передатчики и приемники, работающие на различных частотах.

### Вопросы по теме

1. Что такое топология?
2. Перечислить наиболее используемые типы топологий?
3. Охарактеризовать топологию *Общая шина* и привести примеры использования данной топологии.
4. Какие сетевые технологии используют топологию *Общая шина*?
5. Охарактеризовать топологию *Кольцо* и привести примеры этой топологии.
6. В каких случаях используют топологию *Кольцо*?
7. Охарактеризовать топологию *Звезда* и привести примеры использования этой топологии.
8. К какой топологии относится сеть при подсоединении всех компьютеров к общему концентратору?
9. Привести примеры и охарактеризовать древовидную топологию.
10. Что такое ячеистая топология и в каких случаях она используется?
11. Что такое метод доступа и как влияет метод доступа на передачу данных в сети?
12. Какие существуют методы доступа?
13. Охарактеризовать метод доступа с прослушиванием несущей и разрешением коллизий.
14. При каком методе доступа обе станции могут одновременно начать передачу и войти в конфликт?
15. В каких сетевых технологиях используется метод *CSMA/CD*?
16. Охарактеризовать метод доступа с разделением во времени и перечислить в каких случаях используется данный метод.

17. Что такое маркер?
18. В каком случае рабочая станция может начать передачу данных при использовании метода доступа с передачей полномочия?
19. Охарактеризовать метод доступа с передачей полномочия.
20. Охарактеризовать метод множественного доступа с разделением частоты.
21. Какие существуют варианты использования множественного доступа с разделением во времени?

# 5 ЛВС и компоненты ЛВС

## 5.1 Локальная вычислительная сеть

ЛВС (локальная вычислительная сеть) – это совокупность компьютеров, каналов связи, сетевых адаптеров, работающих под управлением сетевой операционной системы и сетевого программного обеспечения. Основной особенностью ЛВС является низкая территориальная распределенность ЭВМ (в пределах знания, предприятия и т.д.). Чаще всего ЛВС являются элементами более крупномасштабных образований.

В *ЛВС* каждый *ПК* называется *рабочей станцией*, за исключением одного или нескольких компьютеров, которые предназначены для выполнения функций *сервера.* Каждая *рабочая станция* и *сервер* оснащены *сетевыми картами (адаптерами)*, которые посредством *физических каналов* соединяются между собой. В дополнение к локальной ОС на каждой рабочей станции активизируется сетевое ПО (сетевые службы), позволяющее организовывать ее взаимодействие с другими станциями и сервером. Аналогичным образом, на сервере активизируется сетевое ПО, позволяющее ему взаимодействовать с рабочими станциями и другими серверами.

## 5.2 Основные компоненты вычислительной сети

В компьютерной сети можно выделить три основных аппаратных компонента и два программных. Для корректной и согласованной работы устройств в сети они должны быть правильно инсталлированы и настроены. Основными аппаратными компонентами сети являются:

1. абонентские системы: компьютеры (рабочие станции или клиенты и серверы); принтеры; сканеры и др.
2. сетевое оборудование: сетевые адаптеры; концентраторы (хабы); мосты; маршрутизаторы и др.
3. коммуникационные каналы: кабели; разъемы; устройства передачи и приема данных в беспроводных технологиях.

Основными программными компонентами сети являются:

1. сетевые ОС, например, Microsoft Windows NT; Novell NetWare; Unix; Linux и т.д.
2. сетевое ПО (Сетевые службы): клиент сети; протокол; служба удаленного доступа, драйвер сетевого адаптера и др.

**Рабочая станция**

Рабочая станция (workstation) – это абонентская система, специализированная для решения определенных задач пользователя и использующая сетевые ресурсы. К сетевому программному обеспечению рабочей станции относятся следующие службы: клиент для сетей; служба доступа к файлам и принтерам; сетевые протоколы для данного типа сетей; драйвер сетевого адаптера; контроллер удаленного доступа.

В отличие от автономного ПК рабочая станция:

– оснащается сетевым адаптером и каналом связи;
– перед началом работы на рабочей станции необходимо выполнить процедуру входа в сеть;
– после подключения рабочей станции к ЛВС появляются дополнительные сетевые дисковые накопители и появляется возможность использования удаленного оборудования.

**Сервер**

Сервер – это компьютер, предоставляющий свои ресурсы (диски, принтеры, каталоги, файлы и т.п.) другим пользователям сети. Кроме своей первичной функции (предоставление ресурсов), сервер может выполнять ряд дополнительных функций: функции маршрутизации, аутентификации и контроля доступа пользователей и т.д.

По мере усложнения возлагаемых на серверы функций и увеличения числа обслуживаемых ими клиентов происходит все большая специализация серверов. Существует множество типов серверов:

– первичный контроллер домена, сервер, на котором хранится база бюджетов пользователей и поддерживается политика защиты;
– вторичный контроллер домена, сервер, на котором хранится резервная копия базы бюджетов пользователей и политики защиты;
– универсальный сервер, предназначенный для выполнения несложного набора различных задач обработки данных в локальной сети;
– сервер базы данных, выполняющий обработку запросов, направляемых базе данных;
– proxy-сервер, необходимый для организации доступа пользователей ЛВС в Internet;
– web-сервер, предназначенный для предоставления гипертекстовой информации;

– почтовый сервер, предоставляющий сервис электронной почты и т.д.

**Сетевое оборудование**

Для подключения рабочей станции к информационной сети требуется устройство сопряжения, которое называют сетевым адаптером (модулем или картой). Сетевой адаптер устанавливается в PCI- или ISA-разъем материнской платы. Современный материнские могут оснащаться встроенным сетевым адаптером.

Сетевые адаптеры вместе с сетевым программным обеспечением способны распознавать и обрабатывать ошибки, которые могут возникнуть из-за электрических помех, коллизий или плохой работы оборудования.

Различные типы сетевых адаптеров отличаются не только методами доступа к каналу связи и протоколами, но еще и следующими параметрами (на все эти параметры следует обращать внимание при выборе адаптеров):

– скорость передачи;
– объем буфера для пакета;
– тип шины;
– быстродействие шины;
– совместимость с различными микропроцессорами;
– использованием прямого доступа к памяти (DMA);
– адресация портов ввода/вывода и запросов прерывания;
– конструкция разъема.

Если для построения локальных связей между компьютерами используются различные виды кабельных систем, сетевые адаптеры, концентраторы и повторители, то для связей между сегментами ЛВС используются концентраторы, мосты, коммутаторы, маршрутизаторы и шлюзы, а для подключения ЛВС к глобальным сетям могут используются:

– специальные выходы (WAN–порты) мостов и маршрутизаторов;
– аппаратура передачи данных по длинным линиям – модемы;
– устройства подключения к цифровым каналам (ТА – терминальные адаптеры сетей ISDN, устройства обслуживания цифровых выделенных каналов типа CSU/DSU и т.п.).

На рис. 5.1 приведен фрагмент вычислительной сети.

**Сетевая операционная система**

Сетевая операционная система **(**Network Operating System – NOS) – это операционная система со встроенными или надстроенными сетевыми функциями, обеспечивающая доступ рабочей станции в информационную сеть.

Сетевая операционная система необходима для управления потоками сообщений между рабочими станциями и серверами. Она является приклад-

ной платформой, предоставляет разнообразные виды сетевых служб и поддерживает работу прикладных процессов, реализуемых в сетях.

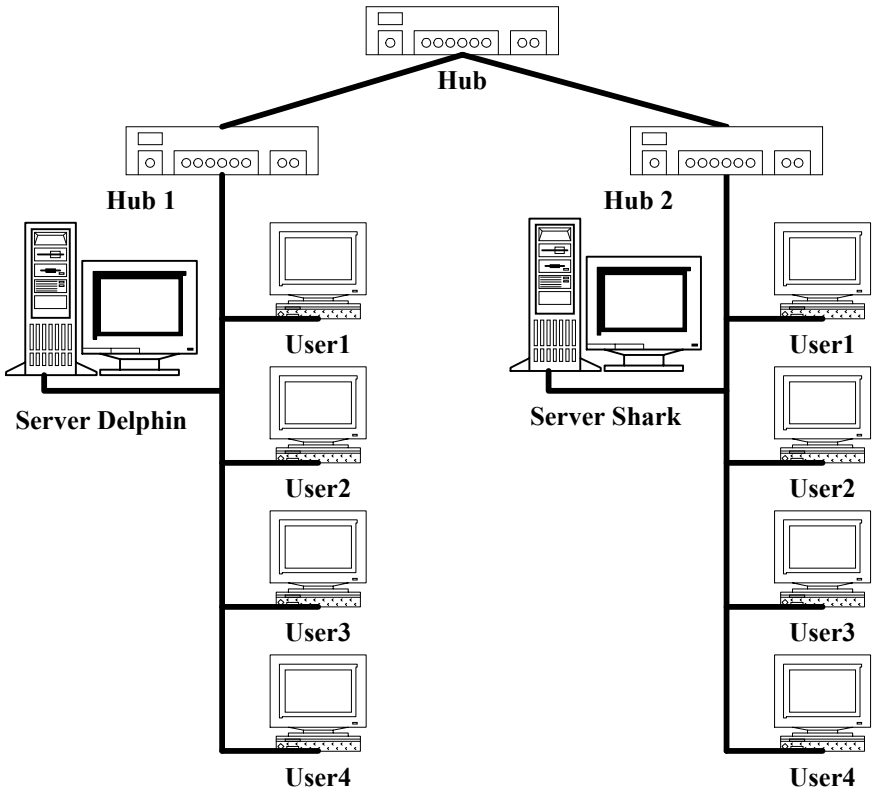


Рис. 5.1 Фрагмент сети

Сетевая ОС определяет группу протоколов, обеспечивающих основные функции сети. К ним относятся:

- адресация объектов сети;
- функционирование сетевых служб;
- обеспечение безопасности данных;
- управление сетью.

### Сетевое программное обеспечение

*Клиент для сетей* обеспечивает связь с другими компьютерами выбранного вида сети, а также доступ к файлам и принтерам.

*Драйвер сетевой карты* является программной, организующей связь ОС с сетевым адаптером.

*Протоколы* используются для установления правил обмена информацией в сетях.

*Служба удаленного доступа* позволяет делать файлы и принтеры доступными для компьютеров в сети.

## Вопросы по теме

1. Перечислить основные компоненты сети.

2. Дать определение рабочей станции.
3. Чем отличается рабочая станция в сети от локального компьютера?
4. Что такое сервер?
5. Какое назначение первичного контролера домена в сети?
6. Для чего используется вторичный контролера домена?
7. Что такое proxy–сервер?
8. Какая информация хранится на сервере баз данных?
9. Может ли сервер баз данных и web–сервер размещаться на одном компьютере?
10. Перечислить сетевое программное обеспечение рабочей станции.
11. Каково назначение сетевой ОС?
12. Перечислить наиболее известные сетевые операционные системы.
13. Чем различаются типы сетевых адаптеров?
14. Что такое сетевая операционная система?
15. Перечислить сетевое программное обеспечение и его назначение.
16. Для какой цели используется web–сервер?
17. Какое сетевое оборудование используется для связи между сегментами ЛВС?

# 6 Физическая среда передачи данных

Физическая среда является основой, на которой строятся физические средства соединения. Сопряжение с *физическими средствами соединения* посредством физической среды обеспечивает *Физический уровень*. В качестве физической среды широко используются эфир, металлы, оптическое стекло и кварц. На физическом уровне находится носитель, по которому передаются данные. Среда передачи данных может включать как кабельные, так и беспроводные технологии. Хотя физические кабели являются наиболее распространенными носителями для сетевых коммуникаций, беспроводные технологии все более внедряются благодаря их способности связывать глобальные сети.

На физическом уровне для физических кабелей определяются механические и электрические (оптические) свойства среды передачи, которые включают:

– тип кабелей и разъемов;
– разводку контактов в разъемах;
– схему кодирования сигналов для значений 0 и 1.

Канальный уровень определяет доступ к среде и управление передачей посредством процедуры передачи данных по каналу. В локальных сетях протоколы канального уровня используются компьютерами, мостами, коммута-

торами и маршрутизаторами. В компьютерах функции канального уровня реализуются совместными усилиями сетевых адаптеров и их драйверов.

## 6.1 Кабели связи, линии связи, каналы связи

Для организации связи в сетях используются следующие понятия:

– кабели связи;
– линии связи;
– каналы связи.

*Кабель связи* — это длинномерное изделие электротехнической промышленности. Из кабелей связи и других элементов (монтаж, крепеж, кожухи и т.д.) строят *линии связи*. Длина линий связи колеблется от десятков метров до десятков тысяч километров..

По уже построенным линиям организуют *каналы связи*. Для удешевления каналов связи применяют аппаратуру каналообразования (уплотнения линии), что позволяет по каждой электрической цепи (два провода) обеспечивать связь не одной паре абонентов.

## 6.2 Типы кабелей и структурированные кабельные системы

В качестве среды передачи данных используются различные виды кабелей: коаксиальный кабель, кабель на основе экранированной и неэкранированной витой пары и оптоволоконный кабель. Наиболее популярным видом среды передачи данных на небольшие расстояния (до 100 м) является *неэкранированная витая пара*, которая включена практически во все современные стандарты и технологии локальных сетей и обеспечивает пропускную способность до 100 Мб/с (на кабелях категории 5).

В качестве среды передачи данных в вычислительных сетях используются также электромагнитные волны различных частот – КВ, УКВ, СВЧ. Однако пока в локальных сетях радиосвязь используется только в тех случаях, когда оказывается невозможной прокладка кабеля. Это объясняется недостаточной надежностью сетевых технологий, построенных на использовании электромагнитного излучения. Для построения глобальных каналов этот вид среды передачи данных используется шире – на нем построены спутниковые каналы связи и наземные радиорелейные каналы, работающие в зонах прямой видимости в СВЧ диапазонах.

Очень важно правильно построить фундамент сети – кабельную систему. В последнее время в качестве такой надежной основы все чаще используется *структурированная кабельная система*.

*Структурированная кабельная система* (Structured Cabling System – SCS) – это набор коммутационных элементов (кабелей, разъемов, коннекторов, кроссовых панелей и шкафов), а также методика их совместного использования, которая позволяет создавать регулярные, легко расширяемые структуры связей в вычислительных сетях.

Преимущества структурированной кабельной системы:

- *универсальность* – структурированная кабельная система при продуманной организации может стать единой средой для передачи компьютерных данных в ЛВС;
- *увеличение срока службы* – срок старения хорошо структурированной кабельной системы может составлять 8-10 лет;
- *уменьшение стоимости добавления новых пользователей и изменения их мест размещения* – стоимость кабельной системы в основном определяется не стоимостью кабеля, а стоимостью работ по его прокладке;
- *возможность легкого расширения сети* – структурированная кабельная система является модульной, поэтому ее легко наращивать;
- *обеспечение более эффективного обслуживания* – структурированная кабельная система облегчает обслуживание и поиск неисправностей;
- *надежность* – структурированная кабельная система имеет повышенную надежность, поскольку обычно производство всех ее компонентов и техническое сопровождение осуществляется одной фирмой-производителем.

## 6.3 Кабельные системы

Выделяют два больших класса кабелей: электрические и оптические, которые принципиально различаются по способу передачи по ним сигнала.

Отличительная особенность оптоволоконных систем – высокая стоимость как самого кабеля (по сравнению с медным), так и специализированных установочных элементов (розеток, разъемов, соединителей и т. п.). Правда, главный вклад в стоимость сети вносит цена активного сетевого оборудования для оптоволоконных сетей.

Оптоволоконные сети применяются высокоскоростных, ответственных каналов.

Оптоволоконные кабели в будущем смогут составить реальную конкуренцию медным высокочастотным, поскольку стоимость производства медных кабелей снижаться не будет, ведь для него нужна очень чистая медь, запасов которой на земле гораздо меньше, чем кварцевого песка, из которого производят оптоволокно.

Основные поставщики оптоволоконного кабеля для России – Mohawk/CDT, Lucent Technologies и AMP.

## 6.4 Типы кабелей

Существует несколько различных типов кабелей, используемых в современных сетях. Ниже приведены наиболее часто используемые типы кабелей. Множество разновидностей медных кабелей составляют класс электрических кабелей, используемых как для прокладки телефонных сетей, так и

для инсталляции ЛВС. По внутреннему строению различают кабели на витой паре и коаксиальные кабели.

**Кабель типа «витая пара» (twisted pair)**

*Витой парой* называется кабель, в котором изолированная пара проводников скручена с небольшим числом витков на единицу длины. Скручивание проводов уменьшает электрические помехи извне при распространении сигналов по кабелю, а *экранированные витые пары* еще более увеличивают степень помехозащищенности сигналов.

Кабель типа «витая пара» используется во многих сетевых технологиях, включая Ethernet, ARCNet и IBM Token Ring.

Кабели на витой паре подразделяются на: неэкранированные (UTP – Unshielded Twisted Pair) и экранированные медные. Последние подразделяются на две разновидности: с экранированием каждой пары и общим экраном (STP – Shielded Twisted Pair) и с одним только общим экраном (FTP – Foiled Twisted Pair). Наличие или отсутствие экрана у кабеля вовсе не означает наличия или отсутствия защиты передаваемых данных, а говорит лишь о различных подходах к подавлению помех. Отсутствие экрана делает неэкранированные кабели более гибкими и устойчивыми к изломам. Кроме того, они не требуют дорогостоящего контура заземления для эксплуатации в нормальном режиме, как экранированные. Неэкранированные кабели идеально подходят для прокладки в помещениях внутри офисов, а экранированные лучше использовать для установки в местах с особыми условиями эксплуатации, например, рядом с очень сильными источниками электромагнитных излучений, которых в офисах обычно нет.

Кабели классифицируются по категории, указанным в таблице 6.1. Основанием для отнесения кабеля к одной из категорий служит максимальная частота передаваемого по нему сигнала.

Таблица 6.1. Категории витой пары

| **Категория** | **Частота передаваемого сигнала, (МГц)** |
|---|---|
| 3 | 16 |
| 4 | 20 |
| 5 | 100 |
| 5+ | 300 |
| 6 | 200 |
| 7 | 600 |

**Коаксиальные кабели**

*Коаксиальные кабели* используются в радио и телевизионной аппаратуре. *Коаксиальные кабели* могут передавать данные со скоростью 10 Мбит/с на максимальное расстояние от 185 до 500 метров. Они разделяются на *тол-*

*стые* и *тонкие* в зависимости от толщины. Типы коаксиальных кабелей приведены в таблице 6.2.

Таблица 6.2. Типы коаксиальных кабелей.

| Тип | Название, значение сопротивления |
|---|---|
| RG-8 и RG-11 | Thicknet, 50 Ом |
| RG-58/U | Thinnet, 50 Ом, сплошной центральный медный проводник |
| RG-58 A/U | Thinnet, 50 Ом, центральный многожильный проводник |
| RG-59 | Broadband/Cable television (широковещательное и кабельное телевидение), 75 Ом |
| RG-59 /U | Broadband/Cable television (широковещательное и кабельное телевидение), 50 Ом |
| RG-62 | ARCNet, 93 Ом |

Кабель Thinnet, известный как кабель RG-58, является наиболее широко используемым физическим носителем данных. Сети при этом не требуют дополнительного оборудования и являются простыми и недорогими. Хотя *тонкий коаксиальный кабель (Thin Ethernet)* позволяет передачу на меньшее расстояние, чем толстый, но для соединений *с тонким кабелем* применяются стандартные байонетные разъемы *BNC* типа *CP-50* и ввиду его небольшой стоимости он становится фактически стандартным для офисных ЛВС. Используется в технологии *Ethernet 10Base2*.

*Толстый коаксиальный кабель (Thick Ethernet)* имеет большую степень помехозащищенности, большую механическую прочность, но требует специального приспособления для прокалывания кабеля, чтобы создать ответвления для подключения к ЛВС. Он более дорогой и менее гибкий, чем тонкий. Используется в технологии *Ethernet 10Base5*. Сети ARCNet с посылкой маркера обычно используют кабель RG-62 A/U.

**Оптоволоконный кабель**

Оптоволоконный кабель (Fiber Optic Cable) обеспечивает высокую скорость передачи данных на большом расстоянии, но он дорог, и с ним трудно работать Они также невосприимчивы к интерференции и подслушиванию. В *оптоволоконном кабеле* для передачи сигналов используется свет.

Для установки разъемов, создания ответвлений, поиска неисправностей в *оптоволоконном кабеле* необходимы специальные приспособления и высокая квалификация. *Оптоволоконный кабель* состоит из центральной стеклянной нити толщиной в несколько микрон, покрытой сплошной стеклянной оболочкой. Все это, в свою очередь, скрыто во внешнюю защитную оболочку.

Оптоволоконные линии очень чувствительны к плохим соединениям в разъемах. В качестве источника света в таких кабелях применяются *светодиоды* (*LED - Light Emitting Diode*), а информация кодируется путем измене-

ния интенсивности света. На приемном конце кабеля детектор преобразует световые импульсы в электрические сигналы.

## 6.5 Кабельные системы Ethernet

### 10Base-T, 100Base-TX

Неэкранированная витая пара (Unshielded Twisted Pair – UTP) – это кабель из скрученных пар проводов.

Характеристики кабеля:

- диаметр проводников 0.4 – 0.6 мм (22~26 AWG), 4 скрученных пары (8 проводников, из которых для 10Base-T и 100Base-TX используются только 4). Кабель должен иметь категорию 3 или 5 и качество data grade или выше;
- максимальная длина сегмента 100 м;
- разъемы (модульные соединители) восьми контактные RJ-45.

Использование контактов модульных соединителей, а также цветовая маркировка проводов стандартизованы. Каждая пара представляется двумя проводами, обозначаемыми Tip и Ring (условно – прямой и обратный провода), для которых определен цвет изоляции и номер контакта разъема. Существует несколько стандартов на модульные соединители, различающихся шириной, количеством используемых контактов и раскладкой пар проводов (см. рис. 6.1. и табл. 6.3).

При монтаже структурированной кабельной системы для передачи данных следует использовать раскладку EIA/TIA-568A (сокращенно T568A) или EIA/TIA-568B (сокращенно T568B). Раскладка T568B известна и под именем AT&T258A или WECO.

Таблица 6.3. Раскладка проводов RJ-45 (вид на розетку).

| № | Раскладка T568A | | Раскладка T568B | |
|---|---|---|---|---|
| | Цвет основной (полоски) | Пара | Цвет основной (полоски) | Пара |
| 1 | Белый (зеленый) | 3 (Tip) | Белый (оранжевый) | 3 (Tip) |
| 2 | Зеленый | 3 (Ring) | Оранжевый | 3 (Ring) |
| 3 | Белый (оранжевый) | 2 (Tip) | Белый (зеленый) | 2 (Tip) |
| 4 | Синий | 1 (Ring) | Синий | 1 (Ring) |
| 5 | Белый (синий) | 1 (Tip) | Белый (синий) | 1 (Tip) |
| 6 | Оранжевый | 2 (Ring) | Зеленый | 2 (Ring) |
| 7 | Белый (коричневый) | 4 (Tip) | Белый (коричневый) | 4 (Tip) |
| 8 | Коричневый | 4 (Ring) | Коричневый | 4 (Ring) |

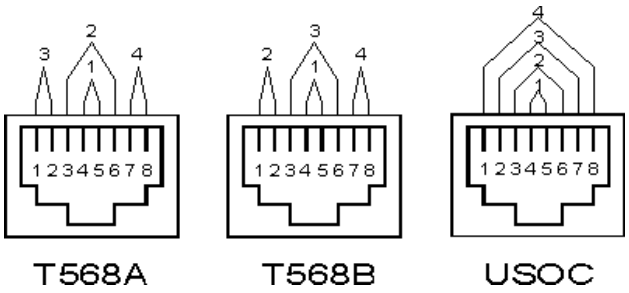


Рис. 6.1. Восьми контактные RJ-45

**10Base2**

Тонкий коаксиальный кабель, характеристики кабеля:

- диаметр 0.2 дюйма, RG-58A/U 50 Ом;
- приемлемые разъемы – BNC;
- максимальная длина сегмента – 185 м;
- минимальное расстояние между узлами – 0.5 м;
- максимальное число узлов в сегменте – 30.

**10Base5**

Толстый коаксиальный кабель, характеристики кабеля:

- волновое сопротивление – 50 Ом;
- максимальная длина сегмента – 500 метров;
- минимальное расстояние между узлами –: 2.5 м;
- максимальное число узлов в сегменте – 100.

## 6.6 Беспроводные технологии

Методы беспроводной технологии передачи данных (Radio Waves) являются удобным, а иногда незаменимым средством связи. Беспроводные технологии различаются по типам сигнала, частоте (большая частота означает большую скорость передачи) и расстоянию передачи. Большое значение имеют помехи и стоимость. Можно выделить три основных типа беспроводной технологии:

- радиосвязь;
- связь в микроволновом диапазоне;
- инфракрасная связь.

**Радиосвязь**

Технологии радиосвязи пересылают данные на радиочастотах и практически не имеют ограничений по дальности. Она используется для соединения локальных сетей на больших географических расстояниях. Радиопередача в целом имеет высокую стоимость и чувствительна к электронному и атмо-

сферному наложению, а также подвержена перехватам, поэтому требует шифрования для обеспечения уровня безопасности.

### Связь в микроволновом диапазоне

Передача данных в микроволновом диапазоне (Microwaves) использует высокие частоты и применяется как на коротких, так и на больших расстояниях. Главное ограничение заключается в том, чтобы передатчик и приемник были в зоне прямой видимости. Используется в местах, где использование физического носителя затруднено. Передача данных в микроволновом диапазоне при использовании спутников может быть очень дорогой.

### Инфракрасная связь

Инфракрасные технологии (Infrared transmission), функционируют на очень высоких частотах, приближающихся к частотам видимого света. Они могут быть использованы для установления двусторонней или широковещательной передачи на близких расстояниях. При инфракрасной связи обычно используют светодиоды (LED – *Light Emitting Diode*) для передачи инфракрасных волн приемнику. Инфракрасная передача ограничена малым расстоянием в прямой зоне видимости и может быть использована в офисных зданиях.

## Вопросы по теме

1. Что такое физическая среда?
2. Что может быть использовано в качестве физической среды передачи данных?
3. Какие вопросы при организации сети решаются на физическом уровне?
4. Что такое кабель?
5. Что такое линии связи?
6. Дать определение каналов связи.
7. Перечислить типы кабелей, используемых для передачи данных в сети.
8. Каково назначение структурированной кабельной системы?
9. На какие классы подразделяются кабельные системы?
10. Что такое 10BaseT?
11. Какой кабель используется в технологии 10Base2?
12. Какой кабель используется в технологии 10Base5?
13. Какие известны кабельные системы Ethernet?
14. Какие существуют типы оптоволоконных кабелей?
15. Какие известны технологи беспроводной передачи данных?
16. В каких случаях используется инфракрасная связь?
17. Назвать преимущества использования радиосвязи.

# 7 Сетевые операционные системы

Сетевые операционные системы (Network Operating System –NOS) – это комплекс программ, обеспечивающих обработку, хранение и передачу данных в сети [32].

## 7.1 Структура сетевой операционной системы

Сетевая ОС составляет основу любой вычислительной сети. Под сетевой ОС в широком смысле понимается совокупность ОС отдельных компьютеров, взаимодействующих с целью обмена сообщениями и разделения ресурсов по единым правилам – протоколам. В узком смысле сетевая ОС – это операционная система отдельного компьютера, обеспечивающая ему возможность работать в сети.

На практике сложилось несколько подходов к построению сетевых ОС (рис. 7.1).

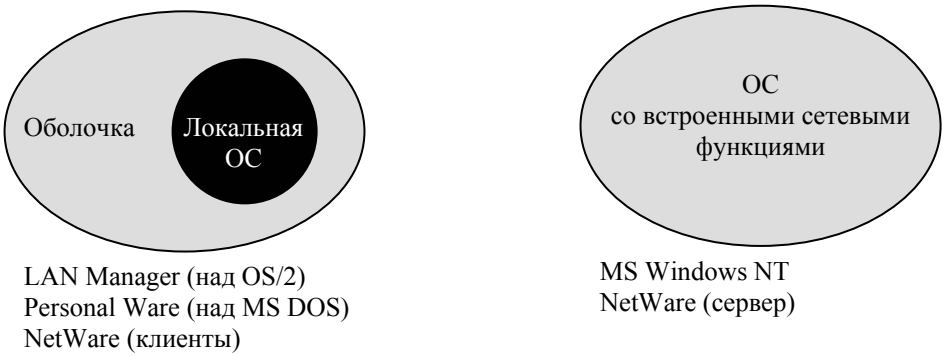


Рис. 7.1. Варианты построения сетевых ОС

Первые сетевые ОС представляли собой совокупность существующей локальной ОС и надстроенной над ней *сетевой оболочки*. При этом в локальную ОС встраивался минимум сетевых функций, необходимых для работы сетевой оболочки, которая выполняла основные сетевые функции. Принцип построения сетевых ОС в виде сетевой оболочки над локальной ОС используется и в современных ОС.

Наиболее эффективным представляется путь разработки операционных систем, изначально предназначенных для работы в сети. Сетевые функции у ОС такого типа глубоко *встроены* в основные модули системы, что обеспечивает их логическую стройность, простоту эксплуатации и модификации, а также высокую производительность.

Общая структура сетевой ОС приведена на рис. 7.2.

В соответствии со структурой, приведенной на рис. 7.2, в сетевой операционной системе отдельной ЭВМ можно выделить несколько частей:

1. средства управления локальными ресурсами компьютера: функции распределения оперативной памяти между процессами, планирования и диспетчеризации процессов, управления процессорами, управления пе-

риферийными устройствами и другие функции управления локальными ресурсами;

2. средства предоставления собственных ресурсов и услуг в общее пользование – серверная часть ОС (сервер). Эти средства обеспечивают, например, блокировку файлов и записей, ведение справочников имен сетевых ресурсов; обработку запросов удаленного доступа к собственной файловой системе и базе данных; управление очередями запросов удаленных пользователей к своим периферийным устройствам;
3. средства запроса доступа к удаленным ресурсам и услугам – клиентская часть ОС. Эта часть выполняет распознавание и перенаправление в сеть запросов к удаленным ресурсам от приложений и пользователей. Клиентская часть также осуществляет прием ответов от серверов и преобразование их в локальный формат, так что для приложения выполнение локальных и удаленных запросов неразличимо;
4. коммуникационные средства ОС, с помощью которых происходит обмен сообщениями в сети. Эта часть обеспечивает адресацию и буферизацию сообщений, выбор маршрута передачи сообщения по сети, надежность передачи и т.п., т. е. является средством транспортировки сообщений.


Рис. 7.2. Структура сетевой ОС

В зависимости от функций, возлагаемых на конкретный компьютер, в его операционной системе может отсутствовать либо клиентская, либо серверная части.

На рис. 7.3 компьютер 1 выполняет функции клиента, а компьютер 2 – функции сервера, соответственно на первой ЭВМ отсутствует серверная часть, а на второй – клиентская.

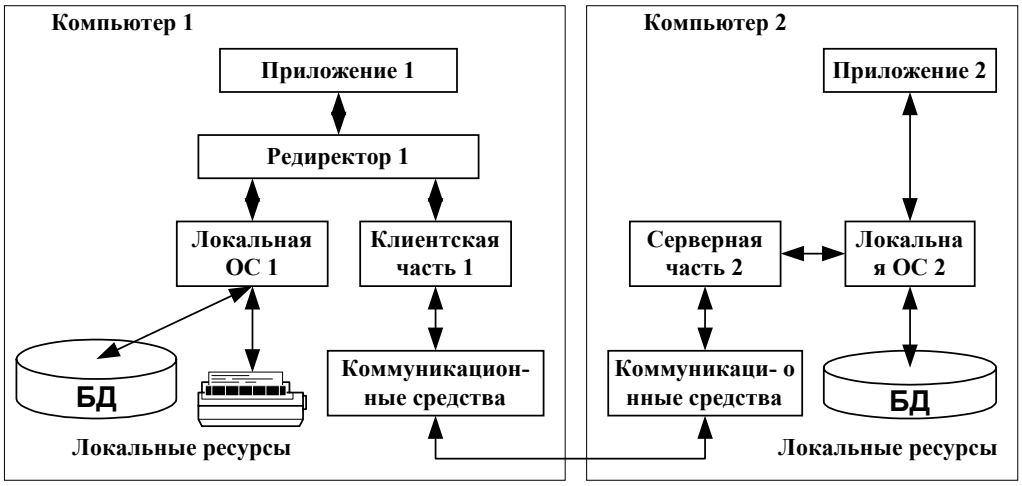


Рис. 7.3. Взаимодействие компонентов ОС

В случае возникновения запроса к ресурсу данного компьютера, он переадресовывается локальной ОС. Если же это запрос к удаленному ресурсу, то он направляется в клиентскую часть, где преобразуется из локальной формы в сетевой формат, и передается коммуникационным средствам. Серверная часть ОС компьютера 2 принимает запрос, преобразует его в локальную форму и передает для выполнения своей локальной ОС. После того, как результат получен, сервер обращается к транспортной подсистеме и направляет ответ клиенту, выдавшему запрос. Клиентская часть преобразует результат в соответствующий формат и адресует его тому приложению, которое выдало запрос.

