**О.Я. Кравец**

# СЕТИ ЭВМ И ТЕЛЕКОММУНИКАЦИИ

## Учебное пособие

*Рекомендовано учебно-методическим объединением*
*по образованию в области прикладной информатики*
*в качестве учебного пособия для студентов высших учебных заведений,*
*обучающихся по специальности 080801 «Прикладная информатика*
*(по областям)» и другим экономическим специальностям*

Воронеж
«Научная книга»
2010

**УДК 681.3**
**ББК 32.973.202**
**К 78**

**Рецензенты:**
**Кафедра «Прикладная математика»** Липецкого государственного технического университета;
д-р техн. наук, профессор **С.Л. Подвальный**

**К 78** **Кравец, О.Я.** Сети ЭВМ и телекоммуникации: учеб. пособие/ О.Я.Кравец. - Воронеж: «Научная книга», 2010. – 224 с.

**ISBN 978-5-98222-661-7**

Учебное пособие охватывает круг вопросов, связанных с основами построения современных вычислительных систем и сетей телекоммуникаций. Рассмотрена концепция семиуровневой архитектуры открытых систем, дан краткий обзор методов оптимизации, проанализированы методы управления (в частности, коммутация пакетов и маршрутизация), приведены базовые сведения о сетевых операционных системах разного масштаба и гетерогенных сетях.

Предназначено для студентов, обучающихся по специальности 230100 "Вычислительные машины, комплексы, системы и сети", 080801 "Прикладная информатика (по областям)" при изучении дисциплин, связанных с сетями ЭВМ и средствами телекоммуникаций, а также может быть полезно и для студентов непрофильных специальностей в качестве введения в проблемы построения информационно-вычислительных сетей и методы управления ими.

УДК 681.3
ББК 32.973.202
К 78

**ISBN 978-5-98222-661-7** 

## Содержание

# Введение

В настоящее время теория вычислительных систем и информационно-телекоммуникационных сетей дает все более разнообразные практические приложения, появляются новые типы систем и сетей, совершенствуется технология обработки, передачи и хранения информации. Системы, имевшие ранее небольшое значение при решении задач управления технологическими процессами, в последние годы получили бурное развитие и заняли достойное место в иерархии современных средств обработки информации.

Информационно-вычислительные сети сегодня являются мощным средством обработки информации, обеспечивающим: большие, распределенные по объединению, предприятию информационно-вычислительные мощности; математические модели, базы данных, информационно-поисковые и справочные службы; эффективное коллективное использование имеющихся ресурсов; высокую надежность обработки информации благодаря резервированию и дублированию ресурсов; интегрированную передачу и обработку данных, речи, изображений; простые формы расширения сети, изменения ее конфигурации и характеристик [1, 21].

Настоящее учебное пособие является введением в современные методы анализа и синтеза высокоэффективных информационно-вычислительных систем, обеспечивающих выполнение экономически выгодных и динамичных процессов обработки, передачи и хранения информации. Издание может являться введением в современные методы анализа и синтеза высокоэффективных информационно-вычислительных систем, обеспечивающих выполнение экономически выгодных и динамичных процессов обработки, передачи и хранения информации. Заинтересованный читатель найдет много полезной дополнительной информации в библиографии. Издание в основном построено на материалах [2, 5, 7, 9, 10, 12, 13, 19].

Пособие предназначено прежде всего для студентов специальностей 220100 "Вычислительные машины, комплексы, системы и сети" и 351400 "Прикладная информатика (по отраслям)", изучающих одноименную дисциплину «Вычислительные системы и сети телекоммуникаций», и может быть полезно для студентов непрофильных специальностей в качестве введения в проблемы построения информационно-вычислительных сетей и методы управления ими.

# 1. Архитектура информационно-вычислительных систем

Рассматривая любой сложный объект, необходимо прежде всего определить поставленные перед ним задачи и выделить главные характеристики и параметры, которыми данный объект должен обладать. В информационно-вычислительной сети для этой цели служит общая модель, определяющая характеристики и функции как всей сети, так и входящих в нее основных компонентов. Описание рассматриваемой модели называют архитектурой информационно-вычислительной сети.

## *1.1. Эталонная модель взаимодействия открытых систем*

Международной организацией стандартизации (МОС) разработана эталонная модель взаимодействия открытых систем (ВОС, или, в некоторых источниках, ЭМВОС), включающая семь уровней: прикладной, представительный, сеансовый, транспортный, сетевой, канальный, физический.

Функции любого уровня определяются выбранными формами обработки и передачи информации. После построения модели обработки описываются необходимые протокольные блоки данных и процедуры, которые должны обеспечить их передачу. На этой основе формулируются требования к видам сервиса, который предоставляется рядом расположенным снизу уровнем. Таким образом, создаются протоколы управления уровнями. Задачи, поставленные перед протоколами уровней, и формы их решения рассматриваются ниже.

На рис. 1.1 представлены протоколы взаимодействия открытых систем, которые непосредственно связаны физическими средствами соединения. Таких пар систем три: А и ПС1, ПС1 и ПС2, ПС2 и Б. На верхних уровнях (прикладном, представительном, сеансовом и транспортном) протоколы определяют прямое (через промежуточные системы) взаимодействие объектов систем А и Б.

*Прикладной уровень* выполняет задачу обеспечения различных форм взаимодействия прикладных процессов, расположенных в одной либо нескольких системах. Протоколы, выполняющие эту задачу, объединяются в комплексы, описывающие наборы выполняемых информационных процессов. Такие комплексы называют функционально-ориентированными. В свою очередь, управление ими осуществляется особым протоколом, именуемым "управление контекстами".

Протокол управления контекстами обеспечивает выполнение операций, связанных с работой в информационно-вычислительной сети множества остальных протоколов верхних уровней, и предоставляет пользователям протоколы управления: терминалами; диалогом; файлами; задачами; системой; сетью; целостностью информации.

Протокол управления терминалами предоставляет удобные и эффективные средства, необходимые для обеспечения интерфейса с пользователями. Протокол управления диалогом обеспечивает возможность соединения прикладных процессов (ПрП) для проведения между ними обмена информацией в форме диалога. Управление удаленными массивами данных, их обслуживание, доступ к ним и передача их между ПрП описываются протоколом управления файлами. Протокол управления системой обеспечивает административное управление всеми уровнями системы. Протокол целостности информации обнаруживает ошибки, возникающие при передаче информации, разрешает возникающие спорные ситуации при передаче, выводит процессы из тупиковых состояний, обеспечивает проведение рестарта, выполняемого для устранения ошибок из-за неисправностей.


Рис. 1.1. Протоколы взаимодействия систем, связанных через промежуточные системы

Функции, выполняемые протоколами прикладного уровня, включают: описание форм и методов взаимодействия ПрП; управление заданиями, передачу файлов, управление системой; идентификацию пользователей; объявление о возможности доступа из сети к указываемым ПрП системы; посылку запросов на соединение с другими ПрП; подачу заявок представительному уровню на необходимые методы описания информации; управление данными, которыми обмениваются ПрП; определение доступности ПрП; синхронизацию взаимодействующих ПрП; определение качества обслуживания.

*Представительный уровень* выполняет задачу представления и преобразования данных, подлежащих передаче между ПрП. Для того, чтобы понять значение представительного уровня, введем некоторые определения. *Семантика* - смысловое значение передаваемых команд и получаемых на них ответов. *Синтаксис* - структура этих команд и ответов. Совокупное описание общей структуры данных наряду с множеством воз-

можных действий над ними принято называть образом представления (ОПр). Таким образом, представительный уровень имеет дело только с синтаксисом данных. Что же касается семантики, то она известна лишь прикладным процессам.

Задача преобразования данных связана с тем, что различные типы ЭВМ имеют разные операционные системы и большое разнообразие форм представления данных. Возникает необходимость введения стандартных форм представления, обеспечивающих выполнение всего многообразия ПрП. Такие виды представления данных принято называть виртуальными. Использование виртуальных форм представления данных позволяет обеспечивать взаимодействие между ПрП даже при отсутствии сведений о том, какие виды изображения данных используют партнеры. Представительный уровень обеспечивает для прикладного уровня следующие виды сервиса: выбор ОПр; преобразование ОПр; преобразование синтаксиса данных; форматирование данных.

Для реализации поставленной перед представительным уровнем задачи его протоколы выполняют большое число разнообразных функций. К ним относятся: передача запросов на установление сеансов; согласование между ПрП необходимого вида представления данных; описание форм представления данных; определение форм описания графического материала; представление речи; преобразование данных; засекречивание информации; передача запросов на прекращение сеансов. Синтаксис взаимодействия, приемлемый как для отправителя, так и для получателя, описывает: символы и их коды; целые числа; числа с плавающей запятой; файлы с фиксированной либо произвольной длиной; используемые способы уплотнения информации.

Протоколы представительного уровня наряду с основными функциями, описанными выше, определяют действия, связанные с управлением процессом выполнения этих функций. К таким действиям относятся активизация уровня (перевод его в рабочее состояние) и контроль ошибок, возникающих при передаче данных.

*Сеансовый уровень* выполняет задачу организации и проведения диалога между ПрП. Он обеспечивает пользователю иллюзию того, что ПрП выполняется не в нескольких, расположенных в различных местах, процессорах, а в одном мощном процессоре.

Инициатором сеанса является прикладной объект, который требует проведения сеанса и указывает представительному объекту адрес партнера. После этого представительный объект-отправитель обращается к сеансовому объекту, инициируя сеанс взаимодействия. В системе-получателе все происходит наоборот. Сеансовый объект предлагает представительному объекту принять участие в сеансе. В свою очередь, представительный объект обращается к прикладному объекту с предложением о сеансе.

Сеансовый уровень обеспечивает выполнение двух основных групп функций: обслуживание сеансов и обеспечение диалоговой формы передачи данных. Задачей первой группы функций является установление и ликвидация сеансового соединения (СеС), по которому передаются данные. Вторая группа обеспечивает управление потоками данных.

Основные функции, выполняемые сеансовым уровнем, заключаются в следующем: установление СеС; обмен данными; управление взаимодействием; синхронизация СеС; извещение об исключительных ситуациях; отображение СеС на транспортное соединение; завершение СеС.

Установление СеС позволяет представительным объектам начать сеанс их взаимодействия и обеспечивает выбор параметров (скорость передачи, необходимость подтверждения запросов и т. д.).

Управление взаимодействием объектов позволяет определить, чья в данный момент очередь выполнять определенные операции сеансового взаимодействия. Стандарты задают три формы взаимодействия объектов во время сеанса: дуплексную (диалог), полудуплексную (диалог) и симплексную (монолог).

Синхронизация СеС позволяет устанавливать и находить точки синхронизации процесса взаимодействия объектов во время сеанса.

Сущность отображения СеС на транспортное соединение заключается в следующем. Во-первых, через одно и то же транспортное соединение могут последовательно передаваться данные, относящиеся к различным сеансам. Во-вторых, один и тот же сеанс может последовательно осуществляться по нескольким транспортным соединениям.

Завершение СеС позволяет представительным объектам так закончить сеанс, чтобы не пропали блоки данных, находящиеся в пути.

Для взаимодействующих ПрП, расположенных в одной и той же системе, сеансовый уровень является самым нижним. Что касается транспортного, сетевого, канального и физического уровней, то они необходимы для взаимодействия тех ПрП, которые находятся в различных системах.

*Транспортный уровень* выполняет задачу предоставления сквозных соединений прикладным объектам. Для выполнения указанной задачи транспортный уровень осуществляет передачу данных между системами сквозь все имеющиеся в сети физические средства соединения.

При создании транспортного уровня должна быть обеспечена полная его независимость от типа и характера взаимодействующих ПрП. Предоставляемые уровнем соединения являются прозрачными, т.е. по ним могут передаваться любые используемые коды и осуществляться всевозможные методы организации диалога на сеансовом уровне.

Для осуществления эффективной передачи данных транспортный уровень обеспечивает несколько классов обслуживания, учитывающих все разнообразные требования к транспорту информации, предъявляемые

различными ПрП. Классы сервиса характеризуются выбранными множествами параметров (пропускная способность, время передачи, время установления соединения, допустимая частота ошибок и т.д.).

Сервис, предоставляемый транспортным уровнем, включает: установление и разъединение транспортных соединений (ТрС); обеспечение взаимодействия СеС с ТрС; управление последовательностью и обеспечение целостности блоков данных, передаваемых через ТрС; обнаружение ошибок, их частичную ликвидацию, сообщение о неисправленных ошибках; восстановление соединения после появления неисправности; укрупнение либо разукрупнение передаваемых блоков данных; управление потоками транспортных блоков; предоставление приоритета в передаче блоков (нормальная и срочная передача); присылку подтверждений о принятых блоках; сброс блоков с ТрС при тупиках.

Благодаря выполнению этих функций транспортный уровень обеспечивает адаптацию системы к любому механизму передачи данных через конкретные физические средства соединения. Более того, транспортный уровень восстанавливает блоки данных, потерянные на уровнях 1-3. Если в физических средствах соединения создается несколько путей доставки блоков данных системе-получателю, то транспортный уровень при отказе одного из сетевых соединений может выбрать другие пути. Причем это он делает так, что прикладной процесс не знает о проводимых переключениях.

Функционирование уровня происходит в трех сменяющих друг друга фазах: установление ТрС; передача данных; завершение соединения. Транспортный уровень использует две стратегии передачи данных, выполняемых на сетевом уровне: дейтаграммы и виртуальные каналы (соединения). Дейтаграммой является блок данных, который передается транспортным уровнем без организации соединения. Последовательность блоков данных может достичь получателя не в том порядке, в котором она была отправлена.

*Виртуальным каналом* называют соединение между транспортным объектом-отправителем и транспортным объектом-получателем, предоставляемое сетевым уровнем. В блоках данных, передаваемых по виртуальным каналам, нет явных адресов отправителя и получателя. Они заключены в номерах каналов.

*Сетевой уровень* выполняет задачу ретрансляции данных, осуществляемой через одну либо несколько систем. Выполнение этой задачи обеспечивает транспортным объектам независимость от методов и средств коммутации и прокладки маршрутов в физических средствах соединения. Создаваемые в рассматриваемых средствах каналы связывают систему-отправитель и систему-получатель.

Основными видами сервиса, предоставляемого сетевым уровнем, являются: организация сетевых соединений (СтС), прокладываемых через

физические средства соединения; идентификация конечных точек СтС; передача блоков данных; обнаружение и извещение об ошибках; управление потоками блоков данных; ликвидация СтС; обеспечение последовательной доставки блоков данных.

При передаче блоков данных сетевой уровень осуществляет исправление многих видов возникающих при этом ошибок. К ним относятся: искажение данных, потеря блоков данных, дублирование блоков данных, нарушение последовательностей блоков данных, передача блоков данных не по назначению. Дублирование блоков данных возникает в тех случаях, когда после передачи блока пришедшая команда ошибочно принята за сигнал его потери. В этом случае блок передается вторично, т. е. происходит его ненужное дублирование. Сетевые блоки данных принято также называть пакетами.

*Канальный уровень* предназначен для передачи блоков данных через физические соединения. Благодаря этому сетевой уровень не зависит от типов физических соединений, используемых в информационно-вычислительной сети. Канальный уровень обеспечивает средства для установления, поддержания и разъединения канальных соединений между сетевыми объектами. Каждый из этих объектов в заданных пределах может динамически управлять скоростью передачи блоков данных по канальным соединениям. Канальный уровень обеспечивает виды сервиса: передачу блоков данных; идентификацию конечных пунктов канальных соединений; организацию последовательностей передачи блоков данных; обнаружение и исправление ошибок; оповещение об ошибках, которые не исправимы на канальном уровне; управление потоком через физические соединения; выбор параметров качества сервиса. Канальные блоки часто именуют *кадрами*.

Функции канального уровня таковы: использование физических соединений; установление и разъединение канальных соединений; обнаружение и устранение ошибок в них; управление потоками данных в этих соединениях; организация последовательностей передачи канальных блоков данных; обеспечение прозрачности соединений.

*Физический уровень* предназначен для сопряжения систем с физическими средствами соединения. Для выполнения указанной задачи уровень определяет механические, электрические, функциональные и процедурные характеристики, описывающие доступ к физическим соединениям. По этим соединениям, связывающим канальные объекты, передаются последовательности бит. При передаче обеспечивается прозрачность соединения, т.е. способность его передавать информацию, использующую любые наборы символов и закодированную любым способом. Физические соединения могут быть постоянными либо временными.

Следует выделить два типа физических соединений: двухточечные и многоточечные. Двухточечным является (рис. 1.2, *а*) соединение, свя-

зывающее две системы (А,Б). Если же физическое соединение обеспечивает взаимодействие (рис. 1.2, *б*) более двух систем (А, Б, ..., К), то его именуют многоточечным.

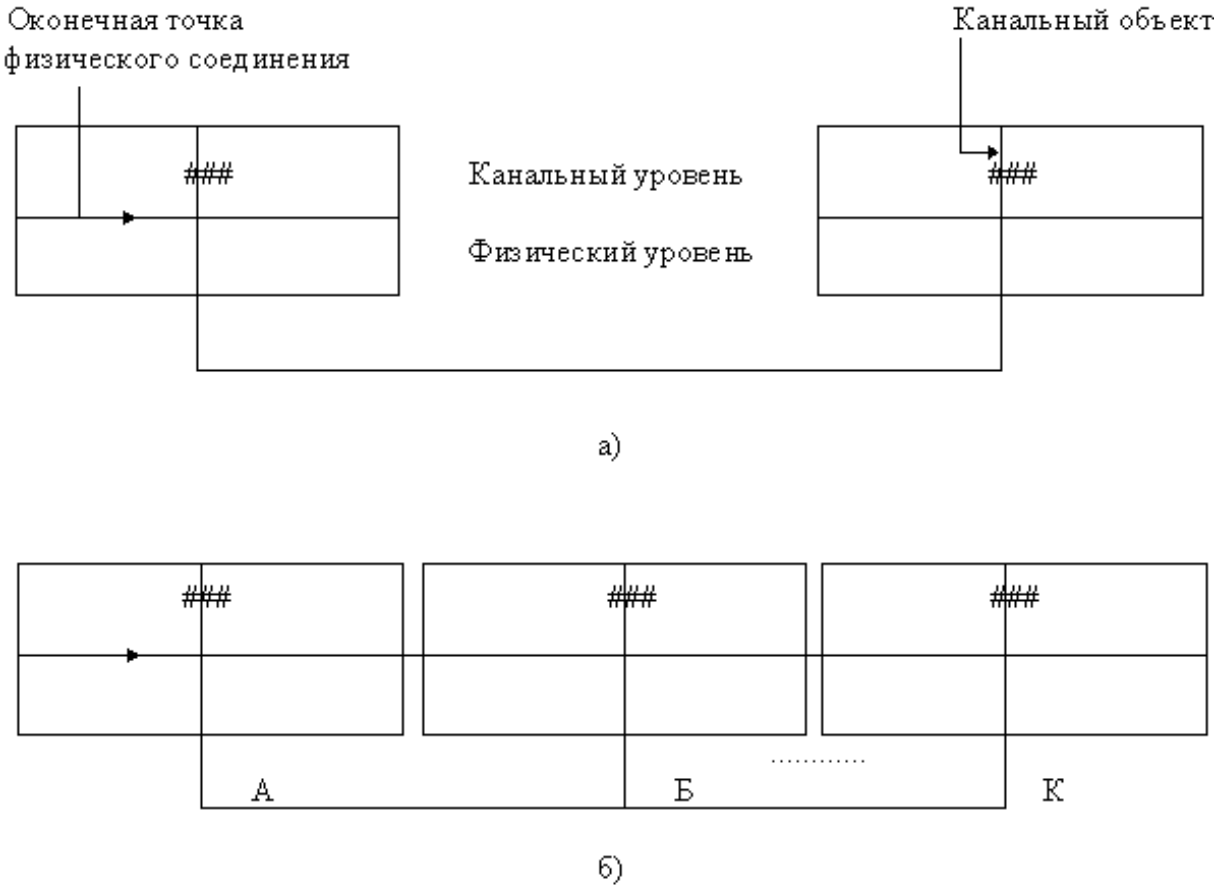



Рис. 1.2. Типы физических соединений: *а* - двухточечное; *б* - многоточечное

Физический уровень обеспечивает следующие виды сервиса: установление временных либо постоянных физических соединений; предоставление физических оконечных пунктов соединений; идентификацию физических соединений; организацию последовательностей передачи бит; оповещение об отказе получателя; установление параметров качества сервиса.

Таким образом, уровни 1-6 информационно-вычислительной сети обеспечивают нарастающий сервис, предоставляемый ПрП. Благодаря этому выполняются все необходимые формы взаимодействия ПрП, распределенных по сети и находящихся в различных системах.

На рис. 1.3 представлена примерная структура сквозного сервиса, предоставляемого уровнями прикладным процессам начиная от физических средств соединения и заканчивая абонентом, взаимодействующим с программным обеспечением.

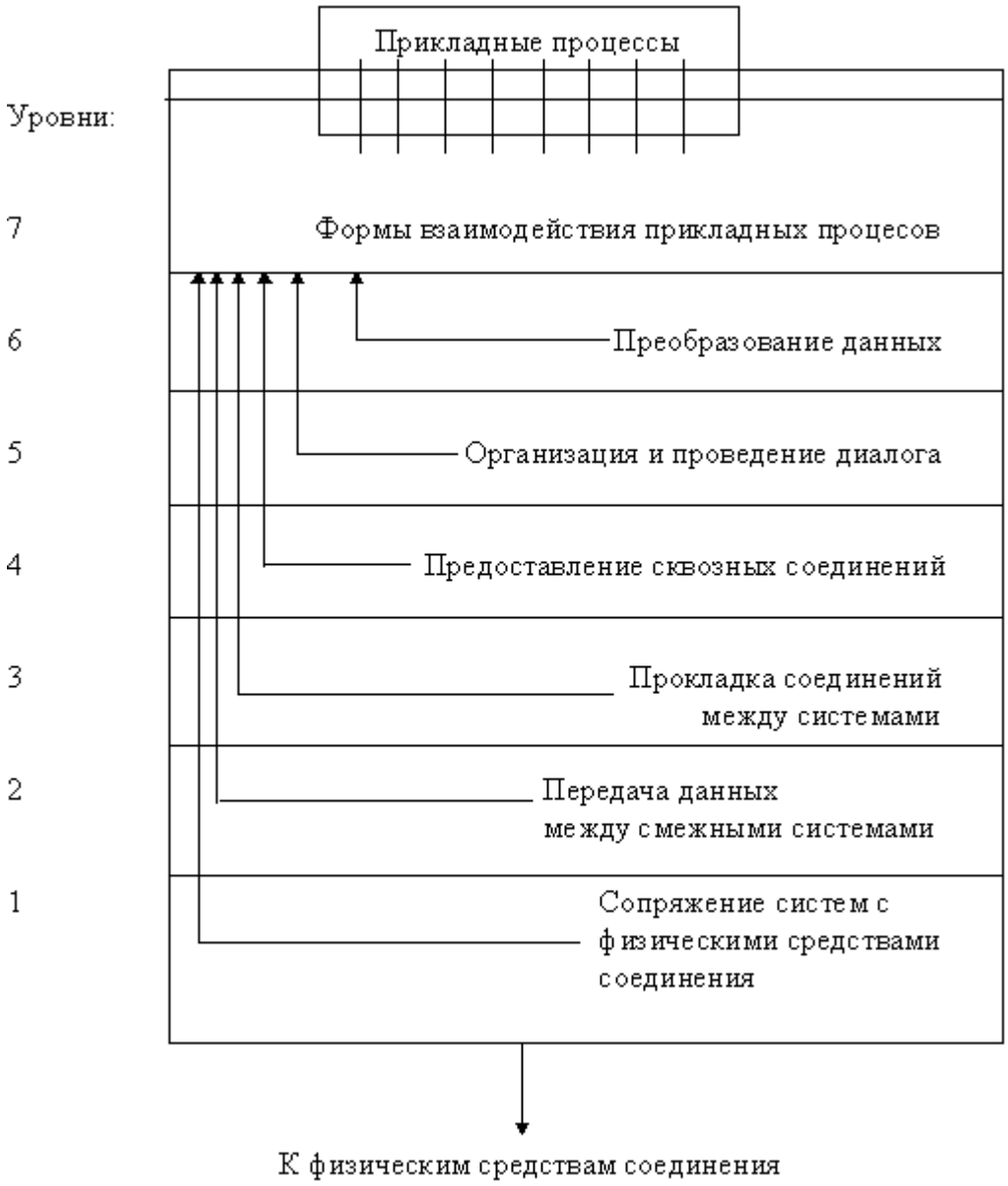



Рис. 1.3. Структура сервиса уровней

### *1.2. Классификация и параметры сетей*

Назовем *абонентами* объекты, генерирующие или потребляющие информацию. В их число входят: ЭВМ и комплексы этих машин; устройства оперативной и внешней памяти; терминалы (дисплеи, графопостроители, принтеры); телетайпы; копировальные и факсимильные аппараты; телевизионные камеры и мониторы; телефоны и диктофоны; роботы, автоматические станки и механизмы; испытательные приборы и т.д. Каждый абонент сопрягается со *станцией* - аппаратом, который выполняет вспомогательные функции, связанные с передачей информации. Совокупность абонента и станции называется *абонентской системой*.

Для обеспечения взаимодействия абонентов необходима *физическая среда*. Она образуется использованием пространства либо материала, свойства которого обеспечивают распространение сигналов, передающих необходимую информацию. В понятие физической среды также включается аппаратура передачи данных, непосредственно связанная с указанным пространством либо материалом. На базе физической среды строится *коммуникационная подсеть*, предназначенная для передачи информации между абонентскими системами. Ассоциацию абонентских систем и коммуникационной подсети назовем *информационно-вычислительной сетью* (ИВС).

В зависимости от размеров ИВС делятся на три вида (рис. 1.4). *Локальной* называется сеть, абоненты которой находятся на небольшом расстоянии друг от друга. Обычно локальные сети охватывают одно либо несколько рядом расположенных зданий. *Региональная* сеть связывает абонентов, расположенных на значительном расстоянии друг от друга. Она включает абонентов города, района, области и даже небольшой страны. *Глобальная* ИВС объединяет абонентов, расположенных в различных странах или на разных континентах. Чаще всего глобальная сеть строится на базе спутников.

В отличие от систем телеобработки, в ИВС используется большое число ЭВМ. Поэтому в сетях широко выполняются и новые информационные задачи. Среди них в первую очередь необходимо отметить следующие: распределение ресурсов; электронная почта; сетевые совещания.

ИВС характеризуется многими параметрами (табл. 1.1). Первые три параметра (стоимость, надежность и ремонтопригодность) являются основными; следующие два (защита, потеря) определяют гарантии, связанные с информацией, и, наконец, последние три (связность, доступность, преобразование) описывают сервис взаимодействия с прикладными процессами.

Локальные ИВС можно классифицировать по числу и типам используемых абонентских систем (рис. 1.5). Многосистемные сети делятся на открытые и однородные. *Открытая сеть* соответствует Базовой эталонной модели взаимодействия открытых систем и поэтому обеспечивает взаимодействие ЭВМ любых объединений и фирм. Естественно, что ЭВМ, входящие в открытую сеть, должны выполнять набор стандартных для сети протоколов. Открытая информационно-вычислительная сеть в соответствии с указанной моделью всегда имеет распределенное управление. Поэтому в ней нет центральной системы, управляющей передачей данных в сети.

Рис. 1.4. Классификация ИВС по протяженности

Таблица 1.1

Параметры, характеризующие информационно-вычислительную сеть

| **Параметр** | **Характеристика** |
|---|---|
| Стоимость | Затраты на создание сетеобразующих программного обеспечения и оборудования; их эксплуатация |
| Надежность | Обнаружение и исправление ошибок, минимизация отказов |
| Ремонтопригодность | Регистрация, локализация и устранение неисправностей |
| Защита | Методы пресечения несанкционированного доступа к ресурсам |
| Потеря | Средства обеспечения гарантии доставки блоков информации |
| Связность | Размер блоков информации, максимальные расстояния между системами, скорость передачи |
| Доступность | Управляемый доступ к ресурсам, мультиплексирование, многоканальность |
| Преобразование | Обеспечение совместной работы абонентов, функционирующих с различными скоростями |

Однородные информационно-вычислительные сети в зависимости от наличия центральной абонентской системы делятся на две группы.

К первой из них относятся сети с *централизованным управлением*. Каждая из таких сетей имеет *центральную систему*, управляющую работой всей сети. Сети с централизованным управлением отличаются простотой обеспечения функций взаимодействия между системами и основаны на том, что большая часть информационно-вычислительных ресурсов находится в центральной системе. Однако они очень ненадежны и малопригодны в тех случаях, когда информационно-вычислительные ресурсы равномерно распределены по большому числу абонентских систем в сети. Поэтому чаще всего на практике используются сети с децентрализованным управлением. Что касается сетей с централизованным управлением, то

они применяются лишь в тех случаях, когда в сети должно быть небольшое количество абонентских систем.

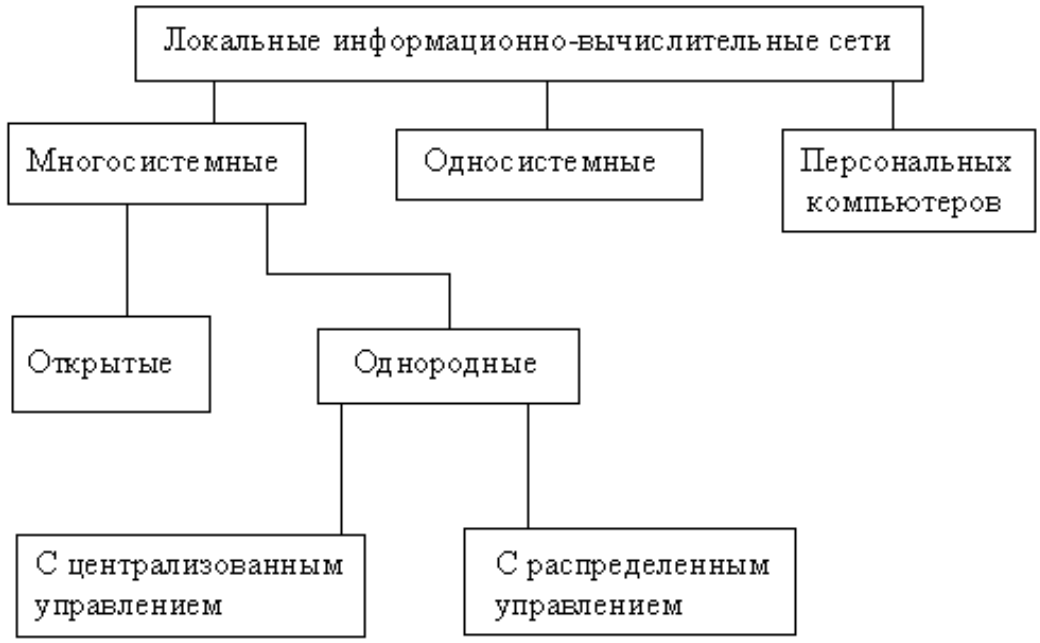



Рис. 1.5. Классификация информационно-вычислительных сетей

Вторую группу однородных информационно-вычислительных сетей образуют *сети с распределенным управлением*. В этих сетях нет центральной системы и функции управления распределены между системами сети. Однако для того, чтобы проводить диагностику, собирать статистику и выполнять ряд других административных функций, в сети используется специальная абонентская система либо прикладной процесс в такой системе. Для того, чтобы двум системам обменяться блоками данных, здесь не требуется чье-нибудь разрешение.

### *1.3. Коммуникационные подсети*

Издавна для передачи информации использовались различного вида узлы коммутации. Благодаря переходу на микропроцессорную технику и сверхбольшие интегральные схемы надежность узлов значительно возросла - и они превращаются в недорогие малогабаритные необслуживаемые аппараты. Идея многочисленных соединений также известна давно и применялась ранее - для подключения взаимодействия равноправных абонентов, а микроминиатюризация радиоэлектронной аппаратуры дала толчок для создания нового класса сетей.

### 1.3.1. Общая характеристика коммуникационных подсетей

*Коммуникационная подсеть* представляет собой совокупность физической среды, программных и аппаратных средств, обеспечивающих передачу информации между группой абонентских систем. Рассматриваемая подсеть является важным компонентом ИВС. В соответствии с этим к ней предъявляются требования, основные из которых сводятся к следующим: высокая надежность передачи блоков данных; небольшая стоимость передачи; высокая скорость передачи; износоустойчивость и долговечность оборудования; малые потери информации; минимальный штат обслуживания; передача данных, закодированных любым способом.

Любая коммуникационная подсеть предназначена для обеспечения различных форм взаимодействия абонентских систем друг с другом. Точки подключения систем к рассматриваемой сети определяются *интерфейсом* коммуникационной подсети. Для всех абонентских систем этот интерфейс один и тот же. Однако в последнее время в коммуникационную подсеть стали включать дополнительные функции, связанные с преобразованием нестандартных интерфейсов в интерфейс коммуникационной подсети. Такие подсети именуются *интеллектуальными*.

Коммуникационную подсеть определяют четыре основные характеристики: трафик, надежность передачи, время установления сквозного (через подсеть) соединения, скорость передачи блоков данных.

В соответствии с определением коммуникационной подсети выделяют пять ее типов: одноузловая, многоузловая, моноканальная, поликанальная, циклическое кольцо. Эта классификация определяется характером доставки блоков данных от абонентской системы-отправителя к абонентской системе-получателю. Что же касается топологии, то указанные типы подсетей могут иметь одинаковую форму. Так, кольцевую форму могут иметь многоузловая подсеть, моноканал и циклическое кольцо.

В коммуникационной подсети следует различать два понятия скорости передачи. Первое из них - физическая скорость передачи данных по каналу. Она определяется числом бит, передаваемых в секунду по конкретному каналу. Вторая скорость именуется сквозной. Она характеризуется числом блоков данных в секунду, передаваемых между рассматриваемой парой точек интерфейса подсети. Эта скорость является главной, ибо она определяет скорость передачи блоков данных сквозь всю подсеть. Именно эта скорость в первую очередь определяет быстродействие коммуникационной подсети. Для удобства сравнения с физической скоростью сквозная скорость часто пересчитывается в биты в секунду.

Факторы, влияющие на сквозную скорость, приведены в табл. 1.2.

Следует отметить, что сквозная скорость определяет второй временной фактор быстродействия коммуникационной подсети - время сквозного прохода блока данных через (сквозь) эту подсеть. Действительно, легко представить подсеть, в точках интерфейса которой данные проходят бы-

стро, например, со скоростью 100 Мбит/с. Однако если подсеть создана неоптимально, то блок данных может проходить сквозь нее в течение недопустимо долгого времени, например, 0.5 с.

Таблица 1.2

Факторы, влияющие на сквозную скорость

| **Фактор** | **Характеристика** |
|---|---|
| Топология | Длина канала определяет время распространения по нему сигнала; повторители, разветвители и другие компоненты канала вносят дополнительные задержки |
| Количество абонентских систем | Чем больше систем, тем значительнее потери времени на согласование их работы в сети |
| Структура станций | Эффективность структуры, число и расположение буферов памяти, степень аппаратной реализации функций, быстродействие микропроцессоров влияют на скорость работы станции |
| Величина трафика | Число и частота передач увеличивают потери времени на управление передачей |
| Число ошибок передачи | Потери времени на проверку, переспрос и повторную передачу блоков данных |
| Эффективность заполнения блоков данных | Чем больше в блоке данных упаковано информационных бит, тем меньше число необходимых блоков |
| Объем операций управления | Минимизация обработки прерываний, сообщений о передаче, упаковки и распаковки позволяет уменьшить потери времени |
| Интерфейс абонента | Качество и скорость передачи данных между станцией и абонентом также определяют возможные потери скорости |

Важной характеристикой коммуникационной подсети является используемая физическая среда. На этой основе создается канал - совокупность физической среды и каналообразующих аппаратных средств, соединяющая две системы. В различных сетях существуют различные процедуры обмена данными между рабочими станциями. Эти процедуры называют протоколами передачи данных [15, 2, 23] .

Международный институт инженеров по электротехнике и радиоэлектронике (Institute of Electrical and Electronics Engineers - IEEE) разработал стандарты для протоколов передачи данных в локальных сетях. Это стандарты IEEE802. Практический интерес представляют стандарты IEEE802.3, IEEE802.4 и IEEE802.5, которые описывают методы доступа к сетевым каналам данных.

Наибольшее распространение получили конкретные реализации методов доступа: Ethernet, Arcnet и Token Ring. Эти реализации основаны соответственно на стандартах IEEE802.3, IEEE802.4 и IEEE802.5. Для простоты мы будем использовать названия реализаций методов доступа, а не названия самих стандартов, хотя между стандартами и конкретными реализациями имеются некоторые различия.

### 1.3.2. Метод доступа Ethernet

Метод доступа, разработанный фирмой Xerox в 1975 г., пользуется наибольшей популярностью. Он обеспечивает высокую скорость передачи данных и надежность. Для данного метода доступа используется топология "общая шина". Поэтому сообщение, отправляемое одной рабочей станцией, принимается одновременно всеми остальными станциями, подключенными к общей шине. Та станция, которой предназначено сообщение, принимает его, остальные игнорируют.

Метод доступа Ethernet является методом множественного доступа с прослушиванием несущей и разрешением конфликтов (CSMA/CD - Carier Sense Multiple Access with Collision Detection). Перед началом передачи рабочая станция определяет, свободен канал или занят. Если канал свободен, станция начинает передачу. Ethernet не исключает возможности одновременной передачи сообщений двумя или несколькими станциями. Аппаратура автоматически распознает такие конфликты. После обнаружения конфликта станции задерживают передачу на некоторое время. Конфликты приводят к уменьшению быстродействия сети только в том случае, если работает 80-100 станций.

Аппаратура Ethernet обычно состоит из кабеля, разъемов, коннекторов различного типа, терминаторов (в случае коаксиального кабеля) и сетевых адаптеров. Кабель используется для передачи данных между рабочими станциями. Для подключения кабеля используют разъемы, которые подключаются к сетевым адаптерам, вставляемым в слоты расширения или встроенные порты материнской платы рабочей станции. Терминаторы подключаются к концам сети.

Для Ethernet могут быть использованы кабели разных типов: тонкий коаксиальный кабель, толстый коаксиальный кабель и витая пара. Для каждого типа кабеля используются свои разъемы и свой способ подключения кабеля к сетевому адаптеру.

### 1.3.3. Метод доступа Arcnet

Метод разработан фирмой Datapoint Corp. Он тоже получил широкое распространение, поскольку оборудование Arcnet дешевле, чем Ethernet или TokenRing. Arcnet используется в локальных сетях с топологией "звезда". Один из компьютеров создает специальный маркер, который

последовательно передается от одного компьютера к другому. Если станция желает передать сообщение другой станции, она должна дождаться маркера и добавить к нему сообщение, дополненное адресами отправителя и назначения. Когда пакет дойдет до станции назначения, сообщение будет "отцеплено" от маркера и передано станции.

Для организации сети Arcnet требуется специальный сетевой адаптер, имеющий один внешний разъем для подключения коаксиального кабеля. Каждый адаптер Arcnet должен иметь для данной сети свой номер, который устанавливается переключателями и находится в пределах от 0 до 255. Сетевые адаптеры рабочих станций через коаксиальный кабель с волновым сопротивлением 93 Ом подключаются к концентратору. Возможно также использование неэкранированной витой пары. Концентраторы бывают пассивными или активными, к ним может подключаться 4, 8, 16 или 32 рабочие станции.

### 1.3.4. Метод доступа Token-Ring

Метод разработан фирмой IBM и рассчитан на кольцевую топологию сети. Этот метод напоминает Arcnet, так как использует маркер, передаваемый от одной станции к другой. В отличие от Arcnet при методе доступа Token-Ring имеется возможность назначать различные приоритеты разным рабочим станциям.

Топология этой сети больше похожа на звезду, чем на кольцо. Рабочие станции Token-Ring подключаются радиально к концентратору типа 8228 производства IBM. В случае нескольких концентраторов они объединяются в кольцо через специальные разъемы. Скорость передачи данных в сети Token-Ring может достигать 4-16 Мбит/с, однако стоимость сетевого оборудования выше, чем для сети Ethernet.

## *1.4. Абонентские и терминальные системы*

Развитие теории ИВС и выпуск необходимого для них оборудования создали возможность построения универсальных сетей, обеспечивающих ввод, хранение, передачу, обработку и выдачу разнообразных видов информации, что позволило: снизить стоимость информационных средств; расширить сервис, реализуя электронную почту, видеоконференции, отсроченную доставку речевых сообщений и т.д.; резко уменьшить численность обслуживающего персонала; повысить надежность хранения и обработки информации; увеличить достоверность передачи информации; унифицировать используемое оборудование; дать возможность получения любой информации в однотипных абонентских пунктах непосредственно на рабочих местах.

Трудно перечислить все задачи, решаемые локальными сетями. Основными из них являются: коллективное использование ЭВМ для проведения расчетов; создание широкого спектра банков данных и информационно-поисковых систем; передача (и временное хранение) информации при помощи электронной почты; сбор, упорядочение и хранение информации о деятельности предприятия либо учреждения; подготовка и редактирование писем, отчетов, справок; обмен документами без распечатки их на бумаге; выписывание счетов, ведение бухгалтерского и складского учетов; выполнение научных, конструкторских и технологических работ, подготовка и передача чертежей, схем, рисунков; управление роботами, машинами, автоматическими станками.

Выделим (рис. 1.6) два типа систем: абонентские и ассоциативные. *Абонентской* называется система, предоставляющая в сети ПрП или использующая эти процессы. Система, предназначенная для обеспечения передачи информации между абонентскими системами, называется *ассоциативной*. В соответствии с этими определениями основными в ИВС являются абонентские системы. Ассоциативные системы являются вспомогательными и могут вообще отсутствовать. В зависимости от функций абонентские системы подразделяются на четыре вида.

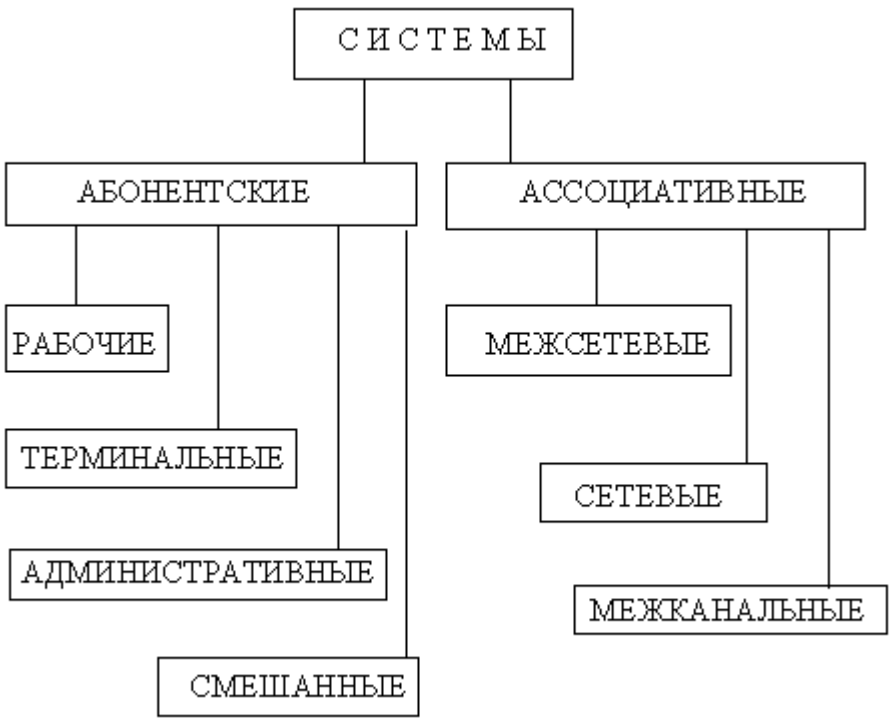



Рис. 1.6. Архитектура открытых систем

*Рабочей* называется система, предоставляющая пользователям один либо несколько ресурсов (банк данных, информационно-поисковая служба, служба выполнения заданий и т.д.). *Терминальной* называется система, в которой имеется один либо несколько терминалов и организовано

взаимодействие через коммуникационную подсеть с информационно-вычислительными ресурсами рабочих систем. Система, на которую возлагаются функции управления всей либо какой-нибудь частью ИВС, называется *административной*. *Смешанной* является комбинированная система, выполняющая функции двух или более рассмотренных выше видов абонентских систем.

Ассоциативная система, предназначенная для обеспечения взаимодействия двух либо более ИВС, называется *межсетевой*. Ассоциативная система, на которую возложены функции объединения коммуникационных подсетей в одной и той же ИВС, называется *сетевой*. К этому типу систем относится и та, которая выполняет функции маршрутизации и коммутации информации между абонентскими системами информационно-вычислительной сети. Ее также называют *коммуникационной*. Если ассоциативная система связывает друг с другом два различающихся по своим параметрам или характеристикам канала, то ее называют *межканальной*.

Межсетевая, сетевая и межканальная системы выполняют функции преобразования информации из одной группы стандартов (протоколов) в другую. Эти группы определяются соединяемыми информационно-вычислительными сетями, коммуникационными подсетями либо каналами. Поэтому такие системы часто называют *интерфейсными*. Исключением иногда является коммуникационная система. В случаях, когда она в коммуникационной подсети одна, коммуникационная система может и не осуществлять преобразования информации, занимаясь только функциями коммутации и маршрутизации информации.

Абонентская система реализуется в одной либо нескольких ЭВМ. Наиболее крупные машины, используемые в сети, выделяются для выполнения функций рабочих систем. Ассоциативная система не требует больших мощностей. Поэтому она реализуется в микроЭВМ или в виде логических дискретных модулей (вентильные матрицы, программируемые логические матрицы).

Программное обеспечение абонентских систем определяет информационно-вычислительные ресурсы сети, формы и методы обработки информации. Все входящие в сеть абонентские системы, независимо от того, на каких ЭВМ они построены, должны выполнять практически одни и те же функции, определенные протоколами уровней 1-7.

В реальных условиях абонентскую систему удобно реализовать в двух технических блоках. Типовая структура такой системы показана на рис. 1.7. Она состоит из абонента и станции. Здесь абонентом является основная часть системы, реализованная в ЭВМ, которая выполняет прикладные процессы и функции части области взаимодействия открытых систем. Станцией называется устройство, сопрягающее ЭВМ с физическими средствами соединения и реализующее функции остальной части области

взаимодействия. Обе части функций области взаимодействия открытых систем связываются внутрисистемной вставкой, обеспечивающей интерфейс абонента со станцией.

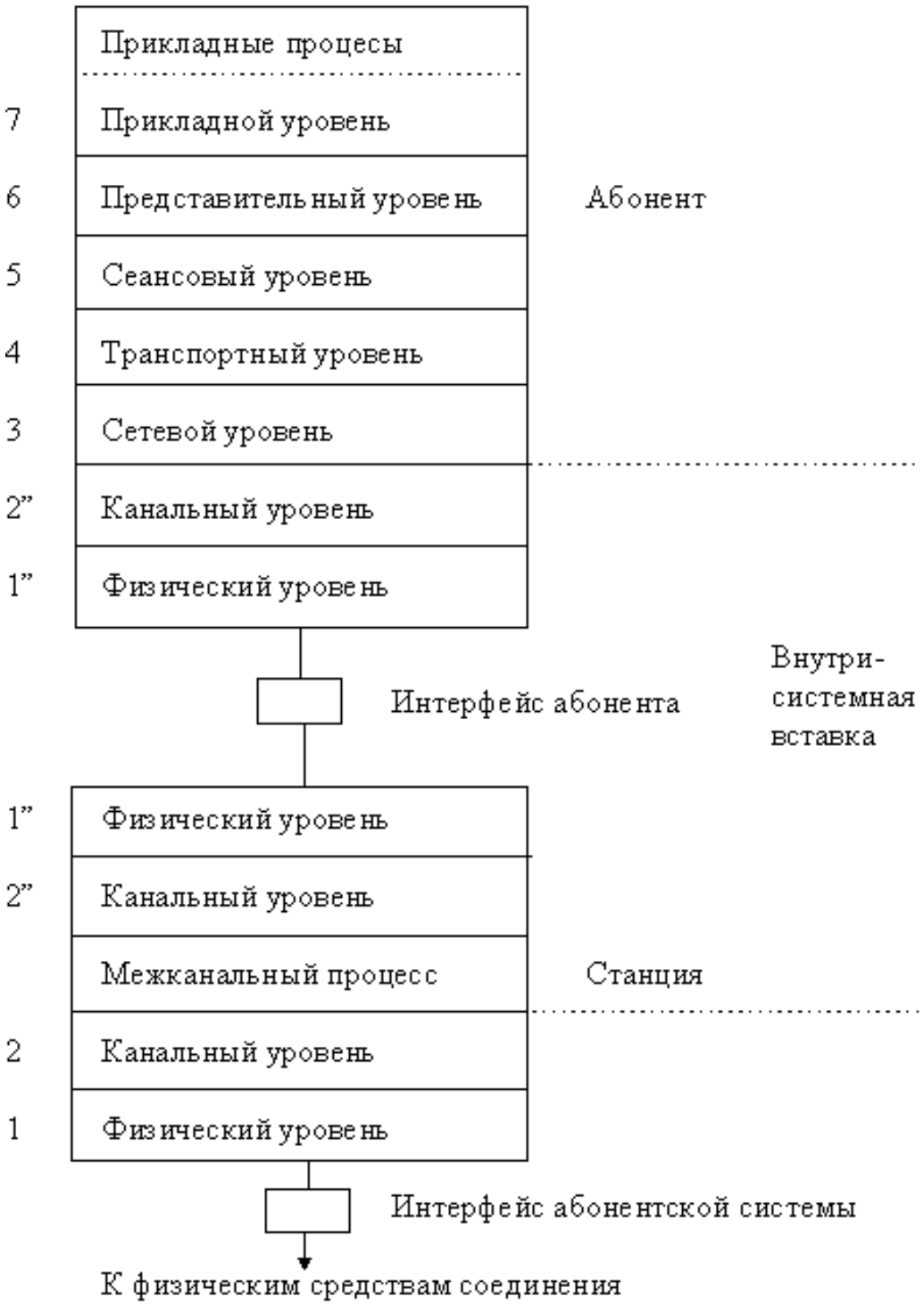



Рис. 1.7. Абонентская система, состоящая из двух блоков

Рабочие системы локальной сети определяют ее основные информационно-вычислительные ресурсы. Однако не менее важную роль играют терминальные системы. Развитие микропроцессорной техники и широкий

ассортимент сверхбольших и больших интегральных схем привели к тому, что все большее значение приобретают персональные терминальные системы, каждая из которых рассчитана на одного пользователя. Одной из абонентских систем ИВС всегда поручается управление сетью. Эта система, именуемая административной, или центром управления сетью, выполняет функции, основные из которых показаны в табл. 1.3. Для надежности работы ЭВМ, выполняющая функции административной системы, часто дублируется резервной машиной.

Таблица 1.3

Функции административной системы

| **Группа функций** | **Функции** |
|---|---|
| Сбор информации | Учет работы компонентов сети<br>Сведения о загрузке каналов<br>Время работы соединений<br>Регистрация ошибок<br>Загрузка ресурсов сети<br>Выполнение отчетов о работе сети |
| Диагностика | Самотестирование<br>Проверка работы компонентов сети<br>Контроль передачи пакетов<br>Индикация состояний<br>Генерация искусственной нагрузки |
| Восстановление работы | Повторное установление соединений<br>Повторная загрузка программ |
| Управление конфигурацией | Включение новых абонентов<br>Ведение справочника о сети<br>Создание резервных трактов передачи<br>Выполнение вынужденных разъединений физических соединений<br>Изоляция неисправных логических компонентов |
| Пользовательский сервис | Показ пользователям динамического состояния сети<br>Помощь в разборе неясных ситуаций<br>Выдача справок об информационно-вычислительных ресурсах сети |

Реализация абонентской системы в двух технических блоках выгодна по двум причинам. Во-первых, станция разгружает ЭВМ, освобождая ее от вспомогательных процессов, связанных с передачей информации. Во-вторых, наличие между физическими средствами соединения и ЭВМ станции позволяет иметь интерфейс абонента, не зависящий от характеристик и параметров этих средств. В результате интерфейс абонента может быть выбран удобным с точки зрения ЭВМ, что позволяет подключить ее

к различным типам физических средств соединения. Для подключения необходимо лишь заменить станцию.

### *1.5. Контрольные вопросы*

1. Что понимается под архитектурой ИВС?
2. Для чего создана эталонная модель взаимодействия открытых систем?
3. Каковы уровни эталонной модели?
4. Какие задачи решает прикладной уровень?
5. Что такое "протокол управления контекстами" и "протокол управления терминалами"?
6. Сформулируйте функции, выполняемые протоколом прикладного уровня.
7. Как связаны синтаксис и семантика с прикладным и представительным уровнями?
8. Какие функции выполняет протокол представительного уровня?
9. Какие функции выполняет протокол сеансового уровня?
10. Каковы задачи, решаемые на транспортном уровне?
11. Сформулируйте особенности использования дейтаграмм и виртуальных каналов.
12. Какие функции выполняет протокол сетевого уровня?
13. Какие функции выполняет протокол канального уровня?
14. Какие функции выполняет протокол физического уровня?
15. Что такое абонентская система?
16. Приведите классификацию ИВС по протяженности.
17. Каковы принципиальные отличия локальных, региональных и глобальных сетей?
18. Приведите классификацию ИВС по числу и типам используемых абонентских систем.
19. Каковы достоинства и недостатки сетей с централизованным и распределенным управлением?
20. Что такое коммуникационная подсеть?
21. Каковы основные характеристики коммуникационной подсети?
22. Чем отличается физическая скорость передачи данных по каналу от сквозной?
23. В чем особенности метода доступа Ethernet?
24. В чем особенности метода доступа Arcnet?
25. В чем особенности метода доступа Token-Ring?
26. Перечислите задачи, решаемые локальными сетями.
27. Что такое абонентские и ассоциативные системы?
28. Охарактеризуйте рабочие и терминальные системы.
29. Каковы функции административной системы?

# 2. Методы оптимизации сетей ЭВМ

Оптимизация в процессе проектирования и функционирования сетей ЭВМ занимает важное место. В главе дан обзор задач и методов оптимизации в соответствующей предметной области. Рассмотрен практически весь спектр задач от структурного синтеза топологии сети до многокритериальных проблем выбора пропускных способностей. Глава завершается кратким обзором инженерных методов оценки надежности ИВС.

## *2.1. Проблемы оптимизации сетей ЭВМ*

В процессе системного проектирования ИВС удается получить набор компонентов сети: узлов коммутации, линий передач, концентраторов и мультиплексоров с известным набором характеристик; например, концентратор может обслуживать не более некоторого числа терминалов. Наивно предполагать, что на этой стадии проектирование завершено. Разработчик сети должен найти наилучший вариант расположения этих компонентов, который отвечал бы требованиям, предъявляемым сетевым трафиком; обычно им должен быть самый дешевый вариант, соответствующий определенному критерию эффективности сети (например, обеспечивающий задержку в сети, производительность или надежность). Эту вторую стадию процесса проектирования можно назвать оптимизацией сети.

Задача оптимизации настолько сложна, что пока не решена в общем виде, но известно множество подходов к решению ее подзадач. Рассмотрим наиболее типичные методы и подходы, уделяя основное внимание их разнообразию. Значительная часть теоретических положений приводится без доказательств.

Рассмотрим общую постановку задачи. Как правило, исходными данными для задачи оптимизации является расположение терминалов (любых устройств, которые могут быть абонентами сети). Поскольку терминалы могут сильно различаться по своим характеристикам, следующая по важности совокупность данных задается *матрицей требований*. В ней содержатся сведения о том, какой объем информации сеть должна передать от каждого терминала ко всем остальным.

Сетевой трафик изменяется в зависимости от времени суток и дня недели. В большинстве случаев сеть проектируется в расчете на максимальный трафик. Во избежание неопределенности, обусловленной статическими флуктуациями, обычно рассматривают средний трафик в час наибольшей нагрузки самого напряженного дня недели.

Следующая совокупность данных касается компонентов, из которых строится сеть: узлов, концентраторов, мультиплексоров и, возможно, спутников связи вместе с их наземными станциями. Каждый из компонентов сети характеризуется своими предельными характеристиками и стои-

мостью. Поэтому может возникнуть необходимость выбора одной из нескольких моделей узлов коммутации, обладающих различными предельными характеристиками по обработке пакетов и стоимостью. Аналогичная проблема возникает и при выборе линий передачи.

Для каждой из компонент существуют топологические ограничения. Например, мультиплексор или концентратор могут обслуживать не более заданного числа терминалов, а в узле коммутации может сходиться не более определенного числа линий связи, возможно зависящего от их пропускных способностей. Основную задачу оптимизации можно сформулировать как размещение и соединение компонент сети.

Критерием успешного завершения являются либо оптимизация некоторой переменной, либо удовлетворение заданных ограничений. Поэтому нельзя требовать, чтобы сеть одновременно имела минимальные как стоимость, так и среднюю задержку пакета. Необходимо либо ввести критерий, который устанавливает компромисс между этими параметрами, либо рассматривать один из них как ограничение, а другой как оптимизируемую переменную.

К основным параметрам сети относятся стоимость, производительность, задержка и надежность. Стоимость часто выступает как оптимизируемый параметр и является величиной, которая определяется одним числом для всей сети.

Производительность сети тесно связана с матрицей требований. Иногда производительность не является заданной величиной и ее необходимо максимизировать. В этих случаях для изменения матрицы требований вводится масштабный множитель, который затем максимизируется. При этом величина полного трафика становится параметром, характеризующим сеть в целом, а матрица требований показывает относительное распределение трафика между терминалами.

Задержку можно представить как среднюю задержку сообщений в сети, однако в большинстве случаев важно знать и полную картину распределения задержек. Часто бывает достаточным рассчитать распределение вероятностей для задержек и оценить, насколько они приемлемы, в то время как средняя задержка выступает как главный параметр оптимизации.

Надежность, по существу, означает доступность услуг сети для всех ее абонентов. Частота отказа узлов и линий связи, а также среднее время их восстановления являются частными характеристиками, однако *коэффициент готовности*, измеряемый долей времени, в течение которого терминал может пользоваться услугами сети, также может быть параметром, используемым при оптимизации. Часто можно показать, что требования к надежности почти эквивалентны требованиям к *связности* сети.

Даже если бы строгое решение задач оптимизации в полной постановке было возможным, реализация этой возможности на практике вряд ли часто необходима. Во всех реально встречающихся случаях имеются

существенные ограничения, снижающие размерность задачи оптимизации. Оптимизация в сетях общего пользования имеет жесткие ограничения, связанные со сложившимся расположением оборудования и точек доступа к линиям связи. В отличие от разработчика частной сети инженер, расширяющий сеть общего пользования, учитывает реальные стоимости и поэтому, вероятно, предпочтет размещение центров коммутации там, где сходятся линии связи. Вместе с тем он должен предусмотреть возможное развитие сети значительно дальше, чем разработчик частной сети, поскольку планирование установки связного оборудования определяется периодом, измеряемым, скорее, десятилетиями, чем годовыми интервалами. Такое планирование строится на принципе обеспечения возможности установки дополнительного оборудования для предполагаемого расширения сети. Минимизация стоимости дополнительного оборудования (т.е. будущего варианта сети) может оказаться более выгодной для экономики сети общего пользования, чем текущая оптимизация, учитывающая только существующие в данный момент потребности в передаче информации.

В силу изложенных выше причин задачи оптимизации, встречающиеся на практике, могут сильно отличаться от задач, обычно рассматриваемых в литературе.

При решении задач оптимизации было бы идеальным применить строгие математические методы и получить точное оптимальное решение. Однако, даже если задачу можно строго сформулировать и в принципе решить, в большинстве случаев такой подход оказывается неоправданно громоздким. Обычно применяются три подхода, которые можно классифицировать как комбинаторный, аналитический и эвристический.

Комбинаторные методы имеют дело с конечными множествами и отношениями между ними. Типичным примером применения этих методов является выбор топологии сети. Узлы и линии связи сети образуют граф, свойства которого исследуются методами известной теории.

Сложность комбинаторных задач обусловлена их большой размерностью и огромным числом возможных вариантов, которые необходимо исследовать. Например, в сети из 30 узлов существует 435 вариантов соединения пары узлов линией связи и $2^{435}$ вариантов расположения линий связи, включая и множество тривиальных случаев. Изучение топологии сетей путем перебора всех возможных вариантов - совершенно бесперспективное занятие. Поэтому возникает потребность в более простых эвристических методах, являющихся, по существу, методами "проб и ошибок".

Задача распределения потока информации по линиям связи с целью увеличения пропускной способности является примером задачи, которая допускает аналитическое решение. Некоторые из аналитических методов оптимизации построены на больших упрощениях, например, сведении нелинейной задачи к линейной.

Какова бы ни была задача, часто наилучшими методами ее решения являются эвристические. Например, программа может генерировать топологии с улучшенными свойствами, пользуясь некоторыми "разумными" соображениями, вместо того чтобы использовать алгоритм с доказанной сходимостью к оптимуму. Такая программа может многократно "улучшать" потоки в сети без какой-либо гарантии оптимальности результата. Чтобы достаточно полно охватить возможные случаи, вводятся случайные изменения, и тогда одно из "наилучших решений" может оказаться близким к оптимуму. В эвристическом алгоритме решения задачи оптимизации случайным образом выбираются начальные точки, число которых должно быть достаточно большим. Результаты, полученные для различных начальных точек, сравниваются между собой. Наилучший из них выбирается в качестве истинного решения задачи.

## *2.2. Структурный синтез и оптимизация топологии*

### 2.2.1. Регулярные графы

Одним из методов построения топологических структур сетей с заданными свойствами является синтез графов с помощью известных математических операций. На рис. 8 приведен граф, построенный таким путем. Обычно такие графы обладают некоторой регулярностью, т.е. все их узлы равноправны в смысле топологии сети.

Граф Петерсона, изображенный на рис. 2.1, состоит из 10 узлов и 15 линий со связностью 3. Степень каждого узла также равна трем, и, следовательно, связность этого графа максимальна. Говорят, что такой граф имеет максимальную связность. Другое свойство оптимальности графа Петерсона связано с *диаметром* графа, т.е. максимальным расстоянием между парами узлов сети, измеряемым числом "шагов" от узла к узлу. На втором уровне будет три узла, так как степень исходного узла была равна трем. По той же причине каждый из трех узлов второго уровня будет соединен не более чем с двумя узлами третьего уровня. На расстоянии двух шагов от исходного будет не более 10 узлов, включая и его самого. Таким образом, граф степени 3 и диаметра 2 не может содержать более десяти узлов. Этот максимум достигается на графе Петерсона.

Регулярные графы такого типа представляют большой теоретический интерес и могут оказаться полезными на практике при конструировании коммутаторов из одинаковых компонентов. В коммутации каналов уже сейчас используются регулярные соединения компонент, и, если экономика производства будет и дальше развиваться в том же направлении, это может стать характерной чертой коммутации вообще. При построении крупномасштабных сетей связи с географически далеко отстоящими друг

от друга узлами регулярные синтезированные графы едва ли найдут применение, поскольку стоимость линий связи, являющихся важными параметрами оптимизации, сильно меняется в зависимости от географических факторов.

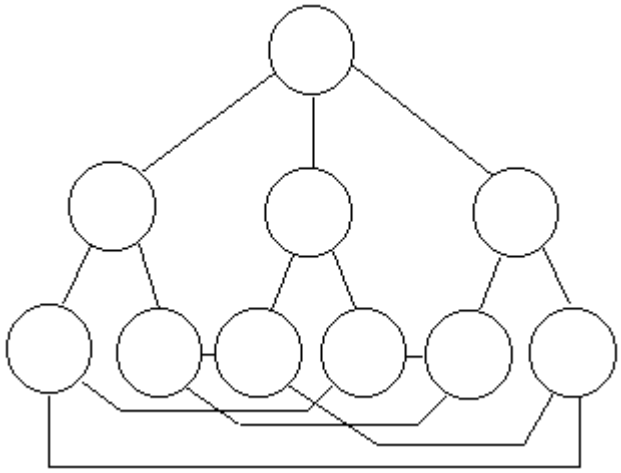


Рис. 2.1. Граф Петерсона

Важным элементом является определение связности и реберной связности графа. В связности небольшой сети можно легко убедиться простой проверкой. Для больших сетей проверка связности представляет весьма сложную задачу, которая, тем не менее, весьма часто встречается на практике.

### 2.2.2. Общая эвристическая схема

Методы оптимизации топологии имеют эвристический характер. Топология связана с выбором пропускных способностей, поскольку отсутствие прямой связи между некоторой парой узлов можно интерпретировать как линию с нулевой пропускной способностью в каждом из направлений. Именно так работает метод исключения ветвей.

Топология, созданная каким-то случайным образом, едва ли будет приемлемой и связной. Если же сеть связна, то может потребоваться внести в топологию сети некоторые изменения, улучшающие ее свойства и сохраняющие связность. В методе перестановки ветвей вносимые в топологию изменения должны по возможности не менять степеней узлов.

На рис. 2.2 показан способ, с помощью которого можно исследовать различные топологии сети и накапливать локально-оптимальные варианты.

На первом шаге этого алгоритма для построения топологий используются случайные числа, причем линии связи вводятся таким образом, чтобы они соединяли узлы с наименьшими степенями. При таком способе построения линий узел со степенью два появится только после того, как степенью каждого из узлов сети будет единица. В ходе дальнейшей оптимизации происходит введение локальных изменений топологии, например,

методом перестановки ветвей. После каждой перестановки вновь проверяется связность сети, и, если она окажется ниже заданной, эта перестановка отвергается. В основной части этой процедуры для сети исследуемой топологии решается задача о распределении потока и выборе пропускной способности, после чего трафик в сети увеличивается до тех пор, пока не будет достигнута верхняя граница средней задержки. Величина пропускной способности регистрируется, определяется стоимость сети, и теперь можно решить, удовлетворяет ли данная сеть необходимым критериям.

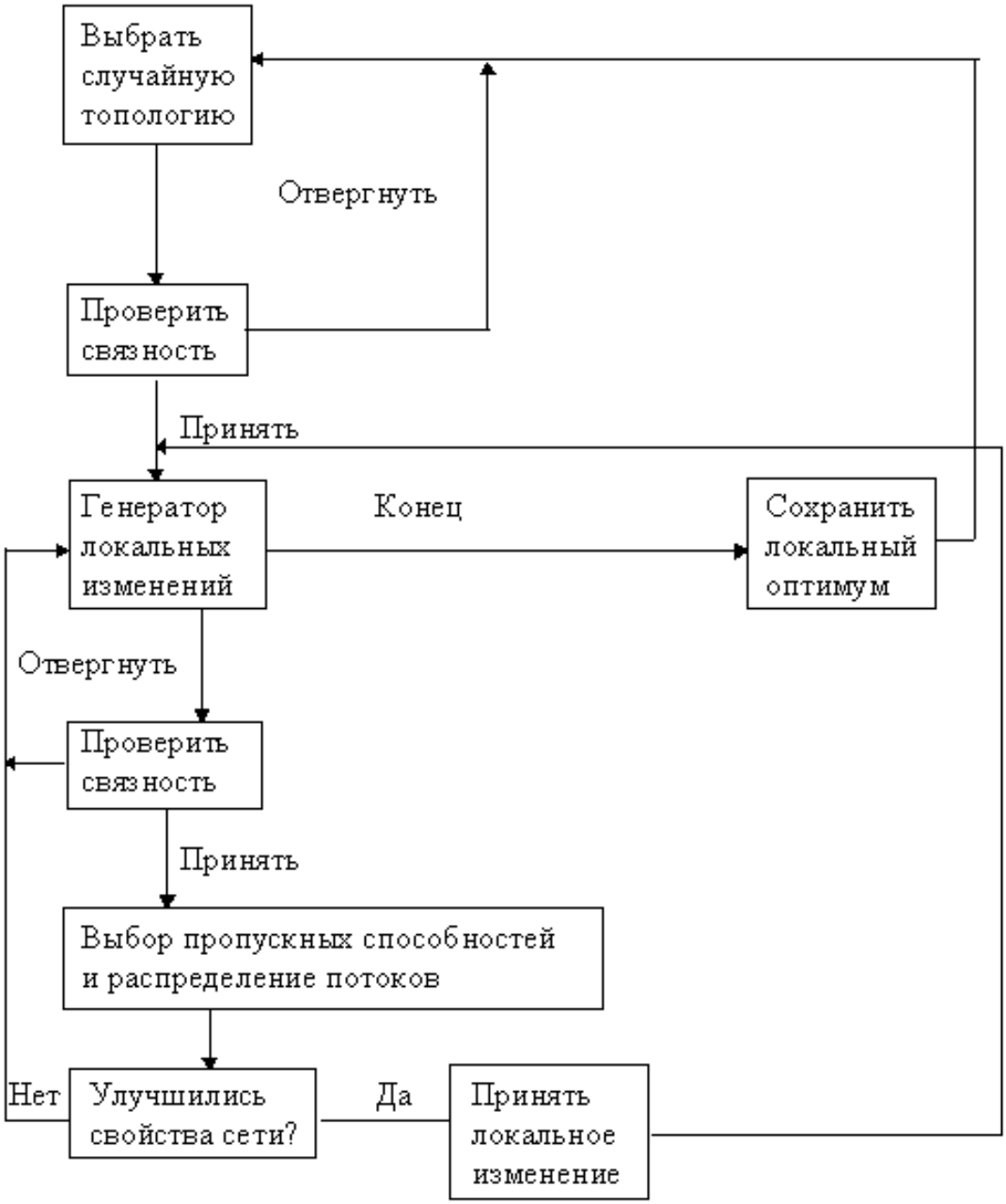



Рис. 2.2. Эвристический метод оптимизации топологии

Топологии, выдержавшие проверку на связность и удовлетворяющие критериям на производительность, подвергаются дальнейшим улуч-

шениям. Когда возможность локальных изменений исчерпывается, полученный вариант запоминается вместе со своими характеристиками как локальный оптимум и программа снова обращается к генератору за новой топологией. После того как будет оптимизировано достаточное число начальных вариантов топологий сети, строится зависимость, определяющая стоимость и уровень трафика для всех полученных локальных решений и являющаяся результатом процесса оптимизации.

