Федеральное агентство по образованию РФ
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ГОУВПО «АмГУ»)

УТВЕРЖДАЮ
Зав. кафедрой ИУС
_____А.В.Бушманов
«_____»_____

УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС

по дисциплине «Сети ЭВМ и телекоммуникации»

для студентов специальности 230102 – Автоматизированные системы обработки информации и управления

Составитель доцент кафедры ИУС Галаган Т.А.

Факультет математики и информатики

Кафедра информационных и управляющих систем

2007

Печатается по решению
редакционно-издательского совета
факультета математики и информатики
Амурского государственного
университета

Т.А. Галаган

Учебно-методический комплекс по дисциплине «Сети ЭВМ и телекоммуникации». Для студентов специальности 230102 «Автоматизированные системы обработки информации и управления» очной формы обучения.- Благовещенск: Амурский гос. ун-т, 2007.

Пособие содержит рабочую программу, курс лекций, методические рекомендации по проведению и выполнению лабораторных работ, самостоятельной работы; составлено в соответствии с требованиями государственного образовательного стандарта.

I. ПРИМЕРНАЯ ПРОГРАММА УЧЕБНОЙ ДИСЦИПЛИНЫ, УТВЕРЖДЕННАЯ МИНИСТЕРСТВОМ ОБРАЗОВАНИЯ РФ

Государственный образовательный стандарт высшего профессионального образования

Направление подготовки дипломированного специалиста 654600 - Информатика и вычислительная техника

Образовательная программа – 230201 Автоматизированные системы обработки информации и управления

Наименование дисциплины – Сети ЭВМ и телекоммуникации

Блок общеобразовательных дисциплин, индекс ОПД.Ф.10.

Всего часов - 170

Содержание разделов:

Классификация информационно-вычислительных сетей. Способы коммутации. Одноранговые сети и сети «клиент-сервер». Уровни и протоколы. Эталонная модель взаимосвязи открытых систем. Аналоговые каналы связи. Способы модуляции. Модемы. Цифровые каналы передачи данных. Разделение каналов по времени и частоте. Характеристики проводных линий связи. Спутниковые каналы. Сотовые системы связи.

Кодирование информации. Локальные вычислительные сети. Методы доступа. Множественный метод с контролем несущей и обнаружением конфликтов. Разновидности сетей Ethernet. Маркерные методы доступа. Сети Token Ring и FDDI. Высокоскоростные локальные сети. Организация корпоративных сетей. Функции сетевого и транспортного уровней. Алгоритмы маршрутизации. Протоколы TCP/IP. Протоколы управления. Адресация в Internet. Особенности технологий Frame Relay, ATM, SDH. Сетевые операционные системы. Технологии распределенных вычислений. Структура и информационные услуги территориальных сетей. Протоколы файлового обмена, электронной почты, дистанционного управления. Виды конференц-связи. Web-технологии. Языки и средства создания Web-приложений.

II. РАБОЧАЯ ПРОГРАММА

Курс 3 семестр 6

Лекции 36 (час.) Экзамен 6 семестр

Практические (семинарские) занятия - (час.) Зачет -

Лабораторные занятия 36 (час.)

Самостоятельная работа 98 (час.)

Всего часов 170 час.

1.ЦЕЛИ И ЗАДАЧИ ДИСЦИПЛИНЫ

1.1. Цель курса - обучение студентов базовым знаниям о принципах построения компьютерных сетей, особенностей традиционных и перспективных технологий локальных и глобальных сетей, основных технологий локальных сетей и их оборудованию. Особое место уделяется технологии семейств Ethernet.

1.2. По завершению обучения дисциплине студент должен знать:

- базовые принципы построения и способы проектирования компьютерных сетей и телекоммуникаций;
- основные аспекты архитектуры локальных вычислительных сетей;
- современные технологии, протоколы и оборудование;

владеть

- навыками работы с сетевыми операционными системами Windows NT/XP;
- методами создания сетевых приложений с использованием протоколов TCP/IP, IPX/SPX
- методами программирования на языке JavaScript.

1.3. Преподавание курса "Сети ЭВМ и телекоммуникации" связано с изучением дисциплин государственного образовательного стандарта "Операционные системы", "Информационные технологии", "Алгоритмические языки и программирование" и является основой для изучения дальнейших дисциплин, использующих основы построения ЛВС: "Сетевые технологии", "Проектирование АСОИУ".

2. СОДЕРЖАНИЕ ДИСЦИПЛИНЫ

2.1. ФЕДЕРАЛЬНЫЙ КОМПОНЕНТ

Программа курса " Сети ЭВМ и телекоммуникации " составлена в соответствие с требованиями с требованиями государственного образовательного

стандарта специализации – Интегрированные системы автоматизированного управления, специализации 230201, блок общеобразовательных дисциплин ОПД.Ф.11.

2.2. ЛЕКЦИИ (36 часов)

2.2.1. Классификация компьютерных сетей. Требования, предъявляемые к современным сетям. Одноранговые сети и сети «клиент-сервер» (2 часа)
2.2.2. Принципы взаимодействия компьютеров в сети. Функциональные группы устройств в сети. Топологии подключения. Логическая и физическая структуризация сети (2 часа)
2.2.3. Аналоговые и цифровые каналы связи Способы модуляции. Модемы. (2 часа)
2.2.4. Понятие «открытая система». Стеки коммуникационных протоколов. Уровни модели OSI (4 часа)
2.2.5. Типы и характеристики линий связи. Стандарты кабелей. Методы передачи дискретных данных (4 часа)
2.2.6. Базовые технологии локальных сетей: Ethernet, Token Ring, FDDI. Высокоскоростные локальные сети. (8 часов)
2.2.7. Функции канального и сетевого уровней. Построение ЛВС с помощью мостов и коммутаторов. Алгоритмы маршрутизации (6 часов)
2.2.8. Основные направления развития современных сетевых операционных систем (2 часа)
2.2.9. Глобальные сети. Протоколы управления Адресация в Интернет. Протоколы файлового обмена, электронной почты. Web-технологии. (6 часов)

2.3. ПРАКТИЧЕСКИЕ ЗАНЯТИЯ -

2.4. ЛАБОРАТОРНЫЕ РАБОТЫ (36 часов)

2.4.1. Основы программирования на JavaScript: Структура программы, разветвляющиеся программы; операторы цикла; встроенные классы JavaScript, работа с окнами и фреймами, динамическое изменение объектов. (16 часов)
2.4.2. Основы сетевого программирования на основе протоколов IPX/SPX, TCP/IP (16 часов)
2.4.3. Методика расчета конфигурации сети Ethernet. (4 часа)

2.5. КУРСОВАЯ РАБОТА –

2.6. САМОСТОЯТЕЛЬНАЯ РАБОТА СТУДЕНТОВ (98 часов)
2.6.1. Вопросы для самостоятельного изучения:
1. Методы кодирования

2. Алгоритмы маршрутизации
3. Особенности технологий Frame Relay, ATM, SDH
4. Беспроводные сети
5. Сети с коммутацией пакетов
6. Организация корпоративных сетей
7. Спутниковые каналы
8. Сотовые системы связи
9. Виды конференц-связи
10. Протоколы файлового обмена, электронной почты

2.6.2. Чтение периодических изданий «Компьютерра», «Компьютер-пресс» и др.

2.6.3. Посещение специализированных сайтов htpp://citforum.amursu.ru

Контроль самостоятельной работы проводится путем тестового опроса по перечисленным темам. Вопросы самостоятельного изучения включены в экзаменационные вопросы.

2.7. ПЕРЕЧЕНЬ И ТЕМЫ ПРОМЕЖУТОЧНЫХ ФОРМ КОНТРОЛЯ ЗНАНИЙ

2.7.1. Вариант тестовой проверки знаний студентов

1. ***Выделите из ниже перечисленного функции, выполняемые на транспортном уровне эталонной модели взаимодействия открытых систем (OSI):***

а) определение начала и окончания сеанса связи;
б) контроль последовательности передачи данных;
в) определение маршрутизации в сети и связь между сетями;
г) установка соответствия между транспортными (логическими) и сетевыми адресами абонентов;
д) определение метода доступа к среде передачи данных;
е) определение времени, длительности и режима сеанса связи;
ж) определение логической топологии сети передачи данных;
з) обнаружение и обработка ошибок передачи данных;
и) обеспечение независимости высших уровней от используемой для передачи информации физической среды;
к) определение точек синхронизации для промежуточного контроля и восстановления при передаче данных;
л) определение физической адресации.

2. ***Повторитель – это устройство, позволяющее***

а) организовать обмен данными между сетевыми объектами, использующими различные протоколы обмена данными;
б) расширить сеть подключением дополнительных сегментов кабеля;
в) объединить несколько сегментов, так что передача данных между станциями внутри одного сегмента не будет влиять на передачу данных в других сегментах;

г) соединять сети разного типа, использующие одну сетевую операционную систему или протокол обмена данными.

3. Коллизия - это

а) ситуация, когда станция, желающая передать пакет, обнаруживает, что в данный момент другая станция уже заняла передающую среду;
б) ситуация, когда две рабочие станции одновременно передают данные в разделяемую передающую среду.

4. В сети Ethernet метод доступа станций к сети носит название

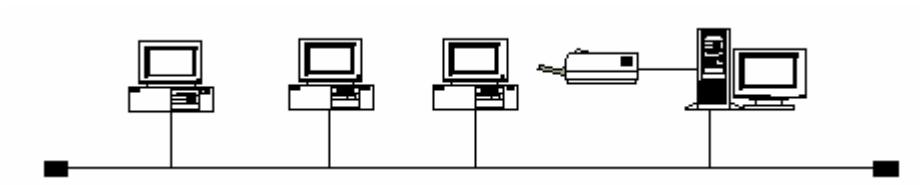
а) маркерное кольцо
б) метод множественного доступа с контролем несущей и обнаружением коллизий
в) маркерная шина

5. При обмене информацией между узлами сети данные передаются в одном направлении при использовании

а) симплексной передачи;
б) дуплексной передачи;
в) полудуплексной передачи.

6. Характеристика сети, предполагающая скрытие (невидимость) особенности сети от конечного пользователя, называется

а) интегрируемость;
б) прозрачность;
в) надежность;
с) масштабируемость.



7. ***На рисунке изображено подключение устройств по топологии:***

а) звезда; б) шина; в) кольцо.

8. Какие из следующих утверждений верны

а) пропускную способность можно измерять между любыми двумя узлами или точками сети;
б) пропускная способность измеряется в битах в секунду;
в) разделение линий связи приводит к повышению пропускной способности канала.

9.Какой из перечисленных вариантов характеристик соответствует технологии

Fiber Optic Ethernet (10BASE-F)

	а	б	в	г	д
Метод передачи сигналов	Одно-Полосной	Одно-полосной	Одно-полосной	Одно-полосной	Широко-полосной
Длина сегмента кабеля, м	500	185	100	2000	1800
Тип кабеля	50 Ом-ный коакси-альный, «тол-стый»	50 Ом-ный коакси-альный, «тонкий»	Витая пара (UTP)	Оптоволокно (FOC)	75 Омный коаксиаль-ный, «толстый»
Топология Подключе-ния устройства	Шина	Шина	Звезда	Звезда, двухточечное соединение	Шина

2.8. ЭКЗАМЕНАЦИОННЫЕ ВОПРОСЫ

1. Структура взаимодействия устройств в сети
2. Модель взаимосвязи открытых систем
3. Иерархическая сеть
4. Сети клиент-сервер
5. Методы передачи информации
6. Структура пакета в ЛВС
7. Функциональные группы устройств сети (узел, рабочая станция, сервер (группы серверов), повторитель, мост, мультиплексор, маршрутизатор, шлюз)
8. Топологии подключения устройств в сети
9. Среда передачи данных
10. Стандарты кабельной системы
11. Локальная сеть Ethernet
12. Высокоскоростные варианты Ethernet
13. Сети Token Ring
14. Распределенный волоконно-оптический интерфейс передачи данных (FDDI)
15. Устройства передачи данных
16. Основные требования, предъявляемые к вычислительным сетям на современном предприятии
17. Классификация по масштабу сети
18. Компоненты информационных систем (прикладная программа, сетевое программное обеспечение, драйвер устройства передачи данных, последовательные порты, платы расширенных последовательных портов, модем, сете-

вые адаптеры)
19. Основные направления развития современных сетевых операционных систем
20. Особенности технологии Frame Relay.
21. Особенности ATM-технологии
22. Особенности технологии SDH
23. Стек коммуникационных протоколов IPX/SPX
24. Протокол TCP/IP
25. Методы обнаружения ошибок
26. Иерархия кабельной системы
27. Физическая структуризация сети
28. Логическая структуризация сети
29. Алгоритмы работы мостов
30. Маршрутизаторы в ЛВС
31. Сетевые адаптеры
32. Коммутаторы
33. Модемы
34. Беспроводные сети
35. Глобальные сети
36. Сети с коммутацией пакетов
37. Цифровая сеть с интеграцией обслуживания
38. Сетевые операционные системы
39. Типы характеристик линий связи и способы их определения
40. Амплитудно-частотная характеристика
41. Полоса пропускания
42. Затухание
43. Пропускная способность линии
44. Формула Клода Шеннона
45. Формула Найквиста
46. Помехоустойчивость и достоверность
47. Адресация в Интернет
48. Протоколы файлового обмена, электронной почты
49. Виды конференц-связи
50. Сотовые системы связи
51. Спутниковые каналы

КРИТЕРИИ ОЦЕНОК ЗНАНИЙ СТУДЕНТОВ

Отлично Студент дает полные ответы на теоретические вопросы билета, показывая глубокое знание учебного материала, свободное владение основными понятиями и терминологией; ответ на дополнительный вопрос; решение задачи.

Хорошо Студент дает ответы на теоретические вопросы билета, показывая прочное знание учебного материала, владение основными понятиями и терминологией; ответ на дополнительный вопрос; решение задачи.

Удовлетворительно Студент дает неполные ответы на теоретические вопросы билета, показывая поверхностное знание учебного материала, владение основными понятиями и терминологией; при неверном ответе на билет ответы на наводящие вопрос; частичное решение задачи.

Неудовлетворительно Студент не дает полные ответы на теоретические вопросы билета, показывая лишь фрагментарное знание учебного материала, незнание основных понятий и терминологии; наводящие вопросы остаются без ответа; решение задачи не найдено.

3. УЧЕБНО-МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ:

3.1. ЛИТЕРАТУРА

ОСНОВНАЯ

1. Олифер В.Г., Олифер Н.А. Компьютерные сети. Принципы, технологии, протоколы. (рекомендовано Мин. образования РФ). СПб: Питер, 2001.
2. В.Г Олифер, Н.А. Олифер Сетевые операционные системы. СПб.: Питер, 2001.
3. Андерсон К., Минаси. М. Локальные сети. Полное руководство. К.: ВЕК+, М.:ЭНТРОП, СПб.:КОРОНА, 1999.
4. Гордеев А. В. Операционные системы. Учебник для вузов (допущено Мин. образования РФ). 2-е изд. СПб.: Питер, 2004.
5. Дунаев В. Самоучитель JavaScript. 2-е изд. СПб.: Питер, 2005.
6. Таненбаум. Э. Архитектура компьютера. – Киев: DHV, 2001.
7. Барановская Т.П., Лойко В.И., Семенов М.И., Трубин А.И. Управляющие вычислительные комплексы. Уч. пособие рек. Мин. Обр. РФ. М.: Финансы и статистика, 2003.
8. Иванов В. Компьютерные коммуникации. Учебный курс. – СПб: Питер. 2001
9. Компьютерные системы и сети. Уч. пособие рек. Мин. Обр. РФ. / Под ред. Косаре В.П., Еремина Л.В. - М. : Финансы и статистика, 2000.

ДОПОЛНИТЕЛЬНАЯ

1. Стэн Шат Мир компьютерных сетей. К.:BHV, 1996, 288с
2. Э.А. Якубайтис. Информационные сети и системы. Справочная книга. М: "Финансы и статистика" 1996, 368 с
3. Д. Веттинг. Novell NetWare для пользователя. К.: BHV. 1994, 480с
4. Э.Титтел, Д.Коннор, Э.Фоллис Компьютерные сети на основе NETWARE для "чайников" К.: "Диалектика", 1997

3.2. МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ

1. Галаган Т.А. Сети ЭВМ и телекоммуникации. Лабораторный практикум. Благовещенск. 2005 (электр. вариант)

2. Галаган Т.А., Михайлов М.В. Работа в сети Novell NetWare. Благове-

щенск: Изд-во АмГУ. 1999. 40 с.

ТЕХНИЧЕСКИЕ СРЕДСТВА ОБЕСПЕЧЕНИЯ ДИСЦИПЛИНЫ

1. Кафедра обладает необходимым количеством современных ЭВМ для проведения лабораторных работ
2. Программное обеспечение: операционные системы Novell NetWare 4.12, Windows 98, Windows XP, программные среды Borland C++, Delphi.

4.УЧЕБНО-МЕТОДИЧЕСКАЯ (ТЕХНОЛОГИЧЕСКАЯ) КАРТА ДИСЦИПЛИНЫ

Номер недели	Вопросы, изучаемые на лекции	Занятия (номера)		Используемые нагляд. и метод. пособия	Самостоятельная работа студентов		Форма контроля
		Практич (сем.)	Лабо рат.		Содержание	часы	
1	2.2.1		2.4.1		2.6.1 - 2.6.2	4	отчет
2	2.2.2		2.4.1		2.6.1 - 2.6.2	4	отчет
3	2.2.3		2.4.1		2.6.1 - 2.6.2	4	отчет
4	2.2.3		2.4.1		2.6.1 - 2.6.2	4	отчет
5	2.2.4		2.4.1		2.6.1 - 2.6.2	4	отчет
6	2.2.4		2.4.1		2.6.1 - 2.6.2, 2.6.4	6	отчет
7	2.2.5		2.4.1		2.6.1 - 2.6.2, 2.6.4	6	отчет
8	2.2.5		2.4.2	3.2.2	2.6.1 - 2.6.2, 2.6.4	6	отчет,
9	2.2.6		2.4.2	3.2.2	2.6.1 - 2.6.2	6	отчет
10	2.2.6		2.4.3	3.2.1	2.6.1 - 2.6.3	6	отчет
11	2.2.6		2.4.3	3.2.1	2.6.1 - 2.6.3	6	отчет,
12	2.2.6		2.4.3	3.2.1	2.6.1 - 2.6.3	6	отчет
13	2.2.7		2.4.3	3.2.1	2.6.1 - 2.6.3	6	отчет
14	2.2.7		2.4.3	3.2.1	2.6.1, 2.6.5	6	отчет,
15	2.2.7		2.4.4	3.2.1	2.6.1, 2.6.5	6	отчет
16	2.2.8		2.4.4	3.2.1	2.6.1 - 2.6.2	6	отчет,
17	2.2.9		2.4.4	3.2.1	2.6.1 - 2.6.2	6	отчет 2.7.1
18	2.2.9		2.4.4	3.2.1	2.6.1 - 2.6.2	6	отчет,

III. МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ПРОВЕДЕНИЮ И ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ

Лабораторные работы проводятся по подгруппам в компьютерном классе. Каждый студент получает индивидуальное задание в соответствии с вариантом. Выполняя задание, студент пользуется материалом, изложенным в тексте лабораторной работы.

Перед созданием любой программы требуется точно продумать алгоритм. Записать его блок-схемой или словесно. Надо четко определить, что в нее требуется ввести и что получить в результате, в какой последовательности выполнять действия. В случае необходимости выделить циклические структуры и подпрограммы. В циклах четко определить параметры, задать их начальные значения, определить условия повторения и завершения цикла. В функциях определить количество передаваемых и возвращаемых значений.

При кодировании программы нужно определить тип используемых данных в зависимости от возможного диапазона принимаемых значений. При вводе величины не забывать осведомить об этом пользователя, а иногда сообщить и о типе, диапазоне или порядке ввода значений. Такое сообщение должно быть информативно и коротко. Вывод данных лучше сопровождать текстом и форматированием. Формат вывода можно уточнить при помощи модификаторов.

В именах переменных необходимо отражать их назначение, что повышает читаемость и понимание программы.

При записи сложных выражений нужно обращать внимание на приоритет операций. Текст программы лучше сопровождать краткими и информативными комментариями, что облегчает как понимание программы, так и ее отладку.

Объявление локальных переменных предпочтительнее по сравнению с использованием глобальных.

Для отладки программы нужно запустить ее на выполнение несколько раз, задавая различные значения вводимых величин. Перед запуском необходимо иметь заранее подготовленные тестовые примеры, содержащие исходные данные и ожидаемые результаты. Их количество зависит от алгоритма. проверьте реакцию программы на заведомо неверные исходные данные.

Для быстрого поиска ошибки в алгоритме рекомендуется выводить промежуточные данные.

При сдаче лабораторной работы студент должен продемонстрировать преподавателю созданную программу, правильно работающую, отлаженную.

Преподаватель, принимая лабораторную работу, тестирует программу студента и задает ему вопросы по конструкциям, используемым в программе и теоретическим основам программирования.

IV. ТЕХНОЛОГИЯ ВЫПОЛНЕНИЯ И ЗАДАНИЯ К ЛАБОРАТОРНЫМ РАБОТАМ

ЧАСТЬ 1. СРЕДСТВА СОЗДАНИЯ WEB-ПРИЛОЖЕНИЙ

Лабораторная работа 1 (2 часа)
Основы JavaScript

Для выполнения программ JavaScript в качестве интерпретатора (исполнительной системы) можно использовать веб-браузер Internet Explorer 6.0 (5.0). В качестве редактора программ текстовый редактор, например Блокнот. Создавать программы на JavaScript можно и с помощью специальных программ, предназначенных для разработки веб-сайтов, например Microsoft FrontPage.

Можно легко создать и собственный редактор программ, например кодом:

```
<HTML>
<H3>Редактор кодов</H3>
код:<br>
<TEXTAREA id="mycode" ROWS=10 COLS=60></TEXTAREA>
<p>Результат:<br>
<TEXTAREA id="myrezult" ROWS=3 COLS=60></TEXTAREA>
<P>
<BUTTON onclick="document.all.myrezult.value=eval(mycode.value)">
Выполнить</BUTTON>
<BUTTON onclick="document.all.mycode.value=' ';
document.all.myrezalt.value=' ' ">
Очистить </BUTTON>
<P>
<!Комментарий>
Введите выражения в верхнее поле.
Выражения разделяются точкой с запятой.
Можно также каждое выражение писать в отдельной строке
</HTML>
```

Ввод-вывод данных

Для обеспечения ввода-вывода JavaScript предоставляет несколько методов:
alert( ), confirm ( ), prompt( ).

Первый из перечисленных выводит на экран диалоговое окно с заданным сообщением и кнопкой ОК.

Синтаксис данного метода:

alert (сообщение)

Сообщение может представлять собой данные любого типа: последова-

тельность символов, заключенную в кавычки, число, переменную или выражение.

Текст необходимо заключить в кавычки. Например,

alert (“Всем привет! ”)

Для формирования строк используют служебные символы:

\n – новая строка,

\t – табуляция,

\f – новая страница,

\b – забой,

\r – возврат каретки.

Окно, создаваемое alert( ) является модальным (останавливающим все последующие действия программы и пользователя). Его можно убрать, щелкнув по кнопке ОК.

Метод confirm выводит на экран диалоговое окно с сообщением и двумя кнопками – ОК и Отмена. Этот метод возвращает логическую величину, значение которой зависит от того, по какой из кнопок щелкнет пользователь. Возвращаемое значение можно обработать в программе, создавая тем самым интерактивный эффект. Синтаксис применения данного метода аналогичен синтаксису метода alert.

Окно, создаваемое confirm также является модальным.

Метод prompt осуществляет вывод диалогового окна с сообщением и кнопками ОК и Отмена, а также с текстовым полем, в которое пользователь может ввести данные. В отличие от alert( ) и confirm( ) данный метод принимает два параметра: сообщение и значение, которое должно появиться в текстовом поле ввода данных по умолчанию. Если пользователь щелкнет по кнопке ОК, метод вернет содержимое поле ввода данных, если – по кнопке Отмена, то возвращается значение ложь. Возвращаемое значение можно также обработать в программе. Синтаксис метода:

prompt (сообщение, значение_поля_ввода данных)

Оба параметра не являются обязательными. Если они не указаны, на экране появится окно без сообщения, а в поле ввода данных подставлено значение по умолчанию – undefined (не определено). Чтобы значение по умолчанию не появилось, в качестве второго параметра указывается пустая строка (“ ”).

Типы данных

Типы данных языка JavaScript приведены таблице 1.

Тип данных	Описание значений
Строковый тип (string)	Последовательность символов, заключенная в кавычки, двойные или одинарные
Числовой (number)	Положительное или отрицательное число; целая и дробная части разделяются точкой
Логический	Два значения: true или false
Null	Отсутствие какого-либо значения

Табл.1. Типы данных JavaScript

При создании программ за типом данных следит сам программист. Интерпретатор не выдаст ошибки при неверном их использовании. Он просто попытается привести данные к типу, требуемому в данной операции.

Например, при написании выражения 7+ “нет” результатом будет строка символов “7нет”. Интерпретатор сначала переводит число в строку, а затем выполняет сложение двух строк, результатом которого в JavaScript является слияние двух строк. Результатом вычисления выражения 5+6 будет 11, а выражения 5+“6” – “56”.

Для преобразования строк в числа предусмотрены встроенные функции parseInt( ) и parseFloat( ). Синтаксис:

parseInt(строка, основание)

parseFloat(строка, основание)

Если основание не указано, то предполагается 10 – десятеричная система счисления. В качестве основания можно также использовать 8, 10, 16.

При преобразовании строки в целое число округление не происходит – дробная часть просто отбрасывается.

Для определения того, является ли значение выражения числом, служит встроенная функция isNaN(значение). Функция возвращает логический тип.

Переменные

Имя переменой представляет собой конечную последовательность символов, содержащую буквы, цифры, символ подчеркивания. Имя переменной не должно начинаться с цифры или содержать пробелы. Для имен переменных нельзя использовать ключевые слова языка.

В отличие от многих других языков программирования переменной не нужно задавать тип при объявлении. Тип переменной определяется типом ее значения. Переменная может принимать значения разных типов и неоднократно его изменять.

Создавать переменную в программе можно несколькими способами. Можно ей просто присвоить значение с помощью оператора присваивания в формате: имя_переменной = значение

Например,

Month= “Январь”

Можно использовать ключевое слово var перед именем переменной. В этом случае переменная не будет иметь первоначальное значение, но в дальнейшем его можно передать с помощью оператора присваивания. Например,

var Month

Month = “Январь”

При использовании var допускается и инициализация переменной.

Можно сразу объявить несколько переменных, используя var и разделяя их запятой, при этом возможно инициализировать их все или некоторые:

var Month= “Январь”, day, pi=3.14, x

Комментарии

В JavaScript допустимы два вида операторов комментария:

- одна строка символов, расположенная справа от //
- произвольное количество строк, заключенных между /* и */.

Операции

В таблице 2 заданы основные операции, определенные в языке JavaScript.

Операция	Краткое описание
Унарные операции	
++	инкремент (увеличение на 1)
--	декремент (уменьшение на 1)
Бинарные операции	
*	умножение
/	деление
%	остаток от деления
+	сложение
-	вычитание
<	меньше
<=	меньше или равно
>=	больше или равно
= =	равно
!	отрицание (не)
&&	логическое И
||	логическое ИЛИ
=	присваивание
*=	умножение с присваиванием
/=	деление с присваиванием
%=	остаток от деления с присваиванием
+=	сложение с присваиванием
- =	вычитание с присваиванием

Табл.2 Операции JavaScript

Операции выполняются в соответствии с приоритетами. Приоритет операций аналогичен языку C++. Для изменения порядка выполнения операций используются круглые скобки.

Операторы ветвления

Операторы ветвления сохраняют преемственность языка C++.

Условный оператор if используется для разветвления процесса вычислений на два направления. Синтаксис оператора if:

if (условие) оператор1 else оператор 2

Пример

```
if ( fvalue>=0.0 )  fvalue = fvalue  else fvalue = -fvalue
// вычисляется модуль произвольного числа.
```

Ветвь с ключевым словом else может отсутствовать. Например,

```
if ( f != 0 )  l = 100 / f
```

Если в какой-либо ветви требуется выполнить несколько операторов, их необходимо заключить в блок (операторные скобки { }), иначе компилятор не сможет определить окончание ветвления.

```
if ( a ) { a++    v=60*a }  else v = a
if ( e = =1000 ) { e /=10  alert(e) }  else { e =10+y*y    y++ }
```

Условные операторы могут быть вложенными.

```
if ( a<b ) { if ( a<c ) m = a  else m = c }  else { if ( b<c ) m = b  else m = c }
```

Необходимо помнить, что в этом случае else относится к ближайшему if. Операторные скобки после первого if необязательны.

Если требуется проверить несколько условий, их объединяют знакам логических операций.

Оператор switch (переключатель) предназначен для разветвления процесса вычислений на несколько направлений. Синтаксис оператора:

```
switch ( выражение) {
case  константное выражение 1  : операторы1   break
case  константное выражение 2  : операторы2   break
…
case  константное выражение n  : операторыN   break
default : операторы }
```

Выполнение оператора начинается с вычисления выражения, а затем управление передается первому оператору из списка, помеченному константным выражением, значение которого совпало с вычисленным.

После этого, если выход из переключателя явно не указан (отсутствует break), последовательно выполняются все нижележащие ветви. Если совпадения не произошло, выполняются операторы, расположенные после ключевого слова default. Ветвь default может отсутствовать.

Параметр выражения оператора switch может принимать строковые, числовые и логические значения. В следующем примере переменная x содержит название языка, который выбрал пользователь. А выражение window.open ( ) открывает новое окно браузера и загружает в него указанный в скобках HTML-документ.

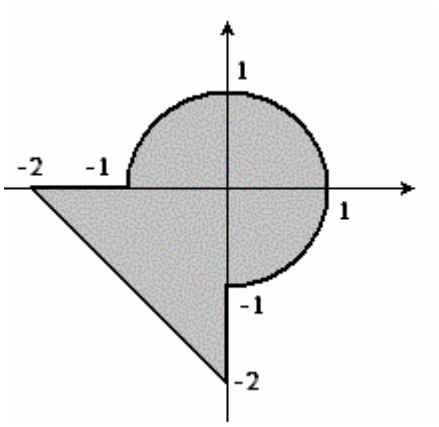
```
 switch (x ) {
case “ английский“ : window.open ( “engl.htm”) ; break
case  “ французский“  : window.open ( “french.htm”) ;   break
case  “ русский“ :  window.open ( “russ.htm”) ;              break
default  :  alert(  “Нет документа на таком языке” )
}
```

Задание

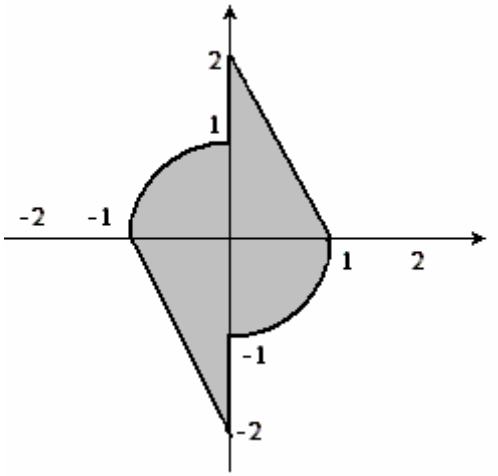
1. Создать собственный редактор программ, на основе HTML-кода, модифицировав код, приведенный выше. Файл открыть в браузере как веб-страницу.
2. Протестировать редактор, изучив в нем функции вывода и основные операции JavaScript.

3. Составить программу на основе разветвляющего алгоритма для задачи: Дана «мишень» в виде закрашенной области, изображенной на рисунке. Создать алгоритм для определения, попадает ли в нее точка с координатами x, y .

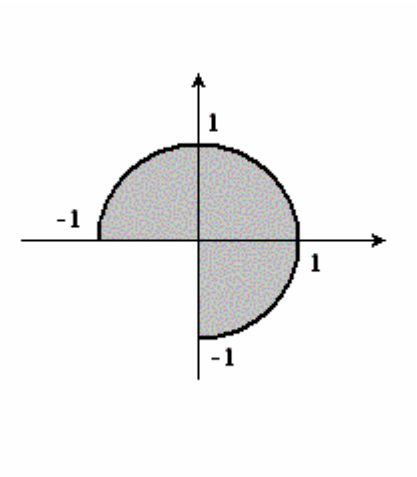
Вариант 1



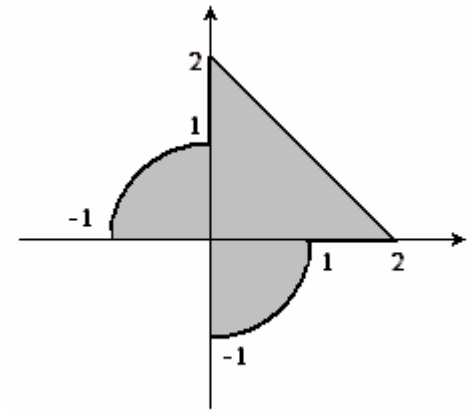
Вариант 2



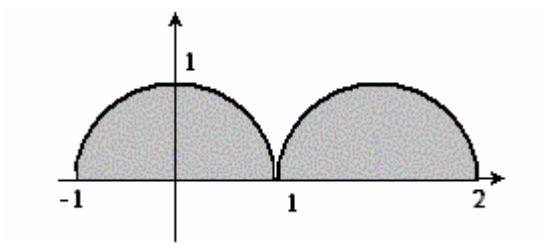
Вариант 3



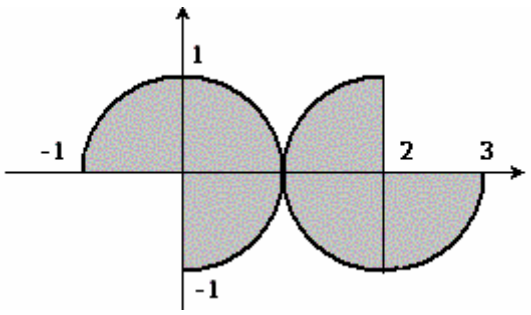
Вариант 4



Вариант 5



Вариант 6



Лабораторная работа №2 (4 часа)

Операторы цикла. Организация пользовательских функций

JavaScript содержит следующие встроенные функции (некоторые уже были рассмотрены):

parseInt(строка, основание) – преобразует указанную строку в целое число по указанному основанию(8, 10, 16); по умолчанию - десятеричная система.

parseFloat(строка, основание) – преобразует строку в число с плавающей точкой.

isNaN(строка, основание) – возвращает true, если указанное в параметре значение не является числом.

eval(строка) – вычисляет выражение в указанной строке, выражение не должно содержать тегов HTML.

Например,

```
var a = 10;                          // значение a равно 10
var b = “if (a<=25) { a *= 2 }”      // значение b равно строке символов
eval (b)                             // значение a равно 20
```

escape(строка) – возвращает строку в виде %XX, где XX – ASCII-код указанного символа. Такую строку называют escape-последовательностью.

unescape(строка) – обратное преобразование.

В программах на JavaScript пользователям также разрешено создавать собственные функции. Объявление функция состоит из заголовка и тела:

```
function имя_функции ( параметры )          //заголовок функции
{ тело функции }
```

Тело функции ограничивается фигурными скобками. Параметры функции, стоящие в круглых скобках перечисляются через запятую.

Возвращение значений из функции происходит с помощью оператора return, за которым помещается само возвращаемое значение.

Для вызова функции можно воспользоваться выражениями вида:

имя_функции (параметры) или

имя_переменной = имя_функции(параметры)

Параметры в вызове функции должны быть представлены значениями.

Например, если описание функции дано в виде:

```
function cube (x)
{ return x*x*x }
```

Ее вызов может быть записан в виде: y=cube(25)

В JavaScript функции могут вызываться как после их определения, так и до него. Можно не поддерживать соответствие количества параметров в определении функции и в ее вызове. Если в определение функции параметров больше, чем в вызове, то недостающим параметрам автоматически присваивается значение null. Лишние параметры в вызове функции игнорируются.

Внутри функции можно создавать локальные переменные с помощью оператора присваивания или с помощью ключевого слова var.

Если в теле функции переменная в составе оператора присваивания

встречается впервые в программе, или она была определена до этого – она действует как глобальная.

Если в теле функции используется переменная, объявленная только во внешней программе, она также является глобальной.

Если для определения переменной в теле функции используется ключевое слово var, она будет локальной вне зависимости от того определена она во внешней программе или нет.

Операторы цикла

Как и другие языки программирования JavaScript имеет три вида оператора цикла: цикл с предусловием (while), цикл с постусловием (do while), цикл с параметром (for).

Синтаксис цикла с предусловием:

```
while ( условие )
{
операторы
}
```

Выражение, стоящее в круглых скобках, определяет условие повторения тела цикла, представленного простым или составным оператором. Если оператор простой операторные скобки { } могут не ставиться.

Выполнение оператора цикла начинается с вычисления выражения. Если оно истинно, выполняется тело цикла. Если при первой проверке выражение ложно (false), цикл не выполнится ни разу.

Значение выражения вычисляется перед каждой итерацией цикла.

Цикл while обычно используется в тех случаях, когда число повторений заранее неизвестно. Его отличительной чертой является выполнение тела цикла хотя бы один раз. И только после первого его выполнения проверяется, надо ли его выполнять еще раз.

