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 магистратуры по направлению 230100.68 «Информатика и вычислительная техника»

Составитель доцент кафедры ИУС Галаган Т.А.

Факультет математики и информатики

Кафедра информационных и управляющих систем

2010

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 магистратуры по направлению 230100.68 «Информатика и вычислительная техника» очной формы обучения. – Благовещенск: Амурский гос. ун-т, 2010.

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

© Амурский государственный университет, 2010

I. ПРИМЕРНАЯ ПРОГРАММА УЧЕБНОЙ ДИСЦИПЛИНЫ, УТВЕРЖДЕННАЯ МИНИСТЕРСТВОМ ОБРАЗОВАНИЯ РФ

Государственный образовательный стандарт высшего профессионального образования

Направление подготовки магистрата 230100.68 - Информатика и вычислительная техника

Наименование дисциплины – Сети ЭВМ и телекоммуникации

Блок дисциплины направления ДН(М).Р.2

Всего часов - 174

Содержание разделов:

Классификация информационно-вычислительных сетей. Способы коммутации. Сети одноранговые и "клиент-серер". Уровни и протоколы. Эталонная модель взаимосвязи открытых систем. Аналоговые каналы передачи данных. Способы модуляции. Характеристики проводных линий связи. Локальные вычислительные сети. Методы доступа. Множественный доступ с контролем несущей и обнаружением конфликтов. Разновидности сетей Ethernet. Маркерные методы доступа. Сети Token Ring и FDDI. Высокоскоростные локальные сети. Организация корпоративных сетей. Алгоритмы маршрутизации. Протоколы TCP/IP. Протоколы управления. Сетевые операционные системы. Web-технологии. Языки и средства создания Web-приложений.

РАБОЧАЯ ПРОГРАММА

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

- методами создания сетевых приложений с использованием протоколов TCP/IP
- методами программирования на языке JavaScript.

2. СОДЕРЖАНИЕ ДИСЦИПЛИНЫ

2.1. ФЕДЕРАЛЬНЫЙ КОМПОНЕНТ

Программа курса " Сети ЭВМ и телекоммуникации " составлена в соответствие с требованиями с требованиями государственного образовательного стандарта направления 230102, блок дисциплины направления ДН(М).Р.2:

2.2. ЛЕКЦИИ (18 часов)

2.2.1. Классификация компьютерных сетей. Требования, предъявляемые к современным сетям. (2 часа)

2.2.2. Принципы взаимодействия компьютеров в сети. Функциональные группы устройств в сети. Топологии подключения. Логическая и физическая структуризация сети.(4 часа)

2.2.3. Понятие «открытая система». Стеки коммуникационных протоколов. Уровни модели OSI (4 часа)

2.2.4. Базовые технологии локальных сетей: Ethernet, Token Ring, FDDI. Высокоскоростные локальные сети. (6 часов)

2.2.5. Основные направления развития современных сетевых операционных систем (2 часа)

2.3. ПРАКТИЧЕСКИЕ ЗАНЯТИЯ (18 часов)

2.3.1. Типы и характеристики линий связи. Стандарты кабелей. Методы передачи дискретных данных. (4 часа)

2.3.2. Адресация компьютеров в сети Ethernet (2 часа)

2.3.3. Расчет характеристик сети Ethernet PDV и PVV (2 часа)
2.3.4. Построение ЛВС с помощью мостов и коммутаторов. Алгоритмы маршрутизации (4 часов)
2.3.5. Протоколы TCP/IP (4 часа)
2.3.6. Беспроводные сети (2 часа)

2.4. ЛАБОРАТОРНЫЕ РАБОТЫ (54 часа)

2.4.1. Основы программирования на JavaScript: Структура программы, разветвляющиеся программы; операторы цикла; встроенные классы JavaScript, работа с окнами и фреймами, динамическое изменение объектов. (30 часов)

2.4.2. Основы сетевого программирования на основе протоколов TCP/IP (14 часов)

2.4.3. Методика расчета конфигурации сети Ethernet. (10 часов)

2.5. КУРСОВАЯ РАБОТА –

2.6. САМОСТОЯТЕЛЬНАЯ РАБОТА СТУДЕНТОВ

1. Методы обнаружения ошибок при передаче данных в ЛВС
2. Беспроводные сети
3. Сети с коммутацией пакетов
4. Правила построения сегментов классических Ethernet
5. Правила построения сегментов Fast Ethernet

2.6. 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 для ЛВС
10. Локальная сеть Ethernet
11. Высокоскоростные варианты Ethernet
12. Сети Token Ring
13. Распределенный волоконно-оптический интерфейс передачи данных (FDDI)
14. Устройства передачи данных
15. Основные требования, предъявляемые к вычислительным сетям на совре-

менном предприятии
16. Классификация по масштабу сети
17. Основные направления развития современных сетевых операционных систем
18. Протоколы TCP/IP
19. Мосты в локальных сетях
20. Алгоритмы работы мостов
21. Маршрутизаторы в ЛВС
22. Типы характеристик линий связи и способы их определения

2.7. ПРИМЕРНЫЕ ВОПРОСЫ ЭКЗАМЕНАЦИОННОГО ТЕСТА

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. Какая из ситуаций является коллизией?

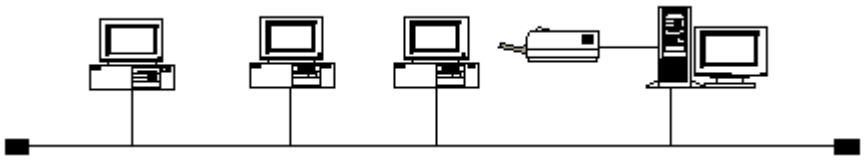
а) ситуация, когда станция, желающая передать пакет, обнаруживает, что в данный момент другая станция уже заняла передающую среду;
б) ситуация, когда две рабочие станции одновременно передают данные в

разделяемую передающую среду.

4. ***Какое название в сети Ethernet носит метод доступа станций к сети?***
   а) маркерное кольцо
   б) метод множественного доступа с контролем несущей и обнаружением коллизий
   в) маркерная шина

5. ***При использовании какого метода при обмене информацией между узлами сети данные передаются в одном направлении?***
   а) симплексной передачи;
   б) дуплексной передачи;
   в) полудуплексной передачи.