### 2.2.3. Алгоритм Прима

В качестве примера эвристического метода приведем алгоритм Прима. Для связи средств вычислительной техники в больших сетях традиционно применяется модемная техника, однако качество и надежность получаемой системы передачи данных существенно зависят от выбора каналов - выделенных или коммутируемых. Проектировщику такой распределенной системы традиционно приходится разрешать дилемму между обеспечением требуемого качества сети и ее стоимости, что связано со значительной разницей в оплате указанных каналов связи.

Пусть имеется множество территориально распределенных объектов $X=\{x_i\}$, характеризуемых: географическими координатами $(a_i,b_i)$; объемом информации $f_i$, генерируемой объектом. Предполагается, что одним из объектов является "центральный" (главный) узел, выделенный в соответствии с некоторыми правилами - административными, территориальными и пр.

Пусть, кроме того, известны приведенные затраты $C_{пер,ij}$ на передачу информации от объекта $x_i$ к объекту $x_j$, зависящие от трафика $f_{ij}$ в данном направлении и длины линий связи $l_{ij}$. Требуется синтезировать структуру минимальной стоимости в классе древовидных структур при ограничениях на максимальный трафик $f_{ij}$ в каждой ветви $(f_{ij},d_{max})$, где $f_{ij}$ определяется как сумма информационных потоков от всех узлов, предшествующих узлу i на путях от концевых вершин к корню дерева, и потока $h_i$, формируемого объектом $x_i$.

Задача синтеза древовидной сети впервые рассматривалась в работе Прима, в которой предложен точный алгоритм синтеза сети минимальной стоимости, но без учета ограничений на пропускные способности линий связи и, как следствие, на суммарный поток, передаваемый по ветвям. Вместе с тем доказано, что алгоритм доставляет оптимальное решение и состоит из следующих шагов. Пусть первоначально рассматривается исходное множество несвязанных объектов (вершин) $X=\{x_i\}$.

1. Выбирается произвольная вершина (подграф) $x_i$ и определяется стоимость введения ребра (i, j), связывающего $x_i$ с некоторым подграфом $x_j$: $c_{ij}$.

2. Если подграфы $x_i$ и $x_j$ состоят из нескольких вершин, то выбирается ребро, доставляющее топологическое расстояние между подграфами (минимум из всех пар возможных расстояний): среди всех пар (i, j) определяется такая пара ($i^*$, $j^*$), для которой

$$c_{i^*j^*} = \min_{\forall(i,j)} c_{ij}.$$

3. Подграфы $x_{i^*}$ и $x_{j^*}$ объединяются в один: $x_{ij^*} = x_{i^*} \cup x_{j^*}$. На этом итерация алгоритма Прима завершается, а сами итерации повторяются до тех пор, пока имеются изолированные подграфы (их количество на каждой итерации уменьшается на единицу).

### 2.2.4. Метод насыщения сечения

Рассмотрим метод насыщения сечения (сечением называется множество узлов и линий, удаление которых разбивает сеть на две несвязные части). Предположим, что нагрузка в сети увеличивается до предела. Начиная с некоторого момента алгоритм маршрутизации или используемая процедура распределения потоков станет направлять потоки по альтернативным путям, пока в сети не образуется сечение из почти насыщенных линий. Далее увеличение некоторых потоков в сети будет сопровождаться чрезмерным увеличением задержки. Появившееся сечение отражает слабость топологии сети, которую можно улучшить, добавив еще одну линию, соединяющую узлы, находящиеся с двух сторон сечения. Как правило, новая линия должна соединять узлы, находящиеся, по крайней мере, на расстоянии двух шагов от краев сечения.

Опытный разработчик обычно умеет находить такие варианты топологий сети, которые оказываются значительно лучше вариантов, найденных с помощью какой-нибудь простой программы, поэтому одним из наиболее мощных инструментов разработки топологии является сочетание метода насыщения сечений с интуицией разработчика, время от времени корректирующего работу программы.

## *2.3. Иерархические сети и сети с неоднородной средой*

Большие сети более выгодно строить по иерархическому принципу, когда каждый уровень строится как отдельная сеть, по своим собственным топологическим правилам. Основными доводами в пользу экономичности иерархических сетей является, во-первых, то, что такие сети позволяют концентрировать трафик на основных маршрутах, на которых можно использовать мощные каналы связи, обеспечивающие "экономию за счет масштаба", а во-вторых, то, что в таких сетях уменьшается число шагов

при передачах и, следовательно, количество узлов коммутации и время задержки в них.

То же самое можно сказать и о сетях с неоднородной средой передачи данных. На разных уровнях иерархии могут использоваться различные способы коммутации, мультиплексирования и концентрации, а также различные средства связи в зависимости от объема и распределения трафика. Например, на верхнем уровне иерархии в зависимости от алгоритма маршрутизации флуктуации трафика могут иметь большую или меньшую величину, чем на нижних уровнях сети; соответственно этому и выбирают способы коммутации и уплотнения [24, 25].

Правила маршрутизации в таких сетях зависят от того, каким образом используется спутниковая система. Это, в свою очередь, зависит от того, какое требование является определяющим - большая производительность или малая задержка при передаче. Например, в спутниковой системе типа ALOHA может использоваться методика типа MASTER, согласно которой повторная передача пакетов после их столкновения в спутниковом канале всегда осуществляется через наземную сеть. Эта методика хорошо работает при больших нагрузках в сети, но при малых нагрузках она недостаточно широко использует наземную сеть, и поэтому все пакеты передаются с достаточно большой задержкой, характерной для спутниковых каналов. Если в сети действуют правила маршрутизации, ориентированные на минимальную задержку в сети, при малой нагрузке основную роль в передаче будет играть наземная сеть, а спутниковая система возьмет на себя избыточный поток, когда возрастут задержки в наземной сети.

Число возможных топологий сети тем меньше, чем жестче предъявленные к ней требования. При ограничениях на расстояние в один шаг между любым узлом сети и ближайшей наземной станцией и расстояние в два шага до следующей ближайшей представляется весьма вероятным, что достаточно хороший вариант топологии можно получить эвристическим методом. Стоимость такой сети в основном зависит от числа наземных станций связи, поэтому, изменяя число и расположение этих станций, можно добиться наилучшего соотношения их стоимости со стоимостью наземных линий, которых требуется тем больше, чем меньше спутниковых линий связи используется в сети.

Рассмотрим иерархическую сеть, состоящую из основной базовой ячеистой сети, к которой подсоединены локальные древовидные сети. Очень часто расположение узлов базовой сети диктуется расположением городов и населенных пунктов, в которых сосредоточены терминалы. Если удастся зафиксировать расположение узлов, проблема оптимизации базовой сети превращается в уже изученную, а оптимизация локальных сетей сведется к хорошо изученной проблеме проектирования централизованных вычислительных сетей. Размещение узлов в пределах обслуживаемой ими зоны может рассматриваться как часть задачи локальной оптимизации, ес-

ли оно не слишком сильно влияет на стоимость базовой сети. Если считать, что вопросы надежности, производительности и задержки в сети относятся в основном к базовой сети, то проблемы оптимизации разных уровней сети почти не пересекаются, кроме, может быть, вопроса выбора и числа расположения мест, в которых сети нижнего уровня подсоединяются к сети верхнего уровня. При добавлении узлов стоимость системы возрастает по очень сложному закону, который определяется свойствами базовой сети. Для упрощения задачи можно положить стоимость узлов постоянной и оптимизировать стоимость линий и концентраторов сети доступа, рассматривая стоимость узла как фиксированную добавку.

Для случаев, когда расположение терминалов не позволяет объединить их в группы естественным образом, были предложены специальные эвристические методы. Их суть состоит в том, что на первом шаге в группы объединяются ближайшие терминалы, а затем каждая группа представляется своим центром масс и числом входящих в него терминалов. Начав с объединения отдельных терминалов, алгоритм затем переходит к объединению ближайших образований независимо от того, являются они отдельными терминалами или группами, до тех пор, пока группы не достигнут заранее заданной предельной величины и дальнейшее их объединение станет невозможным. При этом может понадобиться какое-то правило, запрещающее объединить группы, расположенные слишком далеко друг от друга; иначе остающимся группам, не достигшим максимальной величины, придется охватить слишком большие пространства. Расположение узла внутри группы связано с установкой концентратора при одном из терминалов. Поскольку положение центра тяжести всей группы известно, достаточно исследовать варианты расположения концентратора при терминалах, наименее удаленных от центра тяжести, и выбрать вариант с минимальной стоимостью.

При другом способе организации локальной сети терминалы подключаются или к концентраторам, или непосредственно к центральному узлу. Концентраторы и узлы имеют ограничения на число обслуживаемых ими терминалов. В этом случае используется метод соединения терминалов, в котором терминалы сначала объединяются вокруг концентраторов, затем создаются группы концентраторов, после чего группы концентраторов и оставшиеся свободные терминалы распределяются по узлам. Прежде чем подключать терминал к концентратору, следует проверить, не будет ли выгоднее с экономической точки зрения присоединить терминал непосредственно к узлу.

## *2.4. Оптимизация потоков и пропускных способностей*

### 2.4.1. Алгоритм Форда-Фалкерсона

Связность и реберная связность стали значительно понятней, а их вычисление значительно проще после установления связей между теорией связности и расчетами потоков в сетях. Выделим в ИВС два узла - источник s и сток t. Если бы это была сеть из труб заданного сечения, можно было бы задаться вопросом: какой максимальный поток может перенести эта сеть от s к t?

Пусть узлы $n_1$, $n_2$, ..., $n_k$ соединены ребрами и задана матрица $G=|g_{ij}|$ $(1\le i,j\le k)$ пропускных способностей, соединяющих узел $n_i$ с узлом $n_j$, причем в силу дуплексности каналов связи предполагается симметричность матрицы G. Необходимо для каждой пары $(n_i, n_j)$ определить маршрут, обеспечивающий максимальную пропускную способность. Использование теоремы об (s-t)-разрезе неориентированного графа позволяет сконструировать конечный алгоритм поиска оптимальных потоков.

Согласно теореме Форда-Фалкерсона максимально возможное значение суммарного потока на конечных дугах равно минимальной пропускной способности выбранного разреза. При этом под пропускной способностью разреза понимается сумма пропускных способностей дуг, образующих разрез.

В математической записи соотношение, выражающее содержание теоремы Форда-Фалкерсона, выглядит следующим образом:

$$\sum_i \sum_j f(v_i,v_j) = \sum_i \sum_j c(v_i,v_j) ,$$

где $f(v_i,v_j)$ - значение потока по дугам заданного графа; $c(v_i,v_j)$- пропускная способность дуги; i - все вершины подграфа, образующие разрез; j - дополнение разреза до множества всех вершин графа.

На базе теоремы построен одноименный алгоритм, позволяющий найти минимальные разрезы и оценить соответствующие им значения максимального потока. Принцип, лежащий в основе алгоритма, состоит в том, чтобы найти все возможные насыщенные пути (цепи), ведущие от $v_s$ к $v_t$ для (s-t)-разреза. С этой целью последовательно, начиная с вершины $v_s$, просматривается множество смежных вершин $v_i$. Из множества дуг $(v_s,v_i)$, соединяющих vs с $v_i$, выбирают одну, у которой величина потока ближе всех подходит к значению насыщения. Помечают вершину $v_i$ знаком, показывающим, что она была просмотрена, и приписывают ей величину d, на которую можно увеличить поток по дуге, ведущей в эту вершину. Затем просматривают последующие вершины $v_j$, смежные с $v_i$, и останавливаются на той, в которую ведет дуга с потоком, ближайшим к значению насыщения. По этой дуге переходят в соответствующую вершину $v_j$. Делают отметку и идут далее в направлении вершины $v_t$. Если путь, на котором будут отмечены все пройденные вершины, приведет в вершину $v_t$, то это

говорит о том, что найден один из путей, наиболее близкий к насыщению. Нужно довести поток по нему до насыщения, увеличивая тем самым поток на графе. Значение, на которое можно увеличить поток, находится как минимальное из всех d, найденных ранее для отмеченных вершин. Для выяснения вопроса, является полученный таким образом поток максимальным или нет, следует просмотреть все другие возможные пути, ведущие от $v_s$ к $v_t$. Для этого необходимо возвратиться в $v_s$ и повторить описанные выше действия, но идти следует по еще не отмеченным вершинам.

В результате выполнения указанного алгоритма возможны два варианта:

- насыщенный путь снова приводит от $v_s$ к $v_t$, т.е. удается найти ненасыщенные пути, и, значит, можно увеличить потоки на их дугах;
- в процессе поиска пути обнаруживается вершина, у которой все смежные вершины уже отмечены, и, следовательно, новых путей, ведущих к $v_t$, больше нет. Это указывает на то, что в ходе выполнения алгоритма был найден максимальный поток.

Такая тесная связь между потоком в сети и ее связностью приводит к результату, известному как теорема Уитни. Эта теорема утверждает, что в графе со связностью n любую пару узлов s и t можно соединить по крайней мере n различными цепями, не имеющими общих узлов, кроме s и t. Существует целый ряд подобных теорем, связывающих размеры сечений с числом независимых цепей.

### 2.4.2. Выбор пропускных способностей

Увеличение пропускных способностей уменьшает среднюю задержку в сети, но увеличивает стоимость, и это является основной особенностью задачи оптимизации. Для каждой линии имеется зависимость между ее стоимостью (зависящей также от длины) и пропускной способностью C. Для каждого канала i имеем поток $\lambda_i$; $C_i$- пропускная способность этого канала, а $d_iC_i$ - его стоимость. Предположим также, что задержка в сети зависит только от полного потока в каналах. Величины $\lambda_i$ и $d_i$ считаются заданными, а $C_i$ необходимо найти.

Определение средней задержки T требует анализа сложной системы массового обслуживания. Для получения простой формулы для T делается ряд предположений. Во-первых, все очереди связываются с линиями, выходящими из узла. С каждой i-й линией сопоставляется средняя задержка на этой линии. Пусть $\gamma$ - общий трафик в сети. Тогда среднее число пакетов, находящихся в сети, есть $\gamma T$ (T - средняя задержка пакета). С другой стороны, число пакетов в каждой очереди из тех же соображений есть $\lambda_iT_i$, откуда $\gamma T= \sum\lambda_iT_i$.

Поскольку время обслуживания пропорционально длине пакета, необходимо ввести еще один параметр m, характеризующий среднюю длину

пакета. В очередь на передачу поступают как пакеты, пришедшие из других линий, так и вновь поступившие в сеть. Эти потоки пакетов вышли из разных очередей, и поэтому интервалы между поступлениями отдельных пакетов распределены по весьма сложному закону. Если очереди представить с помощью СМО M/M/1 (что оправдывается на практике) и учесть формулу для средней задержки в этой системе $T_i=1/(\mu C_i-\lambda_i)$, то для средней задержки во всей сети, которую необходимо минимизировать, получаем:

$$T=\sum(\lambda_i/\gamma)[1/(\mu C_i-\lambda_i)].$$

В этом выражении не учтены время обработки пакета в узле и задержки при распространении сигнала по линии связи, а также наличие пакетов подтверждения и другой служебной информации, передаваемой по сети.

Поиск минимума времени T и соответствующих ему значений $C_i$ осуществляется с помощью метода неопределенных множителей Лагранжа. Распределение пропускных способностей каналов, получающихся из уравнений, оказывается равным

$$C_i=\lambda_i/\mu+k\sqrt{d_i\lambda_i}\,.$$

Первое слагаемое есть минимальная пропускная способность, а второе пропорционально квадратному корню из величины потока.

Итак, для задач оптимального распределения пропускных способностей удалось получить точное аналитическое решение, хотя и после введения большого числа упрощающих предположений.

### 2.4.3. Метод девиации потока

Предположим теперь, что пропускные способности каналов известны и требуется найти оптимальное распределение потоков в сети. Сначала необходимо рассмотреть распределение потоков и те ограничения, которым оно должно удовлетворять. Поток в канале li состоит из пакетов, идущих от множества источников к множеству адресатов. Поскольку матрица требований определяет график, который должен идти от каждого источника s ко всем адресатам t, нужно указать распределение потоков по всей сети для каждого из (s,t) типов пакетов. Следовательно, для каждого канала необходимо найти индивидуальные потоки f(s,t,i), а полный поток в канале $\lambda_i$ есть сумма индивидуальных потоков f(s,t,i) от всех пар источник - адресат (s,t), протекающих через этот канал. Это трехпараметрическое пространство неотрицательных величин назовем "потоком" и будем обозначать символом f. Заметим, что индексы s и t пробегают множество узлов сети, а индекс i - множество ее каналов.

Поток f удовлетворяет двум видам ограничений, накладываемых матрицей требований и пропускными способностями каналов связи. Для каждого узла сети можно написать уравнение сохранения для каждого класса пакетов. Эти уравнения представляют собой линейные ограни-

чения, в которых для каждого s и t разность между входящими и выходящими из выделенного узла потоками равна нулю (с учетом пакетов, поступающих извне и покидающих ее).

Если имеются два потока $f_1$ и $f_2$, удовлетворяющие всем ограничениям, то на их основании можно вычислить третий поток, также удовлетворяющий этим ограничениям, а именно, $\alpha f_1+(1-\alpha)f_2$, где $0\leq\alpha\leq1$. Допустимые потоки образуют выпуклое множество, вершины которого (экстремальные потоки) представляют особый интерес.

Метод минимизации T на пространстве допустимых потоков, который описывается в общих чертах, известен под названием "метода девиации потока". При этом методе, начав с некоторого допустимого потока, с помощью частных производных в данной точке определяют градиент отклонения потока для уменьшения T. Затем определяют величину отклонения потока, при которой T будет наименьшим. Этот процесс повторяется до тех пор, пока изменения T на каждом шаге не станут достаточно малыми, что бывает вблизи оптимальной точки. Известно, что задача не имеет локальных оптимумов, и поэтому метод сходится к искомому глобальному оптимуму.

Ключевым моментом метода девиации потоков является способ определения "направления" отклонения потока. Для этого каждому каналу приписывается величина $\partial T/\partial\lambda_i=\mu C_i/\gamma(\mu C_i-\lambda_i i)^2$, которая может считаться "весом" этого канала. Эта величина характеризует влияние малого изменения потока в канале на среднюю задержку, т.е. "сопротивление" увеличению потока. Затем определяется экстремальный поток $f_1$, обладающий минимальным весом при заданном выборе весов. Если $e_1$ является начальным потоком, а $e_2$ - потоком, указывающим направление отклонения потока из $e_1$, то для определения величины отклонения необходимо исследовать все точки $\alpha e_1+(1-\alpha)e_2$, где $0\leq\alpha\leq1$.

Поток $f_1$, который минимизирует T, является наилучшим решением, достижимым на первом шаге алгоритма девиации потока. На следующем шаге частные производные от T и экстремальный поток пересчитываются уже относительно точки $f_2$. Процесс продолжается до тех пор, пока изменения T не станут достаточно малыми.

Для определения начального потока целесообразно использовать веса алгоритма девиации потока, получаемые при нулевых значениях потоков, и по ним построить поток, соответствующий минимальным весам. Если случайно окажется, что этот поток удовлетворяет ограничениям на пропускные способности каналов, то поиск заканчивается. Если нет, можно уменьшить пропорционально все элементы матрицы требований так, чтобы потоки в каналах не превосходили их пропускных способностей. При этом получим допустимый поток, но он слишком мал, чтобы являться решением задачи оптимизации с заданной матрицей требований. На следующем шаге применяем алгоритм девиации потока для минимизации T с

уменьшенной матрицей требований; после этого вновь увеличиваем матрицу требований настолько, чтобы получить максимально возможный поток в пределах заданных ограничений на пропускные способности каналов.

Таким образом, последовательно увеличивая требования и оптимизируя поток, или получим допустимый поток с первоначальной матрицей требований, если такой поток существует, или придется отказаться от попыток его получить.

### 2.4.4. Метод исключения ветвей при выпуклой функции стоимости

В предыдущем разделе рассмотрено применение метода девиации потока к решению задачи выбора пропускных способностей каналов и распределения потоков в предположении, что стоимость линии связи пропорциональна ее пропускной способности. В этом методе потоки направляются согласно весам, представляющим увеличение задержки сообщений при увеличении нагрузки в каналах сети. Если поток в канале становится малым, то пропускная способность также мала и приращение стоимости при увеличении потока становится большим. Как следствие, происходит дальнейшее уменьшение потока в канале. Такие каналы в процессе работы алгоритма пытаются избавиться от проходящих по ним компонент потока; канал с нулевым потоком приобретает вес и в дальнейшем не используется.

На этих соображениях основывается метод оптимизации топологии, в котором в качестве начальной берут сильносвязную сеть, методом девиации потоков решают задачу распределения потоков и выбора пропускных способностей каналов и в получившемся решении в сети сохраняют лишь линии с ненулевыми потоками. Эти линии и образуют искомую топологию.

Оптимизацию топологии сети путем исключения линий можно проводить и в том случае, когда зависимость между стоимостью линии и ее пропускной способностью задается выпуклой функцией. На самом деле существует ограниченный набор возможных пропускных способностей для линий связи, и их стоимости образуют возрастающую ступенчатую функцию. Заменив для удобства анализа эту кривую непрерывной выпуклой функцией, можно применить метод девиации потоков для исключения линий из сильносвязной сети. При этом не следует забывать требований минимальной связности сети и нарушать ее удалением слишком большого числа линий. Такой метод оптимизации топологии сетей известен под названием "метод исключения ветвей при выпуклой функции стоимости". Установлено, что данный метод дает достаточно хорошие результаты, од-

нако порождаемые им топологии сильно зависят от вида функции «стоимость - пропускная способность».

### *2.5. Оценка надежности и коэффициента готовности*

Преимуществом ячеистых сетей передачи данных по сравнению с радиальными, кольцевыми и древовидными является их повышенная устойчивость к возможным отказам линий связи. Следует отметить, что отказ узлов сети также сильно влияют на работоспособность, и поэтому понятие связности, учитывающее узлы и линии, следовательно, является более точной мерой надежности сети, чем реберная связность, учитывающая только состояние линий. Тем не менее для простоты изложения будем рассматривать только отказ линий связи. Учет совместного влияния отказов узлов и линий связи более сложен, однако не вносит принципиальных отличий.

В простейшем случае можно считать, что отказ и восстановление работоспособности каждой линии происходят независимо от остальных, так что ее можно описать единственным параметром $p$, который означает вероятность того, что линия неисправна. Основная цель - выразить характеристику надежности сети как функцию от параметра $p$ и топологии сети. Характеристику надежности сети можно определить различными способами. Например, можно определить ее как долю времени, в течение которого два узла соединены друг с другом, усредненного по множеству всех пар. Если необходимо сохранить связь со всеми узлами сети, требуется более жесткое определение, а именно, вероятность отказа сети $f$, которая определяет вероятность потери связи с любым из ее узлов.

При рассмотрении отказов только линий сеть из m линий может находиться в одном из $2^m$ состояний в зависимости от того, исправна или нет каждая из ее линий. Вероятность каждого из этих состояний сети зависит от того, сколько линий вышло из строя, так как вероятности отказа отдельных линий считаются одинаковыми. Верхнее уравнение в подписи к рис. 10 представляет собой биномиальное разложение, в котором сумма вероятностей различных состояний равна единице. Первое слагаемое этой суммы является вероятностью безотказной работы всех линий сети. Имеется 5 вариантов отказа одной линии, что дает коэффициент 5 при втором члене. Существует 10 различных комбинаций одновременного отказа двух линий и т.д.

Для определения вероятности отказа сети необходимо знать, какие из этих состояний отказа приводят к разъединению сети. Нижнее уравнение (рис. 2.3) описывает вероятность отказа сети f. Для отделения хотя бы одного узла сети должны нарушиться как минимум две связи, поэтому первое ненулевое слагаемое в выражении для f содержит $p^2$. Коэффициент при этом слагаемом определяется тем, сколько пар отказов из 10 возможных отказов двух линий приведут к разъединению сети. На рисунке име-

ются две такие пары; они выделены утолщенными линиями. Это дает коэффициент 2 в выражении для f. Поскольку в любом случае отказа 3, 4 и 5 линий происходит разъединение сети, коэффициенты при этих слагаемых те же, что и в верхнем уравнении, которое описывает всевозможные состояния. Таким образом, вероятность отказа сети f можно выразить полиномом от p, рассматривая все сечения сети. Рис. 2.3 поясняет непосредственный метод подсчета вероятностей.

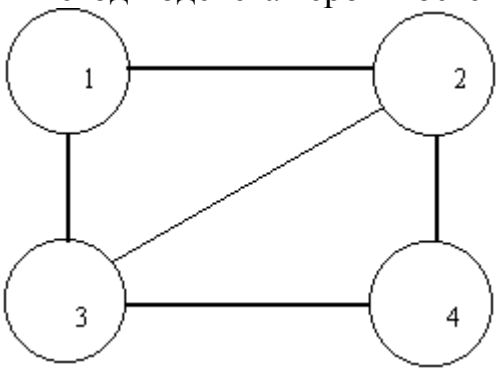



Рис. 2.3. Точное вычисление вероятности отказа:

$$(1-p)^5 + 5p(1-p)^4 + 10p^2(1-p)^3 + 10p^3(1-p)^2 + 5p^4(1-p) + p^5 = 1;$$

$$f = 2p^2(1-p)^3 + 10p^3(1-p)^2 + 5p^4(1-p) + p^5$$

Другой инженерный способ расчета надежности сети можно дать через вероятность отказа сети между некоторой парой узлов *s* и *t*, т.е. вероятность *f*(*s*,*t*) того, что между этими узлами не окажется ни одного пути. Рассмотрим пример (рис. 2.4). Отказ одной линии не может привести к разъединению сети, соединяющей *s* и *t*, поэтому в выражении для вероятности отказа сети не будет членов, содержащих p в первой степени. Количество различных сечений из двух линий, разъединяющих *s* и *t*, определяет коэффициент при $p^2$. Таких сечений всего 6, и они показаны штриховой линией. В подписи приведено приближенное выражение для *f*(*s*,*t*). Множитель, обусловленный вероятностью исправности линий, не включен, так как нет членов, содержащих более высокие степени p. Для типичных вероятностей отказов линий современных сетей часто достаточно использовать лишь первый член разложения. Существует риск ошибки, если число состояний с отказами трех линий велико (сети с лестничной топологией).

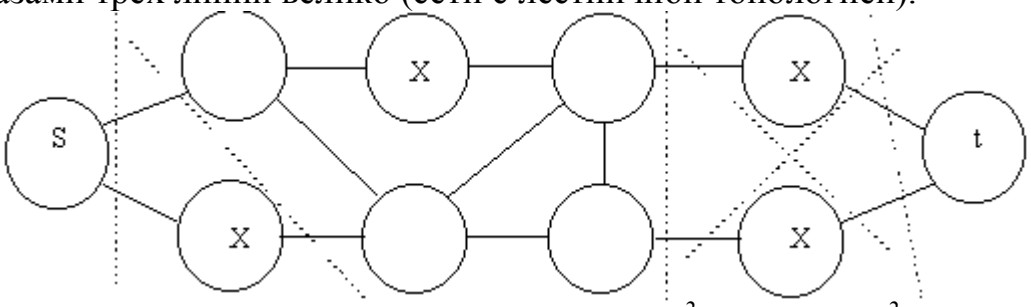



Рис. 2.4. Совокупность сечений. $f(s,t) = 6p^2 + \ldots$ $f = 10p^2 + \ldots$

В примере, приведенном на рис. 2.4, можно так же легко определить первый член разложения вероятности отказа всей сети *f*. Имеются 4 сечения, делающие сеть несвязной, в частности, те, которые отделяют от сети узлы, помеченные знаком *x*. С их учетом коэффициент при первом члене разложения в формуле для *f* возрастает до 10.

Эти приближенные оценки вероятности отказа сети имеют смысл при малых значениях *p*. Их целочисленные коэффициенты в больших сетях могут быть достаточно большими, однако наиболее важным является установить степень в первом ненулевом члене разложения. Эта степень определяется величиной минимального сечения сети, иными словами - реберной связностью.

Все минимальные сечения для малых сетей можно найти простой проверкой. В сетях со связностью 3 и 4 это может оказаться довольно трудным делом. Более того, иногда для больших сетей необходимо точно вычислить полином по *p*. Линии, соединяющиеся последовательно или параллельно, можно объединить и подсчитать вероятность отказа таких комбинаций. Двухтерминальную сеть, которую можно построить таким образом, назовем *ветвью*. Чтобы подсчитать вероятность отказа ветви, рассмотрим три возможных состояния в такой ветви.

В первом (безотказном) состоянии, обозначенном через 0, могут быть неисправные линии, однако нет разъединенных частей и связь между конечными узлами существует.

Во втором состоянии, обозначенном 1, имеются два терминала, связь между которыми оказывается нарушенной, но все внутренние узлы ветви соединены с одним из терминалов. Это состояние замечательно тем, что можно восстановить работу сети, подключив исправную линию параллельно разрыву.

В третьем состоянии, обозначенном 2, имеется совокупность несвязных узлов. Их нельзя достичь из любого терминала, т.е. сеть, содержащая такую ветвь, будет несвязной независимо от других существующих путей. В этом состоянии не возникает вопроса о том, связаны ли между собой два терминала ветви.

На рис. 2.5 показано, как состояние двух ветвей, соединенных последовательно или параллельно, определяет состояние комбинированной ветви. Наличие несвязностей любого компонента, характеризуемого состоянием 2, неизбежно приводит всю ветвь в состояние 2, поскольку несвязность считается отказом сети. При последовательном соединении двух компонент, находящихся в состоянии 1, средний узел ветви не будет иметь связи ни с одним из конечных узлов, и поэтому комбинированная система будет находиться в состоянии 2.

Вероятности можно связать с каждым из этих состояний и, пользуясь таблицами состояний, подсчитать вероятности для трех состояний составной ветви. Для простой линии с вероятностью отказа *p* состояние 2 невоз-

можно, а состояние 1 имеет вероятность *p*. Из таких линий строятся более сложные ветви. С помощью описанных преобразований можно последовательно упрощать сложную сеть, причем этот процесс длится до тех пор, пока в сети не останется ветвей, соединенных последовательно или параллельно. Дальнейшее продвижение в анализе сложных сетей возможно с использованием метода "разложения" относительно одной ветви.

| | 0 | 1 | 2 |
|---|---|---|---|
| 0 | 0 | 1 | 2 |
| 1 | 1 | 2 | 2 |
| 2 | 2 | 2 | 2 |

а

| | 0 | 1 | 2 |
|---|---|---|---|
| 0 | 0 | 0 | 2 |
| 1 | 0 | 1 | 2 |
| 2 | 2 | 2 | 2 |

б

Рис. 2.5. Результирующие состояния последовательных и параллельных комбинаций ветвей: *а* - последовательная; *б* - параллельная

В этом методе выбирается некоторая ветвь, и дальнейший расчет вероятностей ведется тремя различными путями в зависимости от того, в каком из трех состояний находится данная ветвь. Если ветвь находится в состоянии 2, никаких дальнейших расчетов не требуется, так как сеть несвязна. Если ветвь находится в состоянии 1, ее можно удалить из сети, а оставшуюся часть сети можно подвергнуть дальнейшему упрощению путем выделения параллельно-последовательных ветвей. В состоянии 0 два терминала ветви соединены надежно. Эти два узла графа можно объединить, и при удачном результате продолжить упрощение путем выделения параллельно-последовательных ветвей. Успех этой процедуры зависит от удачного выбора ветви, вокруг которой упрощается сеть. В больших сетях к процедуре разложения приходиться обращаться многократно, причем каждый раз число дальнейших расчетов удваивается.

### *2.6. Контрольные вопросы*

1. Сформулируйте наиболее употребительные критерии оптимизации сетей.
2. Какие исходные данные используются при постановке и решении сетевых оптимизационных задач?
3. Что такое топологические ограничения?
4. Каковы особенности решения реальных задач оптимизации?
5. Какие основные подходы применяются при решении оптимизационных задач?
6. Чем обусловлена сложность комбинаторных задач?
7. В чем преимущество эвристических методов решения?
8. Чем примечателен граф Петерсона?
9. Почему вероятность применения регулярных графов при проектировании больших сетей невелика?

10. Опишите способ, с помощью которого можно исследовать различные топологии сети и накапливать локально-оптимальные варианты.

11. Как работает алгоритм Прима?

12. В чем состоит сущность метода насыщения сечения?

13. Почему иерархические сети более экономичны?

14. Каковы особенности построения сетей с концентрирующими узлами?

15. В чем состоит сущность алгоритма Форда-Фалкерсона?

16. В чем проблема многокритериальности задачи выбора пропускных способностей?

17. Что такое метод девиации потока?

18. Какова сущность метода исключения ветвей при выпуклой функции стоимости?

19. Какие инженерные решения по оценке надежности сети Вам известны?

# 3. Многоуровневое управление сетью

Настоящая глава посвящена рассмотрению проблем управления информационно-вычислительными сетями. Первый раздел содержит сведения об ИВС с селекцией информации, которые характеризуются наличием общей среды передачи данных. Во втором разделе освещены вопросы построения сетей с системами коммутации сообщений и пакетов как наиболее распространенными в настоящее время. Маршрутизация информации, основные методы и задачи составляют содержание четвертого раздела. Неотъемлемым элементом системы многоуровневого управления сетью является подсистема обеспечения жизнеспособности и безопасности, представленная в последнем, пятом разделе.

## *3.1. Сети с общим каналом*

### 3.1.1. Принцип селекции информации

Коммуникационная подсеть ИВС с селекцией информации передает информацию от абонентской системы-отправителя всем абонентским системам-получателям, работающим в сети. Далее каждая станция должна проверить адреса всех передаваемых подсетью кадров. Адресованные ей кадры система принимает, а остальные - уничтожает. Только потом абонентская система получает адресованную ей информацию.

Таким образом, в ИВС с селекцией информации в передаче кадров от абонентской системы-отправителя к абонентской системе-получателю участвует не только коммуникационная подсеть (моноканал, поликанал, цик-

лическое кольцо), но и станции. Это привело к созданию и производству таких подсетей, каждая из которых включает как коммуникационную подсеть, так и набор станций. В зависимости от того, сколько уровней протоколов абонентской системы эти станции реализуют, будем их называть транспортными либо канальными. На рис. 3.1 показана логическая структура транспортной подсети. Она состоит из коммуникационной подсети и N станций. Каждая из станций выполняет функции четырех уровней протоколов абонентской системы. К станциям подключаются абоненты сети. Если же станции реализуют только два уровня протоколов - канальный и физический, то логическая структура, изображенная на рис. 3.1, превращается в канальную подсеть.

Общая структура ИВС с селекцией информации имеет вид, изображенный на рис. 3.2. Коммуникационные подсети здесь представлены $q \geq 1$ моноканалами, частотными многоточечными каналами поликанала или циклическими кольцами. Эти q подсетей вместе со станциями образуют транспортную или канальную подсеть.

В сети с селекцией информации могут входить ЭВМ, процессоры с ОЗУ, отдельные процессоры, блоки ОЗУ, магнитные диски, магнитные ленты, принтеры, терминалы и т.д. В зависимости от набора этих абонентов получаются разнообразные виды сетей. Наиболее часто из них используются многомашинная, многопроцессорная и одномашинная.

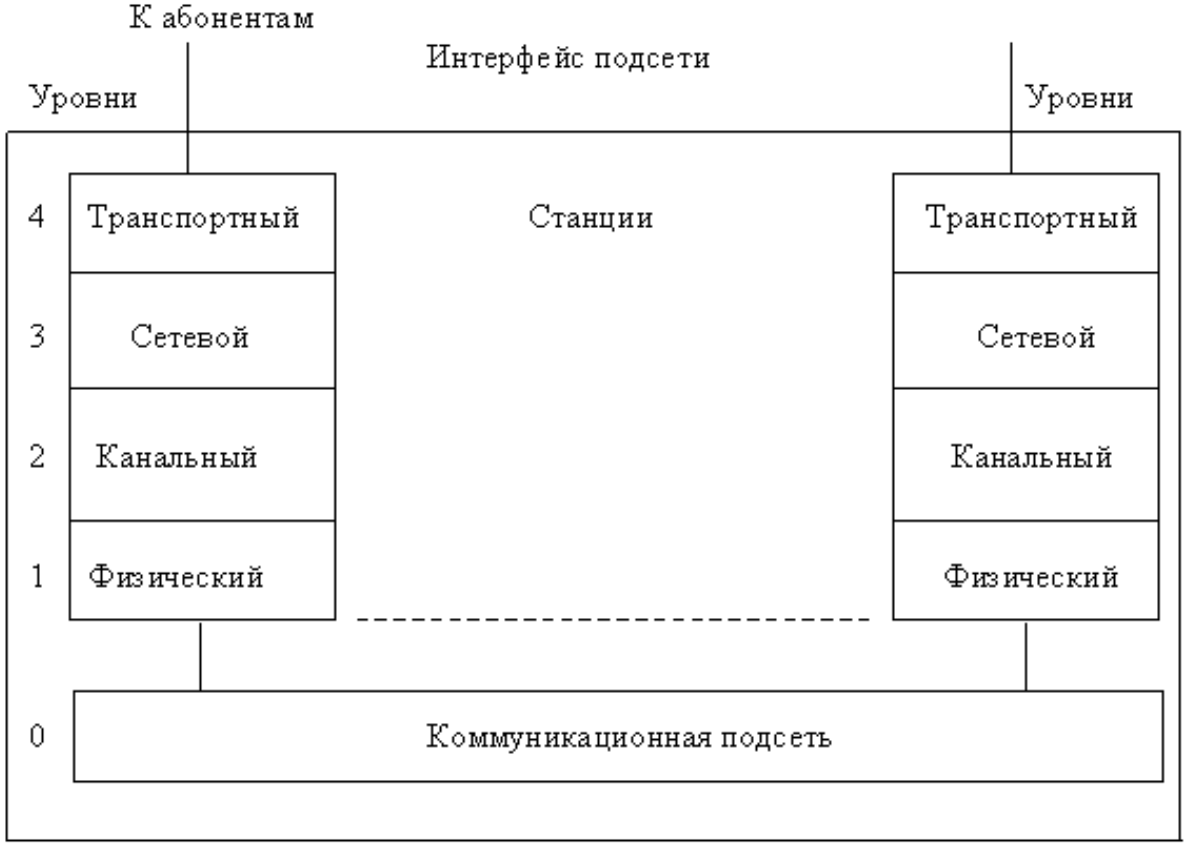



Рис. 3.1. Логическая структура транспортной подсети

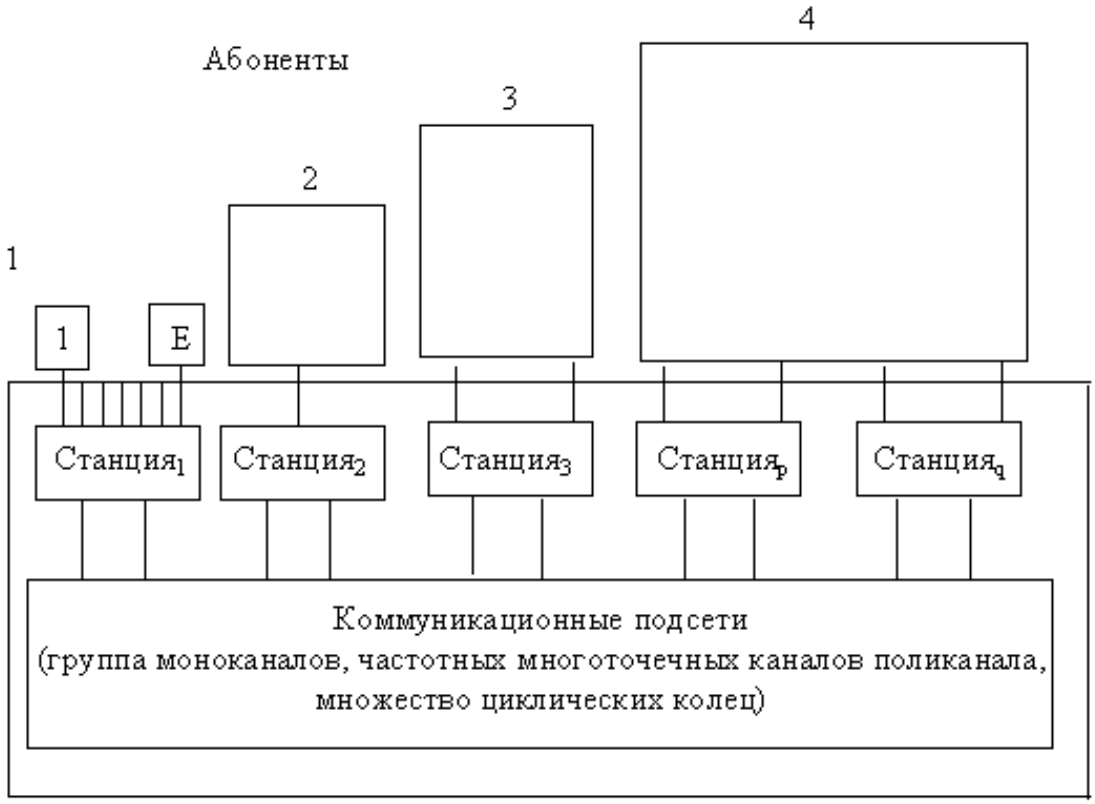



Рис. 3.2. Многомашинная сеть

### 3.1.2. Методы доступа в коммуникационную подсеть

В сетях с селекцией информации к одной и той же коммуникационной подсети может быть подключено значительное число абонентских систем. Однако одновременно через подсеть может вести передачу только одна система. Поэтому возникла проблема разработки таких методов доступа в коммуникационную подсеть, которые обеспечили бы эффективное поочередное ее использование множеством абонентских систем. Существует большое число методов доступа. Но все они могут быть объединены в четыре группы: разделение времени, передача полномочия, случайный доступ и комбинированный метод. Достоинства и недостатки методов представлены в табл. 3.1-3.3.

Таблица 3.1

Метод разделения времени

| **Преимущества метода** | **Недостатки метода** |
|---|---|
| Возможность высококачественной синхронизации прикладных процессов | Ненадежность работы канала из-за наличия в нем диспетчера |
| Простота доступа в многоточечный канал | |
| Легкость выполнения приоритетной политики использования канала | |

Таблица 3.2

Метод передачи полномочия

| **Преимущества метода** | **Недостатки метода** |
|---|---|
| Обеспечение управления синхронными прикладными процессами в реальном времени | Необходимость согласования действий абонентских систем при передаче данных |
| Достаточно полное использование пропускной способности физических средств соединения | |
| Относительная простота диагностики физических средств соединения | |
| Возможность осуществления приоритетов доступа | Раздвоение полномочия |
| Гарантированное время доставки кадров | |

Таблица 3.3

Метод случайного доступа

| **Преимущества метода** | **Недостатки метода** |
|---|---|
| Особенно высокая надежность работы сети | Неопределенное время доставки кадров |
| Независимое функционирование абонентских систем | Неполное использование пропускной способности физических средств соединения |
| Возможность включения новых абонентских систем в работающую сеть без ее остановки | |

Сущность метода разделения времени заключается в том, что в сети имеется устройство, выполняющее функции диспетчера. Его задачей является планирование времени распределения коллективно используемых физических средств соединения. При планировании время работы сети делится на равные либо неравные интервалы, предоставляемые абонентам сети. Во время каждого интервала, в соответствии с принятым алгоритмом, через физические средства соединения данные передает только один из абонентов.

Разделение времени работы физических средств соединения между абонентскими системами является наиболее простым методом.

Метод передачи полномочия заключается в том, что системы передают друг другу полномочие на использование физических средств соединения. Система, получившая полномочие, передает в подсеть разрешенное число кадров и затем направляет полномочие другой системе. Если системе, получившей полномочие, передавать нечего, то она тотчас направляет полномочие далее.

Метод случайного доступа заключается в том, что система передает данные в физические средства соединения, не выполняя явного согласова-

ния с другими системами. Существует много разновидностей этого метода. Однако чаще всего используется метод, связанный с контролем передачи и обнаружением столкновений.

Сущность комбинированного метода доступа заключается в том, что время работы сети делится на чередующиеся интервалы времени. На одном из интервалов для снятия пиковых нагрузок используется метод передачи полномочия, а на других - случайный метод. Комбинированный метод обеспечивает самое полное использование пропускной способности физических средств соединения. Однако он наиболее сложен и требует определенных ресурсов сети.

Выбор наилучшего метода доступа можно связать с областью использования локальной вычислительной сети (табл. 3.4).

Таблица 3.4

Зависимость метода доступа от вида применения

| **Характеристика локальной сети** | **Область использования** | | | | |
|---|---|---|---|---|---|
| | Научно-техническая | Коммерческая | Автоматизация учреждений | Управление в реальном времени | Автоматизация заводов |
| Тип прикладного процесса | Асинхронный | Асинхронный | Асинхронный | Синхронный | Смешанный |
| Допустимое запаздывание в передаче | Ограниченное | Любое | Неограниченное | Относительно ограниченное | Ограниченное |
| Тип режима работы | Смешанный равномерный и неравномерный | Чаще неравномерный | Неравномерный | Смешанный | Смешанный |
| Наилучший метод доступа | Передача полномочия | Случайный доступ или передача полномочия | Случайный доступ | Разделение времени | Передача полномочия |

### 3.1.3. Управление информационным каналом с использованием арбитража

Стремительная компьютеризация современного общества и интеграция производства, а также возрастающая сложность самих объектов

управления диктуют жесткие требования к характеристикам распределенных систем управления технологическими объектами (РСУ ТО). При этом необходимо учитывать, что эффективность работы РСУ ТО во многом определяется функциональными возможностями коммуникационного уровня (КУ) ИВС, которые должны адекватно отражать структурные особенности СУ и специфику управляемого объекта.

Одной из важнейших особенностей рассматриваемых РСУ является однородность входящих в их состав ТО, которые характеризуются равной потребностью в обмене информацией с другими объектами и требуют адекватной ответной реакции со стороны управляющей системы. Эта потребность обусловливает задачу равноправного распределения пропускной способности ИВС между всеми подключенными абонентами. Поставленная задача наиболее ярко выражена на уровне оперативного управления участком производства изделий электромеханики или электроники с характерными параллельными и параллельно-последовательными связями между отдельными ТО и их группами. Равноправное обслуживание абонентов выравнивает загрузку магистрали и обеспечивает резерв повышения производительности ИВС.

Проведем анализ существующих технических средств связи, удовлетворяющих требованиям со стороны РСУ. Известно, что высокой производительностью обменов и малыми аппаратными затратами характеризуются магистральные интерфейсы, а обеспечение равноправного доступа источников к ресурсам сети зависит от способа селекции или арбитража информационного канала. Возможные способы селекции магистральных интерфейсов централизованной структуры представлены на рис. 3.3. Реализация временной селекции магистрали на основе генератора временных интервалов контроллера (см. рис. 3.3, *а*) не обеспечивает нормального функционирования дисциплины "первым пришел - первым обслужен".

Пример пространственной селекции на основе последовательного адресного сканирования источников запроса показан на рис. 3.3, *б*. Данный способ нашел широкое применение в интерфейсе с байтовой магистралью типа HP-IB по стандарту IEC 625-1. Основным достоинством этого способа селекции является гибкость в реализации дисциплин обслуживания. Главный его недостаток - низкое быстродействие.

Схема последовательной (цепочечной) селекции (см. рис. 3.3, *в*) широко распространена в интерфейсах как наиболее простая и достаточно быстродействующая. Поиск источника начинается по сигналу "Запрос". Идентификация наиболее приоритетного устройства выполняется сигналом "Подтверждение", который последовательно проходит через все устройства. В данном случае приоритетным будет устройство, наиболее близко расположенное к контроллеру. При поступлении сигнала "Подтверждение" в источник запроса дальнейшее его прохождение блокируется и устройством выставляется сигнал "Занято".

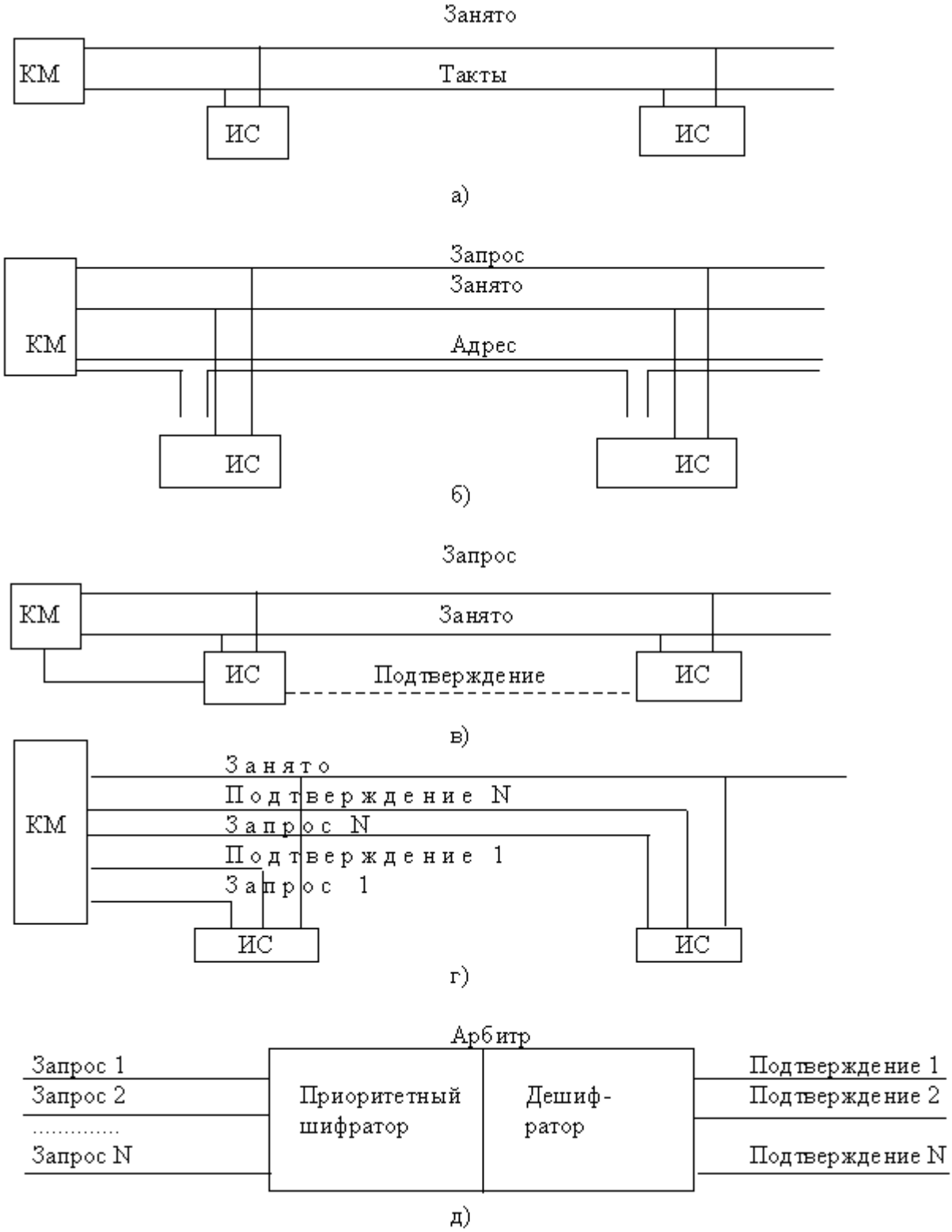



Рис. 3.3. Варианты структур селекции: *а* - реализация временной селекции; *б* - пример пространственной селекции; *в* - схема последовательной селекции; *г* - селекция по выделенным линиям; *д* - схема параллельной селекции с шифратором; КМ - контроллер магистрали ; ИС - интерфейс связи абонента с магистральным интерфейсом

Аналогично цепочечной схеме функционирует и схема селекции по выделенным линиям (см. рис. 3.3, *г*). Отличие ее от предыдущей заключается в том, что общие линии "Запрос" и "Подтверждение" заменяются системой радиальных линий. Данный способ характеризуется гибкостью обслуживания, поскольку контроллер с помощью масок может установить произвольный приоритет и порядок опроса. Однако это достигается за счет существенного увеличения числа линий и усложнения схемотехнического оборудования.

Схема параллельной селекции (см. рис. 3.3, *д*) отличается от предыдущей тем, что арбитраж сети осуществляется приоритетным шифратором, выдающим соответствующий сигнал "Подтверждение.i".

### *3.2. Коммутация сообщений и пакетов*

В 60-70-х годах преобладающим методом передачи данных являлась коммутация сообщений (КоС). Эта технология до сих пор широко используется в электронной почте. Коммутатором обычно является специализированная ЭВМ (СЭВМ).

Именно она отвечает за прием данных с терминалов и ЭВМ, подключенных к СЭВМ посредством вызова набором номера или через выделенную линию. СЭВМ проверяет адрес в головной метке сообщения и коммутирует (направляет) поток данных к принимающему терминалу. В отличие от коммутации цепей в телефонии КоС является технологией типа "запомнить и послать", поскольку при коммутаторах используются запоминающие устройства, обычно дисковые накопители. А так как данные обычно запоминаются, то интерактивный режим и режим реального времени для передачи данных не применяются. Однако данные могут быть переданы через коммутатор сообщений на очень высокой скорости соответствующим определением уровней приоритетов для различных типов потоков данных. Высокоприоритетные потоки задерживаются в очереди на обслуживание более короткое время, чем низкоприоритетные потоки. Таким образом, можно обеспечить интерактивные прикладные задачи. Постановка в очередь на диск также предоставляет возможность сгладить пиковые нагрузки, запоминая низкоприоритетные потоки в период этих нагрузок. Постановка в очередь снижает вероятность случаев блокирования потоков, если какие-либо части сети заняты. Поток может быть временно запомнен, а далее направлен в нужное место, когда оно свободно.

Технология КоС обычно работает с отношением "главный - подчиненный". Традиционно коммутатор выполняет регистрацию и выбор при управлении потоками, входящими и выходящими из него. У КоС имеется три недостатка. Во-первых, в силу структуры "главный - подчиненный" вся сеть выходит из строя при поломке коммутатора. Поэтому многие организации обеспечивают дублированную КоС, которая предлагает выполнение

вторым коммутатором функций первого в случае его поломки. Второй основной недостаток связан с тем, что коммутатор сообщений является потенциально узким методом. Как следствие, появляется увеличенное время обслуживания и малая пропускная способность. И, в третьих, КоС не использует каналы передачи данных с той же эффективностью, как это делают другие подходы.

В связи с этими недостатками была создана другая структура коммутации данных - коммутация пакетов (КоП). КоП уменьшает уязвимость сетей и обеспечивает более эффективное использование каналов связи, чем КоС. Термин "коммутация пакетов" появился из-за того, что пользовательские данные разбиваются на более мелкие порции, которые содержат протокольную информацию, обрамляющую пакеты. Пакеты направляются через сеть как независимые объекты.

КоП первоначально представляла интерес как средство обеспечения секретных переговоров. В 60-е годы исследователи обратились к министерству обороны США с предложением разработать сеть для переключения пакетов, содержащих секретные переговоры. Предполагалось, что каждый отдельный разговор разделяется на малые порции, которые будут направлены по различным каналам в системе. В случае если противник прощупает одну из линий связи и будет способен различать образ акустического сигнала, этот сигнал выявит только часть полных переговоров. Поскольку полные переговоры направлены пакетами по различным путям, отдельная линия не содержит всей передаваемой информации.

Вскоре было установлено, что КоП хорошо работает с потоками данных, поскольку многие терминалы передают потоки данных порциями. Данные передаются в канал, который свободен, пока пользователь терминала вводит данные или пока продолжается пауза в работе пользователя. Время, в течение которого канал не занят, определяет потерянную пропускную способность линии. Одной из концепций КоП является одновременное существование многих передач от нескольких терминалов в одном канале, что означает мультиплексирование с помощью разделения времени линии связи. Этот подход обеспечивает лучшее использование дорогостоящих каналов передачи.

КоП идет дальше, нежели простое мультиплексирование линий связи. Логика пакета может мультиплексировать многие пользовательские сеансы на один порт компьютера. Вместо того, чтобы выделять порт единственному пользователю, система обеспечивает существование частей потоков данных от многих пользователей через один порт. Пользователь же воспринимает порт как выделенный, в то время как пользовательская программа на самом деле разделяет порт с другими пользователями.

Исследование также показало, что потоки данных часто асимметричны, то есть связь осуществляется между терминалами в большей степени в одном направлении, нежели в противоположном. Хорошим примером

асимметричной связи является передача от ЭВМ к терминалу: терминал часто получает меньше данных по сравнению с ЭВМ. КоП осуществляет сглаживание асимметричных потоков в канале, обеспечивая совместное прохождение многих пользователей в канале.

Пакетные сети общего пользования иногда называют носителями с наращиваемой ценностью, поскольку они обеспечивают пользователям возможность наращивания услуг. Например, при добавлении к арендованным линиям пакетных коммутаторов и устройств для сборки/разборки пакетов сеть остается доступной всем, кто пожелает оплатить услуги. Организации, в которых характерны объемы потоков от низкого до среднего, обычно с выгодой подключаются к пакетным сетям общего пользования. Сети общего пользования способны воспринять организации с произвольными потоками для передачи. Для организаций, которые расположены на значительных пространствах, пакетная сеть общего пользования может оказаться более выгодной экономически, поскольку издержки для большинства пакетных сетей общего пользования определяются объемами потоков данных, а не расстояниями.

Технологии коммутации пакетов сопутствовал успех на протяжении последних десяти лет. Подводя итог, основные цели пакетной коммутации таковы: обеспечение мультиплексирования возможностей канала и портов; сглаживание асимметричных потоков между многими пользователями; обеспечение короткого времени реакции для всех пользователей; обеспечение рассредоточения критических компонентов и совместное использование ресурсов.

### *3.3. Маршрутизация информации*

Важной функцией, часто используемой в ИВС, является маршрутизация информации- выбор путей дальнейшей передачи блоков данных в зависимости от адресов их назначений. Для ее обеспечения выполняются следующие функции: установление группы сетевых соединений в одном канальном соединении; доставка упорядоченных последовательностей блоков независимо от маршрута их движения; предоставление приоритетов в коммутации и маршрутизации; объединение нескольких канальных соединений в одно сетевое; сообщение о благополучной доставке последовательности блоков; обнаружение и исправление ошибок, возникающих при передаче по сетевым соединениям; сегментирование и объединение блоков данных; управление потоками; выбор видов сервиса.

ИВС с маршрутизацией информации являются наиболее старым и разработанным классом локальных сетей. Три фактора в последние годы дали толчок к новому этапу развития этих сетей. Первый из них заключается в выпуске небольших и недорогих микропроцессорных коммуникационных систем. Второй фактор состоит в том, что надежность

этих систем и методы их эксплуатации стали таковыми, что системы "закрываются на замок" и управляются дистанционно из центра управления сетью. Третий фактор обусловлен тем, что появилась возможность по одним и тем же каналам передавать не только данные, но и речь, представленную в дискретной форме.

### 3.3.1. Общая характеристика маршрутизации

Маршрутизация в информационно-вычислительных сетях влечет за собой использование логических средств (программных, аппаратных или микропрограммных) в коммутаторах для передвижения пакетов данных сквозь сеть к конечному назначению. Маршрутизация в сети имеет три первичные цели:

1. Обеспечить минимальную возможную задержку и максимальную пропускную способность.
2. Обеспечить прохождение пакета сквозь сеть за минимальную стоимость.
3. Обеспечить каждый пакет максимальной возможной защитой и надежностью.

Приведем некоторые виды классификации методов маршрутизации.

*Классификация по способу управления сетью*

1. Централизованные методы характеризуются наличием единого центра управления, к которому стекается вся информация о загрузке узлов сети или каналов связи. Сбор информации в единый центр связан с дополнительной загрузкой сети из-за передачи служебной информации. Объем этой информации растет пропорционально квадрату размерности системы, и при больших размерностях информация от удаленных элементов поступает со значительной задержкой и не в полной мере отражает состояние сети в текущий момент. Таким образом, задача выбора маршрута в большой сети становится достаточно сложной, и для ее решения приходится применять специальные методы маршрутизации.

2. Децентрализованные методы практически не используют информацию о состоянии удаленных узлов или каналов связи, а учитывают в лучшем случае состояние своих каналов и инцидентных им узлов. Наиболее часто децентрализованные методы применяются в системах с недетерминированной топологической структурой, узлы и объекты которых подключаются к сети и отключаются от нее в случайные моменты времени, а выделить единый центр управления и сбора информации о состоянии сети не представляется возможным. Вместе с тем применение децентрализованных методов в условиях интегрированных информационных и производственных систем представляется разумным, поскольку сами алгоритмы маршрутизации инвариантны относительно содержащего их узла, а объем

необходимо хранимой в узлах информации невелик, что существенно для условий ограниченного объема оперативной памяти управляющих микропроцессорных систем.

3. Распределенные методы характеризуются тем, что каждый узел принимает решение автономно, но с учетом информации, содержащейся в центральном узле. Естественно, эта информация оказывается несколько устаревшей, но в некоторых приложениях она оказывается полезной. Например, распределенные методы целесообразно применять в системах управления технологической подготовкой производства, вариабельность параметров которых имеет более низкую частоту по сравнению с соответствующими параметрами непосредственно технологической системы.

Сводка основных свойств методов приведена в табл. 3.5.

Таблица 3.5

Свойства методов маршрутизации

| **Свойства методов** | | **Принцип управления сетью** | | | **Принятие решений** | |
|---|---|---|---|---|---|---|
| | | Централизовован. | Децентрализов. | Распределен. | Детерминиров. | Вероятностные |
| Топологическая структура сети | | Фиксированн. | Гибкая | Гибкая | Фиксирован. | Гибкая |
| Избыточность служебной информации | | Малая | Средняя | Большая | Малая | Средняя |
| Структурная устойчивость | | Плохая | Хорошая | Хорошая | Плохая | Хорошая |
| Адаптация к изменению трафика | | Хорошая | Средняя | Средняя | Плохая | Средняя |
| Реактивность | Малая сеть | Хорошая | Плохая | Средняя | Хорошая | Средняя |
| | Большая сеть | Средняя | Хорошая | Средняя | Хорошая | Средняя |
| Сложность алгоритмов | | Большая | Малая | Большая | Средняя | Средняя |

*Классификация по способу принятия решений узлом*

1. Детерминированные методы отличаются детерминированным характером принятия решений о выборе того или иного направления передачи. Выбор осуществляется фактически по таблицам принятия решений, которые могут иметь как предопределенную структуру, так и представлять собой дерево целей, переходы по ветвям которого зависят от тех или иных условий.

Детерминированные методы эффективно применяются в высоконадежных сетях со стабильным трафиком, когда дисперсия нагрузки невелика и не возникает необходимости перераспределения потоков информации. Примером таких сетей могут служить коммуникационные подсистемы управления химическим производством, которые имеют практически постоянный трафик на нижних уровнях системы управления и незначительно варьирующийся на верхних.

Однако детерминированные методы маршрутизации оказываются практически неспособными эффективно реагировать на изменение топологической структуры сети в переходных режимах функционирования.

2. Вероятностные методы основаны на том, что при некоторых условиях осуществляется розыгрыш направления передачи транслируемого пакета. Розыгрыш может осуществляться как на основе вычисляемых вероятностей, так и с помощью матрицы поиска, элементы которой содержат вероятности достижения конкретного узла по кратчайшему маршруту. Частным случаем являются градиентно-диффузные алгоритмы, в которых наиболее предпочтительное направление имеет наивысший приоритет, однако в случае занятости его оставшиеся направления разыгрываются случайным образом на равновероятной основе либо с помощью вероятностных таблиц.

Свойства детерминированных и вероятностных методов также сведены в табл. 3.5.

Определим некоторые термины. Эффект размножения пакета означает, что данный пакет генерирует дополнительные, идентичные с ним пакеты. Обход узла состоит в обход поврежденного или занятого узла или канала. Вырождение пакета определяется ослаблением эффекта размножения пакета.

Большинство пакетных сетей выполняют маршрутизацию, используя таблицу или каталог маршрутов. Каталог содержит указания для коммутаторов, как передавать пакет в один из нескольких возможных выходных каналов при переключении. Каталоги пакетных сетей организуются на основании трех подходов:

- фиксированный, или статический, каталог. Изменяется единственный раз при генерации системы. Сохраняется неизменным для всех сеансов;
- каталог, ориентированный на сеансы. Изменяется для каждого сеанса каждого отдельного пользователя. Сохраняется неизменным для отдельного сеанса;
- адаптивный, или динамический, каталог. Изменяется в течение каждого пользовательского сеанса.

Далее системы каталогов можно классифицировать как частичные и полные (по составу маршрутов). Частичные каталоги содержат только узлы, смежные с определенным коммутатором, т.е. узлы, непосредственно

подсоединенные к узлу-коммутатору. Полный каталог содержит весь набор промежуточных узлов, по которому пакет переправится к своему конечному назначению.

Рассмотрим, как работают и где применяются некоторые несложные методы маршрутизации.

### 3.3.2. Лавинные алгоритмы

Одним из подходов к решению задачи маршрутизации в сети является применение лавинных алгоритмов. Используется каждый возможный маршрут между посылающим и принимающим узлами; дубли пакета помещаются по всем выходным каналам и направляются через сеть. Достоинством лавинного метода является то, что, поскольку используются все пути через сеть, первый пакет, который достигнет узла назначения, дойдет с кратчайшей задержкой (что является одной из основных целей маршрутизации в сетях). В то же время при использовании лавинного метода проявляется эффект размножения потоков, а нагрузка на сеть пропорциональна связности сети, т.е. большее число каналов и альтернативных путей создает больший объем общего потока. Однако лавинные алгоритмы предназначены для очень гибких сетей, поскольку копия пакета обязательно проследует до узла назначения, если только существует хотя бы один путь между посылающей и принимающей станциями. Такой метод используется в некоторых военных сетях, поскольку он обеспечивает особую устойчивость в работе.

Эффект размножения потоков можно снизить добавлением определенных средств учета в каждом узле-коммутаторе. Если каждый принимающий узел распознает и уничтожит дублированный пакет, он уменьшает размножение потока. Копии пакетов постепенно исчезают по мере того, как пакеты перемещаются к конечному узлу назначения. Кроме того, каждый пакет может нести в себе собственный маршрут, а узел, найдя себя в маршруте, удаляет пакет из сети.

### 3.3.3. Случайная маршрутизация

Случайная маршрутизация представляет собой метод, используемый для коммутации в сетях коммутации пакетов. В этом подходе необходимо программное обеспечение в каждом узле коммутации для произвольного выбора выходного канала. При чистом режиме случайного выбора маршрутов выходной канал может включить также путь, по которому был получен пакет. Например, если коммутатор пакетов имеет три выходных порта, то он "рандомизирует" пакет по всем трем портам. Следовательно, 33 % времени коммутатор выбирает порт А, 33 % используется порт В; 33 % - порт С.

Для выполнения случайной маршрутизации требуется более сложная логика в коммутаторах, и притоки данных в среднем равномерно распределены по всем коммутаторам. Случайная маршрутизация обеспечивает выравнивание загрузки в сети в целом.

Однако случайная маршрутизация имеет серьезные недостатки. Во-первых, общая длина маршрута через сеть (в среднем) существенно больше, чем при использовании других методов. Во-вторых, большие задержки в сети, которые в большей мере влияют на одну из основных цепей коммутации пакетов - уменьшение задержек. И, в-третьих, пока пакет блуждает по сети, находя в конечном счете узел назначения, существует ненулевая вероятность того, что пакет никогда не достигнет узла назначения. В-четвертых, вследствие "блуждания" пакетов в сети появляется эффект размножения потоков. Из-за перечисленных недостатков этот метод мало используется.

Еще одна из разновидностей случайной маршрутизации носит название метода "горячей картошки". В соответствии с ним узел, получивший пакет, ретранслирует его по первому попавшемуся (но свободному) каналу связи.

### 3.3.4. Фиксированная табличная маршрутизация

Широко используется метод маршрутизации на основе каталогов или таблиц. Например, в установившемся режиме работы сети, топологическая структура которой изображена на рис. 3.4, маршрутная таблица узла 11 приведена в табл. 3.6.

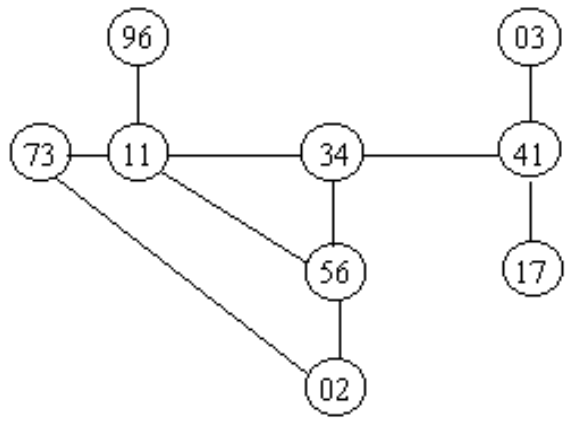



Рис. 3.4. Пример топологической структуры сети

Таблица 3.6

| Узел | 02 | 03 | 17 | 34 | 41 | 56 | 73 | 96 |
|---|---|---|---|---|---|---|---|---|
| Минимальный путь | 2 | 3 | 3 | 1 | 2 | 1 | 1 | 1 |
| Через узел | 56 | 34 | 34 | 34 | 34 | 56 | 73 | 96 |

В случае выхода из строя, например, узла 73 и канала связи 11-56 маршрутная таблица узла 11 изменится так, как это представлено в табл. 3.7 (символом * отмечены изменившиеся столбцы).