Цикл for называют также циклом с заданным числом повторений. Он имеет следующий формат:

```
for (инициализация; выражение (условие); модификации )
{
операторы
}
```

Инициализация используется для объявления и присвоения начальных значений величинам, используемым в цикле. Инициализация выполняется один раз перед выполнением тела цикла.

Выражение определяет условие выполнения цикла: если его результат равен истине, то цикл выполняется. Модификации выполняются после каждой итерации цикла и служат обычно для изменения параметров цикла.

Тело цикла представляет собой простой или составной оператор.

```
for (i = 1, s=1; i<11; i++)
{
s*=i;                    // вычисления факториала 10
}
```

Для принудительного выхода из тела любого цикла используются операторы break и continue. break позволяет переход в точку программы, находящуюся непосредственно за оператором, внутри которого находится, т.е. управление передается первой строке, следующей за телом цикла.

Инструкция continue заставляет программу пропустить все оставшиеся строки цикла, но сам цикл при этом не завершается. Для решения некоторых задач удобно комбинировать инструкции break и continue.

Задание

1. Составьте программу на основе циклического алгоритма для вычисления суммы ряда с заданной точностью ε. Определите и выведите на экран значение суммы и число элементов ряда, вошедших в сумму.

Номер варианта	***Задание***	***Точность***
1	$\frac{\pi}{3}+\sum_{n=1}^{\infty}(-1)^n\frac{(\pi/3)^{2n+1}}{(2n+1)!}$	$0.5*\ 10^{-4}$
2	$\sum_{n=1}^{\infty}(-1)^n\frac{(\pi/6)^{2n}}{(2n)!}$	$0.5*\ 10^{-4}$
3	$\sum_{n=1}^{\infty}\frac{(-1)^n(2n+1)}{(3n)!}$	$0.1*10^{-3}$
4	$\sum_{n=1}^{\infty}\frac{(-1)^{n+1}}{3n^2}$	$0.1*10^{-2}$
5	$\sum_{n=1}^{\infty}\frac{(-1)^n(2n+1)}{n^3(n+1)!}$	$0.1*10^{-2}$
6	$\sum_{n=1}^{\infty}\frac{(-1)^n n}{(2n-1)^2(2n+1)^2}$	$0.1*10^{-2}$

2. Создать пользовательскую функцию, продемонстрировать ее работу в программе.

Вариант 1. Составить функцию вычисления факториала.

Вариант 2. Составить функцию, для вычисления $(a-b)^3$.

Вариант 3. Составить функцию нахождения максимума из трех чисел.

Вариант 4. Составить функцию нахождения минимума из трех чисел.

Вариант 5. Составить функцию проверки на равенство трех чисел

Вариант 6. Составить функцию для вычисления многочлена третьей степени.

Лабораторная работа №3 (2 часа)

Встроенные объекты JavaScript. Строки и массивы

Объекты представляют собой программные единицы, обладающие заданными свойствами. Как любой объект, объект JavaScript, обладает свойствами (данными) и методами (функциями) для их обработки. Программный код встроенных объектов JavaScript недоступен.

Управление web-страницами с помощью сценариев, созданных на JavaScript, заключается в использовании и изменении свойств объектов HTML-документа и самого браузера.

Встроенные объекты имеют фиксированные названия. Наиболее важными объектами в разработке web-сайтов являются String (символьные строки), Array (массивы), Math (математические формулы и константы), Date (работа с датами).

Объекты, названия которых совпадают с их фиксированными названиями, называются статическими. Можно создавать и экземпляры (копии) статических объектов, имеющие собственные имена и наследующие все свойства и методы статических объектов.

Встроенные объекты имеют прототипы (prototype), позволяющие добавлять новые свойства и методы.

Объект String

С помощью объекта String можно создавать строку или строковый объект. Синтаксис:

имя_переменной = new String (“значение”)

Создать строковый объект можно и с помощью обычного оператора присваивания:

имя_переменной = “значение”

или

var имя_переменной = “значение”

Например,

```
mystring1 = new String ( “строка”)
mystring2 = “строка”
```

Свойство String

length – длина (количество символов), включая пробелы.

Доступ к свойствам и методам объекта осуществляется через операцию точка. Например,

```
str = “весна”
str.length          // значение равно 5
“лето”.length       // значение равно 4
```

Методы String

Как и любые функции, методы могут иметь параметры. Параметры перечисляются через запятую в круглых скобках, стоящих после имени метода.

chartAt (индекс) – возвращает символ, занимающий в строке указанную позицию. Индекс является числом. Необходимо помнить, что нумерация элементов строки начинается с нуля.

Примеры

```
y = “Осень”.charAt(3)            // значение “н”
str = “Зима”
str.charAt (str . length – 2)    // значение “м”
```

charCodeAt(индекс) – преобразует символ в указанной позиции в его код. Поддерживает систему кодов Unicode, NN4 – ISO-Latin1.

fromCharCode(номер [1, номер2[, номер3, …, номерn]]) – возвращает строку символов, числовые коды символов которой указаны в строке параметров.

concat(строка) – конкатенация (слияние) строк.

Синтаксис: строка1.concat(строка2)

Возвращает строку, полученную дописыванием символов строки2 к строке1.

Пример

```
x = “Петр”
y = x .concat (“Семенович”)          // результат “Петр Семенович”
```

index0f(строка_поиска [, индекс]) - поиск строки, указанной параметром. Метод возвращает индекс первого вхождения строки. Поиск в пустой строке возвращает -1. Второй параметр, не является обязательным, о чем говоря квадратные скобки. Индекс указывает позицию, с которой начинается поиск.

lastIndex0f (строка_поиска [, индекс]) – поиск первого вхождения строки, указанной параметром. Причем поиск начинается с конца исходной строки, но возвращаемый индекс отсчитывается сначала.

localeCompare (строка) – сравнение строк в кодировке Unicode, то есть с учетом используемого браузером языка общения с пользователем. Синтаксис:

строка1.localeCompare(строка2)

Если строки одинаковы, метод возвращает 0. Если строка1 меньше, чем строка2, метод возвращается отрицательное число, в противном случае - положительное.

slice(индекс1[, индекс2]) – возвращает подстроку исходной строки, начальный и конечный индексы которой указываются параметрами, за исключением последнего символа. Второй параметр не обязателен. Если он не указан – подразумевается до конца строки.

split(разделитель [, ограничитель]) – возвращает массив элементов, полученных из исходной строки. Первый параметр – строка символов, используемая в качестве разделителя строки на элементы. Второй параметр – число, указывающее количество элементов возвращаемого массива из строки, полученной при разделении. Второй параметр необязателен. Если разделитель – пустая строка, возвращается массив символов строки.

Например,

x = “Привет всем!”

x.split(“ ”) // значение – массив из элементов “Привет”, “всем!”

x. split(“е”) // значение – массив из элементов “Прив”, “т вс”, “м!”

substr(индекс[, длина]) – возвращает подстроку исходной строки, начальный индекс и длина, которой указываются параметрами. Если второй параметр не указан, возвращается подстрока с начальной позиции до конца.

substring(индекс1, индекс2) – возвращает подстроку исходной строки, с позиции1 до позиции2.

toLocaleLowerCase( ), toLowerCase( ) – переводят строку в нижний регистр.

toLocaleUpperCase( ), toUpperCase ( ) – переводят строку в верхний регистр.

Тексты web-страниц, как правило, создаются и форматируются с помощью тегов HTML. Это же можно сделать средствами JavaScript.

Например, для вывода строки полужирным шрифтом используют метод bold( ). Данный метод не выводит строку в окно браузера, а лишь форматирует ее. Для вывода строки в HTML-документе используется метод write( ) для объекта document:

```
<HTML>
<SCRIPT>
st = “Доброе утро!”.bold( )
document. write(st)
</SCRIPT>
</HTML>
```

Методы форматирования строк носят названия, соответствующие тегам HTML:

anchor(“anchor_имя”)
blinc( )
bold( )
fixed( )
fontcolor(значение цвета)
fontsize(число от 1 до 7)
italics( )
link(расположение или URL)
big( )
small( )
strike( )
sub( )
sup( )

Объект Array

Массив представляет собой упорядоченный набор данных. Нумерация

элементов массива начинается с нуля. К элементам массива можно обращаться по их порядковому номеру, заключив его в квадратные скобки, расположенные после имени массива. Элементы массива в JavaScript могут быть разного типа.

Существует несколько способов создания массива:

1. имя_массива = new Array ([размерность массива])

Если длина массива не указана, создается пустой массив, не содержащий ни одного элемента. Иначе создается массив указанной длины, все элементы которого имеют значение null. Создав пустой массив, можно присвоить значения его элементам операцией присваивания.

2. инициализация массива при объявлении:

имя_массива = new Array (значение1[, значение2[, … значениеn]])

3. инициализация каждого отдельного элемента массива, подобно свойствам объекта

```
имя_массива = new Array( )
имя_массива. имя_элемента1 = значение1
[имя_массива. имя_элемента2 = значение 2
[… имя_массива. имя_элемента n = значение n] ]
```

Например,

```
child = new Array (4)
child[0]= “Женя”
child[1]= 22
child[2]= “ июнь”
child[3]= 1996
child = new Array ( “Женя”,  22,  “ июнь”, 1996)
y=child.length       // y=4

child = new Array ( )
child.name = “Женя”
child.day= 22
child.month= “ июнь”
child.year= 1996
```

Свойства объекта Array

Свойство length, возвращает количество элементов объекта Array.

Свойство prototype позволяет добавлять новые свойства и методы для всех созданных массивов.

Например,

```
function SumNegative (massiv)
{var s = 0
  for (i=0; i<=massiv.length-1; i++)
          if (massiv[i]<0 s +=massiv[i]
return s
}
mass = new Array(1, -9, 7, -23, -3)
Array.protptype.SumN = SumNegative        //добавляем метод к объекту
```

Massiv.SumN(mass) // применяем метод SumN к массиву mass

Многомерные массивы

Для создания многомерного массива требуется указать все его размерности, заключив каждую из них в квадратные скобки.

Например,

matrix = new Array ( )
matrix[0] = new Array (2, 3, 8)
matrix[1] = new Array (1, -3, 7)
matrix[2] = new Array (0, 2, -6) // массив размерности 3 на 3

Методы объекта Array

concat( ) – объединяет два массива в третий и возвращает полученный массив.

Синтаксис: имя_массива3 = имя_массива1.concat(имя_массива2)

join( ) – создает строку из элементов массива с указанным разделителем между ними, возвращает строку символов.

Синтаксис: имя_массива2 = имя_массива1.join(строка)

pop( ) – удаляет последний элемент массива и возвращает его значение.

Синтаксис: имя_массива1.pop( )

push( ) – добавляет к массиву последний элемент, значение которого указанно в качестве параметра и возвращает новую длину массива.

shift( ) – удаляет первый элемент массива и возвращает его значение.

unshift( ) – добавляет к массиву первый элемент, значение которого указанное в качестве аргумента.

reserve( ) – переписывает массив в обратном порядке, возвращает массив.

slise(индекс1[, индекс2]) – создает массив из элементов исходного массива с индексами указанного диапазона. Возвращает массив. Если второй индекс не указан, то новый массив создается из элементов с индекса1 до конца исходного массива.

sort( ) – упорядочивает элементы массива. Если параметр не указан, сортировка производится на основе ASCII-кодов символов значений, что удобно для строк, но не походит для чисел. Параметром может служить имя функции сравнивающей два элемента массива.

Пример,

massiv = new Array (7, 1, 34, 5, 63)
function cmp (x, y)
{ return x - y }
massiv.sort(cmp) //массив будет сортироваться по возрастанию

Эта функция дает критерий сортировки.

splice(индекс, количество [, элем1[, элем2[, … элемn]]]) – удаляет (заменяет) из массива элементы. Возвращает массив из удаленных элементов. Первый параметр является индексом первого удаляемого элемента, второй - коли-

чеством удаляемых элементов. Если указаны необязательные параметры, то происходит замена элементов указанного диапазона на указанные значения параметров. Но это справедливо, если второй параметр не равен нулю.

Пример

```
b = new Array( “один”, 2, 3,  4, “пять”)
c = b.splice(1, 3, “два”,  “три”, “четыре”)
// массив b из элементов «один», «два», «три», «четыре», «пять»
// массив c из элементов 2. 3, 4
```

toLocaleString( ), toString( ) – преобразуют содержимое массива в символьную строку.

Задание

1. Создайте пользовательскую функцию для работы со строками с использованием методов объекта String.

Вариант 1.

Функция вставки строки в исходную строку. Функция должна иметь три параметра: исходную строку, вставляемую строку и позицию вставки.

Вариант 2.

Функция замены в исходной строке все вхождения заданной подстроки на подстроку замены. Функция должна иметь три параметра: исходную строку, заменяемую подстроку и подстроку, которой следует заменить все вхождения заменяемой подстроки.

Вариант 3.

Функция удаления лишних пробелов в начале исходной строки.

Вариант 4.

Функция удаления лишних пробелов в конце строки.

Вариант 5.

Функция удаления слов, длина которых меньше заданного размера. Функция должна иметь два параметра, исходную строку и длину удаляемых слов.

Вариант 6.

Функция удаления в строке одинаковых слов.

2. Изучите функции форматирования строк. Продемонстрируйте работу функции из задания 1, отформатировав вновь полученную строку тремя различными способами в HTML-документе.

3. Создайте функцию для работы с объектом Array.

Вариант 1.

Замена минимального элемента значением, заданным как параметр функции.

Вариант 2.

Сортировка по возрастанию элементов массива, расположенных между максимальным и минимальным его элементами.

Вариант 3.

Сортировка по убыванию элементов массива, расположенных до максимально-

го значения.

Вариант 4.

Удаление из массива минимального элемента, и добавление утроенного найденного значения в качестве первого элемента.

Вариант 5.

Удаление максимального элемента массива и добавление найденного значения, умноженного на пять, в качестве последнего элемента массива.

Вариант 6.

Создание массива из элементов исходного, расположенных между максимальным и минимальным его элементами.

4. Создайте 2 массива. Добавьте созданную функцию из задания 3 в качестве нового свойства ко всем созданным массивам.

Лабораторная работа № 4 (2часа)

Работа с окнами

Главное окно браузера создается автоматически при запуске браузера. С помощью сценария можно создать любое количество окон, а также разбить окно на несколько прямоугольных областей, называемых фреймами. Окну браузера соответствует объект window, а HTML-документу, загруженному в окно, соответствует объект document. Эти объекты могут содержать в себе другие объекты. В частности, объект document входит в состав window.

Доступ к свойствам и методам данного объекта происходит, как и в других объектах, через точку. Поскольку объект document является подобъектом объекта window, ссылка на HTML-документ, загруженный в текущее окно: window.document. Объект document имеет метод write (запись строки в текущий HTML-документ).

Для его применения используют window.document.write(строка). Объект окна window - корневой объект, имеющий свои подобъекты. Например, location хранит информацию об URL-адресе загруженного документа, screen – данные о возможностях экрана монитора пользователя.

В объектной модели документа объекты сгруппированы в *коллекции*. Коллекция – промежуточный объект, содержащий объекты собственно документа. Коллекция является упорядоченным массивом объектов, отсортированных в порядке упоминания соответствующих им элементов в HTML-документе. Индексация объектов в коллекции начинается с нуля. Синтаксис обращения к элементам коллекции аналогичен синтаксису обращению к элементам массива. Коллекция имеет длину – свойство length.

Коллекция всех графических изображений документа называется images, коллекция всех форм – forms, ссылок – links. Коллекция всех объектов документа называется all.

Один и тот же объект может входить в частную коллекцию (например, images), но он обязательно входит в коллекцию all. При этом его индексы могут быть разными в разных коллекциях.

При использовании документа, загруженного в текущее окно, объект window можно не упоминать, а сразу начинать с объекта document.

Например,

document. images(0)

Вместо индекса можно использовать значение атрибута ID в теге, который определяет соответствующий элемент HTML-документа.

Однако универсальный способ обращения к объектам документа – обращение посредством коллекции all.

С помощью сценария можно создавать любое количество окон. Для этого применяется метод open( ):

window. open(параметры)

Данному методу передаются следующие необязательные параметры:

адрес документа, который нужно загрузить в создаваемое окно;

имя окна (как имя переменной);

строка описания свойств окна (features).

В строке свойств записываются пары свойство = значение, которые отделяются друг от друга запятыми.

Свойства, передаваемые в строке features

Свойство	Значения	Описание
channel mode	yes, no, 1, 0	Показывает элементы управления channel
directories	yes, no, 1, 0	Включают кнопки каталога
ful1screen	yes, no, 1, 0	Полностью разворачивает окно
height	число	Высота окна в пикселях
left	число	Положение по горизонтали относительно левого края экран в пикселях
location	yes, no, 1, 0	Текстовое поле Address
menubar	yes, no, 1, 0	Стандартное меню браузера
resizeable	yes, no, 1, 0	Возможность пользователя изменять размер окна
scrollbars	yes, no, 1, 0	Горизонтальные и вертикальные полосы прокрутки
status	yes, no, 1, 0	Стандартная строка состояния
toolbar	yes, no, 1, 0	Включает панели инструментов браузера
top	число	Положение по вертикали относительно верхнего края экрана в пикселях
width	число	Ширина окна в пикселях

Примеры

window. open (“mypage.htm”, “NewWin”, “height=150, width=300”)

window. open (“mypage.htm”)

strfeatures = “top=100, left=15, height=250, width=300, location=no”

window. open (“www.amsu.ru”, strfeatures)

Вместо строки strfeatures можно использовать значение true, тогда указанный документ загружается в существующее окно, вытесняя предыдущий документ.

Метод window.open() возвращает ссылку на объект окна, сохранив которую, можно использовать позднее, например, при закрытии окна.

Для закрытия используют метод close(). Однако, выражение window.close() закрывает главное окно. Для закрытия других окон используют ссылки.

Пример

```
var str = window. open ( “mypage.htm”, “моя страница”)
str.close()
```

Объект document является центральным в иерархической объектной модели. Он предоставляет всю информацию о HTML-документе с помощью коллекций и свойств и множество методов для работы с документами.

Коллекция document

all	все теги и элементы основной части документа
anchor	якоря (закладки) документа
applets	все объекты документа, включая встроенные элементы управления, графические элементы, апплеты, внедренные объекты
embeds	все внедренные объекты документа
forms	все формы на странице
frames	фреймы, определенные в теге <FRAMESET>
images	графические элементы
links	ссылки и блоки <AREA>
plugins	другое название внедренных документов
scripts	все разделы <SCRIPT> на странице
styleSheets	контейнерные свойства стиля, определенные в документе

Методы document

clear	очищает выделенный участок
close	закрывает текущее окно браузера
createElement	создает экземпляр элемента для выделенного тега
elementFromPoint	возвращает элемент с заданными координатами
execCommand	выполняет команду над выделенной областью
open	открывает документ
queryCommandEnabled	сообщает, доступна ли данная команда
queryCommandIndeterm	сообщает, если данная команда имеет неопределенный статус
queryCommandState	возвращает текущее состояние команды
queryCommandSupported	сообщает, поддерживается ли данная команда
queryCommandText	возвращает строку, с которой работает команда
queryCommandValue	возвращает значение команды, определенное для документа или объекта TextRange

write (writeln)	записывает текст и код HTML в документ, находящийся в указанном окне

Объект window кроме дочерних объектов имеет свои методы, свойства, события.

Свойства window:

parent	возвращает родительское окно для текущего
self	возвращает ссылку на текущее окно
top	возвращает ссылку на главное окно
name	название окна
opener	окно, создаваемое текущим
closed	сообщает, если окно закрыто
status	текст, показываемый в строке состояния браузера
defaultStatus	текст по умолчанию строки состояния браузера
returnValue	позволяет определить возвращаемое значение для события или диалогового окна
client	ссылка, возвращаемая объект навигатора браузеру
document	ссылка только для чтения на объект окна document
event	ссылка только для чтения на глобальный объект event
history	ссылка только для чтения на объект окна history
location	ссылка только для чтения на объект окна location
navigator	ссылка только для чтения на объект окна navigator
screen	ссылка только для чтения на объект окна screen

Например,

window.status = «работает сценарий»

Свойство parent позволяет обратиться к объекту, расположенному в иерархии на одну ступень выше. Для перемещения на две ступени выше используют parent.parent. Для обращения к самому главному окну – окну браузера, используют свойство top.

Свойство status используют для вывода сообщений во время работы сценария. Например, window. status = “сценарий работает”

Методы window

open( )	открывает новое окно браузера
close( )	закрывает текущее окно браузера
showHelp( )	показывает окно подсказки как диалоговое
showModalDialog( )	показывает новое модальное(диалоговое) окно
alert( )	окно предупреждения с сообщением и кнопкой ОК
prompt( )	окно приглашения с сообщение, текстовым полем и кнопками ОК и Cancel (Отмена)
confirm( )	окно подтверждения с сообщением и кнопками ОК и Cancel
navigate( )	загружает другую страницу с указанным адресом
blur( )	убирает фокус с текущей страницы

focus( )	устанавливает страницу в фокус
scroll( )	разворачивает окно на заданную ширину и высоту
setInterval( )	указывает процедуре выполняться автоматически через заданное число миллисекунд
setTimeout( )	запускает программу через заданное количество миллисекунд после загрузки страницы
clearInterval( )	обнуляет таймер, заданный методом setInterval( )
clearTimeout( )	обнуляет таймер, заданный методом setTimeout( )
execScript( )	выполняет код сценария, по умолчанию Jscript

Рассмотренные выше методы позволяют работать с независимыми (немодальными) окнами. Для создания модального окна используется метод showModalDialog(). В качестве параметра данный метод принимает адрес документа (файла), имя окна, и строку свойств.

При работе с модальными окнами пользователь не может обратиться к другим окнам, в том числе и к главному. Окна, создаваемые методами alert(), promt(), confirm() являются модальными.

Одним из главных назначений сценариев в HTML-документе является обработка событий, таких как щелчок кнопки мыши по элементу документа, помещение указателя мыши на элемент, нажатие клавиши и др. Для одного и того же элемента можно определить несколько событий на которые он будет реагировать.

Сообщение о событии формируется в виде объекта, т.е. контейнера для хранения информации. Объект события в одном из свойств содержит ссылку на элемент, с которым связано данное событие (на кнопку, изображение и т.п.)

Обычно обработчики событий оформляются в виде функций, определения которых помещаются в контейнерный тег <SCRIPT>.

События window

onblur	выход окна из фокуса
onfocus	окно становится активным
onhelp	нажатие пользователем клавиши F1
onresize	изменение пользователем размеров окна
onscroll	прокрутка окна пользователем
onerror	ошибка при передаче
onbeforeunload	для сохранения данных перед выгрузкой страницы
onload	страница полностью загружена
onunload	непосредственно перед выгрузкой страницы

В случае открытия нескольких окон браузера, пользователь может переключаться между ними, переводя фокус с одного окна на другое. Эти действия инициируются программными событиями onblur и onfocus. Эти же действия можно вызвать, используя методы blur и focus.

Событие onerror происходит при ошибки загрузки страницы или ее элемента. Его можно использовать в программе при попытке вновь загрузить страницу. Например,

```
<SCRIPT>
function window.onerror( ) {
alert (“ Ошибка! Повтори попытку!”)
}
</SCRIPT>
```

События document

onafterupdate	окончание передачи данных
onbeforeupdate	перед выгрузкой страницы
onclick	при щелчке левой кнопкой мыши
ondblclick	при двойном щелчке левой кнопкой мыши
ondragstart	при возникновении перетаскивания
onerror	ошибка при передаче
onhelp	нажатие клавиши F1
onkeydown	нажатие клавиши
onkeypress	возникает при нажатии клавиши и продолжается при удержании клавиши в нажатом состоянии
onkeyup	пользователь отпускает клавишу
onload	при полной загрузке документа
onmousedown	при нажатии кнопки мыши
omousemove	при перемещении указателя мыши
onmouseout	когда указатель мыши выходит за границы элемента
onmouseover	когда указатель мыши входит на документ
onmouseup	пользователь отпускает кнопку мыши
onreadystatechange	возникает при изменении свойства readystate
onselectstart	когда пользователем впервые запускается выделенная часть документа

Динамическое изменение элементов документа

Элементы HTML-документа задаются тегами, большинство из которых имеют параметры (атрибуты). В объектной модели документа тегам соответствуют объекты, а атрибутам – свойства этих объектов. Названия свойств объектов, как правило, совпадают с названиями атрибутов, но записываются в нижнем регистре.

Наиболее удобный способ динамического изменения HTML-документа основан на использовании свойств innerText, outerText, innerHTML и outerHTML. С их помощью можно получить доступ к содержимому элемента. Изменяя значения перечисленных свойств можно частично или полностью изменить сам элемент. Например, можно изменить только надпись на кнопке, а можно превратить кнопку в изображение или Flash-анимацию.

Значением свойства innerText является все текстовое содержимое между открывающим и закрывающим тегами элемента. Внутренние теги игнорируются. Данные открывающего и закрывающего тегов соответствующего элемента также не входят.

В отличие от предыдущего свойство outerText включает в себя данные открывающего и закрывающего тегов. Таким образом, outerTextесть весь текст, содержащийся в контейнере, включая его внешние теги. Например, задан HTML-код:

<DIV ID = “my” >
<A HREF = ‘raznoe.htm’>
<IMG SRC = ‘picture.jpg> Ссылка на раздел <B> Разное <\B>
</A>
<\DIV>

Здесь свойства innerText и outerText для элемента, заданного контейнерным тегом <DIV>, совпадают:

document.all.my.innerText //значение равно – «Ссылка на раздел Разное»

При присвоении свойствам innerText и outerText новых значений нужно помнить, что если значения содержат теги, то они не интерпретируются, а воспринимаются как обычный текст.

Свойство innerHTML содержит внутренний HTML-код контейнера элемента. Присвоение этому свойству нового значения, содержащего HTML-код, приводит к интерпретации кода. Свойство outerText дополнительно включает внешние открывающие и закрывающие теги элемента.

Для приведенного HTML-кода значение document. all. my. innerHTML равно “<A HREF = ‘raznoe.htm’> <IMG SRC = ‘picture.jpg’> Ссылка на раздел <B> Разное <\B> </A>”/

Значение document. all. my. outerHTML – “<DIV ID = “my” > <A HREF = ‘raznoe.htm’> <IMG SRC = ‘picture.jpg> Ссылка на раздел <B> Разное <\B> </A> <\DIV>”.

Если в сценарии выполнить выражение

document.all.my.innerHTML = “<BUTTON>Щелкни здесь</BUTTON>”

ссылка, изображение и текст будут заменены кнопкой с надписью «Щелкни здесь». При этом контейнерный тег “<DIV ID = “my” > сохранится. Если аналогичным образом использовать outerHTML, кнопка также появится, но уже без контейнера “<DIV ID = “my” >.

Свойства innerHTML и outerHTML могут применяться к элементам, заданным неконтейнерными тегами. Тогда innerHTML и outerHTML совпадают.

Для ускорения загрузки графики можно использовать следующие возможности JavaScript. Можно организовать предварительную загрузку изображений в кэш-память браузера, не отображая их на экране. Это особенно эффективно при начальной загрузке страницы. Пока изображения загружаются в память, оставаясь невидимыми, пользователь может рассматривать текстовую информацию.

Для предварительной загрузки изображения требуется создать его объект в памяти браузера. Это можно сделать следующим выражением:

myimg = new Image (ширина, высота)

Параметры должны соответствовать значениям атрибутов WIDTH и HEIGHT тега <IMG>, который используется для отображения предварительно загруженного изображения.

Для созданного в памяти объекта изображения можно создать имя или URL-адрес графического файла:

myimg.scr = “URL-адрес изображения”

что предписывает браузеру загрузить изображения без его отображения.

После загрузки в кэш-память всех изображений и загрузки всего документа можно сделать их видимыми. Для этого свойству src элемента <IMG> нужно присвоить значение этого же свойства объекта изображения в кэш-памяти. Например,

document. images[0]. scr = myimg. scr

Здесь слева указано свойство scr первого в документе элемента, соответствующего тега <IMG>, справа – свойство scr объекта изображения в кэш-памяти.

С помощью JavaScript можно через заданный интервал времени запускать код или функцию. При этом создается эффект одновременного (параллельного) выполнения вычислительных процессов.

Для организации повторения через заданный интервал выполнения некоторого выражения служит метод setInterval( ) объекта window:

setInterval(выражение, период, [, язык])

Первым параметром является строка, например вызов функции. Период указывается в миллисекундах. Третий параметр – необязательный, в котором указывается язык с помощью которого написано заданное выражение. По умолчанию – JavaScript.

Метод setInterval( ) возвращает некоторое целое число – идентификатор временного интервала, который может быть использован в дальнейшем, например для прекращения выполнения процесса методом clearInterval( ).Например,

```
var pr = setInterval( “myfunc(), 100” )
if (confirm ( “Прервать процесс?”) )
clearInterval(pr)
```

Если требуется выполнить действие с некоторой временной задержкой, используется метод setTimeout( ), имеющий синтаксис аналогичный setInterval( ). Для отмены задержки процесса, запущенного setTimeout( ), используют clearTimeout( ).

Задание

Создать HTML-документ, расположив в нем список названий графических объектов, одно исходное отображение, две кнопки.

Щелчок на элементе списка должен приводить к изменению цвета элемента списка и отображению соответствующего графического элемента, и соответствующего ему тестового сопровождения.

При этом изображение кнопки должно быть также изменено.

Щелчок по первой кнопке через 5 секунд должен инициализировать функцию открытия документа в окне, заданного размера, определенного размера текстового поля и название. Окно должно содержать горизонтальные и вертикальные полосы прокрутки, размер окна не должен изменяться по желанию пользователя. Выведенный в окне текст должен быть синим на сером фоне, иметь выделенный заголовок, ссылки на другие объекты. По выбору продемонстрируйте по пять событий и свойств объектов window и document.

Это действие может быть отменено с помощью второй кнопки.

Лабораторная работа №5 (2 часа)

Работа с фреймами

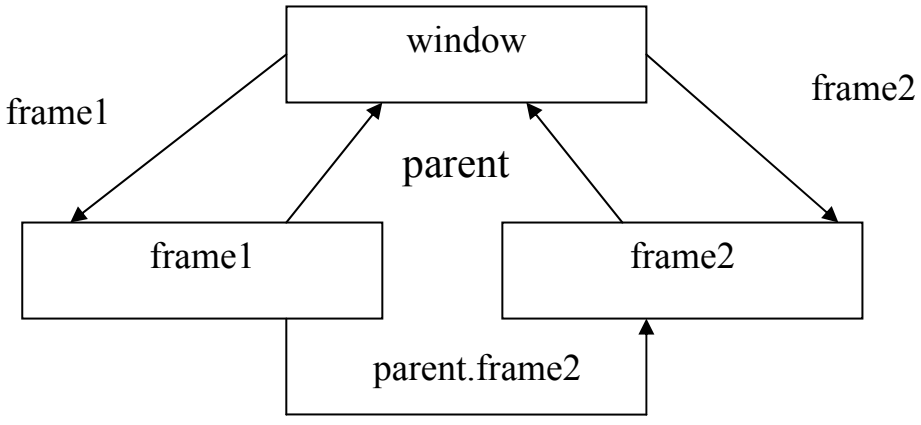
Фрейм – прямоугольная область окна браузера, в которую можно загрузить HTML-документ. Разбиение окна браузера на отдельные окна производится с помощью тега <FRAMESET>, внутрь которого вставляются теги <FRAME> с атрибутами, указывающими имя фрейма и адрес HTML-документа.

Пример

```
<HTML>
<FRAMESET ROWS= “30%, 70%”>
<FRAMESET SRC= “документ1.htm” NAME = “frame1” >
<FRAMESET SRC= “документ2.htm” NAME = “frame2” >
</FRAMESET>
</HTML>
```

Здесь применяется вертикальное расположение фреймов. Для горизонтального размещения фреймов вместо атрибута ROWS в теге <FRAMESET> использовать COLS.

Используя вложение тега <FRAMESET>, можно разбить уже имеющийся фрейм на два других.



При разбиении окна на фреймы и, в свою очередь, фрейма на другие фреймы возникают отношения родитель-потомок. Каждому из фреймов соответствует свой объект document. Обеспечение доступа к в иерархии объектов представлено на рисунке.

Так при обращении из одного фрейма-потомка к другому, необходимо помнить, что прямой связи между фреймами-потомками не существует. Поэтому сначала нужно обратиться к родительскому окну, а затем к его второму потомку:

parent. frame2. document. write(“ Привет от первого фрейма.“)

Можно изменить элемент одного фрейма из другого. Например, при щелчке на тексте в правом фрейме в левом изменится один из текстовых документов. Тогда документ в левом фрейме с именем LEFT:

```
<HTML>
Делай раз<BR>
Делай два
<H1 ID = "XXX"> Делай три </H1>
</HTML>
```

Документ в правом фрейме:

```
<HTML>
<SCRIPT>
function change( ) {
parent.LEFT.document.all.XXX. innerText = "Делай пять!!!!"
}
</SCRIPT>
<H1 onclick = "change( )"> Щелкни здесь</H1>
</HTML>
```

В теле функции change( ) происходит обращение к левому фрейму с именем LEFT (задается в установочном HTML-файле) через parent. Изменение элемента происходит за счет присвоения значения свойству innerText. Кроме данного свойства можно использовать outerText, innerHTML или outerHTML.

Важно, что изменения в одном фрейме по событию в другом происходят без перезагрузки HTML-документа.

Фреймы удобно использовать при создании навигационных панелей. В одном фрейме располагаются ссылки, а второй предназначен для отображения документов, вызываемых при активизации соответствующих ссылок.

Пример

```
// установочный файл frame.htm
<HTML>
<FRAMESET COLS = “25%, 75%”>
<FRAME SRC = “menu.htm” NAME = “menu” >
<FRAME SRC = “start.htm” NAME = “main” >
</FRAMESET>
</HTML>
```

Здесь start.htm – документ, который первоначально показан во фрейме main.

```
//menu.htm – навигационная панель
<HTML>
```

```
<SCRIPT>
function load (url) {
parent. main. location. href = url;
}
</SCRIPT>
<BODY>
<A HREF = “javascript:load(‘первый.htm’)”>Первый </A>
<A HREF = “второй.htm” TARGET = “main”> Второй </A>
<A HREF = “третий.htm” TARGET = “top”> Третий </A>
</BODY>
</HTML>
```

В примере окно браузера разделено на два фрейма. Первый из них играет роль навигационной панели, а второй – окна для отображения документов. Продемонстрированы два способа загрузки новой страницы во фрейм main. В первом случае используется функция load( ), параметр которой указывает, какой файл следует загрузить. При этом место, в которое он загружается, определяется самой функцией load( ). Во второй ссылке используется атрибут TARGET. В третьем ссылке демонстрируется, как можно избавиться от фреймов.

Для удаления фрейма с помощью load( ) достаточно записать:

parent. location. href = url

Атрибут TARGET в теге ссылки <A HREF> обычно применяется в случаях, когда требуется загрузить одну страницу в один фрейм. Язык сценариев используют при необходимости выполнения нескольких действий.

Для ссылок из родительского окна к объектам его дочерних фреймов можно использовать коллекцию frames. Обращение к определенному фрейму из этой коллекции возможно по индексу или по имени фрейма:

window. frames [индекс]

window. имя_фрейма

При обращении к объекту документа, загруженного во фрейм, следует сначала упомянуть объект document:

window. frames(0). document. all. Myinput. Value

window. LEFT. document. all. Myinput. Value

Ссылка из дочернего фрейма на родительский - осуществляется с использованием parent.

При использовании top следует учитывать, что создаваемый сайт может быть загружен в другой. Тогда объект top окажется объектом другого сайта. Поэтому лучше использовать parent для ссылок на вышестоящее окно или фрейм.

Ссылки top или self используют для предотвращения отображения сайта внутри фреймов другого сайта. Сценарий, выполняющий это, следует разместить в начале документа, например:

```
<SCRIPT>
if (top != self )
```

top. Location = location

</SCRIPT>

т.е., ссылка на свойство top на верхнее окно, должна совпадать со ссылкой self на текущее окно.

Для вставки одного HTML-документа в тело другого средствами браузера служит контейнерный тег <IFRAME>:

<IFRAME SRC = “адрес документа” > </IFRAME>

Данный элемент представляет собой прямоугольную область с прокруткой или без. Такое окно называют плавающим фреймом. Данный документ можно позиционировать с помощью параметров таблицы стилей (тег <STYLE> или атрибут STYLE).

Плавающий фрейм аналогичен обычному фрейму. При создании он помещается в коллекцию frames. Среди его свойств широко используется align – выравнивание плавающего фрейма относительно окружающего содержимого документа. Его возможные значения:

absbotton – выравнивает нижнюю границу фрейма по подстрочной линии символов окружающего текста,

absmiddle – выравнивает середину границу фрейма по центральной линии между top и absbotton окружающего текста,

baseline – выравнивает нижнюю границу фрейма по базовой линии окружающего текста,

botton – совпадает с baseline (только IE)

left – выравнивает фрейм по левому краю элемента-контейнера,

middle – выравнивает воображаемую центральную линию окружающего текста по воображаемой центральной линии фрейма,

right – выравнивает фрейм по правому краю элемента-контейнера,

texttop – выравнивает верхнюю границу фрейма по надстрочной линии символов окружающего текста,

top – выравнивает верхнюю границу фрейма по верхней границе окружающего текста.