6. ***Как называется характеристика сети, предполагающая скрытие (невидимость) особенности сети от конечного пользователя?***
   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
10BASE-F?

	а	**б**	**в**	**г**	**д**
Метод передачи сигналов	Одно-Полосной	Одно-полосной	Одно-полосной	Одно-полосной	Широко-полосной
Длина сегмента	500	185	100	2000	1800

кабеля, м					
Тип кабеля	50 Омный коакси-альный, «толстый»	50 Омный коакси-альный, «тонкий»	Витая пара (UTP)	Оптоволокно (FOC)	75 Омный коаксиаль-ный, «толстый»
Топология Подключения устройства	Шина	Шина	Звезда	Звезда, двухточечное соединение	Шина

10. Где располагается горизонтальная подсистема иерархической структурированной кабельной системы?

а) в пределах этажа
б) в пределах здания
в) в пределах одной территории с несколькими зданиями
г) в пределах предприятия

11. Какими характеристиками оценивается степень искажения синусоидальных сигналов линиями связи?

а) затухание, мощность
б) пропускной способностью, полоса пропускания
в) помехоустойчивость, достоверность
г) амплитудно-частотная характеристика, полоса пропускания

12. Какие уровни модели OSI являются сетенезависимыми?

а) прикладной, представления данных
б) канальный, сетевой
в) транспортный, сеансовый
г) физический, сеансовый

13. Какие из перечисленных пар сетевых технологий совместимы по форматам кадров и позволяют обрабатывать составную сеть без необходимости транслирования кадров?

а) Token Ring – Fast Ethernet
б) FDDI – Ethernet
в) Ethernet – Fast Ethernet
г) Token Ring - FDDI

14. Поставьте соответствия между названиями стандартов и технологий, которые они описывают.

а) 802.1	**1)** технология Token Ring
б) 802.3	**2)** технология Ethernet
в) 802.5	**3)** общие определения ЛВС, связь с моделью OSI

15. Что используются в одномодовом кабеле в качестве источника излуче-

ния света?

а) светодиоды
б) полупроводниковые лазеры
с) светодиоды и полупроводниковые лазеры
г) разные виды лазеров

КРИТЕРИИ ОЦЕНОК ЗНАНИЙ СТУДЕНТОВ

Отлично Количество правильных ответов на тестовые задания составляет 91 – 100 %.

Хорошо Количество правильных ответов на тестовые задания составляет 76 – 90 %.

Удовлетворительно Количество правильных ответов на тестовые задания составляет 51 – 75 %.

Неудовлетворительно Количество правильных ответов на тестовые задания составляет 0 – 50 %.

3. УЧЕБНО-МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ:

3.1. ЛИТЕРАТУРА

ОСНОВНАЯ

1. В.Г. Олифер, Н.А. Олифер. Компьютерные сети. Принципы, технологии, протоколы. (рекомендовано Мин. образования РФ). СПб: Питер, 2007, 960 с
2. Э. Таненбаум. Компьютерные сети. СПб: Питер, 2007, 992 с
3. В.Г Олифер, Н.А. Олифер Сетевые операционные системы. СПб.: Питер, 2006. 544 с.
4. Гордеев А. В. Операционные системы. Учебник для вузов (допущено Мин. образования РФ). 2-е изд. СПб.: Питер, 2006. 480 с.
5. В. Дунаев Самоучитель JavaScript. 2-е изд. СПб: Питер, 2005

ДОПОЛНИТЕЛЬНАЯ

1. К.Андерсон с М. Минаси. Локальные сети. Полное руководство. К.: ВЕК+, М.:ЭНТРОП, СПб.:КОРОНА 1999, 624 с

3.2. МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ

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

2. Программирование на языке Java Script. Лабораторный практикум. Благовещенск. Изд.-во АмГУ. 2008.

ТЕХНИЧЕСКИЕ СРЕДСТВА ОБЕСПЕЧЕНИЯ ДИСЦИПЛИНЫ

1. Кафедра обладает необходимым количеством современных ЭВМ для проведения лабораторных работ
2. Программное обеспечение: операционные системы Windows XP/NT, Internet Explorer, программные среды Borland C++, Builder C++.

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 (семин.)	Лаборат.		Содержание	часы	
1	2.2.1		2.4.1	3.2.2	2.6.1		отчет
2		2.3.1	2.4.1	3.2.2	2.6.1		отчет
3	2.2.2		2.4.1	3.2.2	2.6.1		отчет
4		2.3.1	2.4.1	3.2.2	2.6.2		отчет
5	2.2.2		2.4.1	3.2.2	2.6.2		отчет
6		2.3.2	2.4.1	3.2.2	2.6.2		отчет
7	2.2.3		2.4.1	3.2.2	2.6.3		отчет
8		2.3.3	2.4.1	3.2.2	2.6.3		отчет,
9	2.2.3		2.4.1	3.2.2	2.6.3		отчет
10		2.3.4	2.4.1	3.2.1	2.6.3		отчет
11	2.2.4		2.4.2	3.2.1	2.6.4		отчет,
12		2.3.4	2.4.2	3.2.1	2.6.4		отчет
13	2.2.4		2.4.2	3.2.1	2.6.4		отчет
14		2.3.5	2.4.2	3.2.1	2.6.4		отчет,
15	2.2.4		2.4.2	3.2.1	2.6.5		отчет
16		2.3.5	2.4.3	3.2.1	2.6.5		отчет,
17	2.2.5		2.4.3	3.2.1	2.6.5		отчет
18		2.3.6	2.4.3	3.2.1	2.6.5		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.. Надо четко определить, входные и выходные данные, 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.

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 (4 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 С++. Для изменения порядка выполнения операций используются круглые скобки.

Операторы ветвления

Операторы ветвления сохраняют преемственность языка С++.

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.