Таблица 3.7

| Узел | *02 | 03 | 17 | 34 | 41 | *56 | *73 | 96 |
|---|---|---|---|---|---|---|---|---|
| Минимальный путь | 3 | 3 | 3 | 1 | 2 | 2 | ∇ | 1 |
| Через узел | 34 | 34 | 34 | 34 | 34 | 34 | ∇ | 96 |

Когда пользователь начинает использовать сеть с фиксированной маршрутизацией, необходимо определить класс обслуживания. Например, предпочтительный маршрут может содержать: требование использования наземных линий, а не космической связи вследствие соображений времени отклика; явное указание маршрутов по некоторым линиям, которые более защищены, чем остальные; обход некоторых узлов.

Класс обслуживания определяет список предпочтительных путей, которые называются виртуальными маршрутами. Виртуальный маршрут является логическим маршрутом между двумя конечными точками. Пользовательский сеанс принимает первый действующий маршрут в таблице классов обслуживания. Затем каждый виртуальный маршрут отображается в таблицу, чтобы образовать явные маршруты. Явный маршрут представляет собой последовательность региональных узлов и связей от исходного до конечного регионов. Каждый явный маршрут дополнительно определяется в таблице прохождения маршрутов, которая содержит назначения адресов региона, а также номера явных маршрутов.

### 3.3.5. Адаптивная табличная маршрутизация

Сеть ARRANET является примером использования адаптивного (динамического) каталога маршрутизации. Каждый ее узел системы сохраняет сведения о топологии всей сети и независимо вычисляет оптимальный (кратчайший) путь к каждому узлу назначения. Адаптивные сети функционируют на основе концепции знания смежных узлов, т.е. каждый данный узел осведомлен о статусе всех узлов, которые смежны с ним. Как только пакеты посланы из текущего узла в смежные, программа фиксирует время получения подтверждений приема из смежных узлов. Кроме того, каждый узел знает, сколько у него пакетов осталось для других узлов. Каждые 10 с узел вычисляет задержки на своих выходящих связях. Любое существенное отклонение при изменении задержки рассылается пакетной волной во все остальные узлы. После этого узлы могут использовать полученную информацию для перестройки таблицы маршрутизации.

Адаптивная маршрутизация имеет свои недостатки. Во-первых, программы для ее обработки довольно сложны. Во-вторых, существует вероятность, что пакет потеряется в сети, когда будет двигаться от одного узла

к другому в то время, когда их таблицы маршрутизации изменяются. Однако, если таблицы маршрутизации изменяются не часто, проблема потери пакетов не выглядит серьезно. Первоначально узлы сети ARPANET обменивались пакетами обновления со своими соседними узлами каждые 128 мс, что создавало множество проблем. При современном подходе узлы обновляют свои таблицы каждые 10 с.

Адаптивная маршрутизация также представляет некоторые проблемы при сборке пакетов в узле-приемнике. При фиксированной маршрутизации пакеты прибывают в узел назначения в порядке их отправки из узла-источника. При адаптивной маршрутизации пакеты могут перемещаться в сети по разным маршрутам, поэтому во многих случаях они будут прибывать в конечный пункт с нарушением исходной последовательности. Передача пакетов с нарушением последовательности требует от принимающего узла постановки в очередь и сохранения всех пакетов прежде, чем они будут собраны и выданы пользователю.

## *3.4. Целостность и безопасность сетей*

### 3.4.1. Основные пути утечки информации и несанкционированного доступа в ИВС

В последние годы появилось большое количество сообщений о фактах несанкционированных воздействий на аппаратуру обработки, хранения и передачи информации с нанесением большого материального ущерба. Особо широкий размах получили преступления в системах, обслуживающих банковские и торговые учреждения. Опасность злоумышленных несанкционированных действий над информацией является не просто реальной, а приняла угрожающий характер. Неизбежным следствием этой опасности стали постоянно увеличивающиеся расходы и усилия на защиту информации. Однако для того, чтобы принятые меры оказались эффективными, необходимо выявить возможные каналы утечки информации и пути несанкционированного доступа к защищаемой информации. На рис.3.5 приведены результаты анализа возможных направлений утечки информации и путей несанкционированного доступа в каналах телекоммуникаций.

Особую опасность в настоящее время представляет проблема компьютерного вируса, так как с учетом большого числа его модификаций надежной защиты против него не удалось разработать. Все остальные пути несанкционированного доступа поддаются надежной блокировке при правильно разработанной и реализуемой на практике системе обеспечения безопасности.

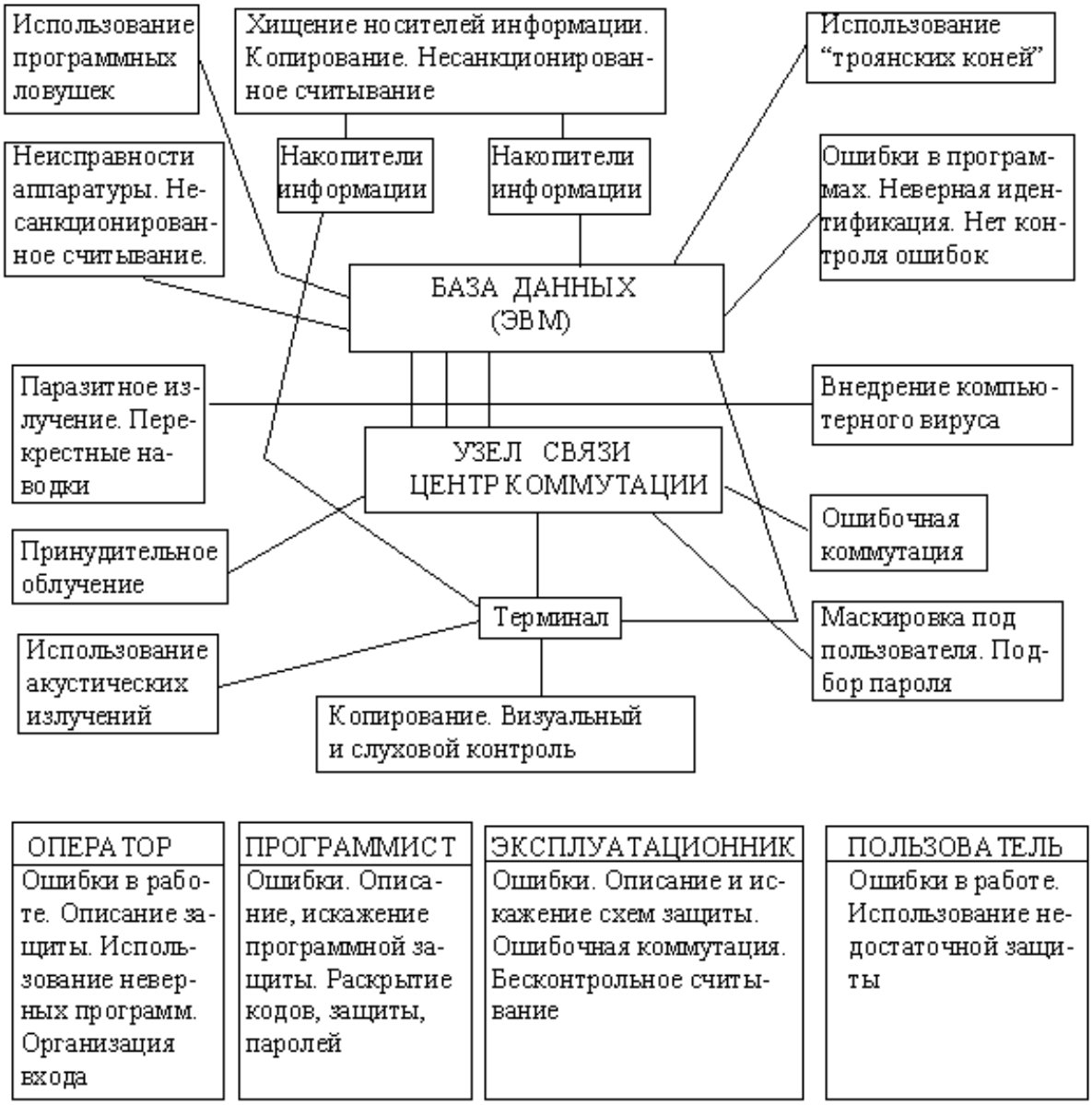



Рис. 3.5. Возможные пути утечки информации при обработке и передаче данных в каналах телекоммуникаций

### 3.4.2. Общая характеристика угроз, служб и механизмов обеспечения безопасности

Комплексное рассмотрение вопросов обеспечение безопасности сетей нашло свое отражение в так называемой архитектуре безопасности, в рамках которой различают угрозы безопасности, а также услуги (службы) и механизмы ее обеспечения.

Под угрозой безопасности понимается действие или событие, которое может привести к разрушению, искажению или несанкционированному использованию ресурсов сети, включая хранимую, передаваемую и обрабатываемую информацию, а также программно-аппаратные средства.

Угрозы делят на случайные (непреднамеренные) и умышленные. Источником первых могут быть ошибки в программном обеспечении, выходы из строя аппаратных средств, неправильные действия пользователей или администрации ИВС и т.п. Умышленные угрозы в свою очередь подразделяют на активные и пассивные и преследуют цель нанесения ущерба пользователям (абонентам) ИВС.

Пассивные угрозы, как правило, направлены на несанкционированное использование информационных ресурсов ИВС, не оказывая при этом влияния на ее функционирование. Пассивной угрозой является, например, попытка получения информации, циркулирующей в каналах передачи данных ИВС, посредством прослушивания.

Активные угрозы имеют целью нарушение нормального процесса функционирования ИВС посредством целенаправленного воздействия на ее аппаратные, программные и информационные ресурсы. К активным угрозам относятся, например, разрушение или радиоэлектронное подавление линий связи ИВС, вывод из строя ЭВМ или ее операционной системы, искажение сведений в пользовательских базах данных или системной информации ИВС. Источниками активных угроз могут быть непосредственные действия пользователей, программные вирусы и т.п.

К основным угрозам безопасности относятся: раскрытие конфиденциальной информации, компрометация информации, несанкционированное использование ресурсов ИВС, несанкционированный обмен информацией, отказ в обслуживании.

Службы безопасности ИВС на концептуальном уровне специализируются на направлениях перечисленных угроз. В свою очередь, указанные направления реализуются необходимыми механизмами безопасности. В рамках идеологии "открытых систем" службы и механизмы безопасности могут использоваться на любом уровне эталонной модели.

В настоящее время документами международной организации стандартизации (МОС) определены следующие службы безопасности: аутентификация (подтверждение подлинности); обеспечение целостности; засекречивание данных; контроль доступа; защита от отказов.

Службы безопасности можно классифицировать как по виду сетей, в которых они применяются (виртуальные, дейтаграммные), и действиям, выполняемым при обнаружении аномальных ситуаций (с восстановлением данных или без него), так и по степени охвата передаваемых данных (блоки в целом либо их элементы, называемые выборочными полями).

### 3.4.3. Компьютерные вирусы

Из наиболее приоритетных направлений работ по обеспечению безопасности ИВС необходимо выделить борьбу с компьютерными вирусными программами (КВП). Предшественниками КВП принято считать так на-

зываемые "троянские программы", тела которых содержат скрытые последовательности команд, выполняющие действия, наносящие вред пользователям. Троянские программы не являются саморазмножающимися и распространяются по ИВС самими программистами, в частности, посредством общедоступных банков данных и программ.

Принципиальное отличие вируса от троянской программы состоит в том, что вирус после запуска его в ИВС существует самостоятельно и в процессе своего функционирования заражает программы путем включения в них своего текста. Таким образом, вирус представляет собой своеобразный генератор троянских программ. Программы, зараженные вирусом, называются также вирусоносителями.

Зараженные программы, как правило, выполняются таким образом, чтобы вирус получил управление раньше самой программы. Для этого он либо встраивается в начало программы, либо имплантируется в ее тело так, что первой командой зараженной программы является безусловный переход на вирус, текст которого заканчивается аналогичной командой безусловного перехода на команду вирусоносителя, бывшую первой до заражения. Получив управление, вирус выбирает следующий файл, заражает его, возможно, выполняет какие-либо другие действия, после чего отдает управление вирусоносителю.

Первичное заражение происходит в процессе поступления инфицированных программ из памяти одной машины в память другой, причем в качестве средства перемещения этих программ могут использоваться как магнитные носители, так и каналы ИВС. Вирусы, использующие для размножения каналы ИВС, принято называть сетевыми.

Цикл жизни вируса обычно включает следующие периоды: внедрения, инкубации, репликации (саморазмножения) и проявления. В течение инкубационного периода вирус пассивен, что усложняет задачу его поиска и нейтрализации. На этапе проявления вирус выполняет свойственные ему целевые функции, например, необратимую коррекцию информации на магнитных носителях.

По характеру размещения в памяти ПЭВМ принято делить вирусы на файловые нерезидентные и резидентные, загрузочные, гибридные и пакетные.

Файловый нерезидентный вирус целиком размещается в исполняемом файле, в связи с чем он активизируется только в случае активизации вирусоносителя, а по выполнении необходимых действий возвращает управление самой программе.

Файловый резидентный вирус отличается от нерезидентного тем, что заражает не только исполняемые файлы, находящиеся во внешней памяти, но и оперативную память ПЭВМ.

Одной из разновидностей резидентных вирусов являются так называемые "загрузочные вирусы". Отличительной особенностью последних

является инфицирование загрузочного сектора магнитного носителя. При этом инфицированными могут быть как загружаемые, так и незагружаемые сменные носители. Голова загрузочного вируса всегда находится в загрузочном секторе, а хвост - в любой другой области носителя.

Близость механизмов функционирования загрузочных и файловых резидентных сделала возможным и естественным проявление файлово-загрузочных, или гибридных, вирусов, инфицирующих как файлы, так и загрузочные секторы.

Особенностью пакетного вируса является размещение его головы в пакетном файле. При этом голова представляет собой строку или программу на язык управления заданиями операционной системы.

Сетевые вирусы, называемые также автономными репликаторами, используют для размножения средства сетевых операционных систем ИВС. Наиболее просто реализуется размножение в тех случаях, когда протоколами ИВС предусмотрен обмен программами. Однако размножение возможно и в тех случаях, когда указанные протоколы ориентированы только на обмен сообщениями.

Эффекты, вызываемые вирусами, принято делить на следующие целевые группы: искажение информации в файлах либо таблице размещения файлов; имитация сбоев аппаратных средств; создание звуковых и визуальных эффектов, включая, например, отображение сообщений, вводящих оператора в заблуждение; инициирование ошибок в программах пользователей или операционной системы; имитация сбоев аппаратных средств.

Наиболее распространенным средством нейтрализации вирусов являются программные антивирусы. Антивирусы, исходя из реализованного в них подхода к выявлению и нейтрализации вирусов, принято делить на следующие группы: детекторы, фаги, вакцины, прививки, ревизоры и мониторы.

Детекторы обеспечивают выявление вирусов посредством просмотра исполняемых файлов и поиска так называемых сигнатур устойчивых последовательностей байтов, имеющихся в телах известных вирусов. Наличие сигнатуры в каком-либо файле свидетельствует о его заражении соответствующим вирусом. Антивирус, обеспечивающий возможность поиска различных сигнатур, называют полидетектором.

Фаги выполняют функции, свойственные детекторам, но, кроме того, "извлекают" инфицированные программы посредством "выкусывания" вирусов из их тел. По аналогии с полидетекторами фаги, ориентированные на нейтрализацию различных вирусов, именуют полифагами.

В отличие от детекторов и фагов вакцины по своему принципу действия напоминают сами вирусы. Вакцина имплантируется в защищаемую программу и запоминает ряд количественных и структурных характеристик последней. Если вакцинированная программа не была к моменту вакцинации инфицированной, то при первом же после заражения запуске

вакцина выполнит проверку соответствия запомненных ею характеристик аналогичным характеристикам, полученным в текущий момент, и при несовпадении будет сделан вывод об изменении текста вакцинированной программы вирусом.

Принцип действия прививок основан на том, что любой вирус, как правило, помечает инфицируемые программы каким-либо признаком, с тем чтобы потом не заражать их повторно. Прививка, не внося никаких других изменений в текст защищаемой программы, помечает ее тем же признаком, что и вирус, который после активации и проверки наличия указанного признака считает ее инфицированной.

Ревизоры обеспечивают слежение за состоянием файловой системы. Программа-ревизор в процессе своего функционирования выполняет сравнение текущих характеристик каждого исполняемого файла с аналогичными характеристиками, полученными в ходе предшествующего просмотра файлов. Если при этом обнаружится, что файл с момента предшествующего просмотра не обновлялся пользователем, а сравниваемые наборы характеристик не совпадают, то файл считается инфицированным.

Монитор представляет собой резидентную программу, перехватывающую потенциально опасные прерывания, характерные для вирусов, и запрашивающую у пользователей подтверждение на выполнение операций, следующих за прерыванием.

Антивирусы рассмотренных типов существенно повышают вирусозащищенность отдельных ПЭВМ и ИВС в целом, однако в связи со свойственными им ограничениями не являются панацеей. Так, для разработки детекторов, фагов и прививок нужно иметь тексты вирусов, что возможно только для выявленных вирусов. Вакцины обладают потенциальной способностью защиты программ не только от известных, но и от неизвестных вирусов, однако обнаруживают факт заражения только в тех случаях, если сами были имплантированы в текст до заражения вирусом. Результативность применения ревизоров зависит от частоты их запуска. Мониторы контролируют процесс функционирования пользовательских программ постоянно, однако характеризуются чрезмерной интенсивностью ложных срабатываний, которые вырабатывают у оператора "эффект подтверждения" и тем самым минимизируют эффект от такого контроля.

Таким образом, большое значение имеют организационные меры и соблюдение определенной технологии защиты от вирусов, предполагающей выполнение следующих этапов: входной контроль дискет, сегментацию информации на жестком диске, защиту системных программ от заражения, систематический контроль целостности и архивацию.

### 3.4.4. Защита операционных систем и баз данных

Существует несколько аспектов проблемы защиты операционных систем, имеющих значение как для ОС автономно функционирующих ЭВМ, так и для сетевых ОС: предотвращение возможности несанкционированного использования и искажения системных ресурсов пользовательскими программами (в частности, вирусами); обеспечение корректности выполнения программ, параллельно функционирующих на одной ЭВМ и использующих общие ресурсы; исключение возможности несанкционированного использования прикладными программами одних пользователей ресурсов, принадлежащих другим, и т.д. Строго говоря, в сетевой ОС и аппаратных средствах ИВС должны быть реализованы механизмы безопасности.

Принято различать пассивные объекты защиты и активные субъекты, которые могут выполнять над объектами определенные операции. Защита объектов реализуется ОС посредством контроля за выполнением субъектами совокупности правил, регламентирующих указанные операции. Указанную совокупность иногда называют статусом защиты.

Субъекты в ходе своего функционирования генерируют запросы на выполнение операций над защищенными объектами. Для реализации статуса защиты в ОС чаще всего используется матрица доступа, содержащая M строк (по числу субъектов) и N столбцов (по числу объектов), причем элемент, находящийся на пересечении i-й строки и j-го столбца, представляет собой множество возможностей i-го субъекта по отношению к j-му объекту.

Еще одним достаточно простым в реализации средством разграничения доступа к защищаемым объектам является механизм колец безопасности. Кольцо безопасности характеризуется своим уникальным номером, причем нумерация идет по принципу "изнутри-наружу", и внутренние кольца являются привилегированными по отношению к высшим. При этом субъекту, оперирующему в пределах кольца с номером i, доступны все объекты в пределах колец с номером j.i.

Для эталонной модели ВОС любая система управления распределенными базами данных (СУ РБД) есть компонента специального программного обеспечения ИВС, использующая свои собственные протоколы только на прикладном уровне. В связи с этим обеспечение безопасности РБД косвенно реализуется в сетевой ОС. Однако все указанные механизмы и средства инвариантны конкретным способам представления информации в БД, возможностям СУ РБД и их языковых средств по доступу к данным, а также возможностям, предоставляемым прикладными программами по работе с БД. Инвариантность приводит к тому, что в случае непринятия специальных мер все пользователи СУ РБД имеют равные права по использованию и обновлению всей информации, имеющейся в базе данных. В то же время указанная информация, как и при ее неавтоматизированном

накоплении и использовании, должна быть разбита на категории по грифу секретности, группам пользователей, которым она доступна, а также по операциям над нею, которые разрешены указанным группам. Реализация этого процесса требует разработки и включения в состав СУ РБД специальных средств.

Известно, что принятие решения о доступе к той или иной информации, имеющейся в БД, может зависеть: от времени и точки доступа; наличия в БД определенных сведений; текучести состояния БД; полномочий пользователя; предыстории обращения к данным.

В первом случае доступ к БД с каждого терминала ИВС может быть ограничен фиксированным временем, после которого любая попытка обращения к БД с этого терминала является запрещенной.

Во втором случае пользователь может получить из БД интересующие его сведения только при условии, что БД содержит некоторую взаимосвязанную с ним информацию определенного содержания.

В третьем случае обновление информации в некоторой БД может быть разрешено пользователю только в те моменты времени, когда она не обновляется другими пользователями.

В четвертом случае для каждого пользователя прикладной программы устанавливаются индивидуальные права на доступ к различным элементам БД. Эти права регламентируют операции, которые пользователь может выполнять над элементами.

В основе пятого фактора лежит то обстоятельство, что интересующую информацию пользователь может получить не непосредственным отбором тех или иных элементов БД, а косвенным путем, т.е. посредством анализа и сопоставления ответов СУБД на последовательно вводимые запросы (команды на обновление данных). В связи с этим для обеспечения безопасности информации в БД в общем случае необходимо учитывать предысторию обращения к данным.

### 3.4.5. Практические рекомендации по обеспечению безопасности информации в коммерческих каналах телекоммуникаций

При обеспечении безопасности возникает сложная для пользователей задача выбора адекватных конкретным обстоятельствам соответствующих технических средств. Поэтому необходимо максимально использовать конкретные условия эксплуатации аппаратуры и учитывать возможные стратегии противоборствующей стороны. Выделим следующие основные направления воздействий.

1. Модификация программного обеспечения.
2. Получение несанкционированного доступа.
3. Выдача себя за другого пользователя.

4. Отказ от факта получения информации, которая на самом деле была получена, или ложные сведения о ее получении.

5. Отказ от факта формирования информации.

6. Утверждение о том, что получателю в определенный момент была послана информация, которая на самом деле не посылалась.

7. Утверждение о том, что информация получена от некоторого пользователя, хотя на самом деле она сформирована им же.

8. Несанкционированное расширение своих законных прав.

9. Несанкционированное изменение прав других пользователей.

10. Подключение к линии связи между другими пользователями.

11. Сокрытие факта наличия некоторой информации (скрытая передача) в другой информации (открытая передача).

12. Изучение прав доступа.

13. Заявление о сомнительности протокола обеспечения безопасности связи из-за раскрытия некоторой информации, которая согласно условиям протокола должна оставаться секретной.

14. Принудительное нарушение протокола.

15. Подрыв доверия к протоколу.

Современная технология обеспечения безопасности связи рекомендует работу по защите информации с учетом перечисленных стратегий проводить по следующим основным направлениям: совершенствование организационных мероприятий; блокирование несанкционированного доступа к обрабатываемой информации; блокирование несанкционированного получения информации с помощью технических средств.

Под организационными мерами защиты понимаются меры общего характера, ограничивающие доступ к ценной информации посторонним лицам, вне зависимости от особенностей метода передачи информации и каналов утечки.

Вся работа по обеспечению безопасности связи в каналах телекоммуникаций должна начинаться с организационных мер защиты.

1. Установление ответственности за обеспечение защиты.

2. Ограничение доступа в помещения, в которых происходят подготовка и обработка информации.

3. Допуск к обработке, хранению и передаче конфиденциальной информации только проверенных должностных лиц.

4. Назначение конкретных образцов технических средств для обработки ценной информации и дальнейшая работа только на них.

5. Хранение магнитных носителей, жестких копий и регистрационных материалов в тщательно закрытых прочных шкафах.

6. Исключение просмотра посторонними лицами содержания обрабатываемых материалов за счет соответствующей установки дисплея, клавиатуры и принтера и т.д.

7. Постоянный контроль работы устройств вывода ценной информации на материальный носитель.

8. Хранение ценной информации только в засекреченном виде.

9. Использование криптографического закрытия при передаче по каналам связи ценной информации.

10. Уничтожение красящих лент, кассет, бумаги или иных материалов, содержащих фрагменты ценной информации.

Учесть специфику канала учета и метода передачи или обработки информации позволяют организационно-технические меры, не требующие для своей реализации нестандартных приемов и оборудования.

1. Организация питания оборудования, обрабатывающего ценную информацию, от отдельного источника питания или через стабилизатор напряжения (сетевой фильтр), мотор-генератор.

2. Ограничение доступа посторонних лиц внутрь оборудования за счет установки механических запорных устройств и прочих механизмов непрерывного действия, обеспечивающих круглосуточный контроль, предотвращающих доступ к охраняемому объекту.

3. Использование жидкокристаллических или плазменных дисплеев при обработке и вводе-выводе информации для отображения, а струйных принтеров - для регистрации.

4. Уничтожение всей информации, содержащейся на магнитных дисках, при отправке в ремонт технических средств

5. Размещение оборудования для обработки ценной информации не менее 2.5 м от устройств освещения, кондиционирования, связи.

6. Установка клавиатуры и принтеров на мягкие прокладки с целью снижения утечки информации по акустическому каналу.

7. Отключение компьютера от локальной сети при обработке ценной информации на нем.

8. Уничтожение информации сразу после ее использования.

Оптимальное решение сложной проблемы обеспечения безопасности связи в настоящее время возможно лишь при комплексном подходе с использованием организационных и технических мер. Достижения микроэлектроники, вычислительной техники и методов криптографического преобразования позволяют оптимистично оценивать перспективы обеспечения безопасности связи. Этому способствует и основная тенденция развития современных систем связи - переход к цифровым методам обработки информации, которые обеспечивают безопасность за счет высокой стойкости криптографического преобразования.

### *3.5. Контрольные вопросы*

1. Сформулируйте отличительные особенности сетей с селекцией информации.

2. Чем отличается транспортная подсеть от канальной?

3. В связи с чем возникла проблема разработки методов доступа в коммуникационную подсеть, обеспечивающих ее эффективное использование?

4. Охарактеризуйте метод разделения времени.

5. Охарактеризуйте метод передачи полномочий.

6. Охарактеризуйте методы случайного доступа.

7. Каким образом необходимо осуществлять выбор наилучшего метода доступа в связи с областью использования сети?

8. Какие положительные свойства несет равноправное обслуживание абонентов ИВС?

9. Каковы особенности реализации временной селекции магистрали на основе генератора временных интервалов?

10. Каковы достоинства и недостатки пространственной селекции на основе последовательного адресного сканирования?

11. Приведите механизм функционирования схемы цепочечной селекции.

12. Охарактеризуйте схему селекции по выделенным линиям.

13. В чем состоит метод коммутации сообщений?

14. Каковы традиционные задачи коммутатора в системе КоС?

15. В связи с чем возникла система коммутации пакетов? В чем ее сущность?

16. Чем отличается коммутация пакетов от простого мультиплексирования линий связи?

17. Каковы положительные аспекты применения систем КоП?

18. Какие задачи решаются функцией маршрутизации? Каковы цели маршрутизации?

19. Приведите известные Вам способы классификации маршрутных алгоритмов.

20. Что такое эффект размножения пакета? обход узла? вырождение пакета? маршрутные таблицы?

21. Охарактеризуйте лавинные алгоритмы маршрутизации.

22. В чем заключается лавинная маршрутизация?

23. Чем отличается фиксированная табличная маршрутизация от адаптивной?

24. Каков порядок использования фиксированной табличной маршрутизации?

25. Каковы достоинства и недостатки адаптивной табличной маршрутизации?

26. В связи с чем возникают проблемы безопасности и целостности информационно-вычислительных сетей?

27. Приведите основные пути утечки информации и несанкционированного доступа в ИВС.

28. Что такое угроза безопасности сети? Какие виды угроз Вы знаете?

29. Что такое компьютерные вирусы? Какие существуют разновидности компьютерных вирусов?

30. Какими средствами можно бороться с компьютерными вирусами? Какие меры предосторожности необходимо предпринимать?

31. Какие методы применяются для защиты операционных систем и баз данных?

32. От каких факторов зависит принятие решения о доступе к той или иной информации, имеющейся в БД?

33. Каковы основные направления реализации современной технологии обеспечения безопасности ИВС?

34. Какие Вам известны организационные и организационно-технические меры защиты?

# 4. Сетевые операционные системы

## *4.1. Структура сетевой операционной системы*

Сетевая операционная система составляет основу любой вычислительной сети. Каждый компьютер в сети в значительной степени автономен, поэтому под сетевой операционной системой в широком смысле понимается совокупность операционных систем 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 [26–36].

В сетевой операционной системе отдельной машины можно выделить несколько частей:

- Средства управления локальными ресурсами компьютера: функции распределения оперативной памяти между процессами, планирования и диспетчеризации процессов, управления процессорами в мультипроцессорных машинах, управления периферийными устройствами и другие функции управления ресурсами локальных ОС.
- Средства предоставления собственных ресурсов и услуг в общее пользование - серверная часть ОС (сервер). Эти средства обеспечивают, например, блокировку файлов и записей, что необходимо для их совместного использования; 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.

- Средства запроса доступа к удаленным ресурсам и услугам и их использования - клиентская часть ОС (редиректор). Эта часть выполняет распознавание и перенаправление в сеть запросов к удаленным ресурсам от приложений и пользователей, при этом запрос поступает от приложения в локальной форме, а передается в сеть в другой форме, соответствующей требованиям сервера. Клиентская часть также осуществляет прием ответов от серверов и преобразование их в локальный формат, так что для приложения выполнение локальных и удаленных запросов неразличимо.
- 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., то есть является средством транспортировки сообщений.

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

На практике сложилось несколько подходов к построению сетевых операционных систем.

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. Примером такого подхода является использование на каждой машине сети операционной системы MS DOS (у которой начиная с ее третьей версии появились такие встроенные функции, как блокировка файлов и записей, необходимые для совместного доступа к файлам). Принцип построения сетевых ОС в виде сетевой оболочки над локальной ОС используется и в современных ОС, таких, например, как LANtastic или Personal Ware.

Однако более 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. Примером такой ОС является нестареющая ветвь системы Windows NT фирмы Microsoft.

### *4.2. 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 двухранговые. Последние чаще называют сетями с выделенными серверами.

Если компьютер предоставляет свои ресурсы другим пользователям сети, то он играет роль сервера. При этом компьютер, обращающийся к ре-

сурсам другой машины, является клиентом. Как уже было сказано, компьютер, работающий в сети, может выполнять функции либо клиента, либо сервера, либо совмещать обе эти функции.

Если выполнение каких-либо серверных функций является основным назначением компьютера (например, предоставление файлов в общее пользование всем остальным пользователям сети или организация совместного использования факса, или предоставление всем пользователям сети возможности запуска на данном компьютере своих приложений), то такой компьютер 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.

Очевидно, что на выделенных серверах желательно устанавливать ОС, специально оптимизированные для выполнения тех или иных серверных функций. Поэтому в сетях с выделенными серверами чаще всего используются сетевые операционные системы, в состав которых входит нескольких вариантов ОС, отличающихся возможностями серверных частей.

Выделенный сервер не принято использовать в качестве компьютера для выполнения текущих задач, не связанных с его основным назначением, так как это может уменьшить производительность его работы как сервера.

Важно понять, что несмотря на то, что в сети с выделенным сервером все компьютеры в общем случае могут выполнять одновременно роли и сервера, и клиента, эта сеть функционально не симметрична: аппаратно и программно в ней реализованы два типа компьютеров - одни, в большей степени ориентированные на выполнение серверных функций и работающие под управлением специализированных серверных ОС, а другие - в основном выполняющие клиентские функции и работающие под управлением соответствующего этому назначению варианта ОС. Функциональная несимметричность, как правило, вызывает и несимметричность аппаратуры - для выделенных серверов используются более мощные компьютеры с большими объемами оперативной и внешней памяти. Таким образом, функциональная несимметричность в сетях с выделенным сервером сопровождается несимметричностью операционных систем (специализация ОС) и аппаратной несимметричностью (специализация компьютеров).

В одноранговых сетях все компьютеры равны в правах доступа к ресурсам друг друга. Каждый пользователь может по своему желанию объявить какой-либо ресурс своего компьютера разделяемым, после чего другие пользователи могут его эксплуатировать. В таких сетях на всех компьютерах устанавливается одна и та же ОС, которая предоставляет всем компьютерам в сети *потенциально* равные возможности.

В одноранговых сетях также может возникнуть функциональная несимметричность: одни пользователи не желают разделять свои ресурсы с другими, и в таком случае их компьютеры выполняют роль клиента, за другими компьютерами администратор закрепил только функции по орга-

низации совместного использования ресурсов, а значит они являются серверами, в третьем случае, когда локальный пользователь не возражает против использования его ресурсов и сам не исключает возможности обращения к другим компьютерам, ОС, устанавливаемая на его компьютере, должна включать и серверную, и клиентскую части. В отличие от сетей с выделенными серверами, в одноранговых сетях отсутствует специализация ОС в зависимости от преобладающей функциональной направленности - клиента или сервера. Все вариации реализуются средствами конфигурирования одного и того же варианта ОС.

Одноранговые сети проще в организации и эксплуатации, однако они применяются в основном для объединения небольших групп пользователей, 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 двухранговые сети, где сервер лучше решает задачу обслуживания пользователей своими ресурсами, так как его аппаратура и сетевая операционная система специально спроектированы для этой цели.

### *4.3. ОС для рабочих групп и ОС для сетей масштаба предприятия*

Сетевые операционные системы имеют разные свойства в зависимости от того, предназначены они для сетей масштаба рабочей группы (отдела), для сетей масштаба кампуса или для сетей масштаба предприятия.

- *Сети отделов* - используются небольшой группой сотрудников, решающих общие задачи. Главной целью сети отдела является разделение локальных ресурсов, таких как приложения, данные, лазерные принтеры и модемы. Сети отделов обычно не разделяются на подсети.
- *Сети кампусов* - соединяют несколько сетей отделов внутри отдельного здания или внутри одной территории предприятия. Эти сети являются все еще локальными сетями, хотя и могут покрывать территорию в несколько квадратных километров. Сервисы такой сети включают взаимодействие между сетями отделов, доступ к базам данных предприятия, доступ к факс-серверам, высокоскоростным модемам и высокоскоростным принтерам.
- *Сети предприятия (корпоративные сети)* - объединяют все компьютеры всех территорий отдельного предприятия. Они могут покрывать город, регион или даже континент. В таких сетях пользователям предоставляется доступ к информации и приложениям, находящимся в других рабочих группах, других отделах, подразделениях и штаб-квартирах корпорации.

Главной задачей операционной системы, используемой в сети масштаба отдела, является организация разделения ресурсов, таких как приложения, данные, лазерные принтеры и, возможно, низкоскоростные модемы. Обычно сети отделов имеют один или два файловых сервера и не более чем 30 пользователей. Задачи управления на уровне отдела относительно просты. В задачи администратора входит добавление новых пользователей, устранение простых отказов, инсталляция новых узлов и установка новых версий программного обеспечения. Операционные системы сетей отделов хорошо отработаны и разнообразны, также как и сами сети отделов, уже давно применяющиеся и достаточно отлаженные. Такая сеть обычно использует одну или максимум две сетевые ОС.

Пользователи и администраторы сетей отделов вскоре осознают, что они могут улучшить эффективность своей работы путем получения доступа к информации других отделов своего предприятия. Если сотрудник, занимающийся продажами, может получить доступ к характеристикам конкретного продукта и включить их в презентацию, то эта информация будет более свежей и будет оказывать большее влияние на покупателей. Если отдел маркетинга может получить доступ к характеристикам продукта, который еще только разрабатывается инженерным отделом, то он может быстро подготовить маркетинговые материалы сразу же после окончания разработки.

Итак, следующим шагом в эволюции сетей является объединение локальных сетей нескольких отделов в единую сеть здания или группы зданий. Такие сети называют сетями кампусов. Сети кампусов могут простираться на несколько километров, но при этом глобальные соединения не требуются.

Операционная система, работающая в сети кампуса, должна обеспечивать для сотрудников одних отделов доступ к некоторым файлам и ресурсам сетей других отделов. Услуги, предоставляемые ОС сетей кампусов, не ограничиваются простым разделением файлов и принтеров, а часто предоставляют доступ и к серверам других типов, например, к факс-серверам и к серверам высокоскоростных модемов. Важным сервисом, предоставляемым операционными системами данного класса, является доступ к корпоративным базам данных, независимо от того, располагаются ли они на серверах баз данных или на миникомпьютерах.

Именно на уровне сети кампуса начинаются проблемы интеграции. В общем случае, отделы уже выбрали для себя типы компьютеров, сетевого оборудования и сетевых операционных систем. Очень часто сеть кампуса соединяет разнородные компьютерные системы, в то время как сети отделов используют однотипные компьютеры.

Корпоративная сеть соединяет сети всех подразделений предприятия, в общем случае находящихся на значительных расстояниях. Корпо-

ративные сети используют глобальные связи (WAN links) для соединения локальных сетей или отдельных компьютеров.

Пользователям корпоративных сетей требуются все те приложения и услуги, которые имеются в сетях отделов и кампусов, плюс некоторые дополнительные приложения и услуги, например, доступ к приложениям мейнфреймов и миникомпьютеров и к глобальным связям. Когда ОС разрабатывается для локальной сети или рабочей группы, то ее главной обязанностью является разделение файлов и других сетевых ресурсов (обычно принтеров) между локально подключенными пользователями. Такой подход не применим для уровня предприятия. Наряду с базовыми сервисами, связанными с разделением файлов и принтеров, сетевая ОС, которая разрабатывается для корпораций, должна поддерживать более широкий набор сервисов, в который обычно входят почтовая служба, средства коллективной работы, поддержка удаленных пользователей, факс-сервис, обработка голосовых сообщений, организация видеоконференций и др.

Кроме того, многие существующие методы и подходы к решению традиционных задач сетей меньших масштабов для корпоративной сети оказались непригодными. На первый план вышли такие задачи и проблемы, которые в сетях рабочих групп, отделов и даже кампусов либо имели второстепенное значение, либо вообще не проявлялись. Например, простейшая для небольшой сети задача ведения учетной информации о пользователях выросла в сложную проблему для сети масштаба предприятия. А использование глобальных связей требует от корпоративных ОС поддержки протоколов, хорошо работающих на низкоскоростных линиях, и отказа от некоторых традиционно используемых протоколов (например, тех, которые активно используют широковещательные сообщения). Особое значение приобрели задачи преодоления гетерогенности - в сети появились многочисленные шлюзы, обеспечивающие согласованную работу различных ОС и сетевых системных приложений.

К признакам корпоративных ОС могут быть отнесены также следующие особенности.

*Поддержка приложений.* В корпоративных сетях выполняются сложные приложения, требующие для выполнения большой вычислительной мощности. Такие приложения разделяются на несколько частей, например, на одном компьютере выполняется часть приложения, связанная с выполнением запросов к базе данных, на другом - запросов к файловому сервису, а на клиентских машинах - часть, реализующая логику обработки данных приложения и организующая интерфейс с пользователем. Вычислительная часть общих для корпорации программных систем может быть слишком объемной и неподъемной для рабочих станций клиентов, поэтому приложения будут выполняться более эффективно, если их наиболее сложные в вычислительном отношении части перенести на специально предназначенный для этого мощный компьютер - *сервер приложений.*

Сервер приложений должен базироваться на мощной аппаратной платформе (мультипроцессорные системы, часто на базе RISC-процессоров, специализированные кластерные архитектуры). ОС сервера приложений должна обеспечивать высокую производительность вычислений, а значит поддерживать многонитевую обработку, вытесняющую многозадачность, мультипроцессирование, виртуальную память и наиболее популярные прикладные среды.

*Справочная служба.* Корпоративная ОС должна обладать способностью хранить информацию обо всех пользователях и ресурсах таким образом, чтобы обеспечивалось управление ею из одной центральной точки. Подобно большой организации, корпоративная сеть нуждается в централизованном хранении как можно более полной справочной информации о самой себе (начиная с данных о пользователях, серверах, рабочих станциях и кончая данными о кабельной системе). Естественно организовать эту информацию в виде базы данных. Данные из этой базы могут быть востребованы многими сетевыми системными приложениями, в первую очередь системами управления и администрирования. Кроме этого, такая база полезна при организации электронной почты, систем коллективной работы, службы безопасности, службы инвентаризации программного и аппаратного обеспечения сети, да и для практически любого крупного бизнес-приложения.

База данных, хранящая справочную информацию, предоставляет все то же многообразие возможностей и порождает все то же множество проблем, что и любая другая крупная база данных. Она позволяет осуществлять различные операции поиска, сортировки, модификации и т.п., что очень сильно облегчает жизнь как администраторам, так и пользователям. Но за эти удобства приходится расплачиваться решением проблем распределенности, репликации и синхронизации.

В идеале сетевая справочная информация должна быть реализована в виде единой базы данных, а не представлять собой набор баз данных, специализирующихся на хранении информации того или иного вида, как это часто бывает в реальных операционных системах.

*Безопасность.* Особую важность для ОС корпоративной сети приобретают вопросы безопасности данных. С одной стороны, в крупномасштабной сети объективно существует больше возможностей для несанкционированного доступа - из-за децентрализации данных и большой распределенности "законных" точек доступа, из-за большого числа пользователей, благонадежность которых трудно установить, а также из-за большого числа возможных точек несанкционированного подключения к сети. С другой стороны, корпоративные бизнес-приложения работают с данными, которые имеют жизненно важное значение для успешной работы корпорации в целом. И для защиты таких данных в корпоративных сетях наряду с различными аппаратными средствами используется весь спектр средств

защиты, предоставляемый операционной системой: избирательные или мандатные права доступа, сложные процедуры аутентификации пользователей, программная шифрация.

## *4.4. Проблемы взаимодействия операционных систем в гетерогенных сетях*

### 4.4.1. Понятия "internetworking" и "interoperability"

До недавнего времени проблемы межсетевого взаимодействия не очень волновали отечественных пользователей и системных администраторов. Они уютно себя чувствовали в замкнутом мире IBM PC совместимых компьютеров, сетей Novell и сетевых адаптеров Ethernet, хотя в "большом" мире многие фирмы, в том числе и Novell, успешно продавали различные средства межсетевой связи. Однако пора монокультурного развития отечественных сетей заканчивается, организации приобретают различную технику, например, бизнес-серверы Hewlett-Packard, графические станции Sun или Silicon Graphics, и другую не менее достойную аппаратуру с разнообразными операционными системами, поэтому проблемы, характерные для западных корпоративных сетей, постепенно становятся актуальными и для нас.

Прежде, чем приступить к рассмотрению межсетевого взаимодействия, уточним, что понимается под термином "сеть". Этот термин может употребляться в широком смысле (сеть - это совокупность связанных между собой компьютеров) и в узком смысле (сеть - это совокупность компьютеров, соединенных между собой в соответствии с одной из стандартных типовых топологий - шина, звезда, кольцо, и использующих для передачи пакетов один из протоколов канального уровня, определенный для этой топологии).

Каждая сеть имеет свой номер, который используется на сетевом уровне при выполнении маршрутизации. Когда две или более сетей организуют совместную транспортную службу, то такой режим взаимодействия обычно называют *межсетевым взаимодействием (internetworking)*. Для обозначения составной сети в англоязычной литературе часто также используются термины *интерсеть* (*internetwork* или *internet).* Интерсеть обеспечивает только передачу пакетов, не занимаясь их содержанием.

Так как понятие "номер сети" определяется на сетевом уровне, то оно является относительным, то есть, если в одном компьютере установлено программное обеспечение, поддерживающее несколько протоколов сетевого уровня (например, TCP и SPX), то данный компьютер может принадлежать нескольким сетям.

При рассмотрении вопросов межсетевого взаимодействия часто используется еще один англоязычный термин - *interoperability.* В то время

как термин internetworking обозначает взаимодействие сетей на нижних уровнях, непосредственно связанных с транспортировкой пакетов, в понятие interoperability входит обеспечение согласования верхних уровней стека коммуникационных протоколов, реализуемых серверами и редикторами операционных систем и некоторыми сетевыми приложениями.

### 4.4.2. Гетерогенность

Только небольшое количество сетей обладает *однородностью (гомогенностью)* программного и аппаратного обеспечения. Однородными чаще являются сети, которые состоят из небольшого количества компонентов от одного производителя.

Немногие организации имеют сети, составленные из оборудования, например, только IBM или DEC. Дни сетей от одного производителя миновали. Нормой сегодняшнего дня являются сети *неоднородные (гетерогенные),* которые состоят из различных рабочих станций, операционных систем и приложений, а для реализации взаимодействия между компьютерами используют различные протоколы. Разнообразие всех компонентов, из которых строится сеть, порождает еще большее разнообразие структур сетей, получающихся из этих компонентов. А если продолжить далее, и рассмотреть более сложные образования, получающиеся в результате объединения отдельных сетей в единую большую сеть, то становится понятным то множество проблем, связанных с проектированием, администрированием и управлением такой гетерогенной интерсетью. В идеале это объединение неоднородных сетей должно быть прозрачным для пользователя.

Так как сети создавались большей частью случайным образом, то приобретенные компьютеры и ОС отвечают, как правило, индивидуальным потребностям группы пользователей. Сети отделов строились для решения конкретных задач групп сотрудников. Отдел, отвечающий за автоматизацию предприятия, должен *интегрировать* все эти плохо совместимые системы в единый прозрачный организм.

Добавление в вычислительную сеть новых, чужеродных элементов может происходить и при всякой значительной реорганизации предприятия, например, при смене владельца, что для нашей страны сейчас тоже становится весьма обычным делом. В этом случае вновь приобретенное предприятие и его вычислительное оборудование также должно быть интегрировано в общую структуру предприятия нового владельца.

## *4.5. Основные подходы к реализации взаимодействия сетей*

Основные проблемы при организации взаимодействия различных сетей связаны с тем, что эти сети используют различные стеки коммуникационных протоколов. В каждом конкретном стеке протоколов средства,

реализующие какой-либо уровень, обеспечивают интерфейс для вышележащего уровня своей системы и пользуются услугами интерфейсных функций нижележащего уровня.

Если бы в компьютерном мире существовал только один стек протоколов, то у независимых разработчиков сетевого и программного обеспечения не было бы никаких проблем: сетевые адаптеры вместе со своими драйверами подходили бы к любой сетевой ОС за счет единого интерфейса между канальным и сетевым уровнями, разработчики транспортных средств новых ОС могли бы использовать существующие реализации протокола сетевого уровня, а разработчики сетевых приложений использовали бы единый API для обращения к сервисным услугам прикладного уровня ОС. К сожалению, в реальном мире компьютерных сетей существует несколько стеков протоколов, уже завоевавших свое место под солнцем и не собирающихся его уступать.

Существование многих стеков протоколов не вносит никаких проблем до тех пор, пока не появляется потребность в их взаимодействии. В этих случаях проявляется несовместимость близких по назначению, но различных по форматам данных и алгоритмам протоколов.

Общность различных стеков протоколов проявляется только на нижних уровнях - физическом и канальном. Здесь в настоящее время почти нет проблем для взаимодействия, так как большинство стеков могут использовать общие протоколы Ethernet, Token Ring, FDDI. Исключение составляют только мейнфреймы IBM, которые на нижнем уровне в основном используют протоколы типа ведущий-ведомый с синхронной передачей данных, ориентированные на иерархическую соподчиненную структуру мейнфрейм - групповой контроллер - терминалы. Да и соединение двух компьютеров, использующих на нижнем уровне различные протоколы, а на верхних - одинаковые не составляет проблемы - эта задача решается аппаратно с помощью транслирующего моста или маршрутизатора.

Сложнее обстоит дело с сопряжением сетей, использующие различные протоколы верхних уровней, начиная с сетевого. Задачи согласования протоколов верхних уровней решить труднее из-за большей сложности этих протоколов и их разнообразия - чем большим интеллектом обладает протокол, тем больше у него аспектов и граней, по которым он может отличаться от своего собрата по функциональному назначению. Сложно осуществить трансляцию транспортных протоколов (таких, как IP и IPX), но гораздо сложнее совместить протоколы верхнего, прикладного уровня, с помощью которых клиенты получают сервис у серверов.

Если рассмотреть наиболее часто используемый в сетях сервис, а именно, файловый сервис, то различия в протоколах файлового сервиса в первую очередь связаны с различиями структур файловых систем. Например, пользователю Windows непривычны приемы монтирования файловой системы UNIX в одно дерево, он хочет работать с разрозненными файло-

выми системами отдельных носителей, отображенными на буквы английского алфавита. Команды, используемые при работе с различными файловыми системами, также различны как по названию, так и по содержанию. Кроме того, даже для одной файловой системы в различных операционных системах предусмотрены различные удаленные сервисы. В ОС UNIX можно работать с удаленной файловой системой с помощью символьных команд протокола прикладного уровня FTP, переписывая файлы с удаленной машины на локальную по одному, а можно работать с протоколом NFS, который обеспечивает монтирование удаленной системы в локальную и требует других команд и приемов. Поэтому проблемы, возникающие на верхних уровнях, гораздо сложнее, чем проблемы замены заголовка пакета на канальном уровне.

Для организации взаимодействия различных сетей в настоящее время используется два подхода.

Первый подход связан с использованием так называемых *шлюзов*, которые обеспечивают согласование двух стеков протоколов путем преобразования (трансляции) протоколов (рис. 4.1, *а*). Шлюз размещается между взаимодействующими сетями и служит посредником, переводящим сообщения, поступающие от одной сети, в формат другой сети.

Второй подход заключается в том, что в операционные системы серверов и рабочих станций встраиваются несколько мирно сосуществующих наиболее популярных стеков протоколов. Такая технология получила название *мультиплексирования стеков протоколов*. За счет ее использования либо клиентские запросы используют стек протоколов той сети, к которой относятся нужные серверы, либо серверы подключают стек протоколов, соответствующий поступившему клиентскому запросу (рис. 4.1, *б*).

Взаимодействие компьютеров, принадлежащих разным сетям, напоминает общение людей, говорящих на разных языках. Для достижения взаимопонимания они также могут использовать два подхода: пригласить переводчика (аналог шлюза), или перейти на язык собеседника, если они им владеют (аналог мультиплексирования стеков протоколов).

Каждый из подходов имеет свои преимущества и недостатки, на которых мы остановимся позже.

### 4.5.1. Шлюзы

Итак, шлюз согласует коммуникационные протоколы одного стека с коммуникационными протоколами другого стека. Программные средства, реализующие шлюз, нет смысла устанавливать ни на одном из двух взаимодействующих компьютеров с разными стеками протоколов, гораздо рациональнее разместить их на некотором компьютере-посреднике. Прежде, чем обосновать это утверждение, рассмотрим принцип работы шлюза.

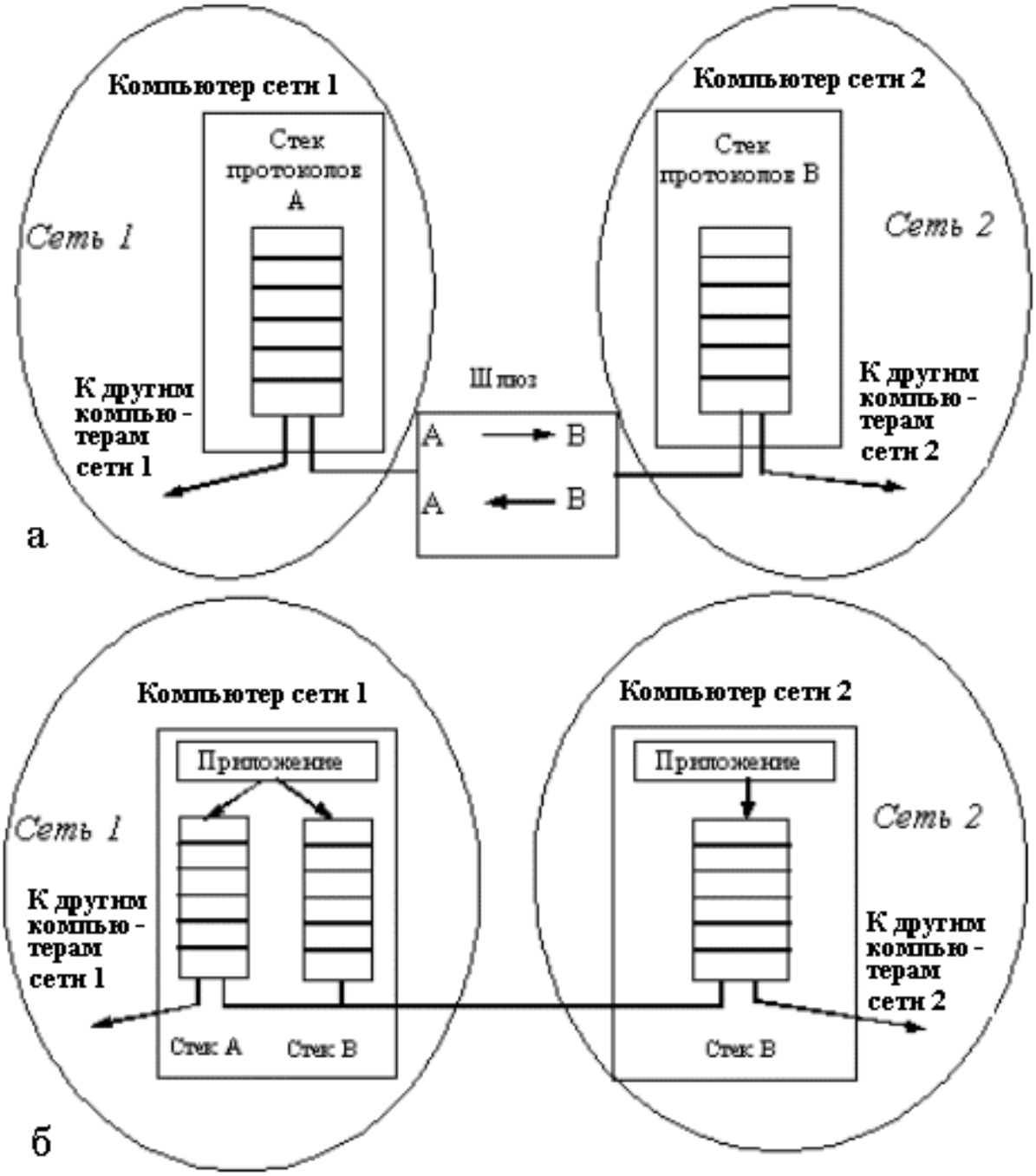



Рис. 4.1. Два основных варианта согласования протоколов: *а* - трансляция протоколов; *б* - мультиплексирование стеков протоколов

Рис. 4.2 иллюстрирует принцип функционирования шлюза. В показанном примере шлюз, размещенный на компьютере 2, согласовывает протоколы клиентского компьютера 1 сети А с протоколами серверного компьютера 3 сети В. Допустим, что две сети используют полностью отличающиеся стеки протоколов. Как видно из рис. 4.2, в шлюзе реализованы оба стека протоколов.

Запрос от прикладного процесса клиентского компьютера сети А поступает на прикладной уровень его стека протоколов. В соответствии с

этим протоколом на прикладном уровне формируются соответствующий пакет (или несколько пакетов), в которых передается запрос на выполнение сервиса некоторому серверу сети В. Пакет прикладного уровня передается вниз по стеку компьютера сети А, а затем в соответствии с протоколами канального и физического уровней сети А поступает в компьютер 2, то есть в шлюз.

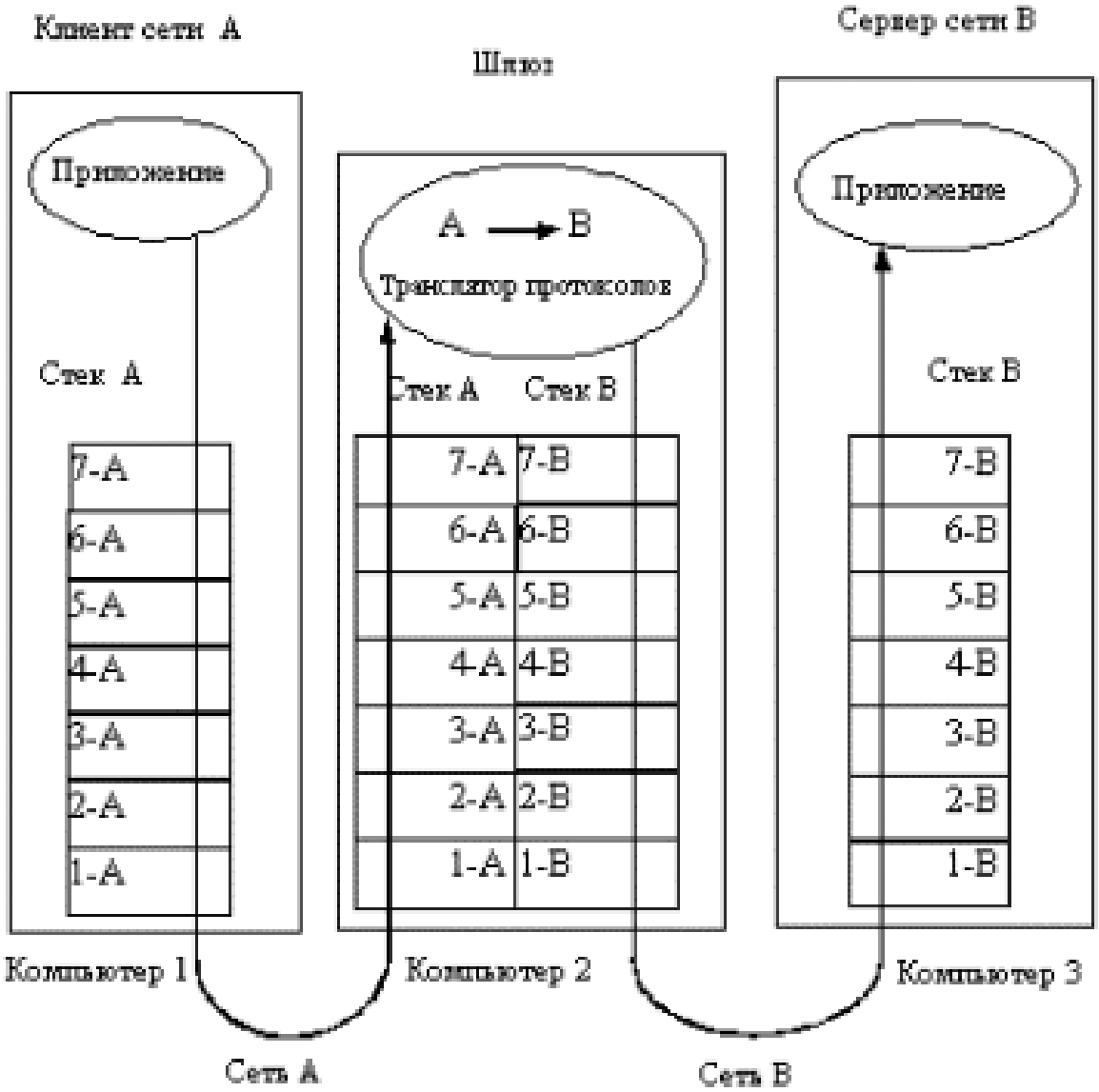



Рис. 4.2. Принципы функционирования шлюза

Здесь он передается от самого нижнего к самому верхнему уровню стека протоколов сети А. Затем пакет прикладного уровня стека сети А преобразуется (транслируется) в пакет прикладного уровня серверного стека сети В. Алгоритм преобразования пакетов зависит от конкретных протоколов и, как уже было сказано, может быть достаточно сложным. В качестве общей информации, позволяющей корректно провести трансляцию, может использоваться, например, информация о символьном имени сервера и символьном имени запрашиваемого ресурса сервера (в частно-

сти, это может быть имя каталога файловой системы). Преобразованный пакет от верхнего уровня стека сети В передается к нижним уровням в соответствии с правилами этого стека, а затем по физическим линиям связи в соответствии с протоколами физического и канального уровней сети В поступает в другую сеть к нужному серверу. Ответ сервера преобразуется шлюзом аналогично.

### 4.5.2. Мультиплексирование стеков протоколов

Вторым использующимся в настоящее время на практике подходом является использование в рабочих станциях технологии мультиплексирования различных стеков протоколов.

При мультиплексировании стеков протоколов на один из двух взаимодействующих компьютеров с различными стеками протоколов помещается коммуникационный стек другого компьютера. На рис. 4.3 приведен пример взаимодействия клиентского компьютера сети 1 с сервером своей сети и сервером сети 2, работающей со стеком протоколов, полностью отличающимся от стека сети 1. В клиентском компьютере реализованы оба стека. Для того, чтобы запрос от прикладного процесса был правильно обработан и направлен через соответствующий стек, в компьютер необходимо добавить специальный программный элемент - мультиплексор протоколов. Мультиплексор должен уметь определять, к какой сети направляется запрос клиента. Для этого может использоваться служба имен сети, в которой отмечается принадлежность того или иного ресурса определенной сети с соответствующим стеком протоколов.

При использовании технологии мультиплексирования структура коммуникационных средств операционной системы может быть и более сложной. В общем случае на каждом уровне вместо одного протокола появляется целый набор протоколов, а мультиплексоров может быть несколько, выполняющих коммутацию между протоколами разных уровней (рис. 4.4). Например, рабочая станция может получить доступ к сетям с протоколами NetBIOS, IP, IPX через один сетевой адаптер. Аналогично, сервер, поддерживающий прикладные протоколы NCP, SMB и NFS может без проблем выполнять запросы рабочих станций сетей NetWare, Windows NT и Sun одновременно.

Предпосылкой для развития технологии мультиплексирования стеков протоколов стало строгое определения протоколов и интерфейсов различных уровней и их открытое описание, так, чтобы фирма при реализации "чужого" протокола или интерфейса могла быть уверена, что ее продукт будет правильно взаимодействовать с продуктами других фирм по данному протоколу.

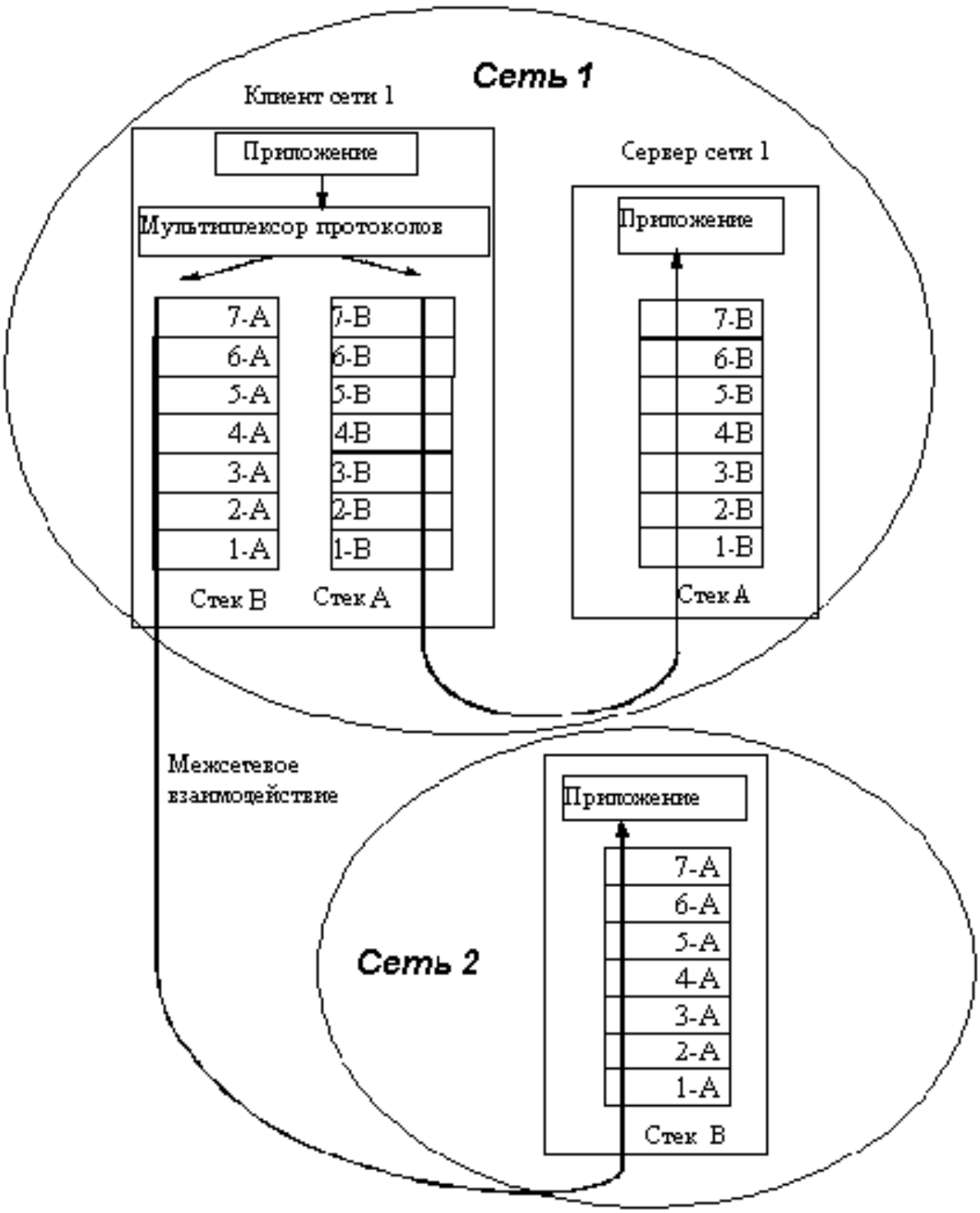



Рис. 4.3. Мультиплексирование стеков

### 4.5.3. Использование магистрального протокола

Хорошим решением был бы переход на единый стек протоколов, но вряд ли эта перспектива осуществится в ближайшем будущем. Попытка введения единого стека коммуникационных протоколов сделана в 1990 году правительством США, которое обнародовало программу GOSIP - Government OSI Profile, в соответствии с которой стек протоколов OSI

должен стать общим знаменателем для всех сетей, устанавливаемых в правительственных организациях США. Но, понимая бесполезность силовых мер, программа GOSIP не ставит задачу немедленного перехода на стек OSI, а принуждает пока к использованию этого стека в качестве "второго языка" правительственных сетей, наряду с родным, первым.

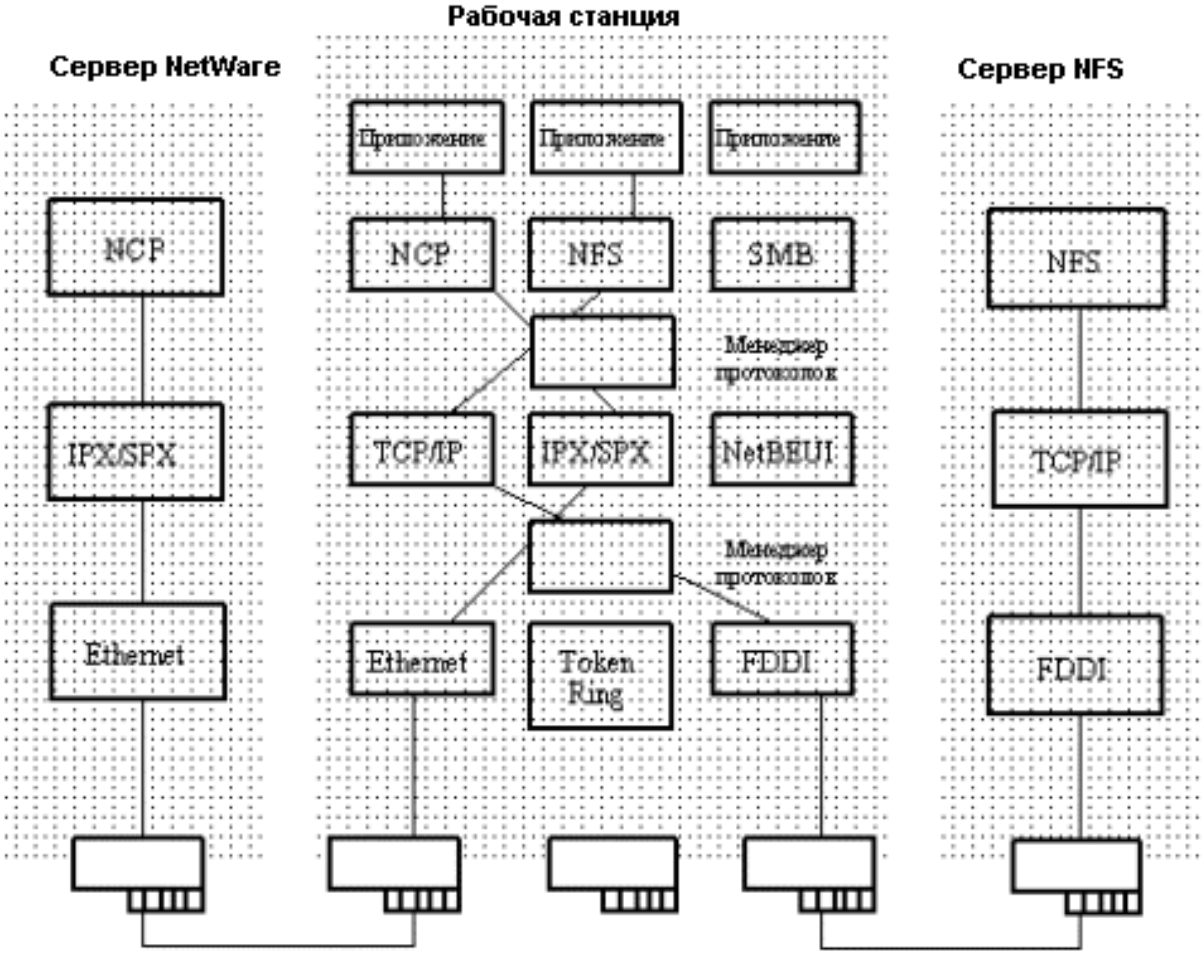



Рис. 4.4. Мультиплексирование протоколов

### 4.5.4. Вопросы реализации

При объединении сетей различных типов в общем случае необходимо обеспечить двухстороннее взаимодействие сетей, то есть решить две задачи (рис. 4.5):

1. Обеспечение доступа клиентам сети А к ресурсам и сервисам серверов сети В.