Задание

Вариант 1. Окно браузера поделить на два фрейма. В левом расположить небольшое изображение. Организовать возможность вывода полномасштабного изображения в левом окне по щелчку мыши на миниатюре изображения в правом фрейме. В левом фрейме дополнительно организовать навигационную модель. Создать новый HTML-документ и вставить его в ранее созданный, как плавающий вертикальный фрейм, выравнивая нижнюю границу фрейма по базовой линии окружающего текста.

Вариант 2. Окно браузера поделить на три фрейма. В левом расположить небольшое изображение. Организовать возможность вывода полномасштабного изображения в среднем окне по щелчку мыши на миниатюре изображения в правом фрейме. В левом фрейме дополнительно организовать навигационную модель. Создать новый HTML-документ и вставить его в ранее созданный, как

плавающий вертикальный фрейм, выравнивая воображаемую центральную линию окружающего текста по воображаемой центральной линии фрейма.

Вариант 3. Окно браузера поделить на три фрейма. В верхнем фрейме расположить небольшое изображение. Организовать возможность вывода полномасштабного изображения в среднем окне по щелчку мыши на миниатюре изображения в верхнем фрейме. В верхнем фрейме дополнительно организовать навигационную модель для среднего фрейма. Создать новый HTML-документ и вставить его в ранее созданный, как плавающий фрейм, выравнивая нижнюю границу фрейма по верхней границе текста.

Вариант 4. Окно браузера поделить на два фрейма. В нижнем расположить небольшое изображение. Организовать возможность вывода полномасштабного изображения в верхнем окне по щелчку мыши на миниатюре изображения в нижнем фрейме. Создать вертикальный, относительно двух ранее созданных, фрейм. В нем организовать навигационную модель для первого фрейма. Создать новый HTML-документ и вставить его в ранее созданный, как плавающий фрейм, выравнивая нижнюю границу фрейма по верхней границе текста.

Вариант 5. Окно браузера поделить на два фрейма. В нижнем расположить небольшое изображение. Организовать возможность вывода полномасштабного изображения в верхнем окне по щелчку мыши на миниатюре изображения в нижнем фрейме. Создать вертикальный, относительно верхнего, из ранее созданных, фрейм. В нем организовать навигационную модель для второго фрейма. Создать 2 новых HTML-документа и вставить их в ранее созданный, как плавающие, выравнивая верхнюю границу фрейма по надстрочной линии символов окружающего текста.

Вариант 6. Окно браузера поделить на четыре горизонтальных фрейма. В правом расположить небольшое изображение. Организовать возможность вывода полномасштабного изображения в самом левом окне по щелчку мыши на миниатюре изображения правого фрейма. Создать вертикальный, относительно ранее созданных, фрейм. В нем организовать навигационную модель для второго фрейма. Создать новый HTML-документ и вставить его в ранее созданный документ, как плавающий фрейм, выравнивая верхнюю границу фрейма по верхней границе окружающего текста.

Лабораторная работа №6 (2 часа)
Простые визуальные эффекты

Смена изображений

Для смены одного изображения на другое достаточно с помощью сценария заменить значение атрибута SCR тега <IMG>. Например:

```
<HTML>
<IMG ID = “myimg” SRC= ‘pict1.gif ’
onclick = “document.all.myimg.src = ‘pict2.gif” ”>
</HTML>
```

Здесь смена изображения из файла pict1.gif на изображение из файла pict2 происходит при первом щелчке на нем. Последующие щелчки не приведут к видимым изменениям, поскольку второе изображение будет заменяться им же. Чтобы при повторном щелчке происходила замена изображения на предыдущее необходимо создать переменную-триггер (флаг), принимающий одно из двух возможных значений, по которому можно определить, какое из двух значений надо отобразить.

```
<HTML>
<IMG ID = “myimg” SRC= ‘pict1.gif ’
onclick = “ imgchange( )”>
<SCRIPT>
var flag=false
function  imgchange( ) {
if (flag) document. all. myimg. src = “pict1.gif ”
        else document. all. myimg. src = “pict2.gif ”
flag=!flag
}
</SCRIPT>
</HTML>
```

Цветовые эффекты

Задача изменения цвета кнопки при наведении на нее указателя мыши и возвращения в первоначальное состояние при удалении указателя с кнопки может быть решена следующим образом:

```
<HTML>
<STYLE>
mystile {font-weight.bold; background-color: a0a0ao} // серый цвет кнопок
</STYLE>

<FORM onmouseover = “colorchange ( ‘yellow’)” onmouseout
= “colorchange (‘a0a0ao’)”>
<INPUT TYPE = “BUTTON” VALUE = “Кнопка” CLASS = “mystile”
onclick = “alert( ‘Вы нажали кнопку’)” >
</FORM>
<SCRIPT>
function colorchange (color){
if (event. scrElement.type = = “button”)
        event. scrElement. style. backgroundColor = color;
}
</SCRIPT>
</HTML>
```

Функция colorchange( ) проверяет, является ли инициатор события объектом типа button. Если это так, то цвет кнопки меняется. Без этой проверки менялся бы не только цвет кнопок, но и текста.

Аналогичным способом можно изменять цвет фрагментов текста. Но в этом случае текст должен быть заключен в контейнер, например в теги <P>, <B>, <I>, <DIV>.

Можно создать прямоугольную рамку, окаймляющую текст, которая периодически изменяет цвет. Рамка создается тегами одноячеечной таблицы с заданием нужных атрибутов и параметров стиля:

```
<TABLE ID= “tab” BORDER=1 WIDH=200 style= “border:10 solid : red”>
<TR><TD> Доброе утро! </TR></TD>
</TABLE>
```

Функция изменения цвета:

```
<SCRIPT>
function flash( ) {
if ( !document.all) return null;
if (tab.style.borderColor = = ‘red’)      tab.style.borderColor = ‘yellow’
else tab.style.borderColor = ‘red’;
}
setInterval (“flash”, 500);                  //мигание рамки с интервалом 500 мс
</SCRIPT>
```

Объемные заголовки

Объемные заголовки часто используются на веб-страницах. Идея создания объемного заголовка состоит в наложении нескольких надписей с одинаковым содержанием с некоторым сдвигом по координатам. Наилучший эффект достигается путем подбора цветов надписей (игрой света и тени) с учетом цвета фона. Для этого используют библиотеку стилей. Функция, создающая заголовок с заданными параметрами:

```
        function d3 (text, x, y, tcolor, fsize, fweight, family, zind) {
/*      text – текс заголовка
         x – горизонтальная координата (left)
         y – вертикальная координата (top)
         tcolor – цвет переднего плана
         fsize – размер шрифта (пт)
        fweight – вес (толщина шрифта)
        family – название семейства шрифтов
         zind z-Index          */
if (!text) return null // если текст не указан ничего не выполняется
//значение параметров по умолчанию
if (!x) x=0
if (!y) y=0
if (!tcolor) tcolor=’00aaff ’
if (!fsize)  fsize=36
if (!fweight)  fweight =800
if (!family) ffamily=’arial’
// внутренние настройки
var sd=5, hd=2
```

```
var xzind= “ ”
if (zind) xzind= “; - Index:”+zind
var xstyle =’font-family:’ + family + ‘;font-size:’ + fsize + ‘;font-weight:’ + fweight
+ ‘;’
var xstr = ‘<DIV STYLE = “position: absolute; top:’  + (y +sd ) + ‘; left :’ +
( x + sd ) +  xzind + ‘ “>’
xstr+=‘<P styl e  = “ ’ + xstyle + ‘color: darked”>’ + text + ‘</P></DIV>’
xstr+= ‘<DIV STYLE = “ position: absolute; top:’ + y  + ‘; left :’ +
x  +  xzind + ‘ “>’
xstr+=‘<P styl e  = “ ’ + xstyle + ‘color: silver”>’ + text + ‘</P></DIV>’
xstr+= ‘<DIV STYLE = “ position: absolute; top:’ + ( y + hd )  + ‘; left :’ +
(x +hd)  +  xzind + ‘ “>’
xstr+=‘<P styl e  = “ ’ + xstyle + ‘color:’ + tcolor +   ‘”>’ + text + ‘</P></DIV>’
document.write(xstr)                //запись в документ
}
```

Параметр z-Index позволяет установить слой, в котором находится заголовок, и тем самым указать, будет ли заголовок располагаться над или под другим видимым элементом документа. Элементы с более высоким значением z-Index находятся над элементами, у которых z-Index меньше. Перекрытие элементов с одинаковыми значениями z-Index определяется порядком их следования в HTML-документе.

Вызов приведенной выше функции может выглядеть так:

```
d3 (“это не графика, это просто стиль текста”, 50, 50, ‘blue’, 72, 800, ‘times’)
```

Задание

Выполнить следующие действия на веб-странице:

1. Создать программу для работы с галереей миниатюр. При щелчке кнопкой мыши по миниатюре изображение должно увеличиваться, а затем при щелчке на увеличенном изображении оно должно уменьшаться. Доработайте приведенную в тексте функцию функцию imgchange( ). Для решения этой задачи потребуется массив флагов и функция обработчик, определяющая на каком именно изображении произошел щелчок:

```
var p1=new Array (“pict1.gif ” , … )  //массив имен исходных файлов
var p2=new Array (“pict2.gif ” , … )  //массив имен замещающих файлов
//формирование тегов, описывающих изображения
var xstr =  “ “
for (i=0; i<p1.length; i++)
xstr+= ‘<IMG ID = “i’ + i +’ ” SRC = “ ’+ p1[i]+’ ” onclick = “imgchange( )” >’
}
document.write(xstr)                // запись в документ
```

2. Выполнить замену фрагмента текста с черного на красный при наведении на него указателя мыши.
3. Выделите фрагмент текста мигающей трехцветной рамкой.
4. Создать эффект динамического изменения цвета ссылок. Различать цвета мерцания использованных и неиспользованных ссылок. (Множество цве-

тов задать массивом. Использовать свойства linkColor и linkColor объекта document). Для изменения цвета случайным образом можно использовать метод random( ) (счетчик случайных чисел) встроенного объекта Math. Если требуется получить случайное число x, лежащее в интервале от A до B, то x=A+(B – A)*Math. random( )

5. Создать три объемных заголовка, поэкспериментировав со значениями внутренних параметров sd, hd, а также с заданием параметров по умолчанию.

Лабораторная работа №7 (2 часа)

Применение фильтров

С помощью фильтров каскадных таблиц стилей можно получить разнообразные эффекты: постепенное появление или исчезновение рисунка, плавное преобразование одного изображения в другое, задание степени прозрачности и др.

Фильтр следует понимать как некий инструмент преобразования изображения, взятого из графического файла и вставленного в HTML – документ с помощью тега <IMG>. Следует иметь в виду, что фильтры работают только в IE4+.

Фильтры можно применять не только к графическим объектам, но и к текстам, текстовым областям, кнопкам.

Прозрачность

С помощью фильтра alpha можно установить прозрачность графического объекта. Сквозь прозрачные графические объекты видны нижележащие изображения. Прозрачность имеет несколько вариантов градиентной формы. Например, она может увеличиваться от центра к краям изображения

Фильтр alpha задается с помощью каскадной таблицы стилей и имеет ряд параметров. В примере для графического изображения стиль определяется с помощью атрибута STYLE:

<IMG ID = “myimg” SCR = “ pict. gif ”
STYLE = “position: absolute; top:10; left: 50;
filter: alpha (opacity = 70, style = 3)”>

Здесь целочисленный параметр opacity определяет степень непрозрачности. Значение 0 соответствует полной прозрачности изображения, а 100 – полной непрозрачности. Параметр style задает градиентную форму распределения прозрачности по изображению как целое число от 0 до 3. По умолчанию значение параметра равно 0, и градиент не применяется. Фильтр имеет и другие параметры, определяющие прямоугольную область изображения, к которому применяется фильтр. По умолчанию фильтр применяется ко всему изображению.

Фильтр можно определить в каскадной таблице силей внутри контейнерного тега <STYLE>:

<HTML>

```
<STYLE>
#myimg{position: absolute; top:10; left: 50; filter: alpha (opacity = 70, style = 3)}
</STYLE>
< IMG ID = “myimg” SCR = “ pict. gif ”
</HTML>
```

Доступ к свойствам фильтра в сценарии:

document. all. id_изображения. filters [“имя_фильтра”]. параметр = значение

Для рассмотренного примера это выражение имеет вид:

document. all. myimg. filters [“alpha”]. opacity = 30

Для остальных параметров alpha аналогично.

Для IE5.5+ можно использовать другой синтаксис, в котором в каскадной таблице стилей задается ссылка на специальный компонент и имя фильтра:

```
#myimg { filter: progid: DXImageTransform . Microsoft . alpha
(opacity = 70, style = 3)}
```

Тогда доступ к свойствам фильтра:

document.all.myimg.filters [“DXImageTransform. Microsoft alpha”]. opacity = 30

Трансформация

Фильтр alpha статический. Существуют и динамические фильтры: apply( ) – фиксирует изображение, play( ) – трансформирует, revealtrans( ) – преобразовывает изображение, stop( ) останавливает процесс преобразования при необходимости.

Фильтр revealtrans( ) имеет параметры: duration – длительность преобразования в секундах (число с плавающей точкой) и transition – тип преобразования (целое от 0 до 23).

Для эффекта появления изображения можно воспользоваться фрагментом, который происходит после загрузки документа, т.е. по событию onload:

```
<HTML>
<BODY onload = “transform()” >
<IMG ID = “myimg” SRC = “ pict. gif ” STYLE = “position: absolute; top:10;
left: 50; visibility = “hidden” filter: revealtrans (duration = 3, transition =12)” >
//transition =12 соответствует плавной трансформации
</BODY>
<SCRIPT>
function transform ( ) {                                    //появление изображения
document.all.myimg.style. visibility = “hidden”     // изображение невидимо
myimg. filters ( “revealtrans”). apply( )
myimg.style. visibility = “visible”
myimg. Filters ( “revealtrans”). play( )                    // выполняем преобразования
}
</SCRIPT>
</HTML>
```

Для замены одного изображения на другое необходимо установить начальное и конечное изображение путем присвоения нужных значений свойству src объекта, соответствующего изображению, например фрагментом:
document.all.myimg.src = “pict2. gif ”

Рассмотренный синтаксис воспринимается браузерами IE4+. Для IE5.5+ в каскадной таблице стилей задается ссылка на специальный компонент и имя фильтра. Так для трансформации изображения по щелчку мыши на графическом объекте в другое, и обратно, можно воспользоваться программой:

```
<HTML>
<STYLE>
#myimg{position: absolute; top:10; left: 50; filter: progid: DXImageTransform . Microsoft revealtrans (duration = 3, transition = 12)}
</STYLE>
< IMG ID = “myimg” onclick = “transform( )” SCR = “ ear. gif ”>
<SCRIPT>
 function transform( ) {
//фиксация исходного изображения
myimg. filters (“DXImageTransform . Microsoft revealtrans”). apply ( )
//определение конечного изображения
if (document. all. mying. src. index0f (“ear”)!= -1)
document. all. mying. src = “s.gif”
      else document. all. mying. src = “ear.gif”
//выполняем преобразование
myimg. filters (“DXImageTransform . Microsoft revealtrans”). play ( )
}
</SCRIPT>
</HTML>
```

В браузере IE5.5+ возможно прменение фильтра basicimage, с помощью которого изображение можно повернуть на угол, кратный 90 градусам, задать прозрачность, зеркально отразить и др.

Задание

В HTML-документе из предыдущей лабораторной работы применить к рисункам методы трансформации и изменения прозрачности изображения и фона.

Лабораторная работа №8 (2 часа)
Динамическое изменение элементов документа

Элементы HTML-документа задаются тегами, большинство из которых имеют параметры (атрибуты). В объектной модели документа тегам соответствуют объекты, а атрибутам – свойства этих объектов. Названия свойств объектов, как правило, совпадают с названиями атрибутов, но записываются в нижнем регистре.

Наиболее удобный способ динамического изменения HTML-документа основан на использовании свойств innerText, outerText, innerHTML и outerHTML. С их помощью можно получить доступ к содержимому элемента. Изменяя значения перечисленных свойств можно частично или полностью изменить сам элемент. Например, можно изменить только надпись на кнопке, а можно превратить кнопку в изображение или Flash-анимацию.

Значением свойства innerText является все текстовое содержимое между открывающим и закрывающим тегами элемента. Внутренние теги игнорируются. Данные открывающего и закрывающего тегов соответствующего элемента также не входят.

В отличие от предыдущего свойство outerText включает в себя данные открывающего и закрывающего тегов. Таким образом, outerTextесть весь текст, содержащийся в контейнере, включая его внешние теги. Например, задан HTML-код:

<DIV ID = “my” >
<A HREF = ‘raznoe.htm’>
<IMG SRC = ‘picture.jpg> Ссылка на раздел <B> Разное <\B>
</A>
<\DIV>

Здесь свойства innerText и outerText для элемента, заданного контейнерным тегом <DIV>, совпадают:

document.all.my.innerText //значение равно – «Ссылка на раздел Разное»

При присвоении свойствам innerText и outerText новых значений нужно помнить, что если значения содержат теги, то они не интерпретируются, а воспринимаются как обычный текст.

Свойство innerHTML содержит внутренний HTML-код контейнера элемента. Присвоение этому свойству нового значения, содержащего HTML-код, приводит к интерпретации кода. Свойство outerText дополнительно включает внешние открывающие и закрывающие теги элемента.

Для приведенного HTML-кода значение document. all. my. innerHTML равно “<A HREF = ‘raznoe.htm’> <IMG SRC = ‘picture.jpg> Ссылка на раздел <B> Разное <\B> </A>”/

Значение document. all. my. outerHTML – “<DIV ID = “my” > <A HREF = ‘raznoe.htm’> <IMG SRC = ‘picture.jpg> Ссылка на раздел <B> Разное <\B> </A> <\DIV>”.

Если в сценарии выполнить выражение

document.all.my.innerHTML = “<BUTTON>Щелкни здесь</BUTTON>”

ссылка, изображение и текст будут заменены кнопкой с надписью Щелкни здесь. При этом контейнерный тег “<DIV ID = “my” > сохранится. Если аналогичным образом использовать outerHTML, кнопка также появится, но уже без контейнера “<DIV ID = “my” >.

Свойства innerHTML и outerHTML могут применяться к элементам, заданным неконтейнерными тегами. Тогда innerHTML и outerHTML совпадают.

Для ускорения загрузки графики можно использовать следующие возможности JavaScript. Можно организовать предварительную загрузку изобра-

жений в кэш-память браузера, не отображая их на экране. Это особенно эффективно при начальной загрузке страницы. Пока изображения загружаются в память, оставаясь невидимыми, пользователь может рассматривать текстовую информацию.

Для предварительной загрузки изображения требуется создать его объект в памяти браузера. Это можно сделать следующим выражением:

myimg = new Image (ширина, высота)

Параметры должны соответствовать значениям атрибутов WIDTH и HEIGHT тега <IMG>, который используется для отображения предварительно загруженного изображения.

Для созданного в памяти объекта изображения можно создать имя или URL-адрес графического файла:

myimg.scr = “URL-адрес изображения”

что предписывает браузеру загрузить изображения без его отображения.

После загрузки в кэш-память всех изображений и загрузки всего документа можно сделать их видимыми. Для этого свойству src элемента <IMG> нужно присвоить значение этого же свойства объекта изображения в кэш-памяти. Например,

document. images[0]. scr = myimg. scr

Здесь слева указано свойство scr первого в документе элемента, соответствующего тега <IMG>, справа – свойство scr объекта изображения в кэш-памяти.

С помощью JavaScript можно через заданный интервал времени запускать код или функцию. При этом создается эффект одновременного (параллельного) выполнения вычислительных процессов.

Для организации повторения через заданный интервал выполнения некоторого выражения служит метод setInterval( ) объекта window:

setInterval(выражение, период, [, язык])

Первым параметром является строка, например вызов функции. Период указывается в миллисекундах. Третий параметр – необязательный, в котором указывается язык с помощью которого написано заданное выражение. По умолчанию – JavaScript.

Метод setInterval( ) возвращает некоторое целое число – идентификатор временного интервала, который может быть использован в дальнейшем, например для прекращения выполнения процесса методом clearInterval( ).Например,

```
var pr = setInterval( “myfunc(), 100” )
if (confirm ( “Прервать процесс?”) )
clearInterval(pr)
```

Если требуется выполнить действие с некоторой временной задержкой, используется метод setTimeout( ), имеющий синтаксис аналогичный setInterval( ). Для отмены задержки процесса, запущенного setTimeout( ), используют clear-Timeout( ).

Задание

1. Создать HTML-документ, в котором отображаются список названий графических объектов и одно исходное отображение. Щелчок на элементе списка должен приводить к отображению соответствующего элемента.
2. Создать в HTML-документе две кнопки. Щелчок по кнопке ПУСК открывает через 5 секунд новое окно и загружает в него некоторый документ. Это действие может быть отменено с помощью кнопки ОТМЕНА.

ЧАСТЬ 2. ОСНОВЫ СЕТЕВОГО ПРОГРАММИРОВАНИЯ

Лабораторная работа 9 (2 часа)
Структура заголовка пакета в IPX

Для передачи данных по локальной сети необходимо не только сетевое оборудование, но и сетевое программное обеспечение. Сетевое программное обеспечение включает:

редиктор, отсылающий запросы в сеть, а не на локальный диск;

драйверы сетевых плат, обеспечивающие связь между операционной системой и сетевой платой;

сетевые протоколы для отправки и приема данных.

Суть применения редиктора: заставить приложение на локальной машине полагать, что оно получает данные с локального, а не с сетевого диска, т. е. метод доступа к запрашиваемому файлу должен быть единым.

Редикторы часто называют клиентами. Они обязательно входят в состав программного обеспечения клиентского компьютера.

Программа-редиктор встроена не в протокол, а в интерфейс прикладных программ (API). Если бы не было API, программистам сетевого программного обеспечения пришлось разрабатывать отдельные программы-редикторы для каждой комбинации операционной системы с транспортным протоколом.

Известным примером API служат гнезда (сокеты). Это временные каналы связи, установленные для передачи информации между клиентскими и серверными программами. Данные программы могут работать как на одной машине, так и на разных машинах через сеть.

Существуют три вида широко применяемых API:

гнезда Novell;

NetBIOS;

гнезда TCP/IP.

API перехватывают сетевые запросы и выполняют поставленную задачу с помощью соответствующего транспортного протокола.

Сетевые транспортные протоколы определяют метод передачи данных по сети, а также методы их пакетирования и адресации.

Передача данных между станциями в сети осуществляется с помощью кадров. Пакет – блок данных, передаваемый между станциями на сетевом уровне. Пакет является частью кадра и имеет свой заголовок. Под протоколом будем понимать системную программу, которая обрабатывает определенные поля кадра.

Транспортный протокол IPX/SPX разработан фирмой Novell. Благодаря протоколам, совместимым с IPX/SPX, их применение не ограничено сетями Novell NetWare.

Фактически протокол состоит из двух: IPX и SPX. IPX (Internet Package Exchange – межсетевой обмен пакетами) – протокол сетевого уровня, не ориентированный на установление соединения. Он отвечает за поиск наилучшего пути передачи пакетов, управляет адресацией и маршрутизацией пакетов. На

уровне IPX назначают логические сетевые адреса. IPX-адрес состоит из четырехбайтового сетевого адреса и шестибайтового адреса узла.

Поскольку IPX не поддерживает обработку ошибок, в некоторых случаях он дополняется средствами протокола SPX (Sequenced Package Exchange – последовательный обмен пакетами). Этот протокол транспортного уровня ориентирован на установление соединений. Он гарантирует установление надежного соединения перед отправлением данных по сети, отвечает за обработку искаженных пакетов и других ошибок.

Для рабочей станции с операционной системой DOS протоколы IPX и SPX входят в состав программы IPXODI.COM, которая загружается с помощью bat-файла STARTNET.BAT.

Протокол IPX обрабатывает так называемый пакет IPX, являющийся основным средством, которое используется при передаче данных в сетях NetWare. Структура пакета IPX представлена в таблице 1. Все поля, кроме последнего (Data), образуют заголовок пакета. Особенностью формата пакета является то, что все поля заголовка содержат значения в перевернутом формате: по младшему адресу записывается старший байт данных.

Длина поля	*Название поля*	*Назначение поля*
2	Checksum	Контрольная сумма
2	Length	Общая длина пакета
1	TransportControl	Счетчик пройденных маршрутизаторов
1	PacketType	Тип пакета
4	DestNetwork	Номер сети получателя пакета
6	DestNode	Адрес станции получателя
2	DestSocket	Гнездо программы получателя
4	SourceNetwork	Номер сети отправителя пакета
6	SourceNode	Адрес отправителя
2	SourceSocket	Гнездо отправителя
0-546	Data	Передаваемые данные

Таблица 1. Структура пакета IPX

Поле Checksum предназначено для хранения контрольной суммы пакета или другой служебной информации. В прикладных программах обычно не используется.

Поле Length определяет общий размер пакета вместе с заголовком. NetWare поддерживает следующие максимальные длины пакетов: Token Ring и ARCnet - 4202 байта, Ethernet - 1514 байтов. Это поле устанавливается протоколом IPX передающей станции.

Поле TransportControl служит счетчиком маршрутизаторов, которые проходит пакет на своем пути от источника до приемника. Первоначальное значение этого поля устанавливается в 0 протоколом IPX передающей станции.

Поле PacketType определяет тип передаваемого пакета. Программа, кото-

рая передает пакет средствами IPX, должна записывать в это поле значение 0x04.

Поле DestNetwork определяет номер сети, в которую передаётся пакет. Устанавливается в прикладной программе. Если в поле указывается нулевое значение, то пакет предается в сеть (сегмент), к которой подключена станция.

Поле DestNode определяет адрес станции, которой предназначен пакет. Устанавливается прикладной программой. Если пакет предназначен всем станциям в сети (сегменте), то в поле указывается значение FFFFFFFFh.

Поле DestSocket предназначено для определения программы, которая запущена на станции-получателе и должна принять пакет. Это поле устанавливается в прикладной программе.

Поля SourceNetwork, SourceNode, SourceSocket содержат соответственно номер сети, из которой посылается пакет, адрес передающей станции и гнездо программы, передающей пакет. Они заполняются протоколом IPX источника.

В поле Data содержатся передаваемые данные. Это поле формируется протоколом IPX передающей станции на основании описания блока ECB.

Блок ECB состоит из фиксированной части размером 36 байтов и массива дескрипторов, описывающих отдельные фрагменты передаваемого или принимаемого пакета данных.

Формат блока ECB, используемого в функциях интерфейса представлен в таблице 2.

Длина поля	*Название поля*	*Назначение поля*
4	Link	Указатель на следующий ECB
4	ESRAddress	Адрес программы ESR
1	InUse	Флаг состояния ECB
1	Ccode	Код завершения запроса
2	Socket	Номер гнезда для приёма или передачи
4	IPXWorkspace	Рабочий буфер для IPX
12	DriverWorkspace	Рабочий буфер
6	ImmAddress	Адрес станции сегмента, которой непосредственно передается пакет
2	FragmentCnt	Количество фрагментов в пакете
Каждая следующая пара полей образует дескриптор фрагмента		
4	Address	Адрес 1-го фрагмента
2	Size	Размер 1-го фрагмента
4	Address	Адрес 2-го фрагмента
2	Size	Размер 2-го фрагмента
.....		

Таблица 2. Поля блока ECB.

Поле Link предназначено для организации списков, состоящих из блоков ECB. Устанавливается протоколом IPX.

Поле ESRAddress содержит адрес программного модуля, который получает управление при завершении процесса чтения или передачи пакета IPX. При необходимости устанавливается прикладной программой.

Поле InUse служит индикатором завершения операции приема или передачи пакета. Вначале устанавливается в 0 прикладной программой.

Поле Ccode содержит код результата выполнения функции API-интерфейса.

Поле Socket содержит номер гнезда. Если ECB используется для приёма, то оно должно содержать номер гнезда принимающей программы. Если ECB используется для передачи, поле содержит номер гнезда передающей программы. Заполняется в прикладной программе и используется протоколом IPX для заполнения поля SourceSocket пакета IPX .

Поля IPXWorkspace и DriverWorkspace зарезервированы для использования протоколом IPX.

При передаче поле ImmAddress содержит адрес узла сегмента, куда непосредственно будет направлен пакет. Если пакет передается в пределах одного сегмента, поле содержит адрес станции-получателя (такой же, как и в поле DestNode заголовка пакета IPX). Если пакет предназначен для другого сегмента и будет проходить через маршрутизатор, поле ImmAddress содержит адрес этого маршрутизатора. Если пакет предназначен всем узлам сегмента, то в поле указывается значение FFFFFFFFh. При передаче пакета это поле заполняется в прикладной программе. Важно отметить, что значение этого поля используется драйвером сетевого адаптера для формирования адреса получателя в заголовке кадра. При приеме поле ImmAddress содержит адрес станции сегмента, от которой пришел пакет. В таком случае поле заполняется протоколом IPX. Следует отметить, что этот адрес станции выбирается из заголовка кадра (поле "Адрес отправителя") и, как правило, используется прикладной программой для передачи ответа.

Поле FragmentCnt устанавливается прикладной программой и содержит количество фрагментов, на которое надо разбить принятый пакет или из которых надо собрать передаваемый пакет, т. е. в программе можно указать отдельные буферы для приёма/передачи заголовка и данных пакета. В таком случае значение поля FragmentCnt должно быть равно 2.

Сразу за полем FragmentCnt располагаются дескрипторы фрагментов, каждый из которых состоит из адреса фрагмента (поле Address) и размера фрагмента (поле Size).

IPX имеет несколько недостатков:

- не гарантирует доставку данных,
- не гарантирует сохранение правильной последовательности приёма пакетов,
- не подавляет прием дублированных пакетов.

Таким образом, обработка ошибок, возникающих при передаче пакетов IPX, возлагается на прикладную программу, принимающую пакеты.

Задание

1. Создайте на языке программирования C++ структуру заголовка пакета

IPX.

2. Создайте структуру блока ECB.

Лабораторная работа 10 (4 часа)
Взаимодействие двух узлов в сети с использованием IPX

В таблице 3 представлена схема взаимодействие двух узлов сети с использованием протокола IPX.

Действие станции A	*Действие станции B*
Открытие гнезда (Open Socket).	Открытие гнезда (Open Socket).
Получение сетевого адреса станции B (Get network address of B).	Получение сетевого адреса станции A (Get network address of A).
Посылка пакета данных для станции B	Прием пакета данных от станции A.
Прием пакета данных от станции B.	Посылка пакета данных для станции A.
Закрытие гнезда	Закрытие гнезда.

Таблица 3. Взаимодействие узлов сети.

Для возможности использования функций протокола IPX требуется загрузить его драйвер. Проверить загрузку драйвера можно при помощи вызова функции 7ah прерывания 2fh. Если драйвер загружен, то в регистре AL помещено значение FF. При положительном результате в регистрах ES:DI передается вход в драйвер протокола IPX/SPX.

Вызов IPX можно осуществить одним из следующих способов.

1. Используя полученный адрес входа ipx_spx:

- в регистры ES:SI загрузить адрес блока ECB;
- в регистр BX поместить код выполняемой команды;
- вызвать драйвер IPX/SPX с помощью полученного выше входа.

2. Используя прерывание 7A:

- аналогично вышеописанному вызову загрузить адрес ECB и код выполняемой команды;
- вместо использования точки входа вызвать прерывания 7Ah.

Недостаток второго способа заключается в том, что и другие пользовательские программы могут использовать прерывание 7A.

Посылка и прием пакета всегда осуществляются через гнездо (сокет), которое служит идентификатором процесса на сетевом компьютере. Гнездо бывает постоянное или временное. Временное гнездо закрывается автоматически по завершению работы программы. Поэтому резидентные программы должны использовать постоянные гнезда.

Для открытия гнезда требуется:

- в регистр DX поместить номер гнезда (от 4000h до 8000h);
- в регистр BX поместить значение 0000h (команда OpenSocket);
- в регистр AL - значение 00h, если гнездо временное, значение FFh - если

гнездо постоянное;
- осуществить вызов протокола IPX/SPX.

Закрытие гнезда производится следующим образом:
- в регистр DX помещается номер открытого гнезда;
- в регистр BX помещается значение 0001h (команда CloseSocket);
- вызывается протокол IPX/SPX.

Задание

1. На языке программирования C++ создайте функцию, проверяющую загрузку драйвера протокола IPX/SPX.
2. Создайте функцию, открывающую гнездо.
3. Создайте функцию, закрывающую гнездо.
4. Пункты 2 и 3 повторите, используя другой способ доступа к драйверу протокола IPX/SPX.

Лабораторная работа 11(4 часа)
Отправление пакета в широковещательном режиме

Самый простой способ рассылки сообщений – отправление их в широковещательном режиме, т.е. сразу всем станциям в пределах данного сегмента локальной сети. В этом случае нет необходимости определять адрес получателя.

Для отправления или приема пакета, содержащего данные, необходимо произвести следующие действия:

1. Проверить загрузку драйвера IPX и получить адрес точки входа.
2. Открыть гнездо (временное или постоянное).
3. Создать структуры заголовка IPX и блока ECB.
4. Инициализировать поля блока ECB и заголовка IPX.
5. Запустить драйвер IPX

Инициализация полей блока ECB и заголовка IPX.

Для отправки пакета в ECB должны быть инициализированы следующие поля: адрес ESR, номер сокета, непосредственный адрес, количество фрагментов и список дескрипторов фрагментов.

Адрес ESR содержит дальний указатель на процедуру обработки события (ESR). Если у вас нет программы ESR, значение адреса равно 0.

Номер сокета должен быть перевернут, т.е. младший байт становится впереди старшего байта.

Непосредственный адрес содержит адрес узла, которому отправляется пакет, или адрес межсетевого моста, если взаимодействующие узлы находятся за пределами одного сетевого сегмента. Для отправки широковещательного сообщения используется адрес, содержащий 6 байтов значения $FF.

Первый буферный фрагмент в списке фрагментов ECB блока должен содержать заголовок IPX. Таким образом, ECB блок должен содержать хотя бы один фрагмент. Данные могут входить в этот фрагмент (располагаться в конце

заголовка IPX) или находиться в отдельных фрагментах.

Пример 1.

```
struct     //список буферных фрагментов в блоке ECB
 { void far * address
   unsigned int length
 } packet [2]
…
ECB.Packet[0].Address=&IPXHead;          //инициализация первого буфера
ECB.Packet[0].Length=sizeof(IPXHead);
```

В заголовке IPX должны быть инициализированы следующие поля: тип пакета, сеть назначения, узел назначения, гнездо назначения.

Тип пакета протокола IPX имеет значение 4.

Если станция назначения находится в той же сети, что и станция, отправляющая пакет, то значение сети назначения можно установить в нуль.

Для отправки пакета всем узлам сети одновременно все шесть байтов адреса станции назначения устанавливаются в значение FF.

Номер гнезда приемника назначается программистом.

Запуск драйвера IPX

После выполнения пунктов 1 – 4 при запуске драйвера протокола IPX требуется сообщить ему код операции и адрес ECB. Для этого в регистр BX помещается значение 0x0003, соответствующее команде отправки сообщения. В регистры ES:SI помещается дальний указатель на блок ECB. Далее идет вызов точки входа IPX или прерывания 7A.

После этого функция драйвера завершит формирование IPX-заголовка, установив значения для полей: контрольная сумма, длина, число пройденных мостов, сеть источника, узел источника и сокет источника. Значение последнего из названых полей берется из поля «номер сокета» поставленного в ECB-блоке. Функция подготавливает блок ECB и IPX-заголовок связанного с ним пакета и передает ECB сетевому коммуникационному драйверу для инициализации операции отправки пакета. Осуществив эту операцию, функция возвращает управление вызвавшему приложению.

Пока пакет ожидает своей отправки, поле “In Use” собственного заголовка ECB содержит значение FFh.

После того, как сетевые драйверы осуществили попытку отправить пакет, значение поля “In Use” сбрасывается в нуль. Затем поле “Код завершения” в ECB будет содержать конечный статус запроса на отправку пакета, и вызывается программа ESR, адрес которой указан в поле “Адрес ESR”, (если она используется). При нежелательности обработки события поле “Адрес ESR” обнулить.

Если запрос отменен посредством функций “Отменить событие” или “Закрыть сокет”, поле “In Use” станет нуль и “Код завершения” установится в значение FCh. В данном случае ESR, связанная с ECB, не будет вызвана.

Значение	Действие

00h	Отправка пакета осуществлена успешно.
FEh	Пакет не доставлен.
FDh	Неверно сформированный пакет (длина пакета менее 30 байтов, либо превышает 576 байтов, или список фрагментов слишком мал для заголовка, или “Количество фрагментов” равно нулю).
FCh	Запрос на отправку пакета был отменен вызовом функций “Закрыть сокет” или “Отменить событие”.
FFh	Невозможность физически отправить пакет (неисправность сетевого оборудования).

Таблица 3. Допустимые значения поля «кода завершения запроса» при отправке пакета.

Задание

1. Создайте функцию отправки пакета в широковещательном режиме с анализом кода возврата.

Лабораторная работа 12 (2 часа)

Прием пакета

При организации приема пакета так же выполняются пункты 1-3 рассмотренные выше, а пункты 4 и 5 отличаются от предыдущего описания своим содержанием.

Инициализация полей блока ECB и заголовка IPX

Для приема пакета в блоке ECB должны быть инициализированы следующие поля: адрес ESR, номер сокета, количество фрагментов и список дескрипторов фрагментов. Все поля инициализируются так же, как при отправке пакета. Структура фрагментов в принимающем заголовке ECB должна совпадать со структурой фрагментов в отправляющем ECB.