В клиентской части ОС можно выделить три основных компонента: редиректоры (redirector), распределители (designator) и имена UNC (UNC pathnames).

Редиректор – сетевое программное обеспечение, которое принимает запросы ввода/вывода для удаленных файлов, именованных каналов или почтовых слотов и затем переназначает их сетевым сервисам другого компьютера.

Редиректоры функционируют на представительском уровне модели OSI. Когда клиент делает запрос к сетевому приложению или службе, редиректор перехватывает этот запрос и проверяет, является ли ресурс локальным (находящимся на запрашивающем компьютере) или удаленным (в сети). Если редиректор определяет, что это локальный запрос, он направляет запрос центральному процессору для немедленной обработки. Если запрос предназначен для сети, редиректор направляет запрос по сети к соответствующему серверу.

Распределитель (designator) представляет собой часть программного обеспечения, управляющую присвоением букв накопителя (drive letter) как локальным, так и удаленным сетевым ресурсам или разделяемым дисководам, что помогает во взаимодействии с сетевыми ресурсами. Когда между сетевым ресурсом и буквой локального накопителя создана ассоциация, из-

вестная также как отображение дисковода (mapping a drive), распределитель отслеживает присвоение такой буквы дисковода сетевому ресурсу. Затем, когда пользователь или приложение получат доступ к диску, распределитель заменит букву дисковода на сетевой адрес ресурса, прежде чем запрос будет послан редиректору.

Редиректор и распределитель являются не единственными методами, используемыми для доступа к сетевым ресурсам. Большинство современных сетевых операционных систем, распознают имена UNC (Universal Naming Convention — Универсальное соглашение по наименованию). UNC представляют собой стандартный способ именования сетевых ресурсов. Эти имена имеют форму *\\Имя_сервера\имя_ресурса*. Способные работать с UNC приложения и утилиты командной строки используют имена UNC вместо отображения сетевых дисков.

## 7.2 Одноранговые сетевые ОС и ОС с выделенными серверами

В зависимости от того как распределены функции между компьютерами сети, сетевые операционные системы, а следовательно, и сети делятся на два класса: одноранговые и сети с выделенными серверами.

Если компьютер предоставляет свои ресурсы другим пользователям сети, то он играет роль сервера. При этом компьютер, обращающийся к ресурсам другой машины, является клиентом. Компьютер, работающий в сети, может выполнять функции либо клиента, либо сервера, либо совмещать обе эти функции. На рис. 7.3, 7.4 приведены примеры структур одноранговых сетей и сетей с выделенными серверами.

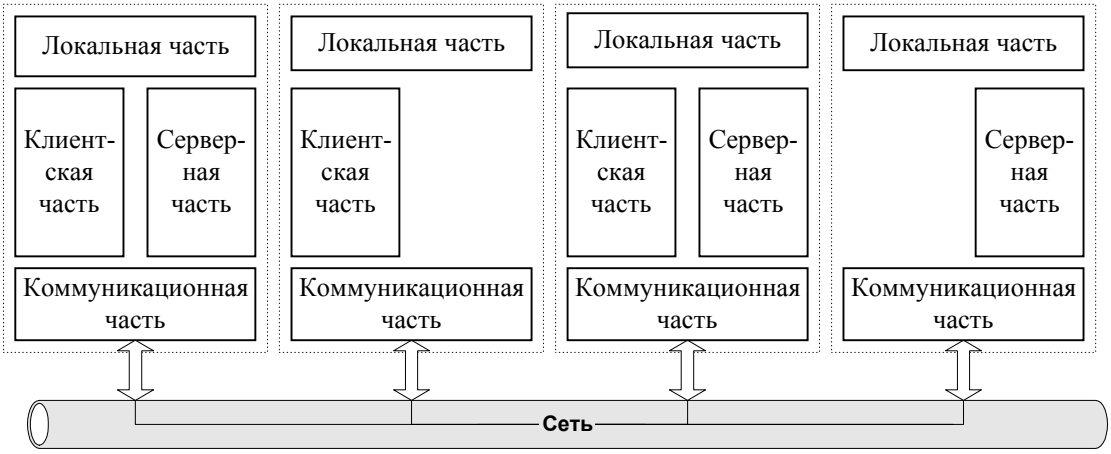


Рис. 7.3 Одноранговая сеть

В одноранговых сетях все компьютеры равноправны. Каждый пользователь может по своему желанию объявить какой-либо ресурс своего компьютера разделяемым, после чего другие пользователи могут его эксплуатировать. В таких сетях в качестве сетевой ОС можно использовать, например, Microsoft Windows 9x, Microsoft Windows for Workgroup, Microsoft Windows NT Workstation. Одноранговые сети проще в организации и эксплуатации. Но они применяются в основном для объединения небольших групп пользовате-

лей, не предъявляющих больших требований к объемам хранимой информации, ее защищенности от несанкционированного доступа и к скорости доступа.

При повышенных требованиях к этим характеристикам более подходящими являются сети с выделенными серверами.

Если выполнение каких-либо серверных функций является основным назначением ЭВМ, то такая ЭВМ называется выделенным сервером. В зависимости от того, какой ресурс сервера является разделяемым, он называется файл-сервером, факс-сервером, принт-сервером и т. д. На выделенных серверах устанавливается ОС специально оптимизированные для выполнения тех или иных серверных функций. Выделенный сервер не принято использовать в качестве компьютера для выполнения текущих задач, не связанных с его основным назначением, так как это снижает его производительность как сервера.

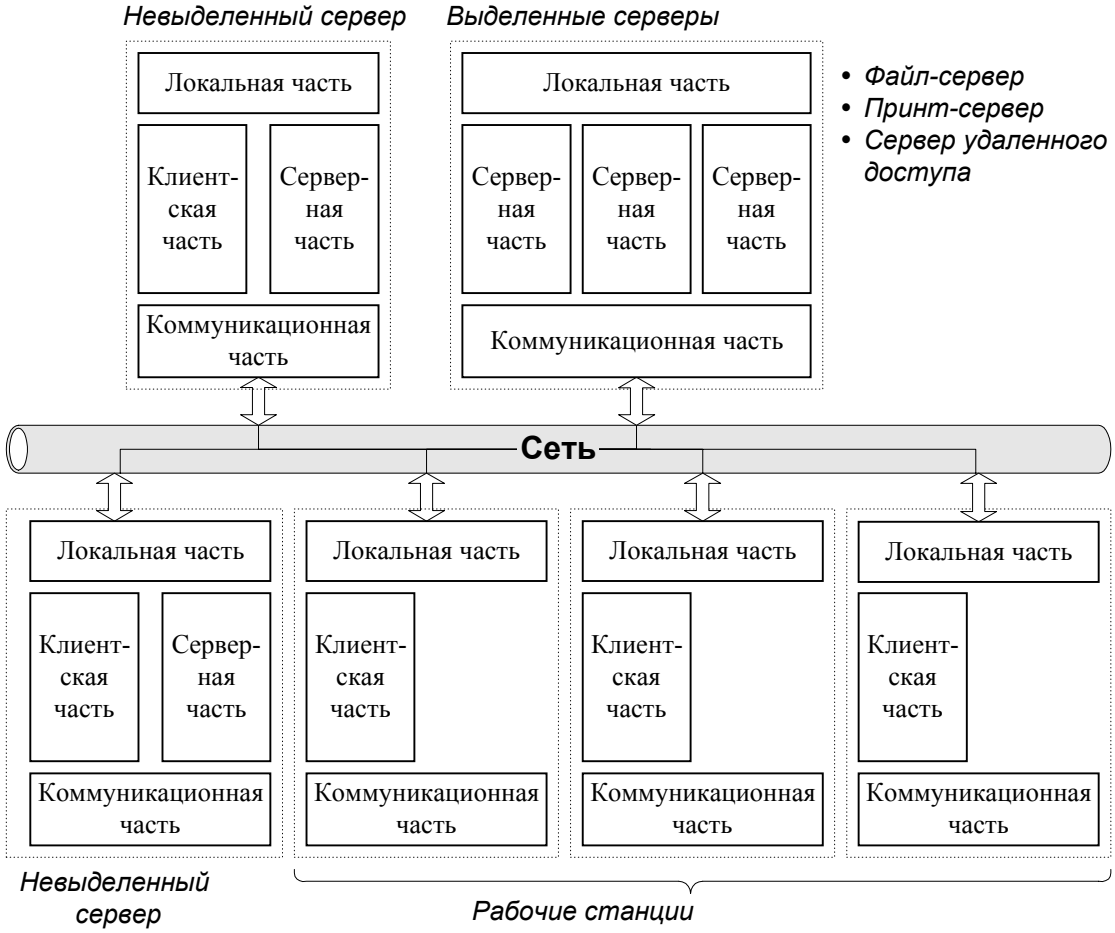


Рис. 7.4. Сеть с выделенным сервером

Сети с выделенными серверами могут быть построены на базе операционных систем NetWare или Microsoft Windows NT Server (для выделенного сервера) и Windows NT Workstation (для рабочей станции).

**Вопросы по теме**

1. Что такое сетевая ОС и каково ее назначение?

2. Какие функции сети выполняет сетевая операционная система?
3. Сформулируйте два основных подхода к построению сетевых ОС.
4. Из каких частей состоит структура сетевой ОС?
5. Что такое редиректор?
6. Охарактеризуйте распределители.
7. Поясните, что такое UNC-имена.
8. Как подразделяются сетевые операционные системы по правам доступа к ресурсам?

# 8 IP-адресация и маршрутизация. DNS

Компьютер, подключенный к сети (ЛВС или Internet) и использующий для связи с другими компьютерами сети протокол TCP/IP, называется *хостом.* Для идентификации каждого хоста в сети имеются следующие два способа адресации: *IP-адресация* и *система доменных имен*. Оба этих способа используются совместно.

## 8.1 Адреса IP

Как уже было отмечено выше, IP-адреса являются 32-битными числами. Каждому компьютеру в рамках сетевой среды должен соответствовать уникальный адрес. Если ЛВС не связана с какими-либо другими сетями, IP-адреса можно назначить исходя из собственных предпочтений. Однако в рамках глобальной сети Internet IP-адреса распределяются специальной организацией Network Information Center (NIC).

Для упрощения восприятия IP-адреса разделяют на четыре части по восемь бит. Эти части называют *октетами (octet).* Например, адресу *isserv.is.mivlgu.ru* соответствует IP-адрес *C0A808FE***,** который записывается как *192.168.8.254.* Такой формат часто называют *точечной нотацией (dotted quad notation).*

IP-адрес, представленный в таком виде, можно разделить на две части, одна из которых является номером сети (располагается в старших октетах), а другая – номером узла в сети (располагается в младших октетах). В зависимости от размера сети эти части могут состоять из разного количества бит. IP-сети делятся на несколько классов. В рамках каждого класса под номер узла отводится различное количество бит.

– *Класс A,* к которому относятся сети с номерами от *1.0.0.0* до *127.0.0.0.* Номер сети содержится в первом октете, а номер сетевого узла состоит из 24 бит.
– *Класс B,* к которому относятся сети с номерами от *128.0.0.0* до *191.255.0.0*, при этом номер сети занимает первые два октета. В этом классе может быть определено 16 320 сетей, каждая из которых может содержать до 65 024 ЭВМ.

– *Класс C,* к которому относятся сети с номерами от *192.0.0.0* до *223.255.255.0*. Номер сети содержится в первых трех октетах. В этом классе может быть определено около 2 миллионов сетей, каждая из которых может содержать до 254 компьютеров.

– *Классы D, E и F*, к которым относятся адреса в диапазоне от *224.0.0.0* до *254.0.0.0.* Эти адреса либо экспериментальные, либо зарезервированы для дальнейшего использования и не предназначены для адресации каких-либо сетей.

Номера сетевых узлов, целиком состоящие из октетов *000* или *256*, зарезервированы для специального использования. Адрес, где номер узла имеет нулевое значение, обозначает целиком всю сеть, а адрес, где номер узла состоит из одних единиц, обозначает адрес широковещательной передачи данных (broadcast address). При пересылке данных по такому адресу сообщение будет принято всеми ЭВМ сети.

Кроме вышеперечисленных существуют также еще два зарезервированных IP-адреса. Это адреса *0.0.0.0* и *127.0.0.0.*

Первый называют *маршрутом по умолчанию (default route),* второй – *адресом интерфейса локальной передачи данных (loopback address).* Адрес маршрута по умолчанию имеет отношение к методу определения пути передачи пакетов через сеть IP. Сеть с номером *127.0.0.0* зарезервирована для передачи сообщений, локальных для сетевого узла. Обычно адрес *127.0.0.1* соответствует специальному интерфейсу, который называют *интерфейсом локальной передачи данных (loopback interface).* Любой пакет IP, переданный через этот интерфейс, будет возвращен обратно сетевому уровню ядра, точно так же, как если бы он был принят из сетевого канала связи. Такой интерфейс широко используется для отладки сетевых программ в процессе их написания на локальной ЭВМ.

## 8.2 Сопоставление сетевых адресов

Протокол Ethernet идентифицирует сетевые узлы при помощи адреса, состоящего из шести октетов (назначается при изготовлении сетевого адаптера), и это число не имеет ничего общего с IP-адресом.

Как уже было отмечено выше для сопоставления этих адресов используются протокол разрешения адреса *Address Resolution Protocol (ARP)* и реверсивный ARP – *RARP (Reverse Address Resolution Protocol)* , т.е. протокол обратного разрешения адреса.

В IP-сетях эти протоколы использует широковещательную передачу сообщений.

## 8.3 Маршрутизация в сетях IP

Для определения по IP-адресу физического местоположения компьютера в сети служит маршрутизация.

**Сети IP**

В качестве примера типичной маршрутизации можно привести работу обычной почтовой службы. Когда человек пишет письмо, он указывает на конверте полный адрес, включая страну, регион, почтовый индекс и т. д. После того как письмо попадает в почтовую службу, она доставляет его по адресу назначения: сначала письмо пересылается в указанную страну, затем местная почтовая служба передает письмо в нужный регион, где региональная почтовая служба пересылает его почтовому отделению соответствующего населенного пункта, и так далее, пока письмо не дойдет до адресата.

IP-сети, структурированы подобным образом. Весь глобальная вычислительная сеть состоит из набора фрагментов, называемых *автономными системами (autonomous systems)*. Каждая такая система осуществляет маршрутизацию между входящими в нее узлами самостоятельно, т.е. как только пакет попадает на один из компьютеров, являющихся членом нужного сетевого фрагмента, дальнейшая передача пакета осуществляется средствами самой автономной системы.

**Подсети IP**

Возможность разделения IP-адреса на части, соответствующие номеру сети и номеру узла. позволяет разбить сеть IP на множество *подсетей (subnet)*. Каждая такая подсеть осуществляет доставку пакетов владельцам IP-адресов, принадлежащих определенному диапазону адресов. Биты, рассматриваемые как номер подсети, определяются так называемой *маской подсети (subnet mask),* или *сетевой маской (netmask)*. Сетевая маска также является 32-битным числом. Биты маски, соответствующие адресу подсети, имеют значение 1, а биты, соответствующие адресу сетевого узла, имеют значение 0. Например, сети класса B *149.76.0.0* соответствует сетевая маска *255.255.0.0*. Вся подобная сеть может быть разделена на 254 более мелкие подсети с адресами от *149.76.1.0* до *149.76.254.0.* Исходя из этого маска подсети будет *255.255.255.0.*

На рис. 8.1 показано, каким образом адрес ЭВМ *149.76.12.4* интерпретируется по разному в зависимости от того, рассматривается ли оно как адрес в рамках обычной сети класса B или как адрес в рамках подсети сети класса B.

Адрес в сети класса B

| Номер сети | | Номер узла | |
|---|---|---|---|
| 149 | 76 | 12 | 4 |

Адрес в одной из подсетей класса B

| Номер сети | | Номер подсети | Номер узла |
|---|---|---|---|
| 149 | 76 | 12 | 4 |

Рис. 8.1. Разделение адреса сети класса B на подсети

Подсети создаются владельцем сети исходя из административных предпосылок, а чаще исходя из существующей сетевой инфраструктуры предприятия или учреждения, на базе которого данная сеть функционирует.

**Шлюзы**

Чаще всего каждая из подсетей представляет собой отдельную самостоятельную ЛВС. Каждая из ЭВМ, входящих в состав подсети, может напрямую связываться лишь с ЭВМ, входящими в состав этой же физической подсети. Связь с другими ЭВМ происходит через так называемые *шлюзы (gateways)*. Шлюзом называется сетевой узел, подключенный одновременно к двум или более физическим сетям и настроенный таким образом, чтобы передавать пакеты между этими сетями.

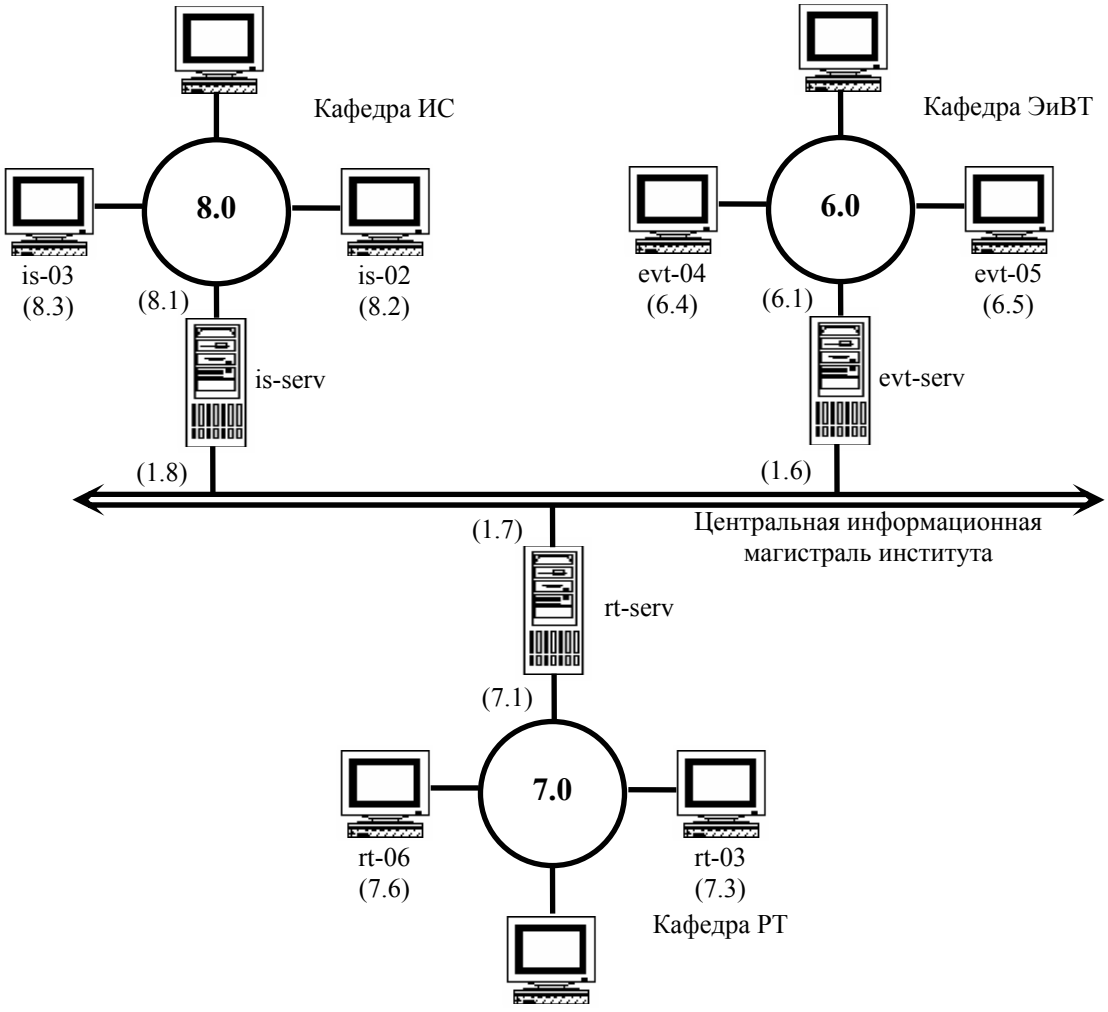


Рис. 8.2. Фрагмент сети института

Чтобы определить, является ли узел членом локальной физической сети или нет, различным физическим сетям назначаются различные IP-адреса.

Например, информационная сеть института состоит из трех локальных физических подсетей с адресами *192.168.8.0* – Кафедра ИС, *192.168.6.0* – Кафедра ЭиВТ и *192.168.3.0* – Кафедра РТ, а также объединяющей их цен-

тральной информационной магистралью – подсетью с адресом *192.168.1.0* (см. рис. 8.2.).

При пересылке пакета данных, предназначенного для ЭВМ с адресом *192.168.6.5* (*evt-05*) сетевое программное обеспечение ЭВМ с адресом *192.168.8.2* (*is-02*), пославшей этот пакет, при помощи имеющейся сетевой маски (*255.255.255.0*) определяет, что компьютер*192.168.6.5* (*evt-05*) принадлежит другой физической сети и поэтому пакет данных можно переслать только через шлюз по умолчанию, которым в данном случае является ЭВМ с адресом *192.168.8.1* (*is-serv*).

ЭВМ с адресом *192.168.8.1* (*is-serv*) при помощи двух интерфейсов подсоединена одновременно к двум различным подсетям с адресами *192.168.8.0* (Кафедра ИС) и *192.168.1.0* (Центральная информационная магистраль института). Таким образом, этот компьютер имеет два адреса: в локальной сети Кафедра ИС (*192.168.8.0*) – адрес *192.168.8.1*, а во внешней сети (*192.168.1.0*) – адрес *192.168.1.8*. Т.е. шлюз имеет столько IP-адресов, к скольким физическим сетям он одновременно подключен. Эти адреса вместе с соответствующими сетевыми масками связаны с интерфейсами, через которые происходит обращение к подсетям. Соответствие адресов и интерфейсов для ЭВМ *192.168.8.1* (*is-serv*) может быть описано следующим образом.

Таблица 8.1. Соответствие адресов и интерфейсов ЭВМ *is-serv*.

| **Интерфейс** | **Адрес** | **Сетевая маска** |
|---|---|---|
| Подключение по локальной сети #1 | 192.168.8.1 | 255.255.255.0 |
| Подключение по локальной сети #2 | 192.168.1.8 | 255.255.255.0 |
| Интерфейс локальной передачи | 127.0.0.1 | 255.0.0.0 |

**Таблица маршрутов**

Рассмотрим, каким образом система IP определяет адрес шлюза, который необходимо использовать для передачи пакета из одной подсети в другую.

Итак (см. рис. 8.2), ЭВМ с адресом *192.168.8.2* (*is-02*), отправляя пакет на ЭВМ с адресом *192.168.6.5* (*evt-05*), при помощи наложения маски обнаруживает, что данная ЭВМ принадлежит к другой физической сети. Следовательно оправить пакет данных на нее можно только через шлюз, т.е. ЭВМ с адресом *192.168.8.1* (*is-serv*). (Адрес шлюза по умолчанию задается при конфигурировании сетевого адаптера). Теперь перед ЭВМ *192.168.8.1* (*is-serv*) возникает аналогичная задача, поскольку ЭВМ с адресом *192.168.6.5* (*evt-05*) не принадлежит ни к одной из физических сетей, к которым подключена *is-serv*. Поэтому необходимо определить, на какой шлюз следует отправить пакет, чтобы он попал в подсеть назначения. Этим компьютером, в данном случае, является ЭВМ с адресом *192.168.1.6* (*evt-serv*) – шлюз между центральной информационной магистралью и подсетью *192.168.6.0*.

В этом случае ЭВМ с адресом *192.168.8.1* (*is-serv*) должна была обладать некоторой дополнительной информацией, необходимой для определения правильного маршрута.

Информация, необходимая для определения правильного маршрута, содержится в специальной таблице, ставящей в соответствие каждой из имеющихся подсетей адрес шлюза, который необходимо использовать для пересылки пакета в эту сеть. В этой таблице также указывается шлюз по умолчанию, который будет использован в случае, если требуется переслать пакет в сеть с неизвестным адресом. Обычно это шлюз, связанный с сетью *0.0.0.0.* На ЭВМ с адресом *192.168.8.1* (*is-serv*) таблица маршрутизации может выглядеть, например, как это показано в таблице 8.2.

Таблица 8.2 Таблица маршрутизации ЭВМ *is-serv.*

| **Сеть** | **Шлюз** | **Интерфейс** |
|---|---|---|
| 192.168.1.0 | – | Подключение по локальной сети #2 |
| 192.168.2.0 | 149.76.1.2 | Подключение по локальной сети #2 |
| 192.168.3.0 | 149.76.1.3 | Подключение по локальной сети #2 |
| 192.168.6.0 | 149.76.1.6 | Подключение по локальной сети #2 |
| 192.168.7.0 | 149.76.1.7 | Подключение по локальной сети #2 |
| 192.168.8.0 | – | Подключение по локальной сети #1 |
| . . . | . . . | . . . |
| 0.0.0.0 | 192.168.1.7 | Подключение по локальной сети #2 |

Сети, к которым непосредственно подключена ЭВМ с адресом *192.168.8.1* (*is-serv*)**,** не требуют использования шлюзов для передачи пакетов. Поэтому в соответствующих им графах стоят прочерки.

Таблицы маршрутизации могут быть построены разными способами. В небольших локальных сетях лучше всего создавать такие таблицы вручную и подключать их к системе IP при помощи команды *routy (rout)*. Для более крупных сетей такие таблицы периодически обновляются средствами маршрутизации в процессе работы сети. Средства маршрутизации для своих целей чаще всего используют протокол RIP (Routing Information Protocol, Информационный протокол маршрутизации).

## 8.4 Назначение IP-адресов узлам

Как уже было отмечено выше, каждое устройство, использующее протокол TCP/IP, должно иметь сетевой адрес. В отличие от протоколов NetBEUI или IPX/SPX, при использовании которых рабочие станции автоматически получают сетевой адрес, каждому узлу IP-сети адрес должен быть назначен тем или иным способом.

Один из способов назначить IP-адрес рабочей станции – выполнить настройку вручную. Если сеть имеет значительное число узлов, то при ручной настройке задание сетевого адреса рабочей станции и прочих параметров может занять значительное время, а управление сетью с заданными вручную

адресами может оказаться очень трудоемким. Для автоматизации процесса назначения адресов можно использовать такие механизмы, как протокол динамической настройки узла (Dynamic Host Configuretion Protocol, DHCP).

**Динамическое назначение адресов при помощи DHCP**

Протокол DHCP является протоколом назначения адресов. Цель DHCP – предоставить механизм передачи рабочим станциям информации о настройках IP, таких как IP-адрес, используемый по умолчанию шлюз и информация о сервере DNS.

Протокол DHCP основан на концепции клиент-сервер. На сервере DHCP хранится таблица доступных и использованных IP-адресов. Клиенты, которым необходимо получить или обновить зарезервированный за ними IP-адрес, отправляют запрос на сервер. Сервер DHCP передает адрес и родственную информацию клиенту, который использует этот адрес в течение указанного времени. По истечении половины указанного срока клиент отправляет на сервер запрос на обновление адреса.

Преимущество использования DHCP в сети заключается в том, что администраторы сети освобождаются от работы по ручному назначению IP-адресов и управлению ими, а также от установки на каждой рабочей станции информации о расположении шлюза по умолчанию и серверов DNS и WINS. После того как сервер DHCP установлен, администратору необходимо лишь определить масштаб его работы и задать информацию о маршрутизаторе, серверах DNS, имени домена и серверах WINS. Настройка клиента DHCP чрезвычайно проста.

После того как сервер и клиенты будут настроены, DHCP работает абсолютно прозрачно для конечного пользователя. Пользователи могут получать новые IP-адреса автоматически, без вмешательства администратора.

**Ограничения DHCP**

Протокол DHCP имеет одно существенное ограничение. Клиенты DHCP передают запросы как широковещательные сообщения, поэтому они не распространяются автоматически через сетевые маршрутизаторы. Таким образом, сервер DHCP будет обрабатывать запросы только тех клиентов, которые находятся с ним в одном сегменте.

Для решения этой проблемы многие маршрутизаторы, включают программный маршрутизатор, входящий, например, в состав Windows NT и имеют возможность передачи DHCP-запросов.

**Использование WINS**

Технология DHCP не позволяет передавать рабочим станциям информацию о других ресурсах сети, например, о серверах в других сегментах. Microsoft разработала службу, которая обеспечивает хранение динамически изменяемой таблицы соответствия IP-адресов рабочих станций (или серверов) их

именам NetBIOS, вне зависимости от того, в каком сегменте расположено устройство. Эта служба называется службой определения имен Интернета (Windows Internet Naming Service, WINS).

Чтобы использовать WINS, необходимо настроить соответствующую службу на сервере Microsoft Windows NT, а на каждой рабочей станции, являющейся клиентом WINS, задан адрес ее сервера WINS (может быть получен при помощи DHCP). В этом случае, при подключении рабочей станции к сети, устанавливается соединение с сервером WINS и ему отправляется запрос на регистрацию имени. Если имя рабочей станции не использовано, сервер WINS добавляет его в свою базу данных. Когда рабочая станция завершает работу, она отправляет запрос серверу WINS на освобождение имени. На протяжении работы станции сервер WINS производит трансляцию ее имени в IP-адрес для всех узлов, запрашивающих информацию об этой станции.

Так же, как и в случае использования DHCP, после того как рабочая станция зарегистрирует свое имя на сервере WINS, проведенная регистрация имеет определенное время жизни. Клиент должен обновить регистрацию на сервере WINS до того, как время жизни истечет.

Поскольку общение между клиентом и сервером WINS производится не при помощи передачи широковещательных сообщений, клиент и сервер могут находиться в абсолютно разных сегментах. Кроме того, сервер WINS независим от базы данных учетных записей пользователей, поэтому один сервер WINS может обслуживать несколько доменов.

Одним из основных ограничений на широкое использование WINS является поддержка только клиентов, разработанных Microsoft. UNIX-системы и пользователи Macintosh не могут использовать WINS для определения адресов. Обычно такие системы используют для определения адресов серверы DNS.

## 8.5 Служба разрешения доменных имен (DNS)

### Разрешение имен сетевых узлов

Как уже было сказано выше, в IP-сетях адрес узла представляет собой 32-битное число. Однако человеку сложно запоминать такие адреса, поэтому сетевые узлы обычно называют информативными символьными именами. Таким образом, чтобы осуществить пересылку данных через сеть, необходимо, исходя из символьного имени, определить соответствующий IP-адрес. Этот процесс называется *разрешением имен сетевых узлов (host name resolution)*, а средство осуществляющие этот процесс – *резольвером (resolver).*

В небольших сетях поддерживать таблицы соответствия символьных имен узлов IP-адресам вручную не сложно. Обычно эта информация хранится в так называемых *hosts*-файлах. Если в сети появляются новые компьютеры, удаляются старые или происходит переопределение какого-либо адреса,

необходимо исправить содержимое файла *hosts* на всех компьютерах сети. Очевидно, что задача обновления всех этих файлов может оказаться весьма трудоемкой, если речь идет о сети с большим количеством узлов. Поэтому ранее в ЛВС для хранения подобной информации использовались специальные ЭВМ, но при распространении Internet это оказалось не эффективным (нагрузка на компьютеры хранящие *hosts*-файлы сильно возрастала). Кроме того присвоение каждому новому узлу уникального имени также стало серьезной проблемой. Для решения этих проблем в 1984 году была принята новая схема разрешения имен. Это была *служба разрешения доменных имен (Domain Name System, DNS)*.

**Введение в DNS**

Служба DNS организует имена узлов в иерархию доменов. Домен – это набор узлов, в некотором смысле связанных между собой. Все эти узлы могут принадлежать к одной сети (например, все машины, входящие в состав локальной сети), все они также могут принадлежать к одной организации объединяющей несколько ЛВС, наконец, все они могут быть просто близко расположены друг от друга в географическом смысле. Например, все учебные заведения США входят в состав домена *edu***,** а каждому университету или колледжу соответствует свой *субдомен (subdomain),* в состав которого входят все его ЭВМ (например, некоторому университету соответствует домен *university.edu***,** а департаменту математики этого университета – *math.university.edu*). Все компьютеры, входящие в состав локальной сети этого департамента, должны содержать в своем названии имя этого домена. Например, полное имя компьютера *comp1* будет в этом случае *comp1. math.university.edu***.** Это имя называется *полным доменным именем (fully qualified domain name, FQDN).* Оно точно идентифицирует сетевой узел в рамках всемирной сети.

На рис. 8.3 показан фрагмент пространства имен. Корневая вершина этого дерева, обозначенная символом точки (.), называется *корневым доменом (root domain)*. Корневой домен включает в себя все остальные домены. В большинстве случаев имя домена верхнего уровня определяет имя национального домена, например, Финляндия использует домен *fi*, Франция – домен *fr*, Германия – домен *de*, Австралия — домен *au*, Россия – домен *ru*. Для США национальный домен по традиции опускается, там самыми крупными объединениями являются сети образовательных (*edu*), коммерческих (*com*), государственных (gov), военных (*mil*) и т.д. учреждений.