2. Обеспечение доступа клиентам сети В к ресурсам и сервисам сети А.

Эти задачи независимы и их можно решать отдельно. Прежде всего нужно понять, необходимо ли полное решение или достаточно и частичного, то есть нужно ли, чтобы пользователи, например, UNIX-машин имели доступ к ресурсам серверов сети NetWare, а пользователи персональных

машин имели доступ к ресурсам UNIX-хостов, или же достаточно обеспечить доступ к ресурсам другой сети только одному виду пользователей.

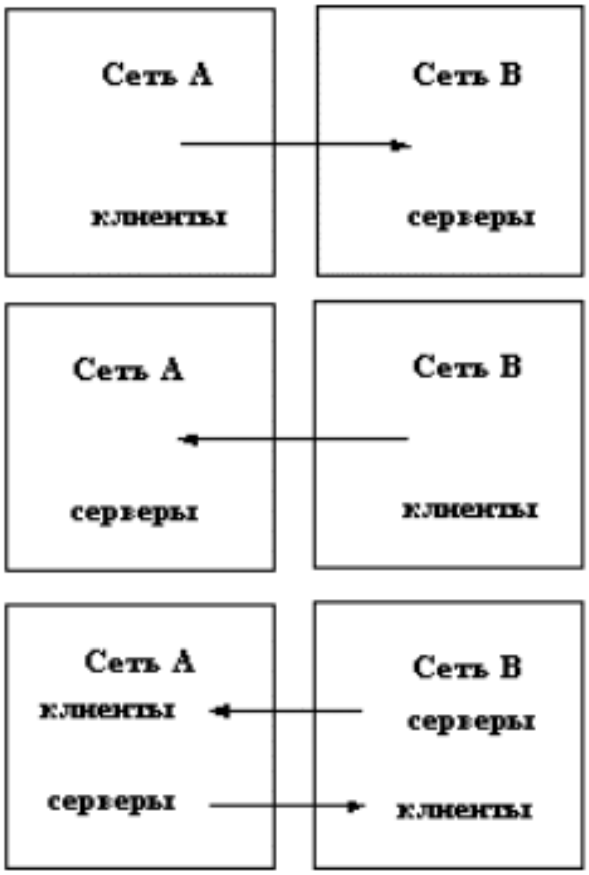



Рис. 4.5. Варианты сетевого взаимодействия

Кроме того, каждую из этих задач можно в свою очередь разделить на части. В сети обычно имеются различные виды разделяемых ресурсов, и с каждым типом ресурсов могут предоставляться различные виды сервиса. Например, в UNIX-сетях файлы являются разделяемым ресурсом, и с ними связаны два вида сервиса - перемещение файлов между машинами по протоколу FTP и монтирование удаленной файловой системы по протоколу NFS. Поэтому при объединении сетей можно предложить пользователям набор средств, каждое из которых позволяет воспользоваться одним каким-либо сервисом чужой сети. Естественно, возможно объединение всех функций в рамках одного продукта. При объединении сетей достаточно иметь средства взаимодействия сетей только в одной из сетей.

В то время, как расположение программных средств, реализующих шлюз, уже было определено - они должны располагаться на компьютере, занимающем промежуточное положение между двумя взаимодействующими машинами, вопрос о размещении дополнительных стеков протоколов остался открытым. Заметим также, что шлюз реализует взаимодействие "многие-ко-многим" (все клиенты могут обращаться ко всем серверам).

Рассмотрим все возможные варианты размещения программных средств, реализующих взаимодействие двух сетей, которые основаны на мультиплексировании протоколов. Введем некоторые обозначения: С - сервер, К - клиент, Δ - дополнительный протокол или стек протоколов.

На рис. 4.6 показаны оба возможных варианта *однонаправленного* взаимодействия АΔВ: а) путем добавления нового стека к клиентам сети А, либо б) путем присоединения "добавки" к серверам сети В.

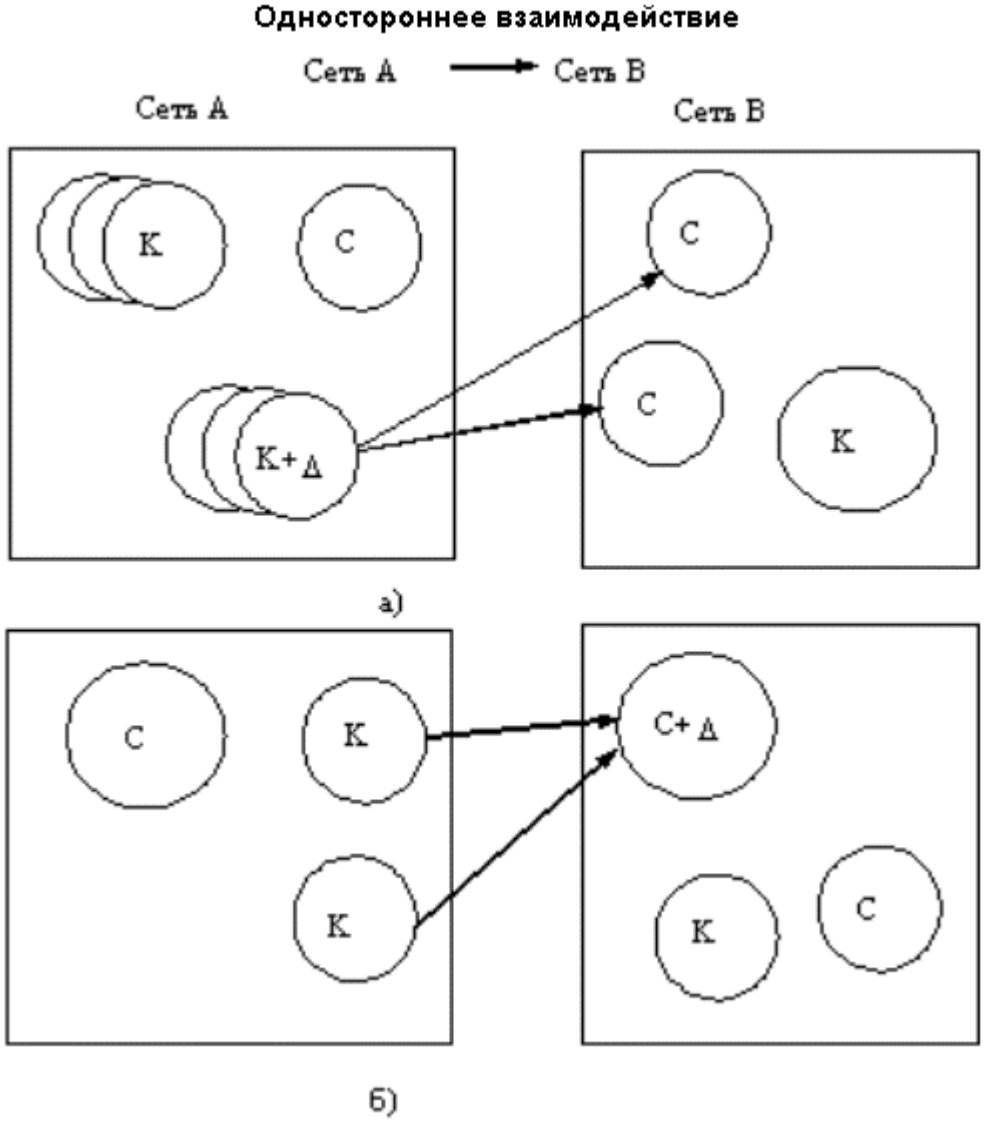



Рис. 4.6. Варианты размещения программных средств (С - сервер, К - клиент, Δ - средства сетевого взаимодействия)

В первом случае, когда средства мультиплексирования располагаются на клиентских частях, только клиенты, снабженные средствами мультиплексирования протоколов, могут обращаться к серверам сети В, при этом они могут обращаться ко всем серверам сети В. Во втором случае, когда набор стеков расположен на каком-либо сервере сети В, данный сервер может обслуживать всех клиентов сети А. Очевидно, что серверы сети В

без средств мультиплексирования не могут быть использованы клиентами сети А. Взаимодействие А Δ В реализуется симметрично.

Если же требуется реализовать взаимодействие *в обе стороны* одновременно, то для этого существует четыре возможных варианта, показанных на рис. 4.7. Каждый вариант имеет свои особенности с точки зрения возможностей связи клиентов с серверами:

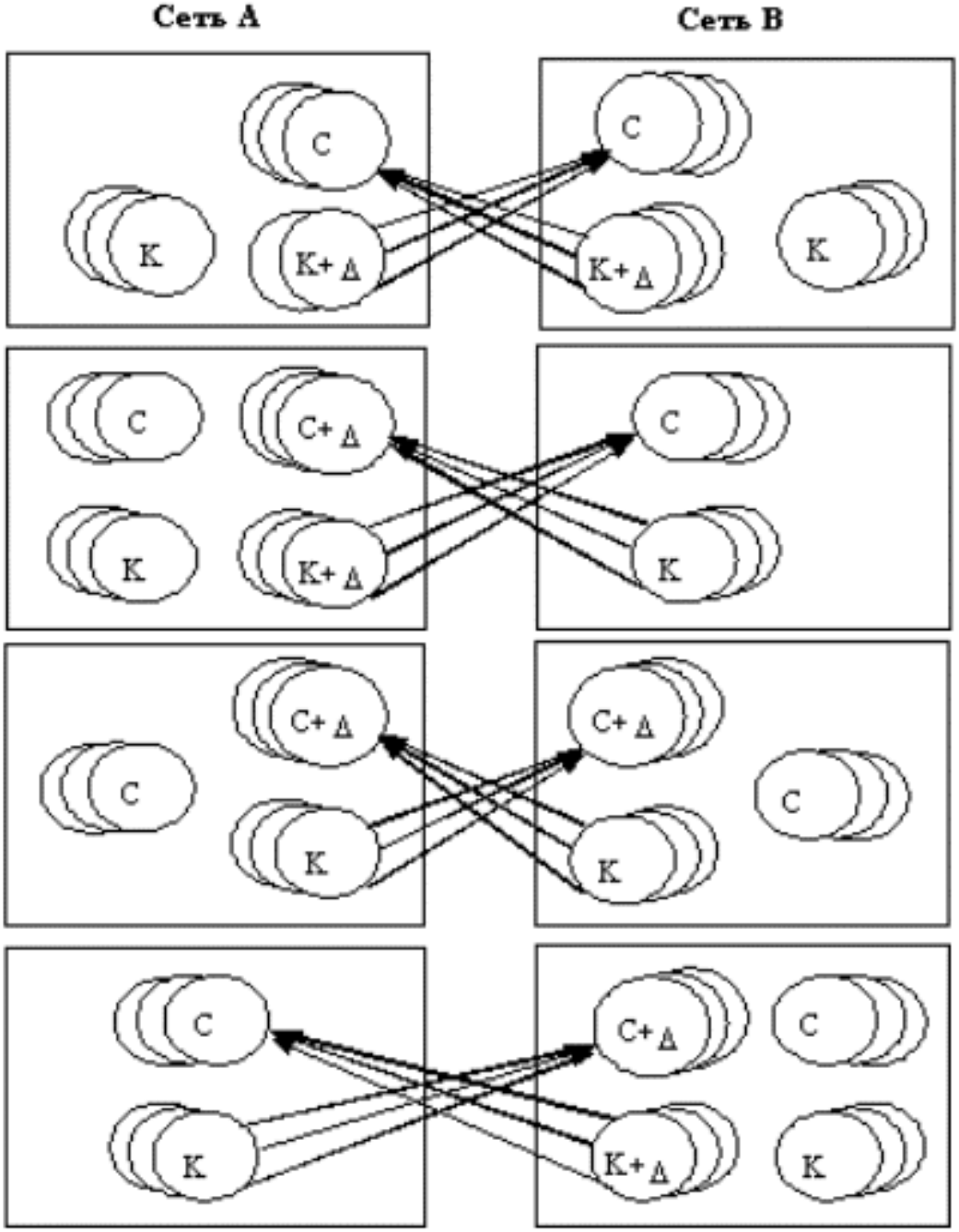



Рис. 4.7. Варианты размещения программных средств при двустороннем взаимодействии (С - сервер, К - клиент, Δ - средства сетевого взаимодействия)

1. Средства обеспечения взаимодействия расположены только на клиентских частях обеих сетей. Для тех и только тех клиентов обеих сетей, которые оснащены "добавками", гарантируется возможность связи со всеми серверами из "чужой" сети.

2. Все средства обеспечения взаимодействия расположены на стороне сети А. Все клиенты сети В могут обращаться к серверам сети А (не ко всем, а только к тем, которые имеют сетевую "добавку"). Часть клиентов сети А, которые обозначены как К+Δ, могут обращаться ко всем серверам сети В.

3. Средства межсетевого взаимодействия расположены только на серверных частях обеих сетей. Всем клиентам обеих сетей гарантируется возможность работы с серверами "чужих" сетей, но не со всеми, а только с серверами, обладающими сетевыми средствами мультиплексирования протоколов.

4. Все средства межсетевого взаимодействия расположены на стороне В. Двусторонний характер взаимодействия обеспечивается модификацией и клиентских, и серверных частей сети В. Все клиенты сети А могут обращаться за сервисом к серверам сети В, обозначенным как С+Δ, а все серверы сети А могут обслуживать клиентов сети В, обозначенных как К+Δ.

Очевидно, что наличие программных продуктов для каждого из рассмотренных вариантов сильно зависит от конкретной пары операционных систем. Для некоторых пар может вовсе не найтись продуктов межсетевого взаимодействия, а для некоторых можно выбирать из нескольких вариантов.

Рассмотрим в качестве примера набор программных продуктов, реализующих взаимодействие Windows NT и NetWare. В ОС Windows NT и в серверной части (Windows NT Server), и в клиентских частях (Windows NT Workstation) предусмотрены встроенные средства мультиплексирования нескольких протоколов, в том числе и стека IPX/SPX. Следовательно эта операционная система может поддерживать двустороннее взаимодействие (по варианту 2) с NetWare без каких-либо дополнительных программных средств. Аналогичным образом реализуется взаимодействие сетей Windows NT с UNIX-сетями.

### 4.5.5. Сравнение вариантов организации взаимодействия сетей

Возвращаясь к принципам организации взаимодействия сетей, сравним два основных подхода - мультиплексирование протоколов и трансляцию протоколов (шлюзы).

Встроенные в сетевую ОС средства мультиплексирования протоколов дают все те преимущества, которые присущи встроенным средствам:

- Эти средства не нужно отдельно приобретать;

- Нет проблем их совместимости с другими продуктами.

Основным недостатком этого подхода является *избыточность*. Хотя средства мультиплексирования обычно позволяют загружать и выгружать по желанию пользователя различные стеки протоколов, но если нужно одновременно работать с тремя различными сетями, то в каждую рабочую станцию необходимо загрузить все три стека одновременно.

Шлюз по своей природе является *выделенным* сервисом, разделяемым всеми источниками запросов к серверам другой сети. Использование шлюзов обеспечивает следующие преимущества:

- Позволяет сосредоточить все функции согласования протоколов в одном месте и разгрузить рабочие станции от дополнительного программного обеспечения, а их пользователей - от необходимости его генерации. Шлюз сохраняет в локальной сети ее родную среду протоколов, что повышает производительность, так как стек протоколов был специально спроектирован для данной операционной среды и наилучшим образом учитывает ее особенности.
- Возникающие проблемы легко локализуются.
- Обслуживающий персонал работает в привычной среде, где можно использовать имеющийся опыт по поддержанию сети. Шлюзы сохраняют различные, несовместимые сети в их первозданном виде. Если имеется несколько различных сетей, то для их совместной работы может понадобиться значительное количество шлюзов. Для доступа пользователей сети UNIX к мейнфрейму понадобится шлюз UNIX-SNA, для подключения пользователей NetWare к компьютерам UNIX и мейнфрейму нужно два шлюза - NetWare-UNIX и NetWare-SNA.

Недостатки использования шлюзов:

- Шлюзы работают, как правило, медленно; пользователи замечают уменьшение производительности при обращении к другой сети через шлюз.
- Шлюз как централизованное средство понижает надежность сети.

### *4.6. Современные концепции и технологии проектирования операционных систем*

Операционная система является сердцевиной сетевого программного обеспечения, она создает среду для выполнения приложений и во многом определяет, какими полезными для пользователя свойствами эти приложения будут обладать. В связи с этим рассмотрим требования, которым должна удовлетворять современная ОС.

### 4.6.1. Требования, предъявляемые к ОС

Очевидно, что главным требованием, предъявляемым к операционной системе, является способность выполнения основных функций: эффективного управления ресурсами и обеспечения удобного интерфейса для пользователя и прикладных программ. Современная ОС, как правило, должна реализовывать мультипрограммную обработку, виртуальную память, свопинг, поддерживать многооконный интерфейс, а также выполнять многие другие, совершенно необходимые функции. Кроме этих функциональных требований к операционным системам предъявляются не менее важные рыночные требования. К этим требованиям относятся:

- *Расширяемость*. Код должен быть написан таким образом, чтобы можно было легко внести дополнения и изменения, если это потребуется, и не нарушить целостность системы.
- *Переносимость*. Код должен легко переноситься с процессора одного типа на процессор другого типа и с аппаратной платформы (которая включает наряду с типом процессора и способ организации всей аппаратуры компьютера) одного типа на аппаратную платформу другого типа.
- *Надежность и отказоустойчивость*. Система должна быть защищена как от внутренних, так и от внешних ошибок, сбоев и отказов. Ее действия должны быть всегда предсказуемыми, а приложения не должны быть в состоянии наносить вред ОС.
- *Совместимость*. ОС должна иметь средства для выполнения прикладных программ, написанных для других операционных систем. Кроме того, пользовательский интерфейс должен быть совместим с существующими системами и стандартами.
- *Безопасность*. ОС должна обладать средствами защиты ресурсов одних пользователей от других.
- *Производительность*. Система должна обладать настолько хорошим быстродействием и временем реакции, насколько это позволяет аппаратная платформа.

Рассмотрим более подробно некоторые из этих требований.

### 4.6.2. Расширяемость

В то время, как аппаратная часть компьютера устаревает за несколько лет, полезная жизнь операционных систем может измеряться десятилетиями. Примером может служить линия ОС UNIX. Поэтому операционные системы всегда эволюционно изменяются со временем, и эти изменения более значимы, чем изменения аппаратных средств. Изменения ОС обычно представляют собой приобретение ею новых свойств. Например, поддержка новых устройств, возможность связи с сетями нового типа, поддержка многообещающих технологий, таких как графический интерфейс пользователя или объектно-ориентированное программное окружение, использо-

вание более чем одного процессора. Сохранение целостности кода, какие бы изменения не вносились в операционную систему, является главной целью разработки.

Расширяемость может достигаться за счет модульной структуры ОС, при которой программы строятся из набора отдельных модулей, взаимодействующих только через функциональный интерфейс. Новые компоненты могут быть добавлены в операционную систему модульным путем, они выполняют свою работу, используя интерфейсы, поддерживаемые существующими компонентами.

Использование объектов для представления системных ресурсов также улучшает расширяемость системы. Объекты - это абстрактные типы данных, над которыми можно производить только те действия, которые предусмотрены специальным набором объектных функций. Объекты позволяют единообразно управлять системными ресурсами. Добавление новых объектов не разрушает существующие объекты и не требует изменений существующего кода.

Прекрасные возможности для расширения предоставляет подход к структурированию ОС по типу клиент-сервер с использованием микроядерной технологии. В соответствии с этим подходом ОС строится как совокупность привилегированной управляющей программы и набора непривилегированных услуг-серверов. Основная часть ОС может оставаться неизменной в то время, как могут быть добавлены новые серверы или улучшены старые.

Средства вызова удаленных процедур (RPC) также дают возможность расширить функциональные возможности ОС. Новые программные процедуры могут быть добавлены в любую машину сети и немедленно поступить в распоряжение прикладных программ на других машинах сети.

Некоторые ОС для улучшения расширяемости поддерживают загружаемые драйверы, которые могут быть добавлены в систему во время ее работы. Новые файловые системы, устройства и сети могут поддерживаться путем написания драйвера устройства, драйвера файловой системы или транспортного драйвера и загрузки его в систему.

### 4.6.3. Переносимость

Требование переносимости кода тесно связано с расширяемостью. Расширяемость позволяет улучшать операционную систему, в то время как переносимость дает возможность перемещать всю систему на машину, базирующуюся на другом процессоре или аппаратной платформе, делая при этом по возможности небольшие изменения в коде. Хотя ОС часто описываются либо как переносимые, либо как непереносимые, переносимость - это не бинарное состояние. Вопрос не в том, может ли быть система перенесена, а в том, насколько легко можно это сделать. Написание переноси-

мой ОС аналогично написанию любого переносимого кода - нужно следовать некоторым правилам.

Во-первых, большая часть кода должна быть написана на языке, который имеется на всех машинах, куда вы хотите переносить систему. Обычно это означает, что код должен быть написан на языке высокого уровня, предпочтительно стандартизованном, например, на языке С. Программа, написанная на ассемблере, не является переносимой, если только вы не собираетесь переносить ее на машину, обладающую командной совместимостью с вашей.

Во-вторых, следует учесть, в какое физическое окружение программа должна быть перенесена. Различная аппаратура требует различных решений при создании ОС. Например, ОС, построенная на 64-битовых адресах, не может быть перенесена на машину с 32-битовыми адресами (разве что с огромными трудностями).

В-третьих, важно минимизировать или, если возможно, исключить те части кода, которые непосредственно взаимодействуют с аппаратными средствами. Зависимость от аппаратуры может иметь много форм. Некоторые очевидные формы зависимости включают прямое манипулирование регистрами и другими аппаратными средствами.

В-четвертых, если аппаратно зависимый код не может быть полностью исключен, то он должен быть изолирован в нескольких хорошо локализуемых модулях. Аппаратно-зависимый код не должен быть распределен по всей системе. Например, можно спрятать аппаратно-зависимую структуру в программно-задаваемые данные абстрактного типа. Другие модули системы будут работать с этими данными, а не с аппаратурой, используя набор некоторых функций. Когда ОС переносится, то изменяются только эти данные и функции, которые ими манипулируют.

Для легкого переноса ОС при ее разработке должны быть соблюдены следующие требования:

- *Переносимый язык высокого уровня*. Большинство переносимых ОС написано на языке С (стандарт ANSI X3.159-1989). Разработчики выбирают С потому, что он стандартизован, и потому, что С-компиляторы широко доступны. Ассемблер используется только для тех частей системы, которые должны непосредственно взаимодействовать с аппаратурой (например, обработчик прерываний) или для частей, которые требуют максимальной скорости (например, целочисленная арифметика повышенной точности). Однако непереносимый код должен быть тщательно изолирован внутри тех компонентов, где он используется.
- *Изоляция процессора*. Некоторые низкоуровневые части ОС должны иметь доступ к процессорно-зависимым структурам данных и регистрам. Однако код, который делает это, должен содержаться в небольших модулях, которые могут быть заменены аналогичными модулями для других процессоров.

- *Изоляция платформы*. Зависимость от платформы заключается в различиях между рабочими станциями разных производителей, построенными на одном и том же процессоре (например, MIPS R4000). Должен быть введен программный уровень, абстрагирующий аппаратуру (кэши, контроллеры прерываний ввода-вывода и т. п.) вместе со слоем низкоуровневых программ таким образом, чтобы высокоуровневый код не нуждался в изменении при переносе с одной платформы на другую.

### 4.6.4. Совместимость

Одним из аспектов совместимости является способность ОС выполнять программы, написанные для других ОС или для более ранних версий данной операционной системы, а также для другой аппаратной платформы.

Необходимо разделять вопросы двоичной совместимости и совместимости на уровне исходных текстов приложений. Двоичная совместимость достигается в том случае, когда можно взять исполняемую программу и запустить ее на выполнение на другой ОС. Для этого необходимы: совместимость на уровне команд процессора, совместимость на уровне системных вызовов и даже на уровне библиотечных вызовов, если они являются динамически связываемыми.

Совместимость на уровне исходных текстов требует наличия соответствующего компилятора в составе программного обеспечения, а также совместимости на уровне библиотек и системных вызовов. При этом необходима перекомпиляция имеющихся исходных текстов в новый выполняемый модуль.

Совместимость на уровне исходных текстов важна в основном для разработчиков приложений, в распоряжении которых эти исходные тексты всегда имеются. Но для конечных пользователей практическое значение имеет только двоичная совместимость, так как только в этом случае они могут использовать один и тот же коммерческий продукт, поставляемый в виде двоичного исполняемого кода, в различных операционных средах и на различных машинах.

Обладает ли новая ОС двоичной совместимостью или совместимостью исходных текстов с существующими системами, зависит от многих факторов. Самый главный из них - архитектура процессора, на котором работает новая ОС. Если процессор, на который переносится ОС, использует тот же набор команд (возможно с некоторыми добавлениями) и тот же диапазон адресов, тогда двоичная совместимость может быть достигнута достаточно просто.

Гораздо сложнее достичь двоичной совместимости между процессорами, основанными на разных архитектурах. Для того, чтобы один компьютер выполнял программы другого (например, DOS-программу на Mac),

этот компьютер должен работать с машинными командами, которые ему изначально непонятны. Например, процессор типа 680x0 на Mac должен исполнять двоичный код, предназначенный для процессора 80x86 в PC. Процессор 80x86 имеет свои собственные дешифратор команд, регистры и внутреннюю архитектуру. Процессор 680x0 не понимает двоичный код 80x86, поэтому он должен выбрать каждую команду, декодировать ее, чтобы определить, для чего она предназначена, а затем выполнить эквивалентную подпрограмму, написанную для 680x0. Так как к тому же у 680x0 нет в точности таких же регистров, флагов и внутреннего арифметико-логического устройства, как в 80x86, он должен имитировать все эти элементы с использованием своих регистров или памяти. И он должен тщательно воспроизводить результаты каждой команды, что требует специально написанных подпрограмм для 680x0, гарантирующих, что состояние эмулируемых регистров и флагов после выполнения каждой команды будет в точности таким же, как и на реальном 80x86.

Это простая, но очень медленная работа, так как микрокод внутри процессора 80x86 исполняется на значительно более быстродействующем уровне, чем эмулирующие его внешние команды 680x0. За время выполнения одной команды 80x86 на 680x0, реальный 80x86 может выполнить десятки команд. Следовательно, если процессор, производящий эмуляцию, не настолько быстр, чтобы компенсировать все потери при эмуляции, то программы, исполняющиеся под эмуляцией, будут очень медленными.

Выходом в таких случаях является использование так называемых прикладных сред. Учитывая, что основную часть программы, как правило, составляют вызовы библиотечных функций, прикладная среда имитирует библиотечные функции целиком, используя заранее написанную библиотеку функций аналогичного назначения, а остальные команды эмулирует каждую по отдельности.

Соответствие стандартам POSIX также является средством обеспечения совместимости программных и пользовательских интерфейсов. Во второй половине 80-х правительственные агентства США начали разрабатывать POSIX как стандарты на поставляемое оборудование при заключении правительственных контрактов в компьютерной области. POSIX - это "интерфейс переносимой ОС, базирующейся на UNIX". POSIX - собрание международных стандартов интерфейсов ОС в стиле UNIX. Использование стандарта POSIX (IEEE стандарт 1003.1 - 1988) позволяет создавать программы стиле UNIX, которые могут легко переноситься из одной системы в другую.

### 4.6.5. Безопасность

В дополнение к стандарту POSIX правительство США также определило требования компьютерной безопасности для приложений, исполь-

зуемых правительством. Многие из этих требований являются желаемыми свойствами для любой многопользовательской системы. Правила безопасности определяют такие свойства, как защита ресурсов одного пользователя от других и установление квот по ресурсам для предотвращения захвата одним пользователем всех системных ресурсов ( таких как память).

Обеспечение защиты информации от несанкционированного доступа является обязательной функцией сетевых операционных систем. В большинстве популярных систем гарантируется степень безопасности данных, соответствующая уровню С2 в системе стандартов США.

Основы стандартов в области безопасности были заложены "*Критериями оценки надежных компьютерных систем*". Этот документ, изданный в США в 1983 году национальным центром компьютерной безопасности (NCSC - National Computer Security Center), часто называют Оранжевой Книгой.

В соответствии с требованиями Оранжевой книги безопасной считается такая система, которая "посредством специальных механизмов защиты контролирует доступ к информации таким образом, что только имеющие соответствующие полномочия лица или процессы, выполняющиеся от их имени, могут получить доступ на чтение, запись, создание или удаление информации".

Иерархия уровней безопасности, приведенная в Оранжевой Книге, помечает низший уровень безопасности как D, а высший - как А.

- В класс D попадают системы, оценка которых выявила их несоответствие требованиям всех других классов.
- Основными свойствами, характерными для С-систем, являются: наличие подсистемы учета событий, связанных с безопасностью, и избирательный контроль доступа. Уровень С делится на 2 подуровня: уровень С1, обеспечивающий защиту данных от ошибок пользователей, но не от действий злоумышленников, и более строгий уровень С2. На уровне С2 должны присутствовать *средства секретного входа,* обеспечивающие идентификацию пользователей путем ввода уникального имени и пароля перед тем, как им будет разрешен доступ к системе. *Избирательный контроль доступа*, требуемый на этом уровне позволяет владельцу ресурса определить, кто имеет доступ к ресурсу и что он может с ним делать. Владелец делает это путем предоставляемых прав доступа пользователю или группе пользователей. *Средства учета и наблюдения* (*auditing*) - обеспечивают возможность обнаружить и зафиксировать важные события, связанные с безопасностью, или любые попытки создать, получить доступ или удалить системные ресурсы. *Защита памяти* - заключается в том, что память инициализируется перед тем, как повторно используется. На этом уровне система не защищена от ошибок пользователя, но поведение его может быть проконтролировано по записям в журнале, оставленным средствами наблюдения и аудитинга.

- Системы уровня B основаны на помеченных данных и распределении пользователей по категориям, то есть реализуют *мандатный контроль доступа*. Каждому пользователю присваивается рейтинг защиты, и он может получать доступ к данным только в соответствии с этим рейтингом. Этот уровень в отличие от уровня C защищает систему от ошибочного поведения пользователя.
- Уровень A является самым высоким уровнем безопасности, он требует в дополнение ко всем требованиям уровня B выполнения формального, математически обоснованного доказательства соответствия системы требованиям безопасности.

Различные коммерческие структуры (например, банки) особо выделяют необходимость учетной службы, аналогичной той, что предлагают государственные рекомендации C2. Любая деятельность, связанная с безопасностью, может быть отслежена и тем самым учтена. Это как раз то, что требует C2 и то, что обычно нужно банкам. Однако, коммерческие пользователи, как правило, не хотят расплачиваться производительностью за повышенный уровень безопасности. A-уровень безопасности занимает своими управляющими механизмами до 90% процессорного времени. Более безопасные системы не только снижают эффективность, но и существенно ограничивают число доступных прикладных пакетов, которые соответствующим образом могут выполняться в подобной системе. Например для ОС Solaris (версия UNIX) есть несколько тысяч приложений, а для ее аналога B-уровня - только сотня.

### *4.7. Построение сетевых баз данных: одно- и многопользовательские решения*

Потребности в хранении и обработке больших объемов информации привели к появлению систем управления базами данных (СУБД), которые настолько прочно вошли в нашу жизнь, что без них уже немыслима работа современного предприятия или организации.

Стремительное развитие этой области знаний, наличие на рынке большого количества самых разнообразных систем и технологий делают зачастую очень сложным вопрос выбора правильного подхода к реализации конкретной задачи. В данной статье весь спектр существующих на сегодняшний день СУБД рассматривается сквозь призму их применимости к решению различных классов задач.

При построении БД необходимо сразу решить, в каком режиме будет осуществляться доступ к данным:

- в однопользовательском режиме, когда возможна одновременная работа с данными только одного пользователя (персональная БД);
- в многопользовательском режиме, когда с БД одновременно могут работать несколько пользователей (многопользовательская БД).

### 4.7.1. Персональные базы данных

Для класса персональных БД существует большое количество СУБД, работающих на ПК под управлением WINDOWS. Системы, работающие под WINDOWS, очень быстро завоевали популярность благодаря красивому и удобному интерфейсу, простоте формирования форм и отчетов и возможности включения в них графиков, а также способности хранить изображения в виде полей записей.

В зависимости от набора предоставляемых возможностей и требований к квалификации разработчика эти СУБД можно разбить на две категории:

- системы, доступные для непрограммистов;
- системы, требующие профессиональных знаний в области программирования и обладающие расширенным набором возможностей.

### 4.7.2. Многопользовательские базы данных

*Многотерминальная система*

Суть этого подхода заключается в том, что к одной машине (серверу) подключается большое количество терминалов (по числу рабочих мест, рис. 4.8). Сервер осуществляет хранение, обработку и обращение к данным. Терминалы не выполняют никакой работы при подготовке и обработке данных, они служат только для ввода данных и отображения информации. Подобные системы показывают высокие результаты в плане надежности и производительности. К сожалению, использование этого подхода в его классическом виде не очень эффективно при создании территориально распределенных систем. С другой стороны, в последнее время наблюдается постепенный отход от применения алфавитно-цифровых терминалов в пользу использования графических интерфейсов на X-терминалах, что резко повышает стоимость подобных систем.

В последнее время наблюдается существенное уменьшение количества разработок, построенных с использованием этого подхода. Это, в первую очередь, связано с тем, что по параметрам цена/производительность более предпочтительным оказывается вариант использования архитектуры клиент-сервер.

*Многопользовательская база данных на файл-сервере*

При наличии локальной сети персональных компьютеров можно использовать второй подход. Его сущность заключается в том, что база данных размещается на одном из компьютеров, который в этом случае выступает как файл-сервер (рис. 4.9).

При этом манипуляция данными осуществляется следующим образом. База данных (а иногда и код приложения) хранится на сервере, а вся работа по обновлению, хранению, добавлению и отображению информации выполняется в локальной системе.

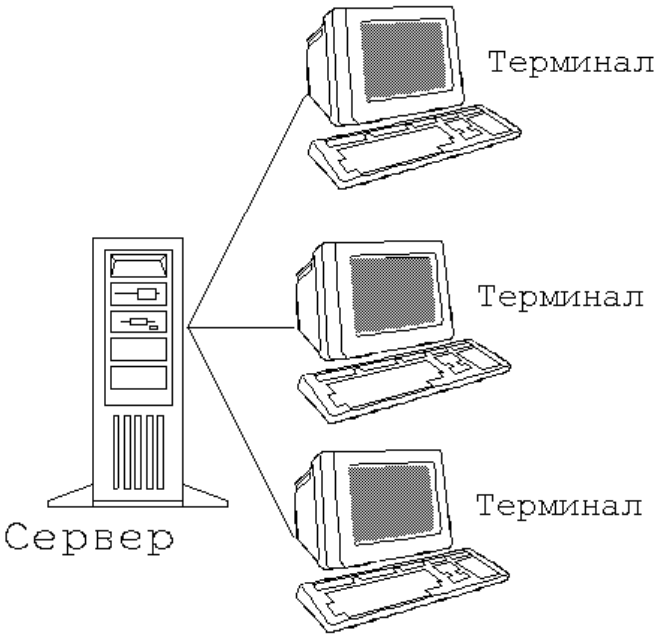



Рис. 4.8. Многотерминальная система

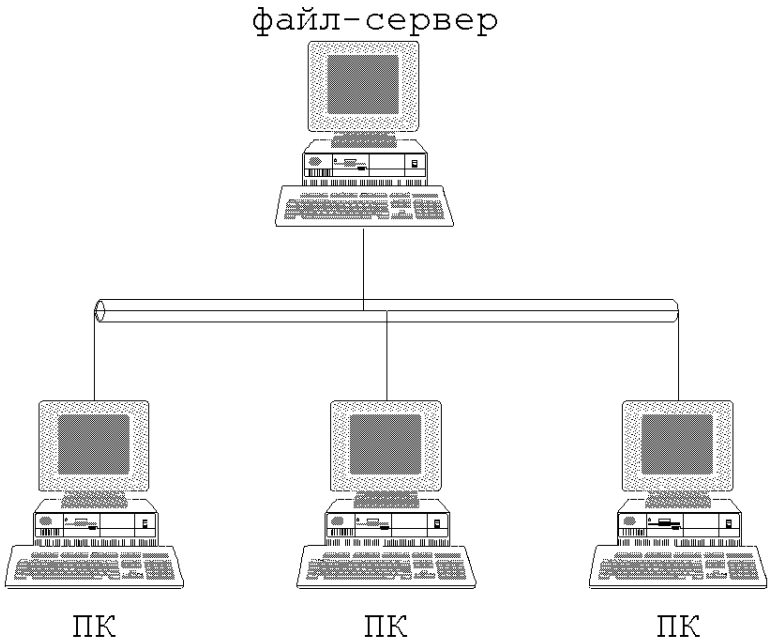



Рис. 4.9. Сеть персональных компьютеров с файл-сервером

Файл-сервер в этом случае выступает в роли удаленного жесткого диска для базы данных и кода приложения. Большинство персональных СУБД поддерживает механизм блокировок для обеспечения целостности данных. Таким образом, эти системы могут быть использованы для построения и работы с подобной многопользовательской базой данных. При этом выбор СУБД определяется уровнем сложности системы и профессиональным уровнем разработчика. Если создаваемая база данных относительно мала, и одновременный доступ к ней будет производить небольшое количество пользователей, то этот подход работает прекрасно. Но производительность такой системы начинает резко падать при увеличении раз-

меров таблиц и количества пользователей, пытающихся одновременно обратиться к базе данных. Особенность этого подхода заключается в том, что при просмотре таблицы ее содержимое полностью передается по сети на локальную машину. Сети имеют сравнительно низкую пропускную способность, так что если один из пользователей начнет выполнять сортировку в большой таблице, то это может сделать работу остальных пользователей в сети невозможной на долгое время. Если же локальная сеть подключена к территориально разнесенной сети (WAN - Wide Area Network) и требуется обеспечить возможность работы с базой данных с удаленного компьютера, то необходимость пересылки больших объемов данных по сетевым мостам и маршрутизаторам через несколько сетевых сегментов еще более обострит проблему.

*Архитектура клиент-сервер*

В системе, построенной на этом подходе, обработка данных разделяется между двумя или более компьютерами. При этом клиентская часть системы (front end) использует ПК для представления данных и манипуляции ими. Сервер (back end) используется для хранения, сортировки, изменения, комбинирования и защиты данных (рис. 4.10). В противоположность предыдущему подходу по сети передаются не таблицы, а выборки, являющиеся результатами выполнения запросов, написанных на языке SQL.

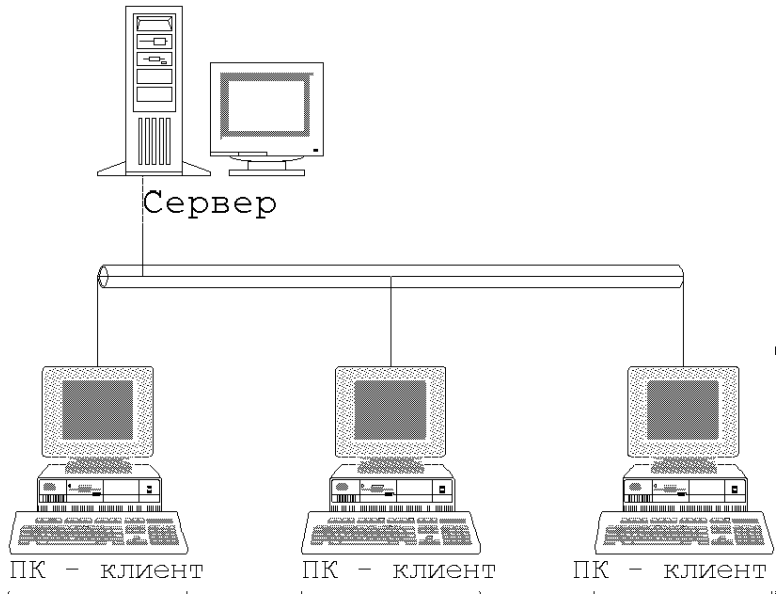



Рис. 4.10. Система с архитектурой клиент-сервер

Совместное использование двух этих компонентов обеспечивает большую гибкость при взаимодействии с данными, чем два предыдущих подхода. Использование архитектуры клиент-сервер позволяет более полно использовать все ресурсы системы (клиентов, сервера и сети). Разделение задач между клиентом и сервером позволяет использовать мощность

всех входящих в систему компьютеров и в то же время пользоваться преимуществами централизованного хранения и возможностью удаленного доступа к данным. В дополнение к этому разработчик получает возможность совмещать в своей системе различные операционные системы, базы данных и клиентские части. Модульная структура системы в архитектуре клиент-сервер облегчает процесс ее модификации: отдельные части системы можно изменять независимо друг от друга.

Можно, к примеру, внести изменения в клиентскую часть или перенести сервер на более мощную машину. Пусть, например, разработчик построил систему типа клиент-сервер с использованием пакета PowerBuilder, работающего на ПК под WINDOWS, для доступа к данным, хранящимся на сервере Oracle, работающем под OS/2. Позднее было решено, что использование RISC-сервера позволит повысить производительность. При этом не потребуется переписывать код приложения или менять операционную систему на ПК-клиенте для того, чтобы Oracle стал работать под UNIX. То же самое приложение можно использовать с базой данных на сервере Oracle, работающем под NetWare NLM. Использование архитектуры клиент-сервер позволяет более гибко подходить к проблеме выбора инструментов для построения клиентских приложений. Например, разработчик может реализовать большую часть своего приложения на Visual Basic, а для создания специальных отчетов использовать какой-то другой инструмент. С другой стороны, архитектура клиент-сервер не лишена недостатков. Возможность совмещения нескольких операционных систем порождает усложнение инфраструктуры разрабатываемой системы, которая может быстро стать технически запутанной. В конце концов разработчик часто сталкивается с необходимостью поддержки нескольких операционных систем - одной для сети, одной или более для сервера базы данных и одной или нескольких для клиентов, что естественным образом повышает сложность управления и администрирования подобной системы.

### 4.7.3. Выбор подхода при построении многопользовательской базы данных

Каждый из перечисленных в предыдущем разделе подходов имеет свои достоинства и недостатки. Сформулируем некоторые рекомендации, не претендующие на роль универсального правила, позволяющего однозначно решить, какой из подходов выбрать.

*Выбор архитектуры клиент-сервер*

Перед переходом на архитектуру клиент-сервер разработчику следует подумать о том, какой тип и размер будет иметь создаваемое приложение, и о типе доступа к информации. Этот вариант следует внимательно рассматривать в случаях, перечисленных ниже.

1. В случае, если размер большинства таблиц в БД может превысить значительную величину, применение архитектуры клиент-сервер может оказаться разумным. Чем больше размер базы данных, тем менее оправдано использование базы данных на файл-сервере, т.к. при этом каждый раз все хранящиеся в таблицах данные будут перемещаться по сети.

2. Технология клиент-сервер является подходящим выбором, если необходимо обеспечить возможность удаленного доступа к данным с использованием средств связи. Передача данных на большое расстояние дорого стоит, и естественно желание разработчиков свести временные затраты к минимуму.

3. Технологию клиент-сервер следует рассматривать в том случае, если разрабатываемая система должна поддерживать работу с группой активных пользователей. Если с системой будет работать более 20-30 пользователей, одновременно обращающихся к БД в любой момент времени, то использование БД на файл-сервере приведет к возникновению проблем с производительностью.

Если требования к системе не превышают вышеуказанных, то разработчику следует оградить себя от сложностей, связанных с использованием технологии клиент-сервер, и использовать персональную СУБД для построения файл-серверной БД.

Перед началом разработки многопользовательской БД необходимо выбрать клиентские и серверные компоненты будущей системы. Несмотря на то, что при проектировании системы в архитектуре клиент-сервер разработчик обладает большей свободой выбора, компоненты таких систем взаимозависимы, так что разумно выбрать все составные части одновременно. При построении системы с самого начала первым делом выбирается аппаратное и программное обеспечение для сервера, а затем проектируется инфраструктура. В последнюю очередь осуществляется подбор инструментов для создания клиентской части.

В качестве программного обеспечения для сервера следует выбирать продукт, обладающий сильной поддержкой со стороны независимых производителей. Это дает разработчику большую свободу при выборе инструментов для создания клиентских приложений и при проектировании инфраструктуры. Также следует убедиться, что БД может работать на разных аппаратных платформах. Informix, Oracle Server, SQL Server и SQL Base обладают этими свойствами. Из перечисленных систем наибольшее распространение в нашей стране получили Oracle и Informix. Обычно Oracle применяется для построения более крупномасштабных БД.

*Построение инфраструктуры системы клиент-сервер*

В данном контексте под инфраструктурой понимается операционная система на сервере, сетевой протокол и аппаратная часть сервера. Ниже

перечислены основные правила, которые следует соблюдать при планировании инфраструктуры.

1. Количество сетевых протоколов должно быть сведено к одному или двум; новые операционные системы не должны использоваться, если в них нет абсолютной уверенности.

2. Компьютер, на котором находится сервер БД, не должен использоваться в качестве файл- и принт-сервера, т.е. не следует подключать к этой машине сетевой принтер и использовать ее диски для хранения файлов коллективного пользования (несмотря на то, что это более просто и экономично). Ни одна из операционных систем не застрахована от сбоя, и использование сервера для большого количества разных задач снижает надежность системы.

3. В качестве операционной системы разработчик должен использовать ОС, с которой он хорошо знаком. В настоящее время идут активные дискуссии по поводу того, какая ОС больше всего подходит для сервера рассматриваемой архитектуры.

4. Следует обратить пристальное внимание на сетевые протоколы, используемые выбираемой СУБД. Несмотря на то, что большинство СУБД использует стандартные протоколы, такие как IPX или TCP/IP, иногда им требуются собственные драйверы. Следует воздерживаться от применения протоколов NetBIOS и NetBEUI, т.к. они используют слишком много памяти на клиентских машинах и недостаточно надежны при работе в больших сетях.

5. В качестве сервера следует выбирать машину, специально спроектированную для выполнения функций сервера, а не просто быстрый ПК. Эта машина должна быть оснащена достаточным количеством оперативной памяти для поддержки кэширования и буферизации исполняемых процессов. Сервер будет содержать самое ценное - данные. При этом производительность является критическим параметром. Так как процесс создания резервной копии базы данных достаточно сложен, можно рассмотреть вариант использования дисковых массивов RAIDs (Redundant Arrays of Inexpensive Disks). Данные будут копироваться на несколько дисков для повышения надежности хранения информации.

*Инструменты для создания клиентской части*

Существует большое количество инструментов для построения клиентских приложений, ориентированных как на профессиональных разработчиков, так и на конечных пользователей. Инструменты разработчика различаются по форме и реализуемым функциям. Это языки программирования, редакторы экранов, CASE-инструменты. Инструменты для конечных пользователей обычно позволяют производить анализ данных, построение отчетов и извлечение данных из БД. Так как каждый из продук-

тов реализует различный набор функций, разработчики часто используют в работе сразу несколько подобных инструментов.

Выбор правильного подхода при построении многопользовательских систем является в высшей степени сложной задачей, требующей профессионального подхода. И если разработка базы данных на файл-сервере под силу одному человеку, то эффективная реализация многопользовательской БД в архитектуре клиент-сервер требует наличия команды разработчиков-профессионалов, имеющих большой практический опыт работы в этой области.

### *4.8. Контрольные вопросы*

1. Дайте определение сетевой операционной системы.
2. Из каких основных компонент состоит сетевая ОС?
3. Перечислите сложившиеся подходы к построению сетевых операционных систем.
4. Дайте определение выделенного сервера.
5. Какие типы выделенных серверов Вам известны?
6. Достоинства и недостатки одноранговых сетей.
7. Эволюция сетей при расширении зоны обслуживания. Проблема интеграции.
8. Признаки корпоративных сетевых ОС.
9. Причины возникновения необходимости в межсетевых взаимодействиях.
10. Функции шлюза.
11. Механизмы мультиплексирования различных стеков протоколов.
12. Что такое персональная БД?
13. Что такое многопользовательская БД?
14. Какие виды многопользовательских баз данных Вы знаете?
15. В чем сущность файл-серверного подхода?
16. Каковы особенности архитектуры клиент-сервер?
17. Какие аспекты необходимо учитывать при построении многопользовательской базы данных?

## 5. Основные сведения о сетях IP

### *5.1. Многоуровневая модель TCP/IP*

*TCP/IP* - собирательное название для набора (стека) сетевых протоколов разных уровней, используемых в Интернет. Особенности TCP/IP:

- открытые стандарты протоколов, разрабатываемые независимо от программного и аппаратного обеспечения;

- независимость от физической среды передачи;
- система уникальной адресации;
- стандартизованные протоколы высокого уровня для пользовательских интерфейсов.

Стек протоколов TCP/IP делится на 4 уровня: *прикладной* (application), *транспортный* (transport), *межсетевой* (internet) *и уровень доступа к среде передачи* (network access). Термины, используемые для обозначения блока передаваемых данных, различны при использовании разных протоколов транспортного уровня: TCP и UDP, поэтому на рис. 5.1 изображено два стека. Как и в модели OSI, данные более верхних уровней инкапсулируются в блоки данных более нижних уровней, например, сегмент или пакет со своими данными и служебными заголовками инкапсулируется внутри поля «Данные» дейтаграммы.

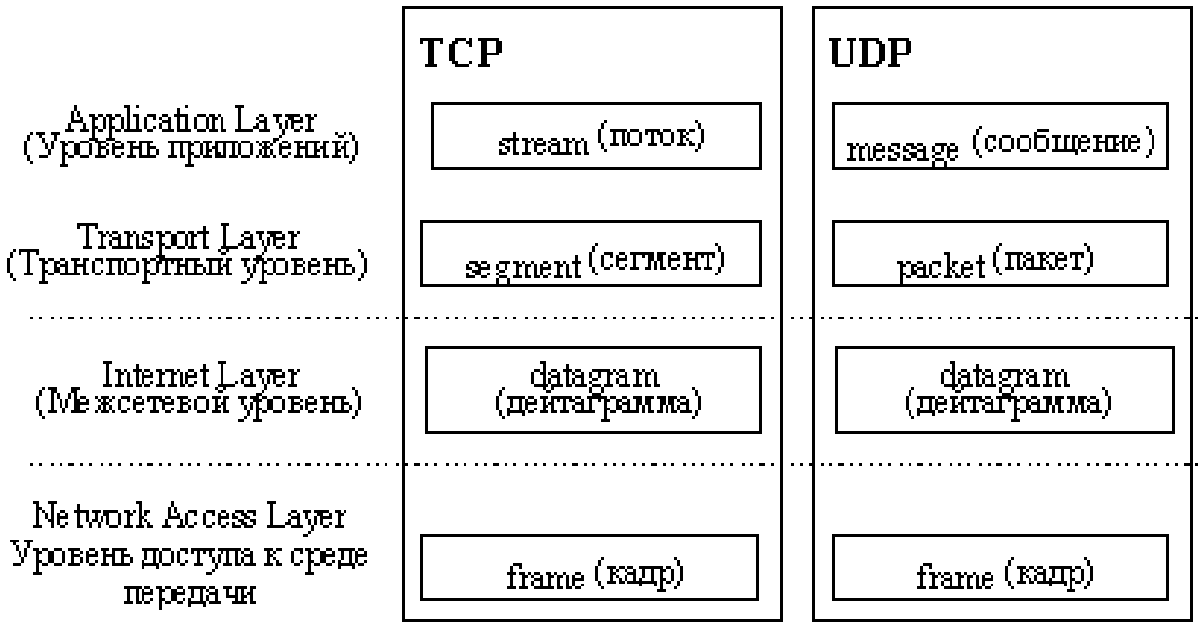



Рис. 5.1. Стек протоколов TCP/IP

Ниже рассматриваются функции каждого уровня и примеры протоколов.

### 5.1.1. Network Access Layer (Уровень доступа к среде передачи)

Функции:

- отображение IP-адресов в физические адреса сети (MAC-адреса);
- инкапсуляция IP-дейтаграмм (datagrams) в кадры (frames) для передачи по физическому каналу и передача кадров.

На этом уровне работает протокол *ARP*, осуществляющий отображение адресов IP->MAC.

### 5.1.2. Internet Layer (Межсетевой уровень) и протокол IP

Основным протоколом этого уровня является протокол *IP* (Internet Protocol).

Функции IP:

- определение дейтаграммы - основного блока передачи данных в Интернет;
- определение схемы адресации в Интернет;
- передвижение данных между транспортным уровнем и уровнем доступа к среде передачи;
- маршрутизация дейтаграмм;
- фрагментация и дефрагментация дейтаграмм.

Протокол IP доставляет данные от одного IP-адреса к другому. Заголовок дейтаграммы содержит IP-адреса отправителя и получателя и другую служебную информацию. При необходимости на уровне протокола IP происходит фрагментация и дефрагментация дейтаграмм. Такая необходимость может возникнуть на границе физических сред с различными MTU (Maximum Transfer Unit - максимальный размер передаваемого блока данных).

Протокол IP является *ненадежным* протоколом *без установки соединения*. Это означает, что протокол IP не подтверждает доставку данных, не контролирует целостность полученных данных и не производит операцию квитирования (handshaking) - обмена служебными сообщениями, подтверждающими установку соединения с хостом назначения. IP-дейтаграмма запускается в сеть и ее дальнейшая судьба никак не контролируется хостом отправления (на уровне протокола IP). Если дейтаграмма не может быть доставлена, она уничтожается. Хост, уничтоживший дейтаграмму, отправляет по обратному адресу *ICMP-сообщение* (см. далее) о причине сбоя.

*Хостом* в Интернет называется любой компьютер, имеющий IP-соединение с сетью. IP-адрес уникально идентифицирует в Интернет IP-интерфейс хоста. Это значит, что хост, имеющий несколько IP-интерфейсов (например, несколько сетевых карточек, подсоединенных к локальным сетям с IP-трафиком), имеет несколько IP-адресов.

### 5.1.3. Протокол ICMP

Вторым важным протоколом межсетевого уровня является протокол управляющих сообщений Интернет — ICMP (Internet Control Message Protocol), являющийся неотъемлемой частью модуля IP.

Протокол ICMP доставляет диагностические и управляющие сообщения от одного IP-адреса к другому. Сообщения делятся на типы, определяемые номерами, внутри типов сообщения идентифицируются числовыми кодами или именами.

### 5.1.4. Transport Layer (Транспортный уровень)

Протоколы транспортного уровня обеспечивают прозрачную доставку данных (end-to-end delivery service) между двумя процессами. Процесс внутри хоста идентифицируется номером, который называется номером порта. Таким образом, роль адреса на транспортном уровне выполняет номер порта (или, проще, - порт). Совокупность IP-адреса и номера порта называется сокетом (socket). Как IP адрес уникально определяет в Интернет IP-интерфейс (хост), сокет уникально идентифицирует в Сети конкретный процесс.

Например, сокет 194.84.124.4.25 состоит из IP-адреса хоста 194.84.124.4 и номера порта 25 и идентифицирует запущенный на хосте 194.84.124.4 демон электронной почты, который всегда использует порт 25. Этот и некоторые другие порты относятся к так называемым «широко известным сервисам», т.е. их номера закреплены за процессами, выполняющими определенные стандартные функции. Например, при обращении на порт номер 80 всегда устанавливается связь с сервером WWW (если таковой вообще запущен на вызываемом хосте). Список портов хорошо известных сервисов в паре с названием обслуживающего каждый сервис транспортного протокола находится в файле /etc/services.

На транспортном уровне работают два основных протокола: *UDP* и *TCP*.

### 5.1.5. Протокол UDP

UDP (User Datagram Protocol, протокол пользовательских дейтаграмм) является *ненадежным* протоколом *без установления соединения*. Фактически он не выполняет каких-либо особых функций, дополнительно к функциям протокола IP. Протокол UDP используется либо при пересылке коротких сообщений, когда накладные расходы на установление связи и проверку успешной доставки данных оказываются выше расходов на повторную (в случае неудачи) пересылку сообщения, либо в том случае, когда сама организация процесса-приложения обеспечивает установление соединения и проверку доставки пакетов (например, NFS).

Протокол UDP используют следующие прикладные процессы:

- NFS (Network File System - сетевая файловая система),
- TFTP (Trivial File Transfer Protocol - простой протокол передачи файлов),
- SNMP (Simple Network Management Protocol - простой протокол управления сетью),
- DNS (Domain Name Service - доменная служба имен).

### 5.1.6. Протокол TCP

TCP (Transmission Control Protocol - протокол контроля передачи) - *надежный байт-ориентированный (byte-stream)* протокол с *установлением соединения*.

Протокол TCP осуществляет передачу данных, которыми является неинтерпретируемая протоколом последовательность (поток) байт, разбиваемая по сегментам. Передача данных осуществляется между сокетами с предварительным выполнением диалоговой процедуры установления соединения и с контролем успешной доставки сегментов в процессе пересылки.

Механизм обеспечения надежности в протоколе TCP действует по принципу PAR (Positive Acknowledgment with Retransmission), который состоит в том, что данные передаются заново, если не получено подтверждение об их успешном приеме. Время ожидания подтверждения определяется таймером повторной передачи; при сбросе таймера неподтвержденные данные передаются заново. Контроль целостности данных осуществляется с помощью контрольной суммы. Байты в TCP-сегментах нумеруются (сквозная нумерация по всему потоку) с целью восстановления их исходного порядка в пункте назначения и определения пропущенных сегментов (так как протокол IP работает без установления соединения, нет никакой гарантии, что все IP-дейтаграммы, посланные от одного хоста к другому, будут следовать одним и тем же маршрутом и прибывать к месту назначения в порядке отправки, а точнее - что они вообще будут прибывать к месту назначения).

Ниже мы рассмотрим некоторые подробности работы протокола TCP.

*TCP. Установка логического соединения между хостами (handshake)*

Предположим, хост A желает установить соединение с хостом B (рис. 5.2). Первый отправляемый из A в B TCP-сегмент не содержит полезных данных, а служит для установления соединения. В его заголовке (в поле Flags) установлен бит SYN, означающий запрос связи, и содержится ISN (Initial Sequence Number - начальный номер последовательности) - число, начиная с которого хост A будет нумеровать отправляемые байты (обычно 0). В ответ на получение такого сегмента хост B откликается посылкой TCP-сегмента, в заголовке которого установлен бит ACK, подтверждающий установление соединения для получения данных от хоста A. Т.к. протокол TCP обеспечивает полнодуплексную передачу данных, то хост B в этом же сегменте устанавливает бит SYN, означающий запрос связи для передачи данных от B к A, и передает свой ISN (обычно 0). Полезных данных этот сегмент также не содержит. Третий TCP-сегмент в сеансе посылается из A в B в ответ на сегмент, полученный из B. Так как соединение A→B можно считать установленным (получено подтверждение

от B), от хост A включает в свой сегмент полезные данные, нумерация которых начинается с номера ISN(A)+1 (обычно 1). Данные нумеруются по количеству отправленных байт. В заголовке этого же сегмента хост A устанавливает бит ACK, подтверждающий установление связи B→A, что позволяет хосту B включить в свой следующий сегмент полезные данные для A.

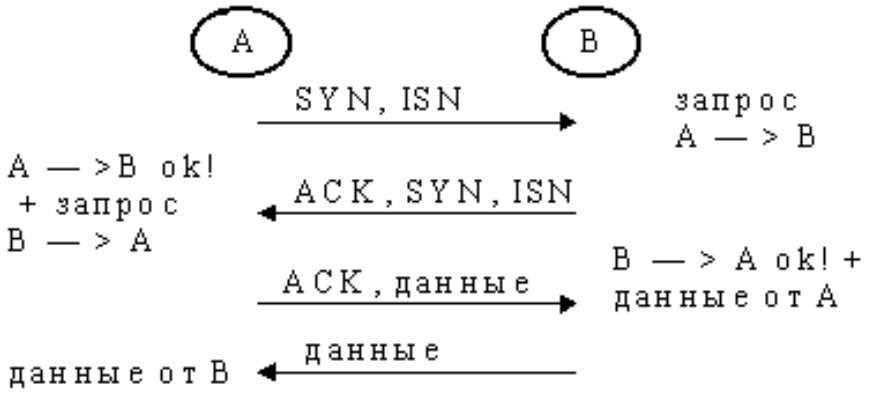



Рис. 5.2. Установка TCP-соединения

Сеанс обмена данными заканчивается процедурой разрыва соединения, которая аналогична процедуре установки, с той разницей, что вместо SYN для разрыва используется служебный бит FIN («данных для отправки больше не имею»), который устанавливается в заголовке последнего сегмента с данными, отправляемого хостом.

*TCP. Использование скользящих окон*

Для обслуживания больших потоков информации TCP использует метод скользящего окна, что позволяет отправителю посылать очередной сегмент не дожидаясь подтверждения о получении в пункте назначения предшествующего сегмента.

Протокол TCP формирует подтверждения не для каждого конкретного успешно полученного пакета, а для всех данных, от начала посылки до некоторого порядкового номера ACK SN (Acknowledge Sequence Number). В качестве подтверждения успешного приема, например, первых 2000 байт, высылается ACK SN = 2000: это означает, что все данные в байтовом потоке под номерами от ISN+1=1 до данного SN (2000) успешно получены (рис. 5.3). Вместе с посылкой отправителю ACK SN получатель объявляет также «размер окна», например - 6000. Это значит, что отправитель может посылать данные с порядковыми номерами от текущего ACK SN+1 = 2001 до (ACK SN + размер окна) = 8000, не дожидаясь подтверждения со стороны получателя. Допустим, в данный момент отправитель посылает тысячебайтовый сегмент с порядковым номером данных SN=4001. Если не будет получено новое подтверждение (новый ACK SN), отправитель будет посы-

лать данные, пока он остается в пределах объявленного окна, т.е. до номера 8001. После этого посылка данных будет прекращена до получения очередного подтверждения и (возможно) нового размера окна. Однако, размер окна выбирается таким образом, чтобы подтверждения успевали приходить вовремя и остановки передачи не происходило - для этого и предназначен метод скользящего окна. Размер окна может динамически изменяться получателем. Например, для временной остановки посылки данных достаточно объявить нулевое окно.

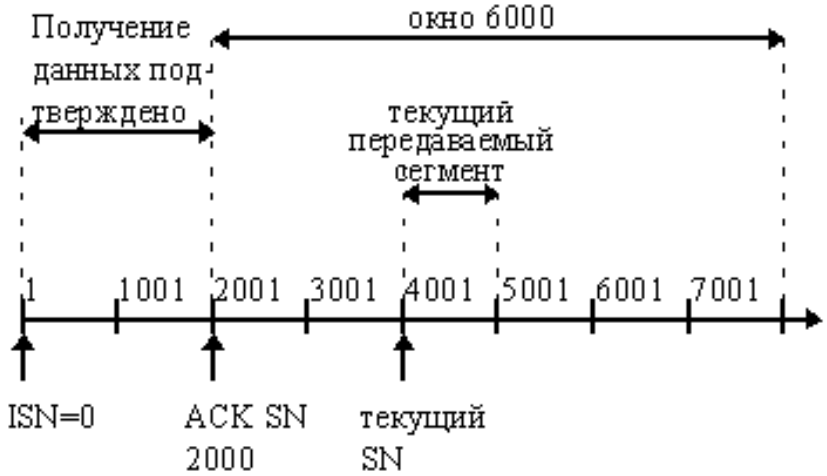



Рис. 5.3. Метод скользящего окна

Как уже было сказано выше, протокол TCP позволяет вести полнодуплексную передачу. Один и тот же сегмент, высылаемый, например, из B в A, может содержать в заголовке служебную информацию по подтверждению получения данных от A, а в поле данных - полезные данные для A. Протокол TCP оптимизирует размер сегмента и уничтожает дубликаты сегментов, которые могут быть высланы из-за задержки подтверждения.

### *5.2. IP-адреса*

IP-адрес является уникальным 32-битным идентификатором IP-интерфейса в Интернет. Часто говорят, что IP-адрес присваивается хосту; это верно в случае, если хост имеет один сетевой IP-интерфейс (например, не является маршрутизатором). IP-адреса принято записывать разбивкой всего адреса по октетам, каждый октет записывается в виде десятичного числа, числа разделяются точками. Например, адрес

*10100000010100010000010110000011*

записывается как

*10100000.01010001.00000101.10000011 = 160.81.5.131.*

IP-адрес состоит из номера сети, который занимает старшую область адреса, и номера хоста в этой сети, который занимает младшую часть. Положение границы сетевой и хостовой части (обычно оно характеризуется

количеством бит, отведенных на номер сети) может быть различным, определяя различные типы IP-адресов, которые рассматриваются ниже.

### 5.2.1. Классовая модель

В классовой модели (рис. 5.4) IP-адрес может принадлежать к одному из четырех классов сетей. Каждый класс характеризуется определенным размером сетевой части адреса, кратным восьми, таким образом, граница между сетевой и хостовой частями IP-адреса в классовой модели всегда проходит по границе октета. Принадлежность к тому или иному классу определяется по старшим битам адреса.

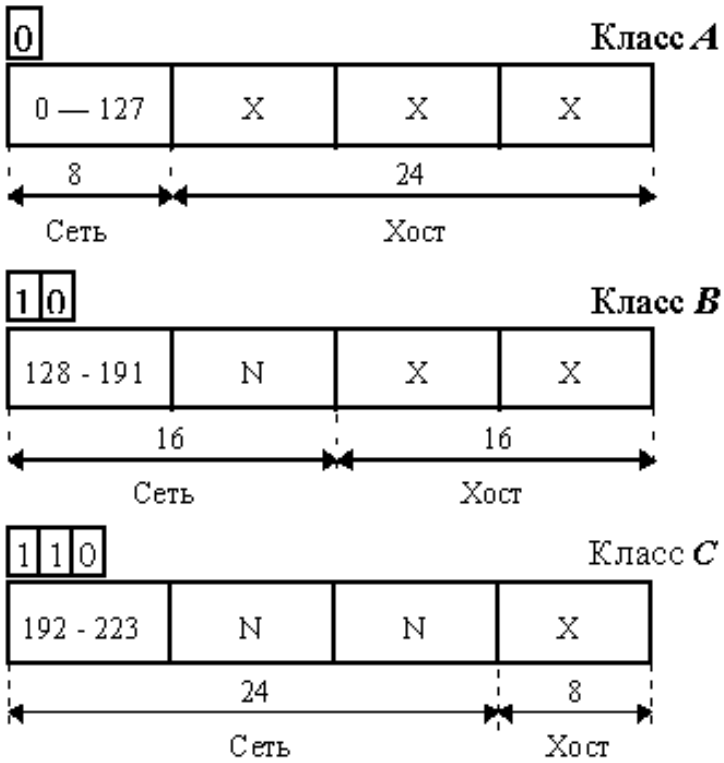



Рис. 5.4. Классы IP-адресов

*Класс A*. Старший бит адреса равен нулю. Размер сетевой части равен 8 битам. Таким образом, может существовать всего примерно $2^7$ сетей класса A, но каждая сеть обладает адресным пространством на $2^{24}$ хостов. Т.к. старший бит адреса нулевой, то все IP-адреса этого класса имеют значение старшего октета в диапазоне 0 — 127, который является также и номером сети.

*Класс B*. Два старших бита адреса равны 10. Размер сетевой части равен 16 битам. Таким образом, может существовать всего примерно $2^{14}$ сетей класса B, каждая сеть обладает адресным пространством на $2^{16}$ хостов. Значения старшего октета IP-адреса лежат в диапазоне 128 — 191, при этом номером сети являются два старших октета.

*Класс C*. Три старших бита адреса равны 110. Размер сетевой части равен 24 битам. Количество сетей класса C - примерно $2^{21}$, адресное пространство каждой сети рассчитано на 254 хоста. Значения старшего октета IP-адреса лежат в диапазоне 192 - 223, а номером сети являются три старших октета.

*Класс D*. Сети со значениями старшего октета IP-адреса 224 и выше. Зарезервированы для специальных целей.

В классе A выделены две особые сети, их номера 0 и 127. Сеть 0 используется при маршрутизации как указание на маршрут по умолчанию. Сеть 127 используется для адресации IP-интерфейсом себя самого (*loopback*). На любом хосте обращение по адресу 127.0.0.1 означает связь с самим собой (без выхода пакетов данных на уровень доступа к среде передачи); для протоколов на уровнях транспортном и выше такое соединение неотличимо от соединения с удаленным хостом, что удобно использовать, например, для тестирования сетевого программного обеспечения.

В любой сети (это справедливо и для бесклассовой модели, которую мы рассмотрим ниже) все нули в номере хоста обозначают саму сеть, все единицы - адрес широковещательной передачи (broadcast).

Например, 194.124.84.0 - сеть класса C, номер хоста в ней определяется последним октетом. При отправлении широковещательного сообщения оно отправляется по адресу 194.84.124.255. Номера, разрешенные для присваивания хостам: от 1 до 254 (194.84.124.1 — 194.84.124.254), всего 254 возможных адреса.

Другой пример: в сети 135.198.0.0 (класс B, номер хоста занимает два октета) широковещательный адрес 135.198.255.255, диапазон номеров хостов: 0.1 — 255.254 (135.198.0.1 — 135.198.255.254).