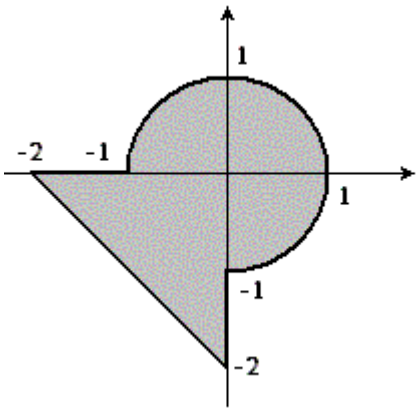
```
 switch (x ) {
case “ английский“ : window.open ( “engl.htm”) ; break
case  “ французский“  : window.open ( “french.htm”) ;   break
case  “ русский“ :  window.open ( “russ.htm”) ;              break
default  :  alert(  “Нет документа на таком языке” )
}
```

Задание

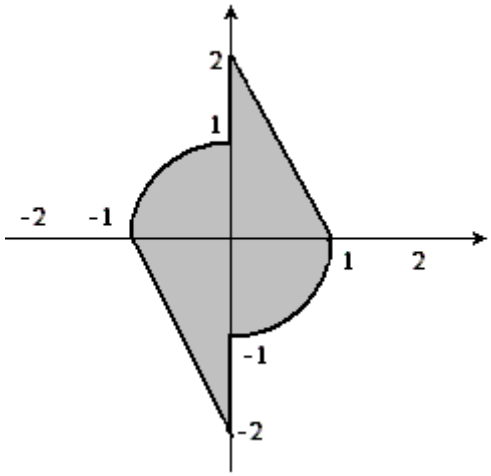
1. Создать собственный редактор программ, на основе HTML-кода, модифицировав код, приведенный выше. Файл открыть в браузере как веб-страницу.
2. Протестировать редактор, изучив в нем функции вывода и основные операции JavaScript.

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 .

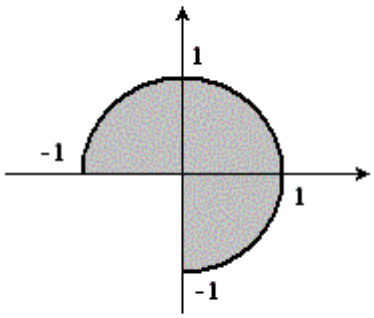
Вариант 1



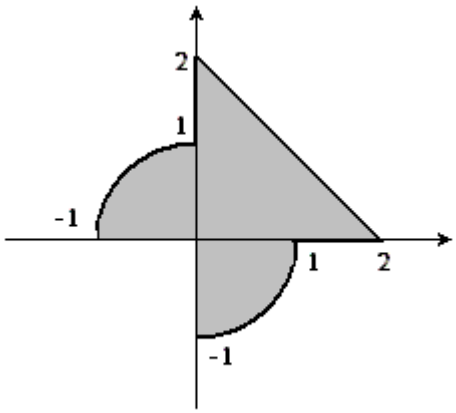
Вариант 2



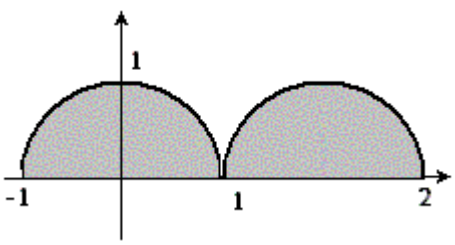
Вариант 3



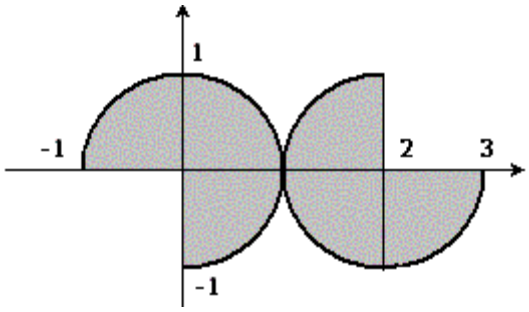
Вариант 4



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 (4 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 (4 часа)
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 (4 часа)
Работа с фреймами

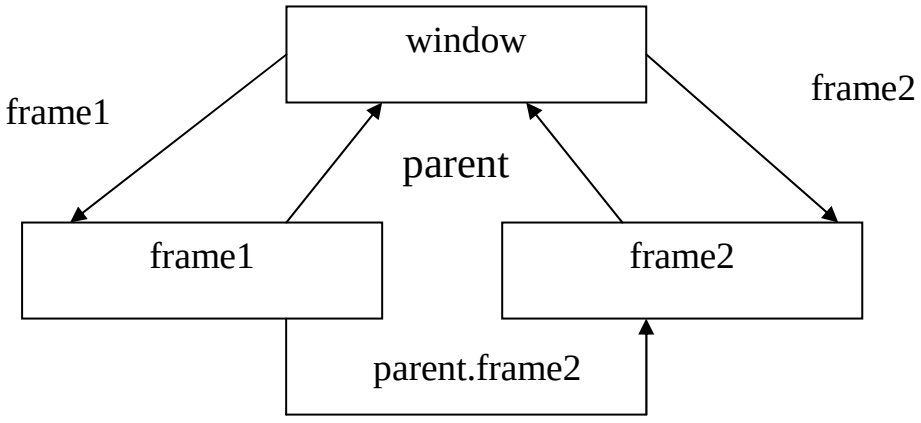
Фрейм – прямоугольная область окна браузера, в которую можно загрузить HTML-документ. Разбиение окна браузера на отдельные окна производится с помощью тега <FRAMESET>, внутрь которого вставляются теги <FRAME> с атрибутами, указывающими имя фрейма и адрес HTML-документа.

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 (4 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 (4 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 из-

менить 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 clear-Timeout( ).

Задание

В задании из предыдущей лабораторной работы к изображениям применить различные эффекты перемещения.

ЧАСТЬ 2. ОСНОВЫ СЕТЕВОГО ПРОГРАММИРОВАНИЯ

Лабораторная работа № 10 (4 часа)
Знакомство со средой программирования Borland C++ BUILDER

Все пользовательские программы в среде Borland C++ BUILDER оформляются в виде проектов. Действия по управлению объектами осуществляет программный специальный программный комплекс – Менеджер проектов.

Команда File→New Application открывает новую, так называемую, форму – заготовку для помещения обработчиков событий компонентов, которые будут размещаться в форме из набора (палитры) и реакции на события. Реакции в программах задаются обработчиками событий.