Благодаря введению подобной иерархической организации доменных имен проблема уникальности имен узлов в рамках Internet была фактически решена. DNS решает не только эту проблему. Она позволяет значительно упростить процедуры поддержки сети, перепоручая обязанности по администрированию субдоменов их администраторам. При этом администраторы подсети могут назначать входящим в него компьютерам любые имена и любые

IP-адреса в рамках предоставленного им диапазона. Целостность всей сети нарушена не будет.

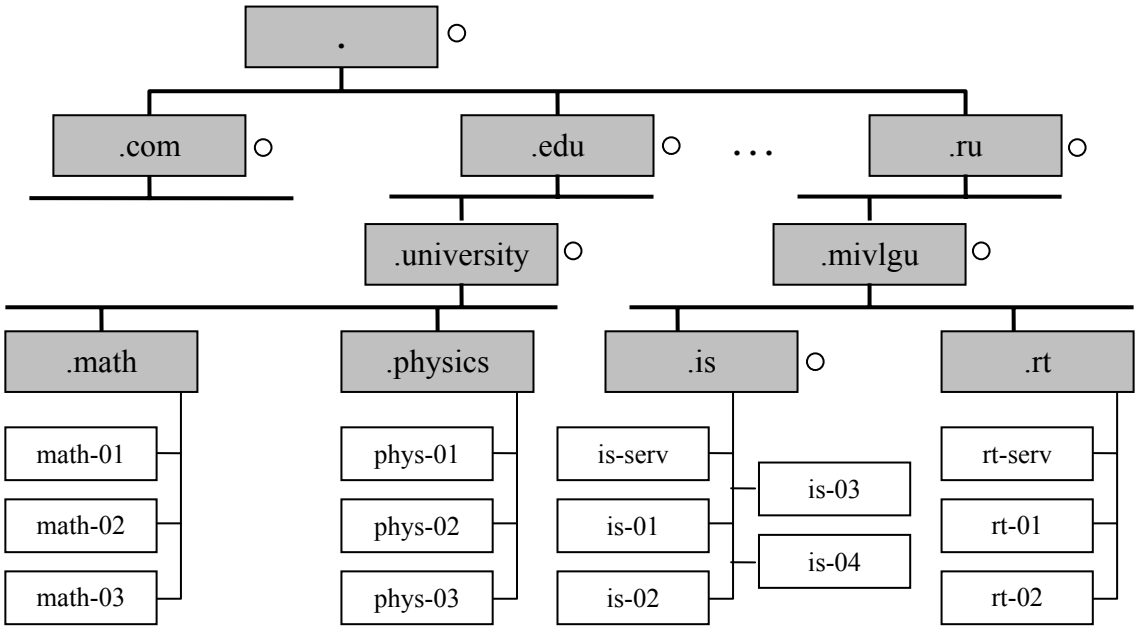


Рис. 8.3. Фрагмент пространства имен доменов

С этой целью пространство имен разделяется на *зоны (zones)*. Следует заметить, что зона и домен несколько разные понятия, например, (см. рис. 8.3) *домен mivlgu.ru* включает в себя все компьютеры института МИВлГУ, в то время как *зона mivlgu.ru* включает только ЭВМ Кафедры РТ, напрямую управляемые компьютерным центром института, ЭВМ Кафедры ИС входят в отдельную зону. На рис. 8.3 начало каждой зоны отмечено кружком справа от имени домена.

**Поиск имени в системе DNS**

Служба DNS представляет собой огромную распределенную базу данных. Доступ к информации из этой базы осуществляется при помощи *серверов имен (name severs),* каждый из которых предоставляет информацию о компьютерах одного или нескольких доменов. В каждой зоне имен существуют как минимум два (а может быть, и больше) сервера имен, хранящих информацию о сетевых узлах этой зоны. Чтобы узнать IP-адрес компьютера *rt-serv***,** необходимо обратиться к серверу имен зоны *mivlgu.ru* (см. рис. 8.3).

Рассмотрим как работает система DNS. Когда некоторая ЭВМ желает получить информацию о компьютере *rt-serv.rt.mivlgu.ru***,** она адресует запрос локальному серверу имен. Не обнаружив в своей базе данных соответствующего имени, локальный сервер имен адресует запрос об имени *rt-serv.rt.mivlgu.ru* серверу имен корневого домена. Сервер имен корневого домена обнаруживает, что это имя не принадлежит к его зоне имен, однако он также обнаруживает, что это имя принадлежит к одной из зон имен домена *ru*. В результате он рекомендует перенаправить запрос серверу имен

*ru*. В результате он рекомендует перенаправить запрос серверу имен домена *ru* и сообщает список всех серверов имен этого домена с соответствующими им адресами. Локальный сервер имен посылает запрос одному из серверов имен домена *ru*, например серверу *xxx.ru.* Сервер *xxx.ru* знает, что это имя принадлежит к зоне *mivlgu.ru***,** обслуживание которой осуществляют собственные серверы имен. Поэтому сервер *xxx.ru* рекомендует перенаправить запрос одному из серверов имен домена *mivlgu.ru***,** и сообщает возможные адреса этих серверов. В результате локальный сервер направляет запрос на один из указанных серверов, который, распознав, что требуемое имя принадлежит его зоне имен, возвращает искомый IP-адрес.

На первый взгляд, может показаться, что при использовании такой технологии вся сеть перегружается обилием передаваемых запросов. На самом деле количество передаваемых данных при использовании системы DNS значительно меньше, чем при хранении всей информации в файле *hosts* и передаче через сеть всего содержимого этого файла.

Для уменьшения временной задержки при последующих обращениях по одному и тому же адресу локальный сервер имен запоминает полученную информацию о соответствии некоторого имени некоторому IP-адресу в собственном кэше. Поэтому, если в следующий раз какая-либо ЭВМ из той же подсети пожелает узнать IP-адрес ЭВМ, принадлежащей домену *mivlgu.ru*, то локальный сервер имен не будет повторять заново весь процесс, а напрямую свяжется с сервером имен домена.

Локальный сервер имен не хранит информацию в кэш вечно. Он стирает ее спустя некоторый период времени, называемый *временем жизни (time to live, TTL)*. Этот параметр сохраняется в базе данных DNS для каждого из адресов. Его значение определяется администраторами соответствующей зоны имен.

### Вопросы по теме

1. Какие способы адресации используются для идентификации узла в сети?
2. На какие две части делится IP-адрес узла в сети?
3. Что такое классы IP-сетей?
4. Какие адреса зарезервированы и не могут использоваться для идентификации узлов в сети?
5. Что такое маршрутизация?
6. Дайте определение автономной системы и поясните ее назначение.
7. Что такое сетевая маска, каково ее назначение?
8. Каким образом производится разделение IP-адреса на адрес подсети и адрес узла?
9. Что такое шлюзы, каково их основное назначение?
10. Какое количество сетевых интерфейсов и IP-адресов имеет шлюз?

11. Что такое таблица маршрутизации?
12. Приведите пример таблицы маршрутизации, поясните смысл содержащихся в ней записей.
13. Каким образом происходит передача пакетов между ЭВМ, находящимися в разных физических подсетях?
14. Что означают прочерки в таблице маршрутизации?
15. Каким образом производится составление таблицы маршрутизации?
16. Каким образом может производиться назначение IP-адресов узлам сети?
17. Что такое DHCP и каковы ограничения использования DHCP?
18. Что такое WINS, каковы недостатки использования WINS?
19. Что значит «разрешение имен сетевых узлов», каковы основные способы разрешения имен?
20. Какие проблемы возникают при использовании для разрешения имен hosts-файлов?
21. Что такое домен? Что такое DNS?
22. Что такое зона?
23. Каким образом осуществляется поиск имен в системе DNS.

# 9 Сетевое оборудование

## 9.1 Сетевые адаптеры

### Назначение

Сетевые адаптеры – это сетевое оборудование, обеспечивающее функционирование сети на физическом и канальном уровнях.

Сетевой адаптер относится к периферийному устройству компьютера, непосредственно взаимодействующему со средой передачи данных, которая прямо или через другое коммуникационное оборудование связывает его с другими компьютерами. Это устройство решает задачи надежного обмена двоичными данными, представленными соответствующими электромагнитными сигналами, по внешним линиям связи. Как и любой контроллер компьютера, сетевой адаптер работает под управлением драйвера операционной системы, и распределение функций между сетевым адаптером и драйвером может изменяться от реализации к реализации.

Внешний вид адаптера показан на рис. 9.1.

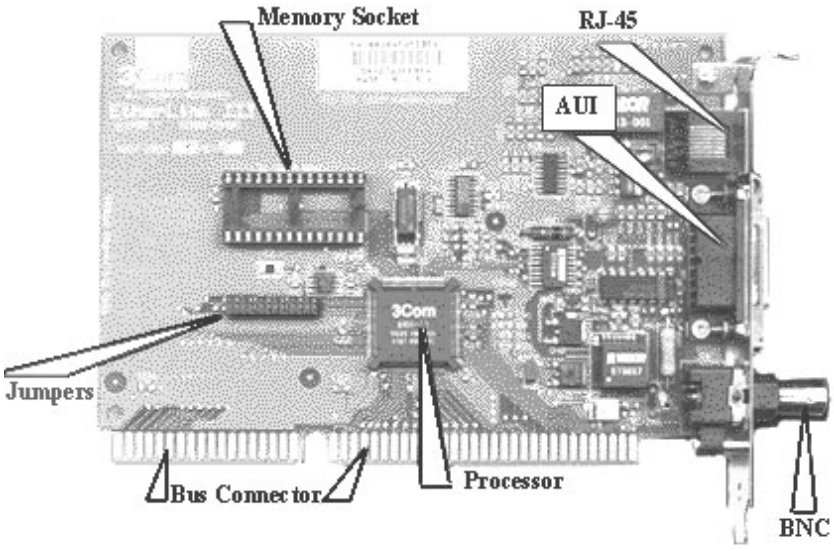


Рис. 9.1. Вид сетевого адаптера

**Настройка сетевого адаптера и трансивера**

Для работы ПК в сети надо правильно установить и настроить сетевой адаптер. Для адаптеров, отвечающих стандарту PnP, настройка производится автоматически. В ином случае необходимо настроить линию запроса на прерывание IRQ (Interrupt Request Line) и адрес ввода/вывода (Input/Output address). Адрес ввода/вывода – это трехзначное шестнадцатеричное число, которое идентифицирует коммуникационный канал между аппаратными устройствами и центральным процессором. Запросы на прерывание IRQ и адреса ввода/вывода для основных устройств ЭВМ можно найти в справочной литературе [1, 3].

Сетевые карты поддерживают различные типы сетевых соединений. Физический интерфейс между самой сетевой картой и сетью называют трансивером (transceiver) – это устройство, которое принимает и посылает данные. Тип трансивера, который должна использовать сетевая карта в соответствии со схемой сети, может выбираться автоматически, а может вручную при помощи перемычек (джамперов) на сетевом адаптере.

**Функции сетевых адаптеров**

Сетевые адаптеры выполняют девять основных функций при приеме или передачи данных:

1. *гальваническая развязка* с коаксиальным кабелем или витой парой. Для этой цели используются чаще всего импульсные трансформаторы;
2. *прием (передача) данных*. Данные передаются из ОЗУ ЭВМ в адаптер или из адаптера в память ЭВМ через программируемый канал ввода/вывода, канал прямого доступа или разделяемую память;
3. *буферизация*. Для согласования скоростей пересылки данных в адаптер или из него со скоростью обмена по сети используются буфера. Во вре-

мя обработки в сетевом адаптере, данные хранятся в буфере. Буфер позволяет адаптеру осуществлять доступ ко всему пакету информации;

4. *формирование пакета.* Сетевой адаптер должен разделить данные на блоки в режиме передачи (или соединить их в режиме приема) данных и оформить в виде кадра определенного формата;
5. *доступ к каналу связи.* Набор правил, обеспечивающих доступ к среде передачи. Выявление конфликтных ситуаций и контроль состояния сети.
6. *идентификация своего адреса* в принимаемом пакете. Физический адрес адаптера может определяться установкой переключателей, храниться в специальном регистре или прошиваться в ППЗУ;
7. *преобразование* параллельного кода в последовательный код при передаче данных, и из последовательного кода в параллельный при приеме. В режиме передачи данные передаются по каналу связи в последовательном коде;
8. *кодирование и декодирование данных.* На этом этапе должны быть сформированы электрические сигналы, используемые для представления данных. Большинство сетевых адаптеров для этой цели используют манчестерское кодирование;
9. *передача или прием импульсов.* В режиме передачи закодированные электрические импульсы данных передаются в кабель (при приеме импульсы направляются на декодирование).

Сетевые адаптеры вместе с сетевым программным обеспечением способны распознавать и обрабатывать ошибки, которые могут возникнуть из-за электрических помех, коллизий или плохой работы оборудования.

**Типы сетевых адаптеров**

Сетевые адаптеры различаются по типу и разрядности используемой в компьютере внутренней шины данных – ISA, EISA, PCI, MCA.

Сетевые адаптеры различаются также по типу принятой в сети сетевой технологии – Ethernet, Token Ring, FDDI и т.п. Как правило, конкретная модель сетевого адаптера работает по определенной сетевой технологии (например, Ethernet). В связи с тем, что для каждой технологии сейчас имеется возможность использования различных сред передачи данных, сетевой адаптер может поддерживать как одну, так и одновременно несколько сред. В случае, когда сетевой адаптер поддерживает только одну среду передачи данных, а необходимо использовать другую, применяются трансиверы и конверторы.

Наиболее распространены перечисленные ниже типы адаптеров.

Адаптеры Ethernet представляют собой плату, которая устанавливается в свободный слот ISA или PCI системной платы ЭВМ (существуют и встроенные в системную плату сетевые адаптеры). Адаптеры Ethernet имеют для связи с сетью два внешних разъема: для коаксиального кабеля (разъем BNC) и

для кабеля на витой паре или один из них. Для выбора типа кабеля применяются перемычки или переключатели, которые устанавливаются перед подключением адаптера к сети.

Адаптеры Fast Ethernet производятся изготовителями с учетом определенного типа среды передачи. Сетевой кабель при этом подключается непосредственно к адаптеру (без трансивера).

Оптические адаптеры стандарта 10BASE-FL могут устанавливаться в компьютеры с шинами ISA, PCI, MCA. Эти адаптеры позволяют отказаться от внешних преобразователей среды и от микротрансиверов. При установке этих адаптеров возможна реализация полнодуплексного режима обмена информацией. Для повышения универсальности в оптических адаптерах сохраняется возможность соединения по витой паре с разъемом RJ-45.

Для спецификации 100BASE-FX соединение концентратора и адаптера по оптоволокну осуществляется с использованием оптических соединителей типа SC или ST. Для этой спецификации выпускаются сетевые адаптеры, совместимые с шиной PCI. Адаптеры способны поддерживать как полудуплексный, так и полнодуплексный режим работы.

Сетевые адаптеры для технологии Gigabit Ethernet предназначены для установки в сервера и мощные рабочие станции. Для повышения эффективности работы они способны поддерживать полнодуплексный режим обмена информацией.

Адаптеры FDRI могут использоваться на разнообразных рабочих станциях и в устройствах межсетевого взаимодействия – мостах и маршрутизаторах. Существуют адаптеры FDDI, предназначенные для работы со всеми распространенными шинами: ISA, EISA, VESA Local Bus (VLB) и т. д. В сети FDDI такие устройства, как рабочие станции или мосты подсоединяются к кольцу через адаптеры одного из двух типов: с двойным (DAS) или одиночным (SAS) подключением. Адаптеры DAS осуществляют физическое соединение устройств как с первичным, так и со вторичным кольцом, что повышает отказоустойчивость сети. Такой адаптер имеет два разъема оптического интерфейса. Адаптеры SAS подключают рабочие станции к концентратору FDDI через одиночную оптоволоконную линию в звездообразной топологии. Эти адаптеры представляют собой плату, на которой наряду с электронными компонентами установлен оптический трансивер с разъемом оптического интерфейса.

## 9.2 Повторители и концентраторы

Основная функция *повторителя* (repeater), как это следует из его названия, – повторение сигналов, поступающих на его порт. Повторитель улучшает электрические характеристики сигналов и их синхронность, и за счет этого появляется возможность увеличивать общую длину кабеля между самыми удаленными в сети узлами.

Многопортовый повторитель часто называют *концентратором* (concentrator) или *хабом* (hub), это отражает тот факт, что данное устройство реализует не только функцию повторения сигналов, но и концентрирует в одном центральном устройстве функции объединения компьютеров в сеть. Практически во всех современных сетевых стандартах концентратор является необходимым элементом сети, соединяющим отдельные компьютеры в сеть.

Концентратор или Hub представляет собой сетевое устройство, действующее на физическом уровне сетевой модели OSI.

Отрезки кабеля, соединяющие два компьютера или какие либо два других сетевых устройства, называются *физическими сегментам,* поэтому концентраторы и повторители, которые используются для добавления новых физических сегментов, являются средством физической структуризации сети.

Концентратор – устройство, у которого суммарная пропускная способность входных каналов выше пропускной способности выходного канала. Так как потоки входных данных в концентраторе больше выходного потока, то главной его задачей является концентрация данных. При этом возможны ситуации, когда число блоков данных, поступающее на входы концентратора, превышает его возможности. Тогда концентратор ликвидирует часть этих блоков.

Ядром концентратора является процессор. Для объединения входной информации чаще всего используется множественный доступ с разделением времени. Функции, выполняемые концентратором, близки к задачам, возложенным на мультиплексор. Современные концентраторы имеют порты для подключения к разнообразным локальным сетям.

Концентратор является активным оборудованием. Концентратор служит центром (шиной) звездообразной конфигурации сети и обеспечивает подключение сетевых устройств.

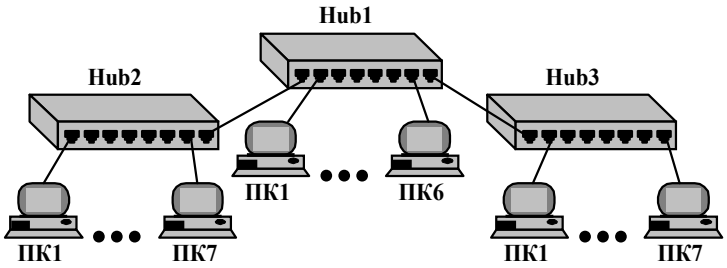


Рис. 9.2 Логический сегмент, построенный с использованием концентраторов

Концентраторы образуют из отдельных физических отрезков кабеля общую среду передачи данных – *логический сегмент*. Логический сегмент также называют доменом коллизий, поскольку при попытке одновременной передачи данных любых двух компьютеров этого сегмента, хотя бы и принадлежащих разным физическим сегментам, возникает блокировка передающей среды. Следует особо подчеркнуть, что, какую бы сложную структуру ни образовывали концентраторы, например путем иерархического соединения

(рис. 9.2), все компьютеры, подключенные к ним, образуют единый логический сегмент, в котором любая пара взаимодействующих компьютеров полностью блокирует возможность обмена данными для других компьютеров.

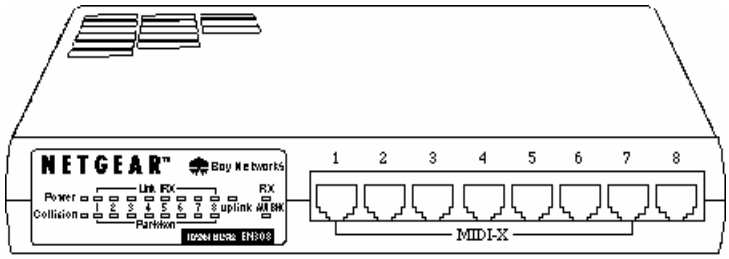


Рис. 9.3 Внешний вид концентратора

На рис. 9.3 показан внешний вид концентратора Концентраторы поддерживают технологию plug and play и не требуют какой-либо установки параметров.

## 9.3 Мосты и коммутаторы

### Мосты

*Мост* (bridge) – ретрансляционная система, соединяющая каналы передачи данных.

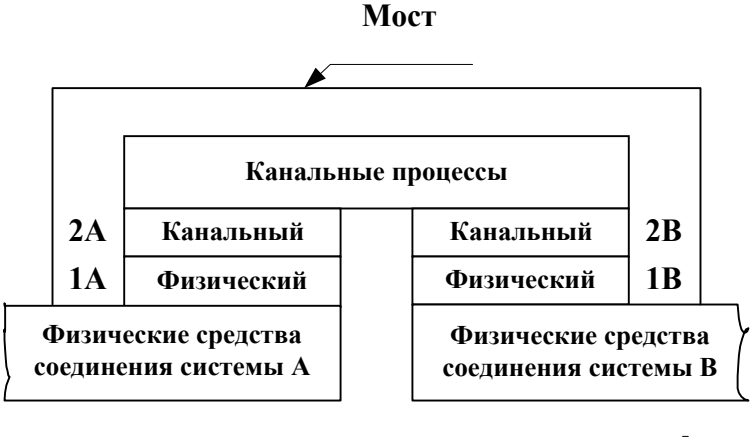


Рис. 9.4 Структура моста

В соответствии с базовой эталонной моделью взаимодействия открытых систем мост описывается протоколами физического и канального уровней, над которыми располагаются канальные процессы. Мост опирается на пару связываемых им физических средств соединения, которые в этой модели представляют физические каналы. Мост преобразует физический (1A, 1B) и канальный (2A, 2B) уровни различных типов (рис. 9.4). Что касается канального процесса, то он объединяет разнотипные каналы передачи данных в один общий.

Мосты могут соединять сегменты, использующие разные типы носителей, например 10BaseT (витая пара) и 10Base2 (тонкий коаксиальный кабель). Они могут соединять сети с разными методами доступа к каналу, например

сети Ethernet (метод доступа CSMA/CD) и Token Ring (метод доступа TPMA). Направленность на объединение сетевых сегментов, имеющих различные физические среды, является основным отличием мостов от коммутаторов.

### Коммутаторы

*Коммутатор* (switch) – устройство, осуществляющее выбор одного из возможных вариантов направления передачи данных и является интеллектуальным развитием мостов.

В коммуникационной сети коммутатор является ретрансляционной системой (система, предназначенная для передачи данных или преобразования протоколов), обладающей свойством прозрачности (т.е. коммутация осуществляется здесь без какой-либо обработки данных). Коммутатор не имеет буферов и не может накапливать данные. Поэтому при использовании коммутатора скорости передачи сигналов в соединяемых каналах передачи данных должны быть одинаковыми. Канальные процессы, реализуемые коммутатором, выполняются специальными интегральными схемами.

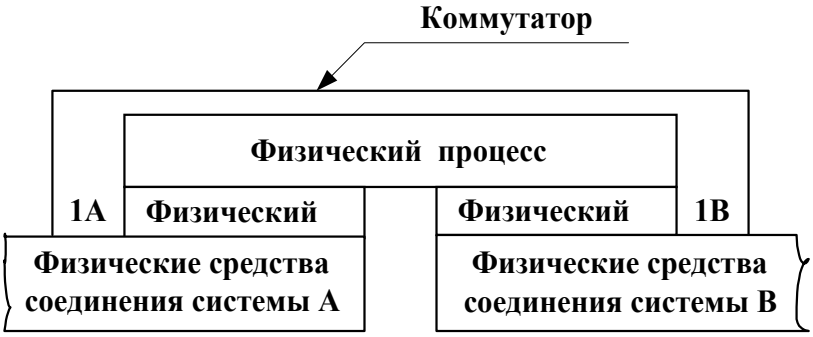


Рис. 9.5. Структура коммутатора

Особенностью коммутатора является то, что при поступлении кадра на какой-либо из портов 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. Таким образом, коммутатор (switch) может соединять серверы в кластер и служить основой для объединения нескольких рабочих групп. Он направляет пакеты данных между узлами ЛВС. Каждый коммутируемый сегмент получает доступ к каналу передачи данных без конкуренции и видит только тот трафик, который направляется в его сегмент. Коммутатор должен предоставлять каждому порту возможность соединения с максимальной скоростью без конкуренции со стороны других портов (в отличие от совместно используемого концентратора). Обычно в коммутаторах имеются один или два высокоскоростных порта, а также хорошие инструментальные средства управления. Коммутатором можно заменить маршрутизатор, дополнить им наращиваемый маршрутизатор или использовать коммутатор в качестве основы для соединения нескольких концентраторов.

## 9.4 Маршрутизаторы

*Маршрутизатор* (router) – ретрансляционная система, соединяющая две коммуникационные сети либо их части.

Каждый маршрутизатор реализует протоколы *физического* (1А, 1В), *канального* (2А, 2В) и *сетевого* (3А, 3В) уровней, как показано на рис.9.6. Специальные сетевые процессы соединяют части коммутатора в единое целое. Физический, канальный и сетевой протоколы в разных сетях различны. Поэтому соединение пар коммуникационных сетей осуществляется через маршрутизаторы, которые осуществляют необходимое преобразование указанных протоколов. Сетевые процессы выполняют взаимодействие соединяемых сетей.

Маршрутизатор работает с несколькими каналами, направляя в какой-нибудь из них очередной блок данных.

Маршрутизаторы обмениваются информацией об изменениях структуры сетей, трафике и их состоянии. Благодаря этому, выбирается оптимальный маршрут следования блока данных в разных сетях от абонентской системы-отправителя к системе-получателю. Маршрутизаторы обеспечивают также соединение административно независимых коммуникационных сетей.

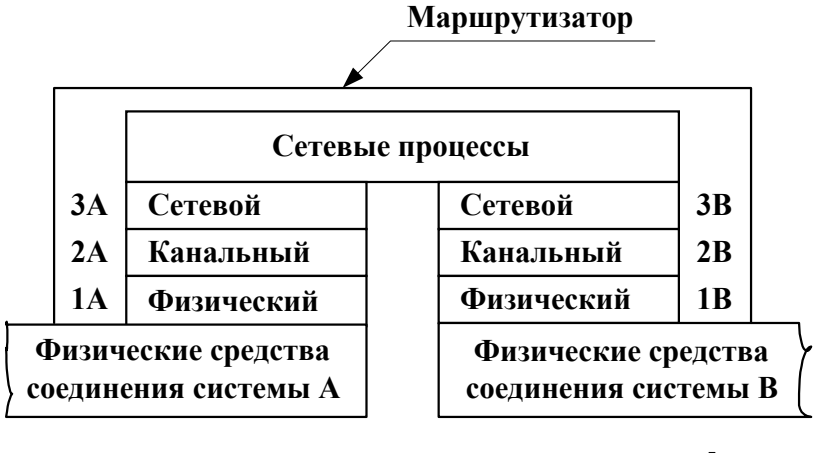


Рис. 9.6 Структура маршрутизатора

Маршрутизаторы превосходят мосты и коммутаторы своей способностью фильтровать и направлять пакеты данных в сети. Так как маршрутизаторы работают на сетевом уровне, они могут соединять сети, использующие разную сетевую архитектуру, методы доступа к каналам связи и протоколы.

Маршрутизаторы не обладают такой способностью к анализу сообщений как мосты, но зато могут принимать решение о выборе оптимального пути для данных между двумя сетевыми сегментами.

Мосты принимают решение по поводу адресации каждого из поступивших пакетов данных, переправлять его через мост или нет в зависимости от адреса назначения. Маршрутизаторы же выбирают из таблицы маршрутов наилучший для данного пакета.

В поле зрения маршрутизаторов находятся только пакеты, адресованные к ним предыдущими маршрутизаторами, в то время как мосты должны обрабатывать все пакеты сообщений в сегменте сети, к которому они подключены.

Тип топологии или протокола уровня доступа к сети не имеет значения для маршрутизаторов, так как они работают на уровень выше, чем мосты (сетевой уровень модели OSI). Маршрутизаторы часто используются для связи между сегментами с одинаковыми протоколами высокого уровня. Наиболее распространенным транспортным протоколом, который используют маршрутизаторы, является IPX фирмы Novell или TCP фирмы Microsoft.

Необходимо помнить, что для работы маршрутизаторов требуется один и тот же протокол во всех сегментах, с которыми он связан. При связывании сетей с различными протоколами лучше использовать мосты. Для управления загруженностью трафика сегмента сети также можно использовать мосты.

## 9.5 Шлюзы

*Шлюз* (gateway) – ретрансляционная система, обеспечивающая взаимодействие информационных сетей.

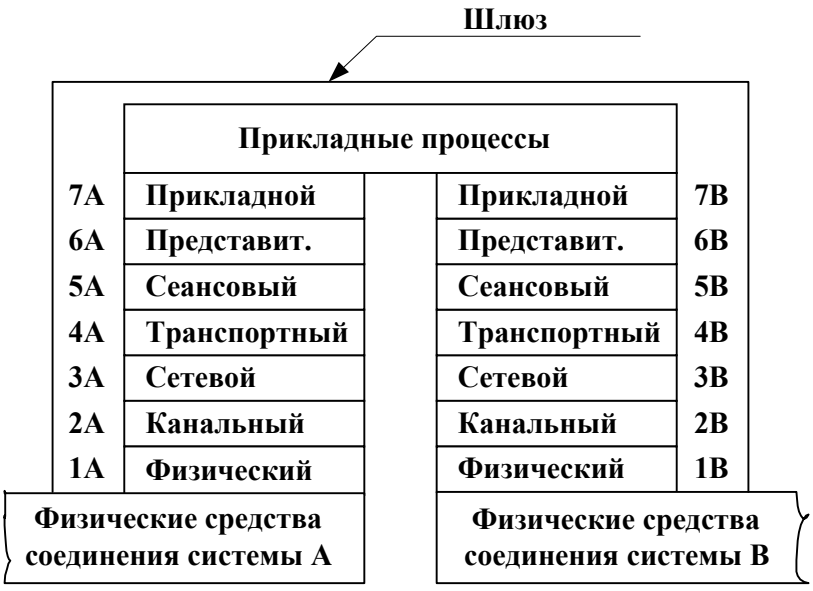


Рис. 9.7 Структура шлюза

Шлюз является наиболее сложной ретрансляционной системой, обеспечивающей взаимодействие сетей с различными наборами протоколов всех семи уровней. В свою очередь, наборы протоколов могут опираться на различные типы физических средств соединения.

В тех случаях, когда соединяются информационные сети, то в них часть уровней может иметь одни и те же протоколы. Тогда сети соединяются не

при помощи шлюза, а на основе более простых ретрансляционных систем, именуемых маршрутизаторами и мостами.

Шлюзы оперируют на верхних уровнях модели OSI (сеансовом, представительском и прикладном) и представляют наиболее развитый метод подсоединения сетевых сегментов и компьютерных сетей. Необходимость в сетевых шлюзах возникает при объединении двух систем, имеющих различную архитектуру. Например, шлюз приходится использовать для соединения сети с протоколом TCP/IP и большой ЭВМ со стандартом SNA. Эти две архитектуры не имеют ничего общего, и потому требуется полностью переводить весь поток данных, проходящих между двумя системами.

В качестве шлюза обычно используется выделенный компьютер, на котором запущено программное обеспечение шлюза и производятся преобразования, позволяющие взаимодействовать нескольким системам в сети. Другой функцией шлюзов является преобразование протоколов. При получении сообщения IPX/SPX для клиента TCP/IP шлюз преобразует сообщения в протокол TCP/IP.

### Вопросы по теме

1. Назначение сетевого адаптера.
2. В чем заключается настройка сетевого адаптера?
3. Перечислить функции сетевых адаптеров.
4. Какие типы сетевых адаптеров наиболее распространены?
5. На каком уровне сетевой модели OSI используется сетевой адаптер?
6. Каково назначение повторителя?
7. Что такое сетевой концентратор и каково его назначение?
8. На каком уровне сетевой модели OSI используется Hub?
9. Назначение моста.
10. На каком уровне сетевой модели OSI используется мост?
11. Какие сегменты сети может соединять мост?
12. Назначение коммутатора.
13. На каком уровне сетевой модели OSI используется коммутатор?
14. Каково различие между мостом и коммутатором?
15. Назначение маршрутизатора.
16. На каком уровне сетевой модели OSI используется маршрутизатор?
17. Каково различие между маршрутизаторами и мостами?
18. Что такое шлюз и каково его назначение.
19. На каком уровне сетевой модели OSI используется шлюз?

# 10 Требования, предъявляемые к сетям

При организации и эксплуатации сети необходимо учитывать следующие требованиями:

– производительность;
– надежность и безопасность;
– расширяемость и масштабируемость;
– прозрачность;
– поддержка разных видов трафика;
– управляемость;
– совместимость.

## 10.1 Производительность

*Производительность* – это характеристика сети, позволяющая оценить, насколько быстро информация передающей рабочей станции достигнет принимающей рабочей станции.

На производительность сети оказывают влияние следующие характеристики:

– конфигурация;
– скорость передачи данных;
– метод доступа к каналу;
– топология сети;
– технология.