Запуск драйвера IPX

При приеме сообщения, аналогично отправке, при запуске драйвера протокола IPX требуется сообщить ему код операции и адрес блока ECB. В регистр BX помещается соответственно значение 0x0004 – код операции приема пакета. В регистры ES:SI помещается дальний указатель на блок ECB. Далее идет вызов точки входа IPX или прерывания 7A.

Функция не ожидает фактического получения пакета и передает управление программе, когда пакет еще не принят.

Когда ECB передается пакет IPX с запросом на «прослушивание» межсетевых пакетов, поле “In Use” устанавливается IPX-ом в значение FEh. Значение этого поля останется установленным до тех пор, пока ожидаемое событие - прием пакета - не произошло, или запрос не отменен, или IPX признал ECB не пригодным для использования.

После того, как IPX сбросил значение поля “In Use” в нуль, одно из перечисленных ниже значений могут быть переданы в поле “Код завершения” данного ECB:

Значение	Действие
00h	Прием пакета осуществлена успешно.
FFh	Сокет, посредством, которого ECB «прослушивал», не был открыт.
FDh	Пакет переполнен. Пакет был принят, но поле «Количество фрагментов» содержит нуль или доступное адресное пространство, определенное спи ском дескрипторов фрагментов не достаточно большого размера, чтобы поместился весь пакет.
FCh	Запрос на отправку пакета был отменен вызовом функций “Закрыть сокет” или “Отменить событие”.

Таблица 4. Допустимые значения поля «кода завершения запроса»при приеме пакета.

Если запрос был отменен функцией “Отменить событие” или “Закрыть сокет”, поле “In Use” будет сброшено в нуль и поле “Код завершения” будет установлен в значение FCh, как было описано выше. Процедура ESR для блока ECB вызвана не будет.

Задание

1. Создайте процедуру приема пакета с анализом кода возврата.
2. Проверьте прием с помощью отправки широковещательного пакета.

Лабораторная работа 13 (8 часов)
Протокол TCP/IP

Протокол TCP/IP предназначен для соединения сетей с разнородным оборудованием. Это единственное средство коммуникации, позволяющее связываться рабочим станциям всех типов – PC, Macintosh, Unix. Он необходим для выхода в Internet. Фактически TCP/IP является набором протоколов. Наиболее известные из них TCP и IP.

Протокол IP (Internet Protocol) функционирует на сетевом уровне, предоставляя различным станциям стандартный набор правил и спецификаций для межсетевой маршрутизации с помощью IP адресов.

Протокол управления передачей данных TCP (Transmission Control Protocol) работает на транспортном уровне модели OSI, обеспечивая прием сетевой и преобразование информации в требуемый на данном уровне формат. Таким образом, IP задает правила установления соединения и обеспечивает соедине-

ние портов компьютера, TCP отвечает за интерпретацию данных.

IP определяет пакеты, называемые *датаграммами* (datagrams). Эти пакеты, обычно имеют длину менее 1000 байт. IP-датаграмма содержит 32-разрядные адреса компьютеров: источника и получателя. IP-адреса идентифицируют компьютеры в сети (в том числе и в Internet). Они используются *маршрутизаторами* для передачи получателям отдельных датаграмм. Маршрутизаторы работают только с адресом получателя и длинной пакета, а их задача заключается в выборе оптимального маршрута передачи данных.

Часть IP-адреса идентифицирует локальную сеть, в которой находится *компьютер*, а другая часть непосредственно компьютер внутри сети. Большинство IP-адресов являются адресами Класса С, формат которого показан на рисунке 1.

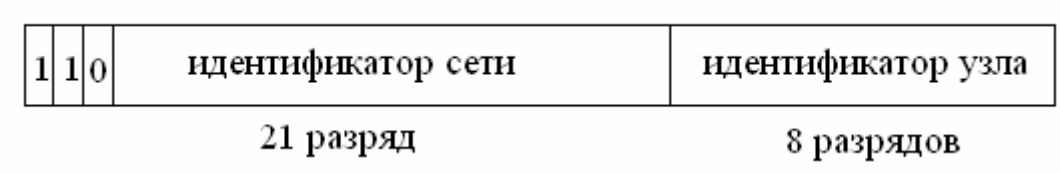


Рисунок 1. Формат IP-адреса класса С.

По соглашению IP-адреса пишутся в десятичном формате с разделением точками. Четыре части адреса соответствуют отдельным байтам. Например, IP-адрес класса С может быть такой: 194.128.198.201. На компьютерах с процессором Intel байты адреса хранятся в формате «младший слева» (little-endian). На большинстве других компьютеров, в том числе под управлением UNIX, байты хранятся в порядке «старший слева» (big-endian). Поскольку в Интернет необходим машинно-независимый стандарт для обмена данными, все многобайтовые значения должны передаваться в порядке «старший слева». Это означает, что программы на Intel-компьютерах должны выполнять преобразование из *сетевого порядка байтов* («старший слева») в *порядок байтов хоста* («младший слева») и обратно. Это относится как к 2-байтовым номерам портов, так и к 4-байтовым IP-адресам.

Уровень IP не сообщает отправителю, достигла ли датаграмма получателя. Это задача стоящего над ним уровня в стеке. Получатель может только проверить контрольную сумму для определения целостности заголовка IP-датаграммы.

На одном уровне с TCP находится протокол UDP (User Datagram Protocol), входящий в состав TCP/IP.

UDP — это всего лишь небольшая надстройка над IP, поскольку приложения никогда не используют IP напрямую.

<table>
<tr><td colspan="3">Датаграмма IP</td></tr>
<tr><td></td><td colspan="2">Датаграмма UDP</td></tr>
<tr><td>Заголовок IP (20 байтов)</td><td>Заголовок UDP (8 байтов)</td><td>Данные UDP</td></tr>
</table>

Рисунок 2 Датаграмма UDP внутри датаграммы IP.

Подобно IP, UDP не сообщает отправителю о доставке датаграммы. Решение оставлено на усмотрение приложения. Например, отправитель может потребовать, чтобы получатель прислал ответ, и может заново отправить датаграмму, если ответ не пришел в течение, скажем, 20 с. UDP хорош для простых однократных сообщений — он используется в системе доменных имен (Domain Name System, DNS1). UDP также применяется для передачи звука и изображения в реальном времени, в которых потеря или неверная последовательность данных не проблематичны. Формат UDP, показанный на рисунке 3.

16 – разрядный номер порта источника	16 – разрядный номер порта получателя
16 – разрядная длина (заголовок UDP + данные)	16 – разрядная контрольная сумма (заголовок UDP + данные)
Данные (если они есть)	

Рисунок 3. Упрощенный формат UDP

В заголовке UDP передается дополнительная информация: *номера портов* источника и получателя. Эти номера используются приложениями на обоих концах канала связи.

UDP — это всего лишь небольшая надстройка над IP, поскольку приложения никогда не используют IP напрямую. Подобно IP, UDP не сообщает отправителю о доставке датаграммы. Решение оставлено на усмотрение приложения. Например, отправитель может потребовать, чтобы получатель прислал ответ, и может заново отправить датаграмму, если ответ не пришел в течение, скажем, 20 с. UDP хорош для простых однократных сообщений — он используется в *системе доменных имен* (Domain Name System, DNS1). UDP также применяется для передачи в реальном времени звука и изображения, для которых потеря или непоследовательность данных не проблематичны.

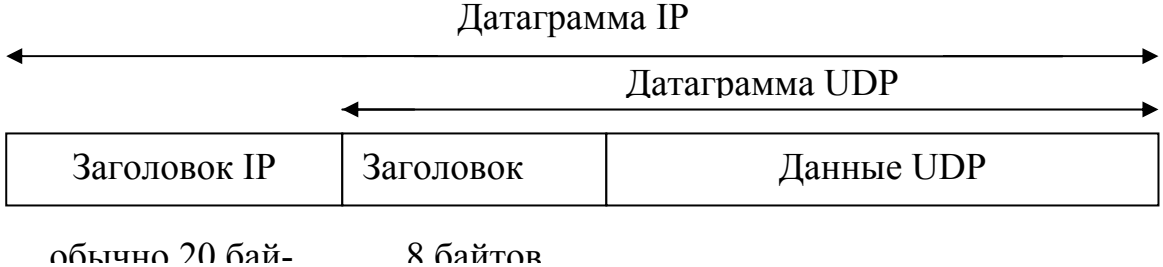


Рисунок 4. Датаграмма UDP внутри датаграммы IP

Все транспортные протоколы на основе IP сохраняют свои заголовки и

данные внутри IP-блока (рисунок 4).

Протокол TCP выполняет безошибочную передачу больших блоков данных. Причем программа-получатель принимает байты в той же последовательности, в какой они были переданы, пусть даже при этом отдельные датаграммы приходили бы в неправильной последовательности. TCP — главный протокол всех приложений Интернета, включая HTTP и FTP. На рисунке 5 показан формат *сегмента* TCP.

<table>
<tr><td colspan="2">16-разрядный номер порта источника</td><td colspan="2">16-разрядный номер порта получателя</td></tr>
<tr><td colspan="4">32-разрядный номер последовательности</td></tr>
<tr><td colspan="4">32-разрядный номер подтверждение</td></tr>
<tr><td>4 разрядная длина заголовка</td><td></td><td>Флажки (SYN, ASK, FIN)</td><td></td></tr>
<tr><td colspan="2">16 разрядная контрольная сумма (заголовок TCP + данные)</td><td colspan="2"></td></tr>
<tr><td colspan="4">Параметры (если есть)</td></tr>
<tr><td colspan="4">Данные (если есть)</td></tr>
</table>

Рисунок 5. Упрощенный формат сегмента TCP

TCP-сегмент располагается внутри IP-датаграммы, как показано на рисунке 5.

Протокол TCP устанавливает между двумя компьютерами *дуплексное соединение* типа «точка-точка». Программы на каждом конце соединения используют собственный порт. Комбинация IP-адреса и номера порта называется *сокет* (socket). Соединение устанавливается путем *трехкратного квитирования* (three-way handshake). Инициирующая программа посылает сегмент с установленным флажком *SYN,* отвечающая программа посылает сегмент с установленными флажками *SYN и ACK* и, наконец, инициирующая программа посылает сегмент с установленным флажком *ACK.*

После установления соединения каждая программа может посылать другой программе поток байтов. Для управления потоком TCP использует поля номеров последовательностей и флажки *ACK.* Программа-отправитель не ожидает подтверждения каждого сегмента, а посылает несколько сегментов вместе и ждет первого подтверждения. Если программа-получатель должна отослать данные обратно отправителю, она может совместить подтверждения и данные в одних и тех же сегментах. Номера последовательности программы-отправителя являются не индексами сегментов, а индексами в потоке байтов. Программа-получатель отсылает обратно номера последовательности (в поле номера подтверждения), удостоверяя тем самым, что все байты приняты и объ-

единены в правильной последовательности. Программа-отправитель заново пересылает неподтвержденные сегменты.

Каждая программа со своей стороны закрывает TCP-соединение, отправляя сегмент с флажком *FIN,* что должно быть подтверждено программой на другой стороне соединения. Программа не может получать байты по соединению, которое закрыто на другой стороне.

КОМПОНЕНТЫ WINSOCK

Winsock — это Windows API нижнего уровня для программирования TCP/IP. Часть кода находится в wsock32.dll (в том числе экспортируемые функции программы пользователя), а часть — в ядре Windows. Новая и более сложная версия Winsock 2 включена в состав Windows NT 4.0, но здесь рассмотрена предыдущая версия, поскольку она является стандартом как для Windows NT, так и для Windows 95/98.

Функция ***Close*** закрывает ранее открытый сокет, вызывая Winsock-функцию *closesocket.* Предварительно должна быть вызвана функция *Create.* Деструктор не вызывает эту функцию, потому что он не сможет *перехватить* (catch) исключение для глобального объекта. Для сокета, ожидающего запросы, сервер может вызвать *Close* в любой момент.

Bind используется для привязки ранее созданного сокета к конкретному адресу. До вызова *Listen* сервер вызывает *Bind* с указанием адреса сокета, содержащего номер порта ожидания и IP-адрес сервера. Если в IP-адресе указать *INADDRANY,* то Winsock использует IP-адрес вашего компьютера. Ее параметр *psa - о*бъект *CSockAddr* или указатель на переменную типа *sockaddr.*

Listen вызывает Winsock-функцию listen. Сервер вызывает Listen в начале ожидания запросов на порту, заданном предшествующим вызовом Bind. Функция завершается сразу.

Accept вызывает Winsock-функцию accept. Сервер вызывает Accept сразу после вызова Listen. Функция завершается, когда клиент подсоединяется к сокету, возвращая новый сокет (через переданный в нее объект CBlockingSockef), соответствующий новому соединению. Параметры: s – ссылка на существующий объект CBlockingSocket, для которого не была вызвана функция Create; psa - объект CSockAddr или указатель на переменную типа sockaddr для адреса подсоединенного сокета. Возвращаемое значение TRUE, если все хорошо.

Connect вызывает Winsock-функцию connect. Клиент вызывает Connect после вызова Create.Возврат из Connect происходит после установления соединения. Параметр – psa объект CSockAddr или указатель на переменную типа sockaddr.

Send вызывает Winsock-функцию send после установки тайм-аута при помощи select. Количество реально переданных байтов при вызове Send зависит от того, насколько быстро программа на другом конце соединения может получить

байты. Send генерирует исключение, если сокет на другом конце соединения был закрыт до того, как были прочитаны все байты. Параметры: pcb – указатель на буфер, содержащий передаваемые данные; nSize – размер (в байтах) блока данных; nSecs – значение тайм-аута (в секундах). Возвращаемое значение – количество переданных данных.

Write постоянно вызывает Send до тех пор, пока не будут переданы все байты или пока получатель не закроет сокет. Параметры: pcb – указатель на буфер, содержащий передаваемые данные; nSize – размер (в байтах) блока данных; nSecs - значение тайм-аута (в секундах). Возвращаемое значение – количество переданных данных.

Receive вызывает Winsock-функцию recv после установки тайм-аута при помощи select.Возвращает полученные байты. Параметры: pcb – указатель на буфер, сохраняющий полученные данные; nSize, nSecs (см. описание выше). Возвращаемое значение – количество полученных данных.

SendDatagram вызывает Winsock-функцию sendto. Программа на другом конце соединения должна вызвать ReceiveDatagram. Для датаграмм не нужно вызывать Listen, Accept или Connect, но надо предварительно вызвать Create с параметром SOCKJDGRAM. Параметры: pcb – указатель на буфер, содержащий передаваемые данные; nSizeб,psa,nSecs (см. описание выше). Возвращаемое значение – количество переданных данных.

ReceiveDatagram вызывает Winsock-функцию *recvfrom*. Возврат из функции происходит, когда программа на другом конце соединения вызывает *SendDatagram*. Предварительно надо вызвать *Create* с параметром *SOCKJDGRAM*. Параметры: уже ранее рассмотренные *pcb*, *nSize*, *psa*, *nSecs*. Возвращаемое значение - количество полученных данных.

GetPeerAddr вызывает Winsock-функцию *getpeername,* возвращает порт и IP-адрес сокета на другом конце соединения. Если Вы подсоединены к Интернету через прокси-сервер, то получите его IP-адрес. Параметр: *psa*.

GetSockAddr вызывает Winsock-функцию *getsockname,* возвращает адрес сокета на этом конце соединения. Если программа на другом конце — сервер локальной сети, то IP-адрес — это адрес, присвоенный сетевой плате компьютера. Если же на другом конце работает Интернет-сервер, то IF -адрес будет присвоен *службой доступа* (service provider) при дозвоне. В обоих случаях Winsock присваивает номер порта, свой для каждого соединения. Параметр: *psa*.

GetHostByName (статическая) вызывает Winsock-функцию *getbostbyname,* запрашивает сервер имен и возвращает адрес сокета, соответствующий имени хоста. Функция сама себе устанавливает тайм-аут. Параметры: *pcbName* – указатель на массив символов, содержащий имя хоста; *ushPort* – номер порта (по умолчанию 0), который станет частью возвращенного адреса сокета. Возвращаемое значение – адрес сокета, содержащий IP-адрес от DNS и номер порта *ushPort*.

GetHostByAddr (статическая) вызывает Winsock-функцию *gethostbyaddr,* запрашивает сервер имен и возвращает имя хоста, соответствующее адресу сокета. Функция сама себе устанавливает тайм-аут. Параметр: p*sa*. Возвращаемое значение – адрес сокета, содержащий IP-адрес от DNS и номер порта *ushPort*.

Классы блокирующих сокетов

Так как MFC невозможно напрямую использовать, можно создать собственные классы Winsock. Класс *CBlockingSocket* — это тонкая оболочка вокруг Winsock API, предназначенная только для синхронных вызовов в рабочем потоке. Единственными исключениями являются генерация исключений в случае ошибок и тайм-ауты при передаче и приеме данных. Они помогают писать более ясный код, поскольку не надо проверять ошибки после каждого вызова Winsock. Тайм-ауты, реализованные через вызов функции *select,* предотвращают бесконечное блокирование потока кода в случае сбоя связи.

Класс *CHttpBlockingSocket,* производный от *CBlockingSocket,* предоставляет функции для чтения HTTP-данных. Классы *CSockAddr* и *CBlockingSocketException* — вспомогательные.

Вспомогательный класс *CSockAddr*

Многим функциям Winsock в качестве параметра нужен адрес сокета. Адрес сокета состоит из 32-разрядного IP-адреса и 16-разрядного номера порта. В Winsock для задания адреса используется 16-байтовая структура *sockaddr_in-*.

```
struct sockaddr_in {
   short sin_family;
   u_short sin_port;
   struct in_addr sin_addr;
   char sin_zero[8]; };
```

IP-адрес хранится как значение типа *in_addr,* определенное следующим образом:

```
struct in_addr {
uion {
struct { u_char s_b1,s_b2,s_b3,s_b4; }
S_un_b; struct { u_short s_w1,s_w2; }
S_un_w; u_long S_addr; } S_un; >
```

Производный от *sockaddrin* класс может быть объявлен в виде:

```
class CSockAddr : public
sockaddr_in { public:
  // конструкторы
  CSockAddrO
  {
     sin_family = AF_INET;
     sin_port = 0;
```

```
      sin_addr.s_addr = 0; }
      // стандартные
   CSockAddr(const SOCKADDR& sa)
   { memcpy (this, &sa, sizeof(SOCKADDR)); }
   CSockAddr(const SOCKADDR_IN& sin)
   { memcpy(this, Ssin, sizeof(SOCKADDR_IN));
    >CSockAddr(const ULONG ulAddr, const USHORT ushPort = 0)
   //параметры должны иметь порядок байтов хоста
   {
      sin_family = AF_INET;
      sin_port = htons(ushPort);
      sin_addr.s_addr = htonl(ulAddr);

   CSockAddr(const char* pchIP,  const USHORT ushort = 0)

   // строка IP-адреса с разделением точками

        sin_family = AF_INET;
      sin_port = htons(ushPort);
      sin_addr.s_addr ■ inet_addr(pchIP);
   } // уже в сетевом порядке байтов
   // Возвращает адрес в десятичном формате с разделением точками
   CString DottedDecimalQ { return inet_ntoa(sin_addr); }
   // создает новый объект CString
   // Получение порта и адреса (даже если они открытые)
      USHORT Port() const { return ntohs(sin_port); }
       ULONG IPAddrO const { return ntohl(sin_addr.s_addr); }
  // операторы добавлены для эффективности
      const CSockAddr& operator=(const SOCKADDR& sa) {
      memcpy (this, &sa, sizeof(SOCKADDR)); return *this; }
    const CSockAddr& operator=(const SOCKADDR_IN& sin) {
      memcpy(this, &sin, sizeof(SOCKADDR_IN));
      return *this; } op-
   erator SOCKADDRO
      { return *((LPSOCKADDR) this); } operator LPSOCKADDR( )
      { return (LPSOCKADOR)
   this;}
       operator LPSOCKADOR_IN()
      { return (LPSOCKADDR_IN) this; }
};
```

Задание

1. Создать программу пересылки сообщений с использованием протокола TCP/IP в широковещательном режиме.

2. Модифицировать программу для передачи сообщений между двумя узлами сети.

Лабораторная работа 14 (2 часа)

Основы проектирования локальной сети

Несмотря на появление новых технологий, классические протоколы ЛВС по прогнозам специалистов будут использоваться еще, по крайней мере, лет 5-10. Поэтому знание их деталей необходимо для успешного применения современной коммуникационной аппаратуры. Кроме того, современные высокопроизводительные технологии, такие как Fast Ethernet, Gigabit Ethernet, в значительной степени сохраняются преемственность своих предшественников /5, с.183 /.

Ethernet – самая распространенная технология локальных сетей. Исторически первые сети технологии Ethernet были созданы на коаксиальном кабеле. В дальнейшем определены и другие спецификации физического уровня, позволяющие использовать различные методы среды передачи (см. таблицу 5).

Ethernet	Thick Wire Ethernet	Thin Wire Ethernet	UTP Ethernet	Fiber Optic Ethernet
IEEE 802.3	10BASE-5	10BASE-2	10BASE-T	10BASE-F
Длина сегмента кабеля, м	500	185	100	2000
Максимальное расстояние между узлами сети (при использовании повторителей), м	2500	925	500	2500 (2740 для 10Base-FB*)
Максимальное число станций на сегменте	100	30	1024	1024
Тип кабеля	коаксиальный кабель «толстый» RG-8 или RG-11	коаксиальный кабель «тонкий» RG-58	Неэкранированная витая пара категории 3, 4, 5	Многомодовый волоконно-оптический кабель

* стандарт 10Base- FB предназначен только для соединения повторителей

Таблица 5. Спецификации сети Ethernet

Число 10 в указанных названиях означают битовую скорость передачи данных, а слово Base – метод передачи на одной базовой частоте 10 МГц (в отличие от методов несущих несколько частот Broadband. Последний символ в названии означает тип кабеля.

Общими ограничениями для всех стандартов Ethernet являются: максимальное число повторителей между любыми станциями сети – 4 (5 для 10Base- FB); максимальное расстояние между узлами сети – 2500 м, максимальное число коаксиальных сегментов – 5.

МЕТОДИКА РАСЧЕТА КОНФИГУРАЦИИ СЕТИ ETHERNET

Гарантию корректной работы сети дает соблюдение многочисленных ограничений, установленных для различных стандартов физического уровня сетей Ethernet.

Наиболее часто приходится проверять ограничения, связанные с длиной отдельного сегмента кабеля, а также количеством повторителей и общей длиной сети. Правила «5-4-3» для коаксиальных сетей и «4-х хабов» для сетей на основе витой пары и оптоволокна не только дают гарантии работоспособности сети, но и оставляют большой «запас прочности» сети. Например, если посчитать время двойного оборота в сети, состоящей из 4-х повторителей 10Base-5 и 5-ти сегментов максимальный длины 500 м, то окажется, что оно составляет 537 битовых интервала. А так как время передачи кадра минимальной длины, состоящего вместе с преамбулой 72 байт, равно 575 битовым интервалам, то видно, что разработчики стандарта Ethernet оставили 38 битовых интервала в качестве запаса для надежности. Тем не менее, комитет 802.3 говорит, что и 4 дополнительных битовых интервала создают достаточный запас надежности.

Комитет IEEE 802.3 приводит исходные данные о задержках, вносимых повторителями и различными средами передачи данных, для тех специалистов, которые хотят самостоятельно рассчитывать максимальное количество повторителей и максимальную общую длину сети, не довольствуясь теми значениями, которые приведены в правилах «5-4-3» и «4-х хабов». Особенно такие расчеты полезны для сетей, состоящих из смешанных кабельных систем, например коаксиала и оптоволокна, на которые правила о количестве повторителей не рассчитаны. При этом максимальная длина каждого отдельного физического сегмента должна строго соответствовать стандарту, то есть 500 м для «толстого» коаксиала, 100 м для витой пары и т. д

Чтобы сеть Ethernet, состоящая из сегментов различной физической природы работала корректно, необходимо выполнение четырех основных условий:

- количество станций в сети не более 1024;
- максимальная длина каждого физического сегмента не более величины, определенной в соответствующем стандарте физического уровня;
- время двойного оборота сигнала (Path Delay Value, PDV) между двумя самыми удаленными друг от друга станциями сети не более 575 битовых интервала;
- сокращение межкадрового интервала IPG (Path Variability Value, PVV) при прохождении последовательности кадров через все повторители должно быть не больше, чем 49 битовых интервала. Так как при отправке кадров конечные узлы обеспечивают начальное межкадровое расстояние в 96 битовых интервала, то после прохождения повторителя оно должно быть не меньше, чем 96 - 49 = 47 битовых интервала /5/.

Соблюдение этих требований обеспечивает корректность работы сети даже в случаях, когда нарушаются простые правила конфигурирования, определяющие максимальное количество повторителей и общую длину сети в 2500 м.

Расчет PDV

Для упрощения расчетов обычно используются справочные данные IEEE, содержащие значения задержек распространения сигналов в повторителях, приемопередатчиках и различных физических средах. В таблице 6 приведены данные, необходимые для расчета значения PDV для всех физических стандартов сетей Ethernet. Битовый интервал обозначен как bt.

Тип сегмента	База левого сегмента, bt	База промежуточного сегмента, bt	База правого сегмента, bt	Задержка среды на 1 м, bt	Максимальная длина сегмента, м
10Base-5	11,8	46,5	169,5	0,0866	500
10Base-2	11,8	46,5	169,5	0,1026	185
10Base-T	15,3	42,0	165,0	0,113	100
10Base-FB	—	24,0	-	0,1	2000
lOBase-FL	12,3	33,5	156,5	0,1	2000
AUI (> 2 м)	0	0	0	0,1026	2+48

Таблица 6. Данные для расчета значения PDV

Комитет 802.3 максимально упростил выполнение расчетов. Данные, приведенные в таблице, включают сразу несколько этапов прохождения сигнала, такие как задержки, вносимые повторителем, состоят из задержки входного трансивера, задержки блока повторения и задержки выходного трансивера. В таблице все эти задержки представлены одной величиной, названной базой сегмента. Складывать задержки дважды, вносимые кабелем, не нужно – в таблице даются удвоенные величины задержек для каждого типа кабеля.

Использование табличных понятий левый сегмент, правый сегмент и промежуточный сегмент рассмотрено на примере сети, приведенной на рисунке 6.

Левым сегментом называется сегмент, в котором начинается путь сигнала от выхода передатчика конечного узла. На рисунке это сегмент 1. Затем сигнал проходит через промежуточные сегменты 2 – 5 и приходит до наиболее удаленного узла сегмента 6, который называется правым. Именно здесь в худшем случае происходит столкновение кадров и возникает коллизия, что и подразумевается в таблице.

С каждым сегментом связана постоянная задержка, названная базой, которая зависит от типа сегмента и от положения сегмента на пути сигнала (левый, промежуточный или правый). База правого сегмента, в котором возникает коллизия, намного превышает базу левого и промежуточных сегментов.

С каждым сегментом также связана задержка распространения сигнала вдоль кабеля сегмента, зависящая от длины сегмента и вычисляемая путем умножения времени распространения сигнала по одному метру кабеля (в битовых интервалах) на длину кабеля в метрах.

Расчет заключается в вычислении задержек, вносимых каждым отрезком кабеля (приведенная в таблице задержка сигнала на 1 м кабеля умножается на длину сегмента), а затем суммировании этих задержек с базами левого, промежуточных и правого сегментов. Общее значение PDV не должно превышать

575.

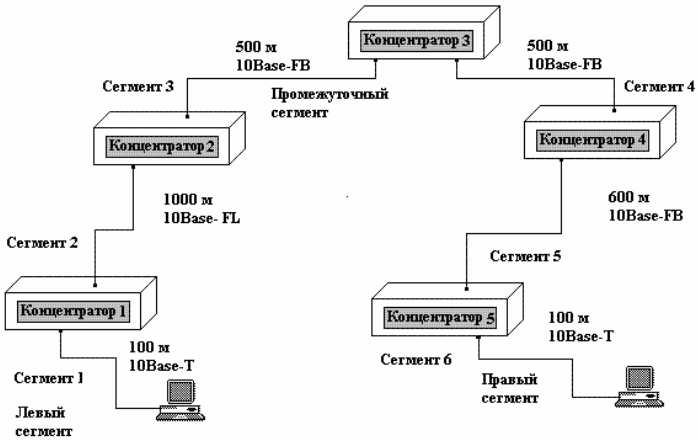


Рисунок 6**.** Пример сети Ethernet

Так как левый и правый сегменты имеют различные величины базовой задержки, то в случае различных типов сегментов на удаленных краях сети необходимо выполнить расчеты дважды: один раз принять в качестве левого сегмента одного типа, а во второй — сегмент другого типа. Результатом можно считать максимальное значение PDV. В примере крайние сегменты сети принадлежат к одному типу — стандарту 10Base-T, следовательно, двойной расчет не требуется. Если же сегменты были разного типа, то в первом случае нужно было бы принять в качестве левого сегмент между станцией и концентратором 1, а во втором считать левым сегмент между станцией и концентратором 5.

Приведенная на рисунке сеть в соответствии с правилом 4-х хабов не является корректной — в сети между узлами сегментов 1 и 6 имеется 5 хабов, хотя не все сегменты являются сегментами 10Base-FB. Кроме того, общая длина сети равна 2800 м, что нарушает правило 2500 м.

Расчет значения PDV для рассматриваемого примера:
Левый сегмент 1: 15,3 (база) + 100 x 0,113 = 26,6.
Промежуточный сегмент 2: 33,5 + 1000 x 0,1 = 133,5.
Промежуточный сегмент 3: 24 + 500 x 0,1 - 74,0.
Промежуточный сегмент 4: 24 + 500 x 0,1 - 74,0.
Промежуточный сегмент 5: 24 + 600 x 0,1 - 84,0.
Правый сегмент 6: 165 + 100 x 0,113 = 176,3.
Сумма всех составляющих дает значение PDV, равное 568,4.

Полученное значение PDV меньше максимально допустимой величины 575, и, следовательно, сеть проходит по критерию времени двойного оборота сигнала, несмотря на то, что ее общая длина составляет больше 2500 м, а количество повторителей — больше 4-х.

Расчет PVV

Чтобы признать конфигурацию сети корректной, нужно рассчитать также уменьшение межкадрового интервала повторителями, то есть величину PVV.

Для расчета PVV также можно воспользоваться значениями максималь-

ных величин уменьшения межкадрового интервала при прохождении повторителей различных физических сред, рекомендованными IEEE и приведенными в таблице 7.

Тип сегмента	Передающий сегмент, bt	Промежуточный сегмент, bt
10Base-5 или 10Base-2	16	11
10Base-FB	—	2
10Base-FL	10,5	8
10Base-T	10,5	8

Таблица 7. Сокращение межкадрового интервала повторителями

В соответствии с этими данными рассчитаем значение PVV для примера.
Левый сегмент 110Base-T: сокращение в 10,5 bt.
Промежуточный сегмент 2 10Base-FL: 8.
Промежуточный сегмент 3 10Base-FB: 2.
Промежуточный сегмент 4 10Base-FB: 2.
Промежуточный сегмент 5 10Base-FB: 2.

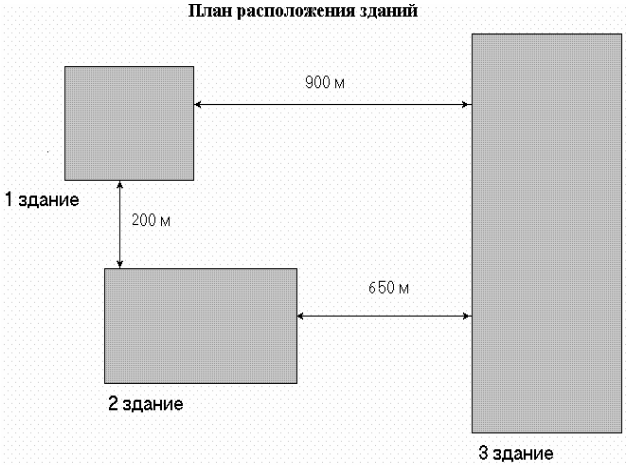
Сумма этих величин дает значение PVV, равное 24,5, что меньше предельного значения в 49 битовых интервала. В результате приведенная в примере сеть соответствует стандартам Ethernet по всем параметрам, связанным и с длинами сегментов, и с количеством повторителей.

Задание

На рисунке в вариантах задания представлен план расположения компьютеров в зданиях. Требуется предложить проект ЛВС по следующему плану:

1. Выбрать топологию ЛВС. Обосновать выбор.
2. Выбрать оптимальную конфигурацию сети Ethernet.
3. Нарисовать функциональную схему ЛВС и составить перечень аппаратных средств.
4. Произвести ориентировочную трассировку кабельной системы и выполнить расчет длины кабельного соединения для выбранной топологии с учетом переходов между этажами.
5. С целью выполнения ограничений на длину кабеля и количества станций в сегменте выбранной конфигурации установить необходимые коммуникационные устройства.
6. Рассчитать стоимость ЛВС.
7. Оценить проектируемую сеть по критерию времени двойного оборота (рассчитать PDV) и характеристики уменьшения межкадрового интервала повторителями (PVV).

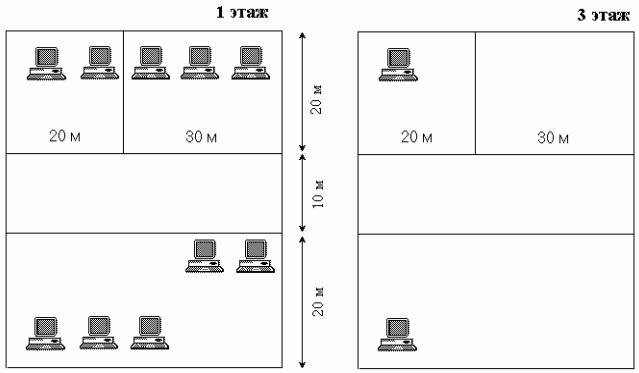
Вариант 1



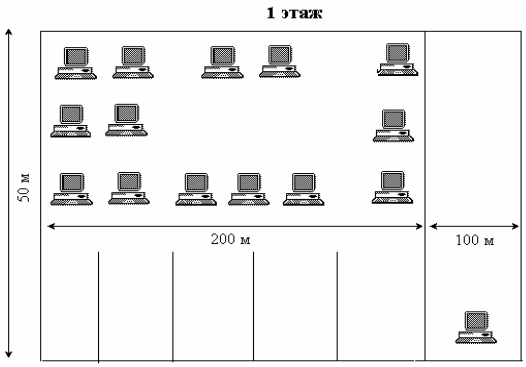
Первое из зданий трехэтажное. Компьютеры расположены в отделах на первом и третьем этажах. Общее число компьютеров – 12 (10 на первом этаже и 2 на втором).

Второе и третье здания одноэтажные. В каждом из них – 15 компьютеров.