C++ BUILDER связывает с каждым приложением, оформленным как проект, следующие исходные файлы:

Unit1.cpp – текст программы;

Unit1.h – интерфейсный модуль с объявлениями компонентов, расположенных в данной форме, глобальных переменных, функций и т.д.;

Unit1.dfm – хранит описания формы и всех расположенных в ней компонентов. Это файл поддерживается и обновляется средой автоматически;

Project1.cpp – главная управляющая программа проекта – проектный файл. Любая новая форма автоматически включается в этот файл. С его помощью вызывается, выполняется и завершается код проекта;

Project1.res – файл ресурсов проекта. В нем хранятся значки, используемые в проекте, заданные как разработчиком, так и системой.

Одновременно с открытием новой формы создается новый элемент, окно которого появляется на экране вместе с формой – Инспектор объекта (Object Inspector). Он также создается для каждого помещенного в форму компонента. Инспектор объекта позволяет увидеть основные свойства (Properties) и события (Events) объекта. Полный перечень свойств, событий и методов можно увидеть по справочной службе Help (F1) при выделенном компоненте.

Инспектор объектов имеет свое контекстное меню, которое открывается правой кнопкой мыши.

Помещение компонента в форму осуществляется следующими способами:

1. найти нужный компонент на вкладках палитры, щелкнуть по нему мышью, перевести указатель мыши на нужное место окна дизайнера форм, снова щелкнуть мышью.
2. дважды щелкнуть на нужном компоненте на вкладке;
3. если требуется несколько экземпляров одного компонента, нажать Shift и щелкнуть на нужном компоненте палитры, затем указывая необходимые места формы. Для прекращения – дважды щелкнуть на компоненте вкладки.

Для редактирования текста модуля требуется:

- Открыть вкладку с именем требуемого модуля в окне Редактора кода (для переключения между окном Редактора и формой используют клавишу F12) или открыть диалоговое окно командой View|Units и выбрать в нем требуемый модуль.

- поместить курсор Редактора в нужное место текста и редактировать по правилам стандартных текстовых редакторов.

У Редактора кодов существует свое контекстное меню, открывающееся правой кнопкой мыши. Суфлер кода помогает быстрее и правильнее набирать текст программы.

Задание

Создать проект в среде Builder C++. Организовать экранную форму с набором кнопок. По нажатию кнопки внутри формы должен строиться график. По нажатию другой - график исчезает. Продемонстрировать построение трех произвольных функций.

Лабораторная работа №10 (2 часа)

Компоненты TPanel, TLabel, TEdit, TMainMenu, TMemo, TPopupMenu

Компонент TPanel используется в качестве базового класса для определения объектов, которые включают в себя другие объекты. При перемещении панели компоненты перемещаются вместе с ней. Перечислим лишь основные ее свойства, события и методы.

Свойства TPanel

BevelInner – определяет стиль внутренней кромки;
BevelOuter – стиль внешней кромки;
BevelWidth – ширина кромок в пикселах;
BorderWidth – расстояние в пикселах между внутренней и внешней кромками;
Align – размещает панель в форме в соответствии со значениями этого свойства;
Aligment – определяет выравнивание названия панели относительно самой панели;
Ct3D – появление панели в трехмерном или двумерном виде.

Методы TPanel

CanFocus( ) – показывает, может ли компонент получить фокус, т.е. стать активным; возвращает true, если свойства Visible и Enabled компонента и его родителя имеют значение true.
Focused( ) – определяет, имеет ли компонент входной фокус;
Hide( ) – делает компонент невидимым, устанавливая его свойство Visible в false.
SetFocus( ) – делает компонент активным;
Show( ) - показывает компонент, если он до этого был спрятан.

Компонент TLabel выводит в форму текст, который пользователь в режиме исполнения приложения не может редактировать.

Свойства TLabel

Aligment – задает способ выравнивания текста;

FocusControl – имеет раскрывающийся список, в который попадают компоненты, которые могут быть связаны с меткой и получать от нее фокус ввода. Для этого после выбора компонента, в ее тексте (свойство Caption) указывается символ & перед символом, который станет горячей клавишей при исполнении приложения.

Layout – размещение текста, заданного в свойстве Caption, в поле метки.

Transparent – если некоторый компонент расположен под меткой, он может быть невидимым. Чтобы этого не произошло, надо сделать метку прозрачной(трансарентной).

WordWrap – если это свойство равно true, а свойство AutoSize – false, все слова текста в свойстве Caption станут располагаться в разных строках.

Компонент TEdit задает в форме однострочное редактируемое окно для ввода и вывода данных.

Свойства TEdit

AutoSelect – значение true, когда компонент получает фокус ввода на весь текст, false – курсор ввода остановится в начале текста.

BorderStyle – наличие или отсутствие окантовки;

CharCase – задает регистр ввода текста: EcLowerCase – текст преобразуется в символы нижнего регистра, EcUpperCase – верхний регистр; EcNormal – оба регистра.

HideSelection – значение true, если при перемещении фокуса ввода выделенный текст не меняется подсветки, false – подсветка исчезает при выделении другого компонента.

PasswordChar – при записи вида Edit->PasswordChar='*'; вместо вводимых символов в поле Edit высветятся звездочки.

ReadOnly – при значение true пользователь не может менять текст в поле компонента.

Text – поле ввода-вывода текста.

Методы TEdit

Clear( ) – очищает поле компонента, удаляя весь текст.

ClearSelection( ) – удаляет выделенный текст.

ClearUndo( ) – очищает буфер Undo, после чего невозможна отмена предыдущих изменений текста.

CopyToClipBoard( ) – удаляет выделенный текст, копируя его в буфер памяти.

CutToClipBoard( ) – вставляет текст из буфера памяти, заменяя выделенный текст.

PasteFormClipBoard( ) – вставляет текст из буфера памяти, заменяя выделенный текст.

SelectAll( ) – выделяет весь текст в поле компонента.

Undo( ) – отменяет все изменения после последнего вызова ClearUndo( ). Если этот метод не вызывался – отменяет все изменения.

CanFocus( ) – определяет, может ли компонент получить фокус ввода.

SetFocus( ) – делает компонент активным.
GetTextLen( ) – возвращает длину текста в компоненте.
Hide( ) – делает компонент невидимым.
Show( ) – делает компонент видимым.