Повысить производительность сети можно следующими образом:

– изменить конфигурацию сети таким образом, чтобы структура сети более соответствовала структуре информационных потоков;
– перейти к другой модели построения распределенных приложений, которая позволила бы уменьшить сетевой трафик;
– заменить мосты более скоростными коммутаторами;
– реконфигурировать сеть с целью микросегментации, которая позволяет уменьшить число пользователей на один сегмент и снизить объем широковещательного трафика, а значит, повысить производительность сети;
– перейти на более скоростную технологию.

## 10.2 Надежность и безопасность

Важнейшей характеристикой вычислительных сетей является *надежность*. Повышение надежности основано на принципе предотвращения неисправностей путем снижения интенсивности отказов и сбоев за счет применения электронных схем и компонентов с высокой и сверхвысокой степенью интеграции, снижения уровня помех, облегченных режимов работы схем,

обеспечение тепловых режимов их работы, а также за счет совершенствования методов сборки аппаратуры.

*Отказоустойчивость* – это такое свойство вычислительной системы, которое обеспечивает ей как логической машине возможность продолжения действий, заданных программой, после возникновения неисправностей. Введение отказоустойчивости требует избыточного аппаратного и программного обеспечения. Направления, связанные с предотвращением неисправностей и отказоустойчивостью, основные в проблеме надежности. На параллельных вычислительных системах достигается как наиболее высокая производительность, так и, во многих случаях, очень высокая надежность. Имеющиеся ресурсы избыточности в параллельных системах могут гибко использоваться как для повышения производительности, так и для повышения надежности.

Следует помнить, что понятие надежности включает не только аппаратные средства, но и программное обеспечение. Главной целью повышения надежности систем является целостность хранимых в них данных.

*Безопасность* – одна из основных задач, решаемых любой компьютерной сетью. Проблему безопасности можно рассматривать с разных сторон – злонамеренная порча данных, конфиденциальность информации, несанкционированный доступ, хищения и т.п. Методы решения данных проблем представлены на рис. 10.1.

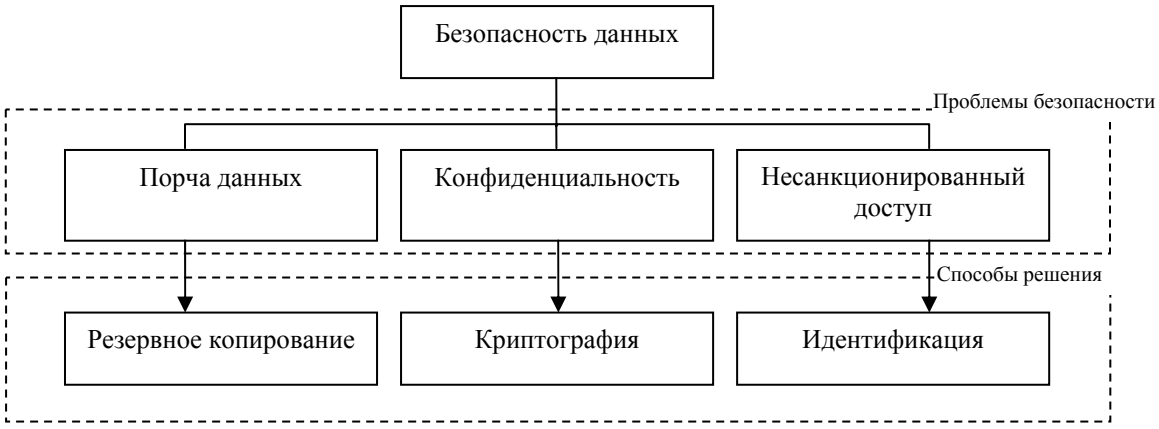


Рис. 10.1. Проблемы безопасности в сетях и способы их решения

## 10.3 Прозрачность

*Прозрачность* – это такое состояние сети, когда пользователь, работая в сети, не видит ее, т.е. работает в информационной сети так же как если бы он работал на локальной ЭВМ.

Коммуникационная сеть является прозрачной относительно проходящей сквозь нее информации, если выходной поток битов, в точности повторяет входной поток. Но сеть может быть непрозрачной во времени, если из-за меняющихся размеров очередей блоков данных изменяется и время прохождения различных блоков через узлы коммутации. Прозрачность сети по скоро-

сти передачи данных указывает, что данные можно передавать с любой нужной скоростью.

Если в сети по одним и тем же маршрутам передаются информационные и управляющие (синхронизирующие) сигналы, то говорят, что сеть прозрачна по отношению к типам сигналов.

Если передаваемая информация может кодироваться любым способом, то это означает, что сеть прозрачна для любых методов кодировок.

*Прозрачная сеть* является простым решением, в котором для взаимодействия локальных сетей, расположенных на значительном расстоянии друг от друга, используется принцип *Plug-and-play*.

*Прозрачное соединение.* Служба *прозрачных* локальных сетей обеспечивает сквозное (end-to-end) соединение, связывающее между собой удаленные локальные сети. Привлекательность данного решения состоит в том, что эта служба объединяет удаленные друг от друга на значительное расстояние узлы как части локальной сети. Поэтому не нужно вкладывать средства в изучение новых технологий и создание территориально распределенных сетей (Wide-Area Network – WAN). Пользователям требуется только поддерживать локальное соединение, а провайдер службы прозрачных сетей обеспечит беспрепятственное взаимодействие узлов через сеть масштаба города (Metropolitan-Area Network – MAN) или сеть WAN. Службы *Прозрачной* локальной сети имеют много преимуществ. Например, пользователь может быстро и безопасно передавать большие объемы данных на значительные расстояния, не обременяя себя сложностями, связанными с работой в сетях WAN.

### 10.4 Поддержка разных видов трафика

Трафик в сети складывается случайным образом, однако в нем отражены и некоторые закономерности. Как правило, некоторые пользователи, работающие над общей задачей, (например, сотрудники одного отдела), чаще всего обращаются с запросами либо друг к другу, либо к общему серверу, и только иногда они испытывают необходимость доступа к ресурсам компьютеров другого отдела. Желательно, чтобы структура сети соответствовала структуре информационных потоков. В зависимости от сетевого трафика компьютеры в сети могут быть разделены на группы (сегменты сети). Компьютеры объединяются в группу, если большая часть порождаемых ими сообщений, адресована компьютерам этой же группы.

Для разделения сети на сегменты используются мосты и коммутаторы. Они экранируют локальный трафик внутри сегмента, не передавая за его пределы никаких кадров, кроме тех, которые адресованы компьютерам, находящимся в других сегментах. Таким образом, сеть распадается на отдельные подсети. Это позволяет более рационально выбирать пропускную способность имеющихся линий связи, учитывая интенсивность трафика внутри каждой группы, а также активность обмена данными между группами.

Однако локализация трафика средствами мостов и коммутаторов имеет существенные ограничения. Использование механизма виртуальных сегментов, реализованного в коммутаторах локальных сетей, приводит к полной локализации трафика; такие сегменты полностью изолированы друг от друга, даже в отношении широковещательных кадров. Поэтому в сетях, построенных только на мостах и коммутаторах, компьютеры, принадлежащие разным виртуальным сегментам, не образуют единой сети.

Для того чтобы эффективно консолидировать различные виды трафика в сети ATM, требуется специальная предварительная подготовка (адаптация) данных, имеющих различный характер: кадры – для цифровых данных, сигналы импульсно-кодовой модуляции – для голоса, потоки битов – для видео. Эффективная консолидация трафика требует также учета и использования статистических вариаций интенсивности различных типов трафика.

### 10.5 Управляемость

ISO внесла большой вклад в стандартизацию сетей. Модель управления сети является основным средством для понимания главных функций систем управления сети. Эта модель состоит из 5 концептуальных областей:

- управление эффективностью;
- управление конфигурацией;
- управление учетом использования ресурсов;
- управление неисправностями;
- управление защитой данных.

#### Управление эффективностью

Цель управления эффективностью – измерение и обеспечение различных аспектов эффективности сети для того, чтобы межсетевая эффективность могла поддерживаться на приемлемом уровне. Примерами переменных эффективности, которые могли бы быть обеспечены, являются пропускная способность сети, время реакции пользователей и коэффициент использования линии.

Управление эффективностью включает несколько этапов:

1. сбор информации об эффективности по тем переменным, которые представляют интерес для администраторов сети;
2. анализ информации для определения нормальных (базовая строка) уровней;
3. определение соответствующих порогов эффективности для каждой важной переменной таким образом, что превышение этих порогов указывает на наличие проблемы в сети, достойной внимания.

**Управление конфигурацией**

Цель управления конфигурацией – контролирование информации о сетевой и системной конфигурации для того, чтобы можно было отслеживать и управлять воздействием на работу сети различных версий аппаратных и программных элементов. Т.к. все аппаратные и программные элементы имеют эксплуатационные отклонения, погрешности (или то и другое вместе), которые могут влиять на работу сети, такая информация важна для поддержания гладкой работы сети.

Каждое устройство сети располагает разнообразной информацией о версиях, ассоциируемых с ним. Чтобы обеспечить легкий доступ, подсистемы управления конфигурацией хранят эту информацию в базе данных. Когда возникает какая-нибудь проблема, в этой базе данных может быть проведен поиск ключей, которые могли бы помочь решить эту проблему.

**Управление учетом использования ресурсов**

Цель управления учетом использования ресурсов – измерение параметров использования сети, чтобы можно было соответствующим образом регулировать ее использование индивидуальными или групповыми пользователями. Такое регулирование минимизирует число проблем в сети (т.к. ресурсы сети могут быть поделены исходя из возможностей источника) и максимизирует равнодоступность к сети для всех пользователей.

**Управление неисправностями**

Цель управления неисправностями – выявить, зафиксировать, уведомить пользователей и (в пределах возможного) автоматически устранить проблемы в сети, с тем чтобы эффективно поддерживать работу сети. Так как неисправности могут привести к простоям или недопустимой деградации сети, управление неисправностями, по всей вероятности, является наиболее широко используемым элементом модели управления сети ISO.

Управление неисправностями включает в себя несколько этапов:

1. определение симптомов проблемы;
2. изолирование проблемы;
3. устранение проблемы;
4. проверка устранения неисправности на всех важных подсистемах;
5. регистрация обнаружения проблемы и ее решения.

**Управление защитой данных**

Цель управления защитой данных – контроль доступа к сетевым ресурсам в соответствии с местными руководящими принципами, чтобы сделать невозможными саботаж сети и доступ к чувствительной информации лицам, не имеющим соответствующего разрешения. Например, одна из подсистем управления защитой данных может контролировать регистрацию пользова-

телей ресурса сети, отказывая в доступе тем, кто вводит коды доступа, не соответствующие установленным.

Подсистемы управления защитой данных работают путем разделения источников на санкционированные и несанкционированные области. Для некоторых пользователей доступ к любому источнику сети является несоответствующим.

Подсистемы управления защитой данных выполняют следующие функции:

- идентифицируют чувствительные ресурсы сети (включая системы, файлы и другие объекты);
- определяют отображения в виде карт между чувствительными источниками сети и набором пользователей;
- контролируют точки доступа к чувствительным ресурсам сети;
- регистрируют несоответствующий доступ к чувствительным ресурсам сети.

### 10.6 Совместимость

*Совместимость и мобильность программного обеспечения.* Концепция программной совместимости впервые в широких масштабах была применена разработчиками системы IBM360. Основная задача при проектировании всего ряда моделей этой системы заключалась в создании такой архитектуры, которая была бы одинаковой с точки зрения пользователя для всех моделей системы независимо от цены и производительности каждой из них. Огромные преимущества такого подхода, позволяющего сохранять существующий задел программного обеспечения при переходе на новые (как правило, более производительные) модели, были быстро оценены как производителями компьютеров, так и пользователями, и начиная с этого времени практически все фирмы-поставщики компьютерного оборудования взяли на вооружение эти принципы, поставляя серии совместимых компьютеров. Следует заметить однако, что со временем даже самая передовая архитектура неизбежно устаревает и возникает потребность внесения радикальных изменений в архитектуру и способы организации вычислительных систем.

В настоящее время одним из наиболее важных факторов, определяющих современные тенденции в развитии информационных технологий, является ориентация компаний-поставщиков компьютерного оборудования на рынок прикладных программных средств.

Этот переход выдвинул ряд новых требований. Прежде всего, такая вычислительная среда должна позволять гибко менять количество и состав аппаратных средств и программного обеспечения в соответствии с меняющимися требованиями решаемых задач. Во-вторых, она должна обеспечивать возможность запуска одних и тех же программных систем на различных аппаратных платформах, т.е. обеспечивать мобильность программного обеспе-

чения. В-третьих, эта среда должна гарантировать возможность применения одних и тех же человеко-машинных интерфейсов на всех компьютерах, входящих в неоднородную сеть. В условиях жесткой конкуренции производителей аппаратных платформ и программного обеспечения сформировалась концепция открытых систем, представляющая собой совокупность стандартов на различные компоненты вычислительной среды, предназначенных для обеспечения мобильности программных средств в рамках неоднородной, распределенной вычислительной системы.

### Вопросы по теме

1. Какие основные требования предъявляются к сетям?
2. Что такое производительность сети?
3. Какие характеристики влияют на производительность сети?
4. Какие есть способы повышения производительности сетей?
5. Чем обеспечивается надежность сети?
6. Что такое отказоустойчивость?
7. Перечислить проблемы безопасности данных в сети и способы их решения.
8. Что такое прозрачность сетей?
9. В каком случае линия прозрачна по отношению к типам сигналов?
10. Что такое прозрачное соединение?
11. Что используется для разделения сети на сегменты?
12. Каким образом можно уменьшить трафик в сети?
13. Дать определение управляемости сетей и перечислить основные функции управления сетями.
14. Что включается в управление эффективностью?
15. Для какой цели используется управление неисправностями?
16. Для чего необходимо управление конфигурацией?
17. Какова цель управления защитой данных?
18. Какие функции подсистемы управления защитой данных?
19. Дать определение понятия совместимости сетей.

## Лабораторные работы по курсу «Информационные сети»

Каждая из представленных ниже лабораторных работ рассчитана на четыре аудиторных часа. В ходе выполнения лабораторной работы студент должен выполнить предложенное задание и подготовить отчет о проделанной работе. Форма сдачи лабораторной работы предполагает демонстрацию выполненного задания (программного продукта) и знаний теоретической части вопроса, рассмотренного в лабораторной работе.

Отчет по лабораторным работам должен содержать:

- тему и цель лабораторной работы;
- вариант задания на лабораторную работу;
- краткие теоретические сведения и описание алгоритма работы программы;
- листинг разработанной программы с подробными комментариями;
- результаты работы программы;
- выводы.

## Лабораторная работа №1 «Знакомство с сетевым протоколом FTP»

### Цель работы

Изучить функции и основные команды сетевого протокола FTP.

### Теоретические сведения

В информационных сетях есть много способов передачи информации с удаленной ЭВМ на локальную. Задача данной лабораторной работы – ознакомиться и научиться одному из них, использующему FTP – File Transfer Protocol. Главное назначение FTP – это пересылка файлов. FTP можно использовать самостоятельно, а также через другие системы, например, WWW использует FTP как часть своего протокола.

FTP-серверы рассредоточены по глобальной вычислительной сети, но для соединения с ними не требуется знания их физического расположения, достаточно знания лишь их адреса. Например, FTP-сервер фирмы Borland имеет адрес ftp.borland.com.

Если адрес FTP-сервера известен, то соединения с ним может использоваться специальная программа, которая называется FTP-клиент. Традиционных FTP-клиентом является командная строка со стандартным набором команд. В настоящее время популярность пользуются FTP-клиенты, имеющий оконный интерфейс и не требующие знания не синтаксиса FTP-команд. Однако и в их основе лежит стандартная система команд FTP.

### Замечания по выполнению лабораторной работы

1. Все операции должны выполняться без использования файловых оболочек типа FAR-manager или Total Commander.
2. Результаты выполнения каждой операции необходимо проконтролировать командой вывода содержимого каталога на экран и отобразить в отчете.
3. Для получения списка доступных команд FTP-клиента необходимо использовать команду *help.*

4. Для получения формата команды необходимо использовать *help <имя команды>*
5. Дополнительная информация по протоколу FTP и другим протоколам может быть получена в справочной литературе по адресам:

   *\\is-serv\docs\сети\руководство по ftp*

   *\\is-serv\docs\сети\protocols*

**Задание на лабораторную работу**

1. Ознакомиться с форматом и параметрами команд протокола FTP.
2. Создать в каталоге группы рабочий каталог на локальной ЭВМ (локальный рабочий каталог).
3. Войти на сервер FTP кафедры (*is-serv.is.mivlgu.ru*) под именем *anonymous* (пароль пустой).
4. Включить выдачу звуковых сигналов после выполнения команд.
5. Создать каталог на FTP-сервере в соответствии с номером своей рабочей станции (удаленный рабочий каталог).
6. Получить из корневого каталога FTP-сервера группу файлов с расширением *txt* и поместить их в локальный каталог.
7. Поместить эти файлы в удаленный рабочий каталог.
8. Изменить расширение файлов, соответствующих маске **d??.txt* на расширение *prn*.
9. В удаленном рабочем каталоге все файлы с расширением *txt* удалить.
10. Изменить режим передачи фалов на двоичный (binary).
11. Сменить локальный рабочий каталог на другой и произвести копирование на FTP-сервер нескольких небольших файлов.
12. Произвести двукратное добавление в файл *<имя_рабочей_станции.txt>*, расположенный в удаленном рабочем каталоге, информации из любого другого, расположенного в локальном рабочем каталоге.
13. Получить созданный файл, разместив его в локальном рабочем каталоге.
14. Войти на FTP-сервер под именем ftp (пароль ftp).
15. Удалить все файлы в удаленном рабочем каталоге.
16. Удалить удаленный рабочий каталог.
17. Разорвать соединение с FTP-сервером.
18. Удалить локальный рабочий каталог.
19. Подготовить отчет о лабораторной работе.

## Лабораторная работа №2 «Сетевое программирование на базе Sockets»

### Цель работы

Получить навыки разработки сетевых приложений, используя механизм Windows Sockets.

### Теоретические сведения

*Работа с компонентами TClientSocket и TServerSocket*

Сокеты – это интерфейс прикладного программирования для сетевых приложений TCP/IP, которые используют сокеты, как виртуальные разъемы для обмена данными между собой.

Сокеты бывают трех видов: клиентские, слушающие и серверные. Клиентские сокеты устанавливают связь с сервером и обмениваются с ним данными. Клиентский сокет включен в компонент *TClientSocket*. Слушающий сокет принимает запрос на соединение от клиентского сокета, и соединяет сервер с клиентом. Слушающий сокет содержится в компоненте *TServerSocket*. Серверный сокет обменивается данными с клиентом по уже установленному соединению.

Ниже рассмотрено создание приложений с архитектурой клиент-сервер в *Borland Delphi* на основе сокетов. Следует заметить, что для создания отдельных приложений клиента и сервера нет необходимости иметь несколько ЭВМ. На одной ЭВМ можно одновременно запускать и приложение-сервер и приложение-клиент. При этом нужно в качестве адреса или имени ЭВМ, к которой будет производиться подключение следует использовать имя *localhost* или IP-адрес *127.0.0.1*.

Сервер, основанный на сокетном протоколе, позволяет обслуживать сразу множество клиентов. Причем, ограничение на их количество можно устанавливать при программировании. Для каждого подключенного клиента сервер открывает отдельный сокет, по которому осуществляется обмен данными с клиентом.

Алгоритм управления сокетным сервером состоит из следующих этапов:

1. *установка свойств Port (порт подключения) и ServerType (тип подключения)* – для нормального взаимодействия сервера и клиента необходимо, чтобы порт, используемый сервером точно совпадал с портом, используемым клиентом (и наоборот);
2. *открытие сокета и указанного порта* – после выполнения указанной операции выполняется автоматическое начало ожидания подсоединения клиентов (Listen);
3. *подключение клиента и обмен данными с ним*;
4. *отключение клиента*;

5. *закрытие сервера и сокета* – по команде администратора сервер завершает свою работу, закрывая все открытые сокетные каналы и прекращая ожидание подключений клиентов.

Пункты 3-4 приведенного алгоритмы выполняются многократно, т.е. для каждого нового подключения клиента.

*Свойства и методы компонента TServerSocket*

Ниже перечислены основные свойства компонента *TServerSocket*:

1. *Socket* – класс *TServerWinSocket*, через который осуществляется доступ к открытым сокетным каналам. Тип: *TServerWinSocket*;
2. *ServerType* – тип сервера, указанное свойство может принимать одно из двух значений: *stNonBlocking* – синхронная работа с клиентскими сокетами (в этом случае работа с клиентами осуществляется через события *OnClientRead* и *OnClientWrite*) или *stThreadBlocking* – асинхронный тип (в этом случае для каждого клиентского сокетного канала создается отдельный процесс (*Thread*)). Тип: *TServerType*;
3. *ThreadCacheSize* – количество клиентских процессов (*Thread*), которые будут кэшироваться сервером. Здесь необходимо подбирать среднее значение в зависимости от загруженности сервера. Кэширование происходит для того, чтобы не создавать каждый раз отдельный процесс и не разрушать закрытый сокет, а оставлять их для дальнейшего использования. Тип: *Integer*;
4. *Active* – показатель того, активен в данных момент сервер, или нет, т.е. значение *True* переводит сервер в активное состояние. Тип: *Boolean*;
5. *Port* – номер порта для установления соединений с клиентами. Как уже было отмечено выше порт сервера и клиента должен совпадать. Рекомендуются использовать значения от 1025 до 65535, т.к. значения от 1 до 1024 – могут быть заняты системой. Тип: *Integer*;
6. *Service* – строка, определяющая службу (*ftp*, *http*, *pop*, и т.д.), порт которой будет использован. Это своеобразный справочник соответствия номеров портов различным стандартным протоколам. Тип: *string*.

Ниже перечислены основные методы компонента *TServerSocket*:

- *Open* – запускает сервер (идентична присвоению *True* свойству *Active*);
- *Close* – останавливает сервер. (идентична присвоению *False* свойству *Active*).

Ниже перечислены основные события компонента *TServerSocket*:

1. *OnClientConnect* – возникает, когда клиент установил сокетное соединение и ждет ответа сервера (*OnAccept*);
2. *OnClientDisconnect* – возникает, когда клиент отсоединился от сокетного канала;

3. *OnClientError* – возникает, когда текущая операция завершилась неудачно, т.е. произошла ошибка;
4. *OnClientRead* – возникает, когда клиент передал серверу какие-либо данные. Доступ к этим данным можно получить через передаваемый параметр *Socket*: *TCustomWinSocket*;
5. *OnClientWrite* – возникает, когда сервер может отправлять данные клиенту по сокету;
6. *OnGetSocket* – в обработчике этого события можно провести редактирование параметра *ClientSocket*;
7. *OnGetThread* – в обработчике этого события можно определить уникальный процесс (*Thread*) для каждого отдельного клиентского канала, присвоив параметру *SocketThread* нужную подзадачу *TServerClientThread*;
8. *OnThreadStart, OnThreadEnd* – возникает, когда подзадача (процесс, *Thread*) запускается или останавливается, соответственно;
9. *OnAccept* – возникает, когда сервер принимает клиента или отказывает ему в соединении;
10. *OnListen* – возникает, когда сервер переходит в режим ожидания подсоединения клиентов.

Таким образом, если работа с клиентом осуществляется через события *OnClientRead* и *OnClientWrite*, то передавать данные клиенту можно через параметр *ClientSocket* (*TCustomWinSocket*).

Следует также отметить несколько полезных свойств и методов *TServerSocket.Socket*:

- *ActiveConnections* (*Integer*) – количество подключенных клиентов;
- *ActiveThreads* (*Integer*) – количество работающих процессов;
- *Connections* (*Array*) – массив, состоящий из отдельных классов *TClientWinSocket* для каждого подключенного клиента. Например, такая команда *ServerSocket1.Socket.Connections[0].SendText('Hello!')* отсылает первому подключенному клиенту сообщение *'Hello!'*. Команды для работы с элементами этого массива – это (*Send/Receive*)(*Text,Buffer, Stream*);
- *IdleThreads* (*Integer*) – количество свободных процессов, такие процессы кэшируются сервером (см. *ThreadCacheSize*);
- *LocalAddress*, *LocalHost*, *LocalPort* – соответственно локальный IP-адрес, хост-имя и порт;
- *RemoteAddress*, *RemoteHost*, *RemotePort* – соответственно удаленный IP-адрес, хост-имя и порт;
- методы *Lock* и *UnLock* – соответственно, блокировка и разблокировка сокета.

*Примеры использования сокетов*

Ниже приведен пример программы, использующей *TServerSocket* и осуществляющей протоколирование всех важных событий сервера, а также имеющей возможность принимать и отсылать текстовые сообщения.

```
{Заголовок файла, определение TForm1 и ее экземпляра Form1}
procedure TForm1.Button1Click(Sender: TObject);
begin
  {Определяем порт и запускаем сервер}
  ServerSocket1.Port := 1025;
  {Метод Insert вставляет строку в массив в указанную позицию}
  Memo2.Lines.Insert(0,'Server starting');
  ServerSocket1.Open;
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
  {Останавливаем сервер}
  ServerSocket1.Active := False;
  Memo2.Lines.Insert(0,'Server stopped');
end;

procedure TForm1.ServerSocket1Listen(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  {Здесь сервер "прослушивает" сокет на наличие клиентов}
  Memo2.Lines.Insert(0,'Listening on port '+IntToStr(ServerSocket1.Port));
end;

procedure TForm1.ServerSocket1Accept(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  {Здесь сервер принимает клиента}
  Memo2.Lines.Insert(0,'Client connection accepted');
end;

procedure TForm1.ServerSocket1ClientConnect(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  {Здесь клиент подсоединяется}
  Memo2.Lines.Insert(0,'Client connected');
end;

procedure TForm1.ServerSocket1ClientDisconnect(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  {Здесь клиент отсоединяется}
  Memo2.Lines.Insert(0,'Client disconnected');
end;

procedure TForm1.ServerSocket1ClientError(Sender: TObject;
  Socket: TCustomWinSocket; ErrorEvent: TErrorEvent;
  var ErrorCode: Integer);
begin
  {Произошла ошибка - выводим ее код}
  Memo2.Lines.Insert(0,'Client error. Code = '+IntToStr(ErrorCode));
end;

procedure TForm1.ServerSocket1ClientRead(Sender: TObject;
  Socket: TCustomWinSocket);
```

```
begin
  {От клиента получено сообщение - выводим его в Memo1}
  Memo2.Lines.Insert(0,'Message received from client');
  Memo1.Lines.Insert(0,'> '+Socket.ReceiveText);
end;

procedure TForm1.ServerSocket1ClientWrite(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  {Теперь можно слать данные в сокет}
  Memo2.Lines.Insert(0,'Now can write to socket');
end;

procedure TForm1.ServerSocket1GetSocket(Sender: TObject; Socket: Integer;
  var ClientSocket: TServerClientWinSocket);
begin
  Memo2.Lines.Insert(0,'Get socket');
end;

procedure TForm1.ServerSocket1GetThread(Sender: TObject;
  ClientSocket: TServerClientWinSocket;
  var SocketThread: TServerClientThread);
begin
  Memo2.Lines.Insert(0,'Get Thread');
end;

procedure TForm1.ServerSocket1ThreadEnd(Sender: TObject;
  Thread: TServerClientThread);
begin
  Memo2.Lines.Insert(0,'Thread end');
end;

procedure TForm1.ServerSocket1ThreadStart(Sender: TObject;
  Thread: TServerClientThread);
begin
  Memo2.Lines.Insert(0,'Thread start');
end;

procedure TForm1.Button3Click(Sender: TObject);
 var i: Integer;
begin
  {Посылаем ВСЕМ клиентам сообщение из Edit1}
  for i := 0 to ServerSocket1.Socket.ActiveConnections-1 do begin
   ServerSocket1.Socket.Connections[i].SendText(Edit1.Text);
  end;
  Memo1.Lines.Insert(0,'< '+Edit1.Text);
end;
```

Нередко возникает необходимость хранить какую-либо информацию для каждого клиента (имя, IP-адрес и т.д.), причем с привязкой этой информации к сокету данного клиента. Для этих целей каждый сокет обладает свойством *Data*. *Data* – это указатель, поэтому, записывая данные клиента в это свойство необходимо следовать правилам работы с указателями (выделение памяти, определение типа, и т.д.).

Ниже приведен пример программы, иллюстрирующей посылку файлов через сокет.

```
{Посылка файла через сокет}
procedure SendFileBySocket(filename: string);
```

```
 var srcfile: TFileStream;
begin
  {Открываем файл filename}
  srcfile := TFileStream.Create(filename,fmOpenRead);
  {Посылаем его первому подключенному клиенту}
  ServerSocket1.Socket.Connections[0].SendStream(srcfile);
  {Закрываем файл}
  srcfile.Free;
end;
```

Нужно заметить, что метод *SendStream* может используется не только сервером, но и клиентом (например, *ClientSocket1.Socket.SendStream(srcfile)*).

Следует учитывать, что при передаче через сокет данных они могут объединяться в один блок или разбиваться по нескольким блокам, что связано с особенностями функционирования сети. Например, если отправить через сокет файл, размером в 100 Кб, то клиентскому приложению придет несколько блоков с размерами, которые зависят от трафика и загруженности линии. Причем, размеры не обязательно будут одинаковыми. Отсюда следует, что для того, чтобы принять файл или любые другие данные большого размера, необходимо принимать блоки данных, а затем объединять их в одно целое. Решением данной задачи является файловый поток – *TFileStream* (либо поток в памяти – *TMemoryStream*). Принимать частички данных из сокета можно через событие *OnRead* (*OnClientRead*), используя универсальный метод *ReceiveBuf*. Определить размер полученного блока можно методом *ReceiveLength*. Ниже приведен пример, иллюстрирующий прием файла через сокет.

```
{Прием файла через сокет}
procedure TForm1.ClientSocket1Read(Sender: TObject;
  Socket: TCustomWinSocket);
 var l: Integer;
     buf: PChar;
     src: TFileStream;
begin
  {Записываем в l размер полученного блока}
  l := Socket.ReceiveLength;
  {Заказываем память для буфера}
  GetMem(buf,l+1);
  {Записываем в буфер полученный блок}
  Socket.ReceiveBuf(buf,l);
  {Открываем временный файл для записи}
  src := TFileStream.Create('myfile.tmp',fmOpenReadWrite);
  {Ставим позицию в конец файла}
  src.Seek(0,soFromEnd);
  {Записываем буфер в файл}
  src.WriteBuffer(buf,l);
  {Закрываем файл}
  src.Free;
  {Освобождаем память}
  FreeMem(buf);
end;
```

### Замечания по выполнению лабораторной работы

Дополнительную информацию по использованию компонентов *TServerSocket* и *TClientSocket* можно получить в справочной информации к среде визуального программирования Borland Delphi в разделе *Writing Distributed Applications – Working With Sockets – Using Sockets Components*.

### Задание на лабораторную работу

Разработать сетевое приложение в соответствии с вариантом задания. Подготовить отчет о проделанной работе.

| Вариант | Задание |
|---|---|
| 4. | Сетевое приложение для пересылки сообщений между рабочими станциями в сети. |
| 5. | Сетевое приложение "Крестики - нолики". |
| 6. | Сетевое приложение пересылки файлов между рабочими станциями. |
| 7. | Сетевое приложение поиска доступных IP адресов (аналог стандартного поиска компьютера в сети). Результат поиска - IP адрес компьютера и его разделяемые ресурсы. |
| 8. | Сетевое приложение для подключения и отключения сетевых ресурсов. |
| 9. | Сетевое приложение-игра «Четыре». |
| 10. | Программа получения информации об ЭВМ по ее IP – адресу (имя компьютера, имя пользователя, ОС, общие ресурсы и т.п.). |
| 11. | Приложение просмотра сетевых ресурсов (аналог Сетевого окружения). |
| 12. | Сетевое приложение «Чат» (предусмотреть отображение состояния: неактивен, активен, вышел из чата) |
| 13. | Разработать программу сетевого мультимедиа-проигрывателя. |

## Лабораторная работа №3 «Знакомство с библиотекой DirectPlay»

### Цель работы

Получение базовых навыков работы с библиотекой DirectPlay, входящей в состав DirectX.

### Теоретические сведения

*Сетевые подключения*

В настоящее время практически любая программа имеет возможность сетевого соединения, особую актуальность эта проблема приобретает при создании компьютерных тренажеров и игр (которые являются частным случае тренажеров).

Для поддержки сетевых игр в библиотеку *DirectX* был включен компонент *DirectPlay*, который предоставляет интерфейсы для сетевых соединений и возможности ведения игры по сети, освобождая разработчика от трудностей, которые возникают при создании сетевой игры [39, 40].

*Одноранговые соединения*

При одноранговых соединениях отсутствует как таковой сервер, хотя один из компьютеров объявляется как сервер имен. Он занимается тем, что отвечает за перечисление запросов, рассылает информацию о состоянии других игроков вновь подсоединившимся компьютерам, генерирует уникальные идентификационные номера для игроков или групп игроков. Сервер имен не выступает в роли обычного сервера и не управляет сообщениями, которые игроки посылают друг другу.