### 5.2.2. Бесклассовая модель

Предположим, в локальной сети, подключаемой к Интернет, находится 2000 компьютеров. Каждому из них требуется выдать IP-адрес. Для получения необходимого адресного пространства нужны либо 8 сетей класса C, либо одна сеть класса B. Сеть класса B вмещает 65534 адреса, что много больше требуемого количества. При общем дефиците IP-адресов такое использование сетей класса B расточительно. Однако, если мы будем использовать 8 сетей класса C, возникнет следующая проблема: каждая такая сеть должна быть представлена отдельной строкой в таблицах маршрутов на маршрутизаторах, потому что с точки зрения маршрутизаторов — это 8 абсолютно никак не связанных между собой сетей, маршрутизация дейтаграмм в которые осуществляется независимо, хотя, фактически, эти IP-сети и расположены в одной физической локальной сети и маршруты к ним идентичны. Таким образом, экономя адресное пространство, мы мно-

гократно увеличиваем служебный трафик в сети и затраты по поддержанию и обработке маршрутных таблиц.

С другой стороны, нет никаких формальных причин проводить границу сеть-хост в IP-адресе именно по границе октета. Это было сделано исключительно для удобства представления IP-адресов и разбиения их на классы. Если выбрать длину сетевой части в 21 бит, а на номер хоста отвести, соответственно, 11 бит, мы получим сеть, адресное пространство которой содержит 2046 IP-адресов, что максимально точно соответствует поставленному требованию. Это будет *одна* сеть, определяемая своим уникальным 21-битным номером, следовательно, для ее обслуживания потребуется только *одна* запись в таблице маршрутов.

Единственная проблема, которую осталось решить: как определить, что на сетевую часть отведен 21 бит? В случае классовой модели, старшие биты IP-адреса определяли принадлежность этого адреса к тому или иному классу, и, следовательно, количество бит, отведенных на номер сети.

В случае адресации вне классов, с произвольным положением границы сеть-хост внутри IP-адреса, к IP-адресу прилагается 32-битовая маска, которую называют *маской сети* (netmask) или *маской подсети* (subnet mask). Сетевая маска конструируется по следующему правилу:

- на позициях, соответствующих номеру сети, биты установлены;
- на позициях, соответствующих номеру хоста, биты сброшены.

Описанная выше модель адресации называется бесклассовой (CIDR - Classless Internet Direct Routing, прямая бесклассовая маршрутизация в Интернет). В настоящее время классовая модель считается устаревшей и маршрутизация и (большей частью) выдача блоков IP-адресов осуществляются по модели CIDR, хотя классы сетей еще прочно удерживаются в терминологии.

*Запись адресов в бесклассовой модели*

Для удобства записи IP-адрес в модели CIDR часто представляется в виде a.b.c.d / n, где a.b.c.d — IP адрес, n — количество бит в сетевой части.

Пример: 137.158.128.0/17

Маска сети для этого адреса: 17 единиц (сетевая часть), за ними 15 нулей (хостовая часть), что в октетном представлении равно

*11111111.11111111.10000000.00000000 = 255.255.128.0.*

Представив IP-адрес в двоичном виде и побитно умножив его на маску сети, мы получим номер сети (все нули в хостовой части). Номер хоста в этой сети, мы можем получить, побитно умножив IP-адрес на инвертированную маску сети.

Пример: IP = 205.37.193.134/26 или, что то же,

*IP = 205.37.193.134 netmask = 255.255.255.192.*

Распишем в двоичном виде:

*IP = 11001101 00100101 11000111 10000110*

*маска =* `11111111 11111111 11111111 11000000`

Умножив побитно, получаем номер сети (в хостовой части - нули):

*network =* `11001101 00100101 11000111 11000000`

или, в октетном представлении, 205.37.193.128/26 или, что то же, 205.37.193.128 netmask 255.255.255.192.

Хостовая часть рассматриваемого IP адреса равна 000110, или 6. Таким образом 205.37.193.134/26 адресует хост номер 6 в сети 205.37.193.128/26. В классовой модели адрес 205.37.193.134 определял бы хост 134 в сети класса С 205.37.193.0, однако указание маски сети (или количества бит в сетевой части) однозначно определяет принадлежность адреса к бесклассовой модели.

Очевидно, что сети классов А, В, С в бесклассовой модели представляются при помощи масок, соответственно, 255.0.0.0 (или /8), 255.255.0.0 (или /16) и 255.255.255.0 (или /24).

### 5.2.3. Установка IP-адреса хоста

В MS Windows IP-адрес компьютера устанавливается через *Настройки - Панель Управления - Сеть - TCP/IP - Свойства -Адрес IP*. Адрес можно либо вписать вручную (вместе с сетевой маской), либо получить автоматически, если стек TCP/IP на хосте конфигурируется по сети с помощью DHCP сервера.

Unix: IP-адрес хоста (точнее, какого-либо интерфейса хоста) и маска подсети устанавливаются с помощью команды ifconfig:

*ifconfig le0 inet 194.84.124.4 netmask 255.255.255.0*

где le0 - имя сетевого интерфейса (может изменяться в зависимости от типа интерфейса, аппаратной части и операционной системы), inet указывает на присвоение IP-адреса. Некоторые системы требуют также явного указания широковещательного адреса (broadcast). Подробнее см. man ifconfig. Для выполнения операции назначения IP-адреса при загрузке системы, вызов команды ifconfig заносится в файл автозагрузки (типа /etc/rc*).

## *5.3. Маршрутизация*

Процесс маршрутизации дейтаграмм состоит в определении следующего узла (*hop*) в пути следования дейтаграммы и пересылки дейтаграммы этому узлу. Здесь под узлом понимается либо хост назначения, либо промежуточный маршрутизатор, задача которого — определить следующий узел и переслать ему дейтаграмму. Ни хост-отправитель, ни никакой промежуточный маршрутизатор не имеют информации о всей цепочке, по которой пересылается дейтаграмма; каждый маршрутизатор, а также хост-отправитель, основываясь на адресе назначения дейтаграммы, нахо-

дит только следующий узел ее маршрута. Маршрутизация дейтаграмм осуществляется на уровне протокола IP.

Маршрутизация выполняется на основе данных, содержащихся в таблице маршрутов. Строка в таблице маршрутов состоит из следующих полей:

- адрес сети назначения,
- адрес узла, который знает, куда дальше отправить дейтаграмму, адресованную в сеть назначения,
- вспомогательные поля.

Таблица может составляться вручную или с помощью специализированных протоколов (особенно на мощных магистральных маршрутизаторах). Каждый хост имеет таблицу маршрутов, хотя бы самую простую.

### 5.3.1. Пример маршрутизации

Рассмотрим процесс маршрутизации на примере. Допустим (рис. 5.5), хосты A и B находятся в сети 1, сеть 1 соединяется с сетью 2 с помощью маршрутизатора G1. К сети 2 подключен маршрутизатор G2, соединяющий ее с сетью 3, в которой находится хост C.

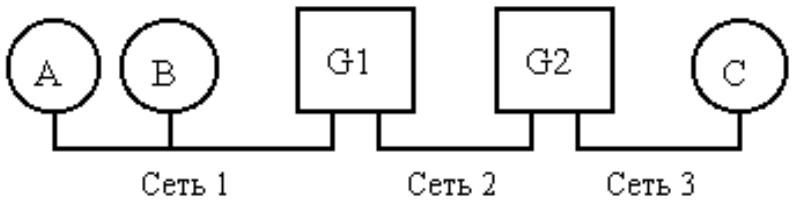



Рис. 5.5. Пример маршрутизации

Дейтаграммы, адресованные хостам сети 1 отправляет сам хост A (т.к. это его локальная сеть), а дейтаграммы, адресованные в любую другую сеть (это называется маршрут по умолчанию), хост A отправляет маршрутизатору G1, чтобы тот занялся их дальнейшей судьбой.

Предположим, хост A посылает дейтаграмму хосту B. В этом случае, поскольку адрес B принадлежит той же сети, что и A, из таблицы маршрутов хоста A определяется, что доставка осуществляется непосредственно самим хостом A.

Если хост A отправляет дейтаграмму хосту C, то он определяет по IP-адресу C, что хост C не принадлежит к сети 1. Согласно таблице маршрутов A, все дейтаграммы с пунктами назначения, не принадлежащими сети 1, отправляются на маршрутизатор G1 (маршрут по умолчанию). При этом хост A не знает, что маршрутизатор G1 будет делать с его дейтаграммой и каков будет ее дальнейший маршрут - это забота исключительно G1. G1 в свою очередь по своей таблице маршрутов определяет, что все дейтаграммы, адресуемые в сеть 3, должны быть пересланы на маршрутизатор

G2. Это может быть как явно указано в таблице, находящейся на G1, так и в виде маршрута по умолчанию.

На этом функции G1 заканчиваются, дальнейший путь дейтаграммы ему неизвестен и его не интересует. Маршрутизатор G2, получив дейтаграмму, определяет, что она адресована в одну из сетей (№3), к которым он присоединен непосредственно, и доставляет дейтаграмму на хост С.

### 5.3.2. Пример подключения локальной сети организации к Интернет

Рассмотрим пример подключения к Интернет локальной сети организации (рис. 5.6). IP-адрес локальной сети - 194.84.124.0/24 (сеть класса С). В эту сеть включен маршрутизатор G1. IP-интерфейс этого маршрутизатора, подключенный к локальной сети Ethernet, имеет адрес 194.84.124.1. Второй IP-интерфейс маршрутизатора подключен к выделенной линии (синхронный последовательный канал), ведущей к провайдеру Интернет. К другому концу этой линии подключен IP-интерфейс маршрутизатора G2, принадлежащего провайдеру. Эти два интерфейса образуют отдельную сеть 194.84.0.116/30. В этой сети на номер хоста отведено всего 2 бита — 4 варианта адресов, из которых один (00) обозначает саму сеть, один (11) — широковещательный, таким образом в подобной сети может находиться всего 2 хоста — это минимальная возможная сеть. Интерфейс маршрутизатора G1 в сети 194.84.0.116/30 имеет адрес 194.84.0.117, а маршрутизатора G2 — 194.84.0.118. Маршрутизатор G2 имеет еще некоторое количество интерфейсов, к части которых подключены выделенные линии от других клиентов, к части — локальные сети коммуникационного центра, другие маршрутизаторы и магистральные линии дальней связи.

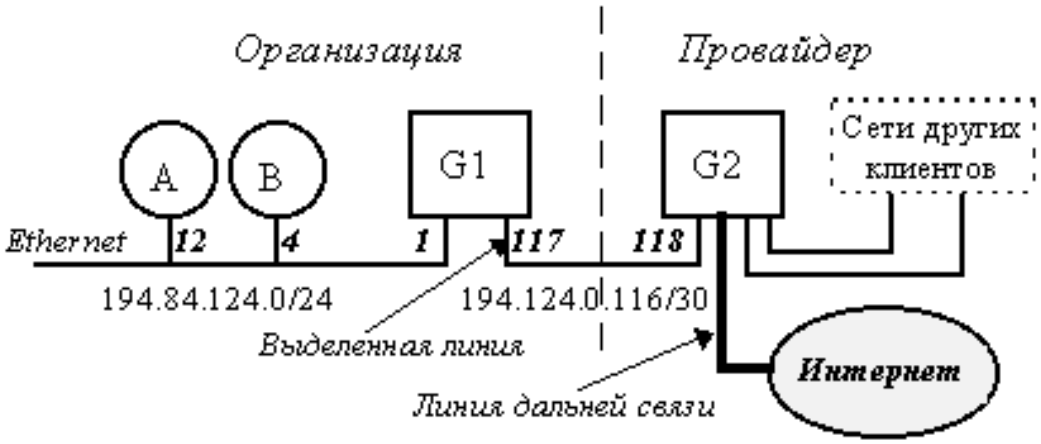



Рис. 5.6. Подключение локальной сети к Интернет

### 5.3.3. Динамическая маршрутизация

В случае сетей со сложной топологией, когда существует несколько вариантов маршрутов от одного хоста к другому и (или) когда состояние

сети (топология, качество каналов связи) изменяется с течением времени, таблицы маршрутов составляются динамически с помощью различных протоколов маршрутизации. Различают протоколы *внешней* (exterior) и *внутренней* (interior) маршрутизации.

Внутренние протоколы применяются на маршрутизаторах, действующих внутри *автономных систем*. Автономная система - это наиболее крупное деление Интернет, представляющее из себя совокупность сетей с одинаковой маршрутизационной политикой и общей администрацией, например, совокупность сетей компании Глобал Один и ее клиентов в России.

Маршрутизация *между* автономными системами осуществляется *пограничными* (border) маршрутизаторами, таблицы маршрутов которых составляются с помощью протоколов внешней маршрутизации.

В настоящее время внутренняя маршрутизация производится, в основном, с помощью протокола OSPF, также встречается применение устаревшего протокола RIP или RIP-2 и других протоколов (собирательное название - IGP, Interior Gateway Protocols). Наиболее популярным из протоколов внешней маршрутизации (EGP, Exterior Gateway Protocols) является BGP.

*Address Resolution Protocol (ARP)*

Протокол ARP предназначен для преобразования IP-адресов в MAC-адреса, часто называемые также физическими адресами.

MAC расшифровывается как Media Access Control, контроль доступа к среде передачи. MAC-адреса идентифицируют устройства, подключенные к физическому каналу, пример MAC-адреса - адрес Ethernet.

Для передачи данных (IP-дейтаграммы) по физическому каналу (будем рассматривать Ethernet) требуется в заголовке кадра указать адрес Ethernet-карты, на которую будут доставлены эти данные для их последующей обработки вышестоящими уровнями стека протоколов. IP-адрес адресует IP-интерфейс какого-либо хоста или маршрутизатора и не заключает в себе никаких указаний ни на физическую среду передачи, к которой подключен этот интерфейс, ни, тем более, на физический адрес устройства (если таковой имеется), с помощью которого этот интерфейс сообщается со средой.

Поиск по данному IP-адресу соответствующего Ethernet-адреса производится протоколом ARP, функционирующим на уровне доступа к среде передачи. Протокол поддерживает в оперативной памяти динамическую arp-таблицу в целях кэширования полученной информации. Порядок функционирования протокола следующий.

С межсетевого уровня поступает IP-дейтаграмма для передачи в физический канал (Ethernet), вместе с дейтаграммой передается, среди прочих параметров, IP-адрес хоста назначения. Если в arp-таблице не содержится

записи об Ethernet-адресе, соответствующем нужному IP-адресу, модуль arp ставит дейтаграмму в очередь и формирует запрос. Запрос (arp-request) представляет из себя Ethernet-кадр с широковещательным адресом, содержащий метку о том, что это arp-запрос, IP-адрес, подлежащий преобразованию, и пустое поле для искомого Ethernet-адреса. Запрос получают все хосты, подключенные к данной сети; хост, опознавший свой IP-адрес, заполняет пустующее поле значением своего Ethernet-адреса и отправляет arp-ответ (arp-response). Полученные данные заносятся в таблицу, ждущая дейтаграмма извлекается из очереди и передается на инкапсуляцию в кадр Ethernet для последующей отправки по физическому каналу.

*ARP для дейтаграмм, направленных в другую сеть*

Заметим, что дейтаграмма, направленная во внешнюю (в другую) сеть, должна быть передана маршрутизатору. Хотя в заголовке дейтаграммы помещен IP-адрес ее хоста назначения (где-то во внешней сети), на физическом уровне кадр, инкапсулирующий эту дейтаграмму, содержит MAC-адрес маршрутизатора. Это достигается тем, что межсетевой уровень передает arp-модулю в качестве IP-адреса хоста назначения вместо действительного адреса адрес маршрутизатора (извлеченный межсетевым уровнем, в свою очередь, из таблицы маршрутов).

Модуль уровня доступа к среде передачи на маршрутизаторе извлекает из кадра присланную дейтаграмму и отправляет ее для обработки на межсетевой уровень. Модуль IP обнаруживает, что дейтаграмма адресована не ему и по своей таблице маршрутов определяет, куда ее следует переслать. Далее дейтаграмма опускается на нижний уровень, к соответствующему физическому интерфейсу, которому передается в качестве IP-адреса хоста назначения адрес следующего маршрутизатора, извлеченный из таблицы маршрутов, или непосредственно адрес конечного хоста, если данный маршрутизатор может доставить дейтаграмму непосредственно к нему.

Работать с arp-таблицей вручную можно с помощью программы arp (Unix, MS Windows).

### 5.3.4. Перечень задач по подключению сети предприятия к Интернет

1. Выбор провайдера

*Функции* провайдера: обеспечение доставки дейтаграмм между Интернет и сетью предприятия.

Провайдер *предлагает* тип физического соединения и его скорость, протокол канального уровня OSI (PPP, Frame Relay и т.п.) для этого соединения, точку подключения.

Провайдер *выделяет* блок IP-адресов для сети предприятия.

Провайдер *берет деньги* за установку IP-сервиса (единовременно) и за предоставление IP-сервиса (абонентская плата и/или плата за объем трафика).

Провайдер *рекомендует* (и продает или сдает в аренду) оборудование канала передачи данных (CSU/DSU, «цифровой модем») и маршрутизатор (за отдельную плату может производиться настройка и поддержка маршрутизатора).

*Дополнительно* провайдер может оказывать платные или бесплатные услуги по регистрации домена клиента в системе DNS, поддержке первичного и вторичных серверов DNS для домена клиента, подключению клиента к службе телеконференций (Usenet news), организации для клиента других сервисов (WWW, email, FTP).

2. Организация физического канала («последняя миля»)

Прокладка кабеля, аренда выделенной линии и кроссирование линий на АТС и т.п. Выполняется по рекомендации и согласованию с провайдером. Провайдер также может взяться за эту работу за отдельную плату.

Выделение и проводка коммутируемых телефонных линий для дозвона в сеть предприятия, если это требуется (например, выход в Интернет для сотрудников предприятия из дома).

3. Определение схемы IP-адресации в сети предприятия и схемы обеспечения безопасности (если требуется)

Будет ли использоваться брандмауэр (firewall) и какова будет схема защиты сети; будет ли основная часть сети предприятия изолирована за брандмауэром?

Как будут распределяться IP-адреса: фиксированно или динамически с помощью DHCP; каков общий объем требуемого адресного пространства?

4. Регистрация

Получение от провайдера адресного пространства на необходимое количество IP-адресов.

Регистрация имени домена для сети предприятия и обратной зоны в системе DNS (как правило, с помощью провайдера).

5. Приобретение оборудования

Требуется приобрести и установить:

- оборудование канала передачи данных (рекомендуется и, как правило, продается провайдером);
- маршрутизатор
- модемы для дозвона (если требуются),
- компьютеры для серверных функций, операционные системы и программное обеспечение (DNS, email, WWW, FTP серверы, возможно сервер доступа для дозвона, сервер телеконференций).

При выборе маршрутизатора следует определить

- тип и количество внутренних интерфейсов - например, 1 Ethernet;

- тип и количество внешних интерфейсов и используемые ими канальные протоколы (в соответствии с тем, что предлагает провайдер) - например, 1 последовательный интерфейс V.35 для передачи PPP-трафика;
- какие протоколы маршрутизации должны быть задействованы;
- требуются ли порты для подключения модемов в целях обеспечения дозвона в сеть предприятия;
- нужна ли модульность маршрутизатора (перспективы расширения сети).

Продукты фирм Cisco и Bay Networks закрывают весь спектр возможных требований.

6. Настройка передачи IP-трафика

Запуск оборудования обслуживания канала, тестирование работоспособности канала с помощью этого оборудования.

Запуск и настройка маршрутизатора (как правило, на статическую маршрутизацию). Выполняется один раз, может быть сделана провайдером. Работа с маршрутизаторами подробно изучается в курсе «Основы телекоммуникаций».

Обеспечение передачи IP-трафика между портами маршрутизатора и провайдером.

Обеспечение передачи IP-трафика между хостом в сети предприятия и хостом в Интернет.

7. Запуск DNS

Выбор хоста для установки DNS-сервера, согласование и назначение вторичных серверов.

Запуск сервера, создание баз данных DNS.

Регистрация сервера, получение подтверждения о выделении серверу полномочий по обслуживанию домена предприятия. Эту задачу поможет выполнить провайдер.

8. Обеспечение предприятия электронной почтой.

Определение схемы предоставления электронной почты и формирования адресов (центральный сервер с доступом по протоколу POP3 с рабочих станций, сервер локальной сети (Netware, Lotus) со своей службой электронной почты и шлюзом в почтовую систему Интернет и т.п.).

Установка и конфигурирование сервера электронной почты.

9. Обеспечение информационных сервисов.

Установка WWW и FTP серверов.

Наполнение серверов (администрирование совместной разработки страниц с определением прав доступа, поддержка архивов).

Обеспечение доступа к телеконференциям.

10. Организация доступа в сеть предприятия по дозвону (если требуется)

Определить, что будет являться сервером доступа, какого типа сетевые услуги будут предоставляться удаленным пользователям, как будет

происходить аутентификация и авторизация пользователей, какое программное обеспечение должен иметь удаленный пользователь, рекомендовать модель модема.

Обеспечение IP-трафика по коммутируемому каналу, в том числе и организация дозвона, рассматриваются в курсе «Основы телекоммуникаций». В Приложении приведена инструкция по запуску сервера удаленного доступа под Windows NT.

11. Установка необходимого программного обеспечения на рабочие станции пользователей, инструктаж пользователей

Для предоставления услуг Интернет неквалифицированным пользователям рекомендуется выбрать один интегрированный программный пакет (типа Netscape Communicator), обеспечивающий весь спектр услуг, необходимых рядовому пользователю, и установить его на рабочие станции.

Провести курсы по «Введению в Интернет».

### *5.4. Работа с утилитами TCP/IP*

TCP/IP соединение устанавливается между двумя сокетами. Сокет вызывающей стороны (клиента) имеет, как правило, номер порта больший 1023, т.е. в диапазоне, не зарезервированном для системного применения или использования большинством хорошо известных сервисов. Номера портов клиентов динамически выделяются системой из свободного пространства при запуске клиента. Сокет вызываемой стороны при обращении к хорошо известному сервису, например, серверу электронной почты, имеет номер порта в соответствии с файлом /etc/services.

Для установления соединения необходимо, чтобы на вызываемой стороне работала программа-сервер, подключенная к требуемому порту (говорят, что программа «слушает порт» на предмет поступления запросов на соединение). В Unix-системах такие программы, запущенные в фоновом режиме, называют демонами.

#### 5.4.1. Основные утилиты TCP/IP

Перечисленные ниже программы имеются как в любой Unix-системе, так и в MS Windows:

**telnet *host* [*port*]** - эмулятор удаленного терминала; устанавливает соединение с указанным хостом и используется для исполнения программ на удаленной машине в интерактивном режиме (командной строке); также с помощью программы telnet можно установить соединение с демоном на удаленной машине, подключенным к любому порту (по умолчанию - порт 23, т.е. собственно telnet).

**ping *host*** - программа генерации эхо-пакетов к указанному хосту; печатает время оборота каждого пакета, статистику утерянных пакетов; ис-

пользуется для тестирования качества связи (в ОС Solaris для вывода статистики используется ключ -s).

**traceroute *host*** (tracert - в MS Windows) - программа исследования маршрута соединения к указанному хосту; печатает последовательность промежуточных узлов и время оборота пакетов; позволяет проследить маршрут и место обрыва соединения в случае необходимости.

**netstat -n** - выдает список установленных IP соединений; формат вывода на разных системах слегка различается, но для каждого соединения присутствует следующая информация: IP-транспорт (TCP, UDP), сокет соединения на локальном хосте, сокет на удаленном хосте, состояние соединения, например:

*TCP 194.84.124.25.1030 194.84.124.4.25 ESTABLISHED*

означает, что на хосте 194.84.124.25 (локальная машина пользователя) с порта 1030 работает клиент, установивший связь с портом 25 хоста 194.84.124.4 (на порте 25 работает сервер электронной почты), используется транспорт TCP, связь установлена (ESTABLISHED).

**netstat -rn** выдает таблицу маршрутов.

**arp** - реализация протокола arp - поиск Ethernet-адреса по указанному IP-адресу; основные ключи:

-a - показать всю arp-таблицу,

-d *hostname* - удалить сведения об указанном хосте,

-s *hostname ethernet_addr* - внести вручную Ethernet-адрес указанного хоста.

**ftp** - реализация протокола передачи файлов FTP; работает в режиме командной строки. Основные команды:

open *hostname* - установить FTP-соединение с указанным хостом;

dir [*directory*] - вывести содержимое каталога *directory* (по умолчанию - текущего) удаленного хоста;

cd [*directory*] - перейти в каталог *directory* (по умолчанию - в домашний или начальный) на удаленном хосте;

lcd [*directory*] - то же, что и cd, но на локальном хосте;

ascii - включить режим передачи текстовых файлов (при этом символы перевода строки будут правильно преобразовываться из формата одной ОС к другой);

bin - включить двоичный режим передачи - файлы передаются «как есть»;

get *filename*, mget *filenames* - скопировать файл с удаленного хоста на локальный, mget копирует несколько файлов, в случае mget допускаются маски *?;

put *filename*, mput *filenames* - то же, что и get, mget, но с локального хоста на удаленный;

hash - включить/выключить вывод «индикатора прогресса»: при завершении копировании определенного количества килобайт на экран выводится символ #; по умолчанию выключено.

prompt - включить/выключить режим запроса действий по каждому файлу при использовании команд mget, mput; изначально включено;

help - список команд;

bye - разрыв соединения и выход из программы.

**finger *username@hostname*** - вывод информации о пользователе *username* на хосте *hostname*; в Windows как правило отсутствует, на многих Unix-хостах может быть заблокирован или ограничен по доступу.

### 5.4.2. Поиск информации об IP-сетях и автономных системах (служба WHOIS)

Служба whois осуществляет поиск информации на серверах, где существуют специальным образом организованные базы данных. Поиск производится с помощью программы whois.

Информация о распределении IP-адресов и автономных систем, о людях, ответственных за управление IP-сетями и т.п. содержится в Интернет-регистрах. Служба whois для Интернет-регистров Европы и России находится по адресу whois.ripe.net.

Пример команды для получения информации о сети 194.84.124.0:

*whois -h whois.ripe.net 194.84.124.0 | more*

Ключ -h позволяет указать хост, к которому программа должна обратиться с запросом. Вообще, далеко не каждый хост поддерживает whois-справочники и отвечает на запросы.

Среди полученной информации указываются, в частности, описание сети (Vladivostok State University network), страна (RU), ответственные лица и их адреса. Техническую ценность представляют поля «route» и «origin». В поле «route» указан блок CIDR, внутри которого выделена данная сеть (194.84.0.0/16), рядом поле «descr» описывает данный блок как RU-ROSPRINT BLOCK, т.е. блок адресов провайдера Роспринт, он же Глобал Один. В поле «origin» указана автономная система, внутри которой находится данная сеть (AS2854).

Пример команды для получения информации по автономной системе №2854:

*whois -h whois.ripe.net AS2854 | more*

Выдается описание автономной системы (Rosprint AS), адрес администрации и информация о политике перемещения данных между данной АС и другими АС. Эта политика используется пограничными маршрутизаторами при построении таблиц маршрутов для трафика между автономными системами. Например, строка

*as-in: from AS2118 100 accept AS-RELCOM*

означает, что данной автономной системой принимаются дейтаграммы, поступающие из AS2118, отправленные из AS-RELCOM (т.е. из той же AS2118). Строка

*as-in: from AS4000 80 accept ANY*

означает, что принимаются дейтаграммы, поступающие из AS4000, отправленные из любой автономной системы.

Строки, начинающиеся с «as-out», аналогично описывают соглашения по передаче данных, исходящих из данной АС.

### *5.5. Динамическое присвоение IP-адресов*

В этом разделе изучается работа с сервером DHCP (Dynamic Host Configuration Protocol), предназначенным для автоматической настройки параметров стека TCP/IP рабочей станции в момент ее загрузки. Эти данные передаются рабочей станции сервером DHCP после того, как станция во время своей загрузки выдаст широковещательный запрос параметров своей конфигурации, на который и откликается сервер. Данные конфигурации включают в себя IP-адрес рабочей станции, а также (опционально) адреса маршрутизатора (шлюза) и сервера DNS, имя домена и т.п.

IP-адрес, присваиваемый рабочей станции, может браться сервером из пространства специально для этого выделенных адресов (берется первый свободный адрес). В этом случае у рабочей станции нет постоянного IP-адреса. IP-адрес, присваиваемый конкретной рабочей станции, может быть и фиксированным, для этого надо знать MAC-адрес (Ethernet-адрес) рабочей станции и соответствующим образом настроить сервер.

В любом случае использование DHCP позволяет избежать конфигурирования стека TCP/IP на каждом хосте сети отдельно и проводить гибкую, централизованную административную политику.

Windows NT имеет поставляемый с системой сервер DHCP. Для работы этого сервера необходимо:

1. В настройках сети (*Настройки - Панель управления - Сеть*), в разделе *Services* добавить *Microsoft DHCP Server*.

2. Запустить сервер через *Control Panel - Services - DHCP Server* кнопкой *Start*.

3. Сервер настраивается с помощью программы *DHCP Manager*, запускаемой из раздела *Administrative Tools*.

Для каждого из серверов (программа позволяет управлять несколькими серверами) существует один или несколько контекстов (*scope*), описывающих конфигурацию и настройки сервера для той или иной сферы действия. В простейшем случае имеется один сервер с одним контекстом. Серверы и их контексты показываются в левой части окна программы.

Если контекста нет, его следует создать через меню *Scope-Create*. Существующий контекст можно редактировать через меню *Scope-*

*Properties*. В конфигурации контекста указывается диапазон IP-адресов, выделенный для динамического распределения адресов для клиентов, а также поддиапазоны, которые следует исключить (*exclude*) из этого диапазона. Параметр *Lease Duration* указывает максимальную продолжительность использования IP-адреса клиентом; значение *Unlimited* определяет неограниченное время использования.

Меню *Scope-Reservations* позволяет зафиксировать IP-адреса за определенными хостами (точнее, за определенными Ethernet-адресами). Ethernet-адрес указывается в поле *Unique Identifier*.

Передача клиентам дополнительной информации (адрес шлюза, адрес DNS-сервера и доменное имя и т.п.) конфигурируется через меню *DHCP Options* (*Global* - для всех контекстов, *Scope* - для данного контекста). Выберите нужные опции, активизируйте их с помощью кнопки *Add* и укажите значения требуемых параметров для каждой опции.

Опции для клиентов с фиксированными адресами устанавливаются через меню *Scope - Active Leases*, далее двойным щелчком вызвать свойства нужного клиента. Для ввода контекста в действие используйте меню *Scope-Activate* (*Deactivate* - для отключения контекста).

DHCP-клиент под Windows активизируется через *Настройки - Панель управления - Сеть - TCP/IP - Свойства - IP-адрес - Получить IP-адрес автоматически*. Если на хосте не сконфигурированы параметры DNS и адрес шлюза, они будут получены от DHCP-сервера, иначе будут использоваться уже имеющиеся настройки.

В случае отсутствия DHCP-сервера в сети при включенном автоматическом получении IP-адреса хост присвоит себе адрес самостоятельно. В этом случае возможно отсутствие связности из-за некорректного адреса.

### *5.6. Получение информации из баз данных DNS*

Как известно, помимо IP-адресов, идентифицируются доменными именами, более легкими для запоминания и отражающими логическое структурирование сети и, часто, функциональное назначение того или иного хоста. Доменное имя состоит из символьных полей, разделенных точками. Крайнее правое поле обозначает домен верхнего уровня, далее, справа налево, следуют поддомены в порядке иерархической вложенности, крайнее левое поле обозначает имя хоста. Например, crypt.iae.nsk.su - хост crypt в домене iae, который находится внутри домена nsk, который в свою очередь находится внутри домена su.

Сетевое программное обеспечение, маршрутизаторы и другая сетевая аппаратура работают с IP-адресами. Преобразования «доменное имя→IP адрес» («прямое») и «IP адрес→доменное имя» («обратное») выполняются службой DNS, которая представляет собой иерархическую структуру серверов, где каждый сервер отвечает за определенную зону -

т.е. свою часть дерева доменных имен, хранит соответствующие базы данных и отвечает на запросы.

### 5.6.1. Конфигурирование клиента DNS

При конфигурировании на хосте стека TCP/IP, кроме указания IP-адреса хоста, создания таблицы маршрутов (в простейшем случае - указания IP-адреса шлюза, т.е. строки default в таблице маршрутов), обычно конфигурируется и клиент DNS. Задача клиента - взаимодействие с DNS-сервером, который будет, по запросу клиента, выполнять описанные выше преобразования. При ручном конфигурировании DNS-клиента указываются:

- имя хоста,
- домен, в котором находится данный хост,
- IP-адрес сервера DNS, обслуживающего этот домен.

Получение всех этих данных возможно автоматически - в случае конфигурирования стека TCP/IP с помощью DHCP-сервера.

В MS Windows адрес DNS-сервера, имя домена и имя хоста устанавливаются в настройках сети (выбрать протокол TCP/IP, перейти к его свойствам и выбрать закладку DNS).

В Unix имя домена и адрес DNS-сервера указываются в файле /etc/resolv.conf в формате:

*domain vvsu.ru*
*nameserver 194.84.124.4*

Имя хоста устанавливается командой hostname *hostname*.

Важным файлом в Unix является файл /etc/hosts. В нем в формате

*127.0.0.1 localhost*
*194.84.124.4 maria*

перечислены соответствия IP-адресов и имен, которые известны компьютеру непосредственно, без обращения к DNS. В небольших изолированных сетях преобразования "имя→адрес" и "адрес→имя" выполняются без использования DNS, только с помощью файла /etc/hosts. Однако и при использовании DNS файл /etc/hosts обычно содержит строку localhost и строку, содержащую имя и адрес самого хоста.

### 5.6.2. Порядок выполнения DNS-запроса

При необходимости произвести какое-либо из DNS-преобразований («адрес→имя», «имя→адрес») хост обращается к своему серверу DNS.

Если DNS-сервер не может выдать ответ на поступивший запрос (т.е. необходимые данные отсутствуют в его базе и кэше предыдущих запросов), он обращается к одному из корневых серверов (root servers). Рассмотрим на примере, что происходит дальше.

Итак, хост ada.vvsu.ru желает узнать IP-адрес хоста crypt.iae.nsk.su. Ada отправляет запрос своему DNS-серверу 194.84.124.4. Действия сервера изображены на рис. 5.7.

Сервер кэширует полученную информацию для дальнейшего возможного использования.

Преобразование «доменное имя→IP-адрес» выполняется всякий раз при попытке установления TCP/IP-соединения с хостом, если указано доменное имя этого хоста (так происходит почти всегда при работе пользователя с приложениями Интернет). Некоторые программы выполняют также и обратное DNS-преобразование. Эти операции скрыты от пользователя.

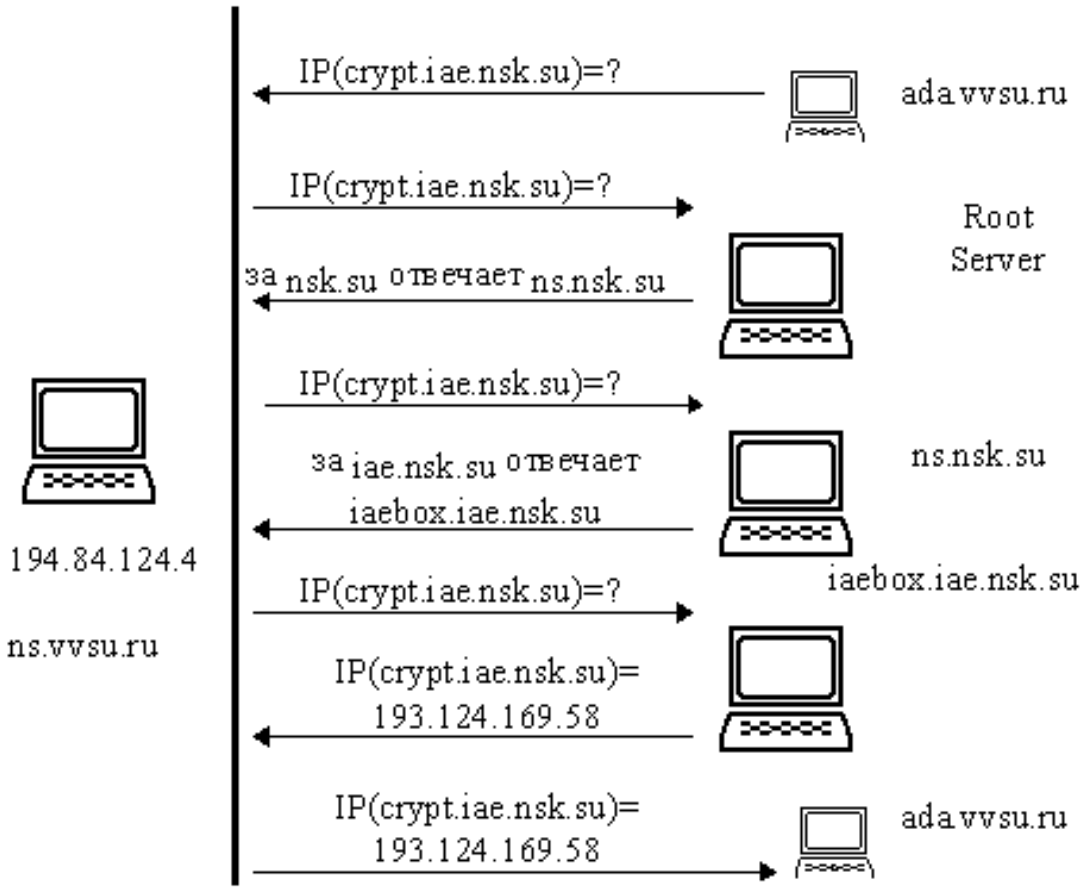



Рис. 5.7. Схема обработки запроса DNS-сервером

### 5.6.3. Программа nslookup

Программа nslookup позволяет произвести DNS-преобразования в явном виде. Например:

*%nslookup www.ibm.com*
*Server: maria.vvsu.ru*
*Address: 194.84.124.4*
*Name: www.ibm.com*
*Address: 204.146.18.33*

Вывод программы означает, что был опрошен сервер maria.vvsu.ru (его IP-адрес 194.84.124.4) и получен ответ IP(www.ibm.com) = 204.146.18.33.

Пример обратного преобразования:

*%nslookup 204.146.18.33*
*Server: maria.vvsu.ru*
*Address: 194.84.124.4*
*Name: www.ibm.com*
*Address: 204.146.18.33*

Программа nslookup работает также в режиме командной строки. Необходимые команды:

server [*имя_опрашиваемого_сервера*] - сменить опрашиваемый DNS сервер, например: server ns.kiae.su. Без аргумента - установить сервер по умолчанию («свой» сервер). Nslookup позволяет напрямую обращаться с запросами к серверам, непосредственно отвечающим за ту или иную зону. Если же ответ поступил от сервера, не отвечающего за зону, для хоста которой запрашивалась информация (например рис. 5.7), такой ответ будет помечен как «*non-authoritative answer*».

set type=*тип_стандартной_записи_ресурса* - установить запрос данных определенного типа. Например:

*>set type=NS*
*>ibm.com*

означает запрос списка DNS-серверов, отвечающих (authoritative) за домен ibm.com. Запрос в этом случае должен состоять из имени домена, а не отдельного хоста. Возможные типы:

- SOA (Start Of Authority) - заголовок зоны,
- NS (Name Server) - сервер DNS,
- A (Address) - IP-адрес (выбран по умолчанию),
- MX (Mail Exchanger) - обработчик почты,
- CNAME (Canonical Name) - каноническое имя,
- PTR (Pointer) - запрос по обратной зоне,
- ANY - все записи.

ls *имя_домена* - вывести список хостов указанного домена, например ls vvsu.ru. Предварительно следует переключиться на опрос сервера, отвечающего (authoritative) за данный домен.

help - помощь.

exit - выход.

Любой ввод, не являющийся командой, воспринимается как запрос. Перенаправление вывода в файл производится с помощью символа >.

### *5.7 Основы программирования с использованием интерфейса Winsock*

Под термином *Winsock* известен реализованный на всех платформах *Win32* сетевой интерфейс прикладного программирования. Он является основным интерфейсом доступа к разным базовым сетевым протоколам и во многих отношениях наследует применяемой на платформах *UNIX* реализации *Berkeley (BSD) Sockets*. В средах *Win32* интерфейс стал совершенно независимым от протокола, особенно после выпуска версии *Winsock 2*. В рассматриваемом контексте термин ***"Сокеты"*** *(sockets)* используется для обозначения описателей или дескрипторов поставщиков транспорта. В *Win32* сокет отличается от описателя файла, поэтому для него введен отдельный тип - *SOCKET*. В литературе по программированию можно встретить разные переводы этого слова: сокеты называют гнездами, разъемами, соединителями, и даже патронами. В силу отсутствия удачного русскоязычного термина, в настоящем тексте объекты *sockets* будут именоваться просто сокетами.

С позиций эталонной модели *OSI* интерфейс *Winsock* расположен между сеансовым и транспортным уровнями. Под управлением *Windows* прикладной, представительский и сеансовый уровни, в основном, относятся к программному приложению.

Программирование с применением сокетов считается сравнительно несложным, однако, к сожалению, в популярной литературе оно недостаточно глубоко описано, документация *Windows Sockets SDK (Software Development Kit)* содержит немало ошибок, как в теоретических описаниях, так и в демонстрационных примерах. Кроме того, существуют весьма заметные отличия реализаций сокетов в *UNIX* и в *Windows*, что создает дополнительные проблемы.

Ввиду ограниченности объема данного материала, основное внимание далее сосредоточено, преимущественно, на одной версии реализации сокетов - библиотеке *Winsock 2*, одном виде сокетов - синхронных (блокируемых) сокетах, а также одном языке программирования – C/C++ (сказанное, как правило, применимо и к языкам *Delphi, Perl* и т.д.).

Как правило, в качестве главного первоисточника при освоении библиотеки сокетов рассматривается *Windows Sockets 2 SDK*, включающий документацию, набор заголовочных файлов и инструментарий разработчика. Из такого инструментария, входящего в *SDK*, особо стоит выделить утилиту sockeye.exe, которую можно использовать как "лабораторный стенд" программиста. Утилита позволяет в интерактивном режиме вызывать различные сокетные функции и анализировать результаты.

Как отмечалось выше, в демонстрационных программах *SDK*, к сожалению, нередко встречаются ошибки. Тем не менее, анализ этих примеров полезен с точки зрения освоения полезных приемов программирова-

ния. Из Internet - ресурсов, посвященных программированию с использованием сокетов и всему, что с ними связано, в дополнение к *SDK*, можно отметить следующие [37–39].

### 5.7.1. Обзор Winsock

Socket-интерфейс представляет собой просто набор системных вызовов, которые можно разделить на четыре группы:

1) локального управления, к числу которых относятся, например, функции создания сокета и связывания сокета;
2) установления связи, в число которых входят функции ожидания установления соединения, запроса на установление соединения, приема запроса на установление соединения, формирования адреса узла сети;
3) обмена данными (ввода/вывода), к числу которых относятся функции посылки данных и получения данных;
4) закрытия связи, включая функции закрытие сокета и сброса буферизованных данных.

Библиотека *Winsock* поддерживает два вида сокетов – блокируемые или ***синхронные*** и неблокируемые или ***асинхронные***. Синхронные сокеты не возвращают управление в течение времени выполнения операции, а асинхронные возвращают немедленно, продолжая выполнение в фоновом режиме, а по завершении работы уведомляют об этом вызывающий код.

Первые массовые версии ОС компании *Microsoft* с графическим интерфейсом пользователя *Windows 3.x* поддерживали только асинхронные сокеты. Причиной было то, что в системе с корпоративной (невытесняющей) многозадачностью захват управления одной задачей "подвешивает" все остальные, включая и саму ОС. В *Windows 9.x* и *NT/2000/XP* поддерживаются оба вида сокетов, но поскольку синхронные сокеты проще программируются, асинхронные не пользуются большой популярностью у программистов. В данном описании рассматриваются, главным образом, синхронные сокеты.

Вообще, библиотека сокетов позволяет работать со множеством протоколов и является удобным средством взаимодействия между процессами, но в данном документе речь пойдет только о сокетах семейств протоколов TCP/IP (использующихся для обмена данными между узлами сети Интернет) и IPX/SPX.

Независимо от вида, сокеты делятся на два типа - ***потоковые*** и ***дейтаграммные***. Потоковые сокеты работают с установлением соединения, обеспечивая гарантированность доставки и правильность сборки данных. Дейтаграмные сокеты работают без установления соединения и не обеспечивают ни идентификации отправителя, ни контроля успешности доставки данных, зато они заметно быстрее потоковых.

Выбор типа сокетов определяется транспортным протоколом, на котором работает сервер, в частности, клиент при всем желании не сможет по своему усмотрению установить с дейтаграммным сервером потоковое соединение.

### 5.7.2. Основные шаги программы при использовании библиотеки WinSock

Для работы с библиотеками *Winsock 1.x* или *Winsock 2.x* в исходный тест программы необходимо включить директивы *"#include <winsock.h>"* или *"#include <winsock2.h>"*. При использовании для разработки утилит командной строки в строке линкера для *Winsock 2.x* необходимо указать "*ws2_32.lib*". В *Microsoft Visual Studio* для этого достаточно нажать *<Alt-F7>*, перейти к закладке "*Link*" и к списку библиотек, перечисленных в строке "*Object/Library modules*", добавить "*ws2_32.lib*", отделив ее от остальных символом пробела.

Перед началом использования функций библиотеки Winsock ее необходимо подготовить к работе вызовом функции *int **WSAStartup** (WORD wVersionRequested, LPWSADATA lpWSAData ),*
передав в старшем байте слова *wVersionRequested* номер требуемой версии, а в младшем - номер подверсии. Аргумент *lpWSAData* должен указывать на структуру *WSADATA*, в которую при успешной инициализации будет занесена информация о производителе библиотеки. Никакого особенного интереса она не представляет, и прикладное приложение обычно ее игнорирует. При неудачной попытке инициализации, функция возвращает ненулевое значение. Далее выполняется ***первый шаг*** - создание объекта "сокет". Это осуществляется функцией *SOCKET **socket** (int domain, int type, int protocol).* Первый слева аргумент в списке указывает на используемый для взаимодействия набор протоколов (вид коммуникационной области). Для Интернет - приложений этот аргумент должен иметь значение *AF_INET*. В данном разделе будем считать, что используется стек протоколов *TCP/IP*, некоторые особенности работы со стеком *IPX/SPX* будут рассмотрены в заключительной части главы.

Следующий аргумент задает тип создаваемого сокета - *потоковый (SOCK_STREAM)* или *дейтаграммный (SOCK_DGRAM)* (еще существуют и сырые сокеты, но они практически не поддерживаются Windows – см. раздел "Сырые сокеты").

Последний аргумент уточняет, какой транспортный протокол следует использовать. Нулевое значение соответствует выбору по умолчанию, например: *TCP* - для потоковых сокетов и *UDP* - для дейтаграммных при использовании стека протоколов *TCP/IP*. В большинстве случаев нет большого смысла задавать протокол вручную и часто программисты полагаются на автоматический выбор по умолчанию.

Если функция завершилась успешно, она возвращает дескриптор сокета, в противном случае – значение *INVALID_SOCKET*.

Дальнейшие шаги зависят от того, является ли приложение сервером или клиентом. Ниже эти два случая рассматриваются раздельно.

**Клиент: *шаг второй*** - для установления соединения с удаленным узлом (сервером) потоковый сокет должен вызвать функцию
*int* ***connect*** *(SOCKET s, const struct sockaddr FAR* name, int namelen)*.
Дейтаграмные сокеты работают без установления соединения, поэтому обычно не обращаются к функции ***connect***.

***Примечание:*** в действительности за словом "обычно" стоит возможность использовать при желании один полезный прием: вызов connect позволяет дейтаграмному сокету обмениваться данными с узлом не только при помощи функций **sendto**, **recvfrom**, но и с использованием более удобных и компактных **send** и **recv**. Этот нюанс описан в Winsocket SDK и широко используется как программистами самой корпорации Microsoft, так и сторонними разработчикам. Таким образом, использование подобного подхода вполне безопасно.

Первый аргумент функции ***connect*** - дескриптор сокета, возвращенный функцией ***socket***; второй - указатель на структуру *"sockaddr"*, содержащую в себе адрес и порт удаленного узла (для случая стека *TCP/IP*), с которым устанавливается соединение. Структура *sockaddr* используется множеством функций, поэтому ее описание вынесено в отдельный раздел "Об адресах в *WinSock*". Последний аргумент сообщает функции размер структуры *sockaddr*.

После вызова ***connect***, система предпринимает попытку установить соединение с указанным узлом. Если по каким-то причинам это сделать не удастся (адрес задан неправильно, узел не существует или "висит", компьютер не находится в сети), функция возвратит ненулевое значение.

**Сервер: шаг третий** - прежде чем сервер сможет использовать сокет, он должен связать его с локальным адресом. Локальный и любой другой адрес узла Интернет, с позиций библиотеки сокетов, состоит из IP-адреса узла и номера порта. Если сервер имеет несколько IP-адресов, то сокет может быть связан как с каким-то конкретным одним, так и со всеми сразу (для этого вместо IP-адреса следует указать константу *INADDR_ANY*, равную нулю). Связывание осуществляется вызовом функции
*int* ***bind*** *(SOCKET s, const struct sockaddr FAR* name, int namelen)*.
В качестве первого слева аргумента передается дескриптор сокета, возвращенный функций ***socket***, за ним следуют указатель на структуру *sockaddr* и ее длина (см. раздел "Об адресах в *WinSock*").

Строго говоря, клиент также должен связывать сокет с локальным адресом перед его использованием, однако практически за него это делает функция ***connect***, ассоциируя сокет с одним из портов, динамически выбранных из диапазона 1024-5000. Сервер же должен настраиваться на за-

ранее определенный порт, например, если речь идет о хорошо известных стандартных сервисах, то 21 порт - для *FTP*, 23 для *telnet*, 25 для *SMTP*, 80 для Web, 110 для *POP3* и т.д. Поэтому серверу приходится осуществлять связывание "вручную".

При успешном выполнении функция возвращает нулевое значение и ненулевое в противном случае.

**Сервер: шаг четвертый** - выполнив связывание, потоковый сервер переходит в режим ожидания подключений, вызывая функцию
*int* ***listen*** *(SOCKET s, int backlog)*, где *s* - дескриптор сокета, а *backlog* - максимально допустимый размер очереди сообщений.

Размер очереди ограничивает количество одновременно обрабатываемых соединений, поэтому к его выбору следует подходить внимательно. Если очередь полностью заполнена, очередной клиент при попытке установить соединение с сервером получит отказ (в заголовке *TCP* пакета окажется установленным флаг *RST*). В то же время максимально разумное количество подключений должно определяться производительностью сервера, объемом его оперативной памяти и т.д.

Дейтаграммные серверы не вызывают функцию ***listen***, т.к. работают без установления соединения и сразу же после выполнения связывания могут вызывать **recvfrom** для чтения входящих сообщений, минуя четвертый и пятый шаги.

**Сервер: шаг пятый** - извлечение запросов на соединение из очереди осуществляется функцией
*SOCKET* ***accept*** *(SOCKET s, struct sockaddr FAR* addr, int FAR* addrlen)*, которая автоматически создает новый сокет, выполняет связывание и возвращает его дескриптор, а в структуру *sockaddr* заносит сведения о подключившемся клиенте (*IP*-адрес и порт). Если в момент вызова ***accept*** очередь пуста, функция не возвращает управление до тех пор, пока с сервером не будет установлено хотя бы одно соединение. В случае возникновения ошибки функция возвращает отрицательное значение.

Для параллельной работы с несколькими клиентами следует сразу же после извлечения запроса из очереди порождать новый поток (*Thread* - нить, процесс), передавая ему дескриптор созданного функцией ***accept*** сокета, затем вновь извлекать из очереди очередной запрос и т.д. В противном случае, пока не завершит работу один клиент, север не сможет обслуживать всех остальных.

**Все вместе** - после того как соединение установлено, потоковые сокеты могут обмениваться с удаленным узлом данными, вызывая функции
*int* ***send*** (*SOCKET s, const char FAR * buf, int len,int flags*) и
*int* ***recv*** (*SOCKET s, char FAR* buf, int len, int flags*) для посылки и приема данных, соответственно.

Функция ***send*** возвращает управление сразу же после ее выполнения, независимо от того, получила ли принимающая сторона ваши данные или

нет. При успешном завершении функция возвращает количество *передаваемых (не обязательно доставляемых в узел назначения!)* данных - т.е. успешное завершение еще не свидетельствует об успешной доставке! В общем-то, протокол *TCP* (на который опираются потоковые сокеты) гарантирует успешную доставку данных получателю, но лишь при условии, что соединение не будет преждевременно разорвано. Если связь прервется до окончания пересылки, данные останутся не переданными, но вызывающая программа не получит об этом никакого уведомления, а код ошибки возвращается лишь в том случае, если соединение было разорвано до вызова функции ***send***.

Что же касается функции ***recv***, то она возвращает управление только после того, как получит хотя бы один байт. Точнее говоря, она ожидает прихода целой *дейтаграммы*. Дейтаграмма - это совокупность одного или нескольких IP пакетов (при необходимости выполнения фрагментации), посланных вызовом ***send***. Упрощенно говоря, каждый вызов ***recv*** за один раз получает столько байтов, сколько их было послано функцией ***send***. При этом подразумевается, что функции ***recv*** предоставлен буфер достаточных размеров, в противном случае ее придется вызвать несколько раз. Однако, при всех последующих обращениях данные будут браться из локального системного буфера, а не приниматься из сети, т.к. *TCP/IP*-провайдер не может получить "кусочек" дейтаграммы, только всю ее целиком.

На работу обеих функций можно влиять с помощью *флагов*, передаваемых в одной переменной типа *int* третьим слева аргументом. Эта переменная может принимать одно из двух значений: *MSG_PEEK* и *MSG_OOB*.

Установка флага *MSG_PEEK* заставляет функцию ***recv*** просматривать данные вместо их чтения. Просмотр, в отличие от чтения, не уничтожает просматриваемые данные. Существует неверное мнение, что при установленном флаге *MSG_PEEK* функция ***recv*** не задерживает управления, если в локальном буфере нет данных, доступных для немедленного получения. Аналогично, встречается ошибочное утверждение о том, что якобы функция ***send*** с установленным флагом *MSG_PEEK* возвращает количество уже переданных байт (вызов ***send*** не блокирует управления). На самом деле функция ***send*** игнорирует этот флаг.

Флаг *MSG_OOB* предназначен для передачи и приема срочных (*Out Of Band*) данных. Срочные данные не имеют преимущества перед другими при пересылке по сети, а всего лишь позволяют оторвать клиента от нормальной обработки потока обычных данных и сообщить ему "срочную" информацию. Если данные передавались функцией ***send*** с установленным флагом *MSG_OOB*, для их чтения флаг *MSG_OOB* функции ***recv*** также должен быть установлен.

***Замечание:*** на практике настоятельно рекомендуется воздержаться от использования срочных данных в своих приложениях. В первую очередь, потому, что они совершенно необязательны - гораздо проще, надеж-

нее и изящнее вместо них создавать отдельное TCP-соединение. Во-вторых, по поводу их реализации до сих пор нет единого мнения, и интерпретации различных производителей очень сильно отличаются друг от друга. Так, разработчики до сих пор не пришли к окончательному соглашению по поводу того, куда должен быть наведен указатель срочности: на последний байт срочных данных или на байт, следующий за последним байтом срочных данных. В результате отправитель никогда не бывает уверен, что получатель сможет правильно интерпретировать его запрос.

В дополнение к вышеупомянутым, существует также флаг *MSG_DONTROUTE*, предписывающий передавать данные без маршрутизации, но он не поддерживаться *Winsock* и поэтому здесь не рассматривается.

Дейтаграммный сокет также может пользоваться функциями ***send*** и ***recv***, если предварительно вызовет ***connect*** (см. "Клиент: шаг третий"), но у него есть и свои, "персональные", функции:

*int* ***sendto*** (*SOCKET s, const char FAR * buf, int len, int flags, const struct sockaddr FAR * to, int tolen* )

и

*int* ***recvfrom*** (*SOCKET s, char FAR* buf, int len, int flags, struct sockaddr FAR* from, int FAR* fromlen*).

Они очень похожи на ***send*** и ***recv*** - разница лишь в том, что ***sendto*** и ***recvfrom*** требуют явного указания адреса узла, принимающего или передающего данные. Точнее говоря, вызов ***recvfrom*** не требует предварительного задания адреса передающего узла - функция принимает все пакеты, приходящие на заданный *UDP*-порт со всех *IP*-адресов и портов. Отвечать же отправителю следует на тот же самый адрес и порт, откуда пришло сообщение. Поскольку функция ***recvfrom*** заносит *IP*-адрес и номер порта клиента в структуру типа *sockaddr* после получения от него сообщения, программисту фактически ничего специально не нужно делать - только передать ***sendto*** тот же самый указатель на структуру *sockaddr*, который был ранее передан функции ***recvfrom***, получившей сообщение от клиента.

***Еще одна деталь*** - транспортный протокол *UDP*, на который опираются дейтаграммные сокеты, не гарантирует успешной доставки сообщений и эта задача ложиться на плечи самого разработчика. Решить ее можно, например, посылкой клиентом подтверждения об успешности получения данных. Правда, клиент тоже не может быть уверен, что подтверждение дойдет до сервера, а не потеряется где-нибудь в дороге. Подтверждать же получение подтверждения - бессмысленно, т.к. подобная рекурсия может принимать неограниченные масштабы. Лучше, стремясь иметь гарантии доставки, вообще не использовать дейтаграммные сокеты на ненадежных каналах.

Во всем остальном обе пары функций полностью идентичны и работают с теми же самыми флагами - *MSG_PEEK* и *MSG_OOB*. Все четыре

функции при возникновении ошибки возвращают значение *SOCKET_ERROR* (== -1).

***Примечание:*** в ОС *UNIX* с сокетами можно обращаться точно так же, как и с обычными файлами, в частности писать и читать в них функциями ***write*** и ***read***. ОС *Windows 3.1* не поддерживала такой возможности, поэтому при переносе приложений их *UNIX в Windows* все вызовы ***write*** и ***read*** должны были быть заменены на вызовы ***send*** и ***recv***, соответственно. Начиная с *Windows 95* (с установленным пакетом *Winsock 2.x*) это упущение исправлено, теперь дескрипторы сокетов можно передавать функциям ***ReadFile***, ***WriteFile***, ***DuplicateHandle*** и др.

**Шаг последний** - для закрытия соединения и уничтожения сокета предназначена функция *int* ***closesocket*** (*SOCKET s*), которая, в случае удачного завершения операции, возвращает нулевое значение.

Перед выходом из программы необходимо вызвать функцию *int* ***WSACleanup*** (*void*) для деинициализации библиотеки *WINSOCK* и освобождения используемых этим приложением ресурсов.

***Внимание:*** завершение процесса функцией **ExitProcess** автоматически не освобождает ресурсы сокетов!

***Примечание:*** существуют более сложные приемы закрытия соединения - протокол TCP позволяет выборочно закрывать соединение любой из сторон, оставляя другую сторону активной. Например, клиент может сообщить серверу, что не будет больше передавать ему никаких данных и закрывает соединение "клиент -> сервер", однако готов продолжать принимать от него данные до тех пор, пока сервер будет их посылать, т.е. хочет оставить соединение "сервер -> клиент" открытым. Для этого необходимо вызвать функцию int **shutdown** (SOCKET s ,int how ), передав в аргументе how одно из следующих значений: SD_RECEIVE для закрытия канала "сервер -> клиент", SD_SEND для закрытия канала "клиент -> сервер", и, наконец, SD_BOTH для закрытия обоих каналов.

Последний вариант выгодно отличается от **closesocket** "мягким" закрытием соединения - удаленному узлу будет послано уведомление о желании разорвать связь, но это желание не будет воплощено в действительность, пока тот узел не возвратит свое подтверждение. Таким образом, можно не волноваться, что соединение будет закрыто в самый неподходящий момент.

***Внимание:*** вызов **shutdown** не освобождает от необходимости закрытия сокета функцией **closesocket**!

### 5.7.3. Взаимосвязь Socket – функций

Для большей наглядности демонстрации взаимосвязи socket-функций друг с другом ниже приведена диаграмма вызовов, показывающая, в каком порядке должны следовать вызовы функций в зависимости от

типа сокетов (потоковый или дейтаграммный) и рода обработки запросов (клиент или сервер). Для случая дейтаграммных сокетов (рис. 5.8) обязательно необходимо иметь в виду то обстоятельство, что в силу дейтаграммного характера обмена по времени функция ***recvfrom*** должна вызываться на каждой абонентской стороне до соответствующего вызова ***sendto*** в приложении, работающем на противоположной стороне.

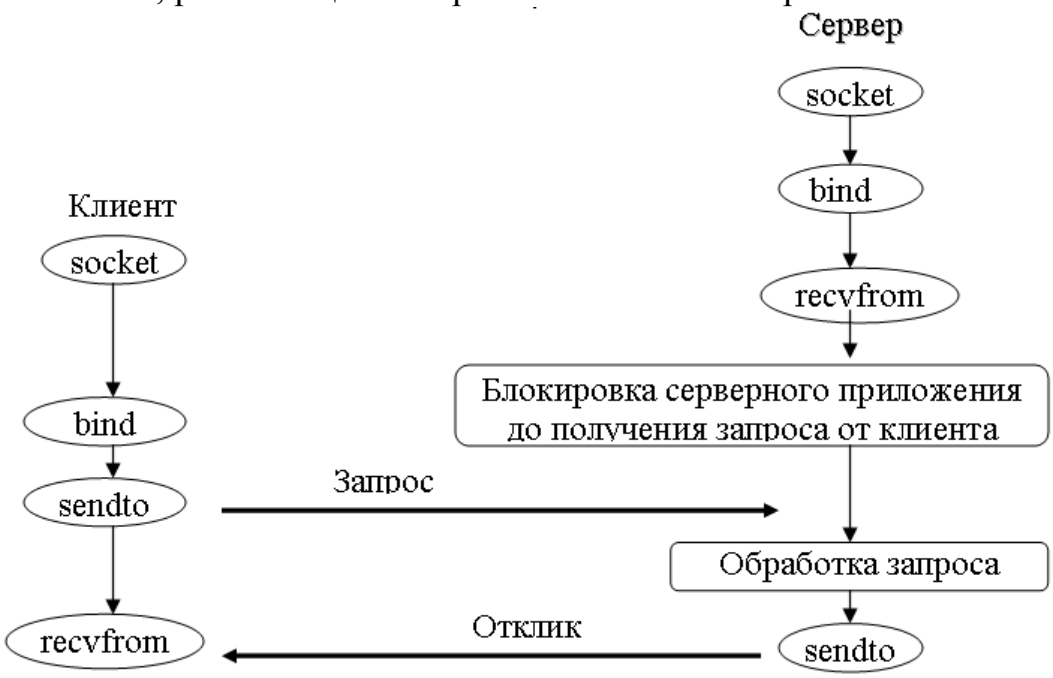



Рис. 5.8. Схема взаимодействия операторов winsock для процедур, не ориентированных на соединение

### 5.7.4. Об адресах в WinSock

С адресами на практике наблюдается некоторая путаница. Прежде всего, структура *sockaddr* определена так:

*struct sockaddr*
{ *u_short sa_family;* // формат адреса, связанный с семейством
// используемых протоколов, для *TCP/IP* должно быть *AF_INET*)
*char sa_data[14]; // IP*-адрес узла и порт
};

Однако, теперь она считается устаревшей, и в *Winsock 2.x* на смену ей введена структура *sockaddr_in*, определенная следующим образом:

*struct sockaddr_in*
{ *short sin_family; //* семейство протоколов (как правило, *AF_INET*)
*u_short sin_port;* // порт
*struct in_addr sin_addr;* // *IP*-адрес
*char sin_zero[8];* // хвост
};


Рис. 5.9. Схема взаимодействия операторов winsock для процедур, ориентированных на соединение

Радикальных изменений не произошло - замена беззнакового короткого целого на знаковое короткое целое для представления семейства протоколов ничего не дает. Но зато теперь адрес узла представлен в виде трех полей - *sin_port* (номер порта), *sin_addr* (*IP*-адрес узла) и "хвоста" из восьми нулевых байт, который остался от 14-символьного массива *sa_data*. Необходимость в нем определяется тем, что структура *sockaddr* не привязана именно к Интернет и может работать также и с другими сетями. Адреса же некоторых сетей требуют для своего представления гораздо больше четырех байт - вот по этой причине и приходится брать объем с запасом!

Структура *in_addr* определяется следующим образом:

```
struct in_addr
{ union {
 struct { u_char s_b1,s_b2,s_b3,s_b4; } S_un_b; // IP-адрес
 struct { u_short s_w1,s_w2; } S_un_w; // IP-адрес
 u_long S_addr; // IP-адрес
        } S_un;
};
```

Как видно, она состоит из одного *IP*-адреса, записанного в трех представлениях: четырехбайтовой последовательности (*S_un_b*), пары

двухбайтовых слов (*S_un_W*) и одного длинного целого (*S_addr*) – можно выбрать что удобно. Однако, во многих программах, технических руководствах и даже демонстрационных примерах, прилагающихся к *Winsock SDK*, встречается обращение к некоторому члену структуры *s_addr*, который явно не описан в *SDK*! Например, можно встретить такую строку:

*local.sin_addr.**s_addr** =(!interface)?INADDR_ANY:inet_addr(interface)*;

Реально, заглянув в файл "*winsock2.h*" можно обнаружить следующее: *#define s_addr S_un.S_addr*. Таким образом, это эквивалент *s_addr*, т.е. *IP*-адреса, записанного в виде длинного целого.

На практике можно с одинаковым успехом пользоваться как "устаревшей" *sockaddr*, так и "новой" *sockaddr_in*. Однако, поскольку прототипы остальных функций не изменились, при использовании *sockaddr_in* придется постоянно выполнять явные преобразования типа, например так:
*sockaddr_in dest_addr;*
***connect** (mysocket, (struct sockaddr*) &dest_addr, sizeof(dest_addr)*.

Для преобразования *IP*-адреса, записанного в виде символьной последовательности наподобие "127.0.0.1" в четырехбайтовую числовую последовательность предназначена функция *unsigned long* ***inet_addr*** (*const char FAR * cp* ). Она принимает указатель на символьную строку и в случае успешной операции преобразует ее в четырехбайтовый *IP*-адрес или -1, если это невозможно. Возвращенный функцией результат можно присвоить элементу структуры *sockaddr_in* следующим образом:

*struct sockaddr_in dest_addr*;
*dest_addr.sin_addr.S_addr=**inet_addr**("195.161.42.222"*).

При использовании структуры *sockaddr* это будет выглядеть так:

*struc sockaddr dest_addr;*
( (*unsigned int* *) (*&dest_addr.sa_data*[*0*]+*2*) )[*0*] =
*inet_addr*("195.161.42.222");

Попытка передать функции ***inet_addr*** доменное имя узла приводит к неудаче. Узнать *IP*- адрес какого-либо домена можно с помощью функции

*struct hostent FAR * **gethostbyname*** (*const char FAR * name*).

Функция обращается к службе *DNS* и возвращает свой ответ в структуре *hostent* или нуль, если *DNS* - сервер не смог определить *IP*-адрес данного домена.

Структура *hostent* выглядит следующим образом:

*struct hostent*
{
*char FAR * h_name*; // официальное имя узла
*char FAR * FAR * h_aliases*;
// альтернативные имена узла (массив строк)
*short h_addrtype*; // тип адреса
*short h_length*; // длина адреса (как правило AF_INET)

*char FAR * FAR * h_addr_list*; // список указателей на *IP*-адреса
// ноль - конец списка
};

Как и в случае с *in_addr*, во множестве программ и прилагаемых к *Winsock SDK* примерах активно используется недокументированное поле структуры *h_addr*. Например, можно встретить строки вида: *memcpy(&(server.sin_addr),hp->h_addr,hp->h_length)*.

Заглянув в "*winsock2.h*" можно найти, что оно обозначает:

*#define h_addr h_addr_list[0]*.

Дело в том, что с некоторыми доменными именами связано сразу несколько *IP*-адресов. В случае неработоспособности одного узла, клиент может попробовать подключится к другому или просто выбрать узел с наибольшей скоростью обмена. Но в приведенном примере клиент использует только первый *IP*-адрес в списке и игнорирует все остальные. Как правило, вышеприведенная форма будет работать, но все же будет лучше, если в программах будет учитываться возможность подключения к остальным *IP*-адресам при невозможности установить соединение с первым.

Функция ***gethostbyname*** ожидает на входе только доменные имена, а не цифровые *IP*-адреса. Между тем, правила "хорошего тона" требуют предоставления клиенту возможности задания, как доменных имен, так и цифровых *IP*-адресов.

Решение заключается в том, чтобы проанализировать переданную клиентом строку и если это *IP* адрес, то передать его функции ***inet_addr,*** в противном случае - ***gethostbyaddr***, полагая, что это доменное имя. Для того, чтобы отличать *IP*-адреса от доменных имен многие программисты используют простейший прием: если первый символ строки - цифра, это *IP*-адрес, иначе - имя домена. Однако, такой способ не совсем универсален - доменные имена могут начинаться с цифры, например, "666.*ru*", могут они и заканчиваться цифрой, например, к узлу "666.*ru*" члены поддомена "666" могут так и обращаться - "666". Особенно неприятно то, что (теоретически) могут существовать имена доменов, синтаксически неотличимые от *IP*-адресов! Поэтому лучше всего действовать так: передать введенную пользователем строку функции ***inet_addr***, если она возвращает ошибку, то вызвать ***gethostbyaddr***.

Для решения обратной задачи - определения доменного имени по *IP* адресу предусмотрена функция *struct HOSTENT FAR* **** gethostbyaddr*** (*const char FAR * addr, int len, int type*), которая во всем аналогична ***gethostbyname***, за тем исключением, что ее аргументом является не указатель на строку, содержащую имя, а указатель на четырехбайтовый *IP*-адрес. Еще два аргумента задают его длину и тип (соответственно, 4 и *AF_INET*).