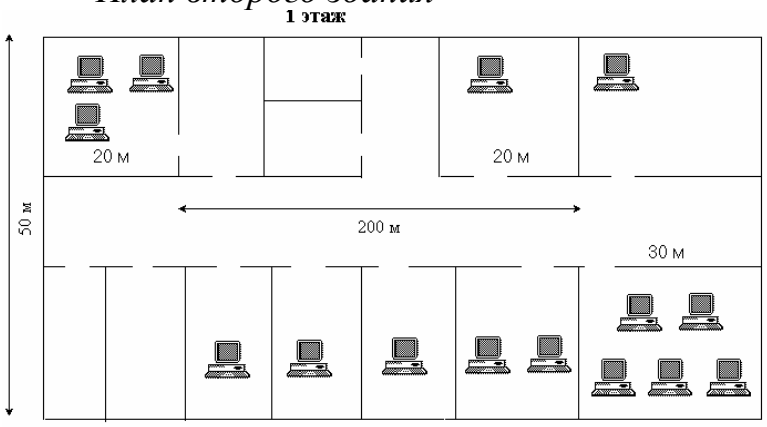
План первого здания



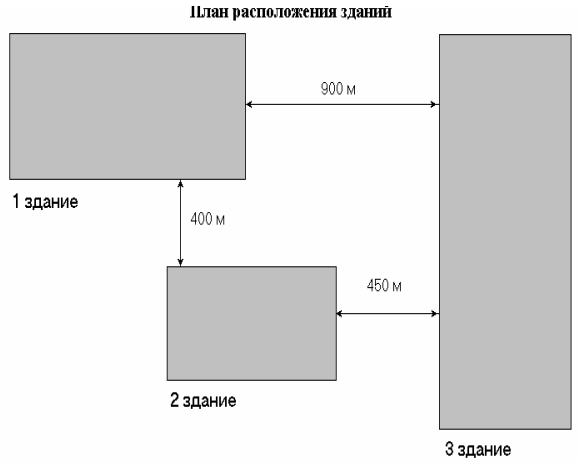
План третьего здания



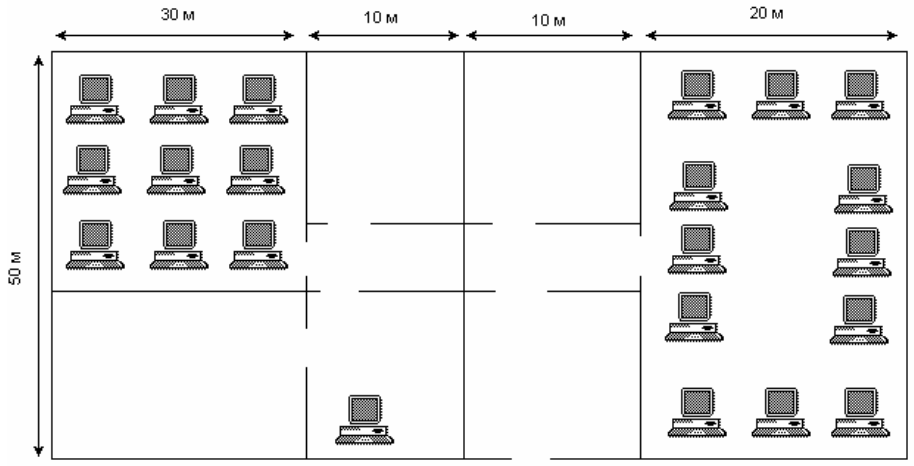
План второго здания



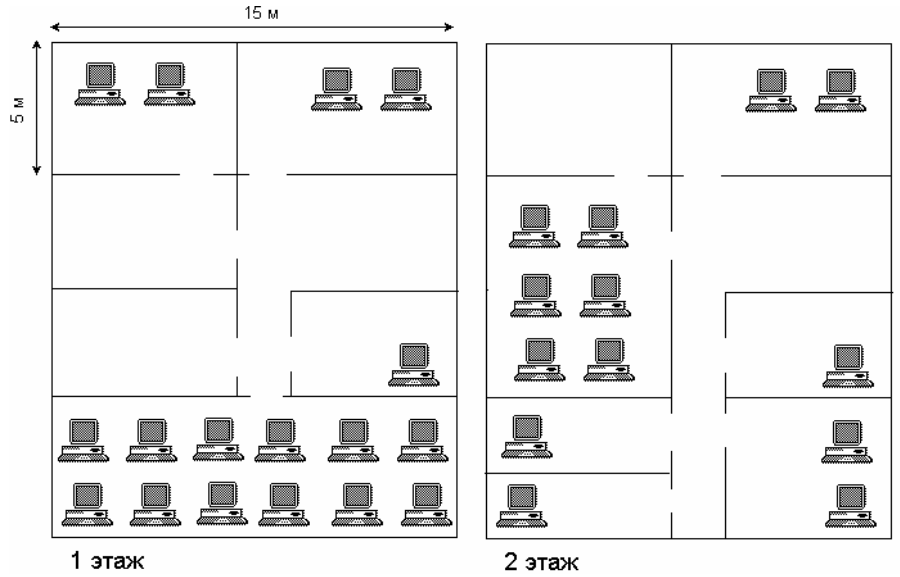
Вариант 2



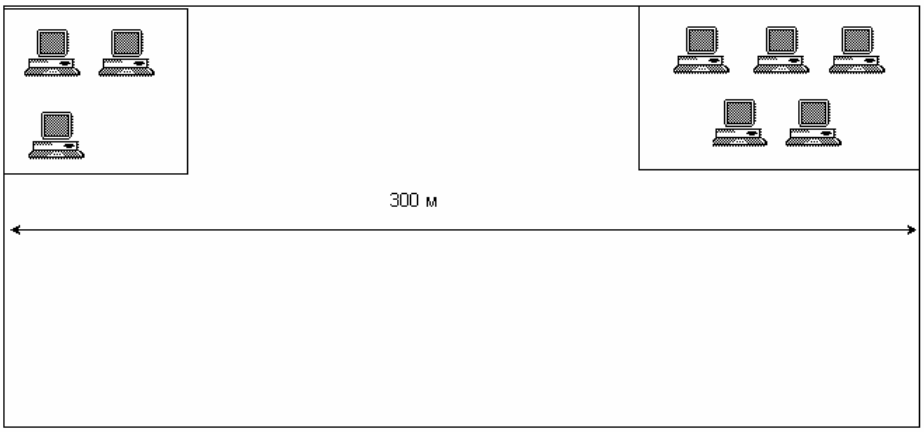
План первого здания



План второго здания

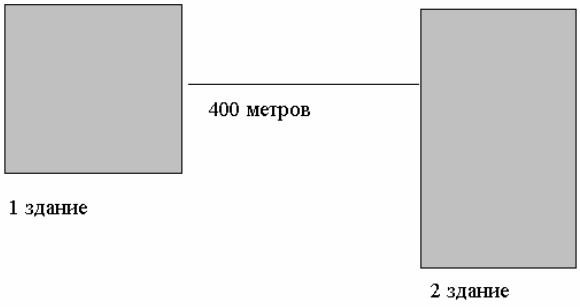


План третьего здания

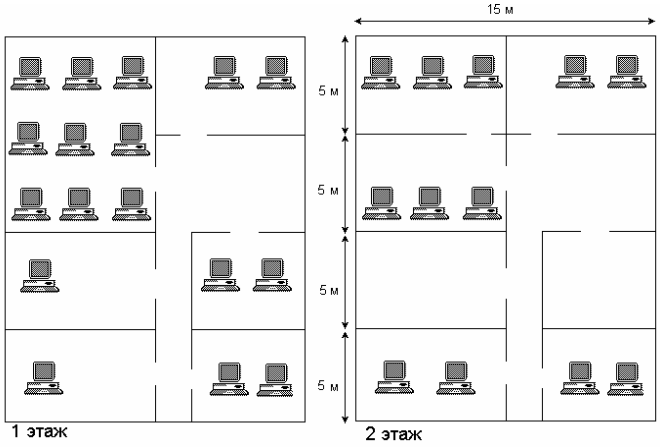


ВАРИАНТ 3

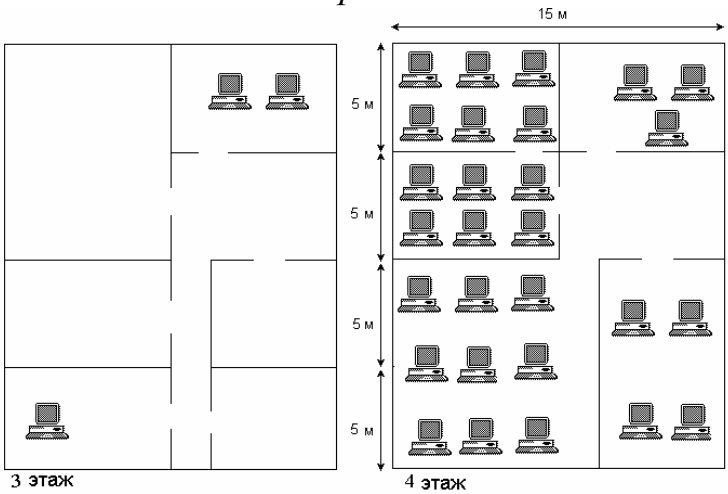
План расположения зданий



План первого здания



План второго здания



V. КОНСПЕКТ ЛЕКЦИЙ

ВВЕДЕНИЕ

Вычислительные сети явились результатом эволюции компьютерных технологий.

Вычислительной сетью называют совокупность компьютеров, соединенную линиями связи и коммуникационными устройствами. Все оборудование сети функционирует под управлением системного и прикладного программного обеспечения.

Использование вычислительных сетей дает следующие возможности:

разделение дорогостоящих ресурсов и периферийных устройств;

совершенствование коммуникаций;

улучшение доступа к информации;

совместное использование приложений;

улучшение взаимодействия в офисе;

быстрое и качественное принятие решений;

свобода в территориальном размещении компьютеров.

В настоящее время информационно-вычислительные системы принято делить на три основных типа по территориальному признаку:

- LAN (Local Area Network) - *локальная сеть*, т.е. сеть в пределах предприятия, учреждения, одной организации. Размер локальной сети не превышает нескольких километров. На практике локальные сети чаще определяют по функциональным, а не физическим характеристикам. Как правило, локальная сеть находится в пределах одного здания.
- MAN (Metropolitan Area Network) - городская или региональная сеть, т.е. сеть в пределах города, области и т.п. Менее распространенный тип сетей. Они используют цифровые магистральные линии связи, часто оптоволоконные, со скоростью до 45 Мбит/с, и предназначены для объединения локальных сетей в масштабах города и их соединение с глобальными сетями. Первоначально они были разработаны для передачи данных, теперь поддерживают видеоконференции и передачу голоса.
- WAN (Wide Area Network) - глобальная сеть, т.е. сеть, соединяющая абонентов страны, континента, всего мира. При организации таких сетей уже существующие линии связи, например телефонные линии. Качество таких линий связи, как правило, очень низкое, что требует использования сложных специальных алгоритмов и процедур передачи данных и дорогой аппаратуры. Скорость обмена данными существенно ниже, чем в LAN-сетях.

В последнее время отличие глобальных и локальных сетей становятся все менее заметными.

Еще одним популярным способом классификации сетей являет их классификация по масштабу.

Локальные *сети рабочих групп* объединяют небольшое количество компьютеров, работающих, как правило, под управлением одной операционной системы. В сети выделен один компьютер, который выполняет сетевые службы, например файловый сервер, сервер печати, сервер факса.

Локальные *сети отделов* могут объединять компьютеры целого отдела. Количество компьютеров может быть в несколько раз больше, чем в сетях рабочих групп. Сетевые службы могут быть распределены между отдельными выделенными компьютерами-серверами.

Сети кампусов преследуют цель объединения нескольких мелких сетей в одну большую сеть. Значительный сетевой трафик локализован в рамках сетей отделов и рабочих групп.

Корпоративные сети объединяют компьютеры и сети в рамках одного предприятия или корпорации. Территориальный признак не имеет никакого значения. Такие сети могут охватывать любую часть земного шара.

ОСНОВНЫЕ ТРЕБОВАНИЯ, ПРЕДЪЯВЛЯЕМЫЕК СОВРЕМЕННЫМ ВЫЧИСЛИТЕЛЬНЫМ СЕТЯМ

Вычислительная сеть создается для обеспечения потенциального доступа к любому ресурсу сети для любого пользователя сети. Качество доступа к ресурсу может быть описано многими показателями, выбор которых зависит от задач, стоящих перед сетью. Среди основных показателей можно выделить:

- производительность,
- надежность,
- управляемость,
- расширяемость,
- прозрачность.

Независимо от выбранного показателя качества обслуживания сети существует два подхода к его обеспечению. Первый состоит в том, что сеть (точнее, обслуживающий персонал) гарантирует пользователю соблюдение некоторой числовой величины показателя качества обслуживания. Например, что средняя пропускная способность между пользователями А и В будет не менее 5 Мбит/с. Второй подход - сеть обслуживает пользователей в соответствии с их приоритетами, т.е. качество обслуживания зависит от степени привилегированности пользователя. Качество обслуживания при этом не гарантируется, а гарантируется только уровень привилегий. Такое обслуживание называется обслуживание с наибольшим старанием.

Глава 1. Принципы построения вычислительных сетей

Изучение сети в целом предполагает знание принципов работы ее отдельных компонентов: компьютеров, коммуникационного оборудования, операционных систем, сетевых приложений.

1.1. Связь компьютера с периферийными устройствами

Для обмена данными между компьютером и периферийным устройством (ПУ) в компьютере предусмотрен внешний интерфейс (рис. 1.1), то есть набор

проводов, соединяющих компьютер и периферийное устройство, а также набор правил обмена информацией по этим проводам (интерфейс или по-другому протокол). Примерами интерфейсов, используемых в компьютерах, являются параллельный интерфейс Centronics, предназначенный, как правило, для подключения принтеров, и последовательный интерфейс RS-232C, через который подключаются мышь, модем и много других устройств.

Рис. 1.1. Связь компьютера с периферийным устройством

Со стороны компьютера интерфейс реализуется совокупностью аппаратных и программных средств: контроллером ПУ и драйвером соответствующего периферийного устройства.

Со стороны ПУ интерфейс чаще всего реализуется аппаратным устройством управления, хотя встречаются и программно-управляемые периферийные устройства.

1.2. Взаимодействия двух компьютеров

В самом простом случае взаимодействие компьютеров может быть реализовано с помощью тех же самых средств, которые используются для взаимодействия компьютера с периферией, например, через последовательный интерфейс RS-232C. Только в этом случае происходит взаимодействие двух программ, работающих на каждом из компьютеров.

Программа, работающая на одном компьютере, не может получить непосредственный доступ к ресурсам другого компьютера - его дискам, файлам, принтеру. Она формирует запрос для программы, работающей на компьютере, которому принадлежат требуемые ресурсы. Запросы выражаются в виде сообщений, передаваемых по каналам связи между компьютерами. Сообщения мо-

гут содержать не только команды на выполнение некоторых действий, но и информационные данные (например, содержимое некоторого файла).

Для передачи сообщения от компьютера А к компьютеру В, приложение А обращается к драйверу COM-порта, сообщая ему адрес в оперативной памяти, по которому драйвер находит сообщение и затем передает его побайтно приложению В. Приложение В, приняв запрос, выполняет его, считывая требуемую область файла с диска с помощью средств локальной ОС в буферную часть своей оперативной памяти и с помощью драйвера COM-порта передает считанные данные по каналу связи в компьютер А.

В операционную систему компьютеров, ресурсы которых должны быть доступны, необходимо добавить модули, постоянно находящиеся в режиме ожидания запросов. Эти модули называются программными серверами. На компьютере, пользователи которого хотят получать доступ к ресурсам других компьютеров, добавляются программные клиенты. Пара модулей «клиент-сервер» обеспечивают совместный доступ пользователей к определенному типу ресурсов. Термины «клиент» и «сервер» используются и для обозначения компьютеров, подключенных к сети.

Схема взаимодействие программных компонентов при связи двух компьютеров приведена на рис. 1.3.

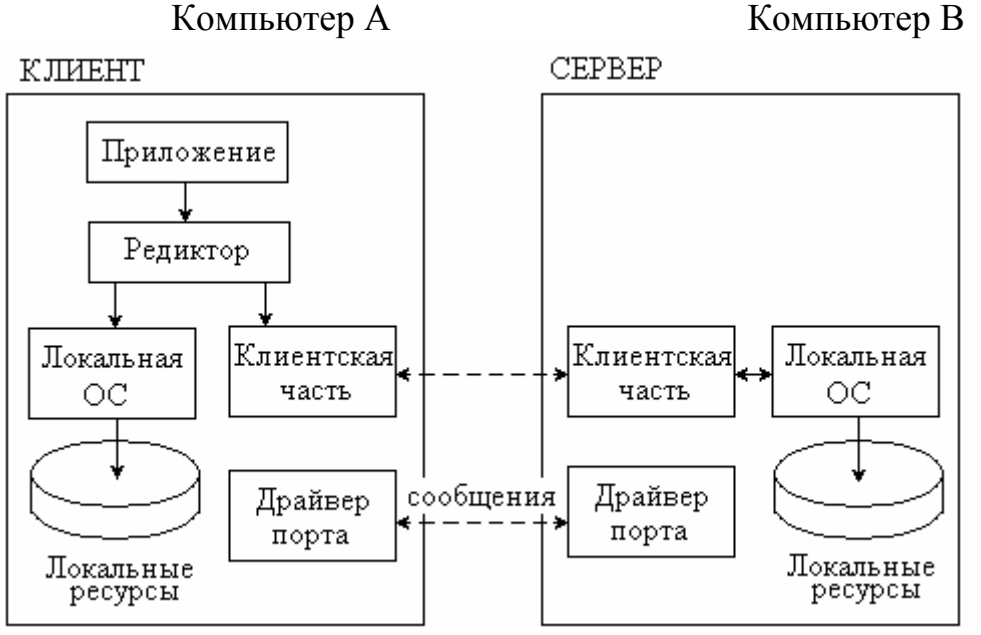


Рис. 1.3. Взаимодействие клиента и сервера

Удобной и полезной функцией клиентской программы является способность отличить запрос к удаленному файлу от запроса к локальному файлу. В этом случае клиентская программа распознает и перенаправляет (redirect) запрос к удаленной машине. Редиктор иногда представляет собой отдельный модуль, а иногда редиктором называют всю клиентскую часть.

В случаях, когда оба компьютера пользуются локальными ресурсами друг друга, на каждом из них должны устанавливаться серверные и клиентские части.

1.3. Проблемы физической передачи данных по линиям связи

В вычислительной технике для представления данных используется двоичный код. Внутри компьютера единицам и нулям данных соответствуют дискретные электрические сигналы. Представление данных в виде электрических или оптических сигналов называется *кодированием*. Существуют различные способы кодирования двоичных цифр 1 и 0, например, потенциальный способ, при котором единице соответствует один уровень напряжения, а нулю — другой, или импульсный способ, когда для представления цифр используются импульсы различной или одной полярности.

Аналогичные подходы могут быть использованы для кодирования данных и при передаче их между двумя компьютерами по линиям связи. Однако эти линии связи отличаются по своим электрическим характеристикам от тех, которые существуют внутри компьютера. Главное отличие внешних линий связи от внутренних состоит в их гораздо большей протяженности, а также в том, что они проходят вне экранированного корпуса по пространствам, зачастую подверженным воздействию сильных электромагнитных помех. Все это приводит к значительным искажениям прямоугольных импульсов (например, «завариванию» фронтов), чем внутри компьютера. Поэтому для надежного распознавания импульсов на приемном конце линии связи при передаче данных внутри и вне компьютера не всегда можно использовать одни и те же скорости и способы кодирования.

В вычислительных сетях применяют как потенциальное, так и импульсное кодирование дискретных данных, а также специфический способ представления данных, который никогда не используется внутри компьютера, — *модуляцию* (рис. 1.3).

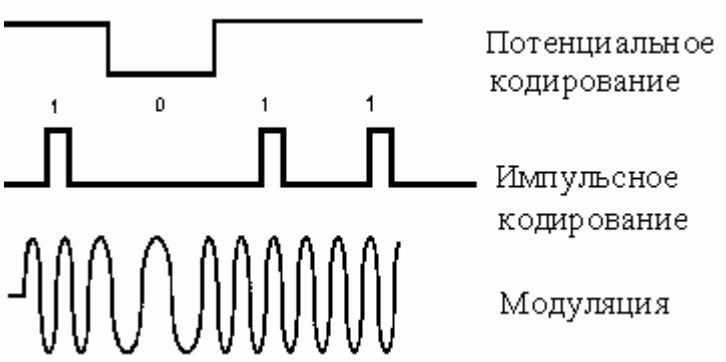


Рис. 1.3. Примеры представления дискретной информации

На способ передачи сигналов влияет и количество проводов в линиях связи между ком

Задачи надежного обмена двоичными сигналами, представленными соот-

ветствующими электромагнитными сигналами, в вычислительных сетях решает определенный класс оборудования. В локальных сетях это сетевые адаптеры, а в глобальных сетях — аппаратура передачи данных, к которой относятся, например, устройства, выполняющие модуляцию и демодуляцию дискретных сигналов, — модемы. Это оборудование кодирует и декодирует каждый информационный бит, синхронизирует передачу электромагнитных сигналов по линиям связи, проверяет правильность передачи по контрольной сумме и может выполнять некоторые другие операции. Сетевые адаптеры рассчитаны, как правило, на работу с определенной передающей средой — коаксиальным кабелем, витой парой, оптоволокном и т. п. Каждый тип передающей среды обладает определенными электрическими характеристиками, влияющими на способ использования данной среды, и определяет скорость передачи сигналов, способ их кодирования и некоторые другие параметры.

1.4. Проблемы объединения нескольких компьютеров

При объединении в сеть большего числа компьютеров возникает целый комплекс новых проблем.

Топология физических связей

В первую очередь необходимо выбрать способ организации физических связей, то есть топологию. Под топологией вычислительной сети понимается конфигурация графа, вершинам которого соответствуют компьютеры сети (иногда и другое оборудование, например концентраторы), а ребрам — физические связи между ними. Компьютеры, подключенные к сети, часто называют станциями или узлами сети.

Необходимо различать физическую и логическую топологии сети. Конфигурация физических связей определяется электрическими соединениями компьютеров между собой и может отличаться от конфигурации логических связей между узлами сети. Логические связи представляют собой маршруты передачи данных между узлами сети и образуются путем соответствующей настройки коммуникационного оборудования.

Рассмотрим некоторые, наиболее часто встречающиеся топологии.

Полносвязная топология (рис. 1.4, а) соответствует сети, в которой каждый компьютер сети связан со всеми остальными.

Все другие варианты основаны на неполносвязных топологиях, когда для обмена данными между двумя компьютерами может потребоваться промежуточная передача данных через другие узлы сети.

Ячеистая топология получается из полносвязной путем удаления некоторых возможных связей (рис. 1.4, б). В сети с ячеистой топологией непосредственно связываются только те компьютеры, между которыми происходит интенсивный обмен данными, а для обмена данными между компьютерами, не соединенными прямыми связями, используются транзитные передачи через промежуточные узлы

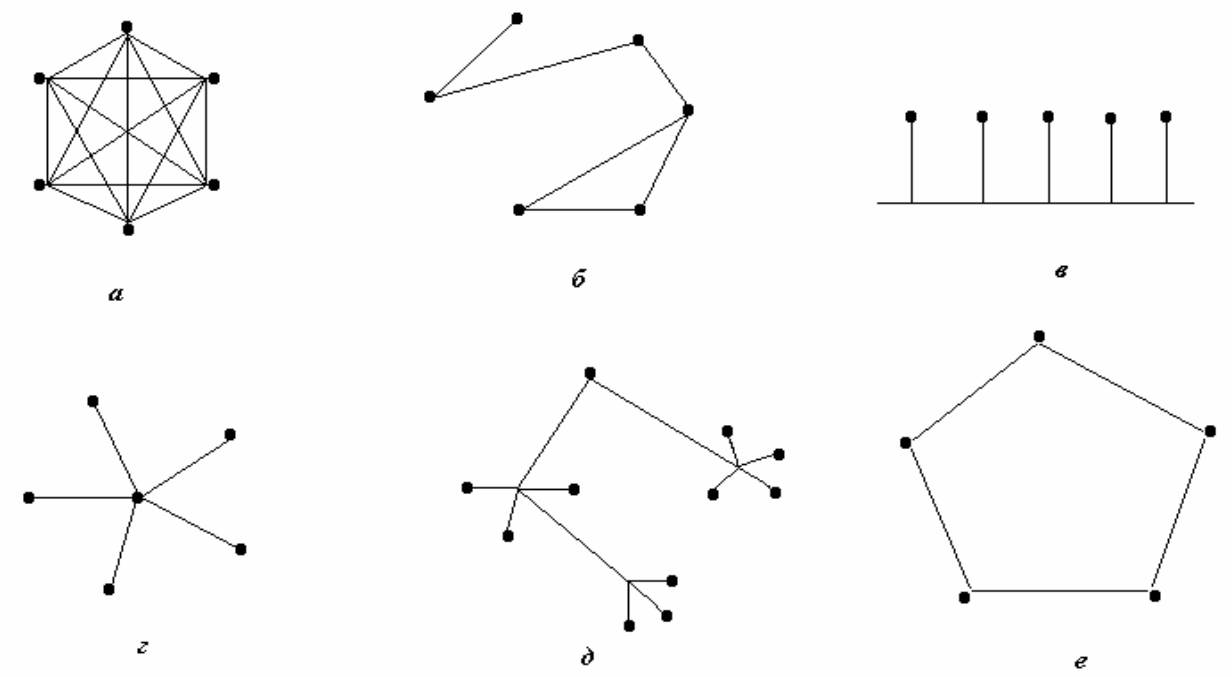


Рис. 1.4. Типовые топологии сетей

Шина (рис. 1.4, в) является распространенной топологией для локальных сетей. Передаваемая информация может передаваться в обе стороны. Основными преимуществами данной схемы – дешевизна и простота проводки кабеля по помещениям. Недостаток – низкая надежность, невысокая производительность.

В топологии звезда (рис. 1.4, г) каждый компьютер подключается отдельным кабелем к общему устройству, называемому концентратором (это может быть Repeater или Hub). В функции концентратора входит направление передаваемой информации одному или всем компьютерам сети. Главное преимущество такой топологии надежность. Кроме того, возможности наращивания количества узлов в сети ограничивается количеством портов концентратора. Иногда имеет смысл строить сеть с использованием нескольких концентраторов, иерархически соединенных друг с другом связями типа звезда (рис. 1.4, д). Иерархическая звезда в настоящее время является самой распространенной топологией, как в локальных, так и в глобальных сетях.

В сетях с кольцевой конфигурацией (рис. 1.4., е) данные передаются по кольцу, как правило, в одном направлении. В то время как небольшие сети, как правило, имеют типовую типологию, для крупных сетей характерно наличие произвольных связей между узлами. В таких сетях можно выделить отдельные фрагменты, имеющие типовую топологию, поэтому их называют сетями смешанной топологии.

В настоящее время выделяют также сотовую (концентрическую) топологию. Она рассматривается как топология подключения устройств в беспроводных сетях. Сотовая топология метод разделения географической области на Зоны или Соты (Cell). В каждой зоне обеспечивается обмен информацией между станциями сети, находящимися внутри зоны (рис.1.5). В каждой зоне могут устанавливаться ретрансляторы, позволяющие обмениваться информацией стан-

циям, находящимся в разных зонах. В этой топологии, как правило, предусматривается возможность перемещения станций между зонами, при этом не должен нарушаться обмен данными с такими станциями.

Сотовая топология позволяет использовать мобильные рабочие станции, легко менять конфигурацию сети. Размер зоны зависит от мощности ретранслятора и мощности приемопередатчиков, установленных на станциях сети.

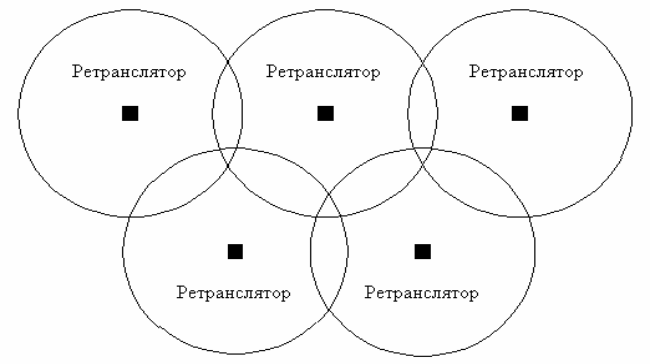


Рис.1.5. Сотовая, концентрическая топология

1.5. Организация взаимодействия устройств в сети

В зависимости от способа организации обработки данных и взаимодействия пользователей выделяют два типа сетей:

- иерархические сети;
- сети клиент/сервер;

В иерархических системах задачи хранение данных; организация доступа пользователей или программ к данным; обработка данных; передача данных или результатов обработки данных пользователям или программам; выполняются аппаратными и программными средствами одного основного компьютера. В архитектуре клиент/сервер часть этих задач решает сервер, а часть компьютер или программа, выступающая в качестве клиента.

СЕТИ КЛИЕНТ/СЕРВЕР

Клиент это какая-либо задача, рабочая станция, наконец просто пользователь. Он может сформировать запрос для выполнения сервером каких-либо сложных или специализированных процедур. Сервер - это устройство или компьютер, выполняющий обработку поступившего от клиента запроса. После выполнения задания результаты передаются клиенту. Как правило, сервер отвечает за хранение данных общего пользования, организацию доступа к этим данным и передачу данных клиенту. Клиент, получив данные, выполняет их обработку и представляет результаты обработки в удобном для пользователя виде. Следует отметить, что иногда обработка данных также выполняется сервером.

В системах клиент/сервер требования к производительности компьютеров, используемых в качестве клиента и сервера, значительно ниже, чем в иерархических системах.

По организации взаимодействия принято выделять два типа систем, использующих метод клиент/сервер:

- равноправная сеть;
- сеть с выделенным сервером.

РАВНОПРАВНАЯ СЕТЬ

Равноправная сеть - сеть, в которой нет единого центра управления взаимодействием рабочих станций, нет единого устройства хранения данных. Операционная система такой сети распределена по всем рабочим станциям, поэтому каждая рабочая станция одновременно может выполнять функции как сервера, т.е. обслуживать запросы от других рабочих станций, так и клиента - направляет свои запросы другим рабочим станциям. Пользователю в такой сети доступны все устройства (принтеры, жесткие диски и т.п.), подключенные к другим рабочим станциям (конечно, если администратор сети предоставил ему соответствующие права).

Достоинства равноправной сети:

- низкая стоимость;
- высокая надежность.

Сети такого типа имеют следующие недостатки:

- сильная зависимость эффективности работы сети от количества одновременно работающих станций (практика показывает, что нормальную работу в такой сети можно обеспечить при количестве рабочих станций не более 10);
- трудности организации эффективного управления взаимодействием рабочих станций и обеспечение секретности информации;
- трудности обновления и изменения программного обеспечения рабочих станций.

СЕТЬ С ВЫДЕЛЕННЫМ СЕРВЕРОМ

Один из компьютеров (сервер сети) выполняет функции хранения данных общего пользования, организации взаимодействия между рабочими станциями, выполнения сервисных услуг. На таком компьютере, как правило, выполняется сетевая операционная система, и все разделяемые устройства (жесткие диски, принтеры, модемы и т.п.) подключаются непосредственно к нему. Взаимодействие рабочих станций в такой сети осуществляется, как правило, только через сервер. Достоинства системы с выделенным сервером:

- выше скорость обработки данных;
- надежная система зашиты информации и обеспечения секретности;
- простота в управлении по сравнению с равноправными сетями.

Недостатки:

- требуется отдельный компьютер под сервер сети, поэтому обычно такая сеть дороже;
- быстродействие и надежность сети зависят от компьютера, используемого в качестве сервера сети;

- сеть с выделенным сервером менее гибкая по сравнению с равноправной.

1.6. Структуризация как средство построения больших сетей

При построении больших сетей однородная структура связей превращается из преимущества в недостаток. В таких сетях использование типовых структур порождает различные ограничения, важнейшими из которых являются:

- ограничения на длину связи между узлами;
- ограничения на количество узлов в сети;
- ограничения на интенсивность трафика, порождаемого узлами сети.

Для снятия этих ограничений используются специальные методы структуризации сети и специальное структурообразующее оборудование — повторители, концентраторы, мосты, коммутаторы, маршрутизаторы. Оборудование такого рода также называют коммуникационным, имея в виду, что с помощью него отдельные сегменты сети взаимодействуют между собой.

Физическая структуризация сети

Простейшее из коммуникационных устройств — повторитель (repeater) — используется для физического соединения различных сегментов кабеля локальной сети с целью увеличения общей длины сети. Повторитель передает сигналы, приходящие из одного сегмента сети, в другие ее сегменты .

Повторитель, который имеет несколько портов и соединяет несколько физических сегментов, часто называют концентратором (Concentrator) или хабом (Hub). Эти названия (Hub — основа, центр деятельности) отражают тот факт, что в данном устройстве сосредоточиваются все связи между сегментами сети.

Сегменты сети подключаются к концентратору физически по топологии «звезда».

Необходимо помнить, что подключение концентратора всегда изменяет физическую топологию сети, но при этом оставляет без изменения ее логическую топологию.

Физическая структуризация сети с помощью концентраторов полезна не только для увеличения расстояния между узлами сети, но и для повышения ее надежности.

Логическая структуризация сети

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

Решение проблемы состоит в отказе от идеи использования единой однородной разделяемой среды.

Логическая структуризация сети - это процесс разбиения сети на сегменты с локализованным трафиком. Для логической структуризации сети используются такие коммуникационные устройства, как мосты, коммутаторы, маршрутизаторы и шлюзы.

Мост (Bridge) делит разделяемую среду передачи сети на части (часто называемые логическими сегментами), передавая информацию из одного сегмента в другой только в том случае, если такая передача действительно необходима, то есть если адрес компьютера назначения принадлежит другой подсети. Тем самым мост изолирует трафик одной подсети от трафика другой, повышая общую производительность передачи данных в сети. Локализация трафика не только экономит пропускную способность, но и уменьшает возможность несанкционированного доступа к данным, так как кадры не выходят за пределы своего сегмента и их сложнее перехватить злоумышленнику.

Мосты используют для локализации трафика аппаратные адреса компьютеров. Поэтому мост достаточно упрощенно представляет деление сети на сегменты. Из-за этого применение мостов приводит к значительным ограничениям на конфигурацию связей сети — сегменты должны быть соединены таким образом, чтобы в сети не образовывались замкнутые контуры.

Коммутатор (switch, switching hub) по принципу обработки кадров ничем не отличается от моста. Основное его отличие от моста состоит в том, что он является своего рода коммуникационным мультипроцессором, так как каждый его порт оснащен специализированным процессором, который обрабатывает кадры по алгоритму моста независимо от процессоров других портов. За счет этого общая производительность коммутатора обычно намного выше производительности традиционного моста, имеющего один процессорный блок. Можно сказать, что коммутаторы — это мосты нового поколения, которые обрабатывают кадры в параллельном режиме.

Ограничения, связанные с применением мостов и коммутаторов — по топологии связей, а также ряд других, — привели к тому, что в ряду коммуникационных устройств появился еще один тип оборудования — маршрутизатор (router). Маршрутизаторы более надежно и более эффективно, чем мосты, изолируют трафик отдельных частей сети
друг от друга. Маршрутизаторы образуют логические сегменты посредством явной адресации, поскольку используют не плоские аппаратные, а составные числовые адреса. В этих адресах имеется поле номера сети, так что все компьютеры, у которых значение этого поля одинаково, принадлежат к одному сегменту, называемому в данном случае подсетью (subnet).

Кроме локализации трафика маршрутизаторы выполняют еще много других полезных функций.

Кроме перечисленных устройств отдельные части сети может соединять шлюз (gateway). Обычно основной причиной, по которой в сети используют шлюз, является необходимость объединить сети с разными типами системного

и прикладного программного обеспечения, а не желание локализовать трафик. Тем не менее, шлюз обеспечивает и локализацию трафика в качестве некоторого побочного эффекта.

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

1.7. Модемы

Модем (**МО**дулятор/**ДЕМ**одулятор) интерфейсное устройство для передачи цифровых данных по аналоговым (например телефонным) линиям. Цифровые данные модем преобразует в аналоговый сигнал и передает по телефонной линии, а при приеме данных выполняет обратное преобразование.

По используемым линиям связи модемы можно разделить на следующие типы:

1) модемы для телетайпных линий связи;
2) модемы для обычных линий связи.

Для передачи данных используются обычные коммутируемые или некоммутируемые телефонные линии связи. Такие линии имеют полосу пропускания 3 кГц, поэтому модемы используют различные методы модуляции и компрессии для передачи данных с высокой скоростью. Так, в стандарте V.34 определяются методы передачи данных по обычным телефонным линиям связи со скоростью 28,8 Кбит/с.

Широкополосные модемы.

Такие модемы работают на выделенных или арендованных линиях связи. За каждой такой линией закрепляется как бы несколько телефонных каналов, поэтому скорость передачи данных для широкополосных модемов может достигать 64 Кбит/с.

Высокоскоростные модемы для специализированных линий связи. При наличии высокоскоростных каналов связи они могут обеспечивать скорость передачи данных до 2 Мбит/с.

При работе с модемами следует различать три типа протоколов.

Первый тип - протокол управления модемом. Этот протокол определяет команды управления работой модема, порядок обмена данными между компьютером или прикладной программой и модемом.

Второй - протокол обмена данными между компьютерами. Как правило, он определяется используемым коммуникационным программным обеспечением.

Третий - протокол обмена данными между модемами. Возможность совместной работы модемов зависит в первую очередь от протоколов или стандартов физического уровня. Они определяют метод модуляции сигналов, порядок установления связи между модемами, скорость передачи данных и т.д.

1.8. МОДЕЛЬ ВЗАИМОСВЯЗИ ОТКРЫТЫХ СИСТЕМ

Для обеспечения обмена данными между компьютерными сетями ISO совместно с CCITT разработала многоуровневый комплект протоколов, известный как Эталонная Модель Взаимосвязи Открытых Систем (ЭМВОС или Open System Interconnection - OSI) (1977 г). Одна из основных идей модели OSI – обеспечить относительно легкий и простой обмен информацией при использовании изготовленных разными фирмами аппаратных и программных средств, соответствующих стандартам OSI. Пользователи должны забыть о проблемах совместимости, которые все еще свойственны системам, включающим устройства различных производителей.

Все задачи, которые необходимо решить для организации взаимодействия между объектами информационной системы, разделены на семь отдельных процедур или уровней (рис. 1.7). Каждый уровень выполняет определенную логическую функцию и обеспечивает определенный набор услуг для расположенного над ним уровня.

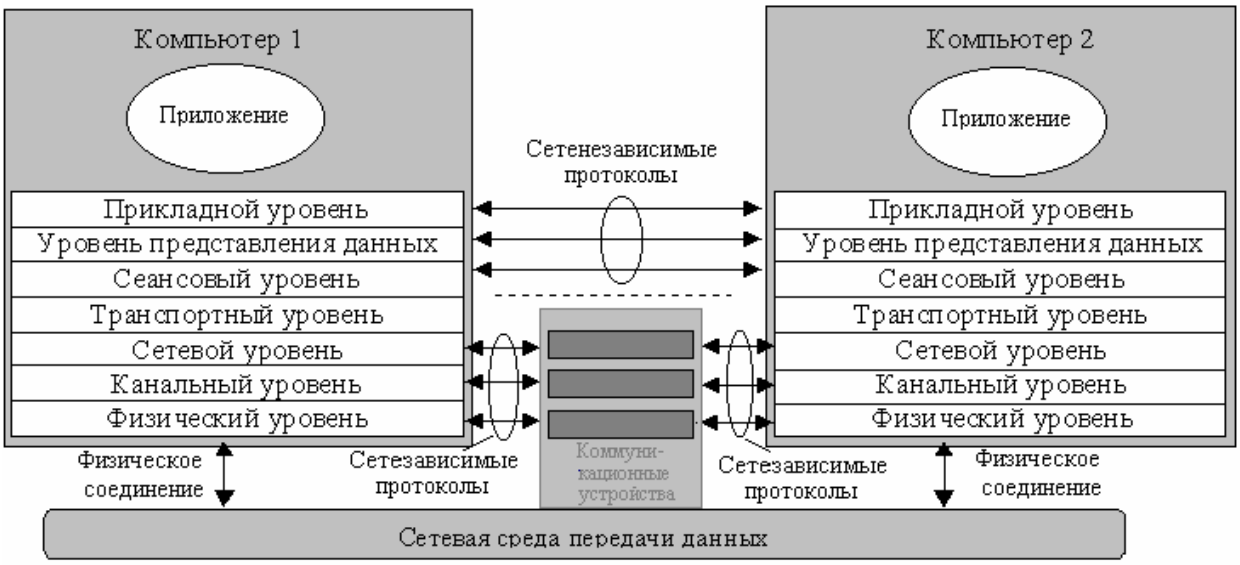


Рис. 1. 7. Модель взаимосвязи открытых систем

Отдельные уровни модели OSI удобно рассматривать как группы программ, предназначенных для выполнения конкретных функций. Программы могут содержать отдельные модули, известные в модели OSI как объекты. Каждый уровень обеспечивает сервис для вышестоящего уровня, запрашивая в свою очередь, сервис у нижестоящего уровня.