Каждый ПК, чтобы присоединиться к сеансу, должен послать соответствующий запрос серверу имен. При ответе ПК, сделавшему запрос, сервер имен посылает статус сеанса.

Компьютер, выбранный в качестве сервера имен, становится сервером сеанса. Только он может изменять данные о состоянии сеанса. При этом сетевой адрес сервера имен становится сетевым адресом сеанса. Если компьютер, являющийся сервером имен, по какой-либо причине покидает сеанс, в этом качестве может быть выбран любой другой компьютер. Когда происходит смена сервера имен, автоматически изменяется сетевой адрес сеанса.

Любой компьютер, присоединенный к сеансу, может получить информацию от сервера имен о его состоянии, отправить сообщение любому игроку или получить от него данные.

При одноранговом соединении каждое изменение статуса игрока должно рассылаться всем участникам. Количество одновременно пересылаемых данных может быть ограничено пропускной способностью сети (особенно при соединении через модем). Эти рамки, в свою очередь, накладывают ограничение на количество компьютеров, имеющих возможность присоединиться к сеансу. Выходов из этого положения два: сократить количество передаваемых данных или уменьшить частоту их передачи.

Каждое сообщение рассылается всем участникам сеанса. Если необходимо отправить сообщение не всем компьютерам, а только определенному ряду ПК, можно воспользоваться средствами, предоставляемыми сетевыми провайдерами. Рассылка сообщения нескольким определенным компьютерам называется *групповой рассылкой сообщений* (*multicasting*).

Рассылка групповых сообщений может существенно загрузить *сетевой трафик.* Для его снижения рекомендуется использовать, в дополнение к серверу имен, групповой сервер. Любой компьютер может отправить одно сообщение групповому серверу с указанием всех машин, которые должны его получить. Групповой сервер разошлет это сообщение указанным компьютерам. Также групповой сервер может сократить количество пакетов в сети, если вместо широковещательной рассылки поручить ему разослать сообщение всем компьютерам текущего сеанса. Данный механизм уменьшает трафик сети, особенно при использовании модемных подключений.

Создание группового сервера в дополнение к серверу имен влияет только на рассылку групповых и широковещательных сообщений, те пакеты, которыми компьютеры обмениваются непосредственно, серверами не маршрутизируются, а следовательно, не меняется концепция построения сетевого приложения.

*Соединение посредством сервера*

При *клиент-серверной* архитектуре построения сетевого приложения один из компьютеров объявляется *сервером.* Этот компьютер отвечает за открытие сеанса, присоединение к нему игроков, пересылку вновь подключившимся компьютерам информации о состоянии сеанса и игроков, присваивание ID-номеров игрокам и группам игроков. В дополнение к этому, все сообщения игроков пересылаются только через сервер.

Серверный компьютер должен запускать специальную версию приложения, которая будет следить за состоянием сеанса, хранить и обновлять состояния игроков, информировать участников об изменении состояний. Объект «игрок», создаваемый на сервере, называют серверным игроком. Серверный игрок ничем не отличается от любого другого игрока и каждый клиент сеанса может обмениваться с ним сообщениями.

Клиентские компьютеры должны запускать клиентскую версию приложения. Они могут получать от сервера сообщения об изменении статуса сеанса и извещать его о смене локального статуса.

Могут применяться два подхода к созданию приложений на основе сервера. Первый заключается в том, что происходит объединение одноранговое подключения с подключением на основе сервера. При этом всем клиентам доступны списки игроков и групп, каждый игрок может отправлять сообщения непосредственно другим игрокам и влиять на изменение их статуса. Ввиду того, что статус игроков может постоянно изменяться, количество сообщений, отправляемых каждому игроку, может быть очень большим, в результате чего клиентские компьютеры окажутся перегружены. Данный тип приложений не может поддерживать более шестнадцати игроков одновременно. Данное соединение отличается от одноранговое лишь тем, что все сообщения проходят через сервер и их копии получает серверный игрок.

Другой метод создания такого приложения является соединением непосредственно используя сервер. При этом клиентское приложение «видит» только серверного игрока и тех игроков, которых создало в течение данного сеанса. Игроки могут непосредственно обмениваться сообщениями только с серверным игроком. Серверное же приложение «видит» всех игроков. Получив сообщение об изменении статуса игрока, сервер должен самостоятельно обработать его, обновить статус сеанса и оповестить об этом всех игроков.

Так же как и в одноранговом соединении, сетевой адрес сервера является сетевым адресом сеанса. Компьютер, желающий присоединиться к сеансу, должен послать запрос серверу. Если сервер по какой-то причине выбывает

из сеанса, игра прекращается. Так как все сообщения обязаны пройти через сервер, он, в дополнение ко всему, является групповым сервером.

*Провайдеры*

Для сетевого соединения между компьютерами необходимы услуги *провайдеров*. В данном контексте провайдерами являются сетевые сервисы, обеспечивающие соединения компьютеров и скрывающие от пользователя все тонкости работы с сетью. В этом случае, приложению, использующему *DirectPlay*, необходимо выбрать провайдера.

Большинство провайдеров используют *сокеты Windows* (см. лабораторную работу №2).

Microsoft предоставляет четыре провайдера для DirectPlay: модемную связь, прямое последовательное соединение, соединение при помощи TCP/IP и соединение при помощи IPX.

*Провайдер TCP/IP*

Провайдер TCP/IP использует сокеты Windows для связи по локальной сети или глобальной сети Интернет. Он использует UDP-протокол для передачи пакетов без гарантии доставки и TPC/IP для пакетов с гарантированной доставкой.

В этом случае, при перечислении сеансов приложение при помощи диалогового окна запрашивает IP-адрес сеанса или DNS-имя ЭВМ. При работе в локальной сети пользователь может опустить адрес, при этом DirectPlay для поиска существующих сеансов разошлет широковещательное сообщение и в результате будут перечислены все сеансы в текущей подсети.

Как отмечалось ранее, протокол TCP/IP имеет порт. Номер порта может принимать любое значение от 1 до 65 535, но следует заметить, что диапазон от 1 до 1024 зарезервирован и используется для системных нужд. Все остальные порты предназначены для свободного использования. Когда *DirectPlay* перечисляет сеансы, протокол TCP/IP использует порт 47624. Для обмена данными служит диапазон портов от 2300 до 2400.

*Провайдер IPX*

Провайдер IPX использует сокеты Windows и протокол IPX для передачи пакетов исключительно по локальной сети. Данный протокол предоставляет только передачу пакетов без гарантии их доставки. Для передачи пакетов с гарантированной доставкой необходимо воспользоваться собственным протоколом *DirectPlay*.

Для перечисления сеансов провайдер IPX всегда использует широковещательную рассылку сообщений. Только после того, как сеанс будет открыт, компьютеры смогут напрямую обмениваться пакетами. Так же как и провайдер TCP/IP, провайдер IPX может перечислять сеансы, находящиеся только в текущей подсети.

*Системный провайдер модема*

Системный провайдер модема использует *TAPI* (Telephony API – программный интерфейс управлением телефоном) для соединения при помощи модемов. Данный провайдер предоставляет только передачу пакетов без гарантированной доставки. Для передачи пакетов с гарантированной доставкой необходимо воспользоваться собственным протоколом *DirectPlay*.

При открытии сеанса вызывается диалоговое окно с возможностью выбора конфигурации модема и установки модема в режим автоответа. При перечислении сеансов вызывается диалоговое окно с возможностью указать используемый модем и телефонный номер, необходимый для соединения. После того как вся эта информация будет получена, *DirectPlay* наберет указанный телефонный номер и попытается определить возможность подключения к сеансу.

*Провайдер серийного соединения*

Провайдер серийного соединения предназначен для установки связи между двумя компьютерами посредством серийного порта или при помощи модемов. Данный провайдер предоставляет только передачу пакетов без гарантированной доставки. Для передачи пакетов с гарантированной доставкой необходимо воспользоваться собственным протоколом *DirectPlay*.

При создании или перечислении сеансов вызывается диалоговое окно для выбора конфигурации серийного порта. Серийные порты обязательно должны быть сконфигурированы на обоих компьютерах, участвующих в соединении.

*Протокол DirectPlay*

Протокол *DirectPlay* используется для гарантированной доставки сообщений при использовании протоколов, не имеющих такой возможности. Например, системный провайдер *IPX* не предоставляет пересылку пакетов с гарантией доставки. В этом случае протокол *DirectPlay* самостоятельно проверяет точность доставки пакетов.

Для увеличения производительности сети протокол *DirectPlay* использует асинхронный способ передачи сообщений, при котором пакеты имеют различный размер. Так же, для того чтобы сообщение не было отправлено прежде, чем оно будет получено следующим сетевым уровнем, протокол *DirectPlay* намеренно уменьшает их количество.

Все системные сообщения всегда отсылаются с гарантией доставки независимо от того, поддерживает эту возможность выбранный протокол или нет, и вне зависимости от использования протокола *DirectPlay*, поскольку при отсылке системных сообщений протокол *DirectPlay* применяется автоматически.

Для того чтоб приложение использовало протокол *DirectPlay* для гарантированной доставки сообщений, необходимо при открытии сеанса устано-

вить флаг *DPSESSION_DIRECTPLAYPROTOCOL*. Чтобы при отсылке несистемных сообщений также иметь гарантию доставки, необходимо установить флаг *DPSEND_GUARANTEED* при вызове функции *IDirectPlay4::SendEx()*. Если использование протокола *DirectPlay* не указано, гарантированная рассылка сообщений будет произведена только в том случае, если ее поддерживает текущий системный провайдер, вне зависимости от того, установлен флаг *DPSEND_ GUARANTEED* или нет.

В таблицах Л3.1 и Л3.2 показано, как будут рассылаться несистемные сообщения в зависимости от наличия флага *DPSESSION_DIRECTPLAYPROTOCOL*.

Таблица Л3.1. Использование протоколов без флага DPSESSION_DIRECTPLAYPROTOCOL

| **Системный провайдер** | **Сообщения с гарантированной доставкой** | **Сообщения без гарантированной доставки** |
|---|---|---|
| IPX | Протокол IPX не предоставляет гарантированную доставку | Протокол IPX |
| TCP/IP | Протокол TCP | Протокол UDP |
| Модем | Модем не предоставляет гарантированную доставку | Модем |

Таблица Л3.2. Использование протоколов с флагом DPSESSION_DIRECTPLAYPROTOCOL

| **Системный провайдер** | **Сообщения с гарантированной доставкой** | **Сообщения без гарантированной доставки** |
|---|---|---|
| IPX | Протокол DirectPlay через IPX | Протокол DirectPlay через IPX |
| TCP/IP | Протокол DirectPlay через UDP | Протокол DirectPlay через UDP |
| Модем | DirectPlay через модем | Протокол DirectPlay через модем |

Для создания простейшего приложения с использованием *DirectPlay* необходимо выполнить следующие шаги:

1. инициализировать *DirectPlay*;
2. перечислить и выбрать необходимый сервис провайдер;
3. перечислить и присоединиться к уже имеющемуся сеансу или создать новый сеанс;
4. создать игрока или группу игроков;
5. ожидание получения сообщения и передача собственных сообщений.;
6. обработка полученных сообщений.

Ниже приведен перечень действий приложения, создающего сеанс:

1. получение информации от пользователя;
2. получение интерфейса *IDirectPlay4*;
3. создание сеанса;
4. создание игрока;
5. ожидание получения сообщения и передача собственных сообщений;

6. обработка полученных сообщений.

Перечень действий приложения, присоединяющегося к сеансу выглядит следующим образом:.

1. получение информации от пользователя;
2. создание интерфейса *IDirectPlay4*;
3. перечисление сеансов в текущей подсети;
4. присоединение к найденному сеансу;
5. создание игрока;
6. ожидание получения сообщения и передача собственных сообщений;
7. обработка полученных сообщений.

*Инициализация DirectPlay*

Есть два способа создания интерфейса *IDirectPlay4*.

Первый способ заключается в использовании функции *DirectPlayCreate*, которая создает интерфейс *IDirectPlay*. Неудобство ее состоит в том, что для получения интерфейса *IDirectPlay4* необходимо использовать функцию *QueryInterface()*, а затем удалить старый интерфейс функцией *Release()*.

Второй метод заключается в использовании функции *CoCreateInstance()*. При этом создается интерфейс *IDirectPlay4* без привлечения промежуточных интерфейсов.

Функция *DirectPlayCreate* создает интерфейс *IDirectPlay*. Ее прототип представлен ниже.

```
HRESULT WINAPI DirectPlayCreate(
      LPGUID IpGUIDSP,
      LPDIRECTPLAY FAR *lplpDP,
      IUnknown *lpUnk );
```

Первым параметром является указатель на уникальный идентификатор системного провайдера, который будет использоваться для передачи сообщении. Следующий параметр – указатель на интерфейс *DirectPlay*, который будет создан (данная функция может создавать только интерфейс *IDirectPlay*, а для получения более высоких версий необходимо воспользоваться функцией *QueryInterface()*.)

Последний параметр в данной версии библиотеки не используется и должен быть равным нулю, иначе функция возвратит код ошибки – *DPERR_INVALIDPARAMS*.

Если интерфейс по какой-либо причине не может быть создан, функция возвратит ошибку *DPERR_UNAVAILABLE*. В случае успешного завершения функция возвратит *DP_OK*.

Приведенный ниже фрагмент программы демонстрирует пример создания интерфейса *IDirectPlay* и последующее получение интерфейса *IDirectPlay4* (первый способ).

```
//Создание интерфейса
IDirectPlay hRet = DirectPlayCreate(GUID_NULL, lpdp, NULL):

//Получение интерфейса необходимой версии
hr = lpdp->QueryInterface(IID_IDirectPlay4, (VOID**)&lpdp4);

//Уничтожение ненужного интерфейса
lpdp->Release():
```

Далее приведен пример, иллюстрирующий второй способ создания интерфейса *IDirectPlay4*.

```
CoCreateInstance(CLSID_DirectPlay, NULL, CLSCTX_ALL, IID_IDirectPlay4A,
(VOID**)&g_pDP)
```

В параметрах функции *CoCreateInstance* указываются идентификатор класса *DirectPlay* и идентификатор интерфейса *IDirectPlay4*. Последний параметр – указатель на интерфейс, который будет создан.

Следует заметить, что в обоих методах создания интерфейса *IDirectPlay*, в начале вызывается функция инициализации интерфейса *CoInitialize()*. Следующий пример демонстрирует один из методов создания интерфейса *IDirectPlay*.

```
//Создание интерфейса IDirectPlay
CoInitialize(NULL);
hRet = DirectPlayCreate (.provider, &lpdp. NULL);
if (hRet!=DP_OK)//==DPERRJNVALIDPARAMS)
{
ErrorHandle(hMainWnd,hRet,"DirectPlayCreate error!"):
return (FALSE):
}

//Получение интерфейса необходимой версии
hRet = lpdp->QueryInterface( IID_IDirectPlay4A, (VOID**)&lpdp4);

//Уничтожение ненужного интерфейса
lpdp->Release()
```

*Поддержка интерфейсов Unicode и ANSI*

*DirectPlay* поддерживает работу как со строками *Unicode*, так и со строками *ANSI*. При этом указатель на Unicode-строку имеет тип *LPWSTR*, а на ANSI – *LPSTR*. Имена интерфейсов для работы с данными в кодировке ANSI заканчиваются на букву «A» (IDirectPlay4 для Unicode и IDirectPlay4A для ANSI). Если используется интерфейс одного типа данных, данные другого типа будут игнорироваться.

*Перечисление и инициализация системных провайдеров*

Как уже было отмечено выше, DirectPlay предоставляет четыре провайдера. Однако вполне вероятно, что в будущем появятся новые провайдеры, использующие новые протоколы. Для универсальности программы необхо-

димо, чтобы она знала обо всех провайдерах, присутствующих в системе. Это возможно при *использовании функции перечисления провайдеров.* DirectPlay перечисляет системные провайдеры при помощи двух функций:

```
IDirectPlay4::DirectPlayEnumerate
IDirectPlay4::EnumConnections()
```

Указанные функции перечисляют все зарегистрированные провайдеры, вне зависимости от того, работают они или нет. Отличие их в том, что первая перечисляет только провайдеры, а вторая в дополнение к провайдерам – еще и возможные подключения.

```
HRESULT WINAPI DirectPlayEnumerate (
      LPDPENUMDPCALLBACK lpEnumCallback,
      LPVOID lpContext);

HRESULT EnumConnections (
      LPCGUID lpguidApplication,
      LPDPENUMCONNECTIONSCALLBACK lpEnumCallback,
      LPVOID lpContext,
      DWORD dwFlags );
```

Функция *DirectPlayEnumerate()* имеет два параметра – указатель на функцию, которая будет вызываться каждый раз, когда будет найден зарегистрированный провайдер, и указатель на структуру, при помощи которой этой функции будут переданы параметры (обычно устанавливается в *NULL*). Данная функция работает со всеми версиями *DirectX*. Так как она не привязана ни к какому интерфейсу, это позволяет перечислять системные провайдеры до создания *IDirectPlay*.

Функция *EnumConnections()* имеет более широкие возможности и, как следствие, большее количество параметров. Первым ее параметром является указатель на уникальный глобальный идентификатор приложения. При этом будут перечислены все провайдеры, которые использует указанное приложение. Если вместо идентификатора приложения указан *NULL*, будут перечислены все присутствующие в системе провайдеры. Вторым параметром является  указатель на функцию, которая будет вызываться каждый раз, когда будет найдет провайдер. Параметры для нее будут передаваться через структуру, указатель на которую следует третьим параметром (обычно установлен в *NULL*). Четвертым параметром являются флаги. По умолчанию установлен флаг *DPCONNECTION_DIRECTPLAY*, указывающий, что перечисляются только системные провайдеры. Для перечисления еще и лобби-провайдеров следует добавлять флаг *DPCONNECTION_DIRECTPLAYLOBBY*.

Для каждой из этих функций существует своя функция обратного вызова, которая будет вызываться каждый раз, когда будет найден провайдер. Эту функцию должен определить программист, *DirectPlay* предоставляет лишь ее прототип. Функция *EnumDPCallback()*, используемая совместно с *DirectPlayEnumerate()* имеет следующий прототип:

```
BOOL FAR PASCAL EnumDPCallback(
      LPGUID lpguidSP,
      LPSTR/LPWSTR lpSPName,
      DWORD dwMajorVersion,
      DWORD dwMinorVersion,
      LPVOID lpContext );
```

В ее параметрах передаются: указатель на уникальный идентификатор провайдера, имя провайдера (этот параметр в зависимости от типа используемого интерфейса может принимать значения как в стандарте Unicode, так и в кодировке ANSI), старшая и младшая составляющие версии провайдера.

Функция обратного вызова *EnumConnectlonCallback()* получает большее количество параметров и ее прототип выглядит следующим образом:

```
BOOL FAR PASCAL EnumConnectionsCallback(
      LPCGUID lpguidSP,
      LPVOID lpConnection,
      DWORD dwConnectionSize,
      LPCDPNAME lpName,
      DWORD dwFlags,
      LPVOID lpContext );
```

Первым параметром функции передается уникальный идентификатор провайдера. Затем следует указатель на буфер, содержащий информацию о подключении. Эта информация необходима при вызове функции *IDirectPlay4::InitializeConnection* и содержит *DirectPlay-адрес* сеанса. Размер этого буфера задается в третьем параметре функции. Четвертым параметром следует указатель на структуру типа *DPNAME*, в которой содержится имя соединения. В зависимости от типа интерфейса имя соединения может быть двух типов – Unicode и ANSI. Параметр *dwFlags* содержит флаги, которые были указаны при вызове *EnumConnections()*.

При использовании функции *DirectPlayEnumerate()* инициализация системного провайдера происходит при создании интерфейса *IDirectPlay* с помощью функции *DirectPlayCreate*, в параметрах которой указывается используемый провайдер.

Функция *EnumConnections()* принадлежит интерфейсу *IDirectPlay* и поэтому не может быть вызвана до его создания. Чтобы использовать эту функцию, при создании интерфейса *IDirectPlay* необходимо вместо указателя на идентификатор провайдера передать *NULL*, а затем при помощи функции *InitializeConnection()*, прототип которой приведен ниже, указать, какой провайдер используется.

```
HRESULT InitializeConnection(
      LPVOID IpConnection,
      DWORD dwFlags );
```

Первым параметром передается указатель на буфер, содержащий информацию о соединении. Вторым параметром следуют флаги, которые в данный момент не используются и предназначены для будущих версий.

*Управление сеансом*

*Сеансом* в *DirectPlay* называют канал связи между двумя или более приложениями. Открыть сеанс можно разными способами. Например, приложение может выполнить перечисление всех существующих сеансов и присоединиться к одному их них. Другая возможность заключается в создании собственного сеанса. После того как приложение стало частью сеанса, появляется возможность создавать игроков и обмениваться сообщениями.

Каждый сеанс управляется сервером, который является владельцем сеанса, и только он может изменять настройки сеанса.

*DirectPlay* предоставляет набор функций для управлением сеансом, наиболее интересные из них рассмотрены ниже.

*Перечисление сеансов*

Функция *IDirectPlay4::EnumSession()* перечисляет все существующие сеансы.

```
HRESULT EnumSessions(
      LPDPSESSIONDESC2 lpsd,
      DWORD dwTimeout,
      LPDPENUMSESSIONSCALLBACK2 lpEnumSessionsCallback2,
      LPVOID lpContext,
      DWORD dwFlags );
```

Первый параметр функции – указатель на структуру *DPSESSIONDESC2*, в которой описываются параметры перечисляемых сеансов. При перечислении будут найдены только сеансы, соответствующие описанию, представленному в этой структуре. Поле *guidApplication* этой структуры указывает на приложение, с которым требуется установить сеанс. Если необходимость в определенном приложении отсутствует, указывается *GUID_NULL*. Если пользователь желает присоединиться к сеансу с определенным паролем, то этот пароль необходимо указать в поле *lpszPassword*. Все остальные поля структуры при перечислении сеансов игнорируются.

Следующий параметр – время ожидания ответа *dwTimeout*. При синхронном соединении этот параметр (измеряется в миллисекундах) задает время ожидания ответа о существовании сеанса. При перечислении сеансов приложение блокируется на указанный в этом параметре промежуток времени, по истечении которого все ответы будут игнорироваться, при этом перед началом перечисления кэш очищается. При асинхронном соединении этот параметр определяет частоту рассылки широковещательных сообщений с запросом о существовании сеанса. Если этот параметр равен нулю, время ожидания будет установлено в соответствии с параметрами используемого провайдера. В асинхронном режиме кэш не очищается, а лишь обновляется.

Третий параметр – указатель на функцию обратного вызова, которая будет вызываться каждый раз при обнаружении сеанса. Четвертый параметр – указатель на структуру, через которую могут быть переданы данные функции

обратного вызова. Обычно такие данные отсутствуют, и этот параметр устанавливается в *NULL*.

Последний параметр функции – флаги. По умолчанию он равен нулю, что является равнозначным флагу *DPENUMSESSIONS_AVAILABLE*. Все возможные для этой функции перечислены ниже:

– *DPENUMSESSIONS_ALL* – перечисление всех сеансов вне зависимости от того, принимают они новые соединения или нет;

– *DPENUMSESSIONS_ASYNC* – запускает асинхронное перечисление (если оно еще не запущено) и обновляет список сеансов до тех пор, пока не будет выполнен повторный вызов функции с параметром *DPENUMSESSIONS_STOPASYNC* или пока не произойдет вызов одной их двух функций – *Open()* либо *Release()*. Если данный флаг не установлен, выполняется синхронное перечисление;

– *DPENUMSESSIONS_AVAILABLE* – перечисляет все сеансы, готовые к принятию новых игроков. Сеансы с паролями не перечисляются, если не указан флаг *DPENUMSESSIONS_PASSWORDREQUIRED*;

– *DPENUMSESSIONS_PASSWORDREQUIRED* – при совместном использовании с флагами *DPENUMSESSIONS_AVAILABLE* или *DPENUMSESSIONS_ALL* разрешает перечисление сеансов с установленным паролем. При этом пароль указывается в поле *lpszPassword* структуры *DPSESSIONDESC2*, если данный флаг не используется, сеансы с установленным паролем не перечисляются;

– *DPENUMSESSIONS_RETURNSTATUS* – использование этого флага означает, что при перечислении не будет выводиться диалоговое окно с параметрами соединения, если соединение не может быть установлено немедленно, функция возвратит ошибку *DPERR_CONNECTING*. При этом приложение должно повторять вызов функции до тех пор, пока не будет получен код возврата *DP_OK*;

– *DPENUMSESSIONS_STOPASYNC* – перечисляет все текущие сеансы в кэше и отключает асинхронное перечисление

Если сеанс уже создан, функция *EnumSession()* возвратит ошибку *DPERR_GENERIC*, ошибка *DPERR_ USERCANCEL* означает, что пользователь прервал перечисление сеансов.

Одним из параметров функции *EnumSessions()* является указатель на функцию обратного вызова. Эта функция определяется программистом и будет вызываться каждый раз при обнаружении необходимого сеанса. Прототип этой функции выглядит следующим образом:

```
BOOL FAR PASCAL EnuniSessionsCallback2(
      LPCDPSESSIONDESC2 lpThisSD,
      LPDWORD lpdwTimeOut,
      DWORD dwFlags,
      LPVOID lpContext );
```

Первым параметром передается указатель на структуру *DPSESSIONDESC2*, в которой размещена информация о найденном сеансе. Если данный параметр равен *NULL*, это означает, что сеанс не был найден в течение отведенного на ожидание ответа промежутка времени. Если был найден сеанс с повышенной безопасностью, то такие поля, как *dwMaxPlayers* и *dwCurrentPlayers*, будут равны нулю.

Вторым параметром функции является указанный при вызове функции *EnumSessions()* период ожидания ответа от сеанса.

Третий параметр – флаги. На данный момент определен лишь один флаг, эквивалентный нулю – *DPESC_TIMEDOUT*. Он указывает на то, что функция перечисления сеансов использует заданный период ожидания.

*Открытие сеанса*

Открытие сеанса подразумевает под собой как создание нового сеанса, так и подключение к уже существующему. Для открытия сеанса используется функция *IDirectPlay4::Open()*, прототип которой представлен ниже:

```
HRESULT Open(
      LPDPSESSIONDESC2 lpsd,
      DWORD dwFlags );
```

Первым параметром этой функции передается указатель на структуру типа *DPSESSIONDESC2*, в которой описываются установки сеанса. Если происходит присоединение к уже существующему сеансу, то заполняются только поля *dwSize*, *guidInstance* и *lpszPassword*. Если же создается новый сеанс, то обязательно заполнение всех полей этой структуры за исключением *guidInstanse*, которое заполняет *DirectPlay*.

Вторым параметром функции *Open()* следуют флаги:

– *DPOPEN_CREATE* – создается новый сеанс и локальный компьютер при этом объявляется как сервер имен;

– *DPOPEN_JOIN* – присоединение к уже существующему сеансу;

– *DPOPEN_RETURNSTATUS* – использование данного флага указывает на то, что в момент присоединения не будет выведено диалоговое окно с текущим статусом. Если соединение не состоялось, функция возвратит ошибку *DPERR_CONNECTING*. При использовании данного флага необходимо выполнять вызов функции *Open()* до тех пор, пока она не возвратит *DP_OK*;

После присоединения к сеансу появляется возможность создания игроков и обмена сообщениями. Но следует заметить, что до тех пор пока игрок не будет создан, приложение не сможет отправлять и принимать сообщения, так как информация об инициаторе действия отсутствует.

Как уже было отмечено выше, для создания сеанса необходимо заполнить все поля структуры *DPSESSIONDESC2*, кроме поля guidInstance. Указанная структура имеет следующее описание:

```
typedef struct {
      DWORD dwSize;
      DWORD dwFlags;
      GUID quidInstance;
      GUID giudApplication;
      DWORD dwMaxPlayers;
      DWORD dwCurrentPlayers;
      union {
            LPWSTR lpszSessionName;
            LPSTR lpszSess-ionNameA;
            };
      union {
            LPWSTR lpszPassword:
            LPSTR lpszPasswordA;
            };
      DWORD dwReservedl;
      DWORD dwReserved2;
      DWORD dwUser1;
      DWORD dwUser2;
      DWORD dwUser3;
      DWORD dwUser4 } DPSESSIONDESC2, FAR *LPDPSESSIONDESC2;
```

Поле *dwSize* – размер структуры, необходим для определения версий структуры и заполняется оператором *sizeof()*.

Поле *dwFlags* содержит флаги, указывающие на параметры сеанса:

– *DPSESSION_CLIENTSERVER* – указывает на создание соединения на основе сервера и должен использоваться при создании сеанса. Если флаг не задан, создается одноранговое соединение. Флаг не может быть использован совместно с *DPSESSION_MIGRATEHOST*;

– *DPSESSION_DIRECTPLAYPROTOCOL* – указывает на использование протокола *DirectPlay* совместно с основным протоколом провайдера;

– *DPSESSION_JOINDISABLED* – указывает на то, что никакое приложение не может присоединиться к сеансу;

– *DPSESSION_KEEPALIVE* – указывает на то, что сеанс автоматически определяет, с какими пользователями прервалась связь;

– *DPSESSION_MIGRATEHOST* – указывает на возможность передачи функций сервера имен другому компьютеру при выходе из строя текущего сервера (если данный флаг не задан, пользователи при отключении сервера имен будут отключены от сеанса);

– *DPSESSION_MULTICASTSERVER* – указывает на создание сервера с групповой рассылкой сообщений. Устанавливается только для одноранговых соединений и не может быть использован совместно с флагом *DPSESSION_MIGRATEHOST*;

– *DPSESSION_NEWPLAYERSDISABLED* – указывает на невозможность создания новых игроков, кроме того, при установке этого флага никакие приложения не могу присоединиться к текущему сеансу;

- *DPSESSION_NODATAMESSAGES* – указывает на то, что при изменении статуса или данных игрока (группы) не последует рассылка системных сообщений;
- *DPSESSION_NOMESSAGEID* – указывает на то, что отсылаемые сообщения не будут дополняться информацией об отправителе и адресате;
- *DPSESSION_NOPRESERVEORDER* – указывает на то, что нет необходимости сохранять порядок получения сообщений с гарантированной доставкой. По умолчанию порядок получения сообщений с гарантированной доставкой сохраняется;
- *DPSESSION_OPTIMIZELATENCY* – указывает на использование оптимизации времени доставки сообщения, предоставляемой провайдером;
- *DPSESSION_PASSWORDREQUIRED* – указывает на использование пароля для поиска сеанса или подключения к нему;
- *DPSESSION_PRIVATE* – указывает на то, что не отвечает на запросы о перечислении сеансов, если пароль запроса не указан;
- *DPSESSION_SECURESERVER* – указывает на то, что для подключения к сеансу необходима идентификация пользователя.

Поле *guidInstance* содержит уникальный идентификатор сеанса. При создании сеанса автоматически генерируется *DirectPlay*, а при присоединении к сеансу его необходимо указывать.

Поле *guidApplication* содержит уникальный идентификатор приложения, открывшего сеанс. Если нет необходимости присоединиться к сеансам, созданным определенными приложениями, поле должно содержать *GUID_NULL*.

Поле *dwMaxPlayers* показывает максимальное количество игроков, которые могут быть созданы в данном сеансе (ноль указывает, что ограничения отсутствуют).

Поле *dwCurrentPlayers* показывает количество игроков, существующих в сеансе на данный момент.

Поле *lpszSessionName* является указателем на строку (*Unicode* или *ANSI*) с именем сеанса.

Поле *lpszPassword* является указателем на строку (*Unicode* или *ANSI*), в которой хранится пароль сеанса.

Поля *dwReserved1* и *dwReserved2* являются зарезервированными.

Поля *dwUser1*, *dwUser2*, *dwUser3*, *dwUser4* содержат пользовательские данные.

*Открытие сеанса с повышенной безопасностью*

DirectPlay предоставляет возможность аутентификации пользователей при помощи SSPI (Security Support Provider Interface – интерфейс провайдера с поддержкой безопасности). При этом существуют следующие особенности:

– пользователю должен аутентифицироваться только один раз, далее эта информация будет использоваться каждый раз при отправке сообщений;
– каждое сообщение имеет электронную подпись, идентифицирующую отправившего его пользователя;
– все системные сообщения шифруются.

Сеансы с повышенной безопасностью требуют присоединения с указанием имени пользователя и пароля. Для открытия сеанса используется функция *IDirectPlay4::SecureOpen()*., которая имеет следующий прототип:

```
HRESULT SecureOpen(
      LPCDPSESSIONDESC2 lpsd,
      DWORD dwFlags,
      LPCDPSECURITYDESC lpSecurity,
      LPCDPCREDENTIALS lpCredentials );
```

Параметры этой функции отличаются от параметров функции *Open()*двумя дополнительными полями – указателями на структуры *DPSECURITYDESC* и *DPCREDENTIALS*.