Компонент TMemo – многострочное редактируемое поле. TMemo – массив пронумерованных текстовых строк типа Tstrings. Для обращения к его i-ой строке следует писать: Memo->Lines->Strings[i];

Количество строк: Memo->Lines->Count;

Длина i-ой строки: Memo->Lines->Strings[i].Lengs;

Очищение текстового поля: Memo->Lines->Clear( );

Свойства TMemo

Lines – открывает Редактор текстовых строк, в котором можно набрать текст как в обычном текстовом редакторе.
MaxLength – задает максимальную длину строки текста.
ScrollBars – задает полосы прокрутки окна Memo-поля.
WantReturns – при значении true можно ли вставлять символы возврата в текст.
События и методы аналогичны соответствующим событиям и методам компонента TEdit.

Компонент TMainMenu создает главное меню приложения и с помощью него управляет работой всех его частей, т.е. запускает разные части приложения на выполнение отдельными командами, обеспечивает выход из приложения.

Меню добавляется в форму перенесением его значка из палитры компонентов. С формой меню связывается через свойство формы Menu. После формирования меню, запуск приложения будет сопровождаться появлением в верхней левой части формы строки, содержащей главные опции меню. Главные опции могут распадаться на детальные команды, расположенные сверху вниз.
Для формирования опций меню используют Дизайнер меню, который открывается через команду MenuDesigner в контекстном меню компонента, либо двойным щелчком по компоненту.
В левом верхнем углу окна Дизайнера меню располагается синие поле, предназначенное для названия первой опции. После задания его названия (свойство Caption), справа появится новое серое поле – заготовка для следующей опции. Каждая опция – новый объект и имеет собственный Инспектор объекта.
Если данная опция – последняя в иерархии, т.е. является исполнительной, нужно с помощью ее Инспектора объекта перейти на вкладку Events и дважды щелкнуть в поле события OnClick. В открывшийся обработчик события нужно вписать требуемые действия. После этого можно продолжить формирование следующей опции горизонтальной строки.

Свойства TMainMenu

Images – обеспечивает появление изображений слева от названия команд и подменю.
Items – массив, хранящий все опции меню. В поле этого свойства содержится кнопка с многоточием, открывающая дизайнер меню. Это свойство само явля-

ется указателем на класс TMenuItem и может использоваться для обращения к свойствам и методам этого класса. Кроме того, оно является массивом, хранящим все опции меню. Например, количество строк меню с именем MainMenu1 можно посчитать так: MainMenu1-> Items->Count. К опции можно обратиться по номеру: MainMenu1->Items[i]
OwnerDraw – задает возможность рисования некоторого объекта при выборе опции меню, по умолчанию имеет значение false. При установление true в обработчике события OnDrwItem будет появляться рисунок

Основные свойства опций TMainMenu
Bitmap – позволяет выбрать значок в открывающемся диалоговом окне. Значок появится слева от названия опции.
Checked – осуществляет контроль выбора данной команды меню: в обработчике события OnClick этой опции надо присвоить свойству Checked значение true. Если свойство RadioItem некоторой опции имеет значение false, то при щелчке мышью на этой опции слева от ее названия появится галочка. Если же наоборот – опция не будет отмечена. При Checked= RadioItem= true будет всегда помечена только одна опции из всех, у которых свойство GroupIndex имеет одинаковое значение. Пометка будет сделана в виде жирной точки слева от названия опции.
Shortcut – в раскрывающемся списке надо выбрать комбинацию клавиш, которая в режиме исполнения приложения заменит нажатие мышью на опцию. Действие этих клавиш возможно только при открытом меню и распространяется только на команды подменю.

Компонент TPopupMenu может быть связан с любым другим компонентом (кнопкой, формой и пр.) у которого имеется свойство PopupMenu (всплывающее меню). При помещении TPopupMenu в форму, его имя будет видно в любом из компонентов формы со свойством PopupMenu. Если меню связано с формой, оно появляется при нажатии правой кнопки мыши в активной форме.

Свойства TPopupMenu
Многие свойства совпадают со свойствами TMainMenu. Свойство Item также является массивом строк-опций меню и указателем на класс TMenuItem. Обращаться к опции можно, указав ее порядковый номер в меню: PopupMenu->Items[i] Нумерация начинается с нуля, а общее количество считается как PopupMenu->Items->Count. Для обращения к вложенным опциям надо указать имя объекта – основной опции.
Aligment – задает расположение меню, при его появлении при нажатии правой кнопки мыши.
AutoPopup – при значении false появление выпадающего меню надо устанавливать программно с помощью метода Popup, иначе оно автоматически привязывается к компоненту, с которым связано.
MenuAnimation – задает анимационный эффект появления меню на экране.
Trackbutton – задает кнопку мыши, при нажатии которой меню появляется.

Задание

Создать приложение согласно заданиям индивидуального варианта. В каждом приложении должна обеспечиваться регистрация пользователя в приложении.

Регистрация пользователя должна реализовываться с помощью дополнительной формы, в которой расположены:

- компоненты TEdit (для ввода имени и пароля пользователя);
- кнопка «Завершение регистрации», которая прекращает регистрацию вместе с приложением;
- экземпляр компонента TMemo, в котором заданы истинные имя пользователя и пароль, с которыми надо сверять вводимые значения. На этапе разработки данный компонент невидимый.

Форма регистрации должна иметь статус главной, т.е. при запуске проекта она вызывается первой. При условии верной регистрации должен осуществляться вызов формы самого приложения.

Варианты

1. Составить программу для расчета выплат по кредиту по месяцам. Входными данными являются: сумма кредита, процентная ставка, количество месяцев, на которые предоставлен кредит. Погашение кредита предполагается равными долями, а формула для начисления процентов по кредиту $A_n = A_0\left(\frac{p}{100}\right)$, где A_n - начисленные проценты, A_0 - сумма, на которую налагаются проценты, p - процентная ставка. Проценты начисляются ежемесячно на остаток кредитной суммы.

В программе создать меню, которое по желанию пользователя выводит полный список выплат или выплаты определенного месяца.

При регистрации в имени и пароле могут присутствовать только символы нижнего регистра.

2. Составить программу начисления процентов и итоговой суммы вклада. $A_n = A_0\left(1+\frac{p}{100}\right)^n$ Входными данными являются: сумма вклада, процентная ставка, количество месяцев, на которые требуется рассчитать доход.