При обращении некоторого сетевого приложения к прикладному уровню на основании запроса программное обеспечение формирует сообщение стандартного формата, состоящее из заголовка и поля данных. После формирования сообщения прикладной уровень направляет его вниз. Протокол каждого уровня на основании информации, полученной из заголовка вышележащего уровня, выполняет требуемые действия и добавляет к сообщению собственную слу-

жебную информацию. Когда сообщение по сети поступает на станцию-адресат, оно принимается физическим уровнем и перемещается вверх. Последовательно каждый уровень анализирует и обрабатывает заголовок своего уровня, выполняя соответствующие данному уровню функции, а затем удаляет этот заголовок.

Три нижних уровня – физический, канальный, сетевой - являются сетенезависимыми, то есть протоколы этих уровней тесно связаны с технической реализацией сети и используемым коммуникационным оборудованием.

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

Транспортный уровень является промежуточным, он скрывает все детали функционирования нижних уровней от верхних, что позволяет разрабатывать приложения, независящие от технических средств непосредственной транспортировки сообщений.

ПРИКЛАДНОЙ УРОВЕНЬ (APPLICATION LAYER)

Прикладной уровень обеспечивает доступ прикладных процессов пользователей к ресурсам и сервису информационной системы. Это могут быть программы, обеспечивающие прием или передачу файлов, управление работой сети, передачу почтовых сообщений и т.п. Главная задача этого уровня - обеспечить удобный интерфейс для пользователя. Одна из важных функций прикладного уровня – электронная почта.

Прикладной уровень, кроме того, содержит несколько так называемых общих элементов прикладного сервиса (ACSE – Application Common Service Elements) и специальных элементов прикладного сервиса (SASE – Specific Application Service Elements). Сервисы ACSE предоставляются прикладным процессам во всех системах. Они включают, например, требование определенных параметров качества сервиса. Допустим, при установлении связи через модем прикладной процесс может запросить качество, предусматривающее подтверждение приема и распознавания информации. SASE обеспечивает сервис для конкретных прикладных программ, таких как пересылка файлов и эмуляции терминала.

На сегодняшний день для прикладного уровня модели OSI не существуют международного стандарта; несколько спецификаций информационно-управляющих протоколов претендует в будущем стать международными.

УРОВЕНЬ ПРЕДСТАВЛЕНИЯ (ДАННЫХ PRESENTATION LAYER)

Уровень представления данных отвечает за физическое отображение (представление) информации, не меняя при этом ее содержание. Уровень выполняет следующие функции:

1) преобразование форматов данных;
2) кодирование/декодирование данных, в том числе компрессию и декомпрессию данных.

Форматы представления данных могут различаться по следующим признакам:

1) порядок следования битов и размерность символа в битах;
2) порядок следования байтов;
3) представление или кодировка символов;
4) структура и синтаксис файлов.

СЕАНСОВЫЙ УРОВЕНЬ (SESSION LAYER)

Сеансовый уровень определяет структуру управления взаимодействием абонентов сети, т.е. определяет и контролирует диалог между сетевыми объектами

Выполняет следующие функции:

1) определяет начало и окончание сеанса связи (нормальное или аварийное);
2) определяет время, длительность и режим сеанса связи;
3) определяет точки синхронизации для промежуточного контроля и восстановления при передаче данных;
4) восстанавливает соединение после ошибок во время сеанса связи без потери данных.

На практике немногие приложения используют сеансовый уровень, и он редко реализуется в виде отдельных протоколов, хотя функции этого уровня часто объединяются с функциями прикладного уровня и реализуются в одном протоколе.

ТРАНСПОРТНЫЙ УРОВЕНЬ (TRANSPORT LAYER)

Транспортный уровень выполняет операции:

- устанавливает и разъединяет транспортные соединения;
- контролирует последовательность передачи данных;
- управляет потоком данных;
- обнаруживает и обрабатывает ошибки передачи данных;
- устанавливает соответствие между транспортными (логическими) и сетевыми адресами абонентов;
- позволяет мультиплексировать передаваемые сообщения или соединения.

Примеры протоколов транспортного уровня: SPX, UDP, Tep, NSP.

Транспортный уровень определяет качество сервиса, которое требуется обеспечить посредством сетевого уровня. Предусмотрено три типа сетевого уровня:

- сервис типа А предоставляет сетевые соединения с приемлемым для пользователя количеством необнаруженных ошибок и приемлемой частотой сообщений об обнаруженных ошибках;
- сервис типа В отличается приемлемым для пользователя количеством необнаруженных ошибок, но неприемлемой частотой сообщений об обнаруженных ошибках;
- сервис типа С предоставляет сетевые соединения с количеством необнаруженных ошибок, неприемлемых для сеансового уровня.

Сервис типа С используется в случаях, когда для установки сетевых соединений необходимы дополнительные протоколы, обеспечивающие обнаружение и устранение ошибок.

Существует пять классов сервиса транспортного протокола:

Класс	Наименование	Тип
0	Простой	A
1	Устранение основных ошибок	B
2	Мультиплексирование	A
3	Обнаружение ошибок и мультиплексирование	B
4	Обнаружение и устранение ошибок	C

СЕТЕВОЙ УРОВЕНЬ (NETWORK LAYER)

Сетевой уровень выполняет следующие функции:

- устанавливает сетевые соединения;
- определяет маршрутизацию в сети и связь между сетями (интерсетевой протокол);
- обеспечивает независимость высших уровней от используемой для передачи информации физической Среды.

Основная задача сетевого уровня - маршрутизация данных (передача данных между сетями). Для определения пути передачи данных между сетями на маршрутизаторах строятся таблицы маршрутов, которые содержат список маршрутов, т.е. последовательность передачи данных через маршрутизаторы для доставки данных адресату. Каждый маршрут содержит адрес конечной сети, адрес следующего маршрутизатора и стоимость передачи данных по этому маршруту. При выборе оптимального маршрута применяют динамические или статические методы.

Транспортный и сетевой уровни в значительной степени дублируют друг друга, особенно в плане функций управления потоком данных и контроля ошибок.

Главная причина такого дублирования в том, что существует два варианта связи – *с установлением соединения* (connection-oriented) и *без установления соединения* (connectionless).

На сетевом уровне определяются несколько видов протоколов. Первый вид – сетевые протоколы – реализуют продвижение пакетов через сеть. Именно эти протоколы подразумевают, когда говорят о протоколах сетевого уровня. Другой вид протоколов – протоколы маршрутизации, собирающие информацию о топологии межсетевых соединений. Протоколы сетевого уровня реализуются программными модулями операционной системы, а также программными и аппаратными средствами маршрутизаторов.

На сетевом уровне работают также протоколы, отвечающие за отображение адреса узла, используемого на сетевом уровне, в локальный адрес сети. Они называются протоколами разрешения адресов. Иногда их относят не к сетевому уровню, а к канальному.

Примерами протоколов сетевого уровня служат протокол IP стека протоколов TCP/IP и протокол межсетевого обмена пакетами IPX стека Novell.

КАНАЛЬНЫЙ УРОВЕНЬ (DATA LINK LAYER)

Определяет:

- логическую топологию сети передачи данных;
- метод доступа к среде передачи данных;
- физическую адресацию;
- услуги по установлению соединений между станциями.

Между компьютерными системами может одновременно существовать несколько независимо работающих каналов передачи данных, и канальный уровень обязан обеспечить отсутствие перекрытия между ними.

Поскольку управление потоком и контроль ошибок также входят в функции канального уровня, то он, отслеживая получаемые кадры, ведет статистические записи. По завершении передачи информации пользователем канальный уровень проверяет, все ли данные приняты правильно, а затем закрывает канал.

В протоколах канального уровня ЛВС заложена определенная структура связей между компьютерами и способы их адресации. К типовым топологиям, поддерживаемым протоколами канального уровня, относятся общая шина, кольцо и звезда, а также структуры, полученные из них с помощью мостов и коммутаторов

Примеры протоколов уровня звена данных: Ethernet, Token Ring, FDDI.

ФИЗИЧЕСКИЙ УРОВЕНЬ (PHYSICAL LAYER)

Физический уровень – наименее противоречивый, т.к. включает международные стандарты на аппаратуру, уже вошедшие в обиход.

Физический уровень определяет механические и электрические характеристики передающей среды и интерфейсного оборудования. Функции на этом уровне обеспечивают установление, поддержку и разрыв физического соединения между узлами сети по запросу от канального уровня. К его также относится установление физического соединения между двумя коммуникационными устройствами, формирование сигнала и обеспечение синхронизации этих устройств.

Примеры протоколов физического уровня: X.21, RS232.

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

В широком смысле открытой системой может называться любая система (компьютер, вычислительная сеть, аппаратные или программные продукты), которая построена в соответствии с открытыми спецификациями.

Термин «спецификация» означает формализованное описание аппаратных или программных компонент, способ их функционирования, взаимодействия с другими компонентами, условий эксплуатации, ограничений и особых ха-

рактеристик. Не всякая спецификация является стандартом. Открытые спецификации – опубликованные, общедоступные, соответствующие стандарту и принятые после всестороннего обсуждения.

Для реальных систем полная открытость является недостижимым идеалом. Даже в системах называемых открытыми, этому определению соответствуют лишь некоторые части, поддерживающие внешний интерфейс.

Модель OSI касается только одного аспекта открытости – открытости взаимодействия устройств, связанных в вычислительную сеть. Если две сети построены с соблюдением принципа открытости, это дает следующие преимущества:

- возможность построения сети из аппаратных и программных средств различных производителей, поддерживающих один и тот же стандарт;
- возможность безболезненной замены отдельных компонентов сети другими, более современными, что позволяет развиваться сети с наименьшими затратами;
- возможность сопряжения одной сети с другой;
- простота освоения и обслуживания сети.

Ярким примером открытой системы является Internet.

Глава 2 Линии связи

2.1 Типы линий связи

Линия связи в общем случае состоит из физической среды, по которой передаются электрические информационные сигналы, аппаратуры передачи данных и промежуточной аппаратуры. Синонимом термина линия связи является термин канал связи.

Физическая среда передачи данных может представлять собой набор проводов, изоляционных и защитных оболочек и соединительных разъемов, а также земную атмосферу или космическое пространство, через которые распространяются электромагнитные волны.

В зависимости от среды передачи данных линии связи разделяют на следующие:

- проводные (воздушные),
- кабельные (медные и волоконно-оптические),
- радиоканалы наземной и спутниковой связи.

Проводные линии связи представляют собой провода без каких-либо изолирующих или экранирующих оплеток, проложенные между столбами и висящие в воздухе. Скоростные качества и помехозащищенность этих линий оставляет желать много лучшего. Сегодня проводные линии вытесняются кабельными.

Кабель состоит из проводников, заключенных в несколько слоев изоляции: электрической, электромагнитной, механической, а также возможно, климатической. В компьютерных сетях применяются три основных типа кабеля:

кабели на основе скрученных пар медных проводов, коаксиальные кабели с медной жилой и волоконно-оптические кабели.

ВИТАЯ ПАРА (TWISTED PAIR CABLE)

Витая пара (другое название телефонная пара) - тип кабеля, используемого в телефонии. Витой парой выполняется подключение абонентов к телефонному коммутатору или станции. Такие линии связи прокладываются при строительстве зданий, поэтому использование витой пары для локальной сети позволяет сократить расходы на монтажные работы при установке сети до минимума, так как монтаж заключается только в коммутации уже существующих линий связи.

Кабель витой пары состоит из двух проводников, заключенных в оболочку. Для уменьшения влияния помех проводники скручиваются с определенным шагом скрутки.

Существуют экранированная (Shielded Twisted Pair - STP) и неэкранированная (Unshielded Twisted Pair - UTP) витые пары, которые различаются наличием дополнительного защитного экранного слоя. Экранированная пара более устойчива к электромагнитным помехам, поэтому при возможности лучше применять такой тип кабеля. На практике чаще используется неэкранированная витая пара. Это происходит, во-первых, из-за того, что такой тип кабеля чаще используется для разводки телефонных линий, а, во-вторых, неэкранированная витая пара дешевле экранированной.

КОАКСИАЛЬНЫЙ КАБЕЛЬ (COAXIAL CABLE)

Наиболее часто используется в качестве передающей среды (рис.2.1) коаксиал обладает более высокой устойчивостью к электромагнитным помехам по сравнению с витой парой. Расстояние между активными устройствами может быть до 2 км, а скорость передачи данных - от 2,5 до 100 Мбит/с, в зависимости от типа локальной сети.

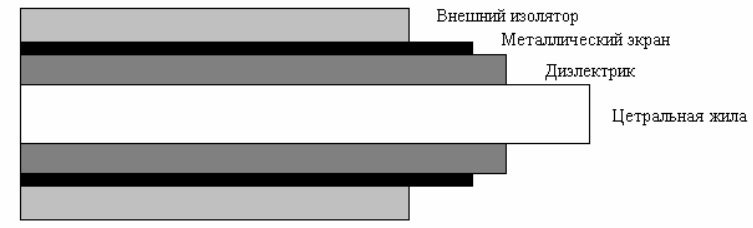


Рис. 2.1. Коаксиальный кабель

ВОЛОКОННО-ОПТИЧЕСКИЙ КАБЕЛЬ (OPTICAL FIBER)

Состоит из тонких (5 – 60 микрон) волокон, по которым распространяются световые сигналы. Это наиболее качественный тип кабеля – он обеспечивает передачу данных с высокой скоростью (до 10 Гбит/с и выше) и лучше других типов передающей среды обеспечивает защиту данных от внешних помех.

РАДИОКАНАЛЫ НАЗЕМНОЙ И СПУТНИКОВОЙ СВЯЗИ

Образуются с помощью передатчика и приемника радиоволн. Различные типы радиоканалов отличаются используемым частотным диапазоном и дальностью канала.

Диапазоны коротких, средних и длинных волн (КВ, СВ, ДВ), называемые также диапазонами амплитудной модуляции (Amplitude Modulation, AM), обеспечивают дальнюю связь, но при невысокой скорости передачи данных.

Более скоростными являются каналы, работающие на диапазонах ультракоротких волн (УКВ), для которых характерна частотная модуляция(Frequency Modulation, FM), а также диапазонах сверхвысоких частот (СВЧ или microwaves). В диапазоне СВЧ (свыше 4 ГГц) сигналы уже не отражаются ионосферой Земли, и для устойчивой связи требуется наличие прямой видимости между передатчиком и приемником. Поэтому эти каналы используют либо спутниковую связь, либо радиорелейные каналы.

Беспроводные локальные сети считаются перспективным направлением развития локальных сетей. Основное преимущество перед другими типами сетей - простота и мобильность. Микроволновые мосты и повторители позволяют объединить локальные сети без использования дорогостоящих оптоволоконных или телефонных линий связи. Современное оборудование обеспечивает высокие помехозащищенность и пропускную способность микроволновых линий связи. При этом необязательно, чтобы между соединяемыми сетями была прямая видимость, так как некоторые компании выпускают оборудование, которое может работать на отраженном сигнале. Для обеспечения защищенности, секретности передаваемых данных применяется их шифрование, а иногда используется метод псевдослучайного изменения несущей частоты. Приемник и передатчик при установлении связи договариваются о порядке изменения рабочей несущей частоты. Когда начинается обмен данными, несущая частота постоянно меняется, поэтому злоумышленнику достаточно сложно перехватить данные.

2.2. Характеристики линий связи

К основным характеристикам линий связи относятся:

- амплитудно-частотная характеристика,
- полоса пропускания,
- затухание,
- помехоустойчивость,
- перекрестные наводки на ближнем конце линии,
- пропускная способность,
- достоверность передачи данных,
- удельная стоимость.

Для определения характеристик линии связи часто используют анализ ее реакции на эталонные воздействия. Чаще всего в качестве эталонных сигналов используют синусоидальные сигналы различных частот.

Степень искажения синусоидальных сигналов линиями связи оценивается с помощью таких характеристик, как амплитудно-частотная характеристика (АЧХ), полоса пропускания и затухание.

АЧХ показывает, как затухает амплитуда синусоиды на выходе линии связи по сравнению с амплитудой на ее входе для всех возможных частот передаваемого сигнала. Вместо амплитуды в этой характеристике часто используют мощность сигнала. На практике вместо АЧХ применяются другие, упрощенные характеристики – полоса пропускания и затухание.

Полоса пропускания – непрерывный диапазон частот, для которого отношение амплитуды выходного сигнала ко входному превышает некоторый заранее заданный предел, обычно 0.5. Ширина полосы пропускания влияет на максимально возможную скорость передачи информации.

Затухание определяется как относительное уменьшение амплитуды или мощности сигнала при передаче определенной частоты. Таким образом, затухание представляет собой одну точку из АЧХ. Затухание А обычно измеряется в децибелах (дБ) и вычисляется по формуле:

$$A=10 \lg (P_{вых}/P_{вх}),$$

где $P_{вых}$ и $P_{вх}$ соответственно мощности сигнала на выходе и входе линии.

Так как мощность выходного сигнала кабеля без промежуточных усилителей всегда меньше, чем мощность входного сигнала, затухание является отрицательной величиной. На практике, часто оперируют с абсолютными значениями затухания.

Пропускная способность линии характеризует максимально возможную скорость передачи, измеряется в битах в секунду. Пропускная способность зависит не только от АЧХ, но и от спектра передаваемого сигнала.

Связь между полосой пропускания линии связи и ее максимально возможной пропускной способностью, вне зависимости от принятого способа физического кодирования, установил Клод Шеннон:

$$C= F \log_2 (1+P_c/P_ш),$$

где C – максимальная пропускная способность линии в битах в секунду, F – ширина полосы пропускания линии в герцах, P_c – мощность сигнала, $P_ш$ - мощность шума.

Близким по сути к формуле Шеннона является соотношение Найквиста, которое определяет максимально возможную пропускную способность, без учета шума на линии:

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

где M – количество различимых состояний сигнала.

Помехоустойчивость линии определяет ее способность уменьшать уровень помех, создаваемых во внешней среде, на внутренних проводниках.

Перекрестные наводки на ближнем конце (Near End Cross Talk - NEXT) определяют помехоустойчивость кабеля к внутренним источникам помех, когда электромагнитное поле сигнала, передаваемого выходом передатчика по одной паре проводников, наводит на другую пару проводников сигнал помехи.

2.3. Стандарты кабеля

В компьютерных сетях применяются кабели, удовлетворяющие определенным стандартам, что позволяет строить кабельную систему из кабелей и соединительных устройств разных производителей. Сегодня наиболее употребительными стандартами в мировой практике являются следующие:

- Американский стандарт EIA/TIA-568A, разработанный ANSI, EIA/TIA и лабораторией underwriters Labs (UL). Он стал приемником стандарта EIA/TIA-568;
- Международный стандарт ISO/IEC 11801;
- Европейский стандарт EN50173.

Наиболее важными характеристиками, оговариваемыми стандартами, являются:

- Затухание.
- Перекрестные наводки на ближнем конце (NEXT).
- Импеданс (волновое сопротивление) – полное (активное и реактивное) сопротивление в электрической сети. Импеданс измеряется в Омах и является относительно постоянной величиной. В области высоких частот (100 – 200 МГц) импеданс зависит от частоты.
- Активное сопротивление – сопротивление постоянному току в электрической цепи. Активное сопротивление не зависит от частоты и возрастает с увеличением длины кабеля.
- Емкость – свойство металлических проводников накапливать энергию. Эта характеристика является нежелательной величиной. Ее большое значение приводит к искажению сигнала и ограничивает полосу пропускания.

2.3.1. Кабели на основе неэкранированной пары

Медный неэкранированный кабель UTP делится на 5 категорий (Category 1 - Category 5):

Категория 1. Кабель предназначен для передачи речевых и цифровых данных для низкоскоростных приложений (до 20 Кбит/с). До 1983 года это был основной тип кабеля для телефонной разводки.

Категория 2. Используется при скорости передачи данных до 4 Мбит/с. Впервые применены IBM. (По классификации IBM: Type 3.)

Кабели категорий 1 и 2 определены в стандарте EIA/TIA-568, но в стандарт EIA/TIA-568A не вошли как устаревшие.

Категория 3. В 1991 определены характеристики кабелей данной категории для частот в диапазоне до 16 Мбит/с, поддерживающие высокоскоростные

приложения и предназначенные для передачи данных и голосовых сообщений. Шаг скрутки равен примерно 3 витка на 1 фут (30,5 см). В настоящее время наиболее используемый кабель.

Категория 4. Используется для передачи на большие расстояния (до 135 метров) и при высоких скоростях передачи данных (до 20 Мбит/с). На практике используются редко.

Категория 5. Кабель с волновым сопротивлением 100 Ом, предназначенный для передачи цифровых данных в высокоскоростных протоколах (до 100 Мбит/с). Кабель данной категории 5 пришел на смену кабеля категории 3. Новые кабельные системы крупных зданий строятся на этом кабеле в сочетании с опто-волокном.

Сравнительно недавно промышленность стала выпускать кабели категорий 6 и 7. Основное их назначение – поддержка высокоскоростных приложений. Некоторые специалисты сомневаются в их необходимости, поскольку стоимость такой кабельной системы соизмерима по стоимости с использованием волоконно-оптических кабелей, а характеристики при этом ниже.

2.3.2. Кабели на основе экранированной пары

Экранированная витая пара STP хорошо защищает передаваемые сигналы от внешних помех. Наличие заземляемого экрана удорожает кабель и усложняет его прокладку, так как требует выполнения качественного заземления. Экранированный кабель применяется только для передачи данных.

Основным стандартом STP является фирменный стандарт IBM, в котором кабели делятся не на категории, а на типы: Type 1, Type 2, . . . , Type 9. Не все типы кабелей стандарта IBM определяют характеристики экранированного кабеля.

Основным типом экранированного кабеля является Type 1. Его электрические характеристики примерно соответствуют параметрам UTP категории 5, однако волновое сопротивление составляет 150 Ом. В случаях, если технология сети может использовать UTP и STP, нужно обратить внимание - на какой тип кабеля рассчитаны трансиверы.

В настоящее время STP Type1 приобрел международный статус и включен в стандарты EIA/TIA-568A, ISO 11801 и др.

2.3.3. Коаксиальный кабель

Основными типа с соответствующими характеристиками являются:

- RG-8 и RG-11 – «толстый» коаксиальный кабель. Имеет волновое сопротивление 50 Ом и внешний диаметр 0,5дюйма (около 12 мм). Этот кабель имеет достаточно толстый внутренний проводник 2.17 мм, обеспечивающий хорошие механические и электрические характеристики (затухание на частоте 10 МГц – не хуже 18 дБ/км). Недостаток – кабель плохо гнется, и следовательно, его сложно монтировать.

- RG-58/U, RG-58 A/U и RG-58 C/U - разновидности «тонкого» коаксиального кабеля. Первый из них имеет сплошной внутренний проводник, второй – многожильный. Все эти кабели имеют волновое сопротивление 50 Ом, но обладают худшими характеристиками по сравнению с «толстым» коаксиалом.

- RG-59 – телевизионный кабель с волновым сопротивлением с волновым сопротивлением 75 Ом.

- RG-62 – кабель с волновым сопротивлением 93 Ома, использовался в сетях Arcnet, оборудование которых сегодня практически не выпускается.

Эти кабели описаны в стандарте EIA/TIA-568. Новый стандарт EIA/TIA-568A коаксиальные кабели не описывает, как морально устаревшие.

2.3.4. Волоконно-оптические кабели

В волоконно-оптическом кабеле для передачи данных используются световые импульсы. Сердечник такого кабеля изготовлен из стекла или пластика. Сердечник окружен слоем отражателя, который направляет световые импульсы вдоль кабеля.

Такой кабель не подвержен воздействию электромагнитных помех, обеспечивает высокую помехозащищенность, секретность передаваемой информации. Производительность волоконно-оптического кабеля составляет до 10 Гбит/с. Передача данных выполняется только в симплексном, однонаправленном режиме, поэтому для организации обмена данными устройства необходимо соединять двумя оптическими волокнами, жилами (на практике оптоволоконный кабель всегда имеет четное, парное количество волокон).

В зависимости от распределения показателя преломления и от величины диаметра сердечника различают:

- многомодовое волокно со ступенчатым изменением показателя преломления
- многомодовое волокно с плавным изменением показателя преломления
- одномодовое волокно.

Понятие «мода» описывает режим распространения световых лучей во внутреннем сердечнике кабеля.

Одномодовый метод (Single-Mode) предъявляет более высокие требования к источнику света и оптоволоконному кабелю. В качестве источника, как правило, используется полупроводниковый лазер. В многомодовом режиме источник света - светодиод. В многомодовом кабеле используется более толстый центральный проводник, в котором одновременно существует несколько световых лучей с разными углами преломления – модами. В этом случае при распространении сигнала происходит "расплывание", что приводит к значительному увеличению коэффициента затухания. Многомодовая технология значительно дешевле одномодовой, поэтому в локальных сетях чаще всего используется именно она.

2.4. Аппаратура линий связи

Аппаратура передачи данных (АПД или DCE – Data Circuit terminating Equipment) непосредственно связывает компьютеры или локальные сети пользователя с линией связи и является пограничным оборудованием. Традиционно аппаратуру передачи данных включают в состав линии связи. Примерами DCE являются модемы, терминальные адаптеры сетей ISDN, оптические модемы, устройства подключения к цифровым каналам.

Аппаратура пользователя линии связи, вырабатывающая данные для передачи по линии связи и подключаемая непосредственно к аппаратуре передачи данных носит название *оконечное оборудование данных* (ООД или DTE – Data Terminal Equipment). Примером DTE могут служить компьютеры или маршрутизаторы локальных сетей.

Разделение оборудования на классы DCE и DTE в ЛВС является достаточно условным.

Промежуточная аппаратура обычно используется на линиях связи большой протяженности и решает задачи:

1) улучшения качества сигнала,
2) создание постоянного составного канала связи между абонентами сети.

Промежуточная аппаратура прозрачна для пользователя. В действительности она образует сложную сеть, которую называют первичной сетью, так как она никаких служб не поддерживает, а только служит основой для построения компьютерных, телефонных и другого рода сетей.

В ЛВС промежуточная аппаратура может совсем не использоваться. В глобальных сетях кроме аппаратуры для усиления сигнала требуется и мультиплексоры, демультиплексоры и коммутаторы, создающие между двумя абонентами сети составной канал из некоммутируемых отрезков, тем самым, решая вторую задачу. Составной канал образуется на долговременной основе, например месяц или год, причем абонент не может влиять на процесс коммутации этого канала.

В зависимости от типа промежуточной аппаратуры все линии связи делятся на *аналоговые* и *цифровые*. В аналоговых линиях промежуточная аппаратура предназначена для усиления аналоговых сигналов, т.е. сигналов, имеющих непрерывный диапазон значений. Для создания высокоскоростных каналов в аналоговом подходе используют технику частотного мультиплексирования (frequency division Multiplexing, FDM). В цифровых линиях связи передаваемые сигналы имеют конечное число состояний (как правило элементарный сигнал, передаваемый за один такт работы передающей аппаратуры, имеет 2-3 состояния, передаваемые импульсами прямоугольной формы). Промежуточная аппаратура цифровых каналов улучшает форму импульсов и обеспечивает их ресинхронизацию. Она работает по принципу временного мультиплексирования каналов (Time Division Multiplexing, TDM), когда каждому низкоскоростному каналу выделяется определенная доля времени (квант) высокоскоростного канала.

Глава 3. БАЗОВЫЕ ТЕХНОЛОГИИ ЛОКАЛЬНЫХ СЕТЕЙ

3.1. Структура стандартов IEEE 802.x

В 1980 году в институте IEEE был организован комитет 802 по стандартизации локальных сетей. Результатом его работы стало семейство стандартов IEEE 802.x, содержащих рекомендации по проектированию нижних уровней локальных сетей.

Стандарты данного семейства охватывают только два нижних уровня модели OSI – физический и канальный. Это обусловлено тем, что именно данные уровни в наибольшей степени отражают специфику локальных сетей. Старшие же уровни, начиная с сетевого, в значительной степени имеют общие черты, как для локальных, так и для глобальных сетей.

Специфика локальных сетей также нашла свое отражение в разделении канального уровня на два подуровня:

- логической передачи данных (Logical Link Control – LLC),
- управления доступом к среде (Media Access Control, MAC).

Уровень MAC появился из-за существования в ЛВС разделяемой среды передачи данных. Этот уровень обеспечивает корректное совместное использование общей среды, предоставляя ее в соответствии с определенным алгоритмом в распоряжении той или иной станции сети. После того как доступ к среде получен, ею может пользоваться более высокий уровень – LLC, организующий передачу логических единиц данных с различным уровнем качества транспортных услуг. В современных ЛВС получили распространение несколько протоколов уровня MAC, реализующих различные алгоритмы доступа к передающей среде. Эти протоколы полностью определяют специфику отдельных технологий.

Уровень LLC отвечает за передачу кадров между узлами с различной степенью надежностью и реализует функции интерфейса с сетевым уровнем. Существует несколько режимов его работы, отличающиеся наличием или отсутствием процедур восстановления кадров в случае их потери или искажения.

Протоколы уровней MAC и LLC взаимно независимы – каждый протокол уровня MAC может использоваться с любым протоколом уровня LLC, и наоборот.

Стандарты IEEE 802 имеют достаточно четкую структуру. приведенную на рис. 2.3.

Описание каждой технологии разделено на две части: описание уровня MAC и описание физического уровня. Как правило, одному уровню MAC соответствует несколько вариантов протоколов физического уровня.

Стандарт LLC курирует подкомитет 802.2. Даже технологии, стандартизированные не в рамках комитета 802, ориентируются на использование протокола LLC, определенного стандартом 802.2, например протокол FDDI, стандартизированный ANSI.

Стандарты, разрабатываемые подкомитетом 802.1, носят общий характер для всех технологий. Здесь разработаны общие определения локальных сетей, их свойства, определена связь трех уровней модели IEEE 802 с моделью OSI.

Наиболее важными являются стандарты 802.1, описывающие взаимодействие различных технологий и построение сложных сетей на основе базовых технологий. Эта группа стандартов носит общее название стандартов межсетевого взаимодействия (internetworking).

Стандарты 802.3, 802.4, 802.5 и 802.12 описывают технологии локальных сетей, полученные в результате улучшений фирменных технологий, легших в их основу.

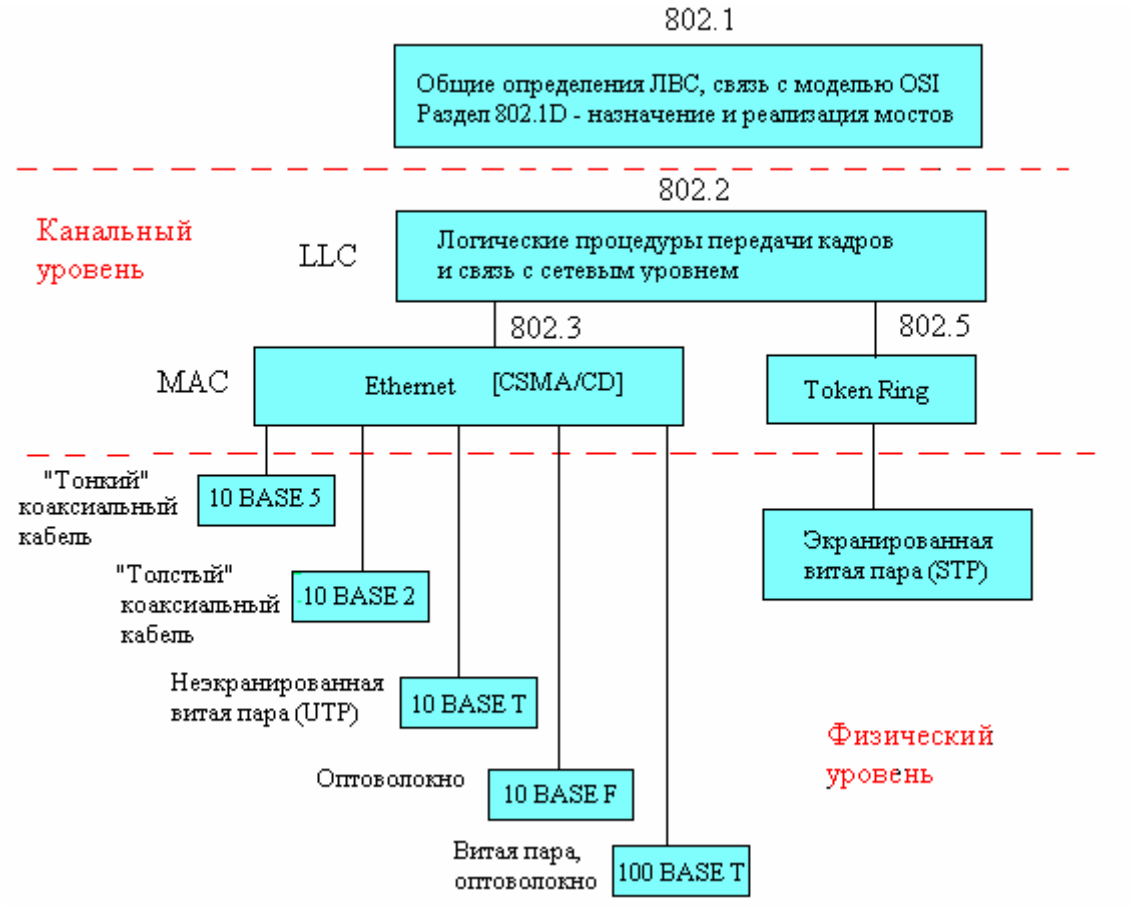


Рис. 3.1 Структура стандартов 802.x

802.6 - Metropolitan Area Network , MAN – сети мегаполисов;

802.8 – Fiber Optic Technical Advisory Group – техническая консультационная группа по волоконно-оптическому кабелю;

802.9 – Integrated Voice and data Networks – интегрированная среда передачи голоса и данных;

802.10 – Network Security – сетевая безопасность;

802.11 – Wireless Networks – беспроводные сети.

3.2. Локальная сеть Ethernet

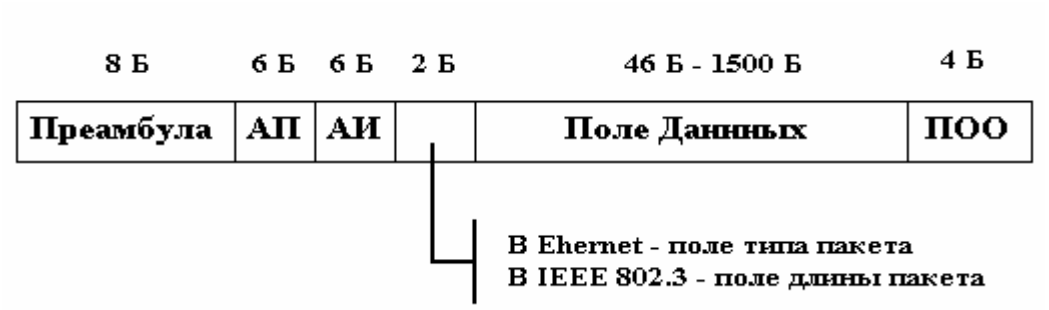
Ethernet – самый распространенный в настоящее время стандарт локальных сетей.

В 1980 года фирмы Xerox, Digital Equipment Corporation и Intel Corporation выпустили стандарт на сеть Ethernet, которую сейчас называют Ethernet версии 1. В 1982 году была опубликована спецификация на Ethernet версии 2.0. Обе версии используются до сих пор, причем между ними существуют различия и по интерфейсу, и по уровню сигналов.

На базе Ethernet версии 2 институтом IEEE был разработан стандарт IEEE 802.3. Различия между ними незначительные, так что оборудование, разработанное для этих стандартов, может одновременно использоваться в одной кабельной системе. В зависимости от типа физической среды стандарт 802.3 имеет различные модификации – 10Base- 5, 10Base- 2, 10Base- T, 10Base- FL, 10Base- FB.

В 1995 году был принят стандарт Fast Ethernet, который во многом не является самостоятельным стандартом. Его описание 802.3u является дополнительным разделом к основному стандарту. Аналогично, принятый 1998 году стандарт Gigabit Ethernet описан в разделе 802.z основного документа.