Определение имени узла по его адресу бывает полезным, например, для серверов, желающих "по именам" знать своих клиентов.

Для преобразования *IP*-адреса, записанного в сетевом формате в символьную строку, предусмотрена функция *char FAR* * ***inet_ntoa*** (*struct in_addr*), которая принимает на вход структуру *in_addr*, а возвращает указатель на строку, если преобразование выполнено успешно и ноль в противном случае.

### 5.7.5. Сетевой порядок байт

Порядок следования младших и старших байт в разных микропроцессорах различается. Так, например, у микропроцессоров *Intel* младшие байты располагаются по меньшим адресам, а у микропроцессоров *Motorola* 68000 - наоборот. Естественно, это вызывает проблемы при межсетевом взаимодействии, поэтому был введен специальный сетевой порядок байт, предписывающий старший байт передавать первым (не так, как у *Intel*!).

Для преобразований чисел из сетевого формата в формат локального хоста (узла) и наоборот предусмотрено четыре функции: *u_short* ***ntohs***(*u_short netshort*)*; u_short* ***htons***(*u_short hostshort*)*; u_long* ***ntohl***(*u_long netlong*)*; u_long* ***htonl***(*u_long hostlong*). Первые две манипулируют короткими целыми (16-битными словами), а две последние - длинными (32-битными двойными словами)

Чтобы в них не запутаться, достаточно запомнить, что за буквой "*n*" скрывается сокращение "***n****etwork*", за "*h*" - "***h****ost*" (подразумевается локальный), "*s*" и "*l*" соответственно короткое (***s****hort*) и длинное (***l****ong*) беззнаковые целые, а "*to*" и обозначает преобразование. Например, "*htons*" расшифровывается так: "***H****ost* ***N****etwork* (***s****hort*)" т.е. преобразовать короткое целое из формата локального хоста в сетевой формат.

***Внимание:*** все значения, возвращенные socket-функциями, уже находятся в сетевом формате и "вручную" их преобразовывать **нельзя**(!) т.к. это преобразование исказит результат и приведет к неработоспособности.

Чаще всего к вызовам этих функций прибегают для преобразования номера порта согласно сетевому порядку. Например: *dest_addr.sin_port =* ***htons***(*110*).

### 5.7.6. Дополнительные возможности

Для дополнительной, "тонкой" настойки сокетов в рассматриваемой библиотеке предусмотрена функция
*int* ***setsockopt*** *(SOCKET s, int level, int optname, const char FAR * optval, int optlen)*. Первый слева аргумент - дескриптор сокета, который собираются настраивать, *level* - уровень настойки. С каждым уровнем связан свой набор опций. Всего определено два уровня - *SOL_SOCKET* и *IPPROTO_TCP*. Ограниченный объем данного раздела не позволяет перечислить все опции, поэтому ниже будут описаны только важнейшие из них. Сведения обо всех остальных можно почерпнуть из *Winsock SDK*.

Третий слева аргумент представляет собой указатель на переменную, содержащую значение опции. Ее размер варьируется в зависимости от рода опции и передается через четвертый слева аргумент.

*Уровень SOL_SOCKET: SO_RCVBUF* (*int*) - задает размер входного буфера для приема данных. К *TCP*-окну никакого отношения не имеет, поэтому, может безболезненно варьироваться в широких пределах. *SO_SNDBUF* (*int*) - задает размер входного буфера для передачи данных. Увеличение размера буферов на медленных каналах приводит к задержкам и снижает производительность.

*Уровень IPPROTO_TCP: TCP_NODELAY* (*BOOL*) - выключает *Алгоритм Нагла*. Алгоритм Нагла был разработан специально для прозрачного кэширования крохотных (*tiny*) пакетов (*тинограмм*). Когда один узел посылает другому несколько байт, к ним дописываются заголовки *TCP* и *IP*, которые в совокупности обычно занимают более 50 байт. Таким образом, при побайтовом обмене между узлами свыше 98% передаваемой по сети информации будет приходиться на служебные данные.

Алгоритм Нагла состоит в следующем: отправляется первый пакет и до тех пор, пока получатель не возвратит *TCP*-уведомление успешности доставки, в сеть не передается никаких пакетов, происходит накопление их на локальном узле, со сборкой в один большой пакет. Такая техника совершенно прозрачна для прикладных приложений и в то же время позволяет значительно оптимизировать трафик, но в некоторых (достаточно экзотических) случаях, когда требуется действительно побайтовый обмен, Алгоритм Нагла приходится отключать (по умолчанию он включен).

Для получения текущих значений опций сокета предусмотрена функция

*int* ***getsockopt*** *(SOCKET s, int level, int optname, char FAR* optval, int FAR* optlen)* ,

которая полностью аналогична предыдущей, за исключением того, что не устанавливает опции, а возвращает их значения.

### 5.7.7. Сырые сокеты

Помимо потоковых и дейтаграммных сокетов существуют, так называемые сырые (*RAW*) сокеты. Они предоставляют возможность "ручного" формирования *TCP/IP*-пакетов, равно как и полного доступа к содержимому заголовков полученных *TCP/IP*-пакетов, что необходимо многим сетевым сканерам, межсетевым экранам (*FireWall*'ам или брандмауэрам) и, разумеется, вредным хакерским программам, например, устанавливающим в поле "адрес отправителя" адрес самого получателя. Спецификация *Winsock 1.x* категорически не поддерживала сырые сокеты. В *Winsock 2.x* положение как будто было исправлено: по крайней мере, формально такая поддержка появилась и в *SDK* даже входил пример, демонстрирующий ис-

пользование сырых сокетов для реализации утилиты *ping*. Однако, попытки использования сырых сокетов для всех остальных целей проваливаются - система игнорирует сформированные "вручную" *IP*- (или *TCP*-) пакеты и создает их самостоятельно.

Документация объясняет, что для самостоятельной обработки заголовков пакетов, опция *IP_HDRINCL* должна быть установлена. Однако вызов ***setsockopt***(*my_sock,IPPROTO_IP, IP_HDRINCL, &oki, sizeof*(*oki*)) возвращает ошибку.

Таким образом, на прикладном уровне получить непосредственный доступ к заголовкам *TCP/IP* невозможно. Это препятствует переносу многих приложений из *UNIX* в *Windows*, более того - определенным образом ущемляет возможности самой *Windows*, не позволяя ей решать целый ряд задач, требующих поддержки сырых сокетов.

### 5.7.8. Программирование IPX/SPX сокетов

Когда говорят о сокетном программировании, чаще всего подразумевается стек *TCP/IP*. Однако, как указывалось выше, и с другими протокольными стеками работа может вестись с использованием *WinSock*. Рассмотрим пример протоколов *IPX/SPX*. Многие программные модули и системы используют их до сих пор [40].

Все начинается, как всегда, с инициализации библиотеки *WinSock*, обработка ошибок ниже отброшена для упрощения понимания кода:

```
#include <winsock.h> // прототипы функций библиотеки
#include <wsipx.h> // IPX/SPX структуры
#include <wsnwlink.h>
// IPX/SPX структуры и константы для NT платформ

void main ()
{
WSADATA wsaData;
// требуемая версия библиотеки winsock.dll
WORD wVersionRequested = MAKEWORD( 1, 1 );
// инициализация winsock.dll
WSAStartup( wVersionRequested, &wsaData );
// собственно работа с сокетами
……
WSACleanup( ); // в заключение сообщаем системе, что работа с
//winsock.dll завершена
}
```

Ниже приводятся конкретные примеры работы с сокетами для стека *IPX/SPX*:

```
WORD SPX_SOCKET = 0x5647; // номер сокета программы сервера
// номер сокета в стеке IPX/SPX служит для сетевой идентификации
// программы, аналогично номеру порта в стеке TCP/IP!
 char localNetNum[IPX_NET_SIZE]; //номер сети
char localNodeNum[IPX_NODE_SIZE]; //номер узла
// открытие сокета
SOCKET spx_skt = socket (PF_IPX, SOCK_STREAM, NSPROTO_SPX);
// проверим открылся ли
if (spx ==INVALID_SOCKET)
MessageBox (NULL, "Ошибка открытия сокета.", "Error", MB_OK);
// структура для адресных компонент сокета этой программы
sockaddr_ipx addr_ipx;
/* так выглядит структура sockaddr_ipx в WSIPX.H
typedef struct sockaddr_ipx {
u_short sa_family;
u_char sa_netnum[4];
u_char sa_nodenum[6]; unsigned short sa_socket;
} SOCKADDR_IPX, *PSOCKADDR_IPX, FAR *LPSOCKADDR_IPX;
*/

int sz = sizeof (addr_ipx);
memset (&addr_ipx, 0, sz); // обнулим её
addr_ipx.sa_family = AF_IPX; // тип протокола
addr_ipx.sa_socket = htons(SPX_SOCKET); // номер сокета
// привязываем его к номеру сокета
bind(spx_skt, (sockaddr*) &addr_ipx, sz);
// узнаем наш адрес
getsockname (spx_skt, (sockaddr*) &addr_ipx, &sz);
// наш номер сети
memcpy (localNetNum, addr_ipx.sa_netnum, IPX_NET_SIZE);
// наш номер узла
memcpy (localNodeNum, addr_ipx.sa_nodenum, IPX_NODE_SIZE);
```

В остальном работа с *SPX* идентична работе с *TCP* на базе библиотеки сокетов. Все написанное выше справедливо и для *IPX* сокетов, за исключением того, что последние не могут быть использованы для операции ***connect***. Открываются они следующим образом:

```
SOCKET ipx_skt = socket (PF_IPX, SOCK_DGRAM, NSPROTO_IPX);
```

Передача данных организуется так:

```
// структура для хранения удалённого адреса sockaddr_ipx addr_ipx;
// обнуляем её, хотя это не всегда нужно, но обычно не мешает memset
(&addr_ipx, 0, sizeof (addr_ipx));
```

*addr_ipx.sa_family = AF_IPX;* // тип протокола
*addr_ipx.sa_socket =* ***htons***(*SOCKET_NR*); // номер сокета
// удалённый номер сети
***memcpy*** (*addr_ipx.sa_netnum, remoteNetNum, IPX_NET_SIZE*);
// удалённый номер узла
***memcpy*** (*addr_ipx.sa_nodenum, remoteNodeNum, IPX_NODE_SIZE*);
*// буфер с будущим сообщением*
*char* buff = "*---Тестовая строка для передачи другому узлу---*";*
// собственно передача данных
***sendto*** (*ipx_skt, buff, strlen* (*buff*), *0,* (*sockaddr**)*&addr_ipx, sizeof* (*addr_ipx*));

Ниже приводятся некоторые дополнительные, полезные сведения по работе с протоколами семейства *IPX/SPX* при помощи библиотеки сокетов.

### 5.7.9. Применение широковещательных пакетов

Широковещательные пакеты могут быть использованы, например, в качестве средства поиска клиентом сервера, в случае, когда известен номер сокета (сокет *IPX* - аналог номера порта в *TCP/IP*!) нужного сервера, но не известен его сетевой адрес.

*sockaddr_ipx addr_ipx;*
// Содержимое информационного поля пакета
*char* broadcast_msg* = "---Кое-что для передачи в широковещательном режиме---";
*SOCKET ipx_skt =* ***socket***(*PF_IPX, SOCK_DGRAM, NSPROTO_IPX*);
*addr_ipx.sa_family = AF_IPX;*
// номер сокета, используемый при посылке *SAP* пакетов в *Netware*
*addr_ipx.sa_socket =* ***htons*** (0x0452);
// для широковещательных пакетов в пределах текущего сегмента
***memset*** (*addr_ipx.sa_netnum, 0, IPX_NET_SIZE*);
***memset*** (*addr_ipx.sa_nodenum, 0xff, IPX_NODE_SIZE*);
// устанавливается флаг для посылки широковещательных пакетов
*int set_broadcast* = 1;
***setsockopt*** (*ipx_skt, SOL_SOCKET, SO_BROADCAST,* (*char**)*& set_broadcast, sizeof* (*set_broadcast*));
//собственно сама широковещательная передача
***sendto*** (*ipx_skt, broadcast_msg, strlen* (*broadcast_msg*), *MSG_DONTROUTE,* (*sockaddr**)*&addr_ipx*, *sizeof* (*addr_ipx*));

### 5.7.10. Установка и изменение поля *DataStreamType* в заголовке *SPX* пакета

Поле *DataStreamType,* в принципе, может быть использовано в собственных целях, например, для искусственной сегментации при необходи-

мости обеспечить совместимость разных реализаций протокола. К примеру, некоторые реализации протокола для ДОС поддерживают максимальную длину пакета в 512 байт, они и используют *DataStreamType*, чтобы указать последнюю порцию данных.

Поле устанавливается следующим образом:

```
// Можно использовать любое значение между 0 - 0xfd
// следующие два значения зарезервированы:
// #define SPX_HANG_UP 0xFE
// #define SPX_HANG_UP_ACK 0xFF
int stream_type = 0x05;
setsockopt (spx_skt, NSPROTO_IPX, IPX_DSTYPE, (char*)&stream_type,
sizeof(stream_type));
```

Подобную установку надо делать перед каждым вызовом ***send***. Обеспечивается нормальная работа при посылке данных ДОС клиенту.

Другие специфические расширения для данных протоколов, применяемые с ***getsockopt/setsockopt***, можно найти в файле *wsnwlink.h*, но необходимо иметь в виду, что расширения для *NT* платформ могут не работать при других реализациях рассмотренных протоколов.

## 6. Ретрансляция кадров (Frame Replay). Характеристики протокола информационного обмена и интерфейса «пользователь-сеть»

Ретрансляция кадров (FR) - метод синхронной доставки сообщений в сетях передачи данных с коммутацией пакетов. Первоначально разработка стандарта FR была ориентирована на цифровые сети интегрального обслуживания (ЦСИО; ISDN - Integrated Services Digital Networks), однако позже стало ясно, что FR применима в качестве коммуникационного стандарта и в других широкомасштабных СПД. К числу ее достоинств прежде всего необходимо отнести: малое время задержки, простой формат кадров, содержащих минимум управляющей информации, независимость от протоколов верхних уровней ЭМВОС и др.

Разработкой и исследованием стандартов FR в настоящее время занимаются четыре международные организации: Форум Ретрансляции Кадров (FRF - Frame Relay Forum, г. Фостер-Сити, штат Калифорния, США; этот международный консорциум включает 17 компаний, среди которых 3com, Northern Telecom, Digital Equipment Corporation, Cisco Systems, Stratacom, Netrix Corporation, Timeplex, Newbridge Networks, Zilog и др.), ANSI (American National Standards Institute - Американский национальный институт по стандартизации), Международный союз электросвязи (ITU-T) и Группа Четырех - G4 (Digital Equipment Corporation, Northern Telecom,

Stratacom и Cisco Systems). Оригинальный стандарт G4 (1990 г.), действительно положивший начало промышленному применению FR, был основан на проекте стандартов ANSI и имел небольшие отличия от последнего. Стандарты FRF и ITU-T, появившиеся несколько позже, идентичны стандартам ANSI.

Любой международный стандарт всегда имеет (и всегда будет иметь) огромный набор прикладных реализаций в различных СПД, что, как правило, приводит к созданию многими фирмами-производителями аппаратно-программных средств, которые зачастую несовместимы. Многими международными организациями предпринимались попытки преодоления ситуации. Результатом одной такой попытки, предпринятой FRF, явился проект, согласовывающий набор стандартов ANSI, обязательный для выполнения членами FRF и реализации большинством фирм-производителей. В январе 1992 года это соглашение было доработано Техническим комитетом FRF и подтверждено собранием членов FRF.

Принятый FRF проект рассматривает только стандарты для постоянных виртуальных каналов (ПВК) и интерфейса "пользователь - сеть" (ИПС). В этот проект не вошли стандарты для коммутируемых виртуальных каналов (КВК) и интерфейса межсетевого взаимодействия. Однако работы по этим направлениям продолжаются, и впоследствии их результаты найдут свое отражение в новых стандартах FR. Кроме того, проект FRF не рассматривает стандарты физических интерфейсов, поэтому при создании сетей FR допускаются различные физические интерфейсы, среди которых V.35, G.703, X.21 и др.

### *6.1. Логическая характеристика протокола FR*

FR является бит-ориентированным синхронным протоколом, использующим "кадр" в качестве основного информационного элемента, и в этом смысле очень похож на протокол HDLC. Однако FR не обеспечивает все функции протокола HDLC, и поэтому многие из элементов кадра HDLC исключены из основного формата кадра FR (в кадре FR адресное поле и поле управления HDLC совмещены в одно адресное поле). Структура кадра FR представлена в табл. 6.1 и включает необходимые поля.

Таблица 6.1

Структура кадра

| HDLC | | | | | | | |
|---|---|---|---|---|---|---|---|
| Флаг | Адрес | Управление | Информация | | Проверка | | Флаг |
| FR | | | | | | | |
| Флаг | Адрес | | Информация | | Проверка | | Флаг |
| Адрес | | CR | EA | Адрес | FECN | BECN | DE | EA |
| 8...3 | | 2 | 1 | 8...5 | 4 | 3 | 2 | 1 |
| | | | "0" | | | | | "1" |

*Флаг.* Все кадры начинаются и заканчиваются комбинацией флаг - *"01111110".* Одна и та же комбинация *флаг* может использоваться как закрывающая один кадр и открывающая следующий. С целью предотвращения имитации комбинации флаг при передаче кадра проверяется все его содержание между двумя флагами и вставляется "0"-й бит после всех последовательностей из пяти идущих подряд битов "1". На приемном конце "0"-е биты отбрасываются. Данная процедура (bit stuffing - "прозрачность") обязательна при формировании любого кадра FR.

*Адрес.* Поле адреса в пределах кадра FR состоит из 6 битов первого октета и 4 битов второго октета (кодировка поля адреса представлена в табл. 6.2. Эти 10 битов определяют абонентский адрес в сети FR (идентификатор канала передачи данных: *Data Link Connection Identifier – DLCI).* Стандарты ANSI и ITU-T допускают размер заголовка кадра до 4 октетов.

Таблица 6.2

Кодировка 10-битового DLCI в поле "адрес" кадра FR

| Октеты поля "Адреса" | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
|---|---|---|---|---|---|---|---|---|
| Первый | $2^9$ | $2^8$ | $2^7$ | $2^6$ | $2^5$ | $2^4$ | | |
| Второй | $2^3$ | $2^2$ | $2^1$ | $2^0$ | | | | |

*Бит "опрос/финал" (Command/Response - CR),* Этот бит протоколом FR не используется, но может применяться в пользовательских приложениях и "прозрачно пропускается" аппаратно-программными средствами сети FR.

*Бит расширения адреса (Extended Address - EA).* DLCI содержится в 10 битах, входящих в два октета поля адреса. Однако возможно расширение адресного поля (заголовка) на целое число дополнительных октетов с целью указания адреса, состоящего из более чем 10 битов. Бит *EA* устанавливается в конце каждого октета заголовка, и если он имеет значение "1", то это означает, что данный октет в заголовке последний. Стандарт FRF рекомендует использовать заголовок, состоящий из двух октетов. В этом случае бит *EA* в первом октете будет установлен в "0", а во втором - в "1".

*Бит уведомления (сигнализации) приемника о явной перегрузке (Forward Explicit Congestion Notification - FECN).* Этот бит устанавливается в "1" АКД сети для уведомления получателя сообщения о том, что произошла перегрузка в направлении передачи кадра, содержащего этот признак. Бит *FECN* устанавливается АКД сети FR (а не передающим ООД пользователя) и не обязателен для терминалов абонентов (рис. 6.1).

*Бит уведомления (сигнализации) источника о явной перегрузке (Backward Explicit Congestion Notification - BECN).* Этот бит устанавливается в "1" АКД сети для уведомления источника сообщения о том, что, произошла перегрузка в обратном направлении относительно направления передачи кадра, содержащего этот признак. Бит *BECN* устанавливается

АКД сети FR (а не передающим ООД пользователя) и не обязателен для терминалов абонентов (рис. 6.1).

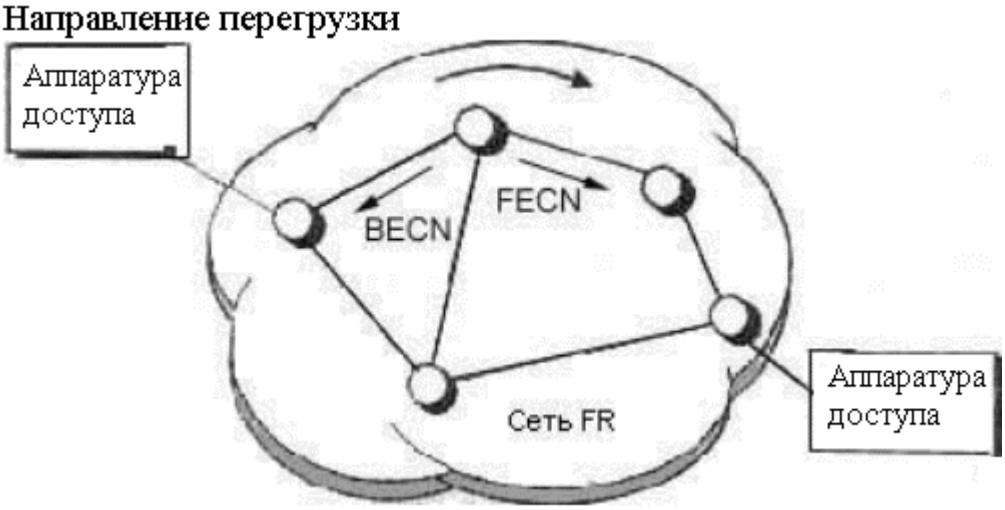



Рис. 6.1. Установка битов перегрузки

*Бит разрешения сброса (Discard Eligibility - DE).* Этот бит устанавливается в "1" в случае явной перегрузки входного графика и указывает на то, что данный кадр может быть уничтожен в первую очередь по отношению к другим кадрам, не имеющим данного признака. Бит *DE* может 6ыть установлен в "1" либо АКД сети FR, либо ООД пользователя (то есть пользователю предоставлено право выбирать, какими кадрами "он может пожертвовать"), при этом повторная установка не допускается. Это предусмотрено для того, чтобы узлы коммутации сети FR могли уничтожать при перегрузках не только кадры с установленным битом *DE* (рис. 6.2).

*Информационное поле.* Информационное поле содержит данные пользователя и состоит из целого числа октетов. Максимальный размер для этого поля определен стандартом FRF и составляет 1600 октетов (минимальный размер - 1 октет). Содержание информационного поля пользователя передается неизменным через сети FR и прозрачно для протокола FR.


Рис. 6.2. Гарантированная скорость передачи информации и кадры, которые могут быть уничтожены

*Проверочная последовательность кадра (Frame Check Sequence - FCS).* Проверочная последовательность кадра используется для обнаружения возможных ошибок при его передаче и состоит из двух октетов. Данная последовательность *FCS* формируется аналогично циклическому коду HDLC.

Все указанные выше поля должны присутствовать в каждом кадре FR, который передается между двумя оконечными пользовательскими системами.

Протокол FR не предусматривает передачу сигнальных сообщений: нет командных (или супервизорных) кадров, как в HDLC. Для передачи служебной информации используется специально выделенный канал сигнализации (ОКС), внутри которого передаются супервизорные кадры.

Другое важное различие между FR и HDLC - отсутствие любой нумерации последовательности передаваемых (принимаемых) кадров. Это является результатом того, что протокол FR не имеет никаких механизмов для подтверждения правильно принятых кадров.

### *6.2. Процедурная характеристика протокола FR*

Протокол FR весьма прост по сравнению с HDLC и включает небольшой свод правил и процедур организации информационного обмена. Основной процедурой протокола является то, что если кадр получен без искажений, то должен быть направлен далее по назначению соответствующим маршрутом. Если возникают проблемы, связанные с перегрузкой в пределах сети, то узлам сети FR разрешается отказаться от каких-либо кадров, чтобы снизить остроту проблемы.

Узлу связи сети FR разрешено уничтожать искаженные кадры, не уведомляя при этом ООД пользователя. Искаженным считается кадр, у которого:

- нет корректного ограничения флагами;
- в составе менее чем пять октетов между флагами;
- в составе нет целого числа октетов после удаления битов прозрачности;
- ошибка FCS;
- поле адреса искажено;
- содержит несуществующий DLCI;
- превышен допустимый максимальный размер.

Однако в некоторых вариантах стандартов FR возможна принудительная обработка кадров, которые превышают допустимый максимальный размер. Обязательные процедуры FR:

- "межкадровое заполнение" флагами в моменты отсутствия данных для передачи;

• резервирование одного DLCI для интерфейса локального управления и сигнализации (Local Management Interface - LMI);

• содержание поля данных пользователя в любом кадре не должно подвергаться какой-либо обработке со стороны сети FR, за исключением данных LMI.

Управление доступом к сети FR возлагается на интерфейс локального управления. Именно LMI реализует ИПС. В качестве аппаратно-программных средств (рис. 6.3), обеспечивающих доступ в сеть FR, используются FR-адаптеры (сборщик/разборщик кадров FR: FR assembler/disassembler - FRAD).


Рис. 6.3. Организация доступа к сети FR с помощью FRAD

В сетях FR для высокой эффективности использования пропускной способности физических линий и каналов связи и исключения перегрузки узлов связи и всей сети FR используется метод статистического мультиплексирования кадров, который заключается в постоянном "наблюдении" (со стороны сети FR) за потоком заявок от пользователей на передачу сообщений, текущим состоянием загрузки сети (линий, каналов и узлов связи), "перераспределении" свободного (и высвобождающегося) ресурса пропускной способности между реальными потребностями в ней абонентов и предоставлении последним каналов информационного обмена в соответствии с требуемыми параметрами. Данный метод обеспечивает синхронный ввод сообщений пользователей в высокоскоростной канал связи на основе предварительных соглашений между ООД пользователя и АКД сети FR, включающих:

• максимальный размер поля информации в кадре FR (в октетах);

• пропускную способность порта, посредством которого ООД абонента подключается к сети FR (port speed);

• гарантированную скорость передачи данных (Committed Information Rate - CIR), обеспечиваемую сетью FR абоненту постоянно и с требуемым качеством;

- гарантированный объем переданной информации (Committed Burst Size - Be) обеспечивается сетью FR с требуемым качеством. С использованием первых трех согласованных параметров может быть вычислено гарантированное время передачи информации с требуемым качеством (Committed Rate Measurement Interval - Te);
- дополнительный объем переданной информации (Excess Burst Size - Be) может обеспечиваться сетью FR, но с худшим качеством.

Использование предварительных соглашений происходит следующим образом:

1. Абонент выбирает (и оплачивает) пропускную способность порта и гарантированную скорость передачи данных для ПВК;

2. Узел доступа к сети FR (FRAD) измеряет "реальный потребляемый абонентом" ресурс пропускной способности канала связи;

3. Если этот ресурс, пересчитанный в реальную скорость передачи информации, не превышает CIR, то кадры передаются без изменений; если превышает, но не более, чем пропускная способность порта, то в кадрах FRAD бит *DE* устанавливается в "1", который разрешает их удалять при возникновении перегрузки (это право имеет и абонент, который сам решает, какие кадры для него наименее важны); если превышает пропускную способность порта, то кадры уничтожаются независимо от каких-либо условий.

Предварительные соглашения могут быть использованы абонентом следующим оригинальным способом. Некоторые администраторы сетей (поставщики услуг) предлагают абонентам значительные скидки при передаче кадров с битом DE, установленным в "1". Поэтому при наличии в сети значительного запаса пропускной способности абонент может сэкономить свои средства за счет установки CIR="0". В этом случае во всех передаваемых кадрах бит DE устанавливается в "1".

Биты уведомления о явной перегрузке используются для передачи информации пользователям о состоянии сети и возникшей перегрузке. Бит *FECN* устанавливается в тех кадрах, которые передаются в направлении перегрузки. Когда ООД абонента получает *FECN,* это указывает на то, что кадр получен из области перегрузки в пределах сети. Бит *BECN* используется в обратном направлении от перегрузки.

### *6.3. Адресация в сетях FR*

Адреса DLCI в кадре FR предназначены только для идентификации логических каналов между пользователями и сетью, другими словами, они имеют только локальное значение и не касаются какой-либо общесетевой адресации. Любые информационные кадры, передаваемые через конкретный логический канал, имеют одинаковый DLCI независимо от направления: от абонента или к абоненту.

В связи с тем, что DLCI носит исключительно локальный характер сеть FR обязана точно интерпретировать межабонентское соединение и при этом может использовать различные сетевые адреса внутри сети FR. Для различных интерфейсов значение DLCI может пользоваться многократно.

Стандарт FR (ANSI, ITU-T) распределяет адреса DLCI (двухоктетные) между пользователями и сетью следующим образом:

- *0* - для канала локального управления (LMI);
- *1…15* - зарезервировано для дальнейшего использования;
- *6...991* - используются абонентами для нумерации ПВК и КВК;
- *992...1007* - используются сетевой транспортной службой для внутрисетевых соединений;
- *1008.. .1022* - зарезервировано для дальнейшего использования;
- *1023* - для управления канальным уровнем (используется в кадрах, которые "переносят" сквозные сообщения управления интерфейсом, связывающим протоколы вышележащих уровней).

Таким образом, для оконечного устройства пользователя в любом интерфейсе FR отводится только 976 адресов DLCI.

### *6.4. Общая характеристика LMI*

Протокол FR обеспечивает в первую очередь высокоскоростную транспортировку данных и в соответствии с этим предоставляет абоненту требуемый ресурс пропускной способности сети (каналов и линий связи). Так как протокол FR разработан только для ПВК, то нет необходимости в процедурах установления и разъединения соединений. Вместе с тем он не предусматривает процедур управления потоком и исправления ошибок. Таким образом, протокол FR представляет собой базовый механизм передачи данных.

Однако протокол FR не предполагает никакого механизма локального управления и контроля за состоянием связи. По этой причине был разработан стандарт интерфейса локального управления (LMI), который предназначен в первую очередь для предоставления пользователю информации о состоянии и конфигурации ПВК. LMI применяется только в оконечном аппаратно-программном оборудовании пользователя (рис. 6.4) и выполняет следующие функции:

- уведомление абонента о подключении, наличии и отключении ПВК;
- уведомление абонента о готовности заранее сконфигурированного ПВК;
- последовательный опрос сети FR в целях поддержании соединения в течение длительного времени.


Рис. 6.4. Применение интерфейса локального управления (LMI)

Интерфейс LMI является необязательной частью стандарта FR. Однако при разработке новых стандартов FR интерфейс локального управления является неотъемлемой их частью, и поэтому, международные организации и фирмы-производители проводят активную работу по скорейшему принятию единого стандарта LMI, обязательного к выполнению всеми разработчиками аппаратно-программных средств и администрациями сетей FR. Единый стандарт LMI становится наиболее актуальным при переходе сетей FR на КВК.

### *6.5. Логическая характеристика LMI*

Интерфейс LMI согласован с базовым стандартом FR относительно логической и процедурной характеристик последнего. Различие состоит в расширении заголовка кадра FR с целью размещения дополнительных кодированных полей стандарта LMI, поэтому такой расширенный кадр FR в дальнейшем будем называть кадром LMI. Базовый формат кадра LMI представлен в табл. 6.3 и включает:

- *Заголовок.* Это - стандартный заголовок FR, в котором адрес DLCI всегда имеет значение "0". Это значение адреса показывает, что это - кадр LMI. (В стандарте LMI, разработанным G4, для кадра LMI используется *DLCI=1023).*
- *Индикатор ненумерованного кадра.* Это поле всегда кодируется как *"00000011".* Такая кодировка обеспечивает процедурную и логическую совместимость с цифровыми сетями интегрального обслуживания (ISDN).
- *Определитель протокола.* Данный октет всегда устанавливается в "*'00001000"* и тем самым обеспечивается процедурная и логическая совместимость с ISDN.
- *Вызываемый номер.* Данный октет зарезервирован для использования при организации КВК. В кадрах LMI при организации ПВК этот октет кодируется как *"00000000".*

- *Тип сообщения.* Этот октет предназначен для идентификации типа управляющего сообщения, передаваемого через интерфейс LMI. В настоящее время стандартизированы три типа управляющих сообщений: *"Запрос установления соединения", "Запрос разъединения"* и *"Смешанное сообщение".* Первые два типа относятся к КВК, а последний - к ПВК.

Таблица 6.3

Базовый формат кадра LMI

| Октеты | Биты | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | Назначение |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | Флаг |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Заголовок: DLCI=0, CR=0, |
| 3 | 0 | 0 | 0 | 0 | | | | 1 | DE=0, FECN=0, BECN=0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | Индикатор ненумерованного кадра |
| 5 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | Определитель протокола |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Вызываемый номер (только для КВК) |
| 7 | | | | | | | | | Тип сообщения |
| 8 | | | | | | | | | Первый информационный элемент |
| … | | | | | | | | | |
| 13 | | | | | | | | | |
| 14 | | | | | | | | | Второй информационный элемент |
| … | … | | | | | | | | … |
| | | | | | | | | | N-й информационный элемент |
| | | | | | | | | | |
| | | | | | | | | | Проверочная последовательность |
| | | | | | | | | | |
| | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | Флаг |

Таблица 6.4

Кодирование поля "Тип сообщения" кадра LMI для смешанных сообщений

| Тип сообщения | Биты | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| Смешанные сообщения | 0 | 1 | 1 | 1 | - | - | - | - |
| Состояние | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| Запрос состояния | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |

Таблица 6.5

Кодирование поля "Информационный элемент" кадра LMI

| Октеты | Биты | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| 1 | 0 | Идентификатор информационного элемента | | | | | | |
| 2 | Размер содержания информационного элемента (в октетах) | | | | | | | |
| 3 | Содержание информационного элемента | | | | | | | |
| 4 ... | … | | | | | | | |

- В этом октете восьмой бит всегда устанавливается в "0". Биты 7...5 устанавливаются в "111", указывая на то, что это смешанное сообщение. Кодирование остальных битов представлено в табл. 6.4.
- *Информационные элементы.* Для информационных элементов кадра LMI отводится один или больше октетов в пределах последнего, то есть информационные элементы имеют переменную длину. Формат поля элемента информации представлен в табл. 6.5.

### *6.6. Процедурная характеристика LMI*

LMI предусматривает три стратегии локального управления:
- синхронное симплексное управление (ССУ);
- синхронное дуплексное управление (СДУ);
- асинхронное управление (АУ).

#### 6.6.1. Синхронное симплексное управление

Для осуществления ССУ используются два типа сообщений:
- *"Запрос состояния"* (STATUS ENQUIRY);
- *"Состояние"* (STATUS).

С помощью этих сообщений LMI "проводит":
- проверку целостности соединения;
- уведомление о включении или выключении ПВК;
- уведомление о готовности ПВК.

Процедура ССУ заключается в периодическом "опросе" ООД пользователя через интерфейс LMI (процедура "биения") состояния сети. Через определенный временной интервал ООД пользователя посылает в сеть сообщение *"Запрос состояния"* (интервал опроса имеет международное обозначение - T391) с целью подтверждения целостности связи, на что АКД сети отвечает сообщением *"Состояние",* содержащим требуемый элемент информации о целостности связи.

Интерфейсом LMI ведется подсчет числа опросов. После определенного числа переданных сообщений *"Запрос состояния"* (этот интервал имеет международное обозначение - N391) ООД абонента запрашивает у сети информацию о так называемом полном состоянии, используя также сообщение *"Запрос состояния".* АКД сети на этот запрос отвечает сообщением *"Состояние", в* котором присутствуют информационные элементы для каждого ПВК (если ООД пользователя имеет несколько портов). В том случае, если информационный элемент для какого-либо ПВК отсутствует в этом ответе, терминал пользователя воспринимает это как отсутствие ПВК в интерфейсе "пользователь - сеть".

Формат сообщения *"Запрос состояния"* представлен в табл. 6.6 (версия ITU-T, в стандарте ANSI вводится дополнительный октет между 7-м и

8-м октетами, который имеет вид " *10010101",* а 8-й и 11-й октеты соответственно - *"00000001", "00011001").* Это сообщение всегда содержит два информационных элемента:

- информационный элемент о типе сообщения;
- информационный элемент проверки целостности связи.

Таблица 6.6

Формат кадра LMI "Запрос состояния"

| Октеты | Биты | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | Назначение |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | Флаг |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Заголовок: DLCI=0, CR=0, |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | DE=0, FECN=0, BECN=0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | Индикатор ненумерованного кадра |
| 5 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | Определитель протокола |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Вызываемый номер (только для КВК) |
| 7 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | Сообщение "Запрос состояния" |
| 8 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | Информационный элемент о типе сообщения |
| 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | |
| 10 | Тип сообщения | | | | | | | | |
| 11 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | Информационный элемент о целостности связи |
| 12 | 2 | | | | | | | | |
| 13 | Номер передаваемого кадра | | | | | | | | |
| 14 | Номер принятого кадра | | | | | | | | |
| 15 | | | | | | | | | Проверочная |
| 16 | | | | | | | | | последовательность |
| 17 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | Флаг |

Элемент информации типа сообщения указывает, какой тип сообщения запрашивается у сети (всего их может быть три, табл. 6.7). *"Запрос о полном состоянии"* посылается с целью получения информации о всех ПВК, сконфигурированных через интерфейс. *"Запрос о целостности соединения"* предназначен для промежуточного опроса о порядке следования кадров, "проходящих" через интерфейс, с целью контроля возможных потерянных кадров. *"Запрос о состоянии отдельного асинхронного ПВК"* посылается для получения информации об отдельном ПВК.

Порядковые номера могут принимать значения (в двоичной форме) от 1 до 255. Порядковый номер *"0"* используется только для начального порядкового номера принятого кадра в начальном сообщении *"Запрос ее стояния".*

Таблица 6.7

Кодирование поля "Тип сообщения" информационного элемента кадра LMI

| Тип сообщения | Биты | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| Полное состояние | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Целостность соединения | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Состояние одиночного асинхронного ПВК | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

Таблица 6.8

Формат информационного элемента о целостности соединения (ITU-T)

| Октеты | Биты | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | Назначение |
| 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | Идентификатор информационного элемента о целостности соединения |
| 2 | | | | | | | | | Длина информационного элемента в октетах (2) |
| 3 | | | | | | | | | Номер передаваемого кадра |
| 4 | | | | | | | | | Номер принятого кадра |

Таблица 6.9

Формат информационного элемента о состоянии ПВК

| Октеты | Биты | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | Назначение |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | Идентификатор информационного элемента о состоянии ПВК |
| 2 | | | | | | | | | Длина информационного элемента в октетах (3) |
| 3 | 0 | 0 | 0 | 0 | -- | -- | -- | -- | 8 - бит расширения; 7 – резерв 6...1 -DLC1 |
| | 0 | 0 | - | - | | | | | |
| 4 | 1 | - | - | - | - | 0 | 0 | 0 | 8 - бит расширения; 7...4 - DLC1; 3...1 - резерв |
| 5 | 1 | 0 | 0 | 0 | - | 0 | - | 0 | 8 - бит расширения; 7...5 - резерв; 4 - новый ПВК; 2 – активный ПВК |

Главное назначение проверки целостности соединения - гарантировать ООД пользователя и АКД сети стабильность и надежность физической и логической связи между ними. Эта процедура основана на генерации последовательности специальных пронумерованных кадров и проверке корректности этой последовательности с помощью специального процесса. ООД абонента каждый раз (с определенной периодичностью) посы-

лает сообщение *"Запрос состояния"* (табл. 6.8), в котором устанавливаются:

- порядковый номер передаваемого кадра (этот номер увеличивается на единицу по мере передачи таких кадров);
- порядковый номер последнего кадра, полученного от сети (в первом сообщении *"Запрос состояния"* имеет значение "0").

На полученное сообщение *"Запрос состояния"* АКД сети передает ООД абонента сообщение *"Состояние",* в котором устанавливаются:

- порядковый номер передаваемого кадра (этот номер увеличивается на единицу по мере передачи таких кадров);
- порядковый номер последнего кадра, полученного от ООД пользователя.

Формат информационного элемента о состоянии ПВК в кадре LMI *"Состояние"* представлен в табл. 6.9 (версия ITU-T, в стандарте ANSI первый октет имеет вид *"00000111").* Этот элемент содержит DLCI (10 бит) имеющегося ПВК и два бита-индикатора, которые указывают, будет ли данный ПВК "новым" и/или "активным". В случае если АКД сети создает новый ПВК, то бит "новый ПВК" устанавливается в "1". АКД сети передает такие кадры ООД абонента до тех пор, пока не получит от него сообщение *"Запрос состояния",* содержащее приемный порядковый номер, равный переданному сетевому порядковому номеру (последний переданный номер в сообщении *"Состояние").* По своей сути эта процедура похожа на синхронизацию счетчиков, смысл которой заключается в информировании ООД пользователя о наличии нового ПВК, а по ее окончании АКД сети устанавливает бит "новый ПВК" в сообщении *"Состояние"* в *"0".*

Однако признак нового ПВК не разрешает оконечному оборудованию пользователя начать передачу сообщения на данном ПВК. "Сигналом" начала передачи является бит "активный ПВК", установленный АКД сети в "1". Он устанавливается АКД сети только тогда, когда последняя непосредственно "убедилась" в том, что найден путь для доставки сообщения к месту назначения (другими словами, когда ПВК полностью подключен). Время подключения ПВК зависит от конкретной сети и реализации протокола.

Процедура оповещения пользователя о состоянии ПВК не является процессом, происходящим в реальном масштабе времени, так как изменения в сети не доводятся до пользователя немедленно. Поэтому возможны проблемы, вызванные некорректным выбором интервала времени, в течение которого ООД пользователя информируется о готовности ПВК, а именно:

Если ПВК становится доступным, то из-за некорректного выбора интервала времени для передачи кадра LMI с информацией о полном состоянии ПВК уведомленным (относительно активного состояния ПВК) может оказаться только один участник информационного обмена. ООД этого або-

нента начинает через ПВК передавать кадры данных, которые направляются к месту назначения до того, как оно получит сообщение *"Состояние".* в котором бит "активный ПВК" установлен сетью в "1".

Если, наоборот, ПВК становится недоступным, то по той же причине ООД абонента может не знать о неактивном состоянии ПВК, но по-прежнему передавать кадры данных в сеть через этот ПВК.

Оба эти случая не нашли своего отражения в стандарте FR. Однако наилучшим выходом из таких ситуаций будет размещение данных в буфере сети до тех пор, пока ПВК не станет активным.

### 6.6.2. Синхронное дуплексное управление

При ССУ, одностороннем по своей природе, ответственность за генерацию сообщения *"Запрос состояния"* лежит полностью на ООД пользователя, а за генерацию сообщения *"Состояние"* - на АКД сети. Такая процедура приемлема для многих приложений, однако предпочтительнее, что5а эта процедура была сбалансирована между двумя сторонами интерфейса LMI и каждая из сторон могла поддерживать параметры противоположной стороны, а также требуемый коэффициент готовности.

СДУ - необязательная часть стандарта FR и может использоваться только при обоюдном соглашении сторон (абонент - сеть). Очевидно СДУ будет наиболее актуальным при взаимодействии коммутаторов FR или различных сетей FR (через интерфейс межсетевого взаимодействия "Network-Network Interface" - NNI), так как обеим сетям "нужна возможность опроса" другой.

СДУ отличается от ССУ только в одном: сообщения *"Запрос состояния"* и *"Состояние"* имеют право передавать обе стороны интерфейса.

При СДУ обе стороны интерфейса FR передают сообщение *"Запрос состояния"* через определенный интервал (Т391), обе "требуют" ответ - сообщение *"Состояние"* (Т392), а также запрашивают информацию о полном состоянии (N391). При использовании этих процедур обе стороны могут запрашивать различные параметры. Вместе с тем обе стороны "ведут учет" номеров принимаемых и передаваемых кадров для каждого направления.

### 6.6.3. Асинхронное управление

Главный недостаток ССУ и СДУ - потенциальная задержка информирования ООД пользователя (или сети) об изменениях сетевых ПВК. Например, при задержке 60 с и CIR, равной 64 Кбит/с, ООД пользователя "направит" в сеть приблизительно 3,5 Мбит данных до того, как получит информацию о состоянии ПВК.

По этой причине была предложена стратегия АУ, когда используются стандартные сообщения *"Запрос состояния"* и *"Состояние",* которые

могут передаваться сразу в случае изменения ПВК сети FR. Эти сообщения содержат информацию только об отдельных ПВК, которые изменили свое состояние. Проверка целостности соединения также основана на генерации последовательности специальных пронумерованных кадров и проверке корректности этой последовательности с помощью специального процесса.

АУ может использоваться совместно с ССУ и СДУ. Однако, когда в сети FR применяются одновременно КВК и ПВК, то рекомендуется использовать только АУ.

### 6.6.4. Процедурная характеристика LMI при возникновении ошибок

Интерфейс локального управления предназначен для передачи минимального количества управляющей информации, чтобы гарантировать нормальное функционирование интерфейса FR. Вместе с тем существуют и специальные процедуры управления при возникновении следующих возможных ошибок в интерфейсе FR (обнаруживаемых сетью):

- прием кадра LMI о целостности соединения с неправильным порядковым номером принятого кадра (не равен порядковому номеру последнего переданного кадра);
- неприем сообщения *"Запрос состояния"* по истечении тайм-аута (этот интервал имеет международное обозначение - T392 и всегда должен быть больше, чем T391);
- прием кадра LMI с ошибкой FCS.

В этих случаях АКД сети FR в сообщении *"Состояние"* устанавливает бит "активный ПВК" в *"0",* указывая тем самым временную неготовность канала. Когда ошибка устранена, АКД сети устанавливает бит "активный ПВК" в "1". Однако данные действия АКД сети происходят не сразу при возникновении ошибок, а только при превышении установленного «порога». Этот порог определяется протоколом FR. Сеть осуществляет подсчет ошибок (максимальное значение этого числа имеет международное обозначение - N392), возникающих в пределах установленного периода (этот интервал имеет международное обозначение N393). Если за интервал N393 порог N392 превышен, то АКД сети переводит ПВК в неактивное состояние. Выход из него - получение сетью безошибочного сообщения *"Запрос состояния".*

Оборудование пользователя может также обнаруживать следующие возможные ошибки:

- прием кадра LMI о целостности соединения с неправильным порядковым номером принятого кадра (не равен порядковому номеру последнего переданного кадра);
- неприем сообщения *"Состояние"* по истечении интервала вр T391 после передачи сообщения *"Запрос состояния";*

- прием кадра LMI с ошибкой FCS.

Действия ООД пользователя аналогичны действиям АКД сети, в основе которых лежит тот же самый пороговый принцип: когда за интервал N393 порог N392 превышен, ООД пользователя прекращает передачу. Выход из него - передача ООД абонента сообщения *"Запрос состояния"*.

Однако стандарт FR не вводит процедур, на основе которых однозначно определяется, что ошибочная ситуация устранена и ООД абонента может передать сообщение *"Запрос состояния"*. Существует лишь одна возможность определения устранения ошибки, когда N392 событий происходят без ошибки. Необходимо также обратить внимание на то, что невозможно обнаружить ошибки в пределах отдельного ПВК интерфейса FR, то есть ошибки затронут все сконфигурированные ПВК.

Окончанием ошибочных ситуаций, которые могут произойти в LMI, являются:

- получение сообщения о состоянии ПВК для существующих ПВК (и для которых бит "новый ПВК" в кадрах LMI не устанавливался);
- получение кадра LMI с информацией о полном состоянии, который не включает ПВК, используемый в настоящее время.

Возможно, что выход ошибочных ситуаций будет происходить тогда, когда:

- сообщения о состоянии LMI были потеряны на линии связи;
- процедуры инициализации ПВК были некорректными. В этих случаях ООД абонента должно в кадрах LMI отмечать соответственно, что ПВК "активен" или "недоступен".

### *6.7. Параметры для синхронизации процедур управления LMI*

Для нормального функционирования процедур управления LMI используется ряд специальных счетчиков событий и времени, назначение которых - синхронизация последовательностей управляющей информации через интерфейс (табл. 6.10 – 6.11).

При ССУ счетчик кадров (N391) "ведет подсчет" кадров LMI *("Запрос состояния"),* переданных ООД пользователя, с информацией о целостности соединения. После того как будут переданы N391 сообщений *"Запрос состояния",* ООД пользователя передает кадр LMI *("Запрос состояния"),* в котором запрашивает информацию о полном состоянии ПВК. Для СДУ АКД сети также может использовать счетчик кадров (N391) для запроса информации о полном состоянии.

Максимально допустимое число (порог) ошибочных событий (N392) используется для подсчета числа ошибочных ситуаций, обнаруженных в интерфейсе, и должно быть всегда меньше или равняться интервалу для контроля ошибочных событий (N393). Анализу в пределах этого интервала подвергаются следующие события:

Таблица 6.10

Счетчики событий, используемые для синхронизации процессов управления LMI

| Международное обозначение счетчика | Содержание | Диапазон возможных значений | Значение "по умолчанию" | Назначение |
|---|---|---|---|---|
| N391 | Счетчик кадров для определения момента передачи сообщения о полном статусе | 1...255 | 6 | Определяет начало последовательности пронумерованных кадров LMI |
| N392 | Порог - максимально допустимое число ошибочных событий | 1...10 | 3 | Подсчет ошибочных событий |
| N393 | Интервал для контроля ошибочных событий | 1...10 | 4 | Подсчет ошибочных событий |

Таблица 6.10

Счетчики времени, используемые для синхронизации процессов управления LMI

| Международное обозначение таймера | Назначение | Диапазон возможных значений | Значение "по умолчанию" |
|---|---|---|---|
| T391 | Таймер для определения начала передачи сообщения о целостности связи | 5...30 | 10 |
| T392 | Временной интервал тайм-аута | 5...30 | 15 |

- получение корректного кадра LMI;
- получение недействительного кадра LMI;
- отсутствие кадра LMI в период тайм-аута (T392).

Таким образом, ситуация, при которой за интервал N393 число ошибочных ситуаций превысило порог N392, интерпретируется как состояние ошибки.

Очевидно, что в практических реализациях не следует устанавливать интервал N393, намного меньший, чем N391, потому что изменение состояния ПВК вследствие ошибочной ситуации произойдет без своевременного уведомления ООД пользователя.

Счетчик времени (T391) при ССУ используется ООД пользователя для определения начала передачи сообщения *"Запрос состояния"* (запрос о

целостности соединения), а при СДУ - АКД сети. Величины Т391, устанавливаемые ООД пользователя и АКД сети, могут различаться.

Величина тайм-аута (Т392) при ССУ используется АКД сети. Если сообщение *"Запрос состояния",* которое передается ООД абонента, не поступает в сеть до истечения таймера Т392, то АКД сети повторно передает абоненту сообщение *"Состояние"* (с номером последнего корректно принятого кадра LMI), при этом число ошибочных событий (N392) увеличивается на единицу. Т392 должен всегда быть больше, чем Т391. При СДУ таймер Т392 также используется ООД пользователя. Величины Т392, устанавливаемые ООД пользователя и АКД сети, могут быть различными.

# 7. Ретрансляция кадров (Frame Relay). Характеристики интерфейса «сеть - сеть» и коммутируемых виртуальных каналов

Современный стандарт FR включает протокол и интерфейс "пользователь-сеть" (ИПС) только для ПВК и поэтому в основном используется в корпоративных (региональных) и локальных сетях со статическими методами и способами маршрутизации информационных потоков.

Исторически сложилось так, что ретрансляция кадров первоначально использовалась в сетях FR, в которых скорость передачи информации составляла около 2,048 Мбит/с. Однако позже выяснилось, что этот метод применим и в сетях с гораздо большими скоростями передачи.

Вместе с тем при создании глобальной широкополосной FR сети, в которой будут применяться коммутируемые виртуальные каналы (КВК) и динамическое управление потоками информации, возникнет необходимость объединения различных существующих корпоративных и локальных FR сетей. Такая интеграция требует единого подхода к "философии" функционирования КВК и разработке стандарта на интерфейс "сеть - сеть" (ИСС). Разработкой и исследованием такого стандарта в настоящее время активно занимается FRF, а также ANSI, ITU-T и G4.

Первые результаты этой работы (проект стандарта на ИСС) были получены FRF в конце 1992 года.

## *7.1. Интерфейс «сеть-сеть»*

ИПС FR - это интерфейс, который определяет порядок доступа в FR сеть (частную или общего пользования), формат данных и процедуры (алгоритм) для взаимодействия ООД и АКД "внутри" одной сети. Однако при взаимодействии нескольких сетей FR между ними необходим шлюз (ИСС).

Области применения интерфейса "сеть-сеть":

- между FR сетью общего пользования с абонентским доступом и транзитной FR сетью общего пользования;
- между транзитными FR сетями общего пользования;
- между частной FR сетью с абонентским доступом и транзитной FR сетью общего пользования;
- между частной FR сетью с абонентским доступом и FR сетью общего пользования с абонентским доступом;
- между частными FR сетями с абонентским доступом.

Основное назначение ИСС - обеспечение эффективного взаимодействия двух (или нескольких) FR сетей в рамках глобальной FR сети с целью высококачественного обслуживания (высокая вероятность обслуживания неявки абонентов) пользователей при ведении ими информационного обмена. Следовательно, ИСС в первую очередь, должен обеспечивать высокоскоростную доставку данных, управление информационными потоками при возникновении перегрузок, сигнализацию и доставку служебной информации о состоянии канала связи. Проект стандарта (FRF) на ИСС аналогичен стандарту на ИПС, но в отличие от последнего рассматривает интерфейс локального управления только с асинхронным дуплексным управлением.

В основе функционирования ИСС лежит понятие "отрезок ПВК". Каждый такой отрезок "заключен" внутри глобальной "составной" FR сети (состоящей из нескольких FR подсетей) и ограничен либо ИПС с одной стороны и ИСС - с другой, либо ИСС с обеих сторон. Многосетевой ПВК (МПВК) является объединением двух или более отрезков ПВК.

При функционировании МПВК составные FR сети будут управляться (в соответствии с процедурами управления) со стороны как пользователя, так и соседней сети. Таким образом, в ИСС используются двунаправленные процедуры управления, то есть каждая из сторон интерфейса имеет право запрашивать у другой стороны информацию о статусе транзитного ПВК, так же как и информировать другую сторону о статусе ПВК. Более того, эти двунаправленные процедуры могут осуществляться асинхронно (с целью отслеживания состояния каналов в реальном масштабе времени).

Необходимо отметить, что ИСС использует локальную адресацию с различными DLCI в каждом интерфейсе (ИПС или ИСС).

Для нормального функционирования процедур управления в ИСС используются те же параметры синхронизации, что и для LMI (ряд специальных счетчиков событий и времени, назначение которых - синхронизация последовательностей управляющей информации через интерфейс: N391, N392, N393, T391 и T392). Очевидно, что текущие значения отдельных таймеров и счетчиков в каждом направлении ИСС будут различны. Однако проект стандарта FRF на ИСС строго оговаривает, что предельные (пороговые) значения таймеров и счетчиков на обеих сторонах ИСС долж-

ны быть одинаковыми. Важным свойством ИСС (требование к параметрам интерфейса) является максимально быстрая (насколько это возможно) передача служебной информации о состоянии ПВК между пользователями. Другими словами, если у одной из сторон интерфейса произошло изменение состояния ПВК, другая сторона ИСС должна быть оповещена незамедлительно (АДУ). Очевидно, что любая задержка в информировании другой стороны ИСС может привести к локальной перегрузке сети.

МПВК считается активным при условии, что все отрезки ПВК между двумя ИПС активны. Для этого необходимо, чтобы:

- все отрезки ПВК были сконфигурированы и функционировали;
- все ИПС и ИСС были сконфигурированы и функционировали;
- между всеми ИПС и ИСС были установлены соединения;
- удаленный пользователь сообщил об активности ПВК при условии, что удаленные ИПС используют двунаправленные процедуры.

Когда все эти условия выполнены, удаленный ИПС (ООД удаленного пользователя) в кадре FR установит бит *"активный ПВК"* в "1", так как только ИПС "имеет на это право". Если все ИСС функционируют нормально, то они "пропустят" этот кадр без изменений, в противном случае - установят бит *"активный ПВК"* в "0" (ИСС не имеет возможности устанавливать его в "1").

Очевидно, что во время установления соединения между ООД абонентов А и В внутри ИПС и ИСС происходит постоянный обмен сообщениями *"Запрос состояния"*.

Существует несколько причин сбоя МПВК: сбой в ИПС, в ИСС или в отрезке ПВК. В каждом случае ответственным за передачу сообщений *"Полное состояние",* в которых указывается неактивное состояние ПВК. является ИСС, что позволяет временно прекратить передачу данных через ИПС до тех пор, пока ПВК не будет реконфигурирован. Если имеет месте сбой в ИСС, то это вызовет сбой механизма синхронизации, что незамедлительно будет обнаружено ООД пользователей.

### *7.2. Коммутируемые виртуальные каналы*

Общепризнанно, что FR становится более эффективным методом доставки сообщений при условии использования КВК (которые создают только на период информационного обмена и "закрываются" сразу после него).

Однако реализация КВК, кажущаяся, на первый взгляд, простой, является наиболее сложной проблемой при стандартизации протоколов и интерфейсов FR. Это связано в первую очередь различными взглядами фи производителей и международных организаций по стандартизации на применение КВК в сетях FR. Более того, существует точка зрения, в соответствии с которой вообще ставится под сомнение необходимость КВК.

Поэтому FRF не принял стандартов на применение КВК. Существует только один стандарт (Рекомендация ITU-T Q.933), который был принят для ЦСИО, - "Система сигнализации для служб ретрансляции кадров". FRF согласился только с тем, что указанная выше рекомендация будет служить основой для будущего стандарта на использование КВК. Этот параграф посвящен логической и процедурной характеристикам протокола FR для КВК в любых FR сетях (и необязательно ЦСИО).

Все управляющие сообщения, используемые при установлении и разъединении КВК н передачи сообщения, передаются по ОКС, для которого в ИПС выделен канал с *DLCI = 0.* (Это вполне согласуется с философией FR, согласно которой процедуры сигнализации осуществляются по выделенному каналу.) Эти сообщения аналогичны кадру LMI. В табл. 7.1 представлен основной формат кадра, используемый при конфигурации КВК (далее - кадр КВК).

Таблица 7.1

Базовый формат кадра КВК

| Октеты | Биты | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | Назначение |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | Флаг |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Заголовок: DLCI=0, CR=0, |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | DE=0, FECN=0, BECN=0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | Индикатор ненумерованного кадра |
| 5 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | Определитель протокола |
| 6 | 0 | 0 | 0 | 0 | | | | | Вызываемый номер |
| 7 | | | | | | | | | |
| 8 | | | | | | | | | Тип сообщения |
| 9 | | | | | | | | | Первый информационный элемент |
| … | | | | | | | | | |
| 14 | | | | | | | | | |
| 15 | | | | | | | | | Второй информационный элемент |
| … | | | | | | | | | |
| 20 | | | | | | | | | |
| … | … | | | | | | | | … |
| | | | | | | | | | N-й информационный элемент |
| … | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | Проверочная последовательность |
| | | | | | | | | | |
| | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | Флаг |

Поля в кадре КВК идентичны полям кадра LMI процедур, за исключением полей: "Вызываемый номер", "Тип сообщения" и "Информационные элементы".

*"Вызываемый номер".* Это поле является локальным идентификатором, который предназначен для распознавания различных вызовов в локальном интерфейсе. Оно не имеет смысла при установлении межтерминального соединения, Длина этого поля может быть равна 2 или 3 октетам и зависит от требуемого количества индивидуальных каналов. Формат 2-октетного поля, наиболее приемлемого в сетях FR, показан в табл. 7.2.

Таблица 7.2

Базовый формат кадра КВК

| Октеты | Биты | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| 1 | 0 0 0 0 | | | | 0 0 0 1<br>Длина вызываемого номера | | | |
| 2 | Флаг | | | | Значение вызываемого номера | | | |

Первый октет указывает на длину вызываемого номера (1 или 2 октета), а второй октет содержит значение вызываемого номера и *"флаг".* Главное назначение *"флага":* определение стороны интерфейса - инициатора установления соединения ("0"; вызываемая сторона - "1") и, таким образом, исключение одинаковой установки этого бита обеими сторонами интерфейса. Поле "вызываемый номер" может иметь длину 3 октета, как правило, в том случае, когда требуется 7...15 битов для идентификации большого числа (>64) КВК в высокоскоростных ИСС (более 8 Мбит/с).

Уникальное значение вызываемого номера определяется инициатором вызова, когда передается сообщение на установление соединения.

*"Тип сообщения".* При создании, разъединении КВК и фазе передачи данных используются семь типов сообщений (Q.933), которые представлены в табл. 7.3.

*"Информационные элементы".* Информационные элементы для создания, разъединения КВК изменяются в соответствии с фазой информационного обмена. Рассмотрим эти фазы.

### 7.2.1. Фаза установления соединения (запрос соединения)

В этой фазе происходит установление соединения (конфигурирование КВК). Инициатором может быть либо ООД пользователя, либо АКД сети FR.

Среди трех типов сообщений, используемых в фазе установления соединения (КВК), наиболее сложным является *"Установка параметров".* Это сообщение (табл. 7.4) содержит семь информационных элементов.

Часть этих информационных элементов необязательна, а некоторые - обязательные - перечислены ниже.

Таблица 7.3

Кодирование поля "Тип сообщения"

| Биты | | | | | | | | Тип сообщения |
|---|---|---|---|---|---|---|---|---|
| 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | |
| 0 | 0 | 0 | - | - | - | - | - | Сообщения для установления соединения (КВК) |
| | | | 0 | 0 | 0 | 1 | 0 | Подтверждение вызова |
| | | | 0 | 0 | 1 | 1 | 1 | соединения (КВК) |
| | | | 0 | 0 | 1 | 0 | 1 | Установка параметров |
| 0 | 1 | 0 | - | - | - | - | - | Сообщения для разъединения (КВК) |
| | | | 0 | 0 | 1 | 0 | 1 | Запрос разъединения |
| | | | 0 | 1 | 1 | 0 | 1 | Согласие на разъединение |
| | | | 0 | 1 | 0 | 1 | 0 | Подтверждение разъединения |
| 0 | 1 | 1 | - | - | - | - | - | Смешанные сообщения |
| | | | 1 | 1 | 1 | 0 | 1 | Состояние |
| | | | 1 | 0 | 1 | 0 | 1 | Запрос состояния |

Таблица 7.4

Информационные элементы, входящие в сообщение "Установка параметров"

| Информационный элемент | Направление передачи | Тип | Максимальный размер |
|---|---|---|---|
| Тип сообщения - "Установка параметров" | Дуплексное | Обязательный | 1 |
| Пропускная способность | Дуплексное | Обязательный | 5 |
| Идентификатор канала передачи данных (DLCI) | Дуплексное | - | 4 |
| Параметры канального уровня (ЭМВОС) | Дуплексное | - | 14 |
| Подадрес вызывающей стороны | Дуплексное | | 23 |
| Адрес вызываемой стороны | Дуплексное | Обязательный | - |
| Подадрес вызываемой стороны | Дуплексное | - | 23 |
| Совместимость нижнего уровня | Дуплексное | - | 16 |

*Пропускная способность.* Назначение этого обязательного элемента заключается в предоставлении Службой FR (ITU-T, I.233) соответствующей услуги по ретрансляции кадров с требуемым качеством. Все поля внутри этого информационного элемента фиксированы и обозначают соответствующие параметры. Но значения этих параметров не стандартизова-

ны, что является предметом дальнейших исследований. Причина того, что этот элемент обязателен, - обеспечение совместимости со стандартами ЦСИО (хотя возможно применение FR и не в ЦСИО).

В табл. 7.5 представлен формат информационного элемента *"пропускная способность".*

Таблица 7.5

Формат информационного элемента "пропускная способность"

| 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | Октеты |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| Идентификатор информационного элемента "пропускная способность" | | | | | | | | |
| Размер информационного элемента "пропускная способность" | | | | | | | | 2 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 3 |
| Бит расширения | Стандарт кодирования (ITU-T) | | Потенциальные возможности по доставке сообщений | | | | | |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 4 |
| Бит расширения | Режим доставки | | Зарезервировано | | | | | |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 5 |
| Бит расширения | Идентификатор канального уровня (ЭМВОС) | | Протокол канального уровня об информации пользователя | | | | | |

Поля внутри этого формата фиксированы и означают:

- *вид кодирования,* стандартизированный ITU-T;
- *любая информация* в цифровой форме (потенциальные возможности по доставке сообщений);
- ретрансляция кадров *(режим доставки);*
- *ANSI T1.618 / ITU-T Q.922* - основные аспекты протокола (протокол канального уровня об информации пользователя).

Бит *расширения* поля используется для индикации размера переменного поля внутри информационного элемента. Для одноктетных полей этот бит всегда устанавливается в "1". Если какое-либо из полей больше, чем 1 октет, то этот бит устанавливается в "1" только в последнем октете данного поля, а в предшествующих октетах - в *"0".* В дальнейшем, для большей ясности, восьмой бит первого октета будет установлен в *"0",* последнего - в *"1",* промежуточных (если таких октетов больше двух) - *"0/1".*

*Идентификатор канала передачи данных. (DLCI).* Обязателен в направлении "АКД сети - ООД пользователя" и необязателен - в направлении "пользователь - сеть". Этот информационный элемент определяет DLCI КВК, который будет использоваться в локальном интерфейсе. В случае, когда инициатором соединения является АКД сети FR (то есть последняя

посылает сообщение *"Установка параметров"),* элемент всегда содержит DLCI, выбранный при установлении соединения. В противном случае, когда инициатором соединения является ООД абонента (то есть последнее посылает сообщение *"Установка параметров"),* этот элемент является необязательным, до тех пор пока ООД абонента не затребует значение DLCI, используемого при информационном обмене. Формат элемента представлен в табл. 7.6 (для двухоктетного адресного поля). Необходимо заметить, что DLCI, размещаемый в стандартном поле этого элемента, является локальным и имеет значение только для конкретной сети FR. Бит *уникальный флаг* всегда устанавливается в *"0".* Если сеть неспособна (по каким-либо "внутренним" причинам) указать конкретное значение номера (DLCI) для требуемого канала, то она будет указывать другое (приемлемое для нее) значение DLCI. Бит *расширения* также применяется для индикации окончания адресного поля и будет использоваться таким же образом (то есть как маркер расширения адреса) в заголовке кадра.

Таблица 7.6

Формат информационного элемента "DLCI"

| 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | Октеты |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| Идентификатор информационного элемента "DLCI" | | | | | | | | |
| Размер информационного элемента "DLCI" | | | | | | | | 2 |
| 0<br>Бит расширения | 0<br>Уник. флаг | DLCI (первые 6 битов) | | | | | | 3 |
| 1<br>Бит расширения | DLCI (последние 4 бита) | | | | Зарезервировано | | | 4 |

### 7.2.2. Параметры канального уровня

Этот информационный элемент (ПКУ) является необязательным в направлении "ООД абонента - АКД сети" и используется только тогда, когда вызывающий адрес затребует у сети значения необходимых параметров. Если параметры в этом направлении не представлены, то вся ответственность за несоблюдение требуемых параметров (их конкретных значений) ложится на сеть. Элемент является обязательным в направлении "АКД сети - ООД абонента" и имеет своей целью информирование ООД абонента о предполагаемых значениях требуемых параметров. Формат элемента представлен в табл. 7.7.

Значению каждого параметра, входящего в информационный элемент, предшествует поле идентификатора. Рассматриваемый информационный элемент включает определенный набор параметров, в который входят:

- максимальный размер поля информации (ПИ) в кадре FR (в октетах);

Таблица 7.7

Формат информационного элемента ПКУ

| **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **Октеты** |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 1 |
| Идентификатор информационного элемента ПКУ | | | | | | | | |
| Размер информационного элемента ПКУ | | | | | | | | 2 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 3 |
| 0 Расш. | Идентификатор "максимальный размер ПИ в кадре FR" | | | | | | | |
| 0 Расш. | Максимальный размер ПИ в передаваемом кадре FR | | | | | | | 4 |
| 0/1 Расш. | Продолжение | | | | | | | 5 |
| 0 Расш. | Максимальный размер ПИ в принимаемом кадре FR | | | | | | | 6 |
| 1 Расш. | Продолжение | | | | | | | 7 |
| 0 Расш. | 0 | 0 | 0 | 1 | 0 | 1 | 0 | |
| | Идентификатор "пропускная способность" | | | | | | | 8 |
| 0 Расш. | Размерность ВПС при передаче информации | | | Значение множителя ВПС при передаче информации | | | | 9 |
| 0/1 Расш. | Продолжение | | | | | | | 10 |
| 0 Расш. | Размерность ВПС при приеме информации | | | Значение множителя ВПС при приеме информации | | | | 11 |
| 1 Расш. | Продолжение | | | | | | | 12 |
| 0 Расш. | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 13 |
| | Идентификатор "ГОИ для перед./приема" | | | | | | | |
| 0 Расш. | Величина ГОИ для передачи | | | | | | | 14 |
| 0/1 Расш. | Продолжение | | | | | | | 15 |
| 0 Расш. | Величина ГОИ для приема | | | | | | | 16 |
| 1 Расш. | Продолжение | | | | | | | 17 |
| 0 Расш. | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 18 |
| | Идентификатор "ДОИ для передачи/приема" | | | | | | | |
| 0 Расш. | Величина ДОИ для передачи | | | | | | | 19 |
| 0/1 Расш. | Продолжение | | | | | | | 20 |
| 0 Расш. | Величина ДОИ для приема | | | | | | | 21 |
| 1 Расш. | Продолжение | | | | | | | 22 |

• величина пропускной способности (ВПС). Этот параметр представляется в значениях скорости передачи информации (бит/с), обеспечиваемой ПВК за определенный интервал времени. Пропускная способность является составной величиной и включает "размерность" ($10^0$...$10^6$) и цело-

численный "множитель". Например, 64 Кбит/с будет иметь вид: 3/64 ("размерность"/"множитель"), что означает $64x10^3$;

- гарантированный объем информации (ГОИ), подлежащей передаче/приему. Этот параметр определяет максимальный размер данных пользователя (в октетах), который согласовывается с сетью для их доставки в каждом направлении за определенный интервал при условии ее нормального функционирования;
- дополнительный объем информации (ДОИ), подлежащей передаче/приему. Этот параметр определяет максимальный дополнительный объем данных (в октетах), который сеть будет "пытаться" доставить в каждом направлении за определенный интервал времени. Во всех кадрах, которые превышают гарантированный объем, но не превышают дополнительный объем, бит DE может быть установлен сетью в "1".Все параметры, содержащиеся в информационном элементе *"параметры канального уровня",* являются необязательными, а их размещение в нем - независимое. Бит расширения в последнем октете каждого поля параметра должен быть установлен в "1" для индикации окончания этого поля, а во всех предшествующих - в "0".