В программе создать меню, которое по желанию пользователя выводит полный список процентов, либо полный список суммы на счете, либо проценты и сумма вклада на определенный месяц.

При регистрации в имени и пароле могут присутствовать только символы верхнего регистра.

Лабораторная работа №11 (2 часа)

Компоненты TListBox, TComboBox, TMaskEdit

Компонент TListBox создает прямоугольную область, в которой отобра-

жается список текстовых строк, которые можно добавлять, удалять, выбирать. Список строк может быть заранее приготовлен в файле, который загружается в ListBox. На этапе разработки приложения часто формируют, так называемый отладочный список, который формируют с помощью редактора текста (аналогично компоненту TMemo). Редактор текста открывается в свойстве Items.

Свойства TListBox

Items является указателем на класс TStrings, который имеет свойство Strings, являющееся массивом строк типа AnsiString. Свойство Count содержит количество строк. Запись вида ListBox1->Items->Count возвращает количество строк.

Класс TStrings содержиттметоды Add( ) – добавить строку в конец списка, Delete( ) – удалить строку и др. Например,

ListBox1->Items-> Add(“новая строка”);

ListBox1->Items->LoadFromFile(“spisok.txt”);

ItemIndex хранит выбранную в списке строку:
ListBox1->Items->Strings[ListBox1-> ItemIndex]

MultiSelect – определяет возможность выбора более одной строки за один раз. Когда это свойство имеет значение true, и выбрано множество строк, значение свойства ItemIndex, указывает на строку, имеющую фокус, т.е. на последнюю отмеченную.

Sorted – устанавливает сортировку по алфавиту.

Style – определяет вид вывода на экран названия элементов.

Методы TlistBox

Clear( ) – удаляет список,
SetFocus( ) – делает список активным,
Hide( ) – скрывает список,
Show( ) – делает список видимым,
Sort( ) – сортирует элементы в порядке возрастания.

Если количество строк не помещается в отведенный размер, автоматически включается вертикальная полоса прокрутки, при условии, что длина строки меньше ширины прямоугольника. Для выполнения этого условия, можно ограничить длину строки:

```
int MaxWidth=500;
ListBox1->Perform(LB_SETHORIZONTALEXTENT, MaxWidth, 0)
```

Компонент TComboBox является комбинацией редактируемого поля и списка TListBox – редактируемое поле с треугольной кнопкой справа. Выбор данных из списка возможен двумя способами: щелчком мыши по нужной строке или вводом нужной строки в редактируемое поле.

Компонент TMaskEdit создает редактируемое текстовое поле (маску) для ввода данных специфического формата: даты, времени, номера телефона и т.п. При задании формата ввода данных проверяется их соответствии формату. Маска налагает ограничения на вводимые символы. Маска состоит из трех полей, разделенных точкой с запятой.

Первое поле маски может содержать перечень специальных символов:
! – его присутствие интерпретирует необязательные символы как лидирующие пробелы;
> – все последующие символы должны быть набраны в верхнем регистре до коца маски, либо до появления <;
< – аналогично предыдущему для нижнего регистра;
<> – возможно использование обоих регистров;
\ – символ ввода, следующий за этим – символьная константа;
L – ввод символов алфавита языка, установленного по умолчанию;
l – в позицию, которую он занимает, можно вводить только символы алфавита;
A – ввод алфавитно-цифровых символов;
a – совпадает с предыдущим, но не требует обязательного ввода;
C – требует обязательного ввода любого символа в позицию, которую он занимает;
0 – требует обязательного ввода цифры;
9 – требует необязательного ввода цифры;
– необязательный ввод цифры или знаков + и –;
: – используется при вводе времени
/ – используется при вводе даты;
; – деление частей маски;
_ – вставляет пробелы в текст.

Остальные символы могут использоваться в качестве литералов, вставляемых автоматически (курсор ввода перескакивает через них). Кроме того, после использования в маске символа \ все последующие символы воспринимаются как литералы.

Второе поле маски – один символ, указывающий, могут ли символы-литералы из маски включаться в данные. Например, маски телефонного номера: (000)_000-0000;0;*. Нуль во втором поле означает, что свойство Text включает 10 цифр, а остальные литералы не включаются в результат ввода.

Третье поле маски содержит символ, который пользователь увидит до того, как будут введены данные. По умолчанию этот символ _.

Маска задается с помощью редактора маски, который открывается нажатием кнопки многоточие, расположенной в свойстве EditMask.

Задание

Создать приложение, содержащее возможность выбора в списке фамилий нескольких строк и копирование их (по нажатию кнопки) в другой список. Исходный список должен храниться в файле. Отсортировать новый список в алфавитном порядке. Для каждой фамилии из списка предусмотреть ввод персо-

нальных данных: имя, дата рождения, телефон и др. Для ввода использовать маски.

Лабораторная работа № 12 (2 часа)

Компоненты TIMage, TShape, TBevel

Через компонент TIMage в форму выводится графическое изображение, которое указывается в свойстве Picture. Компонент TIMage содержит свойства, определяющие формат вывода изображения внутри границ объекта, заданного в виде пустого квадрата. Некоторые из них:

Picture – указатель на класс TPicture, объектами которого являются значки, метафайлы, растровые изображения. Для загрузки изображения в компонент в поле этого свойства находится кнопка с многоточием, с помощью которой можно открыть диалоговое окно загрузки изображения. Методы TPicture: LoadFromFile( ) позволяет загрузку изображения: Image->Pictuche->SaveToFile(“имя файла”); SaveToFile( ) – сохраняет изображение в файле;

Stretch – имеет значение true, если изображение должно принимать размеры и форму компонента и менять их в соответствии с изменением параметров формы. Изменение размеров изображения происходит непропорционально, что приводит к искажению изображения. Во избежание этого требуется пользоваться свойством AutoSize. Если это свойство установлено в true, рамка компонента автоматически настраивается на размер изображения и ее нельзя изменить. Свойство Stretch при этом отключается.

Canvas – класс со свойствами и методами, позволяющими рисовать изображения. Оно доступно только для изображения формата bmp. Орудиями рисования являются Pen, Brush и методы.