Структура *DPSECURITYDESC* описывает свойства безопасности сеанса *DirectPlay*. Если вместо указателя на данную структуру передать NULL, будут использоваться значения по умолчанию.

```
typedef struct {
      DWORD dwSize;
      DWORD dwFlags;
      union {
            LPWSTR lpszSSPIProvider;
            LPSTR lpszSSPIProviderA;
            };
      union {
            LPWSTR lpszCAPI Provider;
            LPSTR lpszCAPIProviderA;
            };
      DWORD dwCAPIProviderType;
      DWORD dwEncryptionAlgonthm;} DPSECURITYDESC, FAR *LPDPSECURITYDESC;
```

Поле *dwFlags* не используется и предназначено для будущих версий.

Поле *lpszSSPIProvider* является указателем на строку (*Unicode* или *ANSI*), описывающую пакеты системного провайдера поддержки безопасности. Этот провайдер будет применяться для аутентификации пользователя. Если указан *NULL*, используется провайдер по умолчанию.

Поле *lpszCAPIProvider* является указателем на строку (*Unicode* или *ANSI*), описывающую пакеты провайдера *CryptoAPI* (программный интерфейс шифрования). Данный провайдер будет применяться для шифрования всех сообщений. Если указан *NULL*, используется провайдер по умолчанию.

Поле *dwCAPIProviderType* указывает на тип провайдера, используемого для шифрования. Если указан ноль, применяется значение по умолчанию – *PROV_RSA_FULL*.

Поле *dwEncryptionAlgonthm* указывает на тип алгоритма, используемого для шифрования сообщений. Если указан ноль, применяется значение по умолчанию – *CALG_RC4*.

Структура *DPCREDENTIALS* содержит всю необходимую для аутентификации информацию – имя домена, а также имя и пароль пользователя. Если вместо указателя на структуру передать *NULL*, аутентификация не будет произведена. Эта структура игнорируется при создании сеанса.

```
typedef struct {
      DWORD dwSize:
      DWORD dwFlags;
      union {
            LPWSTR lpszUsername;
            LPSTR lpszUsernameA;
      };
      union {
            LPWSTR lpszPassword;
            LPSTR lpszPasswordA;
      };
      union {
            LPWSTR lpszDomain;
            LPSTR lpszDomainA
            } DPCREDENTIALS, FAR *LPDPCREDENTIALS;
```

Поле *dwFlags* не используется и предназначено для будущих версий.

Поля *lpszUsername*, *lpszPassword*, *lpszDomain* указывают на имя пользователя, его пароль и имя домена соответственно.

*Закрытие сеанса*

После окончания работы с сеансом его необходимо закрыть. Это делается для того, чтобы игроки получили информацию именно о закрытии соединения, а не об обрыве связи. Для закрытия сеанса используется функция *IDi'rectPlay4::Close ()*, прототип ее представлен ниже:

```
HRESULT Close();
```

При вызове этой функции происходит удаление всех локальных игроков. После этого происходит информирование всех остальных игроков о происшедшем событии при помощи системных сообщений *DPMSG_DELETEPLAYERFROMGROUP* и *DPMSG_ DESTROYPLAYERORGROUP*.

*Получение и установка свойств сеанса*

Каждый сеанс имеет набор свойств, характеризующих его. Эти свойства задаются при открытии сеанса. Любой игрок в сеансе может получить их, а объект, открывший сеанс, еще и изменить.

Получение свойств сеанса осуществляется при помощи функции IDirectPlay4::GetSessionDesc(), прототип которой имеет вид:

```
HRESULT GetSessionDesc( LPVOID lpData, LPDWORD lpdwDataSize);
```

Параметр *lpData* является указателем на буфер, хранящий структуру *DPSESSIONDESC2*, в которую будет помещена информация о сеансе.

В параметре *lpdwDataSize* передается указатель на переменную, в которую будет помещен размер созданного буфера.

Для изменения информации о сеансе используется функция *IDirectPlay4::SetSessionDesc()*, прототип которой представлен ниже:

```
HRESULT SetSessionDesc( LPDPSESSIONDESC2 IpSessDesc, DWORD dwFlags );
```

В параметре *lpSessDesc* передается указатель на структуру типа DPSESSIONDESC2, в которой хранятся новые настройки сеанса.

Параметр *dwFlags* не используется и предназначен для будущих версий.

*Дополнительная информация о сеансе*

Кроме информации о текущих настройках сеанса существует информация о его возможностях. Получить эти сведения можно при помощи функции *IDirect Play4::GetCaps()*, прототип функции:

```
HRESULT GetCaps(LPDPCAPS lpDPCaps, DWORD dwFlags );
```

В параметре *1pDPCaps* передается указатель на структуру типа *DPCAPS*, в которую будет помещена информация о возможностях текущего сеанса.

Параметр *dwFlags* задает флаги и имеет одно возможное значение – *DPGETCAPS_GUARANTED*, указывающее на то, что запрос о возможностях сеанса будет производиться с гарантией доставки сообщений.

Структура *DPCAPS* имеет следующий вид:

```
typedef struct {
      DWORD dwSize;
      DWORD dwFlags;
      DWORD dwMaxBufferSize;
      DWORD dwMaxQueueSize;
      DWORD dwMaxPlayers;
      DWORD dwHundredBaud;
      DWORD dwLatency;
      DWORD dwMaxLocalPlayers;
      DWORD dwHeaderLength;
      DWORD dwTimeout;
      } DPCAPS, FAR *LPDPCAPS;
```

Поле *dwFlags* содержит флаги, характеризующие объект *DirectPlay*:

– *DPCAPS_ASYNCCANCELALLSUPPORTED* – указывает на возможность отмены отправки всех асинхронных сообщений;
– *DPCAPS_ASYNCCANCELSUPPORTED* – указывает на возможность отмены отправки асинхронных сообщений;

- *DPCAPS_ASYNCSUPPORTED* – указывает на поддержку асинхронных сообщений. Эта возможность должна поддерживаться сетевым протоколом, который используется провайдером;
- *DPCAPS_ENCRYPTIONSUPPORTED* – указывает на возможность шифрования сообщений. Эта возможность существует при сеансах с повышенной безопасностью или при поддержке системным провайдером шифрования сообщений;
- *DPCAPS_GROUPOPTIMIZED* – указывает на оптимизацию отправки сообщений, путем рассылки групповых сообщений;
- *DPCAPS_GUARANTEEDOPTIMIZED* – указывает на возможность отправки сообщений с гарантией их доставки, поддерживаемых системным провайдером;
- *DPCAPS_GUARANTEEDSUPPORTED* – указывает на возможность отправки сообщений с гарантией их доставки, поддерживаемой или системным провайдером, или протоколом *DirectPlay*;
- *DPCAPS_ISHOST* – указывает, что ЭВМ является сервером сеанса;
- *DPCAPS_KEEPALIVEOPTIMIZED* – указывает на то, что системный провайдер может определить потерю связи с сервером сеанса;
- *DPCAPS_SENDPRIORITYSUPPORTED* – указывает на использование приоритетов сообщений при их отправке;
- *DPCAPS_SENDTIMEOUTSUPPORTED* – указывает на поддержку системным провайдером или объектом *DirectPlay* периода ожидания подтверждения доставки пакета;
- *DPCAPS_SIGNINGSUPPORTED* – указывает на поддержку объектом *DirectPlay* или системным провайдером аутентификации сообщений;

В поле *dwMaxBufferSize* указывается максимальный размер одного пакета, который может отправить текущий провайдер. Если сообщение имеет большой размер, оно разбивается на несколько пакетов, не превышающих максимальной величины.

Поле *dwMaxQueueSize* в настоящий момент не используется. В предыдущих версиях оно означало максимальный размер очереди сообщений.

Поле *dwMaxPlayers* задает максимальное количество игроков сеанса;

Поле *dwHundredBaud* содержит величину, указывающую пропускную способность канала, единица измерения которой равна 100 бит/с;

Поле *dwLatency* показывает время в миллисекундах, за которое пакет доходит до своего адресата (если равно нулю, то не возможно определить);

Поле *dwMaxLocalPlayer* содержит максимально возможное количество локальных игроков в сеансе;

Поле *dwHeaderLength* указывает размер заголовка, который *DirectPlay* присоединяет к каждому отправляемому сообщению. Эта величина зависит от типа используемого провайдера.

Поле *dwTimeout* содержит время, в течение которого происходит ожидание подтверждения доставки отправленного сообщения.

*Управление игроками*

*Игрок* в терминологии *DirectPlay* представляет собой логический объект, имеющий возможность отправлять и принимать сообщения. Любой сеанс может содержать в себе столько игроков, сколько позволяет пропускная способность сети. Одно приложение может создавать такое количество игроков, какое допускают настройки сеанса.

Каждый игрок при создании получает уникальный идентификационный номер. Данный номер принадлежит ему до тех пор, пока существует сеанс. Если сеанс был прерван и заново открыт, идентификационные номера игроков генерируются повторно.

Все игроки подразделяются на локальные и удаленные в зависимости от того, на локальном или удаленном компьютере они были созданы. Для отправки и приема сообщений компьютер должен иметь хотя бы одного созданного игрока. Также вполне возможно создание нескольких игроков на одном компьютере.

Любое сообщение, отправляемое игроком, предназначается не компьютеру, а конкретному игроку. Сообщение может быть адресовано как удаленному игроку, так и локальному, причем в последнем случае оно не пересылается по сети. Каждое сообщение, полученное приложением, передается игроку с указанием отправителя. При передаче системных сообщений отправитель указывается как *DPID_ SYSMSG*.

Библиотека *DirectPlay* предоставляет набор функций управления игроками. Эти функции используются для создания игрока, его удаления, перечисления существующих игроков, получения и установки различных данных.

*Перечисление существующих игроков*

Для получения списка всех игроков используется функция *IDirectPlay4::EnuinPlayers()*. Если в момент вызова функции нет открытых сеансов, возможно перечисление пользователей в удаленном сеансе, для чего необходимо знать его идентификационный номер – *guidlnstance*. Если удаленный сеанс имеет пароль, перечисление пользователей не возможно. Прототип функции перечисления игроков следующий:

```
HRESULT EnumPlayers(
      LPGUID lpguidInstance,
      LPDPENUMPLAYERSCALLBACK2 lpEnumPlayersCallback2,
      LPVOID lpContext.
      DWORD dwFlags );
```

Параметр *lpguidInstance* является указателем на уникальный идентификационный номер сеанса, игроки которого должны быть перечислены. При установке этого параметра в *NULL* происходит перечисление игроков текущего открытого сеанса. Определение номера удаленного сеанса возможно при использовании функции *EnumSessions()*. Этот параметр игнорируется, если при вызове функции *EnumSessions()* не указан флаг *DPENUMPLAYER_SESSION*.

Параметр *lpEnumPlayersCallback2* является указателем на функцию обратного вызова, обращение к которой происходит при нахождении игрока, удовлетворяющего указанным во флагах функции критериям.

Параметр *dwFlags* задает флаги, определяющие параметры поиска игроков:

- *DPENUMPLAYERS_ALL* – задан по умолчанию и указывает на то, что происходит перечисление всех игроков в активном сеансе;
- *DPENUMPLAYERS_GROUP* – указывает на то, что при перечислении пользователей происходит также перечисление всех групп;
- *DPENUMPLAYERS_LOCAL* – указывает на то, что выполняется перечисление только локальных игроков;
- *DPENUMPLAYERS_REMOTE* – указывает на то, что выполняется перечисление только удаленных игроков;
- *DPENUMPLAYERS_SERVERPLAYER* – указывает на то, что выполняется поиск только того игрока, который был создан на сервере;
- *DPENUMPLAYERS_SESSION* – указывает на то, что выполняется перечисление игроков, принадлежащих только удаленному сеансу. Используется лишь в том случае, если не существует открытых сеансов;
- *DPENUMPLAYERS_SPECTATOR* – указывает на то, что выполняется перечисление только наблюдающих игроков, то есть не участвующих в игровом процессе.

Все параметры найденного игрока передаются функции обратного вызова. Ее прототип представлен ниже:

```
BOOL FAR PASCAL EnumPlayersCallback2(
      DPID dpid,
      DWORD dwPlayerType,
      LPCDPNAME lpName,
      DWORD dwFlags,
      LPVOID lpContext
```

Параметр *dpid* содержит уникальный идентификационный номер игрока или группы игроков, который присваивается им при создании.

Параметр *dwPlayerType* указывает, кто был найден – игрок (*DPPLAYERTYPE_PLAYER*) или группа игроков (*DPPLAYERTYPE_GROUP*).

Параметр *lpName* является указателем на структуру *DPNAME*, содержащую имя найденного пользователя.

Параметр *dwFlags* описывает параметры игрока или группы:

– *DPENUMGROUPS_HIDDEN* – указывает на то, что найденная группа является скрытой;
– *DPENUMGROUPS_SHORTCUT* – указывает на то, что найдена подгруппа, присоединенная к главной группе;
– *DPENUMGROUPS_STAGINGAREA* – указывает на то, что найденная группа является игровой зоной;
– *DPENUMPLAYERS_GROUP* – указывает на то, что происходит перечисление как игроков, так и групп игроков;
– *DPENUMPLAYERS_LOCAL* – указывает на то, что найденный пользователь является локальным;
– *DPENUMPLAYERS_OWNER* – указывает на то, что найденный игрок является владельцем какой-либо группы;
– *DPENUMPLAYERS_REMOTE* – указывает на то, что найденный игрок является удаленным;
– *DPENUMPLAYERS_SERVERPLAYER* – указывает на то, что найденный игрок является серверным;
– *DPENUMPLAYERS_SESSION* – указывает на то, что происходит перечисление пользователей сеанса, указанного в переменной *lpguidInstance* функции *EnumPlayers()*;
– *DPENUMPLAYERS_SPECTATOR* – указывает на то, что найденный пользователь является наблюдателем и не участвует в игровом процессе;

Функция возвращает значение *TRUE*, если перечисление игроков продолжается, и *FALSE*, если прекращается.

*Создание игрока*

Для обмена сообщениями необходимо наличие игроков, которые будут их отправлять и получать. Каждый игрок при создании указывает свое имя и тип, а в ответ получает номер, позволяющий его однозначно идентифицировать. Создание игрока производится при помощи функции *IDirectPlay4::CreatePlayer ()*, прототип функции представлен ниже:

```
HRESULT CreatePIayer(
      LPDPID lpidPlayer,
      LPDPNAME lpPlayerName,
      HANDLE hEvent,
      LPVOID lpData,
      DWORD dwDataSize,
      DWORD dwFlags );
```

Параметр *lpidPtayer* является указателем на переменную, которая будет заполнена уникальным идентификационным номером игрока.

В параметре *lpPlayerName* передается указатель на структуру типа *DPNAME*, содержащую имя создаваемого игрока (не обязательно уникальное), если этот параметр равен *NULL*, создается игрок без имени.

Параметр *hEven* является событийным объектом, которому будут передаваться уведомления о получении сообщений, предназначенных создаваемому игроку.

Параметр *lpData* является указателем на область данных, принадлежащих игроку, если равен *NULL*, то игрок не имеет собственных данных.

Параметр *dwDataSize* задает размер области данных игрока.

Параметр *dwFlags* содержит флаги, определяющие тип создаваемого игрока (*DPPLAYER_SERVERPLAYER* – серверный игрок, *DPPLAYER_SPECTATOR* – игрок наблюдатель и не участвует в игре).

*Удаление игрока*

Если пользователь желает покинуть сеанс, его игрока необходимо удалить. При этом происходит снятие с очереди получения всех сообщений, адресованных этому игроку, а также исключение его из групп, в которые он входил. Идентификационный номер игрока не освобождается, но и не будет больше использоваться в текущем сеансе.

Удалить игрока может либо то приложение, которое его создало, либо сервер сеанса. Удаление игрока производится при помощи функции *IDirectPlay4::DesroyPlayer()*, прототип которой имеет вид:

```
HRESULT DestroyPlayer( DPID idPlayer);
```

Единственный параметр этой функции – идентификационный номер игрока, который будет удален.

При вызове функции *DestroyPlayer()* производится рассылка системных сообщений *DPMSG_DELETEPLAYERFROMGROUP* и *DPMSG_DESTROYPLAYERORGROUP*.

*Получение учетной записи игрока*

При сеансах с повышенной безопасностью каждый игрок имеет учетную запись, которая необходима для аутентификации. Эта информация может быть получена сервером сеанса при помощи функции *GetPlayerAccount()*. Прототип функции приведен ниже:

```
HRESULT GetPlayerAccount(
      DPID idPlayer,
      DWORD dwFlags.
      LPVOID lpData,
      LPDWORD lpdwDataSize );
```

Параметр *idPlayer* является идентификатором игрока.

Параметр *dwFlags* содержит флаги и в данный момент не используется.

Параметр *lpData* является указателем на буфер, в который будет размещена информация об игроке. Чтение данных из буфера производится при помощи структуры *DPACCOUNTDESC*.

В параметре *lpdwDataSize* передается указатель на переменную, в которую будет занесен размер буфера для данных из учетной записи игрока. Если буфер окажется не достаточного размера, функция возвратит ошибку *DPERR_BUFFERTOOSMALL*. Для того чтобы узнать требуемый размер буфера, необходимо в параметре *lpData* передать *NULL*.

*Получение адреса игрока*

Каждый игрок имеет адрес, совпадающий с сетевым адресом ЭВМ, на котором игрок был создан. В некоторых случаях игрок может иметь несколько адресов, например, если на ЭВМ установлены сетевая карта и модем.

Получить адрес игрока можно при помощи функции *IDirectPlay4::GetPlayerAddress()*, прототип которой представлен ниже:

```
HRESULT GetPlayerAddress(
      DPID idPlayer.
      LPVOID lpData,
      LPDWORD lpdwDataSize );
```

Параметр *idPlayer* является идентификационным номером игрока, адрес которого необходимо получить (установка этого параметра в ноль позволяет получить список возможных опций системного провайдера).

Параметр *lpData* является указателем на буфер, в который будет помещен адрес. Для того чтобы узнать размер буфера, необходимый для размещения в нем адреса, этот параметр устанавливается в *NULL*.

Параметр *lpdwDataSize* является указателем на переменную, в которую будет помещен размер созданного буфера. Если буфер окажется слишком мал, функция вернет ошибку *DPERR_BUFFERTOOSMALL*.

*Получение данных игрока, определяемых приложением*

Эти данные хранятся в буфере, который создается при инициализации игрока. При одноранговых соединениях существует возможность получения данных любого игрока текущего сеанса. Для этого используется функция *IDirectPlay4::Get PlayerData()*. Прототип функции приведен ниже:

```
HRESULT GetPlayerData(
      DPID idPlayer,
      LPVOID lpData,
      LPDWORD lpdwDataSize,
      DWORD dwFlags );
```

Первые три параметра функции аналогичны функции *GetPlayerAddress(),* за исключением того что буфер будет заполнен не адресом, а данными.

Параметр *dwF1ags* является флагами, указывающими, данные какого игрока будут получены – локального (*DPGET_LOCAL*) или удаленного (*DPGET_REMOTE*).

*Получение дополнительных данных игрока*

Игроки бывают различных типов – локальные, серверные или наблюдатели. При помощи функции *IDirectPlay4::GetPlayerF1ags()* возможно получение информации о типе игрока. Прототип функции имеет вид:

```
HRESULT GetPlayerFlags (DPID idPlayer, LPDWORD lpdwFlags );
```

Параметр *idPlayer* является идентификационным номером игрока.

Параметр *lpdwFlags* является флагами, описывающими тип игрока. Всего на данный момент существуют три флага: *PDPLAYER_LOCAL* – игрок локальный, *DPLAYER_SERVERPLAYER* – игрок создан на сервере, и *DPPLAYER_SPECTATOR* – игрок является наблюдателем.

*Получение информации о возможностях игрока*

Каждый игрок обладает возможностями, обусловленными как используемыми программными средствами, так и аппаратной частью компьютера. Узнать о возможностях игрока можно при помощи функции *IDirectPlay4::GetPlayerCaps()*, которая имеет следующий прототип:

```
HRESULT GetPlayerCaps(DPID idPlayer, LPDPCAPS lpPlayerCaps, DWORD dwFlags );
```

Параметр *idPlayer* является идентификационным номером игрока.

Параметр *lpPlayerCaps* является указателем на структуру типа *DPCAPS*, описывающую возможности игрока.

Параметр *dwFlags* является флагами и может принимать одно из двух значений: ноль (принятое по умолчанию), указывающее на то, что будут получены значения для сообщений без гарантии доставки, и *DPGETCAPS_GUARANTEED*, которое говорит о том, что будут получены значения для сообщений с гарантией доставки.

*Получение и изменение имени игрока*

Каждый игрок при создании указывает свое имя. Имя не является уникальным параметром и никак не используется механизмами *DirectPlay*. Для получения имени игрока применяется функция *IDirectPlay4::GetPlayerName()*, прототип которой представлен ниже:

```
HRESULT GetPlayerName( DPID idPlayer, LPVOID lpData, LPDWORD lpdwDataSize );
```

Параметры функции аналогичны параметрам функции *GetPlayerAddress(),* за исключением того что буфер будет заполнен не адресом, а структурой *DPNAME*, содержащей имя игрока.

Получить имя игрока может любой компьютер в сеансе. Изменить же его может только компьютер, на котором этот игрок был создан. Для изменения имени игрока используется функция *IDirectPlay4::SetPlayerName()*, прототип которой приведен ниже:

```
HRESULT SetPlayerName(DPID idPlayer, LPDPNAME lpPlayerName, DWORD dwFlags );
```

Параметр *idPlayer* является идентификационным номером игрока, игрок обязательно должен быть локальным.

Параметр *lpPlayerName* является указателем на структуру *DPNAME*, хранящую имя, которое будет присвоено игроку. Если в этом параметре передать *NULL*, игрок будет без имени.

Параметр *dwFlags* является флагами, указывающими, каким образом игроки будут оповещаться об изменении имени (*DPSET_GUARANTEED* – оповещение по возможности будет производиться с использованием гарантированной доставки, *DPSET_LOCAL* – будут оповещены только локальные игроки, т. е. которые созданы на том же ПК, *DPSET_REMOTE* – будут оповещены все игроки в текущем сеансе (флаг по умолчанию)).

При изменении имени игрока производится рассылка системного сообщения *DPMSG_SETPLAYERORGROUPENAME* всем игрокам сеанса.

*Сообщения*

После того как будет открыт сеанс и созданы игроки, появляется возможность обмена данными между компьютерами посредством сети. Данные, передаваемые по сети между игроками, называются сообщениями. Существуют сообщения двух типов: сообщения игроков и системные. Сообщения игроков – это данные, которыми игроки или группы обмениваются в процессе сеанса. Системные сообщения являются оповестительными и информируют о произошедших в сеансе изменениях.

Все сообщения, полученные компьютером, помещаются в очередь, из которой приложение производит выборку.

Из очереди сообщения могут быть получены двумя способами. Первый заключается в проверке очереди сообщений в главном цикле приложения. Обычно этот метод используют для однопоточных приложений. Второй метод состоит в использовании различных потоков для получения сообщения и его обработки.

*Системные и пользовательские сообщения*

Системные сообщения распознаются по полю *lpidFrom*, которое в этом случае должно быть равно *DPID_SYSMSG*. Если сообщение было послано игроком, то поле содержит идентификатор отправителя.

Все сообщения содержат какие-либо данные. В системных сообщениях – это информация о произошедшем событии. Она может быть определена при помощи констант, имеющих префикс *DPSYS*. Все эти константы описаны ниже. Чтобы привести буфер сообщения к необходимому виду, используется структура *DPMSG_GENERIC*. Системное сообщение также содержит дополнительную информацию о произошедшем событии. Для ее получения буфер приводится к типу структуры, описывающей данное событие. Имя этой

структуры совпадает с именем константы, полученной при помощи *DPMSG_GENERIC*, за исключение префикса, который заменяется на *DPMSG*. Следует заметить, что на самом деле буфер системного сообщения содержит именно структуру, хранящую данные о событии, а при помощи *DPMSG_GENERIC* производится чтение ее первого поля с информацией о типе этой структуры.

Системные сообщения:

- *DPSYS_ADDGROUPTOGROUP* – существующая группа включена в другую группу в качестве подгруппы;
- *DPSYS_ADDPLAYERTOGROUP* – существующий игрок включен в группу;
- *DPSYS_CHAT* – получено текстовое сообщение;
- *DPSYS_CREATEPLAYERORGROUP* – создан новый игрок или группа;
- *DPSYS_DELETEGROUPFROMGROUP* – присоединенная подгруппа отсоединена от главной группы;
- *DPSYS_DELETEPLAYERFROMGROUP* – игрок исключен из группы;
- *DPSYS_DESTROYPLAYERORGROUP* – удален игрок или группа;
- *DPSYS_HOST* – текущий сервер имен покинул сеанс и выбран новый;
- *DPSYS_SECUREMESSAGE* –получено сообщение с цифровой подписью или зашифрованное сообщение;
- *DPSYS_SENDCOMPLETE* – асинхронная пересылка завершена;
- *DPSYS_SESSIONLOST* – связь с сеансом потеряна;
- *DPSYS_SETGROUPOMNER* –установлен новый владелец группы;
- *DPSYS_SETPLAYERORGROUPOATA* – произведена установка определяемых приложением данных игрока или группы;
- *DPSYS_SETPLAYERORGROUPNAME* – произошла смена имени игрока или группы;
- *DPSYS_SETSESSIONDESC* – изменено описание сеанса;

Вид данных, представляющих сообщения игроков, определяется только тем приложением, которое отправило эту информацию. Обычно первым элементом этих данных является идентификатор сообщения. Это позволяет проверить очередность получения сообщений. Буфер после приема должен быть приведен к тому же типу данных, который использовался при пересылке сообщения.

*Синхронные сообщения*

За отправку сообщений отвечают две функции: *IDirectPlay4::Send()* и *IDirectP1ay4::SendEx()*. Первая из них является более простой в использовании и отправляет сообщения только асинхронно. При ее вызове происходит

блокировка процессов приложения до тех пор, пока не будет получено подтверждение доставки. Прототип функции *Send()* показан ниже:

```
HRESULT Send(
      DPID idFrom,
      DPID -idTo,
      DWORD dwFlags.
      LPVOID lpData,
      DWORD dwDataSize );
```

Параметр *idFrom* – это идентификационный номер игрока-отправителя;

Параметр *idTo* – идентификационный номер игрока-получателя (*DPID_ALLPLAYERS* – для широковещательных сообщений).

Параметр *dwFlags* содержит флаги, уточняющие особенности доставки сообщения (*DPSEND_GUARANTEED* – доставка сообщения гарантируется, *DPSEND_ENCRYPTED* –сообщение зашифровано и будет отправлено как системное, *DPSEND_SIGNED* – сообщение имеет цифровую подпись).

Параметр *lpData* является указателем на буфер, содержащий данные для отправки.

Параметр *dwDataSize* задает размер буфера.

Функция *Send()* может отправлять сообщения как с гарантией доставки, так и без. Отправка с гарантией доставки требует протокола *DirectPlay* или системного провайдера, поддерживающего такую возможность.

Если сообщение рассылается как широковещательное, то отправитель не получает его копии. Исключение составляет случай, когда указан флаг *DPSESSION_NOMESSAGEID*, при котором сообщения не имеют заголовка и их владельца невозможно определить.

*Асинхронные сообщения*

Возможность асинхронной связи освобождает приложение от блокировки во время отправки каждого сообщения. Также это позволяет приложению использовать мониторинг очереди сообщений и возможность отмены отправки сообщений.

Асинхронные сообщения возможны, только если сеанс использует протокол *DirectPlay* или при поддержке асинхронных сообщений системным провайдером. Проверить эту возможность системного провайдера можно с помощью функции *GetCaps()*, указав при ее вызове флаг *DPCAPS_ASYNCSUPPORTED*.

При отправке асинхронного сообщения с использованием функции *SendEx()* происходит возврат в программу без ожидания уведомления об отправке сообщения и, если указана гарантированная доставка, о его получении. Позже, когда сообщение достигнет своего адресата, приложение получит системное сообщение *PDMSG_SENDCOMPLETE*. Если в получении этого системного сообщения нет необходимости, при отправке асинхронного

сообщения нужно указать флаг *DPSEND_ NOSENDCOMPLETEMSG*. Ниже приводится прототип функции *SendEx()* с описанием параметров.

```
HRESULT SendEx(
      DPID idFrom.
      DPID idTo,
      DWORD dwFlags,
      LPVOID lpData.
      DWORD dwDataSize,
      DWORD dwPriority.
      DWORD dwTimeout,
      LPVOID IpContext,
      LPDWORD lpdwMsgID );
```

Большинство параметров аналогично параметрам функций Send(), поэтому ниже описаны лишь новые параметры.

Параметр *dwFlags* имеет два дополнительных флага (*DPSEND_NOSENDCOMPLETEMSG* – нет необходимости в системном сообщении, подтверждающем доставку, *DPSEND_ASYNC* – асинхронная отправка сообщения).

Параметр *dwPnonty* указывает на приоритетность сообщения. Приоритет влияет на очередность отправки сообщений, но никак не сказывается на его доставке. Самый высокий приоритет – 65535, самый низкий – 0. Отправка сообщений с использованием приоритета должна поддерживаться системным провайдером.

Параметр *dwTimeout* задает интервал ожидания отправки сообщения в миллисекундах. Если время, указанное в этом параметре, прошло, а сообщение так и не достигло своего адресата, отправка автоматически отменяется. Если указан нулевой интервал ожидания, то значение этого параметра не используется.

Параметр *lpdwMsgID* является указателем на переменную, в которую будет помещен идентификационный номер сообщения, генерируемый *DirectPlay*. Этот номер может быть использован для отмены отправки сообщения. Если отменять сообщение не требуется, параметр может быть установлен в *NULL*.

При успешной доставке функция возвращает *DP_OK* для синхронных сообщений и *DPERR_PENDING* – для асинхронных.

Отмена сообщений и использование приоритетов возможны только при асинхронном методе отправки. Причем отменить сообщение можно как при помощи идентификационного номера, генерируемого *DirectPlay*, так и при помощи приоритетов.

Для отмены сообщения необходимо несколько условий, таких как обязательное использование протокола *DirectPlay* и наличие отменяемого сообщения в очереди. Если сообщение является групповым и оно уже было получено хотя бы одним членом группы, отменить его нельзя.

Успешная отмена сообщения приводит к генерации системного сообщения *PDMSG_SENDCOMPLETE*, указывающего на завершение доставки.

Отмена сообщения с указанием его идентификационного номера возможна при использовании функции *IDirectPlay4::CancelMessage()*, прототип которой выглядит следующим образом:

```
HRESULT CancelMessage( DWORD dwMsgID, DWORD dwFlags )
```

Параметр *dwMsgID* является идентификационным номером отменяемого сообщения, генерируемым *DirectPlay*.

Параметр *dwFlags* является флагами, в данной версии не используются.

Если удаление невозможно, функция возвращает код ошибки *DPERR_CANCELFAILED*. Если указан неверный идентификационный номер сообщения, возвращается код ошибки *DPERR_UNKNOWNMESSAGE*.

Кроме отмены сообщения по его идентификационному номеру возможна отмена группы сообщений, имеющих определенный приоритет. Эту возможность реализует функция *IDirectPlay4::CancelPriority()*, прототип которой приведен ниже:

```
HRESULT CancelPrionty(
      DWORD dwMinPriority,
      DWORD dwMaxPriority,
      DWORD dwFlags. );
```

Параметры *dwMinPriority* и d*wMaxPriority* указывают диапазон приоритетов сообщений, отправка которых будет отменена. Второй параметр обязательно должен быть больше, чем первый. Для указания максимального приоритета можно использовать константу *MAXPRIORITY*.

Параметр *dwFlags* является флагами и не используется в данной версии.

*Получение сообщений*

Все получаемые приложением сообщения помещаются в очередь для последующей выборки. Забрать сообщение из очереди можно при помощи функции *IDirectPlay4::Receive()*, прототип и описание которой представлены ниже:

```
HRESULT Receive(
      LPDPID lpidFrom,
      LPDPID lpidTo.
      DWORD dwFlags,
      LPVOID lpData,
      LPDWORD lpdwDataSize );
```

Параметр *lpidFrom* является указателем на переменную, в которую будет помещен идентификационный номер игрока, отправившего сообщение. Если в качестве отправителя получено значение *DPID_SYSMSG*, сообщение является системным.

Параметр *lpidTo* является указателем на переменную, в которую будет помещен идентификационный номер игрока, которому предназначается сообщение.

Параметр *dwFlags* перечисляет флаги, уточняющие, как и какие именно сообщения будут получены (*DPRECEIVE_ALL* – все сообщения, *DPRECEIVE_PEEK* – полученное сообщение не будет удалено из очереди, *DPRECEIVE_TOPLAYER* и *DPRECEIVE_FROMPLAYER* – предписывают получить сообщения для игрока, указанного в параметре *lpidTo*, или от игрока, указанного в параметре *lpidFrom*, соответственно).

Параметр *lpData* является указателем на буфер, в который будет помещено полученное сообщение.