*Подадрес вызывающей стороны.* Этот информационный элемент может входить в сообщение *"Установка параметров"* в направлении "ООД абонента - АКД сети" только тогда, когда пользователь потребует указать подадрес вызывающей стороны, а в обратном направлении - "АКД сети - ООД абонента", - если пользователь (ООД абонента - инициатор соединения) сам указывает этот подадрес.

В табл. 7.8 представлен формат информационного элемента *"подадрес вызывающей стороны".*

Таблица 7.8

Формат информационного элемента "подадрес вызывающей стороны"

<table>
<tr><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>Октеты</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td></td></tr>
<tr><td colspan="8">Идентификатор информационного элемента "подадрес вызывающей стороны"</td><td>1</td></tr>
<tr><td colspan="8">Размер информационного элемента "подадрес вызывающей стороны"</td><td>2</td></tr>
<tr><td rowspan="2">1<br>Бит расширения</td><td rowspan="2">Тип подадреса</td><td rowspan="2">Вид кодирования</td><td>0</td><td>0</td><td>0</td><td colspan="2" rowspan="2">3</td></tr>
<tr><td colspan="3">Зарезервировано</td></tr>
<tr><td colspan="8">Значение подадреса</td><td>4 и т. д.</td></tr>
</table>

Параметры, входящие в этот информационный элемент:

- *тип подадреса,* который определяет либо тип используемой адресации, либо X.213 (так называемый адрес точки доступа, физический адрес, к сети) - *"000",* либо специфический адрес пользователя - *"010";*
- *вид кодирования адреса,* который определяет вид представления (двоичное или четверичное) адреса, имеющего десятичную форму;
- *значение подадреса,* максимальное значение которого - 20 октетов.

*Подадрес вызываемой стороны.* Этот информационный элемент используется для индикации адреса вызываемой стороны при запросе ООД абонента установления соединения. Он включается в сообщения *"Установка параметров",* передаваемые в обоих направлениях ("ООД абонента - АКД сети", "АКД сети - ООД абонента"). В табл. 7.9 представлен формат этого элемента.

В информационный элемент входят:

- *тип номера,* который указывает тип нумерации (международная - "001", национальная - "010", абонентская - "100", специальная - "011" и сокращенная - "110");
- *идентификатор системы нумерации* (ISDN/E.164 - "0001"', X.121 - "ООН". "Телекс"/P.69 - "0100" и частная - "1001");
- *значение номера* кодируется IA5 (международный алфавит 5 - International Alphabet 5) символами.

Таблица 7.9

Формат информационного элемента "подадрес вызываемой стороны"

<table>
<tr><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>Октеты</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td></td></tr>
<tr><td colspan="8">Идентификатор информационного элемента "подадрес вызывающей стороны"</td><td>1</td></tr>
<tr><td colspan="8">Размер информационного элемента "подадрес вызываемой стороны"</td><td>2</td></tr>
<tr><td rowspan="2">1/0<br>Бит расширения</td><td colspan="3">Тип номера</td><td colspan="4">Идентификатор системы нумерации</td><td>3</td></tr>
<tr><td colspan="7">Значение номера</td><td>4 и т.д.</td></tr>
</table>

*Совместимость нижнего уровня (СНУ).* Этот информационный элемент включается в сообщение *"Установка параметров"* в направлении "ООД абонента - АКД сети" только тогда, когда пользователь хочет передать информацию о совместимости нижнего уровня вызываемому абоненту, а в направлении "АКД сети - ООД абонента" этот элемент включается вызывающим абонентом. Такой информацией обмениваются между собой взаимодействующие пользователи с целью обеспечения совместимости

своих нижних уровней. Содержание этого информационного элемента прозрачно для сети.

В табл. 7.10 представлен формат этого информационного элемента. В него входят следующие параметры:

Таблица 7.10

Формат информационного элемента "СНУ"

<table>
<tr><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>Октеты</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td colspan="8">Идентификатор информационного элемента "СНУ"<br>Размер информационного элемента "СНУ"</td><td>2</td></tr>
<tr><td rowspan="2">0/1<br>Расш.</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td rowspan="2">3</td></tr>
<tr><td colspan="3">Стандарт кодирования</td><td colspan="5">Потенциальные возможности по доставке информации</td></tr>
<tr><td rowspan="2">1<br>Расш.</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td rowspan="2">4</td></tr>
<tr><td colspan="2">Режим доставки</td><td colspan="5">Зарезервировано</td></tr>
<tr><td rowspan="2">0/1<br>Расш.</td><td>1</td><td>0</td><td colspan="5" rowspan="2">Информация ООД пользователя о протоколе 2-го уровня</td><td rowspan="2">5</td></tr>
<tr><td colspan="2">Идентификатор уровня 2</td></tr>
<tr><td>1<br>Расш.</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td colspan="2">Дополнение адреса</td><td>6</td></tr>
<tr><td>1<br>Расш.</td><td colspan="7">Установочные данные пользователя</td><td>7</td></tr>
<tr><td rowspan="2">0/1<br>Расш.</td><td>1</td><td>1</td><td colspan="5" rowspan="2">Информация ООД пользователя о протоколе 3-го уровня</td><td rowspan="2">8</td></tr>
<tr><td colspan="2">Идентификатор уровня 3</td></tr>
<tr><td>1<br>Расш.</td><td colspan="7">Необязательная информация о протоколе 3-го уровня</td><td>9</td></tr>
</table>

- *стандарт кодирования/потенциальные возможности по доставке информации.* Это фиксированные значения, показывающие универсальность цифровой информации, подлежащей доставке;
- *режим доставки.* Это поле указывает на услуги по ретрансляции кадров;
- *информация ООД пользователя о протоколе 2-го уровня.* Это поле содержит служебную информацию о протоколе 2-го уровня, которая передается внутри FR кадра. Кодирование поля включает: X.25 (канальный уровень), X.75 (канальный уровень); 8802.2; Q.922; HDLC (сбалансированный режим, режим ответов и нормальный режим);
- *дополнение адреса.* Это поле указывает наличие либо адреса канального уровня, либо логического управления внутри этого кадра. Если

поле закодировано как "01" (наличие адреса), это означает, что имеют место протоколы канального уровня X.25, X.75 и HDLC, если - "10" (кадр логического управления), то имеет место 8802.2;

- *информация ООД пользователя о протоколе 3-го уровня.* Это поле содержит служебную информацию о протоколе 3-го уровня, которая передается внутри FR кадра. Кодирование поля включает: X.25, X.223, ISO 8208 и ISO 8473;
- *необязательная информация о протоколе 3-го уровня.* Это дополнительное поле предназначено специально для ООД абонента и служит для обмена специфической информацией между пользователями. Кодирование поля включает: X.25, X.223, ISO 8208, ISO 8473 и ISO T.70.

### 7.2.3. Фаза установления соединения (подтверждение вызова и соединения)

В ответ на сообщение *"Установка параметров"* сеть ответит сообщением *"Подтверждение вызова",* в котором подтвердит наличие соединения и укажет на то, что требуемые параметры при установлении соединения обеспечены. В табл. 7.11 представлены информационные элементы, входящие в сообщение *"Подтверждение вызова".* Последнее включает DLCI, указывающее номер КВК, входящий в информационный элемент сообщения *"Установка параметров".* Однако значение DLCI может не соответствовать значению DLCI, указанному в информационном элементе сообщения *"Установка параметров", "Подтверждение соединения".*

Таблица 7.11

Информационные элементы, входящие в сообщение "Подтверждение вызова"

| Информационный элемент | Направление передачи | Тип | Максимальный размер (октеты) |
|---|---|---|---|
| Тип сообщения - "Подтверждение вызова" | Дуплексное | Обязательный | 1 |
| DLCI | Дуплексное | - | 4 |

Финальным сообщением, используемым в течение фазы установления соединения, является сообщение *"Подтверждение соединения".* Оно передается как ООД вызываемого абонента в направлении АКД сети, так и АКД сети в направлении ООД вызывающего абонента. В табл. 7.12 представлены информационные элементы, входящие в сообщение *"Подтверждение соединения".* Последнее включает DLCI, указывающее номер КВК, входящий в информационные элементы сообщений *"Установка параметров", "Подтверждение вызова".* Рассмотренная выше фаза установления соединения относится к "удачной процедуре вызова". Рекомендация ITU-T

Q.933 содержит описание ошибочных ситуаций и действий в случае их возникновения.

Таблица 7.12

Информационные элементы, входящие в сообщение

| Информационный элемент | Направление передачи | Тип | Максимальный размер (октеты) |
|---|---|---|---|
| Тип сообщения - "Подтверждение соединения" | Дуплексное | Обязательный | 1 |
| DLCI | Дуплексное | - | 4 |

### 7.2.4. Фаза разъединения

В этой фазе происходит разъединение ранее установленного КВК. Процедура разъединения может быть инициирована одной из сторон независимо от того, кто был организатором данного сеанса связи.

В фазе разъединения используются три сообщения: *"Запрос разъединения", "Согласие на разъединение"* и *"Подтверждение разъединения"*.

Сообщение *"Запрос разъединения"* может быть передано либо ООД абонента в направлении АКД сети с требованием разъединения КВК, либо АКД сети с целью индикации начала процедуры разъединения. Сообщение *"Запрос разъединения"* содержит только один информационный элемент *"причина разъединения"*.

Этот элемент предназначен только для указания причины разъединения. Кодирование полей информационного элемента *"причина разъединения"* представлено в табл. 7.13.

Таблица 7.13

Информационный элемент "причина разъединения"

<table>
<tr><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>Октеты</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td colspan="8">Идентификатор информационного элемента "причина разъединения"</td><td rowspan="2">2</td></tr>
<tr><td colspan="8">Размер информационного элемента "причина разъединения"</td></tr>
<tr><td rowspan="2">0/1<br>Расшир.</td><td>0</td><td>0</td><td rowspan="2">0<br>Резерв</td><td colspan="3" rowspan="2">Местонахождение непосредственного инициатора разъединения</td><td rowspan="2">3</td></tr>
<tr><td colspan="2">Стандарт кодирования</td></tr>
<tr><td>1<br>Расшир.</td><td colspan="6">Код причины разъединения</td><td>4</td></tr>
</table>

Параметры, содержащиеся в информационном элементе:

- стандарт кодирования ITU-T;
- местонахождение непосредственного инициатора разъединения ("0001" -местный пользователь, обслуживаемый частной FR сетью, "0010" - местный пользователь, обслуживаемый FR сетью общего пользования,

"OO11" - транзитная FR сеть, "0101" - удаленный пользователь, обслуживаемый частной FR сетью, "0100" - удаленный пользователь, обслуживаемый FR сетью общего пользования, "0001" - международная сеть);

- код причины разъединения. Возможные причины разъединения стандартизированы в Рекомендациях ITU-T Q.933 и Q.931 и имеют свое уникальное кодирование. Кодовая комбинация "16" означает нормальное разъединение. Необходимо заметить, что для указания причины разъединения не рекомендуется использование кодовой комбинации "00000000" с целью исключения возможных конфликтных ситуаций.

В ответ на сообщение *"Запрос разъединения"* пользователь посылает *"Согласие на разъединение"*, в котором "сообщает" сети, что ООД разрывает КВК и чтобы последняя информировала об этом вызывающую сторону. Сеть отвечает на это сообщением *"Подтверждение разъединения",* которое подтверждает разъединение КВК и служит для индикации того, что DLCI свободен для следующего использования.

Форматы сообщений *"Согласие на разъединение"* и *"Подтверждение разъединения"* одинаковы, и они содержат (помимо заголовка) один информационный элемент *"причина разъединения"* (табл. 7.14). Этот элемент идентичен тому, который используется в первоначальном сообщении *"Запрос разъединения".*

Возможна такая ситуация, когда самой сети предпочтительнее начать процедуру разъединения, если внутри этой сети произошел сбой, причиной которого мог быть (в том числе) разрыв межабонентской цепи. В этом случае АКД сети передает сообщение *"Согласие на разъединение"* (а не *"Запрос разъединения')* каждому индивидуальному ООД абонента, после чего ожидает соответствующий ответ *"Подтверждение разъединения".*

Таблица 7.14

Информационные элементы, входящие в сообщения "Согласие на разъединение" и "Подтверждение разъединения"

| Информационный элемент | Направление передачи | Тип | Максимальный размер (октеты) |
|---|---|---|---|
| Тип сообщения - "Согласие на разъединение" и "Подтверждение разъединения" | Дуплексное | Обязательный | 1 |
| Причина разъединения | Дуплексное | Обязательный | 4...32 |

# 8. Интеграция FR сетей

Обычно любой протокол управления высокого уровня (от сетевого уровня и выше) функционирует на основе базового протокола ретрансляции кадров. В настоящее время имеется несколько стандартов сетевых протоколов, то есть существует несколько типов ИВС. Следовательно, если даже FR сеть является интегрирующей транспортной основой высокоуровневой сети, построенной с использованием технических средств различных фирм-производителей, то возможна такая ситуация, при которой ООД абонентов будут взаимодействовать только с такими же (изготовленными той же фирмой) ООД (рис. 8.1).

Такая ситуация для потребителей сетевых услуг крайне нежелательна. Понимая это, ANSI предложил пакет проектов стандартов протоколов и интерфейсов FR для утверждения их в качестве международных, позволяющих взаимодействовать с высокоуровневыми (сетевыми и выше) протоколами, включая ISO 8208, 8802.2 и IP. Тем не менее, являясь международной организацией по стандартизации, ANSI не издает стандарты протоколов де-факто, а направляет последние на рассмотрение FRF, которая является международной организацией потребителей (т.е. организацией фирм-производителей, реализующих стандарты протоколов в своих коммуникационных аппаратно-программных комплексах).

## *8.1. Характеристика FR протокола для интеграции сетей, функционирующих по различным сетевым протоколам*

Важным преимуществом использования сети FR в качестве транспортной среды (применение кадра FR в качестве базового информационного элемента, в который "вкладывается" пакет сетевого протокола) является то, что сетевые протоколы имеют возможность достаточно простого построения (конфигурации) распределенной СПД на основе "простых физических" FR соединений. Для достижения этой простоты необходимо иметь некоторый уникальный механизм в рамках FR протокола (дополнение к процедурной и логической характеристикам FR протокола) для идентификации сетевого протокола при передаче пакета, "вкладываемого" в FR кадр. Главной целью такого механизма является обеспечение корректной доставки сообщения пользователя адресату. Рассмотрим основные положения проекта международного стандарта, разработанного ANSI (T1.617, Annex F).

Сущность методологии идентификации сетевого протокола при передаче пакета, "вкладываемого" в FR кадр, заключается в использовании стандартного формата FR кадра, в котором имеется специальное поле идентификатора протокола 3-го уровня в поле данных (табл. 8.1). В даль-

нейшем такой FR кадр будем именовать многопротокольным кадром, включающим в себя следующие поля:

- *адресное поле.* Имеет стандартный размер - 2 октета, но может быть увеличено при необходимости до 3 или 4 октетов;
- *идентификатор ненумерованного информационного кадра* (ИНИК). Это поле кодируется в соответствии с Рекомендацией ITU-T Q.922 (используется так же, как и в кадре LMI);

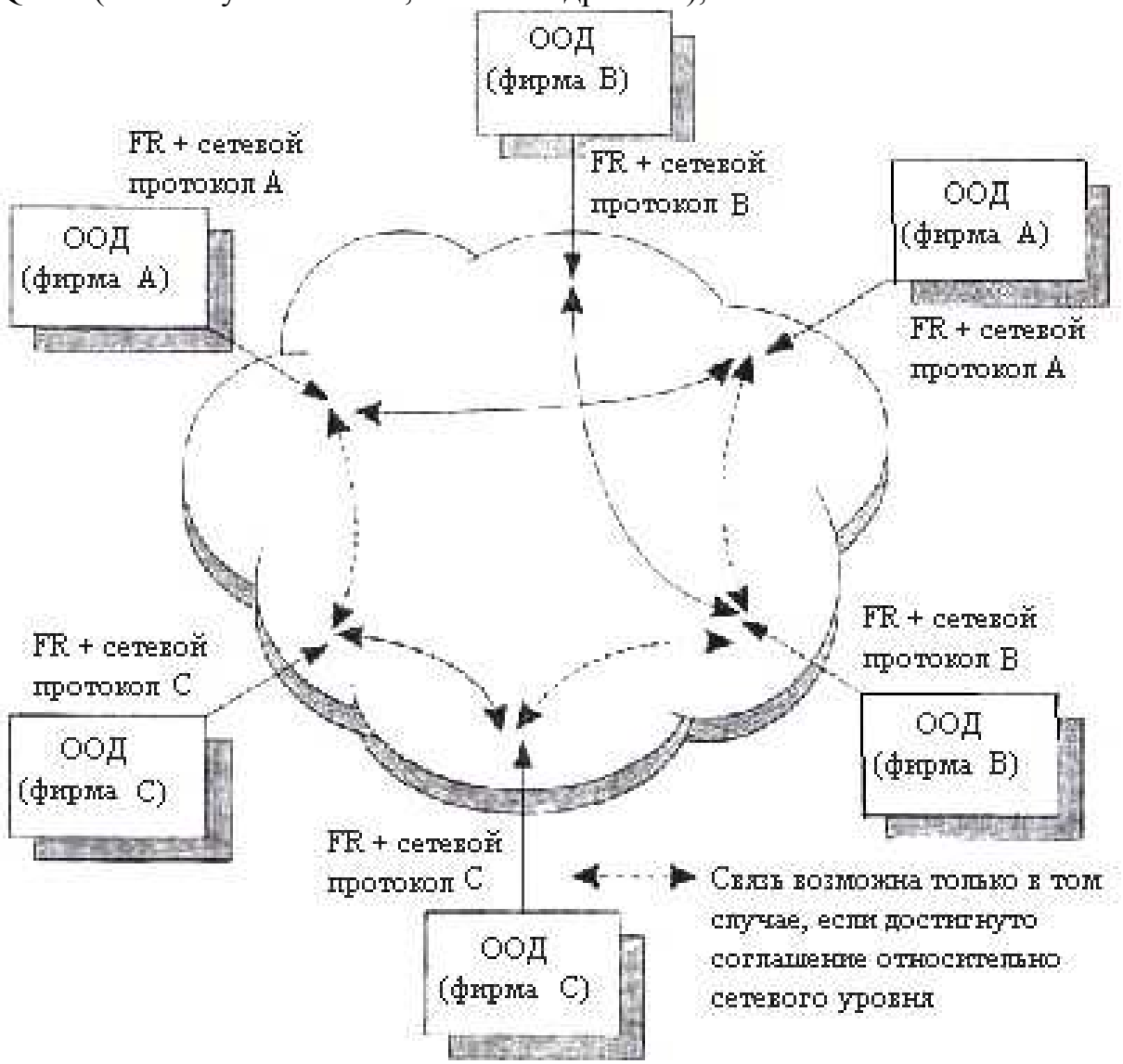



Рис. 8.1. Взаимодействие протоколов сетевого уровня

- *необязательное поле.* Это поле может быть полезным для отправителя кадра в целях более четкого разграничения границы заголовка кадра, а также для "выравнивания", если это необходимо, размера кадра. Если это поле (либо несколько октетов) используется, то оно должно иметь только нулевое заполнение;
- *идентификатор протокола сетевого уровня* (ИПСУ). Это поле содержит код сетевого протокола, в соответствии с процедурной характе-

ристикой которого осуществляется передача пакета. Кодирование этого поля осуществляется в соответствии со стандартами ANSI и ITU-T при условии, что все фирмы-производители примут единую систему идентификации.

Таблица 8.1

Формат многопротокольного кадра

| Октеты | Биты | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | Назначение |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | Флаг |
| 2 | | | | | | | | | Адресная часть заголовка стандартного FR кадра |
| 3 | | | | | | | | | |
| 4 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | Идентификатор ненумерованного информационного кадра |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Необязательное поле |
| 6 | | | | | | | | | Идентификатор протокола сетевого уровня |
| 7... | | | | | | | | | ДАННЫЕ |
| | | | | | | | | | Проверочная последовательность |
| | | | | | | | | | |
| | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | Флаг |

Если некоторый сетевой протокол не имеет ИПСУ, определенного стандартами ANSI/ITU-T, в этом случае может быть использован специальный код (шестнадцатеричная *"08"),* который определен в Рекомендации ITU-T Q.933. Если для ИПСУ используется 4-октетное поле, то оно служит для идентификации протоколов канального и сетевого уровней (табл. 8.2). Кодирование этих полей определено стандартом ANSI T 1.617 и Рекомендацией ITU-T Q.933 и тем самым обеспечивает совместимость на нижних уровнях. Более того, возможность использования специального кода (идентификатора) в первом октете этих двух полей (7...10-й октеты) позволяет идентифицировать уникальные пользовательские протоколы.

Такой формат многопротокольного кадра обеспечивает возможность идентифицировать протоколы взаимодействия объединенных локальных вычислительных сетей (ЛВС) и маршрутизации, определенные стандартами Американского института инженеров в области электротехники и электроники (IEEE - Institute of Electrical and Electronics Engineers "Комитет 802 по стандартизации ЛВС") для протоколов доступа к ЛВС (ПДЛВС). В этом случае для идентификации ПДЛВС IEEE (заголовок ПДЛВС) используются пять октетов, следующих после октета ИПСУ. Заголовок ПДЛВС имеет следующее кодирование:

- первые три октета (уникальный идентификатор организации) определяют организацию, которая устанавливает кодирование последних двух октетов;
- последние два октета - идентификатор протокола.

Таблица 8.2

Формат многопротокольного кадра, использующего Рекомендацию ITU-T для идентификации протокола сетевого уровня

| Октеты | Биты | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | Назначение |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | Флаг |
| 2 | | | | | | | | | Адресная часть заголовка стандартного FR кадра |
| 3 | | | | | | | | | |
| 4 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | ИНИК |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Необязательное поле |
| 6 | | | | | | | | | Идентификатор протокола сетевого уровня |
| 7<br>8 | | | | | | | | | Идентификатор протокола 2-го уровня |
| 9<br>10 | | | | | | | | | Идентификатор протокола 3-го уровня |
| 11... | | | | | | | | | ДАННЫЕ |
| | | | | | | | | | Проверочная последовательность |
| | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | Флаг |

Все узлы связи и маршрутизаторы должны распознавать и соответствующим образом интерпретировать поле ИПСУ и заголовок ПДЛВС.

Заголовок ПДЛВС пригоден как для протоколов маршрутизации в ЛВС, так и для протоколов взаимодействия объединенных ЛВС. При использовании последних существует дополнительная необязательная функция поля FCS в FR кадре, которая заключается в том, что в этом поле размещается циклическая проверка корректности пакета, сформированного протоколом управления ЛВС и "вложенного" в FR кадр (то есть имеет место двойное применение поля FCS). Кодирование идентификаторов протоколов взаимодействия объединенных ЛВС представлено в стандартах IEEE 802.3, 802.4, 802.5, 802.6 и на волоконно-оптическую интерфейсную ЛВС (ЛВС ВОИ: FDDI - Fiber Distributed Data Interface).

Использование многопротокольного FR кадра обеспечивает возможность передачи через межсетевой интерфейс, осуществляющий многопротокольное пакетирование, пакетов ЛВС, которые имеют максимальный размер, превышающий размер поля данных FR кадра. В этом случае необходимо иметь устройство доступа, обеспечивающее "разбиение" (фрагментирование) пакетов ЛВС на меньшие информационные блоки. При этом

элементарные блоки пакета ЛВС должны иметь индивидуальные номера, с помощью которых можно "собирать" (восстанавливать) исходный пакет ЛВС. Процедура фрагментирования и формирования FR кадра представлена на рис. 8.2.

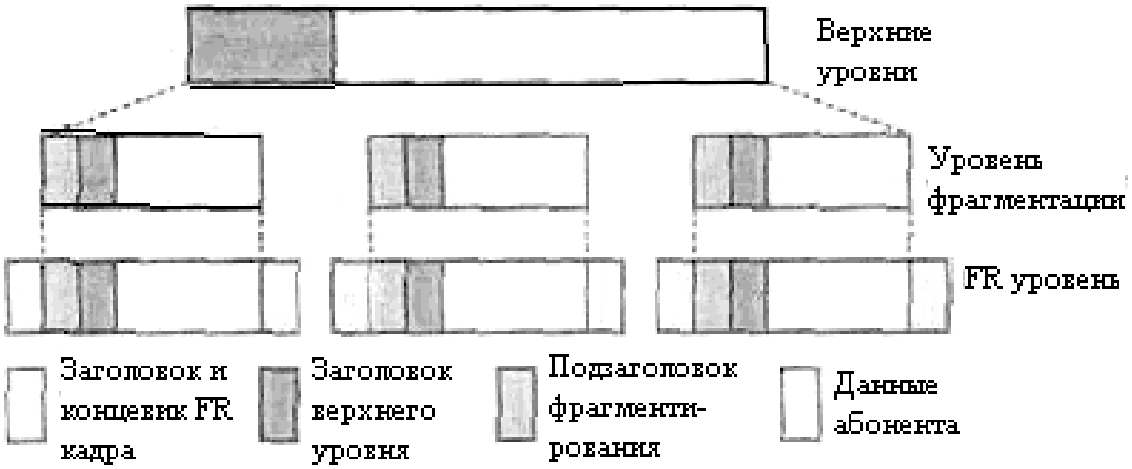



Рис. 8.2. Процедура фрагментирования и формирования FR кадра

На первом этапе устройство доступа "добавляет" к пакету ЛВС пятиоктетный заголовок (см.рис. 8.2). Это большое сообщение, с "добавленным" заголовком, затем разбивается на блоки меньшей длины, равной размеру информационного поля кадра FR сети. Далее эти маленькие блоки рассматриваются как блоки "чистых" данных, к которым добавились (еще до поступления в сеть) так называемые заголовки фрагментирования. После чего каждый такой блок "вкладывается" в многопротокольный FR кадр, представленный в табл. 8.3. "Внутреннее содержимое" информационного поля многопротокольного FR кадра (включая заголовки фрагментирования) определяется исключительно протоколами верхних уровней (но не канального).

Заголовок FR кадра (при фрагментировании исходного сообщения) содержит следующие поля:

- *адресное поле.* Стандартный формат FR кадра, имеющий 2-октетную или, если это необходимо, 3- и 4-октетную длину;
- *идентификатор ненумерованного информационного кадра.* Этот идентификатор определяется Рекомендацией ITU-T Q.922 (его значение аналогично значению кадра LMI).

На приемной стороне блоки собираются в исходное сообщение, после чего выполняются процедуры протокола верхнего уровня. Сборка исходного сообщения осуществляется на основе анализа значений смещения, которые передаются в подзаголовке фрагментирования. Любой кадр может быть потерян в сети, что будет обнаружено протоколом верхнего уровня ввиду отсутствия одного (или более) блока исходного сообщения. В этом

случае ответственность за восстановление исходного сообщения (путем его перезапроса) полностью ложится на протокол более высокого уровня.

Таблица 8.3

Формат многопротокольного кадра, "переносящего" один из блоков исходного сообщения, которое подверглось делению

| Октеты | Биты | | | | | | | | Назначение |
|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | Флаг |
| 2 | | | | | | | | | Адресная часть заголовка стандартного FR кадра |
| 3 | | | | | | | | | |
| 4 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | ИНИК |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Необязательное поле |
| 6 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | ИПСУ |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Уникальный идентификатор организации |
| 8 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | |
| 9 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Идентификатор протокола |
| 11 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | |
| 12 | | | | | | | | | Последовательный номер |
| 13 | | | | | | | | | |
| 14 | Смещение | | | Резерв | | | | F | Подзаголовок фрагментирования |
| 15 | Смещение (продолжение) | | | | | | | | |
| 6... | | | | | | | | | БЛОК ДАННЫХ |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | Проверочная последовательность |
| | | | | | | | | | |
| | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | Флаг |

На рис. 8.3 показана процедура фрагментирования и формирования первых двух FR кадров из исходной "большой" IP дейтаграммы.

### *8.2. Интеграция FR и X.25 сетей*

Рекомендация ITU-T X.25 определяет стандарты протоколов и интерфейсов канального и сетевого уровней. И с точки зрения ретрансляции кадров протоколы X.25 являются высокоуровневыми "механизмами" управления информационным обменом. Поэтому существует система взглядов, в соответствии с которой кадры X.25 достаточно корректно вписываются в формат многопротокольного FR кадра.

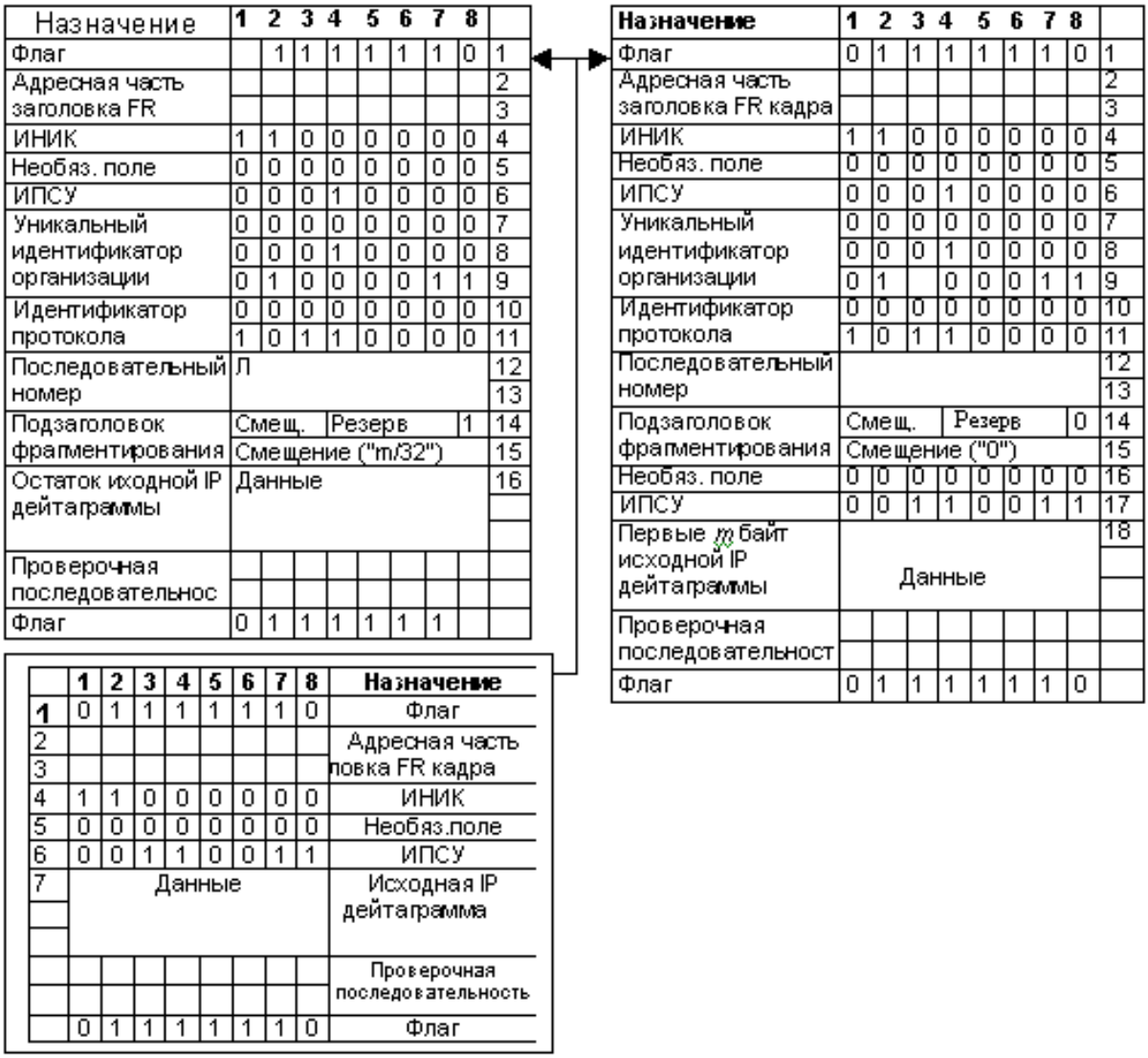



Рис. 8.3. Пример фрагментирования IP дейтаграммы

Однако многие фирмы, производящие аппаратно-программные средства для X.25 сетей, считают, что механизмы формирования многопротокольного FR кадра чрезвычайно сложны и существует более простая реализация протоколов доставки пакетов X.25 через FR сеть. Более того, последнее активно внедряется в сетевое оборудование и абонентские пункты.

В связи с этим в стандарт ANSI T1.617 (приложение G) были внесены поправки, которые детализируют упрощенный метод обрамления пакета X.25 и формирование FR кадра. Этот метод может быть применен только с обоюдного предварительного согласия сторон, так как он не предусматривает каких-либо процедур (механизмов) сигнализации с целью оповещения ООД пользователя об использовании такого метода.

На рис. 8.4 представлена диаграмма многоуровневого управления доставкой пакетов X.25 на основе упрощенного метода формирования FR кадра.

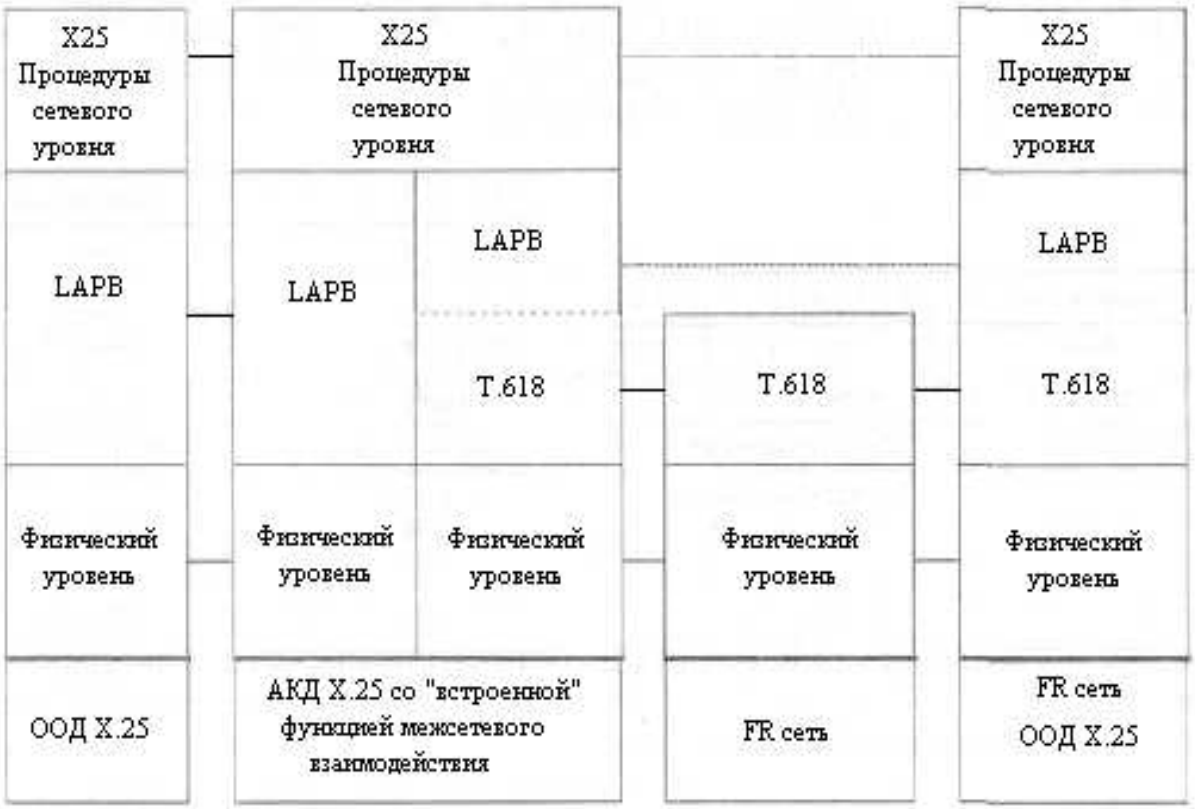



Рис. 8.4. Диаграмма многоуровневого управления доставкой пакетов X.25 на основе упрощенного метода формирования FR кадра

Априори считается, что взаимодействие на сетевом уровне осуществляется между абонентскими ООД, а на втором (канальном) уровне поддерживается сбалансированная процедура доступа к каналу (Link Access Procedure Balanced - LAPB). Кадр канального уровня (LAPB) обрамляется, и формируется FR кадр, который пересылается к месту назначения через определенный DLCI (при этом используются согласованные параметры и условия доставки по FR сети). На приемной стороне кадр канального уровня «освобождается» от обрамления FR кадра и направляется для обработки на канальный уровень управления (LAPB), а после этого, соответственно, - на сетевой уровень. Это означает, что протокол канального уровня (LAPB) взаимодействует с FR сетью так же, как и с физическим уровнем обычной сети X.25; при этом FR сеть "прозрачна" для всех типов кадров (информационных и управляющих, таких как RR, RNR, REJ и др.).

На рис. 8.5 представлен формат FR кадра, "переносящего" кадр канального уровня (кадр LAPB). Адресное поле, поле управления и информационное поле кадра канального уровня полностью без изменений вставляются в информационное поле FR кадра. Одним из характерных условий формирования такого FR кадра является то, что в этом случае в заголовке FR кадра не используется идентификатор типа протокола. Другое характерное условие касается применения стандартного 2-октетного адресного поля заголовка. И последнее условие относится к полю проверочной по-

следовательности (FCS): FCS кадра канального уровня заменяется FCS FR кадра, которая определяется с учетом всех полей - 2-октетного адресного поля заголовка FR кадра, адресного поля, поля управления и информационного поля кадра канального уровня. Соответственно на приемной стороне осуществляются процедуры, обратные указанным выше.

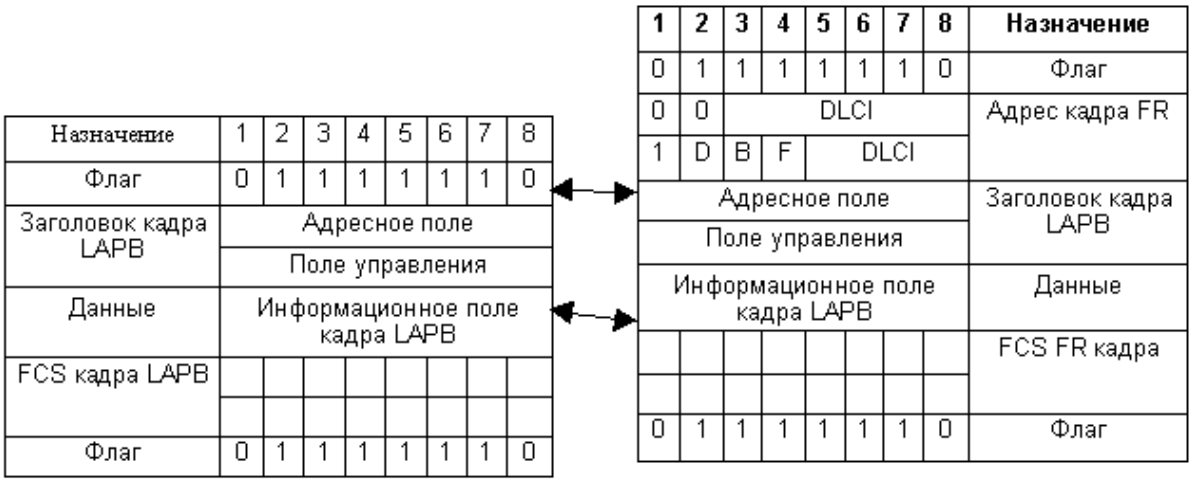



Рис. 8.5. Формат FR кадра, "переносящего" кадр LAPB

Все сказанное выше имело одну цель: разработка более простой схемы взаимодействия FR и X.25 сетей. Последствия такого упрощения, по мнению фирм-производителей, очевидны: улучшение качества обслуживания пользователей. При реализации такой схемы взаимодействия FR и X.25 сетей желательно выполнение следующего условия: максимальный размер "окна" на 2-м уровне (ЭМВОС) должен соответствовать рекомендованному значению (128). Это вызвано, в первую очередь, тем, что практика применения протоколов управления потоков на канальном уровне с использованием меньшего (чем 128) размера "окна" показала значительное снижение эффективности функционирования процедуры LAPB.

### *8.3. Ретрансляция кадров и речевой трафик*

Метод ретрансляции кадров разрабатывался как синхронный метод доставки данных в ЦСИО (и не только). Соответственно все реализующие этот метод механизмы и качество обслуживания определялись для различных видов графика, кроме речевого. Все традиционные сети с пакетной коммутацией, как правило, использующие различные способы коммутации пакетов и низкоскоростные каналы связи, не предусматривали возможность доставки сообщений, чувствительных к задержке. Другими словами, эти сети имеют большую и часто меняющуюся задержку доставки сообщений.

Известно, что такая задержка является функцией, с одной стороны, скорости коммутации в УС, а с другой - пропускной способности магист-

ральной линии связи. Значительное снижение задержки может быть достигнуто за счет применения метода ретрансляции кадров и магистральных линий связи с высокой пропускной способностью. Таким образом, FR сеть способна "транспортировать" чувствительный к задержкам трафик. Однако одно дело - передача такого типа трафика по сети с динамической маршрутизацией, а другое - обеспечение приемлемого качества обслуживания пользователей.

Одной из проблем, связанных с передачей речевого трафика, является то, что скорость такой передачи должна быть постоянна. Это следует из того, что вся информация, содержащаяся в оцифрованном речевом сигнале (импульсно-кодовая модуляция - ИКМ), передаваемом со скоростью 64 Кбит/с, важна и необходима для восстановления исходного речевого сообщения на приемной стороне. Вместе с тем в настоящее время разработаны методы и способы снижения требуемой полосы пропускания оцифрованного речевого сигнала, среди которых:

*Компрессия (сжатие речевого сигнала)*. Этот метод позволяет снизить скорость передачи с 64 до 8 Кбит/с и ниже. Во многих известных мультиплексорах (аппаратуре временного объединения (разделения) каналов) реализованы алгоритмы, позволяющие уменьшить такую скорость. (На самом деле нижний предел сжатия речевого сигнала пока еще не достигнут, исследования в этой области продолжаются и по сей день.) Конечно, дальнейший рост компрессии (то есть снижение скорости передачи) начинает сказываться на качестве восстанавливаемого речевого сообщения. Однако человеческое ухо способно уловить и распознать речь, которая была подвергнута очень сильному сжатию.

*Детектирование шума (речевых пауз)*. Исследования показывают, что типичная человеческая речь включает до 60...70% пауз, которые необходимо детектировать. Последняя процедура нужна прежде всего для того, чтобы исключить передачу "бесполезной" информации через сеть и тем самым обеспечить высокую эффективность функционирования сети.

Эти, а также другие методы могут быть с успехом использованы в процессах пакетирования оцифрованных речевых сообщений. Поэтому в настоящее время активно проводятся работы по их стандартизации и внедрению в различные службы передачи речевого трафика в пакетной форме. Большинство проблем стандартизации в данной области связано с "природой" самих сетей с пакетной коммутацией. В первую очередь это относится к нумерации пакетов, которая необходима для обеспечения гарантированной доставки пакетов в их естественной последовательности. Более того, различные пакеты могут иметь и различные внутрисетевые задержки, так как возможны различные экстремальные ситуации, возникающие при отказе линий и узлов связи, перегрузках и блокировках и т.п.

ITU-T принял Рекомендацию G.764, которая определяет механизм сегментирования оцифрованного речевого сигнала и формирование соот-

ветствующих пакетов. Однако этот стандарт содержит много проблем, которые послужат дальнейшему поиску новых решений в области передачи речевого графика в сетях с пакетной коммутацией. К этим проблемам относятся:

*Детектирование шума с целью снижения входного трафика* Эта рекомендация детализирует процедуры анализа входного речевого трафика, детектирования пауз и передачу последних через сеть в форме синхронизирующих последовательностей для определения начала и окончания речевых и неречевых последовательностей;

*Нумерация последовательностей пакетов, обеспечивающая доставку последних в их естественной последовательности* В случае потери пакета возможно одно из двух решений:

а) повторная передача пакета из УС-источника (что резко повышает общесетевую задержку);

б) передача паузы к УС-получателю в том месте последовательности, где должен был находиться утерянный пакет;

*Задержка при обеспечении синхронизма, цель которого -исключение нарушений в обслуживании пользователей* Процедура синхронизации представляет собой тактирование каждого пакета, при передаче которого каждый УС вносит свою индивидуальную транзитную задержку. На приемной стороне все входящие пакеты накапливаются в буфере и поступают в ООД абонента с постоянной задержкой.

С Рекомендацией G.764 тесно связана Рекомендация G.727, которая определяет процедуры обработки речевого сигнала в соответствии с алгоритмом адаптивной дифференциальной ИКМ. (АДИКМ) и вводит понятия "информационные" и "дополнительные служебные" биты.

Рекомендация G.727 вводит механизм разделения "речевого" пакета на составные части, в одной из которых размещаются информационные биты, а в другой - дополнительные служебные биты. Целью такого разделение является обеспечение возможности уничтожения (при необходимости) дополнительных служебных битов при доставке "речевых" пакетов, что приводит к уменьшению длины последних. А это в свою очередь способствует снижению возможной сетевой нагрузки.

Базовая FR сеть должна учитывать все сказанное выше, а также обеспечивать:

- требуемое качество обслуживания, что подразумевает малую вероятность ошибки и предоставление пользователю минимально необходимой пропускной способности. Сеть должна обеспечивать доставку пакетов, содержащих информационные биты (АДИКМ), в ООД абонентов при любых условиях функционирования;
- возможность обслуживания пользователей, имеющих различные приоритеты. Это вызвано тем, что чувствительный к задержке график должен иметь наивысший приоритет. При этом сеть должна приоста-

навливать передачу другого графика (сообщений, находящихся в выходной очереди). Это важное свойство сети пока не отражено в международных стандартах, и его реализация полностью зависит от фирм-производителей аппаратно-программных средств для FR сетей;

- функционирование специальных процедур, с помощью которых уничтожаются дополнительные служебные биты и одновременно с этим защищаются информационные биты. Это позволит избежать негативных последствий, связанных с сетевой перегрузкой, которая будет снижать качество речевого сообщения;
- функционирование методов детектирования речевых пауз и/или компрессии речевого сигнала (в точках доступа). Это позволит минимизировать объем графика, передаваемого в сети;
- уменьшение максимального размера кадров (наиболее вероятно 128 октетов) от других абонентов сети (не речевой трафик). Это позволит избежать появления задержек, связанных с нахождением в очереди на передачу (в УС) очень длинных кадров. Однако это требование противоречит основной цели применения сетей с ретрансляцией кадров, в соответствии с которой последние выступают в качестве транспортной среды ЛВС, использующих, как правило, кадры больших размеров;
- достаточно большую скорость передачи в магистральных линиях связи с целью уменьшения задержки, связанной с распространением сигналов. Эта скорость должна быть равна 2,048 Мбит/с и выше.

Если эти условия выполнены и речевым кадрам действительно присваивается наивысший приоритет, сеть обеспечивает низкую вероятность ошибки на бит, а также реализует методы передачи только информационных бит, то в этом случае существует возможность передачи речевого трафика через FR сеть.

FRF принял только один стандарт для FR сетей, специализирующихся на передаче речевого графика. Этим самым была предпринята попытка "отображения" Рекомендации ITU-T G.764, определяющей стандарты для пакетирования речевого графика, в стандарты FR. На рис. 8.6 представлен механизм "отображения" пакета G.764 в кадр FR. Пакет G.764 включает две части, в первой из которых размещены информационные биты, а в другой - служебные. Следовательно, этот пакет может быть "вложен" в два кадра FR, один из которых включает заголовок кадра и информационные биты, а другой - заголовок кадра и служебные биты. АКД сети всегда установит такое значение CIR, при котором кадры с информационными битами будут гарантированно доставляться через сеть. В кадрах с дополнительными битами бит *DE* (Discard Eligible - бит, указывающий на то, что данный кадр может быть уничтожен) будет всегда устанавливаться в "7" ООД пользователя (в точке доступа к сети - интерфейс ИПС), и, следовательно, они будут "восприниматься" сетью как не требующие выделения дополнительного ресурса пропускной способности. По существу, такие

кадры будут передаваться сетью и только при необходимости уничтожаться.

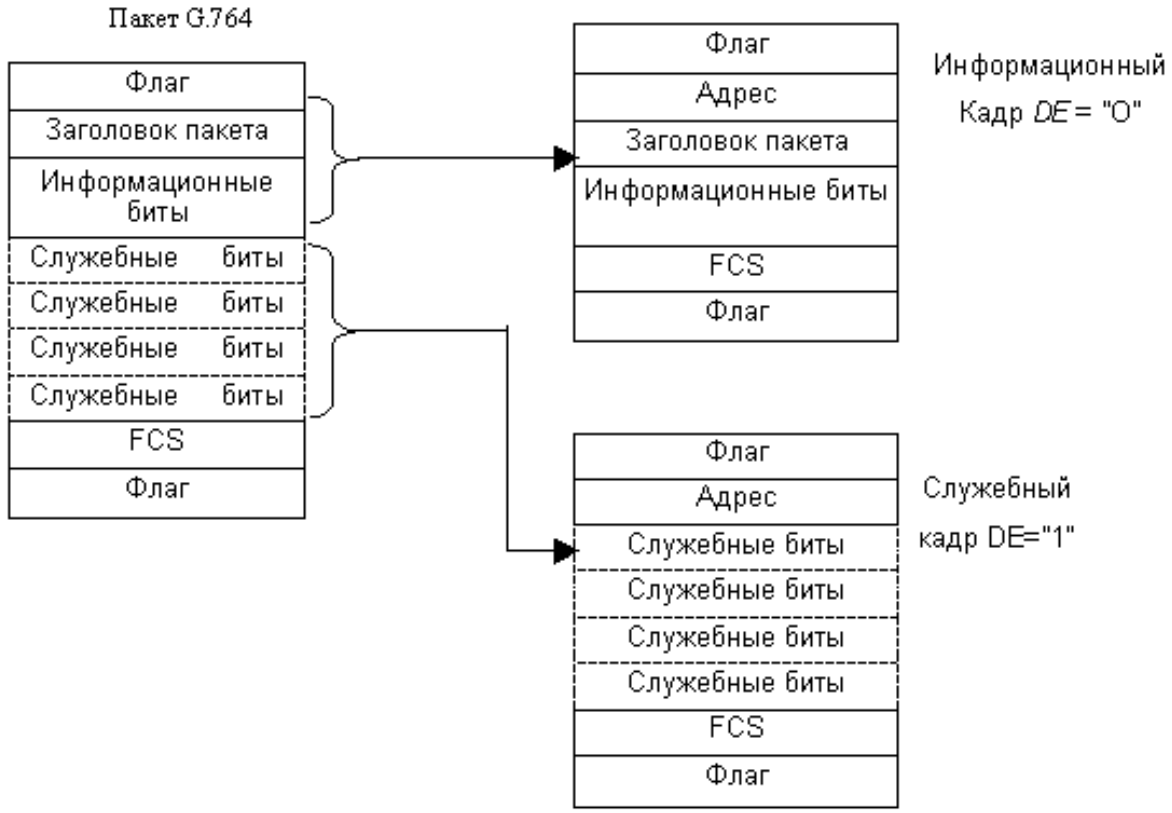



Рис. 8.6. "Отображение" пакета G.764 в FR кадры

При таком подходе возможна передача речевого графика через FR сеть, но при этом должны быть заранее оговорены все процедурные детали механизма доставки.

## 9. Организация доставки сообщений в широкополосных цифровых сетях интегрального обслуживания (ATM - Asynchronous Transfer Mode)

Тенденции развития систем передачи информации связаны с широким внедрением сетевой архитектуры в повседневную деятельность людей, ростом объемов передаваемой информации, возникновением новых проблемно-ориентированных служб, требующих для своей реализации каналы связи (КС) с пропускной способностью до $10^8$ бит/с, интеграцией различных служб в пределах одной СПД, организацией высокоскоростных (до $10^9$ бит/с) КС на основе волоконно-оптических, спутниковых и радиорелейных линий связи, повсеместным переходом на цифровые методы и средства передачи информации.

### *9.1. Широкополосная цифровая сеть интегрального обслуживания (Ш1-1СИО, B-ISDN - Broadband Integrated Services Digital Network)*

Наиболее полно упомянутые тенденции реализуются в концепции создания ЦСИО, развивающихся в направлении от узкополосных к широкополосным. Именно от внедрения последних ожидается наибольший эффект в области передачи информации, поскольку при этом будет создан прототип СПД, глобальной в отношении интеграции видов служб и географического расположения пользователей.

Согласно ITU-T ЦСИО - это цифровая сеть связи, в которой одни и те же устройства коммутации и каналы используются для установления соединений более чем одного вида служб. Под ШЦСИО понимается такая сеть интегрального обслуживания, в которой доступ пользователей обеспечивается к относительно широкополосным службам (высокоскоростной передаче данных, различным видеослужбам). К ШЦСИО предъявляются следующие требования:

- поддержка дистрибутивных и интерактивных видов служб; коммутация низкоскоростных и высокоскоростных КС;
- обеспечение как непрерывного (критичного к задержкам и потерям), так и прерывистого графика;
- обмен информацией в режиме установления соединения между абонентами либо без установления;
- рациональное с той или иной точки зрения распределение функций обработки сигналов по узлам и элементам сети;
- обеспечение взаимодействия пользователей по двухтерминальной или широковещательной схеме;
- гибкость в выборе скоростей передачи данных.

Такое обилие требований, часто взаимоисключающих, затрудняет разработку и внедрение как сети в целом, так и отдельных ее элементов.

В документах ITU-T, занимающегося проблемами международной стандартизации ЩЦСИО, для обозначения процесса уплотнения в КС фрагментов сообщений и их коммутации в узлах сети был введен термин "режим доставки" (transfer mode), который следует отличать от термина "режим передачи" (transmission mode, transmission), связанный с работой физической линии связи. Термин "режим доставки" применяется для описания способа, который используется в телекоммуникационной сети и включает в себя все аспекты, связанные не только с передачей, но и с мультиплексированием и коммутацией. Выбор режима доставки определяет структуру и параметры используемых сигналов.

### *9.2. Асинхронный режим доставки*

Асинхронный режим доставки (АРД), выбранный ITU-T в качестве основы для ШЦСИО, определяется как "обеспечивающий большой диапазон скоростей и малую задержку метод уплотнения и коммутации сообщений, представленных в пакетизированной форме, поддерживающей любую службу, независимо от того, ориентирована она на режим без установления соединения или с установлением соединения".

На рис. 9.1 иллюстрируется принцип АРД. Подлежащее передаче сообщение делится на фрагменты фиксированной длины по *m* бит, которые совместно с идентифицирующим адрес получателя заголовком длины *p* символов в заголовке образуют ячейку (cell).

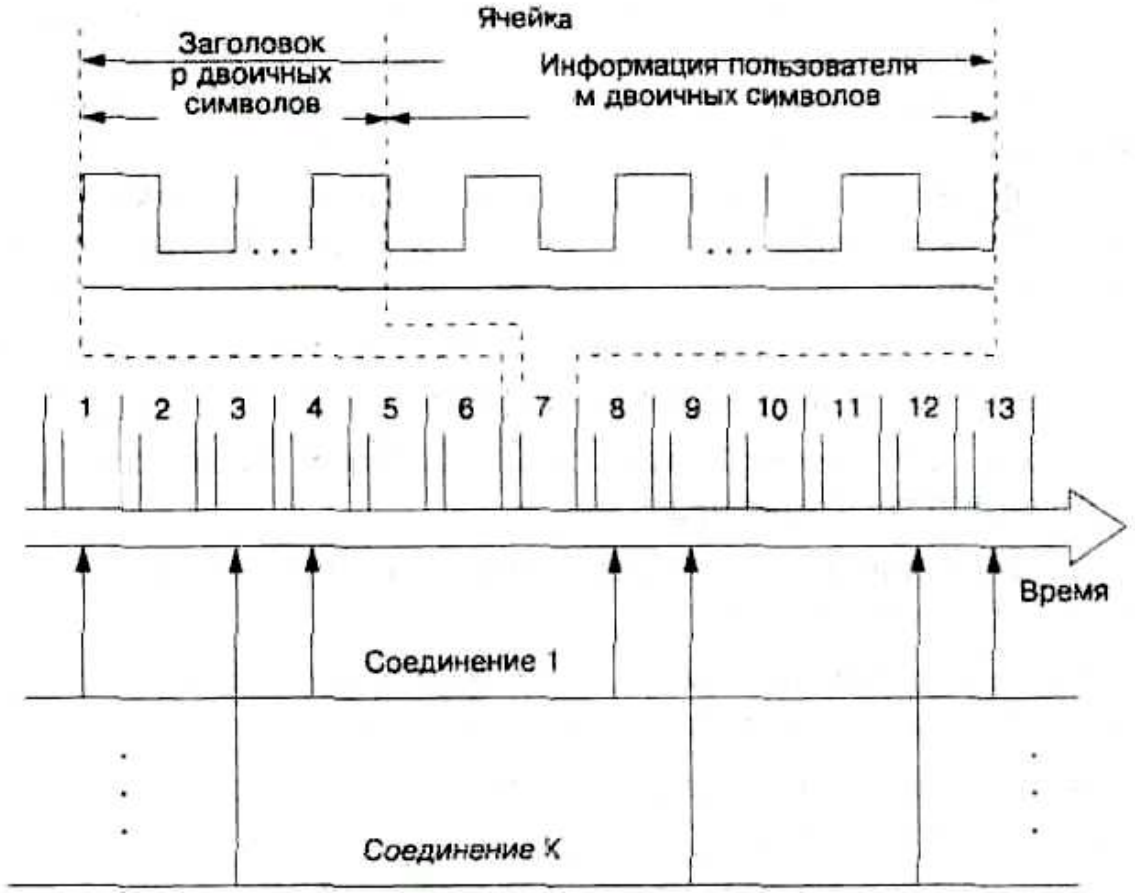



Рис. 9.1. Принцип асинхронного режима доставки на основе статистического уплотнения ячеек

Термин "асинхронный" означает, что ячейки, принадлежащие одному соединению, поступают в КС нерегулярно, поскольку временные интервалы предоставляются источнику сообщений в соответствии с его реальными потребностями. При этом следует заметить, что АРД реализуется при синхронном режиме работы КС, поскольку ШЦСИО планируется как синхронная цифровая СПД. Под синхронным режимом работы КС понимается такая организация его функционирования, при которой синхрони-

зация двоичных символов поддерживается аппаратурой канала независимо от того, передаются при этом данные пользователя или нет (при этом образуется виртуальный синхронный символьный тракт).

Наиболее полно концепция АРД описана в Рекомендации ITU-T 1.121 "Принципы широкополосной ЦСИО", где приведены эталонные модели ШЦСИО, иерархия протоколов, конфигурации интерфейсов и ряд общих положений.

В документах других международных и национальных организаций, в частности технического подкомитета T1S1 комитета T1 США, рассматриваются иные подходы к реализации АРД, отличающегося от предлагаемого ITU-T структурами сигналов, иерархией протоколов, организацией обмена данными в целом. Это свидетельствует о том, что не существует общего соглашения относительно ШЦСИО, отсутствие которого создает существенные трудности на пути создания глобальной СПД.

### *9.3. Эталонная модель ШИСИО*

Эталонная модель ШЦСИО на основе АРД как открытой системы, определенная в Рекомендации ITU-T 1.321, отличается от ЭМВОС наличием двух дополнительных уровней: уровня АРД и уровня адаптации, расположенных последовательно над физическим уровнем. Уровень АРД определяет все особенности режимов доставки, не зависит ни от топологии сети, ни от вида физической среды, ни от вида службы. Его функции определяются составом заголовка ячейки, перечислены в Рекомендации ITU-T 1.361 "Спецификация уровня АРД" и могут быть следующими: идентификация соединения; замена идентификатора соединения или его продолжение; уплотнение/разуплотнение ячеек; контроль (контроль и исправление) ошибок в заголовке; синхронизация ячеек; идентификация типа сообщения (пользовательское или служебное), переносимого ячейкой.

В отличие от уровня АРД уровень адаптации зависит от вида службы. Общее описание этого уровня и спецификация предоставляемых услуг определены в рекомендациях 1.362 и 1.363 соответственно. Основная его функция - сегментирование информационных блоков на части фиксированной длины. При этом в качестве сервисных блоков данных, то есть блоков, подлежащих делению, могут выступать как кадры канального уровня, так и блоки более высоких уровней эталонной модели. Так, на рис. 9.2, *а* отражено, что в ШЦСИО пользовательская информация сегментируется на транспортном уровне, то есть непосредственно в аппаратуре пользователя, тогда как служебная информация (информация контроля, управления и обслуживания) - на канальном уровне, что осуществляется в аппаратуре узлов доступа и транзитных узлов сети. На рис. 9.2, *б* представлена иерархия протоколов ШЦСИО при взаимодействии этой сети с традиционной СПД. В этом случае оба типа сообщений преобразуются в узлах коммутации.

Именно в преобразовании блоков данных, имеющих специфическую структуру для каждой из служб, к единому виду и заключается основной смысл термина "адаптация".


Рис. 9.2. Иерархия протоколов ЩЦСИО: а - для "чистого" АРД; б - при сопряжении ЩЦСИО и ЦСИО

### *9.4. Процедурная и логическая характеристики протокола АРД*

Схема преобразования цифрового сигнала на уровнях адаптации и АРД приведена на рис. 9.3. При передаче информации сервисные блоки данных (СБД) уровня адаптации делятся на сегменты фиксированной длины, образуя конечный набор протокольных блоков данных (ПБД), каждый из которых в свою очередь является СБД уровня АРД. На последнем к каждому СБД добавляется управляющая информация протокола (УИП) - заголовок. Образованный таким образом ПБД уровня АРД и есть ячейка. Процесс преобразования сигнала при приеме протекает в обратном порядке.

Как показано на рис. 9.2, ячейка АРД состоит из заголовка и информационного поля. При этом их размеры и соотношение определяют такие параметры СПД, как величина задержки на сборку/разборку ячеек, на очередь, на обработку в узлах коммутации, и в конечном счете сложность оборудования и стоимость сети. Удовлетворить требованиям по уменьшению всех видов задержки одновременно - задача трудновыполнимая. Оп-

тимальное значение того или иного параметра достигается при различных значениях длины ячейки и ее полей. Именно это при отсутствии соответствующих международных соглашений стало причиной многообразия структур ячеек, что влечет возможную несовместимость СПД, их использующих.

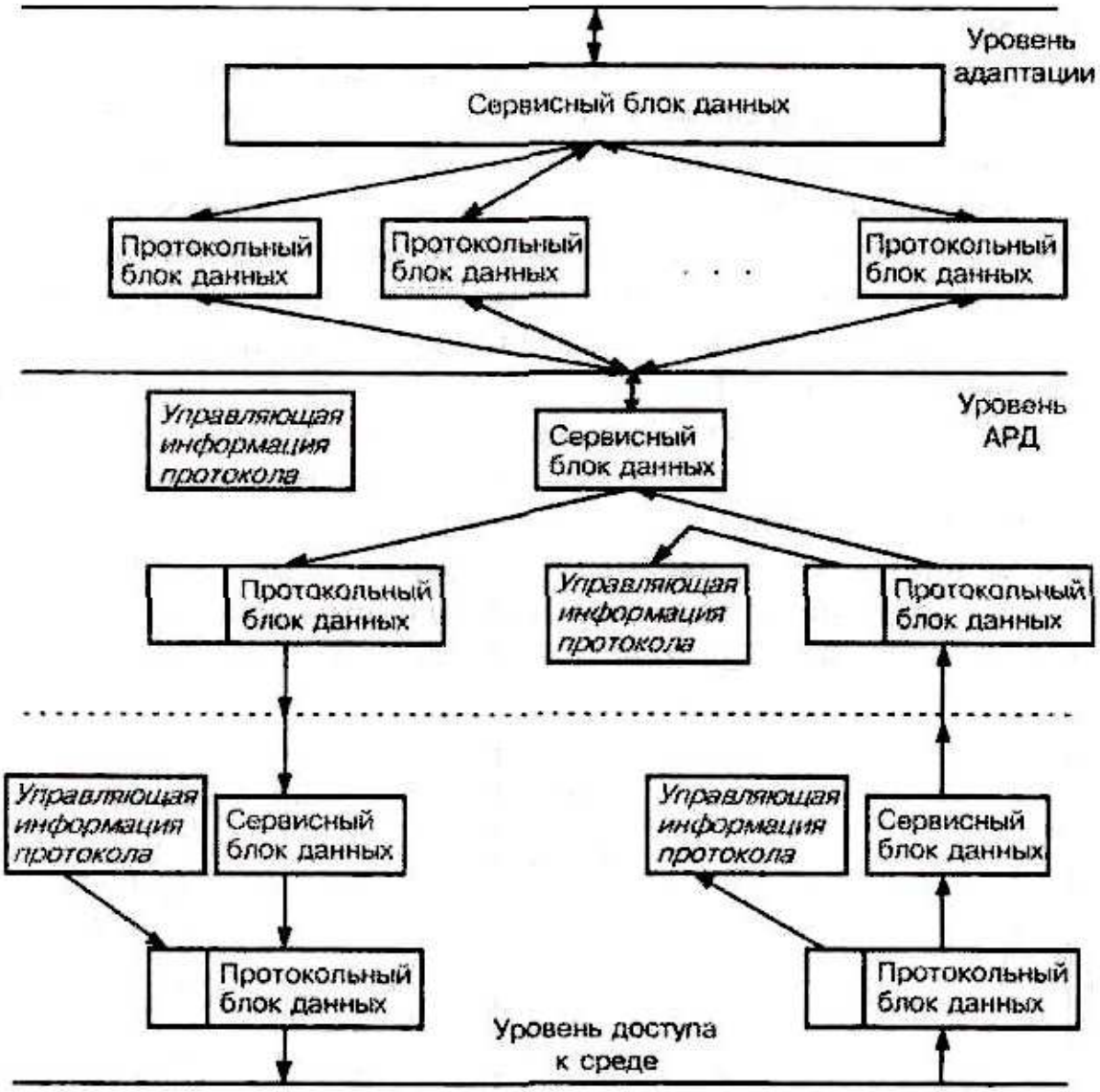
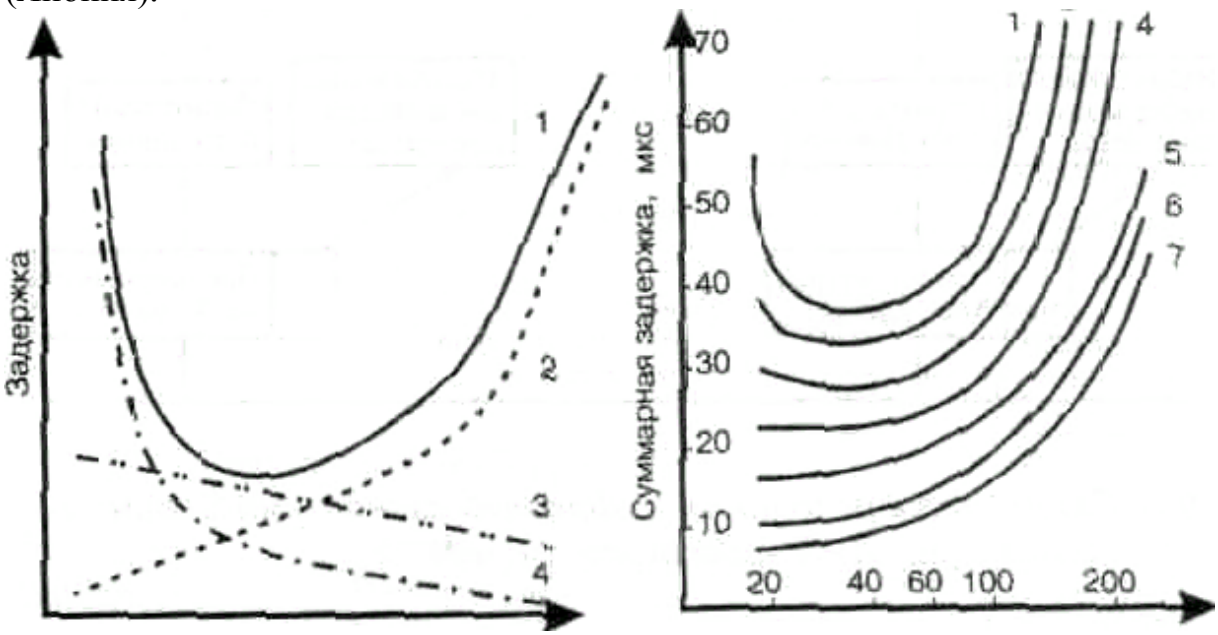



Рис. 9.3. Схема преобразования цифрового сигнала на уровнях адаптации, АРД, и доступа к среде

На рис. 9.4, а представлена зависимость времени задержки ячейки в узлах сети от ее длины, причем суммарная задержка определяется задержками на очередь, на распространение по КС и обработку в узлах ШЦСИО. Несмотря на отсутствие единственного оптимального значения длины ячейки, можно выделить область значений, в пределах которой суммарная задержка изменяется незначительно. Границы этой области различными исследовательскими организациями устанавливаются по-разному: 32...120 октетов, либо нижняя граница области снижается до 16 октетов, либо

верхний предел поднимается до нескольких сот октетов. Выбор единственного значения длины ячейки затрудняет и то, что сам вид суммарной зависимости (рис. 9.4, *а)* меняется при изменении степени загрузки сети, что иллюстрируется графиками, приведенными на рис. 9.4, *б.* В качестве оптимальных значений длин ячеек в различных проектах выбраны: 32 октета (Германия), 35 октетов (Бельгия), 36 октетов (Европейский институт по стандартизации в области связи - ETSI), 53 октета (ITU-T), 64 октета (Исследовательская лаборатория CCL США), 69 октетов (ANSI), 72 октета (Япония).