Assign – используется при копировании изображения в буфер по правилу : куда->Assign(откуда); Для запоминания изображения в буфере в программный модуль должен быть включен файл Clipbrd.hpp.

Clipboard(0->Assign(Image1->Picture) //копирование изображения в буфер

Компонент TShape предназначен для рисования простых геометрических фигур, вид которых задается свойством Shape. Цвет и способ штриховки фигуры задаются в свойстве Bruch с помощью вложенных свойств Color и Style. Характеристики контура фигуры задаются во вложенных свойствах свойства Pen, а заливка свойством Bruch.

Компонент TBevel задает обрамление: рамку, кадр или линии (свойство Shape), которые могут быть вогнутыми или выпуклыми (свойство Style).

Задание

Создать программу, демонстрирующую свойства перечисленных компонент.

Лабораторная работа №13 (6 часов)

Некоторые компоненты вкладки Internet C++Builder

Компонент TServerSocket

Данный компонент делает приложение сервером, работающим по протоколу TCP/IP.

Свойства *TServerSocket*

Свойство Active - это свойство указывает, открыто ли так называемое *стыковочное соединение* и доступно ли оно для связи с другими машинами. Это свойство унаследовано от абстрактного класса TAbstractSocket, который содержит свойства и методы, позволяющие работать приложению со стыковочными компонентами (sockets) Windows, которые, в свою очередь, включают в себя набор коммуникационных протоколов, позволяющих приложению соединяться с другими машинами для чтения и записи информации. Стыковочные компоненты позволяют приложению формировать соединения с другими машинами, не заботясь о деталях существующего сетевого программного обеспечения. Свойства Компонента TAbstractSocket описывают IP-адрес стыковочного соединения и обслуживают его. IP-адрес - это строка из четырех чисел, разделенных между собой точками (например, 192.200.1.15).

Чтобы создать стыковочное соединение с другой машиной используется класс TClientSocket, а для создания стыковочного соединения, которое отвечает на запросы о соединении, поступающие от других машин используется класс TServerSocket. Перед тем как использовать или изменить стыковочное соединение, следует посмотреть его свойство Active, чтобы определить, открыто ли соединение и готово ли оно к работе. Для клиентских стыковочных соединений (компонент TClientSocket) установка свойства Active открывает или закрывает стыковочное соединение с другой машиной. Для серверных стыковочных соединений установка свойства Active открывает или закрывает прослушивание запросов от клиентов (прослушивание — это процесс, когда сервер ждет появления запросов от других машин и устанавливает соединение с той из них, от которой он принял запрос). В режиме проектирования приложения установка свойства Active в true заставляет стыковочное соединение сервера открыть соединение (т. е. будет установлено прослушивание запросов от других машин), когда приложение запустится.

Свойство Port - задание числового идентификатора, идентифицирующего серверное стыковочное соединение. Номера портов позволяют отдельной системе, заданной на стороне клиента свойствами Host или Address принимать одновременно множество соединений. (В свойстве Host задается URL, синоним IP-адреса, например, www. Mail.ru, в свойстве Address задается IP-адрес.) У свойства Port может быть много значений: это связано с обслуживанием данных, передаваемых по разным протоколам. Серверные стыковочные соединения устанавливают свойство Port в определенное значение, которое могут использовать клиенты для инициирования соединений. Для Клиентских стыко-

вочных соединений Port — это идентификатор порта сервера, с которым клиент желает осуществить соединение. Значение свойства Port обычно связано с видом обслуживания, которое требуется клиенту от серверного приложения. Изменение значения свойства должно происходить при неактивном соединении, иначе возникает исключительная ситуация типа EsocketError.

Свойство ServerType указывает, каким будет каждое принятое сервером соединение: выделяется ли оно автоматически в отдельный исполняемый поток, или нет. Первое происходит, если ServerType установлено в stThreadBlocking. В этом случае исполнение потока не происходит, пока не будет прочитана или записана вся информация, проходящая через соединение. Для каждого соединения такой поток генерирует события OnClientRead или OnClientWrite. Если свойство serverType имеет значение stNonBiocking, то обработка всех процессов чтения и записи информации, проходящих через стыковочные соединения, происходит асинхронно. При этом режиме все клиентские соединения обрабатываются по умолчанию в одном исполнительном потоке.

Свойство Service задает имя службы, для которой используется стыковочное соединение. Служба – это подсистема сетевого программного обеспечения, выполняющая конкретную задачу. Для серверных стыковочных соединений использование свойства Service вместо свойства Port означает, что сервер будет прослушивать TCP/IP-запросы соответствующего ему порта.

Свойство Socket указывает на TServerwinSocket-объект, который описывает концевую точку прослушиваемого соединения. С помощью этого свойства можно получить (пользуясь свойствами и методами класса TServerWinSocket) следующие данные:

информацию о потоках соединения, которые сформированы для обработки серверным стыковочным соединением;

информацию о соединениях в виде Socket->connections[int Index] — массива открытых соединений с клиентскими стыковочны ми узлами для данного прослушивающего стыковочного узла;

другие данные.

Свойство ThreadCacheSize – задает максимальное число потоков, используемых для новых клиентских соединений.

Компонент *TCIientSocket*

Этот компонент превращает приложение в клиента сетевого соединения по протоколу **TCP/IP.**

Свойства *TCIientSocket:*

Свойство Active аналогично одноименному свойству компонента TServerSocket.

Свойство Address задается IF-адрес серверноного проиложения. Для каждого стыковочного клиентского соединения надо придать свойству Address значение IP-адреса того сервера, с которым это стыковочное соединение должно быть установлено. Когда соединение открыто, значение Address связывается с соединением. Если стыковочное соединение содержит значение свойства Host

вместо значения свойства Address, то IP-адрес извлекается из специальной таблицы. Одна серверная система может поддерживать более одного IP-адреса.

Свойство ClientType определяет, как осуществляет клиентский стыковочный узел чтение и запись информации: асинхронно или нет. Если ClientType = ctNonBlocking, клиентский стыковочный узел отвечает на асинхронные события записи и чтения.

Свойство Host – это URL, синоним IP-адреса серверного приложения, например. **www.rambler.com.** Если задан IP-адрес, это свойство указывать не надо.