Параметр *lpdwDataSize* является указателем на переменную, в которую будет помещен размер буфера, необходимый для получения сообщения.

Ниже приведен пример использования функции *Receive()*.

```
while (true)
{
hRet=lpdp4->Receive(&recID, SfromID, DPRECEIVE_ALL, message, &bufsize);
if (hRet==DPERR_BUFFERTOOSMALL)
{
if (message!=NULL)
{
delete message:
 message=NULL;
}
message = new char [bufsize];
if (message==NULL)
{
MessageBox(NULL, "out of memory", "error", MB OK);
return (FALSE);
}
continue;
}:
if (fron)ID==DPID_SYSMSG)
AddSysMessage(niessage) else
AddMessage(Playerna[ne->lpszShortNameA, message):
if (hRet==DPERR_NOMESSAGES) return (TRUE);
if (FAILED(hRet)) return (FALSE);
}
```

В приведенном примере выборка сообщений происходит в бесконечном цикле, который имеет два условия выхода из него – получение всех сообщений из очереди или возникновение ошибки.

Первый вызов функции *Receive()* практически всегда приводит к ошибке *DPERR_BUFFERTOSMALL*, так необходимый размер буфера для получения сообщения не известен. Повторный вызов функции должен увенчаться успехом, так как перед этим был создан буфер требуемого размера, полученного через параметр *lpdwDataSize*.

После получения сообщения необходимо проверить, не является ли оно системным. Если параметр *lpidFrom* равен *DPID_SYSMSG*, значит, получено системное сообщение. В противном случае это сообщение было отправлено другим игроком.

Получить из очереди сообщения для определенного игрока можно, указав флаг *DPRECEIVE_TOPLAYER* и поместив в переменную, на которую указывает параметр *lpidTo*, идентификационный номер получателя. Если же необходимо получить сообщение от определенного игрока, указывается флаг *DPRECEIVE_FROMPLAYER*, а идентификационный номер игрока помещается в переменную, на которую указывает параметр *lpidFrom*.

### Задание на лабораторную работу

1. Освоить основные подходы программирования сетевых соединений, предоставляемых *DirectPlay*, и принципы работы *DirectPlay*.
2. На основе предложенного приложения «Чат» разработать новое приложение «Чат», в котором обработать еще несколько любых системных сообщений и создать сеанс с повышенной безопасностью.
3. Подготовить отчет о проделанной работе.

## Лабораторная работа №4 «Программирование сетевых соединений, предоставляемых DirectPlay»

### Цель работы

Получение практических навыков программирования сетевых соединений, предоставляемых DirectPlay.

### Теоретические сведения

Теоретический материал по работе с DirectPlay приведен в лабораторной работе №3.

### Задание на лабораторную работу

Разработать сетевое приложение в соответствии с вариантом выданного задания. Подготовить отчет о проделанной работе.

| Вариант | Задание |
|---|---|
| 1. | Передать от сервера клиенту изображение размером M*N пикселей. Произвести медианную фильтрацию полученного клиентом изображение и вернуть серверу. Отобразить на сервере исходное и полученное изображение. |
| 2. | Сгенерировать сервером две матрицы M*N и передать их клиенту, который должен произвести поэлементное сложение полученных матриц. Результат передать следующему клиенту, который должен произвести умножение всех элементов матрицы на коэффициент K. Вернуть конечный результат серверу и отобразить его. |
| 3. | Передать сервером текстовый файл длиной N байт. Полученный файл отредактировать клиентом и вернуть серверу. Отобразить конечный результат. |
| 4. | Подсчитать количество потерянных сообщений в загруженной сети в режиме передачи сообщений без гарантии доставки. |
| 5. | Отобразить перемещение клиентов в 2D пространстве на карте, расположенной на сервере. |

| | |
|---|---|
| 6. | Передать от сервера клиенту массив размером N. Произвести сортировку полученного клиентом массива и вернуть его серверу. Отобразить на сервере исходный и полученный массив. |
| 7. | Передать от сервера клиенту N текстовых файлов. Произвести на клиенте их объединение и вернуть результат серверу. Отобразить на сервере результат. |
| 8. | Передать от сервера клиенту текущее время. Произвести сравнение с текущим временем клиента, вернуть результат серверу. Отобразить на сервере результат. |

## Лабораторная работа №5 «Программирование сетевых соединений, предоставляемых DirectPlay, с использованием множества пользовательских сообщений»

### Цель работы

Получение практических навыков программирования сетевых соединений, предоставляемых DirectPlay.

### Теоретические сведения

Основной теоретический материал по работе с DirectPlay приведен в лабораторной работе №3.

При разработке сложных сетевых проектов для передачи данных по сети используется множество пользовательских сообщений. Обычно описание всех необходимых структур выделяется в отдельный модуль. Такой подход обеспечивает легкость добавления новых сообщений как на передающей стороне, так и на принимающей.

Принцип описания сообщений следующий:

1. номерам сообщений присваиваются имена;
2. создается заголовочная запись, которая содержит общие поля всех структур;
3. описываются сообщения в виде записей, первым полем которых обязательно должен быть номер сообщения.

Для обработки множества пользовательских сообщений на принимающей стороне, как правило, создается метод обработки пользовательских сообщений. Ниже приведен пример такого метода:

```
procedure ProcessUserMessage(pMsg:Pointer; size:integer);
```

Параметр *pMsg* является указателем на принятое пользовательское сообщение. Этот указатель следует привести к указателю на заголовочную запись и далее произвести идентификацию переданного сообщения путем сравнения поля заголовочной записи с именованным номером сообщения.

Файл описания сообщения выглядит следующим образом:

```
//именованные номера сообщений
const
```

```
      NET_MSG1 = 2000;
      NET_MSG2 = 2001;
      NET_MSG3 = 2002;
.............
      NET_MSGn = NET_MSG1+n;
Type
// заголовочное сообщение
      THeaderMsg = record
      Command:word;
// другие общие поля
      End;
// описание всех сообщений
      TMessage1=record
      Command:word;
// другие общие поля
      // остальные поля
      End;
      TMessage2=record
      Command:word;
// другие общие поля
      // остальные поля
      End;
.............
      TMessageN=record
      Command:word;
// другие общие поля
      // остальные поля
      End;
PTHeaderMsg = ^THeaderMsg;
PTMessage1 =^TMessage1;
PTMessage2 =^TMessage2;
...............
PTMessageN =^TMessageN;
```

Ниже приведен пример процедуры обработки полученного пользовательского сообщения:

```
procedure ProcessUserMessage(pMsg:Pointer; size:integer);
var
pHdr:PTHeaderMsg;
pMsg1: PTMesage1;
pMsg2: PTMesage2;
.............
pMsgN: PTMesageN;
begin
      pHdr := PTHeaderMsg (pMsg);
      case pHdr.Command of
            NET_MSG1:
            begin
                  pMsg1 := PTMessage1(pMsg);
                  // необходимые операции с сообщением
                  .............
            end;
            NET_MSG2:
            begin
                  pMsg2 := PTMessage2(pMsg);
                  // необходимые операции с сообщением
                  .............
            end;
.           ...........
            NET_MSGn:
```

```
            begin
                  pMsgN := PTMessageN(pMsg);
                  // необходимые операции с сообщением
                  ............
            end;
      end;
end;
```

На передающей стороне заполняются поля необходимого сообщения и происходит с использованием метода Send (или SendEx).

```
Var Msg1:TMessage1;
...
Msg1.Command = NET_MSG1;
Msg1.необходимое_поле = значение;
............
hRet := pDPlay.Send(PlayerID, flag1, flag2, @Msg1, sizeof(TMessage1));
............
```

**Задание на лабораторную работу**

Разработать сетевое приложение в соответствии с вариантом выданного задания, используя множество пользовательских сообщений (некоторые варианты предполагают использование API-функций на клиентской или серверной части). Подготовить отчет о проделанной работе.

| Вариант | Задание |
|---|---|
| 1. | Сетевое приложение-игра «Крестики-нолики». |
| 2. | Сетевое приложение-игра «Морской бой». |
| 3. | Сетевое приложение «Карточная игра Дурак». |
| 4. | Сетевое приложение-игра «Четыре». |
| 5. | Сетевое приложение-игра «Шашки». |
| 6. | Сетевое приложение-игра «Master Mind» |
| 7. | Сетевое приложение «Сетевая доска рисования» (предусмотреть смену цвета и размера пера клиентом, обновление состояния «доски» на всех подключенных клиентах). |
| 8. | Сетевое приложение управления курсором мыши удаленного компьютера, используя полученный вид удаленного экрана (предусмотреть возможность смены значка курсора и нажатие кнопок мыши на элементы управления удаленного компьютера). |
| 9. | Сетевое приложение удаленного администрирования (выключение компьютера, открытие/закрытие лотка CD, запуск/закрытие приложения). |
| 10. | Сетевое приложение для получения информации о выбранном клиенте (имя компьютера, имя пользователя, ОС, общие ресурсы и т.п.). |
| 11. | Сетевое приложение просмотра HTML-страниц, присылаемых клиентом. |
| 12. | Сетевое приложение синхронизации системного времени на рабочей станции (клиенте). |
| 13. | Сетевое приложение получения списка и завершения запущенных процессов на удаленном компьютере. |

# Курсовое проектирование по дисциплине «Информационные сети»

## Введение

Компьютерная сеть в настоящее время стала неотъемлемым атрибутом современного предприятия, инструментом для успешного ведения дел в условиях высокой конкуренции и насыщенности информационных потоков.

Современные вычислительные сети характеризуются много сегментной структурой, большим числом рабочих станций, наличием нескольких серверов (файловых, баз данных, печати), модемов, маршрутизаторов, мостов и т.п. Эффективное использование технологии клиент-сервер в таких сетях ставит ряд сложных задач перед администраторами и пользователями ЛВС. Важнейшими задачами является обеспечение требуемой производительности, пропускной способности сети и планирование ее мощности.

Производительность и пропускная способность ЛВС определяется рядом факторов: выбором серверов и рабочих станций, сетевого оборудования, операционных систем рабочих станций, серверов и их конфигураций, организацией распределенного вычислительного процесса, защиты, поддержания и восстановления работоспособности в ситуациях сбоев и отказов и т.п. Для решения задач оптимизации производительности и пропускной способности информационной сети предприятия или организации используются методы и средства проектирования и расчета.

## Основные цели и задачи оптимизации локальных сетей

Для обеспечения эффективного функционирования вычислительной сети, необходимо решить следующий ряд задач:

– сформулировать критерии эффективности работы сети. Чаще всего такими критериями служат производительность и надежность, для которых в свою очередь требуется выбрать конкретные показатели оценки, например, время реакции и коэффициент готовности, соответственно.

– определить множество варьируемых параметров сети, прямо или косвенно влияющих на критерии эффективности. Эти параметры действительно должны быть варьируемыми, то есть нужно убедиться в том, что их можно изменять в некоторых пределах. Так, если размер пакета какого-либо протокола в конкретной операционной системе устанавливается автоматически и не может быть изменен путем настройки, то этот параметр в данном случае не является варьируемым, хотя в других случаях он может относиться и к варьируемым. Другим примером может служить пропускная способность внутренней шины маршрутизатора, которая рассматриваться как параметр оптимизации только в том случае, если допускается возможность замены маршрутизаторов в сети. Все варьируемые параметры могут быть сгруппированы различным образом. На-

пример, параметры протоколов (максимальный размер кадра протокола Ethernet или размер окна неподтвержденных пакетов протокола TCP) или параметры устройств (размер адресной таблицы или скорость фильтрации моста, пропускная способность внутренней шины маршрутизатора). Параметрами настройки могут быть как устройства, так и программные средства, например, протоколы. Так, например, улучшить работу сети с медленными и зашумленными глобальными каналами связи можно, перейдя со стека протоколов IPX/SPX на протоколы TCP/IP или добиться значительных улучшений с помощью замены сетевых адаптеров на более качественные и дорогие;

– определить порог чувствительности для значений критерия эффективности. Так, производительность сети можно оценивать логическими значениями «Работает» – «Не работает», и тогда оптимизация сводится к диагностике неисправностей и приведению сети в любое работоспособное состояние. Другим крайним случаем является тонкая настройка сети, при которой параметры работающей сети (например, размер кадра или величина окна неподтвержденных пакетов) могут варьироваться с целью повышения производительности (например, среднего значения времени реакции) хотя бы на несколько процентов. Как правило, под оптимизацией сети понимают некоторый промежуточный вариант, при котором требуется выбрать такие значения параметров сети, чтобы показатели ее эффективности существенно улучшились.

Таким образом, можно предложить три различных трактовки задачи оптимизации.

Во-первых, приведение сети в любое работоспособное состояние. Обычно эта задача решается первой, и включает:

– поиск неисправных элементов сети – кабелей, разъемов, адаптеров, компьютеров;

– проверку совместимости оборудования и программного обеспечения;

– выбор корректных значений ключевых параметров программ и устройств, обеспечивающих прохождение сообщений между всеми узлами сети – адресов сетей и узлов, используемых протоколов, типов кадров Ethernet и т.п.

Во-вторых, грубая настройка – выбор параметров, резко влияющих на характеристики (надежность, производительность) сети. Если сеть работоспособна, но обмен данными происходит очень медленно (время ожидания составляет десятки секунд или минуты) или же сеанс связи часто разрывается без видимых причин, то работоспособной такую сеть можно назвать только условно, и она, безусловно, нуждается в грубой настройке. На этом этапе необходимо найти ключевые причины существенных задержек прохождения пакетов в сети. Обычно причина серьезного замедления или неустойчивой работы сети содержится в одном неверно работающем элементе или некор-

ректно установленном параметре, но из-за большого количества возможных вариантов поиск может потребовать длительного наблюдения за работой сети и громоздкого перебора вариантов. Грубая настройка во многом похожа на приведение сети в работоспособное состояние. Здесь также обычно задается некоторое пороговое значение показателя эффективности и требуется найти такой вариант сети, при котором это значение было бы не хуже порогового.

В-третьих, тонкая настройка параметров сети (собственно оптимизация). Если сеть работает удовлетворительно, то дальнейшее повышение ее производительности или надежности вряд ли можно достичь изменением только какого-либо одного параметра, как это было в случае полностью неработоспособной сети или же в случае ее грубой настройки. В случае нормально работающей сети дальнейшее повышение ее качества обычно требует нахождения некоторого удачного сочетания значений большого количества параметров.

Даже при тонкой настройке сети оптимальное сочетание ее параметров (в строгом математическом понимании термина "оптимальность") получить невозможно, да и не нужно. Нет смысла затрачивать колоссальные усилия по нахождению строгого оптимума, отличающегося от близких к нему режимов работы на величины такого же порядка, что и точность измерений трафика в сети. Достаточно найти любое из близких к оптимальному решений, чтобы считать задачу оптимизации сети решенной. Такие близкие к оптимальному решения обычно называют рациональными вариантами, и именно их поиск интересует на практике администратора сети или сетевого интегратора.

Поиск неисправностей в сети – это сочетание анализа (измерения, диагностика и локализация ошибок) и синтеза (принятие решения о том, какие изменения надо внести в работу сети, чтобы исправить ее работу).

Анализ – определение значения критерия эффективности (или, что одно и то же, критерия оптимизации) системы для данного сочетания параметров сети. Иногда из этого этапа выделяют подэтап мониторинга, на котором выполняется процедура сбора первичных данных о работе сети: статистики о количестве циркулирующих в сети кадров и пакетов различных протоколов, состоянии портов концентраторов, коммутаторов и маршрутизаторов и т.п. Далее выполняется этап собственно анализа, под которым в этом случае понимается более сложный и интеллектуальный процесс осмысления собранной на этапе мониторинга информации, сопоставления ее с данными, полученными ранее, и выработки предположений о возможных причинах замедленной или ненадежной работы сети. Задача мониторинга решается программными и аппаратными измерителями, тестерами, сетевыми анализаторами и встроенными средствами мониторинга систем управления сетями и системами. Задача анализа требует более активного участия человека, а также использования таких сложных средств как экспертные системы, аккумулирующие практический опыт многих сетевых специалистов.

Синтез – выбор значений варьируемых параметров, при которых показатель эффективности имеет наилучшее значение. Если задано пороговое значение показателя эффективности, то результатом синтеза должен быть один из вариантов сети, превосходящий заданный порог. Приведение сети в работоспособное состояние – это также синтез, при котором находится любой вариант сети, для которого значение показателя эффективности отличается от состояния «не работает». Синтез рационального варианта сети – процедура чаще всего неформальная, так как она связана с выбором слишком большого и очень разнородного множества параметров сети: типов применяемого коммуникационного оборудования, моделей этого оборудования, числа серверов, типов компьютеров, используемых в качестве серверов, типов операционных систем, параметров этих операционных систем, стеков коммуникационных протоколов, их параметров и т.д. Очень часто мотивы, влияющие на выбор «в целом», то есть выбор типа или модели оборудования, стека протоколов или операционной системы, не носят технического характера, а принимаются из других соображений – коммерческих, «политических» и т.п. Поэтому формализовать постановку задачи оптимизации в таких случаях просто невозможно.

Ниже основное внимание уделяется этапам мониторинга и анализа сети, как более формальным и автоматизируемым процедурам. В тех случаях, когда это возможно, даются рекомендации по выполнению некоторых последовательностей действий по нахождению рационального варианта сети или приводятся соображения, облегчающие его поиск.

**Критерии эффективности работы сети**

Все множество наиболее часто используемых критериев эффективности работы сети может быть разделено на две группы. Одна группа характеризует производительность работы сети, вторая – надежность.

Производительность сети измеряется с помощью показателей двух типов – временных, оценивающих задержку, вносимую сетью при выполнении обмена данными, и показателей пропускной способности, отражающих количество информации, переданной сетью в единицу времени. Эти два типа показателей являются взаимно обратными, и, зная один из них, можно вычислить другой.

*Время реакции*

Обычно в качестве временной характеристики производительности сети используется такой показатель как время реакции. Термин «время реакции» может использоваться в очень широком смысле, поэтому в каждом конкретном случае необходимо уточнить, что понимается под этим термином.

В общем случае, время реакции определяется как интервал времени между возникновением запроса пользователя к какому-либо сетевому сервису и получением ответа на этот запрос (рис. К1). Очевидно, что смысл и значение

этого показателя зависят от типа сервиса, к которому обращается пользователь, от того, какой пользователь и к какому серверу обращается, а также от текущего состояния других элементов сети – загруженности сегментов, через которые проходит запрос, загруженности сервера и т.п.

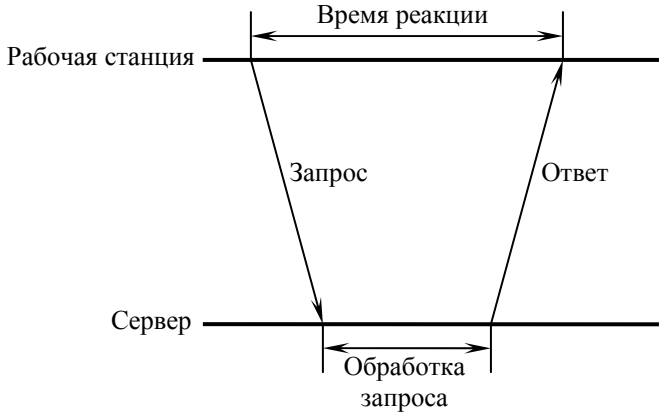


Рис. К1. Время реакции – интервал между запросом и ответом

Приведенный ниже рис. К2 иллюстрирует определение показателя «время реакции».

На рис. К2 цифрой 1 обозначено время реакции, которое проходит с момента обращения пользователя к сервису FTP для передачи файла с сервера 1 на клиентский компьютер 1 до момента завершения этой передачи (время реакции на прикладном уровне). Это время имеет несколько составляющих. Наиболее существенный вклад вносят такие составляющие времени реакции как: время обработки запросов на передачу файла на сервере, время обработки получаемых в пакетах IP частей файла на клиентском компьютере, время передачи пакетов между сервером и клиентским компьютером по протоколу Ethernet в пределах одного коаксиального сегмента. Можно было бы выделить еще более мелкие этапы выполнения запроса, например, время обработки запроса каждым из протоколов стека TCP/IP на сервере и клиенте.

Для конечного пользователя, таким образом, определенное время реакции является понятным и наиболее естественным показателем производительности сети (размер файла, который вносит некоторую неопределенность в этот показатель, можно зафиксировать, оценивая время реакции при передаче, например, одного мегабайта данных). Однако сетевого специалиста интересует в первую очередь производительность собственно сети, поэтому для более точной ее оценки целесообразно вычленить из времени реакции составляющие, соответствующие этапам несетевой обработки данных – поиску нужной информации на диске, записи ее на диск и т.п. Полученное в результате таких сокращений время можно считать другим определением времени реакции сети на прикладном уровне.

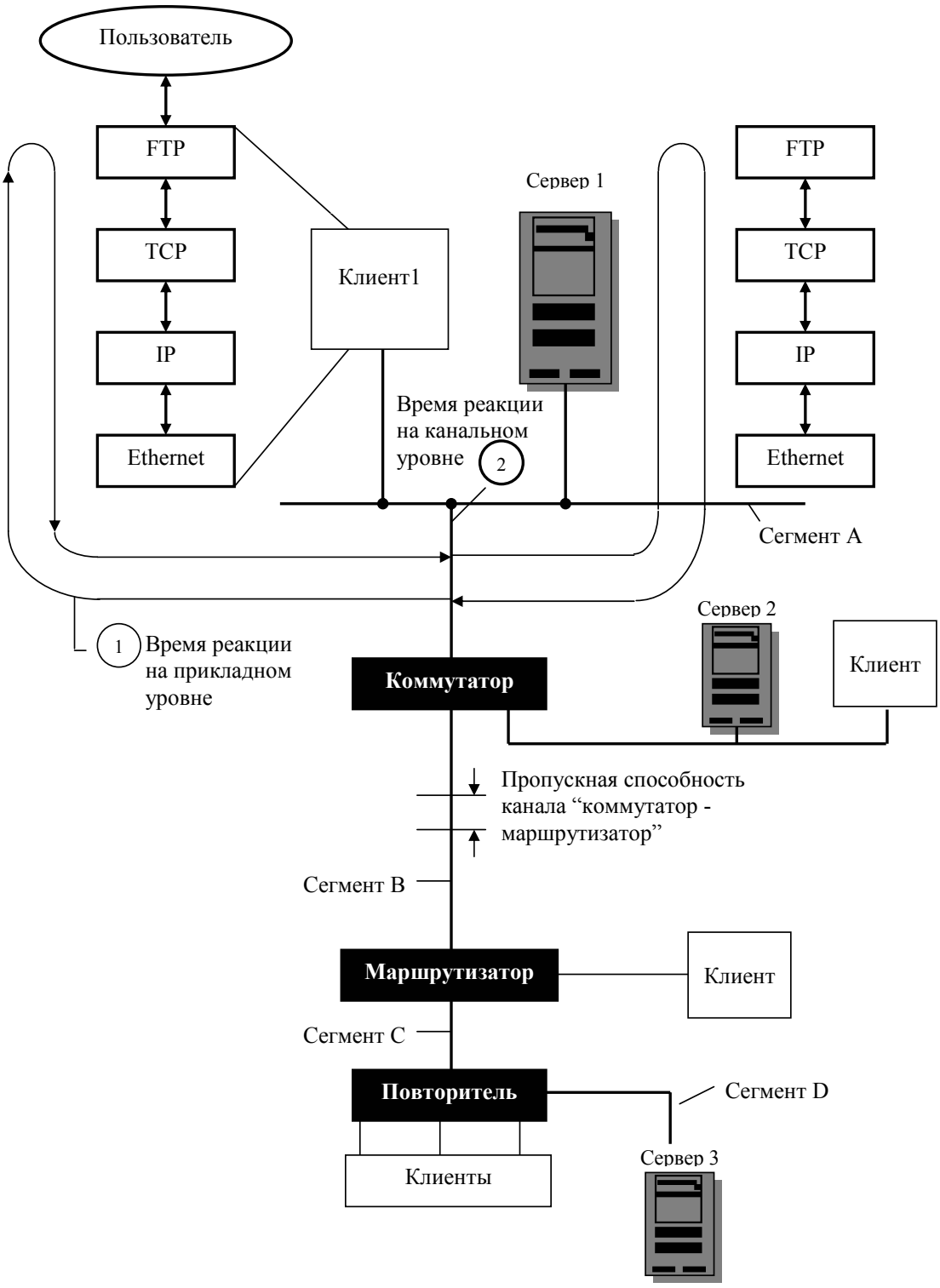


Рис. К2. Показатели производительности сети

Вариантами этого критерия могут служить времена реакции, измеренные при различных, но фиксированных состояниях сети:

– полностью ненагруженная сеть. Время реакции измеряется в условиях, когда к серверу 1 обращается только клиент 1, то есть на сегменте сети, объединяющем сервер 1 с клиентом 1, нет никакой другой активности – на нем присутствуют только кадры сессии FTP, производительность которой измеряется. Так как ненагруженный сегмент в реальной сети по-

лучить практически не возможно, то данный вариант показателя производительности имеет ограниченную применимость – его хорошие значения говорят только о том, что программное обеспечение и аппаратура данных двух узлов и сегмента обладают необходимой производительностью для работы в облегченных условиях. Для работы в реальных условиях, когда будет иметь место борьба за разделяемые ресурсы сегмента с другими узлами сети, производительность тестируемых элементов сети может оказаться недостаточной;

– нагруженная сеть. Это более интересный случай проверки производительности сервиса FTP для конкретного сервера и клиента. Однако при измерении критерия производительности в условиях, когда в сети работают и другие узлы, и сервисы, возникают свои сложности – в сети может существовать слишком большое количество вариантов нагрузки, поэтому главное при определении таких критериев – проведение измерений при некоторых типовых условиях работы сети. Так как трафик в сети носит пульсирующий характер, и характеристики трафика существенно изменяются в зависимости от времени дня и дня недели, то определение типовой нагрузки – процедура сложная, требующая длительных измерений на сети. Если же сеть только проектируется, то определение типовой нагрузки еще больше усложняется.

Цифрой 2 на рис. К2 обозначено время задержки, между передачей кадра Ethernet в сеть сетевым адаптером клиентского компьютера 1 и поступлением его на сетевой адаптер сервера 3. Этот критерий также относится к критериям типа «время реакции», но соответствует сервису нижнего – канального уровня. Так как протокол Ethernet – протокол дейтаграммного типа, то есть без установления соединений, для которого понятие «ответ» не определено, то под временем реакции в данном случае понимается время прохождения кадра от узла-источника до узла-получателя. Задержка передачи кадра включает в данном случае время распространения кадра по исходному сегменту, время передачи кадра коммутатором из сегмента *A* в сегмент *B*, время передачи кадра маршрутизатром из сегмента *B* в сегмент *C* и время передачи кадра из сегмента *C* в сегмент *D* повторителем. Критерии, относящиеся к нижнему уровню сети, хорошо характеризуют качества транспортного сервиса сети и являются более информативными для сетевых интеграторов, так как не содержат избыточную для них информацию о работе протоколов верхних уровней.

При оценке производительности сети не по отношению к отдельным парам узлов, а ко всем узлам в целом используются критерии двух типов: средневзвешенные и пороговые.

Средневзвешенный критерий представляет собой сумму времен реакции всех или некоторых узлов при взаимодействии со всеми или некоторыми серверами сети по определенному сервису, то есть сумму вида:

$$\left(\sum_i \sum_j T_{ij}\right) \Big/ (n \times m) \qquad (1)$$

где $T_{ij}$ – время реакции $i-го$ клиента при обращении к $j-му$ серверу, $n$ – число клиентов; $m$ – число серверов.

Если усреднение производится и по сервисам, то в приведенном выражении добавится еще одно суммирование – по количеству учитываемых сервисов. Оптимизация сети по данному критерию заключается в нахождении значений параметров, при которых критерий имеет минимальное значение или, по крайней мере, не превышает некоторое заданное число.

Пороговый критерий отражает наихудшее время реакции по всем возможным сочетаниям клиентов, серверов и сервисов:

$$max_{ijk}\, T_{ijk} \qquad (2)$$

где $i$ и $j$ имеют тот же смысл, что и в предыдущем случае, а $k$ обозначает тип сервиса. Оптимизация также может выполняться с целью минимизации критерия, или же с целью достижения им некоторой заданной величины, признаваемой разумной с практической точки зрения.

Чаще применяются пороговые критерии оптимизации, так как они гарантируют всем пользователям некоторый удовлетворительный уровень реакции сети на их запросы. Средневзвешенные критерии могут дискриминировать некоторых пользователей, для которых время реакции слишком велико при том, что при усреднении получен вполне приемлемый результат.

Можно применять и более дифференцированные по категориям пользователей и ситуациям критерии. Например, можно поставить цель гарантировать любому пользователю доступ к серверу, находящемуся в его сегменте, за время, не превышающее 5 секунд, к серверам, находящимся в его сети, но в сегментах, отделенных от его сегмента коммутаторами, за время, не превышающее 10 секунд, а к серверам других сетей – за время до 1 минуты.

*Пропускная способность*

Основная задача, для решения которой строится любая сеть – быстрая передача информации между компьютерами. Поэтому критерии, связанные с пропускной способностью сети или части сети, хорошо отражают качество выполнения сетью ее основной функции.

Существует большое количество вариантов определения критериев этого вида. Эти варианты могут отличаться друг от друга: выбранной единицей измерения количества передаваемой информации, характером учитываемых данных (только пользовательские или же пользовательские вместе со служебными), количеством точек измерения передаваемого трафика, способом усреднения результатов на сеть в целом. Рассмотрим различные способы построения критерия пропускной способности более подробно.

*Критерии, отличающиеся единицей измерения передаваемой информации*

В качестве единицы измерения передаваемой информации обычно используются пакеты (или кадры, далее эти термины будут использоваться как синонимы) или биты. Соответственно, пропускная способность измеряется в пакетах в секунду или же в битах в секунду.

Так как вычислительные сети работают по принципу коммутации пакетов (или кадров), то измерение количества переданной информации в пакетах имеет смысл, тем более что пропускная способность коммуникационного оборудования, работающего на канальном уровне и выше, также чаще всего измеряется в пакетах в секунду. Однако, из-за переменного размера пакета (это характерно для всех протоколов за исключением ATM, имеющего фиксированный размер пакета в 53 байта), измерение пропускной способности в пакетах в секунду связано с некоторой неопределенностью – пакеты какого протокола и какого размера имеются в виду? Чаще всего подразумевают пакеты протокола Ethernet, как самого распространенного, имеющие минимальный для протокола размер в 64 байта (без преамбулы). Пакеты минимальной длины выбраны в качестве эталонных из-за того, что они создают для коммуникационного оборудования наиболее тяжелый режим работы – вычислительные операции, производимые с каждым пришедшим пакетом, в очень слабой степени зависят от его размера, поэтому на единицу переносимой информации обработка пакета минимальной длины требует выполнения гораздо больше операций, чем для пакета максимальной длины.

Измерение пропускной способности в битах в секунду (для ЛВС более характерны скорости, измеряемые в мегабит в секунду – Мб/с) дает более точную оценку скорости передаваемой информации, чем при использовании пакетов.

*Критерии, отличающиеся учетом служебной информации*

В любом протоколе имеется заголовок, переносящий служебную информацию, и поле данных, в котором переносится информация, считающаяся для данного протокола пользовательской. Например, в кадре протокола Ethernet минимального размера 46 байт (из 64) представляют собой поле данных, а оставшиеся 18 являются служебной информацией. При измерении пропускной способности в пакетах в секунду отделить пользовательскую информацию от служебной невозможно, а при побитовом измерении – можно.

Если пропускная способность измеряется без деления информации на пользовательскую и служебную, то в этом случае нельзя ставить задачу выбора протокола или стека протоколов для данной сети. Это объясняется тем, что даже если при замене одного протокола на другой мы получим более высокую пропускную способность сети, то это не означает, что для конечных пользователей сеть будет работать быстрее – если доля служебной информа-

ции, приходящаяся на единицу пользовательских данных, у этих протоколов различная (а в общем случае это так), то можно в качестве оптимального выбрать более медленный вариант сети. Если же тип протокола не меняется при настройке сети, то можно использовать и критерии, не выделяющие пользовательские данные из общего потока.

При тестировании пропускной способности сети на прикладном уровне легче всего измерять как раз пропускную способность по пользовательским данным. Для этого достаточно измерить время передачи файла определенного размера между сервером и клиентом и разделить размер файла на полученное время. Для измерения общей пропускной способности необходимы специальные инструменты измерения – анализаторы протоколов или SNMP или RMON агенты, встроенные в операционные системы, сетевые адаптеры или коммуникационное оборудование.

*Критерии, отличающиеся количеством и расположением точек измерения*

Пропускную способность можно измерять между любыми двумя узлами или точками сети. При этом получаемые значения пропускной способности будут изменяться при одних и тех же условиях работы сети в зависимости от того, между какими двумя точками производятся измерения. Так как в сети одновременно работает большое число пользовательских компьютеров и серверов, то полную характеристику пропускной способности сети дает набор пропускных способностей, измеренных для различных сочетаний взаимодействующих компьютеров – так называемая матрица трафика узлов сети. Существуют специальные средства измерения, которые фиксируют матрицу трафика для каждого узла сети.