Минимальная длина пакета в IEEE 802.3 и Ethernet - 64 байта, соответственно длина поля данных - 46 байтов. IEEE 802.3 позволяет использовать символы-заполнители (PAD) в поле данных, если требуется передать сообщение короче, чем 46 байтов. В Ethernet пакет с длиной поля данных менее 46 байтов



просто не будет обрабатываться. На рисунке 2.1. показан пакет в сети Ethernet.

Рис. 3.2. Формат пакета Ethernet

Назначение полей:

1. Преамбула - служит для синхронизации работы приемника и передатчика.

2. АП - Адрес Приемника (DA - Destination Address) адрес станции, которой направляется пакет.

3. АИ - Адрес Источника (SA - Source Address) - адрес передающей станции.

4. Поле Типа Пакета (Type) определяет тип Ethernet пакета. Содержимое этого поля на канальном (Data Link) уровне не обрабатывается. Эта информация предназначена для протоколов более высокого уровня. Существует специальный комитет, который занимается регистрацией типов пакетов.

5. Поле Длины Пакета (Length) определяет в IEEE 802.3 количество байт данных в поле данных. Количество символов-заполнителей (PAD) не входит в это число. В Ethernet нет необходимости в этом поле, поскольку длина является фиксированной.

6. Поле Данных (Data) содержит данные и символы-заполнители в IEEE 802.3, данные - в Ethernet.

7. ПОО Поле Обнаружения Ошибок (CRC) служит для определения достоверности полученной информации.

IEEE 802.3 определяет, что после поля адреса источника следует двухбайтное поле длины пакета, а в Ethernet - поле типа пакета

Спецификация Ethernet и IEEE 802.3 определяют несколько конфигураций подключения оборудования. Они различаются типом кабеля, топологией подключения устройств и т.п. Во всех конфигурациях используется один метод доступа станций к среде (МДКН/ОС) - Carrier Sense Multiple Access/Collision Detection (CSMA/CD).

Во время работы станция постоянно проверяет среду передачи. Передающая среда может быть:

1) свободна - ни одна другая станция не передает данные;

2) занята - идет передача данных другой станцией.

При использовании метода CSMA/CD любая станция, обнаружив, что среда свободна, может начать передачу своих данных. Поэтому возможна ситуация, когда одновременно несколько станций начнут передавать данные.

Коллизия - это ситуация, которая возникает при одновременной передаче данных несколькими станциями сети. При этом происходит смешение и искажение сигналов, поэтому информация, передаваемая в этот момент, недостоверна.

Четкое распознавание коллизий всеми станциями сети является необходимым условием корректной работы сети Ethernet. Если какая-либо станция не распознает коллизию и решит, что переданный ею кадр передан верно, то кадр будет утерян или значительно искажен.

Для надежного распознавания коллизий должно выполняться соотношение:

$T_{min} \geq PDV$,

где T_{min} – время передачи кадра минимальной длины, а PDV – время, за которое сигнал коллизии успевает распространиться до самого дальнего узла сети. Так как в худшем случае сигнал дважды должен пройти между наиболее удаленными станциями (в одну сторону проходит неискаженный сигнал, на обратном пути уже искаженный коллизией), то это время называется *временем двойного оборота* (Path Delay Value).

Выполнение этого условия зависит, с одной стороны, от длины минимального кадра и пропускной способности сети, с другой стороны, от длины кабельной системы и скорости распространения сигнала в кабеле.

Все параметры протокола Ethernet подобраны таким образом, чтобы коллизии всегда четко распознавались.

Основные конфигурации Ethernet

Исторически первые сети технологии Ethernet были созданы на коаксиальном кабеле. В дальнейшем определены и другие спецификации физического уровня, позволяющие использовать различные методы среды передачи (см. рис.3.3).

Стандарт 10Base-5 считается классическим Ethernet. Станция должна подключаться к кабелю при помощи приемопередатчика – *трансивера*.

Ethernet	Thick Wire Ethernet	Thin Wire Ethernet	UTP Ethernet	Fiber Optic Ethernet	Broadband Ethernet
IEEE 802.3	10BASE-5	10BASE-2	10BASE-T	10BASE-F	10BROAD36
Скорость передачи данных, Мбит/с	10	10	10	10	10
Метод передачи сигналов	Одно-Полосной	Одно-полосной	Одно-полосной	Одно-полосной	Широко-полосной
Длина сегмента кабеля, м	500	185	100	2000	1800
Максимальное расстояние между узлами сети (при использовании повторителей), м	2500	925	500	2500	
Максимальное число станций на сегменте	100	30	1024	1024	
Максим. число повторителей между любыми станциями сети	4	4	4	4	
Тип кабеля	50 Омный	50 Омный	Неэкранированная	Многомодовый воло-	75 Омный коаксиаль-

	коакси-альный, «тол-стый» RG-8 или RG-11	коакси-альный, «тонкий» RG-58	витая пара категории 3, 4, 5	конно-оптический кабель	ный, «толстый»

Рис. 3.3. Спецификации сети Ethernet

Трансивер выполняет следующие функции:
- прием и передачу данных с кабеля на кабель;
- определение коллизий на кабеле;
- электрическая развязка между кабелем и остальной частью адаптера;
- защита кабеля от некорректной работы адаптера.

Трансивер соединяется с сетевым адаптером интерфейсным кабелем AUI .Допускается подключение к одному сегменту не более 100 трансиверов, причем расстояние между подключениями трансиверов не должно быть меньше 2.5 м. На кабеле обычно имеется разметка, обозначающая точки подключения трансиверов.

Стандарт 10Base-5 определяет возможность использования повторителя. Правило применение повторителей в данном стандарте носит название «правило 5-4-3»: 5 сегментов, 4 повторителя, 3 нагруженных сегмента. Ограничение числа повторителей объясняется дополнительными задержками распространения сигнала, которые они вносят. Каждый повторитель подключается к сегменту одним трансивером, поэтому к нагруженному сегменту можно подключать не более 99 узлов. Максимальное количество конечных узлов в сети 99*3=297 узлов.

10Base-2 обладает худшей помехозащищенностью, худшей прочностью и более узкой полосой пропускания. Станции подключаются к кабелю с помощью высокочастотного BNC Т-коннектора, который представляет собой тройник, один отвод которого соединяется с сетевым адаптером, а два других – с двумя концами разрыва кабеля. Минимальное расстояние между станциями – 1 м. Повторители используются по «правилу 5-4-3». Общее количество станций в сети 29*3=87.

В **10Base-T** конечные узлы соединяются со специальным устройством - многопортовым повторителем (хабом, концентратором) с помощью двух витых пар. Концентраторы 10Base-T можно соединять друг с другом, образовывая иерархию. Для обеспечения синхронизации станций и надежного распознавания коллизий число концентраторов между двумя любыми станциями ограничено 4 – «правило 4-х хабов».

Функционально сеть Ethernet на оптическом кабеле состоит из тех же элементов, что и сеть10Base-T - сетевых адаптеров, многопортовых повторителей и отрезков кабеля, соединяющих адаптер с повторителем.

Высокоскоростные варианты Ethernet

Коммутированная Ethernet. Эта технология предусматривает разбиение большой сети на меньшие сегменты с соответственно меньшим числом пользователей в каждом сегменте. Каждый коммутационный порт отвечает за фильтрацию трафика, передаваемого в подключенный к нему сегмент. Следует отметить, что коммутатор Ethernet хорош только в качестве временного решения, поскольку число его портов ограничено.

Дуплексная Ethernet – это коммутированная специализированная версия стандартной Ethernet, в которой каналы передачи со скоростью 10 Мбит/с можно формировать в двух направлениях, чтобы добиться суммарной пропускной способности 20 Мбит/с. При этом один из каналов служит для приема, а другой для передачи данных. Недостаток такой сети – ограничение по производительности, т.к. скорость, близкую к 20 Мбит/с, можно достичь только тогда, когда трафик сбалансирован в обоих направлениях. Данную технологию внедрила фирма Kalpana в конце 1993 года.

100BaseVG AnyLAN.. Предусматривается поддержка волоконно-оптических кабельных систем и экранированных витых пар. Используется другой метод доступа – обработка запросов по приоритету. Существует ощутимый недостаток совместимости с существующими сетями Ethernet. Данная технология, несмотря на множество хороших технических решений, не нашла большого количества сторонников и уступает по популярности Fast Ethernet.

Отличия технологии Fast Ethernet от Ethernet проявляются на физическом уровне. В стандарте Fast Ethernet определены три спецификации физического уровня: 100Base-TX для двух пар UTP категории 5 или 2-х пар STP типа 1, 100Base-FX для многомодового волоконно-оптического кабеля с двумя оптическими волокнами и 100Base-T4, работающую на 4-х парах UTP категории 3, но использующую 3 пары для передачи данных, а четвертую - для обнаружения коллизии.

Технология *Gigabit Ethernet* – поддерживает скорость обмена до 1 Гбит/с и добавляет новую ступень иерархии семейства Ethernet. Первый стандарт утвержден в 1998 г. Разработчики сохранили большую степень преемственности с технологиями Fast Ethernet и Ethernet. Gigabit Ethernet использует те же форматы кадров, работает в дуплексном и полудуплесном режимах, использует метод доступа CSMA/CD с минимальными изменениями. Типы физической среды: одномодовый и могомодовые волоконно-оптические кабели и двойной коаксиальный кабель с волновым сопротивлением 75 Ом.

3.3. Сеть Token Ring

Крупные предприятия имеют сети смешанной структуры. Доминирует среди них Ethernet, но присутствует и большое количество сетей Token Ring (маркерное кольцо).

Этот стандарт разработан фирмой IBM в 1984 году. В качестве передающей среды применяется неэкранированная или экранированная витая пара (UTP или STP) или оптоволокно. Скорость передачи данных 4 Мбит/с или 16

Мбит/с. В качестве метода управления доступом станций к передающей среде используется метод маркерное кольцо (Token Ring). Основные положения этого метода:

1) устройства подключаются к сети по топологии кольцо;
2) все устройства, подключенные к сети, могут передавать данные, только получив разрешение на передачу (маркер);
3) в любой момент времени только одна станция в сети обладает таким правом.

На базе IBM Token Ring в 1985 году комитетом IEEE 802 был разработан стандарт IEEE 802.5. Этот стандарт в отличие от IBM Token Ring не определяет среду передачи данных и топологии подключения устройств.

Типы пакетов

В IBM Token Ring используются три основных типа пакетов (Рис.3.4).

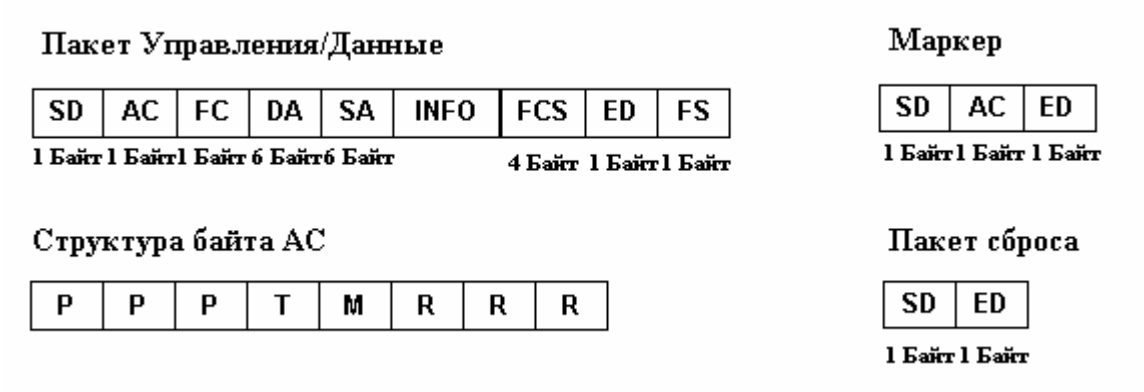


Рис. 3.4. Типы пакетов, используемые в IBM Token Ring

Пакет Управление/Данные (Data/Command Frame)**.** С помощью такого пакета выполняется передача данных или команд управления работой сети. (Каким именно является пакет определяется содержимым поля управления "FC").

Маркер (Token)**.** Станция может начать передачу данных только после получения такого пакета. В одном кольце может быть только один маркер и, соответственно, только одна станция с правом передачи данных.

Пакет Сброса (Abort)**.** Посылка такого пакета вызывает прекращение любых передач.

Функции полей:

SD (Start Delimiter) начальный ограничитель пакета. Синхронизирует работу приемника и передатчика, подготавливает станцию к приему пакета.

AC (Access Control) поле управления доступом. Содержит три бита приоритета (P), бит маркера (T), бит монитора (M), три бита приоритетного резервирования (R). Биты приоритета позволяют передавать срочные данные в первую очередь.

FC (Frame Control) - поле управления пакета. Если пакет - пакет управления, то в этом поле содержится команда управления.

DA (Destination Address) - адрес приемника.

SA (Source Address) - адрес источника.

INFO поле данных. Максимальная длина этого поля зависит от загрузки

сети и может меняться в пределах от 1 до 8 Кбайт.

FCS (Frame Check Sequence) - контрольная последовательность пакета. Это поле позволяет проверять достоверность принятых данных. Представляет собой циклический код обнаружения ошибок (CRC), вычисленный для полей FC, DA, SA, INFO.

ED (End Delimiter) - конечный ограничитель пакета.

FS (Frame Status) поле статуса пакета.

Взаимодействие станций в сети

Станция, которая приняла маркер, получает право на передачу, и может передавать данные. Для этого станция удаляет маркер из кольца, формирует пакет данных и передает его следующей станции. В сети Token Ring все станции принимают и ретранслируют все пакеты, проходящие по кольцу. При приеме станция сравнивает поле адреса приемника пакета (DA) с собственным адресом. Если адреса не совпадают, то пакет передается далее по кольцу без изменений. Если адреса совпадают, или принят пакет с широковещательным адресом, то содержимое пакета копируется, а по результатам приема данных вносятся изменения в поле статуса пакета (FS). Затем пакет передается далее по сети и таким образом возвращается на станцию - отправитель. Получив пакет, станция-отправитель проверяет поле статуса пакета (FS). Если при приеме пакета станцией-приемником была обнаружена ошибка (поле "C" равно "0"), выполняется повторная передача пакета. Если ошибок при приеме не было ("C" равно "1") или станция-приемник в данный момент не работает ("A" равно "0"), станция - отправитель удаляет пакет из кольца, формирует маркер и передает его следующей станции. Таким образом, следующая станция получит право на передачу данных. Такой алгоритм доступа применяется в сетях Token Ring со скоростью передачи 4 Мбит/с, описанный в стандарте 802.5.

В сетях Token Ring 16 Мбит/с используется алгоритм раннего освобождения маркера, в котором маркер передается следующей станции сразу после передачи последнего бита кадра, не дожидаясь возвращения по кольцу этого кадра с подтверждением приема. В этом случае пропускная способность используется более эффективно, так как по кольцу одновременно циркулируют кадры различных станций.

Логически сеть Token Ring представляет собой кольцо, а физически – звезду, в которой устройства подключаются к сети через специальный концентратор - MAU (Multistation Access Unit) или MSAU (Multi-Station Unit) – устройство многостанционного доступа.

Технология позволяет использовать различные типы кабелей: STP Type1, UTP Type 3 и 6, а также волоконно-оптический кабель. Максимальное расстояние от станции до MSAU – 100 м для STP и 45 м для UTP. Возможно также расширение сети с помощью мостов и повторителей. Сеть Token Ring может включать 260 узлов.

Максимальная длина кольца Token Ring составляет 4000 м. Ограничения на длину кольца и количество станций не являются такими жесткими, как в технологии Ethernet. Здесь ограничения связаны со временем оборота станции по кольцу.

Сеть Token Ring может строиться на основе нескольких колец, разделенных мостами. Недавно компания IBM предложила новый вариант технологии Token Ring, названный High-Speed Token Ring, HSTR. Эта технология поддерживает скорости в 100 и 155 Мбит/с, сохраняя основные особенности технологии Token Ring 16 Мбит/с.

3.4. Распределенный волоконно-оптический интерфейс передачи данных (FDDI)

На сегодняшний день самым быстрым и не требующим больших затрат решением для повышения производительности сети продолжает оставаться распределенный волоконно-оптический интерфейс передачи данных (FDDI), предложенный ANSI в 1988 г. Это первая технология локальных сетей, в которой средой передачи является волоконно-оптический кабель. FDDI обеспечивает передачу данных со скоростью 100 Мбит/с по двойному волоконно-оптическому кольцу на расстояние до 100 км.

Стандарт FDDI определяет кольцевую топологию с маркерным доступом, способную охватить большую площадь. Этот стандарт обеспечивает совместимость с сетями Token Ring, поскольку форматы кадров у них одинаковы.

Стандарт FDDI определяет перечень компонентов сети, который включает однократно подключенную станцию (SAS – Single Attached Station), двукратноподключенную станцию (DAS – Dual Attached Station) и концентраторы проводных линий. Соединения SAS с концентраторами имеют топологию звезды.

Интерфейс двукратного подключения обеспечивает отказоустойчивость системы благодаря своей избыточности. В случае разрыва кабеля сеть выполняет “заворачивание” – включает второе кольцо для обхода отказавшей станции. Сеть продолжает работать, но ее производительность падает.

Существуют также версии FDDI на медных проводах. «Медные» варианты FDDI не являются как таковыми стандартами и не могут гарантировать совместимость. Второе ограничение в “медных” версиях и даже в самом FDDI заключается в том, что здесь используется иная топология(двойное кольцо) и и другой размер пакета, нежели например в Ethernet.

Максимальное количество станций двойного подключения в кольце – 500. Максимальные расстояния между соседними узлами для многомодового кабеля – 2 км, для витой пары UTP категории 5 – 100 м, а для многомодового волокна зависит от его качества.

3.5. Беспроводные сети

Среди новых возможностей объединения сетей, которые в последнее время стали доступны сетевым администраторам, в первую очередь следует назвать беспроводную технологию. Беспроводные сегменты ЛВС могут оказаться особенно полезными в тех средах, где кабельную систему проложить трудно. Хотя эта технология пока находится на начальном этапе своего развития и не

свободна от изъянов (например, узкая полоса канала передачи), она очень быстро развивается.

Сегодня в беспроводных сетях в основном используются три технологии передачи:

- передача в инфракрасном диапазоне,
- передача данных с помощью радиосигналов с распределенным спектром,
- передача с помощью радиосигналов с узкополосным спектром.

Беспроводные инфракрасные ЛВС по своему быстродействию не уступают проводным ЛВС, поскольку скорость передачи – до 16 Мбит/с, обычной для сетей Token Ring. Кроме того, они обеспечивают более высокую, по сравнению с радио сигналами безопасность (перехватить передачу в инфракрасном диапазоне гораздо труднее).

Главный недостаток этой технологии в том, что она требует прямой видимости между передатчиками и приемниками. Если такие сети монтируются вне помещений, то в плохую погоду, вследствие возрастания поглощения и рассеивания инфракрасного излучения в атмосфере, при передаче сигнала возможны прерывания. Наконец, инфракрасные ЛВС могут передавать данные только на расстояния до 30 метров.

Радиочастотные ЛВС с распределенным по спектру сигналом имеет два варианта реализации:

- метод множественного доступа с кодовым разделением каналов с прямой последовательностью заключается в том, что пользовательское сообщение модулируется псевдослучайной кодовой последовательностью, т.е. формируется широкополосный сигнал. Несущая модулируется закодированной последовательностью, и полученный шумоподобный сигнал передается одновременно на нескольких частотах в пределах используемого диапазона. Приемник выделяет сообщение каждого пользователя из шума, отыскивая собственную «подпись» конкретного псевдослучайного кодового элемента сигнала.
- метод множественного доступа с кодовым разделением каналов со скачущей частотой, при котором диапазон разделен на большое число узких частотных каналов и передатчик постоянно «скачет» с одной частоты на другую; приемное устройство изменяет частоты в том же порядке и учитывает время пребывания на каждой частоте.

Достоинства данной технологии: высокая помехоустойчивость, не требуется прямая видимость. К недостаткам можно отнести ограниченную дальность (примерно 250 м), хотя она обеспечивает большую дальность, по сравнению с инфракрасной технологией, возможную электромагнитную несовместимость рядом расположенных ЛВС, не нужна лицензия FCC, низкую скорость передачи (около 2 Мбит/с).

В радиочастотные ЛВС с узкополосной передачей используется выделенная лицензируемая полоса частот в диапазоне 18-19ГГц. Сигнал не проникает

сквозь металлические и бетонные стены внутри зданий, но за счет высокой частоты позволяет охватывать площадь в 465 м^2.

В процессе перехода предприятий на сети, охватывающие все их структуры, беспроводная технология будет играть все более значительную роль. Ожидается распространение гибридов беспроводных и проводных сетей, в которых беспроводные сегменты обеспечивают выполнение сетевых функций, где прокладка кабеле затруднена. Следует отметить, что беспроводная технология более важна в глобальных сетях, которые по внедрению этой технологии значительно опережают локальные сети.

Глава 4. Построение локальных сетей по стандартам физического и канального и сетевого уровней

4.1. Иерархия кабельной системы

Кабельная система составляет фундамент любой компьютерной сети. От ее качества зависят все основные свойства сети. От ее качества зависят все основные свойства сети.

Структурированная кабельная система – это набор коммутационных элементов (кабелей, разъемов, коннекторов, кроссовых панелей и шкафов), которые удовлетворяют стандартам, а также методика их совместного использования, позволяющая создавать регулярные, легко расширяемые структуры связей в вычислительных сетях.

При ее построении подразумевается, что каждое рабочее место на предприятии должно быть оснащено розетками для подключения телефона, компьютера, даже если в данный момент этого не требуется. Т.е. система строится избыточной, чтобы в будущем легко можно было ее конфигурировать.

Структурированная кабельная система планируется и строится иерархически, с главной магистралью и многочисленными ответвлениями. Она может быть построена на базе существующих современных телефонных кабельных систем. В нашей стране далеко не во всех зданиях телефонные линии прокладываются витыми парами, поэтому они не пригодны для создания компьютерных сетей, и кабельную систему приходится строить заново.

Типичная иерархическая структура структурированной кабельной системы (рис. 4.1) включает:

- горизонтальные подсистемы (в пределах этажа);
- вертикальные подсистемы (внутри здания);
- подсистему кампуса (в пределах одной территории с несколькими зданиями).

Горизонтальная подсистема соединяет кроссовый шкаф этажа с розетками пользователей. Подсистемы этого типа соответствуют этажам здания. Вертикальная подсистема соединяет кроссовые шкафы каждого этажа с центральной аппаратной здания. Подсистема кампуса соединяет несколько зданий с главной аппаратной всего кампуса. Эта часть кабельной системы называется магистралью.

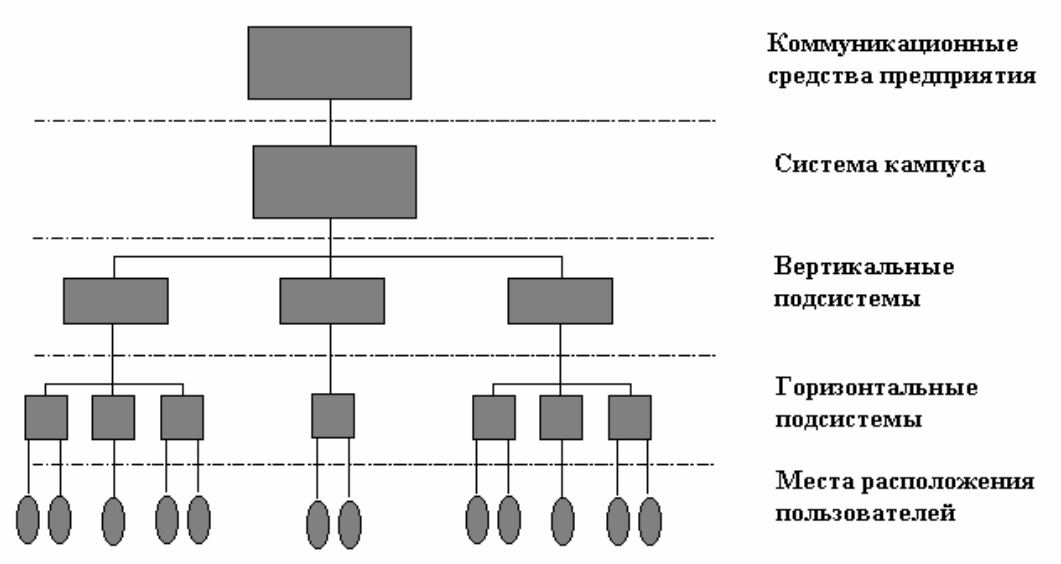


Рис. 4.1. Иерархия структурированной кабельной системы

Использование структурированной кабельной системы дает следующие преимущества:

- Универсальность. Единая среда передачи данных позволяет автоматизировать многие процессы контроля, мониторинга и управления хозяйственными службами и системами жизнеобеспечения предприятия.
- Увеличение срока службы. Срок морального старения хорошо структурированной кабельной системы может составлять 10-15 лет.
- Уменьшение стоимости добавления новых пользователей и изменения их мест размещения. Более выгодно произвести прокладку кабеля один раз с большим запасом по длине, чем несколько раз проводить прокладку кабеля.
- Возможность легкого расширения сети. Кабельная система является основой для деления сети на легко управляемые логические сегменты, т.к. сама уже разделена на физические.
- Обеспечение более эффективного обслуживания. Отказ одного сегмента, как правило, не действует на другие, т.к. объединение осуществляется на основе концентраторов.
- Надежность. Производительность системы гарантирует не только качество, но и совместимость отдельных компонентов.

4.2. Выбор кабеля

Большинство проектировщиков начинают разработку структурированной кабельной системы с горизонтальных подсистем, т.к. к ним подключаются конечные пользователи.

Горизонтальная подсистема характеризуется большим количеством ответвлений кабеля. На этаже обычно устанавливается кроссовая панель, соединяющая пользовательское оборудование и концентраторы/коммутаторы.

Медный провод является предпочтительнее. Если нужна очень высокая производительность системы, прокладывается волоконно-оптический кабель. Стоимость такой установки очень высока.

Коаксиальный кабель - устаревшая технология, которую нужно избегать, если только она широко не используется на предприятии. Беспроводная связь является новой и многообещающей технологией, однако из-за новизны и низкой помехоустойчивости лучше ограничить масштабы ее использования.

При выборе кабеля учитывают следующие характеристики: полоса пропускания, расстояние, физическая защищенность, электромагнитная помехозащищенность, стоимость. Необходимо учитывать особенности имеющейся на предприятии системы и тенденции рынка на данный момент.

STP позволяет передавать данные на большие расстояния, чем UTP, но является более дорогой. На практике UTP используется чаще, т.к. может передавать еще и голос. Преобладающим кабелем для горизонтальной подсистемы является неэкранированная витая пара категории 5.

Вертикальная система состоит из более протяженных отрезков, имеет меньшее количество ответвлений.

В прошлом для вертикальных подсистем использовался коаксиальный кабель. Теперь чаще используется оптоволокно. В настоящее время выбор ограничивается тремя вариантами:

- Волоконно-оптический кабель;
- Толстый коаксиальный кабель;
- Широкополосный кабель, используемый в кабельном телевидении.

Для подсистемы кампуса характерна нерегулярная структура связей с центральным зданием. Предпочтительный тип кабеля – оптоволокно в специальной изоляции

4.3. Концентраторы и сетевые адаптеры

Сетевой адаптер (Network Interface Card, NIC) вместе со своим драйвером реализует канальный уровень модели OSI в конечном узле сети, выполняя функции передачи и приема кадра.

От производительности сетевых адаптеров зависит производительность любой сложной сети. Сетевые адаптеры характеризуются типом поддерживаемого протокола, производительностью, шиной компьютера, к которому они могут присоединяться, типом приемопередатчика, наличием собственного процессора.

В качестве примера классификации адаптеров используется подход фирмы 3Com, лидера в области адаптеров сети Ethernet.

Адаптеры первого поколения выполнены на логических микросхемах. Они обладали низкой надежностью, имели буферную память на один кадр. Задание конфигурации адаптера проходило вручную, с помощью перемычек. Для каждого типа адаптеров использовался свой драйвер.

В сетевых адаптерах второго поколения использовался метод многокадровой буферизации, использовались микросхемы с высокой степенью интеграции, что повысило надежность. Драйверы этих адаптеров основаны на стандартных спецификациях.

В сетевых адаптерах третьего поколения (примером служит EtherLink III) осуществляется конвейерная схема обработки кадра, в которой прием кадра из оперативной памяти компьютера и передачи его в сеть совмещаются по времени. Адаптеры базируются на специализированных интегральных схемах (ASIC), что повышает их производительность и надежность при снижении стоимости.

Современные адаптеры можно отнести к четвертому поколению. Они базируются на ASIC, выполняющих функции MAC-уровня и большое количество высокоуровневых функций. В серверных вариантах адаптеров почти обязательное наличие мощного процессора. Примером такого адаптера является 3Com Fast EtherLink XL10/100.

Концентраторы, кроме основной функции протокола (побитного повторения кадра на всех или последующем порту, в соответствии с алгоритмом, определенным стандартом), всегда выполняют ряд полезных дополнительных функций, определяемых производителем, которые либо в стандарте вообще не оговорены, либо являются факультативными.

Рассмотрим особенности реализации основной функции концентраторов на примере концентраторов Ethernet. В данной технологии, устройство, соединяющее несколько сегментов коаксиала, называлось повторителем. Обычными были двухпортовые повторители.

С появлением спецификации 10Base-T появились многопортовые повторители – концентраторы (или хабы). Они имеют от 8 до 72 портов, причем основная часть из них – для подключения на витой паре. Многопортовый повторитель-концентратор Ethernet может по-разному рассматриваться при использовании правила 4-х хабов. В большинстве моделей все порты связаны с единственным блоком повторения, и при прохождении сигнала между двумя портами повторителя вносят задержку всего один раз.

Различия при выполнении основной функции концентраторов невелики, по сравнению с разбросом их возможностей при реализации дополнительных функций.

Способность концентратора отключать некорректно работающие порты называется *автосегментацией*. Для концентраторов FDDI эта функция является основной, так как определена в протоколе. Концентраторы Ethernet и Fast Ethernet выполняют отключение порта в ситуациях:

- Ошибки на уровне кадра. Если интенсивность прохождения через порт кадров с ошибкой, превышает заданный порог, порт отключается.
- Множественные коллизии. Если данный порт стал виновником коллизии 60 раз подряд.
- Затянувшаяся передача. Если время прохождения одного кадра превышает время передачи кадра максимальной длины в три раза.

Свойство резервных связей концентраторов позволяет соединять отключенные порты, чтобы не нарушать логику работы сети. При конфигурации концентратора администратор должен определить, какие порты являются основными, а какие – резервными.

В число дополнительных функций входят функции *защиты сети от несанкционированного доступа*, запрещающие подключение к концентратору компьютеров с неизвестными MAC-адресами, а также заполняющие нулями поля данных кадров, поступающих не к станции назначения.

На конструктивное устройство концентраторов большое влияние оказывает область их применения. Концентраторы рабочих групп чаще всего выпускаются как устройства с фиксированным количеством портов, корпоративные концентраторы – как модульные устройства. Первые представляют собой наиболее простое конструктивное исполнение, когда устройство представляет собой отдельный корпус со всеми необходимыми элементами, которые нельзя изменять. Вторые выполняются в виде отдельных модулей с фиксированным количеством портов, устанавливаемых на общее шасси. Шасси имеет внутреннюю шину для объединения отдельных модулей в единый повторитель. Модульные концентраторы позволяют подобрать конфигурацию концентратора и более гибко реагируют на изменение конфигурации сети.

Стековые концентраторы сочетают преимущества модульных концентраторов и концентраторов с фиксированным количеством портов.

Сложные концентраторы, выполняющие дополнительные функции, обычно доступны централизованному управлению по сети по протоколу SNMP.

4.4. Логическая структуризация сети с помощью мостов и коммутаторов

Сеть, разделенная на логические сегменты, обладает большей производительностью и надежностью. Логическая структуризация сети необходима при построении сетей средних и крупных размеров. Использование общей разделяемой среды приемлемо только для сети, состоящей из 5 – 10 компьютеров. Устройства логической структуризации сетей мосты и коммутаторы работают на канальном уровне стеков протоколов. Структуризация сети возможна также на основе маршрутизаторов, которые используют протоколы сетевого уровня. В современных сетях часто используют комбинированные способы структуризации – небольшие сегменты объединяются устройствами канального уровня, а крупные подсети – маршрутизаторами.

Мост и коммутатор – функциональные близнецы. Оба устройства продвигают кадры на основании одних и тех же алгоритмов: алгоритм прозрачного моста, описанного в стандарте IEEE 802.1D, либо алгоритма моста с маршрутизацией от источника компании IBM. Отличие моста и коммутатора в том, что мост обрабатывает кадры последовательно, а коммутатор параллельно.

Коммутаторы – наиболее быстродействующие из современных коммуникационных устройств. Они позволяют соединять высокоскоростные сегменты без блокирования (уменьшения пропускной способности) межсегментного тра-

фика. Пассивный способ построения адресной таблицы коммутатора – с помощью слежения за проходящим трафиком – приводит к невозможности работы в сетях с петлевидными связями. Другим недостатком сетей на основе коммутаторов является отсутствие защиты от широко вещательного шторма, который они обязаны передавать в соответствии с режимом работы.

Постепенно коммутаторы вытеснили из ЛВС классические однопортовые мосты. Процесс вытеснения мостов начался с 1994 года, и сегодня локальные мосты практически не производятся. Коммутаторы вобрали в себя дополнительные функции: поддержка виртуальных сетей, приоритезация трафика, использование магистрального порта.

Мосты работают сегодня на достаточно медленных глобальных связях между удаленными ЛВС.

4. 5. Принципы работы мостов

Описывая алгоритмы работы мостов и коммутаторов, будем пользоваться термином мосты, т. к. именно для них данные алгоритмы были стандартизированы**.**

Мосты функционируют на подуровне управления доступом к среде передачи канального уровня модели OSI. Они не зависят от типа используемых протоколов. Обычно их не касается, какой из транспортных протоколов используется – IPX или TCP/IP. При проектировании сетей мосты являются необходимыми элементами, с их помощью обеспечивается повышение эффективности, безопасности и дальности.

Чаще всего мосты устанавливаются в целях повышения эффективности. Администратор сети может воспользоваться мостом для уменьшения перегрузки и повышения быстродействия: большая сеть делится на несколько подсетей, соединенные мостами. Две небольшие сети будут работать быстрее, поскольку трафик локализуется в пределах подсети, кроме этого они более просты в управлении и обслуживании. Использование мостов приводит к повышению эффективности работы сети еще и потому, что разработчик может использовать разные топологии среды передачи, а затем соединить эти сети посредством мостов. Поскольку комитет IEEE 802 разработал для различных сетевых архитектур общий уровень управления логическим каналом, то существует возможность объединения, например, двух сетей Token Ring, разделенных ЛВС Ethernet. ЛВС Ethernet может пересылать пакеты так же, как почтальон может доставлять письма, написанные на иностранном языке, если конверты (пакеты) оформлены в соответствии со стандартом.

Мосты часто используют в целях повышения безопасности, т.к. программирую мосты на передачу только тех пакетов, которые содержат определенные адреса отправителя и получателя, можно ограничить круг рабочих станций, которые могут посылать и принимать информацию из другой подсети, тем самым, предотвращая несанкционированный доступ. Здесь часто используют интеллектуальные мосты.

Мосты можно также использовать и в целях повышения отказоустойчивости системы, если с помощью внутренних мостов связать два файловых сервера, которые постоянно будут подстраховывать друг друга, причем при этом снизится уровень трафика. Мосты увеличивают дальность охвата сети. Поскольку мост ретранслирует пакет в широковещательном режиме, то он функционирует как повторитель.

Алгоритм работы прозрачного моста

Мост самого простого типа анализирует 48-битовое поле адреса пункта назначения пакета и сравнивает этот адрес с таблицей, в которой указаны адреса всех рабочих станций данного сегмента сети. Если адрес не соответствует ни одному из указанных в таблице, мост передает пакет в следующий сегмент. Так продолжается до тех пор, пока не достигнут сегмента сети, содержащего компьютер с указанным адресом. Такие мосты называются *прозрачными мостами*. Этот метод используется во всех Ethernet-мостах и в некоторых мостах в сетях Token Ring.

Прозрачный мост составляет адресную таблицу. Поля адресной таблицы могут быть динамическими, создаваемыми в процессе самообучения моста, и статическими, создаваемыми администратором сети вручную.

Отличие работы моста от повторителя состоит в том, что он передает кадры не побитно, а с буферизацией. Когда мост передает кадр с сегмента на сегмент, он заново пытается получить доступ к сегменту, в котором находится адресат, по правилам доступа, например в сети Ethernet – CSMA/CD, и выполняет операцию продвижения (forwarding) кадра. Если компьютеры принадлежат к одному сегменту, то кадр удаляется из буфера и работа с ним прекращается. Такая операция называется фильтрацией (filtering).

Если адрес назначения на данный момент неизвестен, то мост передает кадр на все порты, кроме порта – источника кадра. Мост постоянно находится в состоянии обучения.

Мосты и алгоритм остовного дерева

Алгоритм остовного (связывающего) дерева (STA - Spanning Tree Algorithm) был разработан фирмами DEC и Vitalink, а в последствии принят комитетом IEEE 802.1 как стандарт. Этот алгоритм применяется для объединения мостами нескольких Ethernet сетей при возможности существования более одного непустого пути.