*а*) Длина ячейки *б*) Длина ячейки, дв. символы

Рис. 9.4. Зависимость задержки ячейки в ЩЦСИО от ее длины: *а*) для составляющих задержки, где 1 - суммарная задержка; 2 - на очередь; 3 - на распространение; 4 - на обработку; *б*) суммарная задержка при различных, степенях, загрузки сети, где 1 - загрузка 0,65; 2 - 0.7; 3 - 0,75; 4 - 0,8; 5 - 0.82; 6 - 0,835; 7 - 0,85

Еще одной причиной потенциальной несовместимости существующих проектов ШЦСИО являются различия размеров и структуры заголовка ячеек, обусловленные разнообразием подходов к определению перечня функций, возлагаемых на протокол уровня АРД. Эта причина не столько "технична", сколько обусловлена "различными философиями сетей будущего". Выделяют две крайние точки зрения. Согласно первой, принятой в Европе, АРД считается формой коммутации каналов, что обусловливает простоту заголовка и выбор режима виртуальных соединений для передачи информации. В этом случае большинство функций управления и контроля переносится на ООД, оборудование узлов сети упрощается, поскольку на них возлагается только функция маршрутизации ячейки с входящей линии на требуемую выходящую (при этом возможно обнаружение и исправле-

ние ошибок в заголовке), что удешевляет построение СПД. Внутри сети ячейки, принадлежащие различным службам, неразличимы. Заголовок каждой используется исключительно для идентификации соединения.

Изложенный подход реализован во Франции в рамках экспериментальной программы, в соответствии с которой ячейка состоит из 15 октетов информационного поля и одного октета поля заголовка, содержащего идентификатор соединения. Одним из итогов выполнения этой программы является вывод о необходимости увеличения длины ячейки до 35 октетов (32 - поле информации и 3 - заголовок, причем 6 двоичных символов которого содержат проверочную последовательность). Формат заголовка, предложенный ETSI (табл. 9.1), имеет длину 4 октета, из которых первые 3 - идентификатор соединения, а последний - проверочная последовательность (5 двоичных символов) и дополнительное поле, содержащее указатель типа сообщения, - один символ и два резервных символа. Формат ETSI также разработан в соответствии с принципами коммутации каналов.

Таблица 9.1

Формат заголовка ячейки АРД, предложенный ETSI

| Байты | Биты | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 1 | ИВП / ИВК | | | | | | | |
| 2 | | | | | | | | |
| 3 | | | | | | | | |
| 4 | ЦИП | | | | Тип | | Резерв | |

Другая точка зрения распространена в Северной Америке, где АРД рассматривается как упрощенная форма коммутации пакетов. При таком подходе на заголовок накладывается большое количество функций, что усложняет процесс обработки в узлах сети. Например, в документе технического подкомитета T1S1 комитета T1 США предлагается формат заголовка (табл. 9.2), включающий следующие поля: управление доступом, идентификатор виртуального канала, тип сообщения, приоритет службы и проверочная последовательность заголовка.

Таблица 9.2

Формат заголовка ячейки АРД, предложенный ANSI

| Байты | Биты | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 1 | Поле доступа | | | | | | | |
| 2 | ИВК | | | | | | | |
| 3 | | | | | | | | |
| 4 | | | | | Тип | | Резерв | |
| 5 | ЦИП | | | | | | | |

Существование различий по определению требований к АРД частично объясняется тем, что при одинаковой долговременной цели создания ШЦСИО - предоставление пользователю доступа к широкополосным службам будущего, - ближайшие цели внедрения этого режимы различны. Если в Европе требуется обеспечить совместимость ШЦСИО с существующими сетями общего пользования, функционирующими в большинстве случаев в режиме установления соединения, то в Северной Америке широкополосные сети должны осуществлять транспортировку данных в режиме без установления (подобно дейтаграммному режиму) между локальными и региональными сетями.

Объединить оба подхода - такова одна из целей ITU-T, реализованная в июле 1989 года в Женеве, в результате которой было достигнуто соглашение о некоторых параметрах АРД. В частности, предложен формат ячейки, содержащий заголовок длиной 5 октетов и поле информации - 48 октетов (табл. 9.3*)*. Для пользовательского интерфейса заголовок включает:

- поле общего управления потоком (4 бита), которое может использоваться в качестве поля управления доступом, как это предусмотрено в предложении комитета T1S1, в режиме без установления соединения, что не исключает возможности применения тех же ячеек при организации обмена информацией в режиме с установлением соединения;
- поле идентификатора (ИВП) виртуального пути (12 битов);
- поле идентификатора (ИВК) виртуального канала (12 битов);
- поле указателя типа сообщения (2 бита), используемое для служебных целей в интересах управления, эксплуатации и обслуживания сети;
- резервное поле (2 бита). Примечание: биты поля указателя типа сообщения и резервного поля могут быть при необходимости использованы для расширения ИВК до 16 двоичных символов;
- поле циклической избыточной проверки (ЦИП) заголовка (8 битов).

Таблица 9.3

Формат заголовка ячейки АРД, предложенный ITU-T

<table>
<tr><td rowspan="2">Байты</td><td colspan="8">Биты</td></tr>
<tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td></tr>
<tr><td>1</td><td colspan="4">Поле доступа</td><td colspan="4" rowspan="2"></td></tr>
<tr><td>2</td><td colspan="4">ИВП</td></tr>
<tr><td>3</td><td colspan="5" rowspan="2">ИВК</td><td colspan="3" rowspan="2"></td></tr>
<tr><td>4</td><td colspan="2">Тип</td><td colspan="2">Резерв</td></tr>
<tr><td>5</td><td colspan="7">ЦИП</td></tr>
</table>

### *9.5. Управление доступом*

Необходимость использования поля управления доступом возникает тогда, когда пользовательский интерфейс реализуется по схеме многотерминального соединения, что характерно для СПД, использующих моноканал (локальные сети, а также сети, задействующие радиолинии). При этом в протокол физического уровня ЭМВОС вносятся дополнительные функции управления доступом к среде, часто выделяемые в соответствующий подуровень. Поле доступа предназначено для размещения УИП этого подуровня (схема преобразования сигнала на нем выделена пунктиром на рис. 9.3). Если в комитете T1 полагают, что пользовательский интерфейс должен реализоваться именно по этой схеме, то в ITU-T многие делегации придерживаются мнения о необходимости стандартизации только двухтерминального соединения. Приведенное выше описание предложенного ITU-T формата заголовка ячейки учитывает обе точки зрения и может обеспечить глобальную совместимость ШЦСИО. Рекомендация ITU-T 1.361 определяет, что в сетевом интерфейсе при двухтерминальном соединении поле общего управления потоком не используется и отводится для расширения адреса.

### *9.6. Идентификаторы виртуального пути и виртуального канала*

Для создания гибкой, сравнительно дешевой СПД, какой и планируется быть ШЦСИО, необходимо по возможности упростить архитектуру сети и процесс обработки служебной информации в ее узлах. Этих целей достигают за счет использования концепции виртуального пути для управления виртуальными каналами на транзитном участке ШЦСИО. Если под виртуальным каналом подразумевается логическое соединение сетевого уровня, которое может быть динамически установлено и разъединено, то термином "виртуальный путь" обозначают совокупность таких соединений, имеющих в пределах некоторой транзитной (магистральной) сети общую пару терминалов. Последние непосредственно соединяются с коммутационными системами, межсетевыми преобразователями локальных сетей общего пользования или частных сетей. Такое объединение виртуальных каналов позволяет упростить процессы установления, обработки и управления каждым отдельным соединением и мультиплексированием их совокупности. Поэтому в рамках упомянутой концепции номер виртуального канала состоит из двух частей: ИВП и ИВК.

ИВП может присоединяться к каждой ячейке или группе ячеек, причем в последнем случае формируется либо кадр фиксированной длины, либо объединенная ячейка переменной длины (рис. 9.5).

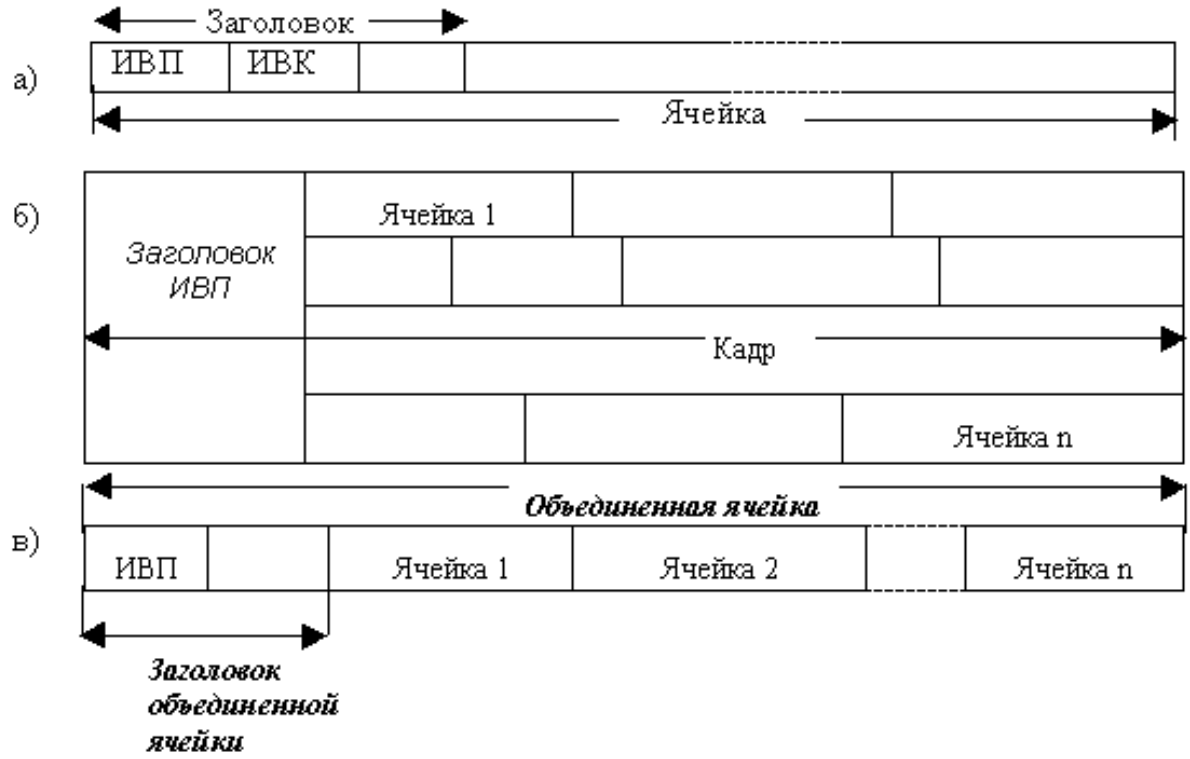



Рис. 9.5. Присоединение ИВП: а) к каждой ячейке; б) к кадру фиксированной длины; в) к объединенной ячейке

Хотя два последних метода уменьшают количество переключении коммутатора, что важно при скоростях передачи информации порядка $10^8$ бит/с и более, реализуемых в ШЦСИО, оба требуют расширения буферной памяти того же коммутационного узла, что увеличивает задержку. Кроме того, переменная длина объединенной ячейки усложняет процесс обработки служебной информации в узле. Поэтому наиболее распространен первый метод, а в сетевом интерфейсе применяется и второй.

Различают две схемы формирования номера виртуального канала, объединяющего ИВП и ИВК: точную и неточную. В первой схеме упомянутые идентификаторы неразделимы, точная граница между ними в пределах соответствующего поля заголовка не устанавливается, и любой узел сети использует весь номер виртуального канала для определения маршрута ячейки. Эта схема реализована в форматах заголовков, предложенных T1S1 и ETSI. Во второй схеме для обоих идентификаторов выделяются отдельные поля в заголовке ячейки, каждое из которых избирательно используется либо транзитным, либо конечным узлом сети.

### *9.7. Служба приоритетов*

До сих пор остается предметом обсуждения необходимость введения службы приоритетов в ШЦСИО. При ее реализации проблема коммутации будет решаться на основе следующих принципов:

- задержка ячейки с более низким приоритетом для передачи ячейки с более высоким приоритетом;
- уничтожение ячейки с более низким приоритетом при необходимости передачи ячейки с более высоким приоритетом.

В соответствии с этими принципами выделяют следующие типы приоритетов. Приоритет по задержке (delay priority) введен для упорядочения очереди ячеек в буфере узла коммутации. Более высокий приоритет обеспечивает (в среднем) меньшую задержку передачи сообщения от одного терминала сети до другого. Однако изменение приоритета по задержке в пределах одного виртуального канала может привести к изменению последовательности прихода ячеек к пользователю, а следовательно, и к искажению сообщения или увеличению задержки передачи сообщения в целом.

Другим типом является приоритет по потере (loss priority) ячеек, введенный для упорядочивания процесса уничтожения ячеек при аварийных ситуациях сети, возникающих главным образом при перегрузке буфера узла. Именно этот тип приоритета ячеек предусмотрен в предлагаемой ITU-T концепции ШЦСИО, указатель которого занимает один из двоичных символов резервного поля заголовка. Однако для этой цели могут использоваться и более сложные, чем механизм приоритетов, алгоритмы.

Существуют различные модели процесса присвоения приоритетов:

- детерминированные, когда приоритет либо присваивается на этапе установления соединения, как это планируется в программе ETSI, либо постоянно закреплен за конкретным видом службы, пользователем (например, высокий приоритет по задержке для специфической группы пользователей на уровне правительственных или военных учреждений);
- динамические, когда приоритет пересчитывается в зависимости от фактической величины задержки отдельного сообщения (так называемый относительный приоритет ячеек);
- выделение окон, что соответствует резервированию;
- задержки ячейки или более крупного сегмента на цикл коммутации
- различные режимы организации очередей.

Не все перечисленные модели требуют при своей реализации введения в заголовок ячейки поля указателя приоритета. Поэтому формат заголовка ячейки, предложенный ITU-T, и предусматривает возможность расширения адресного поля (ИВП, ИВК) до 28 битов за счет полей указателей типа сообщений и приоритета (резервное поле).

### *9.8. Защита заголовка ячейки АРА (циклическая проверка)*

Проверочная последовательность, расположенная в одноименном поле заголовка, предназначена для обнаружения или для обнаружения и исправления ошибок, возникающих в процессе передачи ячейки по КС ШЦСИО. Как правило, организуется циклическая избыточная проверка, известная из традиционного пакетного режима доставки, поскольку она обеспечивает лучшее по сравнению с другими методами качество решения этой задачи. Необходимость введения такой проверки обусловливается тем, что случайное искажение хотя бы одного символа ИВК влечет за собой потерю всей ячейки. Потеря ячейки в свою очередь может привести к потере части или даже всего сообщения соответствующей службы. ЦИП позволяет исправлять как минимум одиночные ошибки и обнаруживать их пакеты. Длина ЦИП выбирается исходя из планируемой длины заголовка. Так, ЦИП длиной 5 битов обнаружит одиночную ошибку в блоке из 24 битов, ЦИП длиной 6 битов - в блоке из 32 битов. Более длинные ЦИП обладают большими возможностями. Вид проверочного полинома определяется аппаратурой транзитных узлов коммутации ШЦСИО и стандартизации в отличие от полиномов, используемых в кадрах протоколов канального уровня традиционных СПД, еще не подвергался.

### *9.9. Принципы информационного обмена и синхронизация в АРА*

В соответствии с концепцией ШЦСИО АРД основывается на двух типах ячеек: пользовательских, то есть ячеек, информационное поле которых содержит фрагмент сообщения пользователя, и служебных, не переносящих пользовательскую информацию.

Наличие ячеек второго типа является необязательным. Их основная функция - определение границ информационных ячеек - по целям аналогична синхронизации кадров в пакетном режиме доставки и может быть выполнена иными способами.

В Рекомендации ITU-T 1.121 описаны принципы обмена сообщениями между двумя пользователями при организации между ними виртуального канала, которые определяют основу функционирования ШЦСИО. Схема формирования последовательности ячеек представлена на рис. 9.7 и включает:

- ООД передает цифровой сигнал, синхронизованный местными часами в КС;
- сборщик пакетов буферирует этот сигнал, чтобы сгладить пиковые нагрузки и образовать стандартный информационный блок - ячейку;
- ячейка считывается под управлением сетевых часов и ее структура соответствует требованиям сети;

- ячейки, поступающие от различных комплектов ООД, объединяются в канале связи, образуя групповой сигнал, и коммутируются в узлах сети;
- в точке приема ячейки буферируются для компенсации различия задержек распространения;
- разборщик пакетов восстанавливает переданный сигнал, который считывается оконечным оборудованием данных в пункте приема под управлением местных часов.

Ячейки, передаваемые по КС ШЦСИО, который является синхронным символьным трактом, следуют без специальных разделителей. Начало каждой определяется временным положением ее первого символа в цифровом потоке. Для выделения отдельных ячеек из общего потока их необходимо синхронизировать. Как правило, синхронизация выполняется с помощью синхронизирующей последовательности (СП) двоичных символов, размещение которой в передаваемом сигнале определяется двумя подходами: непериодическое (рис. 9.6, *а*) размещение, когда интервалы следования СП различны, и периодическое (рис. 9.6, *б*), когда СП следуют через равные интервалы времени.

Статистическое машинное моделирование работы мультиплексора ячеек показывает, что его средняя загруженность никогда не бывает равной 100% и обычно не превышает 85%. Это позволяет реализовать первый подход следующим образом: СП вставляется в цифровой поток в тот временной интервал, когда буфер мультиплексора пуст и информационная ячейка не передается. Если же поток информационных ячеек становится слишком длинным, что определяется используемой аппаратурой канала связи, то СП вставляется директивно. Очевидно, такой подход практически не вносит дополнительной задержки, что существенно для ШЦСИО и обусловливает его применение.

Принцип синхронизации ячеек при их приеме заключается в следующем: каждый раз, когда обнаруживается СП (по методу сопоставления с образцом), счетчик обнуляется и ведет счет до тех пор, пока не поступит следующая СП. Алгоритмы синхронизации при этом базируются на двух операциях:

- обнаружение двух последовательно следующих СП;
- оценка расстояния между ними, которое должно быть кратным длине ячейки.

При периодическом размещении СП вставляется в передаваемый сигнал через равные промежутки времени.

Если второй подход уменьшает требуемое время на обработку в узле сети, то первый сокращает временные затраты на выравнивание ячеек при сбоях в КС. Поэтому предпочтительнее комбинировать оба подхода (рис. 9.6, *в*).

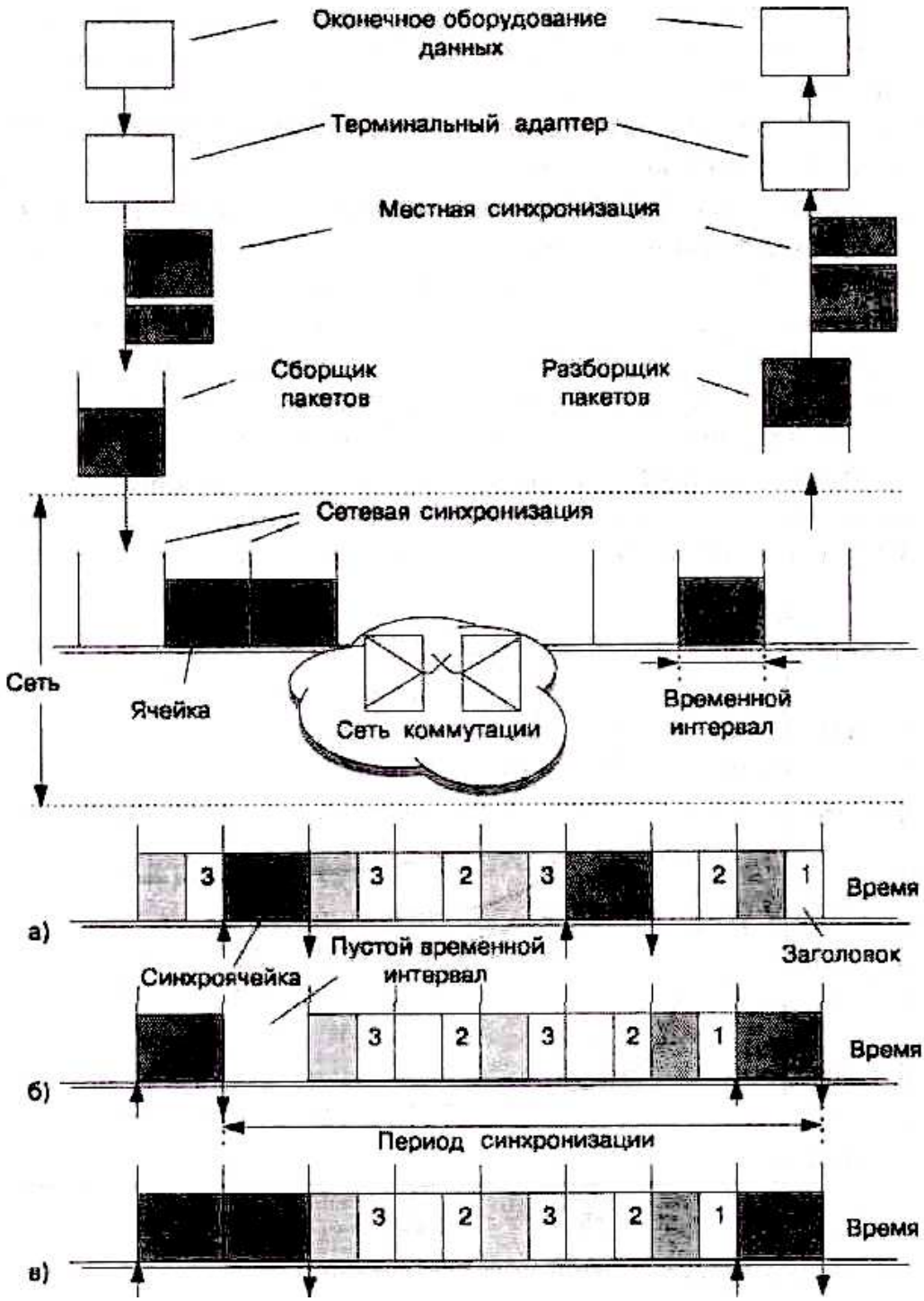



Рис. 9.6. Схема формирования последовательности ячеек. АРД: *а*) синхронизируемой непериодически; *б*) периодически; *в*) комбинированно

На рис. 9.7 представлена зависимость требуемого интервала следования СП от ее длины при различных вероятностях удержания синхронизма для следующих условий: загрузка сети информационными ячейками - 70%, длина ячейки - 72 октета, максимальный интервал следования синхропоследовательностей - 33 ячейки. Видно, что для $P$ = 0,95 увеличение длины

СП более 6...18 двоичных символов не приводит к уменьшению требуемого интервала синхронизации, а следовательно, и не является необходимым.

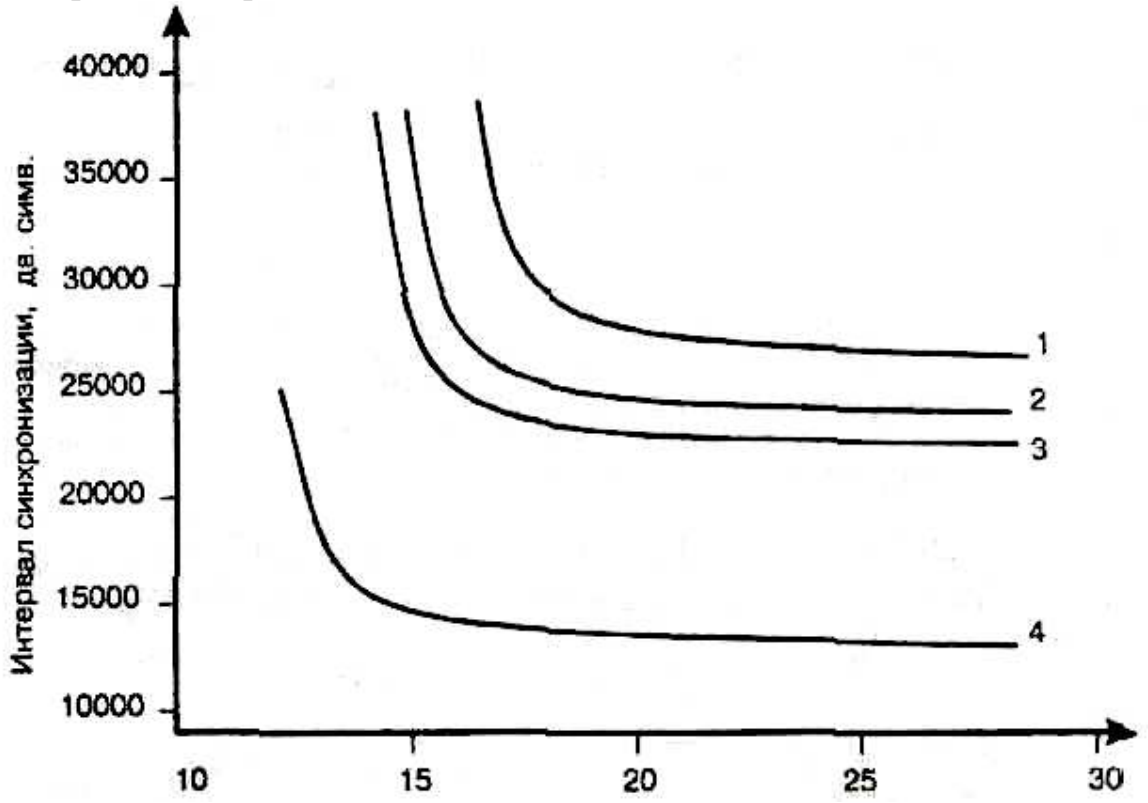



Рис. 9.7. Зависимость требуемого интервала следования синхронизирующей последовательности от ее длины при различных значениях вероятности удержания синхронизма: 1 – $P_c$= 0,99; 2 – $P_c$= 0,95; 3 – $P_c$ = 0.90; 4 – $P_c$ = 0,5

Однако такое уменьшение длины повышает вероятность имитации СП, поэтому необходимо определять значение ее длины с учетом обоих факторов. Возможно применение для синхронизации и уникальных последовательностей (например, некоторые последовательности Баркера), что позволит еще более сократить их длину без ухудшения качества процесса синхронизации. Но это потребует введения дополнительного преобразования ячеек, в частности кодирования со вставкой символов, что усложнит процесс обработки в узлах сети и увеличит задержку доставки сообщений.

Поскольку алгоритмы синхронизации базируются на оценке расстояния между двумя СП, то в КС ШЦСИО должны предъявляться жесткие требования к качеству тактовой синхронизации, поскольку любой ее сбой может привести к потере информационных ячеек. Для решения этой проблемы можно использовать скремблирование. При этом скремблируется весь передаваемый сигнал, за исключением участка, содержащего СП и указатель, выполняющий функцию установки скремблера/дескремблера.

На рис. 9.6 (*а, б, в*) стрелками показаны моменты установки и сброса скремблера для случая, когда длина СП совпадает с длиной ячейки.

Для улучшения качества восстановления тактовой синхронизации широкополосной сети помимо скремблирования рассматривалось использование канального кодирования. Однако этот подход часто связан с необходимостью расширения используемой полосы частот, что затрудняет его применение в ШЦСИО.

# 10. Сравнение сетевых архитектур

В связи с развитием компьютерных технологий разработка сетей усложнилась. Архитектура используемых сетей должна измениться так, чтобы соответствовать требованиям распределенных приложений. При этом новые архитектуры часто предполагают использование новых технологий. К сожалению, нередко разработчики сетей бывают плохо о них осведомлены, что затрудняет выбор наиболее подходящей технологии и оценку эффективности сети. В разделе рассматриваются сетевые разработки и проводится сравнительный анализ сетевых технологий.

## *10.1. Требования к современным компьютерным сетям*

"Сеть - это компьютер", гласит девиз фирмы Sun, в котором нашли свое отражение веяния нашего времени. Представление о сети как о "трубопроводе", по которому передается информация от компьютера к компьютеру, безнадежно устарело. При более пристальном взгляде на современные информационные технологии оказывается, что сеть - это, прежде всего, основа для работы программного обеспечения. Разработчики программ уже не могут не учитывать связи между компьютерами, которые обеспечивает сеть. Это заставляет владельцев вычислительных систем пересмотреть стратегию создания сети и работы с ней. Современные вычислительные сети не только обеспечивают связь между компьютерами, но и являются основой для распределенных вычислений.

Какой же должна быть современная компьютерная сеть? Специалисты, занимающиеся разработкой вычислительных систем, и сетевые администраторы стремятся обеспечить выполнение трех основных требований, предъявляемых к сети, а именно:

- хорошей масштабируемости,
- высокой производительности,
- управляемости.

Хорошая масштабируемость необходима для того, чтобы можно было менять число пользователей, работающих в сети, или количество программ, которые в ней используются. Высокая производительность сети требуется для ускорения работы программ. И наконец, сеть должна быть

управляемой, чтобы ее можно было легко перенастроить для удовлетворения постоянно меняющихся потребностей современных предприятий.

Эти требования отражают новый этап в развитии сетевых технологий - этап создания высокопроизводительных сетей. Большинство организаций с успехом прошло через более ранние этапы установления связи между компьютерами и создания базового взаимодействия между основными системами. Теперь им необходимо решить проблему развертывания в производственной среде сетевого программного обеспечения непрерывного действия. При использовании сети необходимо, чтобы программы (и сетевая инфраструктура) могли обеспечивать большое количество операций "клиент-сервер". Если же архитектура сети не предназначена для распределенных вычислений, то система будет работать неэффективно. Поэтому многие организации, используя передовые информационные технологии, стремятся добиться максимальной производительности своей сети.

Так ли уж важна производительность? Задержки в компьютерной сети измеряются сотыми или даже тысячными долями секунды. Но задержка на одну сотую долю секунды может сильно замедлить работу пользователей, если она будет повторяться тысячу раз во время каждой сетевой операции. Огромное количество таких ничтожных, на первый взгляд, задержек происходит и в современных клиент-серверных системах, что может существенно снизить эффективность работы сети. Небольшая разница в скорости, обеспечиваемой различными сетевыми технологиями (мостами, маршрутизаторами, коммутаторами и т.д.), приводит к серьезным различиям в важных характеристиках работы сети, в которой использованы эти технологии, - в скорости реакции программного обеспечения и возможном количестве пользователей.

Специалисты по проектированию сетей могут, конечно, несколько увеличить число пользователей, тщательно настроив сеть на одну конкретную программу. Однако это обычно вредит другим вычислительным процессам. Например, если настроить сеть для передачи больших файлов, то большее число пользователей получит возможность одновременно заниматься обработкой изображений. Но такая настройка снизит производительность программ классического клиент-серверного типа (например, при работе с базами данных). Поэтому при выборе архитектуры вычислительной системы нужно учитывать, как и в каком режиме будет использоваться сеть.

### *10.2. Примеры сетевых архитектур*

Своеобразие новых программ и технологий усложняет разработку вычислительных систем. Централизованные ресурсы, новые классы программ, новые принципы их применения, изменение информационных потоков, увеличение числа одновременно работающих пользователей и более

мощные вычислительные платформы - все эти факторы нужно учитывать при разработке компьютерной сети. Создано большое количество технологических и архитектурных решений, и выбрать из них наиболее подходящее - достаточно сложная задача.

Различные сетевые технологии обеспечивают разное время реакции и общей пропускной способности сети (рис. 10.1). Технологии коммутации кадров и ячеек позволили увеличить пропускную способность сети, дали возможность передавать большие объемы данных за короткое время. Такие технологии, как ATM и Ethernet, обеспечивают пропускную способность сети в диапазоне от 10 до 100 Мбит/с и выше. В ЛВС и глобальной сети диапазон пропускной способности составляет шесть порядков (от 1 тыс. бит/с до 1 млрд бит/с), а сетевых задержек - четыре порядка (от 1 с до 1 мс). Для сравнения - различие в скорости передачи данных между X.25 и ATM-коммутацией в ЛВС сопоставимо с разницей в размерах дома на одну семью и Североамериканского континента. К сожалению, трудно определить, каким может быть максимальное число пользователей и каково будет время реакции сети на их запросы при использовании каждой из этих технологий.

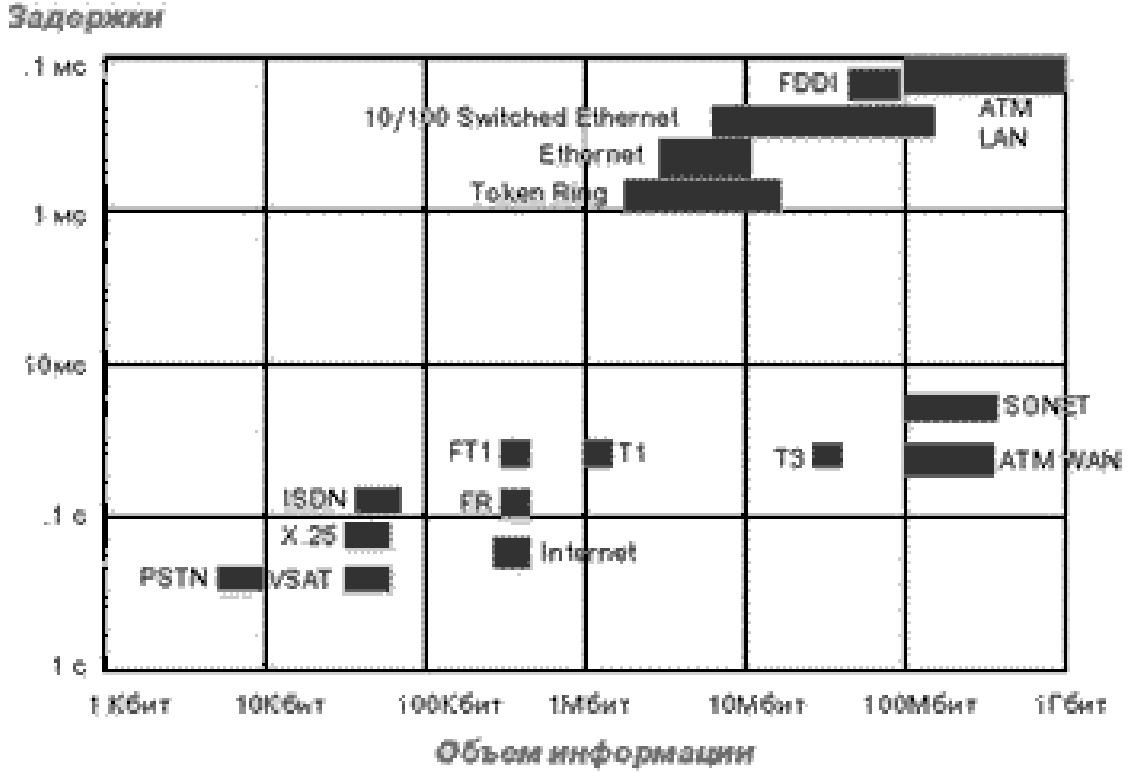



Рис. 10.1. Ландшафт производительности

Как же выбрать подходящую сетевую архитектуру? Общих рекомендаций нет. Требования к сети, обусловленные используемым программным обеспечением, так же многочисленны и разнообразны, как сами програм-

мы. Мы рассмотрим некоторые сетевые архитектуры, используемые при проектировании:

- маршрутизируемая фрагментированная магистраль;
- FDDI-магистраль;
- сеть с коммутацией кадров 10/100;
- ATM и коммутация кадров.

Первые две конфигурации считаются стандартными и уже давно используются в компьютерных сетях, две последние разработаны недавно. Современные технологии, основанные на коммутации, позволяют не только повысить производительность сети, но и улучшить ее управляемость. Это становится возможным за счет так называемой виртуализации сетевых ресурсов, которая позволяет создавать логические группы пользователей и компьютеров. Такие логические группы более наглядны, их проще поддерживать и изменять, чем физические подсети, определяемые маршрутизатором. Кроме того, коммутация кадров повышает удобство использования уже имеющейся сети. В тех случаях, когда коммутация кадров 10/100 сопоставима с FDDI-магистралью, она допускает непосредственное подключение к любому устройству, имеющему интерфейс Ethernet или Token Ring. Создание же FDDI-магистрали требует дополнительных вложений в маршрутизаторы и FDDI-интерфейсы. Поэтому реализация архитектуры на основе коммутации, вероятно, обойдется дешевле.

Маршрутизируемая фрагментированная магистраль (рис. 10.2) состоит из маршрутизатора среднего класса, поддерживающего различные протоколы и связывающего рабочие группы Ethernet с сегментом центрального сервера или серверов. Такая архитектура позволяет поддерживать связь между магистралью и рабочей группой. Достоинствами маршрутизируемой магистрали являются отличное управление протоколами, разделение рабочих групп и простота обслуживания. Для настройки всех подсетей достаточно управлять всего одним устройством - маршрутизатором.

Недостатком такой сети является ее ограниченная масштабируемость. Кроме того, для поддержания в маршрутизируемой магистрали достаточно большой скорости передачи данных необходим очень производительный маршрутизатор. Эта архитектура не предусматривает никакой иерархической структуры магистрали, поскольку сервер напрямую подключается к ней через 10 Мбит/с Ethernet. Такое подключение может создавать заторы, например когда большое число пользователей хочет получить доступ к совместно используемой базе данных.

FDDI-магистраль (рис. 10.3) - это единый канал, связывающий FDDI-серверы с рабочими группами Ethernet через один или несколько маршрутизаторов среднего класса. Такая сеть может объединять компьютеры, расположенные в отдельном здании или небольшом университетском городке.

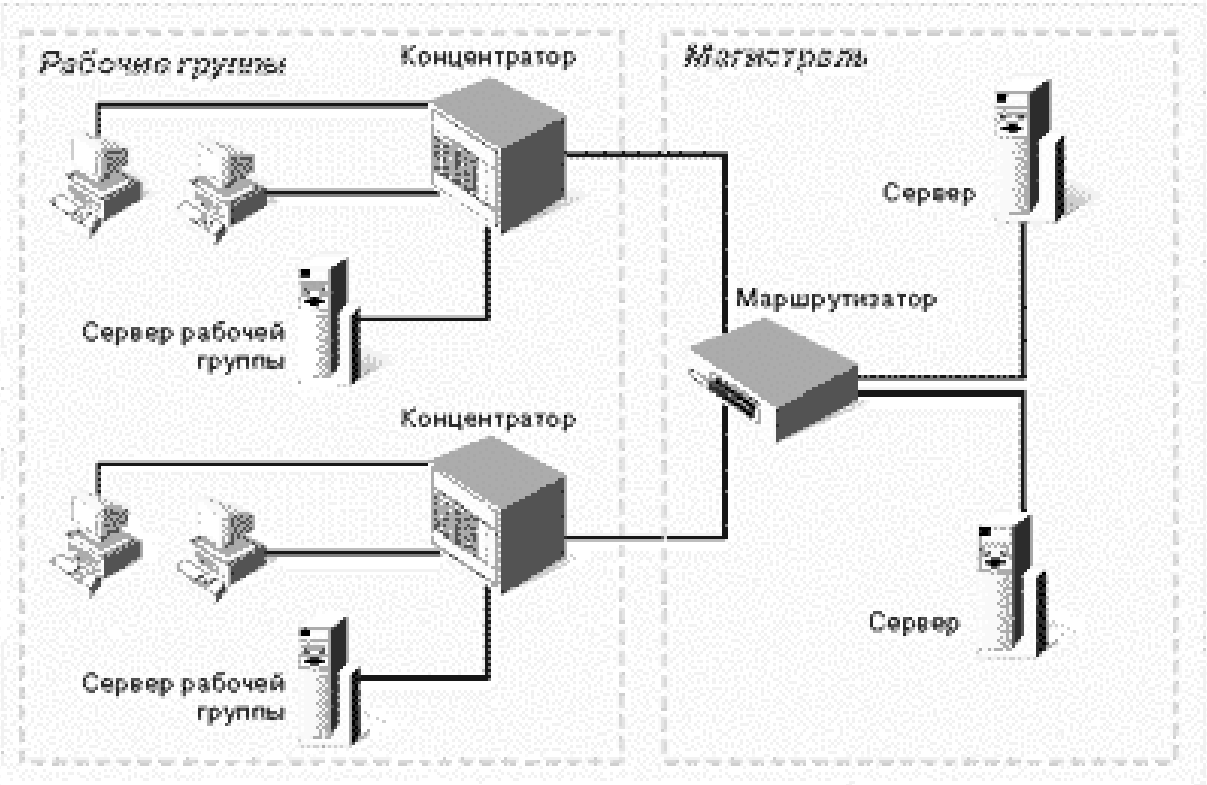



Рис. 10.2. Маршрутизируемая магистраль

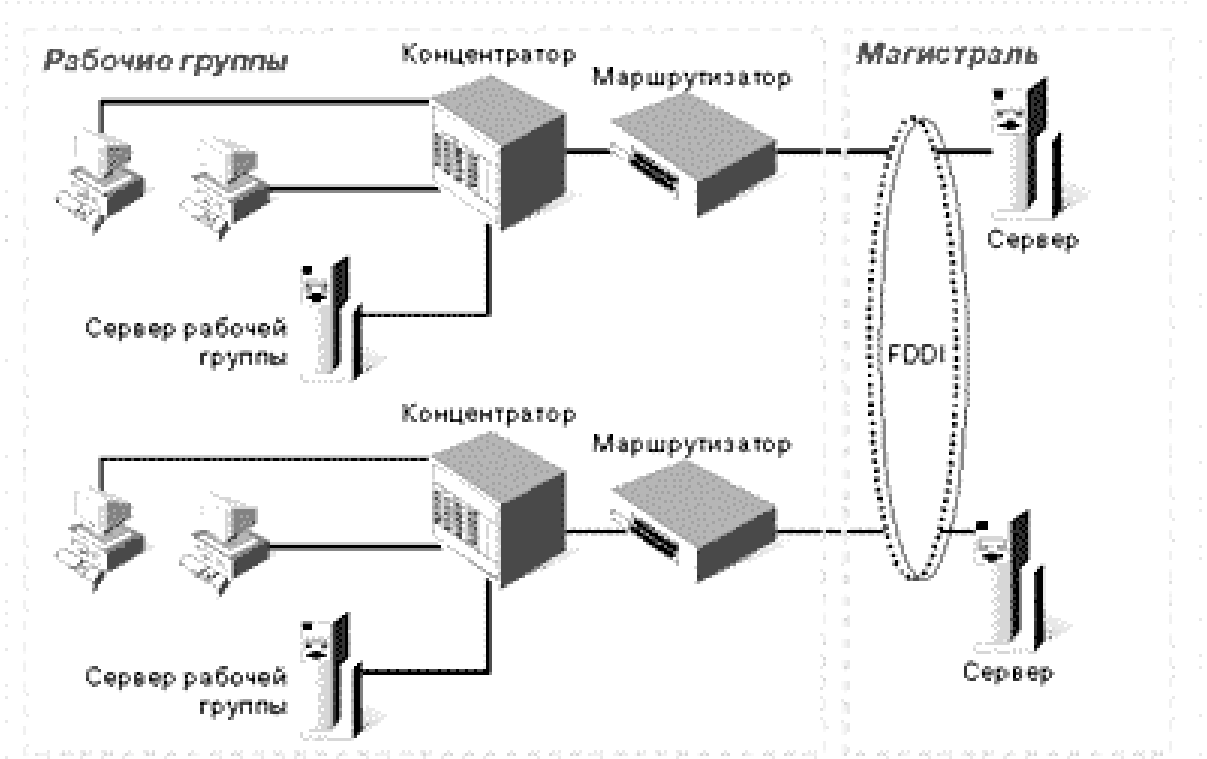



Рис. 10.3. FDDI-магистраль

Простота управления протоколами и возможность установки защитного экрана на границе между рабочей группой и магистралью - основные достоинства такой архитектуры. Высокоскоростная магистраль обрабаты-

вает общий поток информации и высокоскоростные операции сервер-сервер. Хорошая масштабируемость обеспечивается тем, что к FDDI-магистрали можно подключить много рабочих групп и маршрутизаторов, прежде чем будут полностью исчерпаны ресурсы этой архитектуры.

Однако изменение конфигурации сети приводит к появлению большого количества портов на маршрутизаторах, каждый со своим адресом подсети. Управление всеми устройствами и адресами - тяжелая работа, с которой может справиться только опытный администратор. Ретрансляция между маршрутизатором Ethernet и FDDI-сетью может снизить производительность программного обеспечения. Проблема усугубляется, если FDDI-магистраль необходимо сегментировать для передачи больших объемов информации.

Сеть с коммутацией кадров 10/100 (рис. 10.4) строится на основе коммутаторов, каждый из которых имеет двенадцать 10 Мбит/с интерфейсов с концентраторами рабочих групп (или рабочими станциями) и два 100 Мбит/с интерфейса для связи с серверами. Такая архитектура может использоваться для обеспечения высокой производительности сети в рабочих группах или для создания магистрали.

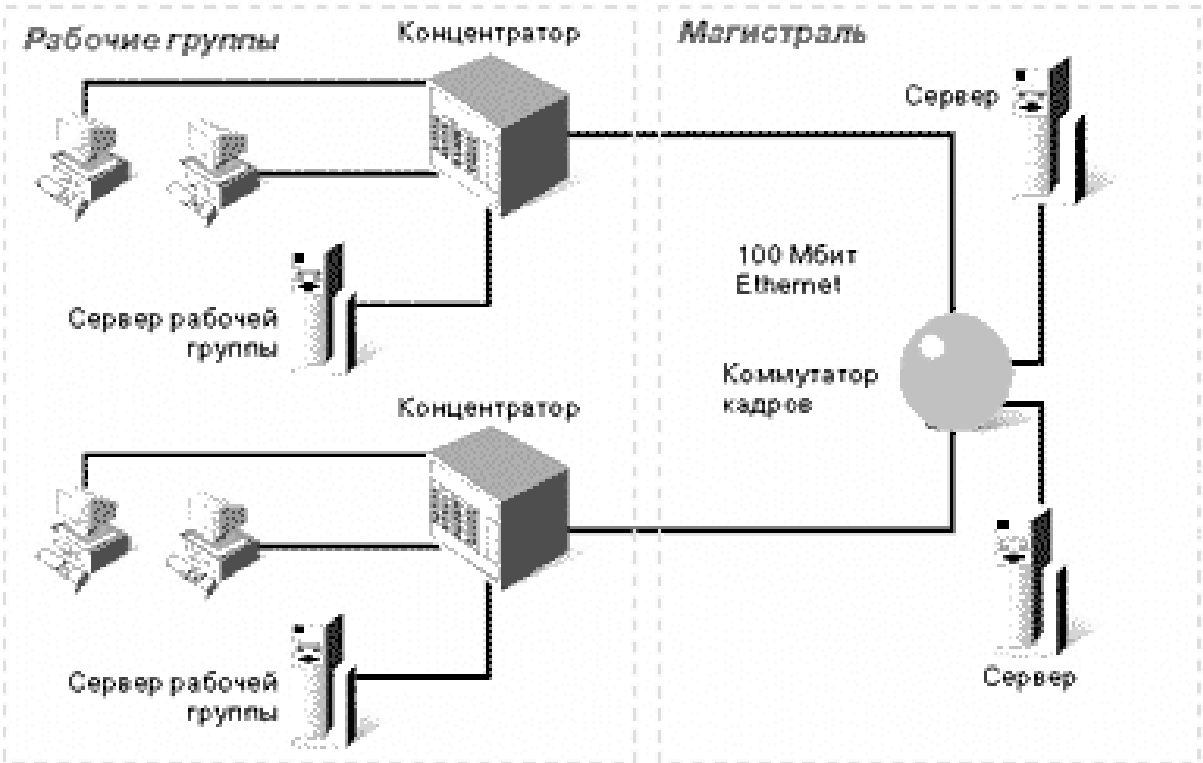



Рис. 10.4. Сеть с коммутацией кадров 10/100

Эта архитектура очень проста, что облегчает управление сетью. При этом "чистые" Ethernet-сети обычно работают по принципу plug-and-play. "Virtual LAN" позволяет создавать логические рабочие группы и устанавливать защитный экран. Высокая производительность сети обеспечивает

хорошее время реакции клиент-серверного программного обеспечения при передаче информации между серверами и централизованными ресурсами.

К сожалению, продукты для такой архитектуры, поддерживающие сети Token Ring, появились только к концу 1995 г., поэтому их "развитие" несколько запоздало. Кроме того, способы объединения пользователей и устройств в логические группы с помощью коммутаторов не стандартизированы, и реализация этой возможности у различных производителей может различаться. Поэтому при создании сети очень важно правильно выбрать производителя продуктов для коммутации кадров.

АТМ-коммутатор связывает АТМ-серверы, адаптеры Adj Path и 150 Мбит/с АТМ-каналы с коммутатором ячеек магистрали (рис. 10.5). Адаптеры Adj Path обеспечивают 10 Мбит/с Ethernet-связи серверов с рабочими группами или отдельными компьютерами. Такая архитектура может использоваться для обеспечения высокой производительности в рабочих группах или создания магистралей в одном или нескольких зданиях.

Созданная в соответствии с такой архитектурой высокоскоростная магистраль позволяет обрабатывать большое количество информации и эффективно осуществлять операции сервер-сервер. Отличная масштабируемость этой архитектуры позволяет создавать смешанную систему коммутаторов кадров или ячеек. Скорость отдельного интерфейса можно увеличить с помощью Fast Ethernet или более скоростных АТМ-связей. "Virtual LAN" позволяет создавать рабочие группы управления, а серверы могут быть централизованы, хотя логически будут оставаться близко к пользователям, что упрощает администрирование сети.

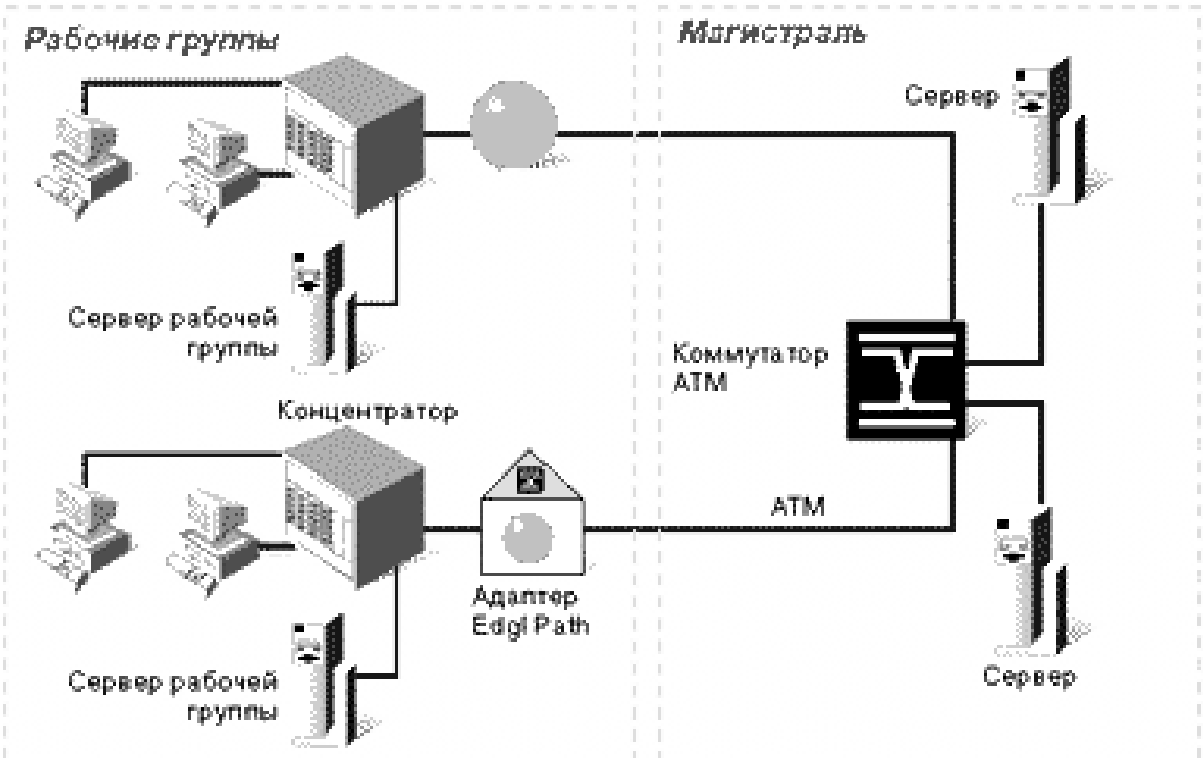



Рис. 10.5. АТМ и коммутацией кадров

АТМ - сравнительно молодая технология, поэтому стандарты для нее еще не до конца сформированы. Следовательно, АТМ-решения потребуют контактов с поставщиками оборудования.

### *10.3. Методика оценки сетевых архитектур*

Сравнение сетей, построенных на основе описанных выше сетевых архитектур, производилось по скорости выполнения трех различных операций:

- классического обмена информацией между клиентом и сервером;
- совместной обработки изображений;
- математического моделирования.

Результаты сравнения обобщаются на следующих рисунках, где показаны зависимости времени реакции сети от количества обслуживаемых пользователей при выполнении каждой из указанных операций и предложены различные архитектуры сетей для их поддержания. Для построения зависимостей использовались данные, полученные в результате моделирования сетевых операций с помощью процесса планирования Traffic Mappingo фирмы NCRI. Эти данные не универсальны и предназначены только для сравнения средней относительной производительности разных сетевых архитектур. Скорости передачи информации в реальных сетях могут отличаться от указанных. Это зависит от конкретной реализации продукта, дизайна и настройки программного обеспечения, а также способов их применения.

При моделировании сетевых операций были приняты следующие важные допущения и ограничения.

Реакция сети - это время, необходимое для выполнения исследуемой операции или группы операций. Время ответа сервера не учитывалось, исследовалась только производительность сети. Чтобы получить возможность полноценного сравнения архитектур, для всех рабочих групп использовались только адаптеры Ethernet. Предполагалось, что каждая операция выполняется в сети независимо от других. Например, в соответствии с результатами выполнения классической клиент-серверной операции в одном Ethernet-сегменте может работать более 40 пользователей, но это справедливо лишь в том случае, когда сеть выполняет только эту операцию. Использовался только протокол TCP/IP. Параметры производительности сетевых устройств, такие как задержки, время ожидания, общий диапазон и другие, соответствовали характеристикам реально существующих маршрутизаторов и коммутаторов.

Число пользователей, которые могут одновременно работать в сети с каждой конкретной архитектурой, определялось по следующей схеме.

1. Подсчитывалось количество пользователей в рабочей группе, которые могут одновременно запустить исследуемую операцию.

2. Определялся объем информации, который может сгенерировать одна рабочая группа.

3. Подсчитывалось количество рабочих групп, которые могут одновременно использовать ресурсы магистрали.

4. Число рабочих групп в магистрали умножалось на число пользователей в одной рабочей группе.

С помощью этой схемы можно достаточно точно оценить число пользователей, которых может обслужить каждая сетевая архитектура. Максимальное число пользователей означает, что производительность какой-либо части сети достигла своего предела. Следует отметить, что число пользователей указано для обычной, а не "расщепленной" сетевой архитектуры.

### *10.4. Построение сетей*

Исследование различных архитектур и возможностей их применения выявило ряд важных проблем. Дизайнеры сетей, которые используют только интуицию и прошлый опыт, могут оказаться в трудном положении. "Искусство" сетевой разработки должно преобразиться в науку. Если принимать решения без тщательного учета свойств программного обеспечения, которое будет использоваться в сети, то это может привести к плохому исполнению проекта.

Тот факт, что Ethernet с коммутацией кадров 10/100 поддерживает больше пользователей при моделировании окружающей среды, чем магистральная FDDI-сеть, может вызвать удивление у многих разработчиков сетей. При разработке сети для такого программного обеспечения "интуиция" наверняка привела бы к разработке FDDI-сети - то есть к более дорогостоящему решению. При этом сеть могла бы обслуживать меньшее число пользователей. При использовании программ групповой обработки изображений сети с коммутацией кадров или ATM могли бы сэкономить компании 300 тыс. долл. в год, обеспечивая лучшее время реакции и большую продуктивность работы команды инженеров.

В современных условиях для правильной разработки сети и ее обслуживания администраторы должны научиться решать следующие проблемы.

**Изменение организационной структуры.** При выполнении проекта не следует разделять разработчиков программного обеспечения и сетевой архитектуры. Многие организации, внедряющие информационные технологии, имеют различные группы для выполнения сетевых операций и разработки вычислительных систем. Обычно единственным человеком, входящим в обе группы, является директор по информационным системам. В результате такого разделения связь между этими группами осуществляется плохо, а в итоге принимаются неэффективные решения. При разработке

сетей и всей системы в целом нужно создавать единую команду из специалистов разного профиля.

**Оценка экономической выгоды.** В стоимость сети должны входить стоимости серверов, рабочих станций, конфигурирования сети, обучения обслуживающего персонала и пользователей. При переходе от мэйнфреймов к миникомпьютерам также нужно учитывать стоимость усиления сети, которая должна обеспечить увеличение потока информации и уменьшение времени реакции, необходимого для распределенных вычислений.

**Использование новых программ.** Необходимо знакомиться с новым программным обеспечением еще на ранней стадии разработки, чтобы можно было своевременно изменить сеть. В одной из компаний, входящих в группу Fortune 100, недавно было обнаружено, что менеджеры этой компании планируют использовать около 60 новых клиент-серверных программ за 18 месяцев, а сетевые администраторы знают только о 12 программах. Правильное планирование избавляет от неприятных сюрпризов.

**Исследование различных решений.** Необходимо оценивать различные архитектуры программ и их возможное влияние на сеть (а также время реакции), прежде чем начинать программирование. Надо оценивать топологии систем, а также проверять, как влияет на работу этих систем приближение серверов к большим скоплениям пользователей и выполнение фоновых модификаций на главной базе данных.

**Проверка сетей.** Важно использовать тесты на ранних стадиях разработки. Для этого можно создать прототип сети, который позволит оценить правильность принятых решений. С помощью такого прототипа можно предусмотреть возможные заторы и определить производительность разных архитектур. Пусть пользователи помогут проектировщикам оценить работу системы. Однако не стоит демонстрировать работу программы на линии Т-1, если она будет работать в коммутируемой 56 Кбит/с сети.

**Выбор протоколов.** Чтобы правильно выбрать конфигурацию сети, нужно оценить возможности различных наборов протоколов. Важно определить, как сетевые операции, оптимизирующие работу одной программы или пакета программ, могут повлиять на производительность других.

**Выбор физического расположения.** Выбирая место установки серверов, надо, прежде всего, определить местоположение пользователей. Возможно ли их перемещение? Будут ли их компьютеры подключены к одной подсети? Будут ли эти пользователи иметь доступ к глобальной сети?

**Вычисление критического времени.** Необходимо определить время использования каждой программы и периоды максимальной нагрузки. Важно понять, как чрезвычайная ситуация может повлияет на сеть, и определить, нужен ли резерв для непрерывной работы предприятия.

**Испытание сети.** Чтобы понять, какую нагрузку может выдержать сеть, надо ее смоделировать в уже работающей сети, проанализировать

причины возникновения замедлений и заторов и определить, как увеличение количества пользователей может повлиять на работу сети.

**Анализ вариантов.** Важно проанализировать различные варианты использования программного обеспечения в сети. Централизация данных часто означает дополнительную нагрузку в центре сети, а распределенные вычисления могут потребовать усиления ЛВС рабочих групп.

Прежде чем появились технологии коммутации кадров и ячеек, было отмечено несколько этапов увеличения сетевой производительности. Сегменты Ethernet и Token Ring подключались к маршрутизаторам. Сети Token Ring, для которых требовалась большая производительность, имели пропускную способность кольца до 16 Мбит/с. Затем предприятия развернули FDDI-магистрали для передачи информации между рабочими группами.

Сегодня в некоторых вычислительных системах коммутаторы Ethernet с 10 Мбит/с портами дополняют или заменяют маршрутизаторы, а коммутаторы кадров 10/100 конкурируют с FDDI. Как показывают приведенные в статье примеры, коммутация кадров в среднем обеспечивает гораздо лучшее время реакции и поддерживает большее количество пользователей по сравнению с маршрутизируемыми сетями и FDDI-магистралью. Коммутаторы можно установить в кластерной конфигурации, что обеспечивает высокоскоростное взаимодействие с серверами или магистралью на уровне предприятия. С помощью коммутации можно создавать более масштабируемые и управляемые сети.

Архитектура сетей сейчас изменяется, поэтому маршрутизаторы больше не стоят на пути между клиентом и сервером. Большинство из них не было предназначено для поддержки операций клиент-сервер с низким временем задержки и высокой производительностью. Теперь маршрутизаторы возвращаются к своей первоначальной роли - обеспечению связи между разнородными сетями (например, Ethernet и Token Ring) и межсетевой защиты.

Хотя FDDI по-прежнему остается главной составляющей в больших магистральных сетях, коммутация ячеек в ATM начала вытеснять FDDI как более эффективная магистральная технология. Возможно, к концу десятилетия технология ATM получит широкое распространение.

Наконец, технология коммутации кадров и ячеек позволяет изменить соотношение цены и производительности. Ее использование уменьшает расходы на эксплуатацию сети. Часто затраты на создание сетей оцениваются по стоимости на один порт. Раньше, когда обеспечение связи и взаимодействия было главной целью сети, такой способ оценки затрат был оправдан, но теперь он устарел. Сегодня основная задача разработки сетей заключается не в обеспечении связи, а в перемещении больших объемов данных, необходимом для распределенных вычислений. Поэтому новый принцип определения стоимости сети должен отражать ее способность пе-

ресылать данные. Стоимость порта не играет роли, так как не позволяет оценить производительность, которую обеспечивает ЛВС. При использовании более современного способа оценки учитываются затраты на переданный мегабит и скорость передачи данных по сети. В технологии коммутации каждый компьютер получает канал с известной скоростью передачи данных. Если оценивать коммутацию в соответствии с новыми принципами, то она является более экономичной, чем традиционные ЛВС совместного доступа. Коммутация обеспечивает высокую производительность, отличное время реакции и позволяет разработчикам сетей делать их более управляемыми - три качества, которые имеют принципиальное значение для современных и будущих сетей.

## Заключение

Настоящее учебное пособие охватывает лишь малую часть огромного мира сетей ЭВМ и телекоммуникаций. Эта отрасль науки, практики и технологий стремительно развивается и поэтому на момент выхода пособия в свет часть сведений могут оказаться устаревшими.

Вместе с тем базовые принципы и идеи, положенные в основу сетей и телекоммуникаций, вероятно, еще долго будут оставаться неизменными, что позволит освоившим материал данного пособия непрерывно совершенствоваться, понимая происходящие процессы и изучая новые методы и технологии.

Автор благодарит рецензентов – С.Л. Подвального, заведующего кафедрой автоматизированных и вычислительных систем Воронежского государственного технического университета, и кафедру «Прикладная математика» Липецкого государственного технического университета (зав. кафедрой А.К. Погодаев) за ценные замечания, сформулированные и учтенные при подготовке рукописи к изданию.

## Список литературы


1. Ассоциация пользователей электронной передачи информации: Информационный бюллетень. Вып. 1. – М.:Мортехинформреклама, 1991. – 48 с.
2. Блэк Ю. Сети ЭВМ: Протоколы, стандарты, интерфейсы. –М.: Мир, 1990. – 506 с.
3. Богуславский Л.Б., Дрожжинов В.И. Основы построения вычислительных сетей для автоматизированных систем. – М.: Энергоатомиздат, 1990. – 256 с.
4. Виноградов В.И. Информационно-вычислительные системы: Распределенные модульные системы автоматизации. – М.: Энергоатомиздат, 1986. – 336 с.
5. Вычислительные сети и сетевые протоколы/ Д.Дэвис, Д.Барбер, У.Прайс, С.Соломонидес. – М.:Мир, 1982. – 562 с.
6. Гладкий В.С., Гавлиевский С.Л. Численные методы анализа процессов маршрутизации на сетях ЭВМ// Программирование. 1986. – N3. – С. 78–87.
7. Кравец О.Я. Вычислительные сети: архитектура, оптимизация, управление. – Воронеж: ВГТУ, 1996. – 120 с.
8. Кристофидес Н. Теория графов. – М.: Мир, 1978. – 511 с.
9. Локальные вычислительные сети: Справочник. В 3-х кн. Кн. 1. Принципы построения, архитектура, коммуникационные средства/ С.В.Назаров, А.Г.Барсуков, В.П.Поляков, А.В.Луговец. – М.: Финансы и статистика, 1994. – 208 с.
10. Локальные вычислительные сети: Справочник. В 3-х кн. Кн. 2. Аппаратные и программные средства/ С.В.Назаров, В.П.Поляков, А.В.Луговец, В.С.Назаров. – М.: Финансы и статистика, 1994. – 264 с.
11. Максименков А.В., Селезнев М.Л. Основы проектирования информационно-вычислительных систем и сетей ЭВМ. – М.: Радио и связь, 1991. – 320 с.
12. Морозов В.К., Долганов А.В. Основы теории информационных сетей. – М.:Высшая школа, 1987. – 271 с.
13. Олифер В., Олифер Н. Сетевые операционные системы. WWW.Citforum.ru.
14. Прангишвили И.В. Микропроцессоры и локальные сети микроЭВМ в распределенных системах управления. – М.: Энергоатомиздат, 1985. – 272 с.
15. Протоколы информационно-вычислительных сетей: Справочник/С.А.Аничкин, С.А.Белов, А.В.Бернштейн и др.; Под ред. И.А.Мизина, А.П.Кулешова. – М.:Радио и связь, 1990. – 504 с.
16. Самойленко С.И. Сети ЭВМ. – М.: Наука, 1986. – 160 с.

17. Технология электронных коммуникаций. Т. 20. Безопасность связи в каналах телекомуникаций. – М.: Экотрендз, 1992. – 124 с.

18. Технология электронных коммуникаций. Т. 25. Сети NETWARE: телекомуникации и базы данных. – М.: Экотрендз, 1992. – 218 с.

19. Якубайтис Э.А. Локальные информационно-вычислительные сети. – Рига: Зинантне, 1985. – 284 с.

20. Bennet M., Heimes S. The Use of Hierarchical Networks of Computers in Totally Integrated FMS Systems// European Advanced Manufacturing Systems (Exhibitions and Conference). Italy, 1988. – P. 243-255.

21. Boyd J. Integrated Communications Network: Key to Future Advances in Information Industry// Communications News. 1982. March. – P. 50-51.

22. Dahod A. Local Network Standarts: no Utopia// Data Communications. 1983. March. – P. 173-180.

23. Functional Requirements for Lower-level Interfaces. Small Computer-to-peripheral Bus Interface. Data Transfer between Computer and Peripherals: Draft Proposal ISO/TC/97/SC 13. 1982. – N 290. – P. 108.

24. Harper C. Interface System Weds Instruments to Small Computer// Electronic Design. 1982. Vol.29. – N 26. – P. 78-87.

25. Muralidhar K.H., Sundareshan Malur K. Combined Routing and Flow Control in Computer Communication Networks: a Two-Level Adaptive Scheme// IEEE Trans. Autom. Contr. 1987. Vol. 32, – N1. – P. 15-25.

26. Prim R.C. Shortest Connection Networks and Some Generalizations// Bell Syst. Techn. J. 1957. Vol. 36. – P. 1389-1401.

27. Schematic Model. ISO/TS 97/SC 16. 1979. March. 14 p.

28. Д.Р.Левин, К.Бароди. Секреты Интернет. – Киев: Диалектика, ICE, 1996.

29. Ю.А.Семенов. Протоколы и ресурсы Интернет. – М.: Радио и Связь, 1996.

30. Д.С.Армстронг. Секреты Unix. – Киев: Диалектика, 1996.

31. K. Dowd. Getting Connected: the Internet at 56K and Up. - O'Reily and Associates, Inc.

32. C. Hunt. TCP/IP Network Administration. - O'Reily and Associates, Inc.

33. Albitz and Liu. DNS and BIND. - O'Reily and Associates, Inc.

34. Costales B., Alman E. Sendmail. - O'Reily and Associates, Inc.

35. Chapman and Zwicky. Building Internet Firewalls. - O'Reily and Associates, Inc.

36. Макстеник М. Сравнение сетевых архитектур // Сети, 1997. –№2.

37. http://www.sockaddr.com/.

38. http://www.winsock.com/.

39. http://www.sockets.com/.

40. http://www.sources.ru/protocols/bsp08/index.html.

Учебное издание

**Олег Яковлевич Кравец**

**Сети ЭВМ и телекоммуникации**

Издание публикуется в авторской редакции

Подписано в печать 09.08.2010. Формат 60х84 1/16.
Усл. печ. л. 14,0. Уч.-изд.л. 13,8. Заказ 217. Тираж 500.

ООО Издательство «Научная книга»
394077, г.Воронеж, ул. Маршала Жукова, 3-244
http://www.sbook.ru/

Отпечатано в ООО ИПЦ «Научная книга»
394026, г.Воронеж, ул. 303-й Стрелковой дивизии, 1[а]
тел. (4732)205715