Так как в сетях данные на пути до узла назначения обычно проходят через несколько транзитных промежуточных этапов обработки, то в качестве критерия эффективности может рассматриваться пропускная способность отдельного промежуточного элемента сети – отдельного канала, сегмента или коммуникационного устройства.

Знание общей пропускной способности между двумя узлами не может дать полной информации о возможных путях ее повышения, так как из общей цифры нельзя понять, какой из промежуточных этапов обработки пакетов в наибольшей степени замедляет работу сети. Поэтому данные о пропускной способности отдельных элементов сети могут быть полезны для принятия решения о способах ее оптимизации.

В рассматриваемом примере (рис. К2) пакеты на пути от клиентского компьютера 1 до сервера 3 проходят через следующие промежуточные элементы сети: «Сегмент А», «Коммутатор», «Сегмент В», «Маршрутизатор», «Сегмент С», «Повторитель» и «Сегмент D». Каждый из этих элементов обладает определенной пропускной способностью, поэтому общая пропускная способность сети между компьютером 1 и сервером 3 будет равна минималь-

ной из пропускных способностей составляющих маршрута, а задержка передачи одного пакета (один из вариантов определения времени реакции) будет равна сумме задержек, вносимых каждым элементом. Для повышения пропускной способности составного пути необходимо в первую очередь обратить внимание на самые медленные элементы – в данном случае таким элементом, скорее всего, будет маршрутизатор.

Имеет смысл определить общую пропускную способность сети как среднее количество информации, переданной между всеми узлами сети в единицу времени. Общая пропускная способность сети может измеряться как в пакетах в секунду, так и в битах в секунду. При делении сети на сегменты или подсети общая пропускная способность сети равна сумме пропускных способностей подсетей плюс пропускная способность межсегментных или межсетевых связей.

*Надежность и отказоустойчивость вычислительных систем*

Важнейшей характеристикой вычислительной сети является надежность – способность правильно функционировать в течение продолжительного периода времени. Надежность имеет два основных аспекта: собственно надежность и готовность.

Повышение надежности заключается в предотвращении неисправностей, отказов и сбоев за счет применения электронных схем и компонентов с высокой степенью интеграции, снижения уровня помех, облегченных режимов работы схем, обеспечения тепловых режимов их работы, а также за счет совершенствования методов сборки аппаратуры. Надежность измеряется интенсивностью отказов и средним временем наработки на отказ. Надежность сетей как распределенных систем во многом определяется надежностью кабельных систем и коммутационной аппаратуры (разъемов, кроссовых панелей, коммутационных шкафов и т.п.), обеспечивающих собственно электрическую или оптическую связность отдельных узлов между собой.

Повышение готовности предполагает подавление в определенных пределах влияния отказов и сбоев на работу системы с помощью средств контроля и коррекции ошибок, а также средств автоматического восстановления циркуляции информации в сети после обнаружения неисправности. Повышение готовности представляет собой борьбу за снижение времени простоя системы.

Критерием оценки готовности является коэффициент готовности, который равен доле времени пребывания системы в работоспособном состоянии и может интерпретироваться как вероятность нахождения системы в работоспособном состоянии. Коэффициент готовности вычисляется как отношение среднего времени наработки на отказ к сумме этой же величины и среднего времени восстановления. Системы с высокой готовностью называют также отказоустойчивыми.

Основным способом повышения готовности является избыточность, на основе которой реализуются различные варианты отказоустойчивых архитектур. Вычислительные сети включают большое количество элементов различных типов, и для обеспечения отказоустойчивости необходима избыточность по каждому из ключевых элементов сети. Если рассматривать сеть только как транспортную систему, то избыточность должна существовать для всех магистральных маршрутов сети, то есть маршрутов, являющихся общими для большого количества клиентов сети. Такими маршрутами обычно являются маршруты к корпоративным серверам, серверам баз данных, Web-серверам, почтовым серверам и т.п. Поэтому для организации отказоустойчивой работы все элементы сети, через которые проходят такие маршруты, должны быть зарезервированы: должны иметься резервные кабельные связи, которыми можно воспользоваться при отказе одного из основных кабелей, все коммуникационные устройства на магистральных путях должны либо сами быть реализованы по отказоустойчивой схеме с резервированием всех основных своих компонентов, либо для каждого коммуникационного устройства должно иметься резервное аналогичное устройство.

Переход с основной связи на резервную или с основного устройства на резервное может происходить как в автоматическом режиме, так и вручную, при участии администратора. Очевидно, что автоматический переход повышает коэффициент готовности системы, так как время простоя сети в этом случае будет существенно меньше, чем при вмешательстве человека. Для выполнения автоматических процедур реконфигурации необходимо иметь в сети интеллектуальные коммуникационные устройства, а также централизованную систему управления, помогающую устройствам распознавать отказы в сети и адекватно на них реагировать.

Высокую степень готовности сети можно обеспечить в том случае, когда процедуры тестирования работоспособности элементов сети и перехода на резервные элементы встроены в коммуникационные протоколы. Примером такого типа протоколов может служить протокол FDDI, в котором постоянно тестируются физические связи между узлами и концентраторами сети, а в случае их отказа выполняется автоматическая реконфигурация связей за счет вторичного резервного кольца. Существуют и специальные протоколы, поддерживающие отказоустойчивость сети, например, протокол SpanningTree, выполняющий автоматический переход на резервные связи в сети, построенной на основе мостов и коммутаторов.

Существуют различные градации отказоустойчивых компьютерных систем, к которым относятся и вычислительные сети. Приведем несколько общепринятых определений:

– высокая готовность (highavailability) – характеризует системы, выполненные по обычной компьютерной технологии, использующие избыточные аппаратные и программные средства и допускающие время восстановления в интервале от 2 до 20 минут;

– устойчивость к отказам (faulttolerance) – характеристика таких систем, которые имеют в горячем резерве избыточную аппаратуру для всех функциональных блоков, включая процессоры, источники питания, подсистемы ввода-вывода, подсистемы дисковой памяти, причем время восстановления при отказе не превышает одной секунды;
– непрерывная готовность (continuousavailability) – это свойство систем, которые также обеспечивают время восстановления в пределах одной секунды, но в отличие от систем устойчивых к отказам, системы непрерывной готовности устраняют не только простои, возникшие в результате отказов, но и плановые простои, связанные с модернизацией или обслуживанием системы. Все эти работы проводятся в режиме online. Дополнительным требованием к системам непрерывной готовности является отсутствие деградации, то есть система должна поддерживать постоянный уровень функциональных возможностей и производительности независимо от возникновения отказов.

Так как сети обслуживают одновременно большое количество пользователей, то при расчете коэффициента готовности необходимо учитывать это обстоятельство. Коэффициент готовности сети должен соответствовать доле времени, в течение которого сеть выполняла с должным качеством свои функции для всех пользователей. Очевидно, что в больших сетях очень трудно обеспечить значения коэффициента готовности, близкие к единице.

Между показателями производительности и надежности сети существует тесная связь. Ненадежная работа сети очень часто приводит к существенному снижению ее производительности. Это объясняется тем, что сбои, и отказы каналов связи и коммуникационного оборудования приводят к потере или искажению некоторой части пакетов, в результате чего коммуникационные протоколы вынуждены организовывать повторную передачу утерянных данных. Так как в локальных сетях восстановлением утерянных данных занимаются, как правило, протоколы транспортного или прикладного уровня, работающие с тайм-аутами в несколько десятков секунд, то потери производительности из-за низкой надежности сети могут составлять сотни процентов.

### Расчет показателей эффективности вычислительной системы

Показатель эффективности (ПЭ) вычислительной сети (ВС) – это количественная характеристика ВС, рассматриваемая применительно к определенным условиям ее функционирования. При оценке эффективности ВС необходимо учитывать характеристики трудовой деятельности человека, взаимодействующего с ЭВМ и другими техническими средствами сети, т.е. сеть должна рассматривается как система «человек – машина».

Показатель эффективности ВС определяется процессом ее функционирования, и является функционалом от этого процесса. В общем виде:

$$W = W\left(t, L_n, L_{mn}, L_a, L_д, L_y\right) \qquad (3)$$

где $W$ – множество ПЭ сети, $t$ – время, $L_п, L_{mn}, L_а, L_д, L_у$ – множество параметров соответственно входящих потоков запросов на обслуживание пользователей $(L_п)$, технических и программных средств сети $(L_{mn})$, алгоритмов обработки и передачи информации в сети $(L_а)$, деятельности пользователей и администраторов $(L_д)$, условий функционирования сети $(L_у)$.

В свою очередь:

$$L_д = (L_т, L_в, L_н) \tag{4}$$

где $L_т, L_в, L_н$ – множество выходных показателей деятельности пользователей (и администраторов) ВС соответственно точных $(L_т)$, временных $(L_в)$, надежностных $(L_н)$.

Значения множеств $L_т, L_в, L_н$ определяются конкретными процессами деятельности пользователей и администраторов в рассматриваемой ВС, средствами, которые имеются в их распоряжении для выполнения своих функций, и условиями работы.

В соответствии с конкретизацией понятия эффективности показатели множества $W$ можно разделить на три группы:

$$W = (W_ц, W_т, W_э) \tag{5}$$

где $W_ц$ – показатели эффективности функционирования ВС, или эффективности использования (целевого применения) ВС – это количественная мера соответствия сети своему назначению; $W_т$ – показатели технической эффективности ВС – это количественная мера, отражающее техническое совершенство сети; $W_э$ – показатели экономической эффективности функционирования ВС – это количественная мера экономической целесообразности использования сети.

*Показатели целевой эффективности ВС*

Выбор показателей целевой эффективности сети определяется ее назначением, в связи, с чем имеет место большое многообразие показателей группы $W_ц$. С помощью этих показателей оценивается результат (целевой результат), получаемый за счет решения тех или иных прикладных задач на ЭВМ сети (с использованием общесетевых ресурсов – аппаратных, программных, информационных), а не с использованием других малоэффективных средств. Для количественной оценки этого эффекта могут применяться самые различные единицы измерения.

Примеры показателей целевой эффективности:

– точностные $(W_т)$, надежностные $(W_н)$ и временные $(W_в)$ показатели, применяемые в системах специального назначения для оценки использования в них сетевых структур. Например, прирост вероятности выпол-

нения некоторого задания, сокращение времени на выполнение этого задания, повышение точности решения некоторой задачи;

- временные показатели целевого использования сетевых структур в управлении народным хозяйством на различных его уровнях, характеризующие повышение оперативности управления;
- показатели целевой эффективности ВС при решении задач планирования народного хозяйства на различных его уровнях (отрасль, подотрасль, объединение, организация, фирма, предприятие и т.д.). Здесь могут быть две группы этих показателей: а) показатели использования ресурсов ВС для составления краткосрочных текущих планов. Эффект определяется тем, что разработка планов при этом осуществляется быстрее, точнее и позднее, с учетом большего количества факторов; б) показатели эффективности использования сетевых структур для составления долгосрочных (перспективных) планов. В этом случае эффект определяется не только тем, что разработанный с применением ВС перспективный план будет получен быстрее и окажется точнее и полнее, но что он вообще стал возможным благодаря использованию сетевых ресурсов;
- показатели, характеризующие повышение качества продукции, технология производства которой включает использование ВС (например, использование ЛВС на предприятиях);
- показатели, характеризующие экономику производства продукции с применением сетевых структур (например, повышение производительности труда, снижение ее себестоимости, увеличение доли экспортируемой продукции и т. д.), если цель использования ТВС заключается именно в улучшении характеристик производственно-хозяйственной деятельности предприятия или организации. В этом случае показатели целевой эффективности одновременно являются и показателями экономической эффективности.

*Показатели технической эффективности ТВС*

С помощью этих показателей оценивается эффективность ВС как сложной аппаратно-программно, информационной кибернетической системы при работе ее в различных режимах. При этом не принимается во внимание эффект, получаемый за счет реализации результатов решения задач (удовлетворения запросов) пользователей ВС. Показатели группы $W_m$ могут использоваться для количественной оценки эффективности всей сети, ее отдельных систем и подсистем, звеньев и узлов сети.

Для оценки технической эффективности сети целесообразно использовать следующие показатели:

- $V_{no}$ – пропускная способность сети, т. е. средний поток данных, фактически передаваемых через сеть (измеряется в Мбит/с). Этот показатель может использоваться как для оценки многомагистральной ВС, так и для

одномагистральной (например, локальной сети, где данные передаются по моноканалу). Следует отличать фактическую пропускную способность канала или линии связи от физической пропускной способности $V_к$, которая определяется возможностями и свойствами передающей среды и является одним из главных ее параметров. Очевидно, что величина $V_{пд}$ существенно зависит от физической пропускной способности канала или линии связи. Но она определяется многими другими факторами: используемыми методами доступа в передающую среду, загрузкой канала, способами управления сетью, качествами и возможностями сетевой операционной системы и т.д. Все эти факторы обуславливают потоки передаваемых данных и фактическую скорость их передачи, т.е. фактическую (а не физическую) пропускную способность канала;

– $T_{зс}$ – задержка в сети, вносимая в передачу данных пользователя, т.е. время доставки сообщения от отправителя к получателю;

– $V_ф$ – скорость передачи фреймов (коротких сообщений длиной 1000-2000 бит), т.е. количество фреймов, передаваемых в единицу времени по сети. Это дополнительный показатель, используемый в случае, когда поток данных (трафик) содержит в основном только короткие фреймы;

– $T_{зс} = f(V_ф)$ – зависимость времени задержки сообщения в сети от средней пропускной способности. Описание эффективности сети с помощью такой зависимости имеет большое значение, так как при увеличении загрузки сети (увеличение физического потока данных) пользователь должен ожидать больше времени для начала передачи своих данных.

Для оценки технической эффективности отдельных звеньев ТВС (узлов обработки, узлов связи, центров коммутации пакетов и т.д.), обслуживающих запросы пользователей сети, удобными оказываются следующие показатели:

1. Интегральная пропускная способность звена сети на отрезке времени $[0,t]$:

$$\delta_u(0,t) = n_0(0,t) \Big/ n_n(0,t), \qquad (6)$$

где $n_0(0,t)$, $n_n(0,t)$ – число запросов, соответственно обслуженных звеном сети на отрезке времени $[0,t]$ и поступивших на этом же отрезке. Параметр показывает, как в среднем звено сети справляется с обслуживанием входящего потока запросов от момента начала отсчета работы до некоторого момента $t$ (например, за смену, сутки, месяц).

2. Динамическая пропускная способность $\delta_д(\Delta t,t)$, представляющая собой отношение числа запросов $n_0(\Delta t,t)$, обслуженных звеном сети на сравнительно небольшом интервале $\Delta t$ к моменту времени $t$, к числу запросов $n_n(\Delta t,t)$, поступивших в звено на том же интервале и к тому же моменту $t$:

$$\delta_д(\Delta t,t) = n_0(\Delta t,t) \Big/ n_n(\Delta t,t) \qquad (7)$$

Динамическая пропускная способность позволяет судить о том, как звено сети справляется с обслуживанием входящего потока запросов на любом заданном (наиболее характерном) отрезке времени к любому текущему моменту. Она дает возможность отслеживать работу звена сети в динамике и вырабатывать рекомендации по обеспечению ритмичности его функционирования.

3. Среднее время реакции звена цепи на запрос пользователя – $T_p$. Оно складывается из времени ожидания обслуживания запроса и времени собственного обслуживания. Этот показатель очень важен для оценки эффективности системы обслуживания при работе в интерактивном режиме.

4. Максимально возможное число активных абонентов, т.е. абонентов, обращающихся с запросами на обслуживание в данный момент.

5. Коэффициент задержки обслуживания абонентов; это отношение среднего времени реакции на запрос абонента при максимальном количестве активных абонентов к этому же времени при минимальном их количестве.

Возможна ситуация, когда показатели технической эффективности звена сети одновременно являются и показателями целевой эффективности.

*Показатели экономической эффективности использования ТВС*

Для оценки экономической эффективности всей сети или отдельных ее элементов и звеньев могут использоваться две группы показателей: интегральные показатели и частные показатели.

С помощью интегральных показателей оценивается общий (суммарный, интегральный) эффект, а затем и интегральная экономическая эффективность ВС (элемента или звена сети) с учетом всех капитальных и текущих и текущих (эксплутационных) затрат и всей экономии за счет использования ВС, т.е. по всем источникам прямой и косвенной экономии и по отдельным ее видам. Частные показатели необходимы для оценки частного экономического эффекта, получаемого по отдельным источникам экономии, которые создаются при внедрении новых аппаратных, программных, информационных средств или новых технологий работы ВС.

В качестве интегральных показателей экономической эффективности ВС можно рекомендовать давно апробированные показатели:

– $Э_г$ – годовой экономический эффект, руб.;

– $\widetilde{Э}_г$ – среднегодовой экономический эффект, руб.;

– $Э_п$ – полный экономический эффект за расчетный период, руб.;

– $E_э$ – коэффициент экономической эффективности капитальных вложений (или единовременных затрат, имеющих характер капитальных вло-

жений) на создание и внедрение всей сети или отдельных ее элементов (звеньев) или на совершенствование и развитие сети, 1/год;

– $T_{ок}$ - срок окупаемости этих капитальных вложений, год.

Эти показатели могут быть как ожидаемыми (при априорной оценке), так и физическими (при апостериорной оценке).

Величина $Э_г$ определяется как разность приведенных затрат, связанных с созданием, совершенствованием и эксплуатацией некоторой системы (сети в целом, ее отдельных элементов и звеньев) для базового и рассматриваемого (исследуемого) вариантов. В качестве базовой выбирается система, которая аналогична (является прототипом) исследуемой системе по назначению, структуре, объему и характеру выпускаемой продукции или предоставляемых услуг и считается лучшей на данном этапе развития подобных систем. Однако в базовой системе отсутствуют новейшие средства и технологии, внедрение которых повышает ее эффективность. Рассматриваемая (исследуемая) система отличается от базовой использованием новейших средств и технологий, эффективность которых следует оценивать.

Приведенные затраты $З_п$ представляют собой сумму текущих затрат $С$ и капитальных вложений $К$, приведенных к одинаковой размерности с помощью нормативного коэффициента экономической эффективности капитальных вложений $E_н$:

$$З_п = C + E_н \cdot K \tag{8}$$

Следовательно,

$$Э_г = З_{п1} - З_{п2} = (C_1 + E_н \cdot K_1) - (C_2 + E_н \cdot K_2) = (C_1 - C_2) - E_н(K_2 - K_1) \tag{9}$$

где $З_{п1}, З_{п2}$ – годовые приведенные затраты соответственно для базового и исследуемого вариантов системы; $C_1, C_2$ – годовые текущие затраты для этих же вариантов системы; $K_2, K_1$ – капитальные вложения для базового и исследуемого вариантов системы.

Величины $E_э$ и $T_{ок}$ определяются по формулам

$$E_э = (C_1 - C_2) \Big/ (K_2 - K_1) \quad и \quad T_{ок} = 1 \Big/ E_э \tag{10}$$

Использование этой системы экономически целесообразно, если выполняются условия

$$E_э \geq E_н \; или \; T_{ок} \leq T_н \tag{11}$$

где $T_н$ – нормативный срок окупаемости капитальных вложений.

Расчет приведенных затрат по формуле (8), а следовательно, и расчет годового экономического эффекта по формуле (9) можно проводить только в простейшем случае, когда капитальные вложения осуществлены единовременно, а текущие затраты неизменны по времени. Более сложным и общим является случай, когда капитальные вложения осуществляются не единовре-

менно, а в течение определенного периода, а текущие затраты изменяются в течение срока службы исследуемой системы. Этот случай проводится к простейшему с помощью коэффициентов приведения.

Оценка частного экономического эффекта от внедрения новых аппаратных, программных, информационных средств или новых технологий ВС проводится с целью обоснования экономической целесообразности их внедрения (особенно тех средств и технологий, экономическая эффективность которых вызывает сомнение и которые вместе с тем не дают сколько-нибудь заметного целевого эффекта, ради которого можно было бы пожертвовать экономическим эффектом), сравнения конкурирующих вариантов внедряемых средств и технологий по частным показателям, поскольку в ряде случаев эти показатели имеют решающее значение при выборе того или иного варианта.

Частные показатели отличаются большим многообразием. Примеры частных показателей: сокращение численности обслуживающего персонала всей сети или отдельных ее систем, элементов, звеньев за счет внедрения новых средств и технологий; годовая экономия на текущих затратах за счет продления эффективного срока эксплуатации сети, вызванного совершенствованием профподготовки ее обслуживающего персонала; годовая экономия на текущих затратах за счет реализации мероприятий, направленных на улучшение условий труда обслуживающего персонала и, следовательно, способствующих повышению эффективности их трудовой деятельности и др.

**Расчет работоспособности сети**

Для определения работоспособности сети производят расчет суммарной задержки в сети по данным величина задержек каждого типа сегмента. Данные о спецификаций сети и длин ее отдельных участков должны быть известны. Для работоспособности сети необходимо, чтобы величина максимальной задержки не превышала 512 битовых интервалов.

Методика расчетов сводится к следующему:

1. В сети выделяется путь максимальной длины. Все дальнейшие расчеты ведутся для него. Если этот путь не очевиден, то расчеты ведутся для всех возможных путей, и на основании этих расчетов выбирается путь максимальной длины.
2. Если длина сегмента, входящего в выбранный путь, не максимальна, то рассчитывается двойное время прохождения в каждом сегменте выделенного пути по формуле: $t_s = Lt_1 + t_0$, где $L$ – длина сегмента в метрах (при этом учитывается тип сегмента: начальный, промежуточный или конечный).
3. Если длина сегмента равна максимально допустимой, то из таблицы для него берется величина максимальной задержки $t_m$.

4. Суммарная величина задержек всех сегментов выделенного пути не должна превышать предельной величины 512 битовых интервалов (51,2 мкс).
5. Выполняются те же действия для обратного направления выбранного пути (то есть конечный сегмент считается начальным, и наоборот). Из-за разных задержек передающих и принимающих узлов концентраторов величины задержек в разных направлениях могут отличаться.
6. Если задержки в обоих случаях не превышают величины 512 битовых интервалов, то сеть считается работоспособной.

Методику расчета работоспособности сети удобно показать на следующем примере (см. рис. К3).

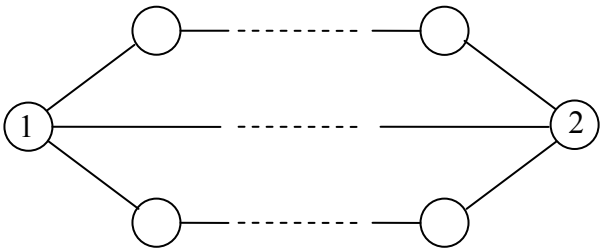


Рис. К3. Тестовая сеть

На данном рисунке видно, что в сети есть множество узлов. Произведем расчет суммарной задержки в сети для сегментов 1 и 2. Для расчета понадобятся данные о спецификации сети ($d_i$) и длин ее отдельных участков ($L_i$), а также величина задержек каждого типа сегмента ($t_j$). Следует учитывать то обстоятельство, что сегменты могут быть 3 типов: начальный, конечный и промежуточный. Следовательно, сегмент 1 будем считать начальным, сегмент 2 конечным, а сегменты обеспечивающие взаимосвязь этих сегментов – промежуточными. Для расчета необходимо определить количество сегментов, соединяющие начальный и конечный сегменты. Пусть количество сегментов $n$, тогда имеем:

$$L_1 d_1 + t_1 = r_1$$

…

$$L_n d_n + t_n = r_n$$

Далее необходимо произвести суммирование результатов $r_i$, $(i = 1...n)$:

$$R = \sum_{1}^{n} r_i$$

Если суммарная величина $R$ задержек всех сегментов выделенного пути не превысит предельной величины 512 битовых интервалов (51,2 мкс), то сеть на этом этапе считается работоспособной. Сеть считается полностью работоспособной, если все максимальные пути между сегментами удовлетворяют условию $R \le 512$.

**Тематика курсовых проектов**

В курсовом проекте, как правило, решается задача анализа, синтеза и расчета работоспособности сетей различных топологий и технологий представления и обработки информации в глобальной сети Internet. В проекте должны решаться следующие вопросы: технико-экономического обоснования региональных, административных, промышленных и непромышленных, офисных и других информационных систем для использования их в сети; разработки технического задания на проектирование с обоснованием требований к создаваемой сети; эскизной разработки сети с описанием структуры сети и выбранной топологии.

Кроме проектирования сети, задание на курсовое проектирование включает разработку программного продукта, нацеленного на сетевое использование. Вопросы разработки сетевых приложений рассматриваются в методических указаниях к лабораторным работам.

Содержание курсового проекта может быть разделено на следующие части:

– обследование объекта и формулирование целей создания сети;
– информационный поиск, нахождение аналогов, их критический анализ и выбор прототипа;
– определение перечня задач, которые будут решаться при помощи разрабатываемой сети;
– разработка функциональной структуры сети с описанием ее физического и логического функционирования;
– выбор программных средств для решения поставленной задачи;
– обоснование характеристик и выбор типа вычислительных средств;
– решение вопросов защиты информации и организация доступа пользователей.

Перечень решаемых в проекте вопросов и требуемая глубина их разработки уточняются в индивидуальном задании на курсовое проектирование.

В случае, если темой является углубленное научное исследование в каком либо узком направлении, преподаватель должен предложить студенту ввести в одну из частей его работы вопросы построения сети. По возможности тема проекта определяется с ориентацией на дальнейшее ее развитие в тему дипломного проекта. Сквозная тематика способствует большей результативности проведенных работ, что принесет очевидную пользу студенту и предприятию (кафедре).

**Содержание курсового проекта**

Пояснительная записка должна включать следующие обязательные части и материалы:

1 Титульный лист курсового проекта.

2 Задание на курсовой проект.
3 Титульный лист пояснительной записки.
4 Аннотацию.
5 Содержание.
6 Введение.
7 Анализ технического задания.
8 Системотехническая часть.
8.1 Разработка и проектирование информационной сети.
8.2 Расчет параметров сети.
9 Программная часть
9.1 Разработка программного продукта.
9.2 Тестирование программного продукта и результатов его работы.
9.3 Руководство программиста, пользователя, администратора.
10 Заключение.
11 Библиографический список.
12 Приложения (при необходимости).

Основная часть пояснительной записки включает в себя введение и разделы, отражающие исследовательскую и программную части, обоснование выбора языка программирования, определение ресурсов вычислительных устройств и его технических характеристик, обоснование выбора топологической сети. Особое внимание студентам следует уделять разделу, включающему в себя руководство программиста, пользователя и администратора, которые необходимо оформлять в соответствии с ГОСТ 2.105-95 или СТП 02068054-0101(/02/03)-99, ГОСТ 19.101-77, 19.102-77, 19.103-77, 19.701-90. Основная часть проекта завершается заключением.

**Подготовка доклада и защита проекта**

Оценка проекта осуществляется на основании открытой защиты курсового проекта перед комиссией, назначенной заведующим кафедрой. Сроки защиты определяются учебным планом. Для предоставления проекта студенту отводится 5-7 минут времени для доклада. В начале доклада подчеркивается актуальность работы, дается анализ возможных вариантов решений поставленной задачи и обосновывается выбранный вариант с учетом заданных условий. Далее дается описание разработанной сети, программы и доказывается необходимость ее разработки для конкретного предприятия. Затем студент докладывает о результатах разработки и практическом использовании выполненной работы. Доклад должен быть четким, лаконичным и кратким.

При оценке проекта учитывается качество пояснительной записки и графического материала, содержание доклада и ответы на вопросы, ритмичность работы над проектом. Хорошо выполненный проект комиссия может выдвинуть для участия в конкурсе «Студенческих работ», рекомендовать на

очередную студенческую научно-техническую конференцию в качестве доклада. После защиты пояснительная записка со сложенными по стандарту чертежами сдается руководителю проекта.

## Литература


1. Якубайтис Э.А. Информационные сети и системы: Справочная книга. – М.: Финансы и статистика, 1996.
2. Бэрри Нанс. Компьютерные сети пер. с англ. – М.: БИНОМ, 1996.
3. Основы современных компьютерных технологий под редакцией А.Д. Хомоненко– СПб КОРОНА принт, 1998.
4. Ресурсы Microsoft Windows NT Workstation 4.0 пер. с англ. яз. BNV – СПб, 1998.
5. Титтел Эд, Хадсон Курт, Дж. Майкл Стюард Networking Essentials – СПб ПИТЕР, 1999.
6. Титтел Эд, Хадсон Курт, Дж. Майкл Стюард TCP/IP – СПб ПИТЕР, 1999.
7. Компьютерные сети: Учебный курс Microsoft Corporation – М.: Издательский отдел «Русская редакция», 1999.
8. Глоссарий сетевых терминов http://www.bilim.com/koi8/library/glossary/
9. Орлов С.Б. Справочник Novell Netware 4, по заказу ИИЦ "Попурри", 1994. http://www.citforum.kts.ru/operating_systems/nw4/
10. CISCO Internetworking Technology Overview Сервер Марк-ИТТ, Владимир Плешаков http://www.citforum.ru/win/nets/ito/index.shtml.
11. Стэн Шатт Мир компьютерных сетей пер. с англ. – К.: BHV, 1996 – 288 с.: – ISBN 5–7733–0028–1.
12. Модель OSI Сервер BiLiM Systems Ltd.
13. http://www.citforum.ru/win/nets/switche/osi.shtml.
14. Руководство по сетям Ethernet для начинающих – http://www.citforum.ru/win/nets/ethernet/starter.shtml.
15. Базовые технологии локальных сетей http://www.citforum.ru/win/nets/protocols2/index.shtml.
16. Введение в IP-сети http://www.citforum.ru/win/nets/ip/contents.shtml
17. Практическое руководство по сетям Plug-and-Play Ethernet http://www.citforum.ru/win/nets/ethernet/pract.shtml.
18. Семейство протоколов TCP/IP http://www.citforum.ru/win/internet/tifamily/index.shtml.
19. Карпов Д. Статическая IP-маршрутизация, http://www.citforum.ru/win/internet/tifamily/iproutng.shtml.
20. Комер Д. Протоколы TCP/IP "Межсетевой обмен с помощью TCP/IP" http://www.citforum.ru/win/internet/comer/contents.shtml.

21. Усманов Р. Протокол IP http://www.citforum.ru/win/internet/tifamily/ipspec.shtml.
22. Операционные системы http://citforum.ru/operating_systems/index.shtml.
23. Концентраторы. http://www.idcom.ru/rationet/sysint/active.htm#nic.
24. Структурированные кабельные системы http://www.idcom.ru/rationet/sysint/cabsys.htm#magistral.
25. Типы соединений по витой паре http://ixbt.stack.net/comm/cable_utp.html.
26. Кабельные системы Ethernet http://www.bilim.com/koi8/bay/netgear/cables.htm.
27. Кабельные системы http://old.pcweek.ru/97_40/koi/re1.htm.
28. Физический уровень 100Base-FX - многомодовое оптоволокно http://www.citforum.ru/nets/protocols2/2_06_06.shtml.
29. Средства согласования протоколов на физическом и канальном уровнях http://www.citforum.ru/win/nets/tpns/glava_3.shtml.
30. Кабельные каналы http://www.idcom.ru/rationet/sysint/chanels.htm.
31. Олифер Н., Олифер В. Роль коммуникационных протоколов и функциональное назначение основных типов оборудования корпоративных сетей, ЦИТ http://www.citforum.ru/win/nets/protocols/index.shtml.
32. Олифер Н., Олифер В. Физическая структуризация локальной сети. Повторители и концентраторы, ЦИТ http://www.citforum.ru/win/nets/protocols/1_03_04.shtml.
33. Олифер Н., Олифер В. Сетевые операционные системы, ЦИТ, http://www.citforum.kcn.ru/operating_systems/sos/contents.shtml.
34. Кульгин М. Технология корпоративных сетей. – СПб ПИТЕР, 1999.
35. Пятибратов А.П. и др. Вычислительные системы, сети и телекоммуникации: Учебник / А.П. Пятибратов, Л.П. Гудынко, А.А. Кириченко: Под ред. А.П. Пятибратова – М.: Финансы и статистика, 1998. – 400 с. ил.
36. Олифер Н., Олифер В. Компьютерные сети: принципы, технологии, протоколы. – СПб: Питер, 2001. – 672 с.: ил.
37. Новиков Ю. В. Кондратенко С.В. – Локальные сети: архитектура, алгоритмы, проектирование. М.: Издательство ЭКОМ, 2000
38. Кулаков Ю.А., Луцкий Г.М.. Компьютерные сети. Учебное пособие. К.: Юниор, 1998.
39. Microsoft DirectX 7.0 SDK
40. Гончаров Д., Салихов Т. Direct 7.0 для программистов. Учебный курс. – СПб.: Питер, 2001.– 528 с.
41. Кирх О. Linux для профессионалов. Руководство администратора сети. – СПб: Питер, 2000. – 368 с.