В системе STA каждому мосту присваивается идентификатор, состоящий из поля приоритета и глобально назначаемого адреса станции. Мосты взаимодействуют между собой и определяют маршрут, по которому должны передаваться данные. В ходе переговоров решается вопрос о назначении корневого (главного) моста. Им становится мост с наивысшим приоритетом. Если таких мостов два, корневым становит тот, у которого выше адрес станции. Этот про-

цесс выполняется автоматически, но администратор может присвоить наивысшее значение приоритета определенному мосту.

После выбора корневого моста каждый мост определяет, какой из его портов осуществляет связь в направлении корневого моста, и обозначает его как корневой порт. Если к одной ЛВС подключено два и более мостов, то выбирается тот, который имеет наименьшую «стоимость» (согласно критериям, установленным администратором сети). Стоимость может учитывать такие элементы, как быстродействие линии и емкость буфера. Если все стоимости одинаковые, то из выбранных мостовых портов создается древовидная структура, которая обеспечивает минимально возможное количество межмостовых переходов при передаче пакета из одной ЛВС в другую.

Далее каждый мост переводит в режим передачи свой корневой порт по направлению к корневому мосту и порт, который осуществляет связь по направлению от корневого моста. Остальные порты блокируются. Таким образом, в ЛВС работает только один мостовой порт в каждом направлении, причем выбирается самый эффективный маршрут.

Если какой-нибудь из портов прекратит свою работу, то заблокированный порт переводится в режим анализа, изучает пакеты идущие по сети, и обновляет свою таблицу адресов станций. После чего, он переходит в режим передачи и направляет в корневой мост уведомление об этом переходе. Корневой мост уведомляет об этом все мосты сети, и они корректируют свои таблицы адресов, включая в них новый порт моста.

Данный алгоритм является статическим: он предусматривает обновление пути только тогда, когда найденные маршруты становятся недоступными

Алгоритм с маршрутизацией от источника

В Token Ring и FDDI используется маршрутизация от источника - метод, при котором ответственность за разработку для кадра полного маршрута возлагается на рабочую станцию – источник. Станция-отправитель помещает в посылаемый в другое кольцо кадр всю адресную информацию о промежуточных мостах и кольцах. При получении каждого пакета мост просматривает поле маршрутной информации на наличие в нем своего идентификатора. Если он там присутствует, мост копирует кадр в указанное кольцо. Хотя в названии данного способа входит термин «маршрутизация», настоящей маршрутизации нет, т.к. мосты и станции по-прежнему работают MAC-информацию.

Преимуществами данных мостов, по сравнению, с прозрачными мостами являются: более рациональные маршруты, простота и дешевизна, более высокая скорость. Недостатки: более дорогие сетевые адаптеры, принимающие участие в маршрутизации; непрозрачность сети; увеличение трафика за счет широковещательных пакетов.

Существуют также *интеллектуальные мосты*, которые можно програм-

мировать на фильтрацию пакетов по определенным критериям.

Часто мосты *каскадируют,* соединяя ЛВС последовательно.

4.6. Сетевые операционные системы

Сетевые операционные системы выполняют функции уровней, начиная с сетевого и выше, согласно модели OSI.

В общем случае сетевая операционная система (ОС), установленная на отдельном компьютере, имеет определенную структуру и состоит из следующих частей:

- любая сетевая операционная система должна иметь средства управления локальными ресурсами компьютера и выполнять функции локальных ОС: функции распределения оперативной памяти между процессами, планирования и диспетчеризация процессов, управления процессорами в мультипроцессорных машинах, управления периферийными устройствами и другие функции.
- сетевая ОС должна быть способной предоставлять собственные ресурсы и определенные услуги в общее пользование, то есть иметь серверную часть или сервер.
- сетевая ОС должна иметь клиентскую часть, или редиректор, которая обеспечивает доступ к удаленным ресурсам и услугам их использование.

В зависимости от задач, решаемых с помощью сетевого компьютера, на него устанавливается определенный набор модулей сетевой ОС. Сетевые модули ОС могут быть реализованы разными способами. В зависимости от распределения функций между компьютерами в сети можно выделить одноранговые сетевые ОС и сетевые ОС с выделенным сервером.

Одноранговые сетевые ОС используются для построения одноранговых сетей, где каждый компьютер может выполнять как функции клиента, так и функции сервера. К одноранговым ОС можно отнести практически все современные сетевые ОС для ЛВС. В них, как правило, реализованы базовые сетевые функции для обеспечения сетевого взаимодействия. Одноранговые сетевые ОС просты при инсталляции и эксплуатации. С другой стороны, они обладают низкой производительностью, имеют ограниченные возможности по обеспечению связи удаленных сегментов сети. Они не обладают развитыми средствами управления сетью, не обеспечивают распределенный режим работы «клиент-сервер».

При построении сложных сетей, как правило, один или несколько компьютеров выделяют для выполнения отдельных сетевых функций. Устанавливаемые в этом случае сетевые ОС – операционные системы с выделенным сервером. Признанными лидерами сетевых ОС с выделенным сервером являются Windows NT и Novell Netware.

Основное направление развития современных сетевых операционных систем перенос вычислительных операций на рабочие станции, создание систем с распределенной обработкой данных. Это в первую очередь связано с рос-

том вычислительных возможностей персональных компьютеров и все более активным внедрением мощных многозадачных операционных систем: OS/2, Windows NT, Windows 98. Кроме этого внедрение объектно-ориентированных технологий (OLE, OCE, IDAPI) позволяет упростить организацию распределенной обработки данных. В такой ситуации основной задачей ОС становится объединение неравноценных операционных систем рабочих станций и обеспечение транспортного уровня для широкого круга задач: обработка баз данных, передача сообщений, управление распределенными ресурсами сети (directory/name service). В сетевых операционных системах применяют три основных подхода к организации управления ресурсами сети.

Первый подход - это таблицы объектов. Используется в сетевых операционных системах NetWare 286 и NetWare v3.lx. Такая таблица находится на каждом файловом сервере сети. Она содержит информацию о пользователях, группах, их правах доступа к ресурсам сети (данным, сервисным услугам и т.п.). Такая организация работы удобна, если в сети только один сервер. В этом случае требуется определить и контролировать только одну информационную базу. При расширении сети, добавлении новых серверов объем задач по управлению ресурсами сети резко возрастает. Администратор системы вынужден на каждом сервере сети определять и контролировать работу пользователей. Абоненты сети, в свою очередь, должны точно знать, где расположены те или иные ресурсы сети, а для получения доступа к этим ресурсам - регистрироваться на выбранном сервере.

Второй подход используется в LAN Server и LAN Manager - структура доменов. Все ресурсы сети и пользователи объединены в группы. Домен можно рассматривать как аналог таблиц объектов, только здесь такая таблица является общей для нескольких серверов, при этом ресурсы серверов являются общими для всего домена. Пользователю для получения доступа к сети, достаточно подключиться к домену (зарегистрироваться), после этого ему становятся доступны все ресурсы домена, ресурсы всех серверов и устройств, входящих в состав домена. Однако и с использованием этого подхода также возникают проблемы при построении информационной системы с большим количеством пользователей, серверов и, соответственно, доменов.

Третий подход - служба наименований директорий или каталогов. Все ресурсы сети: сетевая печать, хранение данных, пользователи, серверы и т.п. рассматриваются как отдельные ветви или директории информационной системы. Таблицы, определяющие DNS, находятся на каждом сервере. Это, во-первых, повышает надежность и живучесть системы, а во-вторых, упрощает обращение пользователя к ресурсам сети. Зарегистрировавшись на одном сервере, пользователю становятся доступны все ресурсы сети. Управление такой системой также проще, чем при использовании доменов, так как здесь существует одна таблица, определяющая все ресурсы сети, в то время как при доменной организации необходимо определять ресурсы, пользователей, их права доступа для каждого домена отдельно.

Глава 5. Глобальные сети

Совокупности вычислительных машин, объединенных коммуникационной средой, охватывающей значительные по расстоянию территории, получили название *глобальных компьютерных сетей*.

Лидером глобальных сетей является Интернет. Его основу составляют высокоскоростные магистральные сети. Независимые сети подключаются к магистральной сети через точки сетевого доступа NAP (Network Access Point). Автономные сети могут образовывать компании, специализирующиеся на предоставлении услуг доступа в сети Интернет – провайдеры. Например, в России – компания Relcom.

Каждый компьютер в Интернет имеет уникальный IP-адрес. IP-адрес состоит из четырех байтов и записывается в виде четырех десятичных чисел, разделенных точками, например:
194.85.120.64

IP-адрес состоит из двух логических частей: номера сети и номера узла. В зависимости от количества байтов выделяют несколько классов IP-адресов.

В сети Интернет в основном используется семейство протоколов TCP/IP.

5.1. Протокол TCP/IP

Протокол TCP/IP предназначен для соединения сетей с разнородным оборудованием. Это единственное средство коммуникации, позволяющее связываться рабочим станциям всех типов – PC, Macintosh, Unix. Он необходим для выхода в Internet. Фактически TCP/IP является набором протоколов. Наиболее известные из них TCP и IP.

Протокол IP (Internet Protocol) функционирует на сетевом уровне, предоставляя различным станциям стандартный набор правил и спецификаций для межсетевой маршрутизации с помощью IP адресов.

Протокол управления передачей данных TCP (Transmission Control Protocol) работает на транспортном уровне модели OSI, обеспечивая прием сетевой и преобразование информации в требуемый на данном уровне формат. Таким образом, IP задает правила установления соединения и обеспечивает соединение портов компьютера, TCP отвечает за интерпретацию данных.

Протокол TCP устанавливает между двумя компьютерами *дуплексное соединение* типа «точка-точка». Программы на каждом конце соединения используют собственный порт. Комбинация IP-адреса и номера порта называется *сокет* (socket). Соединение устанавливается путем *трехкратного квитирования* (three-way handshake)

После установления соединения каждая программа может посылать другой программе поток байтов. Программа-отправитель не ожидает подтверждения каждого сегмента, а посылает несколько сегментов вместе и ждет первого подтверждения. Если программа-получатель должна отослать данные обрат-

но отправителю, она может совместить подтверждения и данные в одних и тех же сегментах. Номера последовательности программы-отправителя являются не индексами сегментов, а индексами в потоке байтов. Программа-получатель отсылает обратно номера последовательности (в поле номера подтверждения), удостоверяя тем самым, что все байты приняты и объединены в правильной последовательности. Программа-отправитель заново пересылает неподтвержденные сегменты.

Каждая программа со своей стороны закрывает TCP-соединение, что должно быть подтверждено программой на другой стороне соединения. Программа не может получать байты по соединению, которое закрыто на другой стороне.

5.2. Управление электронной почтой в сети масштаба предприятия

Для многих электронная почта – наиболее популярная сетевая служба. Если все сети предприятия пользуются одним пакетом программ электронной почты, то проблем с совместимостью нет. Базовым протоколом на протяжении нескольких лет является простой протокол пересылки почты (SMTP), входящий в комплект TCP/IP. SMTP состоит всего из 14 команд. Он эффективен, но его возможности ограничены. В то время, как электронная почта существенно усовершенствовалась, особенно для ЛВС, SMTP практически стоял на одном месте. Для организации взаимодействия систем электронной почты, работающих на разных сетевых платформах, комитет CCITT выпустил ряд стандартов X.400 на средства электронной почты, которые стали частью модели OSI. X.400 выпущен в 1984 году, а 1988 был опубликован комплект изменений, проходящий в настоящее время испытания. Изменения предусматривают возможность совместной передачи текстовых и нетекстовых элементов, например, электронных таблиц и графических файлов, аудио- и видеоинформации. Интерфейс пользователя сети с электронно-почтовой системой называется пользовательским агентом. Этот интерфейс позволяет сетевому пользователю отправлять и получать почту. Пользовательский агент строит “конверт” стандарта X.400 и помещает в соответствующие поля заголовка адресную информацию. Он ищет необходимые адреса и при необходимости создает списки рассылки. Затем пользовательский агент передает конверт с заголовками и сообщением агенту пересылки сообщений, который следует рассматривать как почтовые отделения, оказывающие типовой набор почтовых услуг при передаче электронных сообщений, включая передачу с промежуточным накоплением. Агент пересылки сообщений также выполняет контроль ошибок и обеспечивает правильность форматирования конверта и заголовков. Служба передачи сообщений предоставляет расширенные услуги по доставке, например доставку с проверкой или без проверки.

Комплект спецификаций X.400 включает протокол P1, описывающий конверт. Конверт X.400 имеет заголовок, который содержит сведения об отправителе, получателе, предмете сообщения и списки рассылки копий. Пользовательские агенты взаимодействуют с агентами пересылки сообщений посредст-

вом протокола P2, регламентирующего структуру сообщений и порядок доставки.

Для организации связи между пользователями сетей, работающих в разных пунктах на разных файловых серверах, требуется служба каталогов. CCITT разрабатывает стандарт на службы каталогов (X..500). Каталоги построены в виде информационного дерева каталогов. Корень этого дерева – чисто концептуальный объект. Под ним находятся страны, еще ниже – организации, а затем практически все внутрифирменные сведения. Компонентами дерева каталога управляют отдельные системные агенты каталога, которые могут быть распределенными или централизованными - это зависит от метода создания базы каталога.

5.3. Протокол передачи гипертекста (HTTP)

Протокол передачи гипертекста (HTTP) – стандартный протокол для передачи документов между серверами и броузерами в системе World Wide Web.

Клиент устанавливает соединение с сервером по указанному номеру порта. Если в качестве клиента выступает броузер, номер порта указывается в URL-запросе. Если номер не указан, по умолчанию указывается порт 80.

Команды HTTP-клиента принято называть методами.

В Интернете существует два типа поисковых систем – классификаторы и поисковые машины.

Классификаторы хранят упорядоченные списки ссылок на Web-узлы. Их удобно использовать для достаточно общей информации.

Поисковые машины просматривают страницы, размещенные в Интернете, и составляют индексы используемых слов. Основным элементом поисковых машин является индексатор. К элементам поисковой машины также относятся индекс и аппарат поиска.

В последнее время Web-серверы, предназначенные для поиска информации сочетают возможности классификаторов и поисковых машин.

Методы индексирования делятся на статические, теоретико-информационные и вероятностные.

В статических методах документы рассматриваются как точки информационного пространства. Чем ближе группы методов, составляющие документ, тем ближе находятся отображающие их точки. В качестве терминов индексации выбираются понижающие плотность пространства документов.

Теоретико-информационные методы основаны на предположении о том, что наибольшую информационную ценность имеют наименее часто встречающиеся слова. Для оценки полезности термина применяются концепции теории информации.

Вероятностные методы предполагают наличие обучающего множества документов для оценки релевантности результатов обработки запросов. Обучающее множество применяется для вычисления весовых коэффициентов, получаемых путем оценки условной вероятности вхождения термина в данный документ в случае его релевантности (или нерелевантности).

При построении индекса реальные документы заменяются поисковыми образцами. При индексировании нетекстовой информации в поисковые образцы входят главным образом универсальные адреса ресурсов.

Формальную релевантность вычисляет система, на основании чего ранжируется выборка найденных документов.

Реальная релевантность – оценка самим пользователем ценности найденных документов.

Некоторые поисковые машину показывают дату, когда был проиндексирован тот или иной документ, что позволяет пользователю понять, насколько актуальным является ресурс.

Часто поисковые машины не включают определенные слова в свои индексы или в запросы пользователей. Такими словами, например, являются предлоги.

5.4. Передача цифровых сигналов

В 1957 году компания Bell ввела в эксплуатацию первую магистраль Т-1, обеспечивающую высокоскоростную передачу речевых сигналов в цифровом виде. Сопряжение аппаратуры клиента с цифровой сетью телефонной компании осуществляется посредством устройства обслуживания канала и устройства обслуживания данных, которые функционируют как цифровые модемы, кроме того устройства обслуживания канала выполняют некоторые функции по согласованию линии и диагностике. Магистраль Т-1 содержит 24 канала, каждый из которых может обрабатывать 64000 битов в секунду. Еще 64000 битов необходимо для контроля ошибок, поэтому для одной линии Е-1 требуется полоса пропускания 1,544 Мбит/с:

64000 бит/с на канал * 24 канала = 1,536 Мбит/с

8 бит/с на выборку * 8000 выборок в секунду = 64000 бит/с

Сигнал, передающий с такой скоростью, в промышленности связи обозначают как DS-1 (цифровой сигнал, уровень1). Существуют и другие сигналы:

Сигнал	*Скорость передачи (Мбит/с)*	*Число каналов*
DS-1	1,544	1
DS-2	6,312	4
DS-3	45	28
DS-4	274	168

Широкую популярность завоевали каналы Т-1 и Т-3. Их популярность обусловлена следующим:

передача в цифровом виде позволяет получить более высококачественные сигналы, чем передача в аналоговом виде;

многие мультиплексоры Т-1 могут осуществлять автоматическую альтернативную маршрутизацию, если основной путь недоступен;
полосу пропускания можно распределять разными способами в зависимости от конкретных условий передачи речи и данных;
мультиплексоры Т-1 имеют, как правило, синхронный и асинхронный интерфейсы передачи данных;
с помощью установки в мультиплексоры специальных адаптеров можно быстро увеличить количество линий.

Следует отметить, что европейская система отличается от североамериканской. В Европе DS-1 состоит из 32 каналов, каждый из которых обеспечивает скорость передачи 64 Кбит/с, и имеет общую полосу пропускания 2,048 М бит/с.

Недавним нововведением в области передачи по каналам Т-1 стал так называемый дробный сервис Т-1. Пользователи могут покупать ту часть канала, которая им нужна.

Иногда WAN состоит из нескольких ЛВС, соединенных удаленными мостами, что позволяет им функционировать как единая глобальная сеть независимо от сетевых протоколов верхних уровней (TCP\IP, SPX/IPX и т.п.).

Комитет IEEE 802.1G работает над созданием стандарта на удаленные мосты. Он решил использовать для удаленных мостов тот же алгоритм остовного дерева, который принят в качестве стандарта для мостов в ЛВС. Но в глобальных сетях этот метод работает неэффективно, поскольку выбирается только один путь, а остальные переводятся в резервный режим. Некоторые фирмы при решении этой проблемы добиваются, чтобы резервные линии не только повышали отказоустойчивость, но и обеспечивали распределение нагрузки по WAN.

5.5. Сети с коммутацией пакетов (X.25)

Эти сети в масштабах страны и всего мира обеспечивают коммутацию данных в виде пакетов, соответствующих требованиям X.25 CCITT, рекомендация которой построена на первых трех уровнях модели OSI. Он определяет принципы организации связи между компьютерами и терминалами с одной стороны и узлами коммутации пакетов с другой стороны. Он устанавливает порядок взаимодействия между оконечным оборудованием данных (DTE) и оборудованием для передачи данных (DCE) посредством узлов коммутации пакетов. Как правило, DTE подключается к устройству сборки-разборки пакетов (PAD), которое обеспечивает преобразование протоколов: собственный протокол потока данных преобразует в протокол X.25. В пункте назначение выполняется обратное преобразование. PAD способствует повышению скорости передачи X.25, концентрируя потоки данных из нескольких устройств DTE. PAD помещает данные, которые принимает, в пакет, содержащий управляющую информацию для контроля ошибок и упорядочения. Если станция, принимающая пакет, выявляет ошибку, она передает запрос на повторную передачу. Маршрут, по которому следует пакет, определяется коммутационным оборудовани-

ем, причем два пакета по одному маршруту идти не могут. Существует большая вероятность того, что пакеты прибудут в пункт назначения в беспорядке, и их нужно будет упорядочивать. Все эти операции выполняются прозрачно для пользователя, поэтому сети с коммутацией пакетов иногда называют “облаками”.

Поскольку это международный стандарт, сети X.25 используются во многих крупных корпорациях. Сеть с коммутацией пакетов в США может обмениваться пакетами с сетью в ЮАР, Англии или Японии, используя шлюз X.75.

Различными фирмами обеспечивается широкий диапазон скоростей передачи: ОТ 110 бит/с до 64 Кбит/с.

5.6. Цифровая сеть с интеграцией обслуживания

Цифровая сеть с интеграцией обслуживания (ISDN) – это находящаяся в стадии разработки совокупность международных стандартов, регламентирующих соединение аппаратуры передачи речи, данных и видеосигналов. Пользователи могут одновременно вести разговор по телефону, просматривать видеоизображения или другую информацию на компьютере. Все эти виды информации могут “путешествовать” в одном ISDN-пакете и направляться на терминал интегрированной обработки речи, изображений, данных. Кроме того, пользователь может выбирать, к какой сети подключаться: к частной сети передачи речевой информации или к общедоступной сети передачи данных.

ISDN-интерфейсы между местными телефонными сетями и конечными пользователями могут заменить многие каналы, которые сейчас пользуются сетевые администраторы: магистрали Т-1, цифровые соединительные линии, линии глобальной службы телекоммуникаций, традиционные аналоговые каналы и т.д.

Услуги ISDN пока предоставляются не повсеместно, но в этом заключается серьезная проблема для администратора, проектирующего глобальные сети.

ISDN станет действительно лучше альтернативных вариантов способов связи тогда, когда эти услуги будут доступны везде.

Интерфейс базового доступа к сети ISDN (BRI- Basic Rate Interface) состоит из двух B-каналов информационного обмена с пропускной способностью 64 Кбит/с и одного D-канала служебных данных с пропускной способностью 16 Кбит/с. B-каналы являются открытыми, т.е. никаких ограничений на форматы и типы передающей по ним информации нет. D-канал используется для передачи сигналов и служебной информации. Интерфейс базового доступа называют также цифровой абонентской линией.

Интерфейс основного доступа (PRI – Primary Rate Interface) используется для подключения к ISDN групп пользователей. Известный также как расширенная цифровая абонентская линия, он будет применяться в основном для подключения к сети ISDN локальных сетей, учреждений АТС и других многопользовательских коммутационных систем. Согласно североамериканскому стандарту, которого придерживаются в США, Канаде, Мексике, Японии и Юж-

ной Корее, PRI состоит из двадцати трех B-каналов со скоростью передачи 64 Кбит/с в каждом и одного D-канала с такой же скоростью передачи. Суммарная пропускная способность, таким образом соответствует пропускной способности магистрали T-1 – 1,544 Мбит/с. T-1 считается основной технологией, используемой для PRI североамериканского стандарта. По европейскому стандарту PRI состоит из тридцати B-каналов и одного D-канала. Суммарная пропускная способность PRI – 2,048 Мбит/с.

D-канал на сетевом уровне модели OSI с помощью определенных протоколов распознает тип пакета и тип передаваемого сообщения. Этот уровень в сети ISDN отвечает за установку, сопровождение и завершение соединений в сети. На канальном уровне протоколы управляют потоком кадров, движущихся по D-каналу, и предоставляют необходимую информацию для обнаружения и исправления ошибок и управлением потоком данных, а также существуют отдельные протоколы для B-канала и D-канала, которые допускают в интерфейсе соответственно одного или позволяет формировать множество таких каналов, поскольку D-канал может управлять большим количеством B-каналов. На физическом уровне отдельные протоколы предназначены для управления соответственно BRI и PRI.

Комитет IEEE 802.9 разрабатывает комплект спецификаций стандарта на передачу речи и данных по одной сети. Цель комитета – создать систему, которая будет предоставлять услуги уровня MAC (подуровень управления доступом к среде передачи канального уровня) в соответствии со стандартами IEEE 802, будучи в то же время ISDN-совместимой. Этот стандарт будет регламентировать обмен между терминалом обработки речи и данных и ЛВС и обеспечивать множественный доступ к одному каналу передачи, что достигается с помощью временного мультиплексирования и выделения каждому каналу определенного временного промежутка. На этот протокол уже ссылаются как на стандарт, но его успех зависит от ISDN.

Примером того, как клиенты собираются использовать ISDN, является корпорация McDonald’s. По сообщениям McDonald’s, она собирается установить в своей штаб-квартире 2100 линий BRI и PRI, которые соединят ее с 40 отделениями, более чем 8000 торговыми точками в США 3000 торговыми предприятиями за рубежом. Главное преимущество заключается в том, что McDonald’s сможет объединить 21 существующую сеть передачи данных в одну сеть. Пользователь любого терминала ISDN сможет обращаться к базам данных по всей сети.

Существует и широкополосный вариант ISDN, имеющий пропускную способность порядка 600 Мбит/с, что позволяет осуществлять пересылку видеоизображений с высокой разрешающей возможностью одновременно с речью и данными.

5.7. Простой протокол управления сетью (SNMP)

Простой протокол управления сетью (SNMP – Simple Network Managment Protocol). Сначала он разрабатывался как протокол сетевого управления для

TCP/IP. Сейчас его возможности позволяют контролировать сетевой трафик и выявлять аппаратные неисправности узкие места в широком диапазоне не TCP/IP сетевых устройств. Нескольким элементов SNMP присвоен статус стандартов.

Протокол SNMP обеспечивает связь системы управления сетью (NMS – Network Managment System) с управляющим устройством. На этом устройстве содержится программа-агент, которая взаимодействует с NMS. Информация хранится в базе управляющей информации, где также содержатся статистические сведения о сети (количество переданных пакетов, ошибки и т.д.) В SNMP агенты – это очень маленькие программы, поскольку их функция заключается лишь в том, чтобы отвечать на запросы сообщениями пяти видов. Основной же объем необходимых операций обработки выполняется в NMS, что позволяет фирмам-производителям создавать для своих изделий необходимые программы-агенты, чтобы обеспечить их работу в среде SNMP.

Серьезным недостатком SMNP является отсутствие средств защиты, и поэтому в 1991 году был образован комитет по разработке “безопасного SMNP”, который разработал спецификацию SMNPv2. В этом протоколе предусмотрено четыре уровня защиты:

1. защита узлов
2. предусматривает верификацию личности сетевого пользователя, обеспечивает шифрование аутентификационной информации, например паролей.
3. секретность конфигурации, но администратор сети может настраивать конфигурационные файлы из любой точки сети и шифровать данные по стандарту шифрования данных федеральной системы обработки информации.
4. полная секретность.

SMNPv2 поддерживает возможность массовых запросов данных, решает проблему отсутствия межадминистраторской связи, регламентируя обмен между администраторами высшего и нижестоящих уровней.

Существует и ряд недостатков данной версии:

его отличия от предыдущей версии делают эти протоколы несовместимыми,
сложность затрудняет его реализацию для производителей оборудования,
система защиты довольно громоздкая и ресурсоемкая,
из-за большого объема работ, необходимых для реализации данного продукта, цена систем управления на его основе высока.

5.8. Протокол общей управляющей информации и протокол общего управления над TCP/IP

Протокол общей управляющей информации (CMIP – Common Managment Information Protocol) – это стандарт управления для сетей, соответствующих модели OSI. Он содержит гораздо более надежный набор средств сетевого управления, чем SMNP. Основное различие между CMIP и SMNP состоит в том, что в SNMP не различаются объект и его атрибуты. В среде SNMP приходится формировать новые определения для каждого из устройств даже если такие устройства очень похожи на уже существующие. При работе с CMIP для

новых устройств можно использовать уже созданные определения, подключив дополнительные атрибуты, т.е. новые объекты “наследуют” определения старых. Еще одно преимущество CMIP – безопасность.

CMIP еще не получил широкого распространения, потому что пока мало сетей, работающих по протоколам модели OSI. Временным решением для фирм, собирающихся перейти на модель OSI, является протокол общего управления над TCP/IP , который состоит из стандартного CMIP с уровнем представления данных TCP/IP, который соответствует уровням модели OSI. с шестого и ниже. Этот протокол был разработан для того, чтобы фирмы, которые пользовались TCP/IP, но планировали в перспективе переход к модели OSI, могли сразу же заменить системы управления сетями, будучи уверенными, что в будущем им не придется менять программное обеспечение для управления сетью.
Одним из лидеров в области управления сетями в среде SMNP является система OpenView.

VI. МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ К ВЫПОЛНЕНИЮ И КОНТРОЛЮ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

На самостоятельную работу студентов отводится 98 часов. Часть этого времени предусматривает на доработку лабораторных работ и подготовки отчетов по ним – 36 часов (по 2 часа в неделю семестра).

Кроме этого студент должен самостоятельно изучить материалы по следующим вопросам:

1. Методы кодирования (3 – С.197–200)
2. Алгоритмы маршрутизации (1 – С.349 – 360)
3. Особенности технологий Frame Relay, ATM, SDH(1 – С.252–253, 540–564, 530–539; 3 – С. 167–185)
4. Беспроводные сети (3 – С.105 – 113)
5. Сети с коммутацией пакетов (1 – С.520 – 530; 2 – С. 198 – 203)
6. Организация корпоративных сетей (1 – С.597- 600, 2 – С. 369–378)
7. Спутниковые каналы (1 – С.111)
8. Сотовые системы связи (1 – С.479 – 486)
9. Виды конференц-связи (4)
10. Протоколы файлового обмена, электронной почты (3 – С. 241–247)

Рекомендованная литература:

1. В.Г. Олифер, Н.А. Олифер. Компьютерные сети. Принципы, технологии, протоколы. (рекомендовано Мин. образования РФ). СПб: Питер, 2001,668 с
2. К.Андерсон с М. Минаси. Локальные сети. Полное руководство. К.: ВЕК+, М.:ЭНТРОП, СПб.:КОРОНА 1999, 624 с
3. Стэн Шат Мир компьютерных сетей. К.:BHV, 1996, 288с
4. Периодические издания «Компьютерра», «Компьютер-пресс» и др.
5. Специализированные сайты htpp://citforum.amursu.ru и др.

Контроль самостоятельной работы проводится путем тестового опроса по перечисленным темам. Вопросы самостоятельного изучения включены в экзаменационные вопросы.

VII. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ОРГАНИЗАЦИИ МЕЖСЕССИОННОГО КОНТРОЛЯ ЗНАНИЙ СТУДЕНТОВ

Межсессионный контроль осуществляется на лабораторных занятий. По итогам сдачи лабораторных работ и тестовых заданий по лекционным материалам и самостоятельно изученным теоретическим вопросам в сроки, установленные деканатом (как правило, на 6-ой и 12-ой неделе семестра) преподавателем выставляется аттестационная оценка.

VII. КОМПЛЕКТЫ ЭКЗАМЕНАЦИОННЫХ БИЛЕТОВ

Экзаменационный билет № 1

1. Структура взаимодействия устройств в сети
2. Мосты в локальных сетях
3. Каким будет теоретический предел скорости передачи данных в битах в секунду по каналу с шириной полосы пропускания в 20 кГц, если мощность передатчика 0.063 мВт, а мощность шума в канале 0.001мВт

Экзаменационный билет № 2

1. Эталонная модель взаимосвязи открытых систем
2. Амплитудно-частотная характеристика, полоса пропускания, затухание
3. Запишите значения следующих характеристик топологии 10BASE 2: скорость передачи, длина сегмента кабеля, максимальное число станций в сегменте, тип кабеля.

Экзаменационный билет № 3

1. Основные требования, предъявляемые к компьютерным сетям на современном предприятии
2. Алгоритмы работы мостов
3. Определите пропускную способность канала связи для каждого направления дуплексного режима передачи, если известно, что полоса пропускания равна 600 кГц, а в методе кодирования используется 16 состояний сигнала.

Экзаменационный билет № 4

1. Типы характеристик линий связи и способы их определения
2. Классификация сетей по масштабу
3. Запишите значения следующих характеристик топологии 10BASE 5: скорость передачи, длина сегмента кабеля, максимальное число станций в сегменте, тип кабеля.

Экзаменационный билет № 5

1. Локальная сеть Ethernet (метод доступа, структура пакета, взаимодействие и подключение устройств в сети)
2. Маршрутизаторы в ЛВС
3. Определите пропускную способность канала связи для каждого направления дуплексного режима передачи, если известно, что полоса пропускания равна 400 кГц, а в методе кодирования используется 8 состояний сигнала.

Экзаменационный билет № 6

1. Методы передачи информации
2. Стандарты кабельной системы
3. Если мощность сигнала относится к мощности шума в 100 раз, то какой процент увеличения пропускной способности линии дает повышение мощности передатчика в 2 раза.

Экзаменационный билет № 8

1. Концентраторы и сетевые адаптеры
2. Структура стандартов IEEE 802.x
3. Каким будет теоретический предел скорости передачи данных в битах в секунду по каналу с шириной полосы пропускания в 20 кГц, если мощность передатчика 0.063 мВт, а мощность шума в канале 0.001мВт

Экзаменационный билет № 9

1. Среда передачи данных
2. Организация корпоративных сетей
3. Каким будет теоретический предел скорости передачи данных в битах в секунду по каналу с шириной полосы пропускания в 15 кГц, если мощность передатчика 0.0127 мВт, а мощность шума в канале 0.0001мВт

Экзаменационный билет № 10

1. Протоколы электронной почты
2. Помехоустойчивость и достоверность
3. Запишите значения следующих характеристик топологии 10BASE 2: скорость передачи, длина сегмента кабеля, максимальное число станций в сегменте, тип кабеля.

Экзаменационный билет № 11

1. Сеть Token Ring
2. Особенности ATM-технологий
3. Каким будет теоретический предел скорости передачи данных в битах в секунду по каналу с шириной полосы пропускания в 15 кГц, если мощность передатчика 0.0127 мВт, а мощность шума в канале 0.0001мВт

Экзаменационный билет № 13

1. Протоколы IPX/SPX
2. Топологии подключения устройств
3. Каким будет теоретический предел скорости передачи данных в битах в се-

кунду по каналу с шириной полосы пропускания в 25 кГц, если мощность передатчика 0.0127 мВт, а мощность шума в канале 0.0001мВт

Экзаменационный билет № 14

1. Прикладной, сеансовый и уровень представления данных в модели OSI
2. Логическая структуризация сети
3. Запишите значения следующих характеристик топологии 10BASE 2: скорость передачи, длина сегмента кабеля, максимальное число станций в сегменте, тип кабеля.

Экзаменационный билет № 15

1. Транспортный и сетевой уровни модели OSI
2. Интерфейс передачи данных FDDI
3. Если для кодирования данных использовать вместо 2 уровней 8, то на сколько процентов теоретически увеличится пропускная способность канала?

Экзаменационный билет № 16

1. Канальный и физический уровни модели OSI
2. Сети Token Ring
3. Каким будет теоретический предел скорости передачи данных в битах в секунду по каналу с шириной полосы пропускания в 20 кГц, если мощность передатчика 0.063 мВт, а мощность шума в канале 0.001мВт

Экзаменационный билет № 17

1. Стек протоколов TCP/IP
2. Физическая структуризация сети
3. Определите пропускную способность канала связи для каждого направления дуплексного режима передачи, если известно, что полоса пропускания равна 600 кГц, а в методе кодирования используется 16 состояний сигнала.

Экзаменационный билет № 18

1. Особенности технологии Frame Relay
2. Типы характеристик линий связи и способы их определения
3. Если мощность сигнала относится к мощности шума в 127 раз, то какой процент увеличения пропускной способности линии дает повышение мощности передатчика в 3 раза

Экзаменационный билет № 19

1. Иерархия кабельной системы
2. Протоколы файлового обмена
3. Каким будет теоретический предел скорости передачи данных в битах в секунду по каналу с шириной полосы пропускания в 20 кГц, если мощность передатчика 0.063 мВт, а мощность шума в канале 0.001мВт

Экзаменационный билет № 20

1. Основные требования, предъявляемые к сетям в масштабе предприятия
2. Мосты в ЛВС
3. Каким будет теоретический предел скорости передачи данных в битах в секунду по каналу с шириной полосы пропускания в 10 кГц, если мощность передатчика 0.063 мВт, а мощность шума в канале 0.001мВт

IX. КАРТА ОБЕСПЕЧЕННОСТИ ДИСЦИПЛИНЫ КАДРАМИ ПРОФЕССОРСКО-ПРЕПОДАВАТЕЛЬСКОГО СОСТАВА

Все виды занятий по данной дисциплине ведет канд. техн. наук, доцент Галаган Т.А.

СОДЕРАНИЕ

I.	ПРИМЕРНАЯ ПРОГРАММА УЧЕБНОЙ ДИСЦИПЛИНЫ, УТВЕРЖДЕННАЯ МИНИСТЕРСТВОМ ОБРАЗОВАНИЯ РФ	3
II.	РАБОЧАЯ ПРОГРАММА	4
III.	МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ПРОВЕДЕНИЮ И ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ	12
IV.	ТЕХНОЛОГИЯ ВЫПОЛНЕНИЯ И ЗАДАНИЯ К ЛАБОРАТОРНЫМ РАБОТАМ	13
V.	КОНСПЕКТ ЛЕКЦИЙ	74
VI.	МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ К ВЫПОЛНЕНИЮ И КОНТРОЛЮ САМОСТОЯТЕЛЬНОЙ РАБОТЫ	129
VII.	МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ОРГАНИЗАЦИИ МЕЖСЕССИОННОГО КОНТРОЛЯ ЗНАНИЙ СТУДЕНТОВ	129
VIII.	КОМПЛЕКТЫ ЭКЗАМЕНАЦИОННЫХ БИЛЕТОВ	130
IX.	КАРТА ОБЕСПЕЧЕННОСТИ ДИСЦИПЛИНЫ КАДРАМИ ПРОФЕССОРСКО-ПРЕПОДАВАТЕЛЬСКОГО СОСТАВА	133

Татьяна Алексеевна Галаган,
доцент кафедры ИиУС АмГУ

Учебно-методический комплекс по дисциплине «Сети ЭВМ и телекоммуникации»

Изд-во АмГУ. Подписано к печати Формат 60х84/16.
Усл. печ. л. Тираж Заказ