Свойство Port – номер порта стыковочного узла сервера, с которым клиент собирается осуществить соединение.

Свойство Service аналогично одноименному свойству компонента TServerSocket.

Свойство Socket описывает клиентский стыковочный узел с помощью свойств и методов класса TClientWinSocket, указателем на который оно является. Например, свойство позволяет:

определить адрес и порт стыковочных соединений и клиента, и сервера, связанных с соединением;

читать и писать информацию через стыковочное соединение:
AnsiString S=Socket->ReceiveText (), Socket->SendText(S);

- определять, на какие замечания сервера должен отвечать клиент.

События *TCIientSocket*

Событие onConnect происходит сразу после того, как установлено соединение со стыковочным узлом сервера. В зависимости от службы, заданной в соответствующем свойстве, в этот момент могли бы начаться чтение или запись через стыковочный узел клиента. Когда стыковочный узел открывает соединение, возникают следующие события:

onLookup возникает раньше попытки найти стыковочный узел сер вера;

OnConnecting возникает после того, как найден стыковочный узел сервера, требование соединения принято сервером и подготовлено клиентским стыковочным узлом;

OnConnect возникает после того, как соединение установлено.

Событие OnDisconnect происходит прямо перед тем, как клиентский стыковочный узел закрывает соединение с сервером. OnDisconnect происходит после проверки того, что свойство Active установлено в false, но до закрытия существующего соединения.

Событие OnError – это реакция на ошибки, возникающие при работе стыковочного соединения. При обработке ошибки, надо установить параметр ErrorCode обработчика в 0, иначе после выхода из обработчика возникнет исключительная ситуация типа ESocketError.

Событие OnLookup возникает перед попыткой найти серверное стыко-

вочное соединение, с которым собирается соединиться данный клиент.

Событие OnRead возникает при чтении информации из стыковочного соединения. Если стыковочное соединение – блокированное, то для чтения из него надо использовать объект TwinSocketstream. В противном случае надо использовать методы параметра Socket. (В стыковочных соединениях Nonblocking для последнего бита данных, передаваемых через это соединение, не всегда возникает событие OnRead. Поэтому в этих случаях надо проверять непрочитанные данные в обработчике события OnDisconnect, чтобы быть уверенным, что все обработано верно.)

Событие OnWrite возникает тогда, когда клиентское стыковочное соединение пишет информацию в стыковочное соединение сервера. Если стыковочное соединение блокировано, то для записи следует использовать методы объекта TwinSocketstream. В противном случае используются методы параметра Socket.

Задание

Используйте компоненты TServerSocket и TClientSocket в приложении, обеспечивающем обмен сообщениями между клиентом и сервером. Приложения после запуска должно выполнять прослушивание сервером клиентов. Режим запускается либо командой File|Listen меню приложения, либо дублирующей эту команду кнопкой со знаком "?".

В нижней части формы приложения расположить компонент TStatusBar, в поле которого отображаются производимые действия. После запуска прослушивания можно запускать соединение с сервером со стороны клиента (командой File|Connect меню или кнопкой с изображением конверта). В этом случае запрашивается URL сервера с помощью функции InputQuery( ). Для проверки работы компонентов на локальной машине, в ответ па запрос следует задать собственного компьютера, которое можно найти либо в реестре Windows, либо с помощью команды Сведения о системе.

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 (10 часов)

Основы проектирования локальной сети

Выбор конфигурации Fast Ethernet

Для определения работоспособности сети Fast Ethernet стандарт IEEE 802.3 предлагает две модели, называемые Transmission System Model 1 и Transmission System Model 2. Первая модель основана на нескольких несложных правилах. Она исходит из того, что все компоненты сети (в частности, кабели) имеют наихудшие из возможных временные характеристики, поэтому

всегда дает результат со значительным запасом. Вторая модель использует систему точных расчетов с реальными временными характеристиками кабелей. В связи с этим ее применение позволяет иногда преодолеть жесткие ограничения модели 1.

Правила модели 1

В соответствии с первой моделью, при выборе конфигурации надо руководствоваться следующими принципами:

Сегменты, выполненные на электрических кабелях (витых парах) не должны быть длиннее 100 метров. Это относится к кабелям всех категорий – 3, 4 и 5, к сегментам 100BASE-T4 и 100BASE-TX.

Сегменты, выполненные на оптоволоконных кабелях, не должны быть длиннее 412 метров.

Если используются адаптеры с внешними (выносными) трансиверами, то трансиверные кабели (MII) не должны быть длиннее 50 сантиметров.

Модель 1 выделяет три возможные конфигурации сети Fast Ethernet:

1. Соединение двух абонентов (узлов) сети напрямую, без репитера или концентратора (рисунок 2). Абонентами при этом могут выступать не только компьютеры, но и сетевой принтер, порт коммутатора, моста или маршрутизатора. Такое сопряжение называется соединением DTE - DTE или двухточечным.

Рисунок 2. Двухточечное соединение компьютеров без концентратора

2. Соединение двух абонентов сети с помощью одного концентратора (репитера) класса I или класса II (рисунок 3).

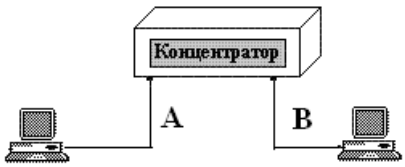


Рисунок 3. Соединение с одним концентратором

3. Соединение двух абонентов сети с помощью двух концентраторов класса II (рисунок 4). При этом предполагается, что для связи концентраторов всегда используется электрический кабель длиной не более 5 метров.

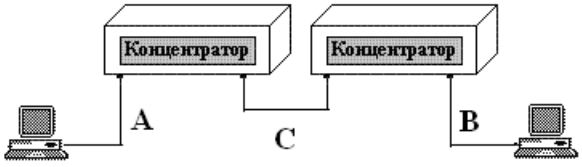


Рисунок 4. Соединение с двумя концентраторами

Концентраторы класса II имеют меньшую задержку, поэтому их может быть два. Использование трех концентраторов в соответствии с моделью 1 не допускается.

В случае выбора первой конфигурации (двухточечной) правила модели 1 предельно просты: электрический кабель не должен быть длиннее 100 метров, полудуплексный оптоволоконный – не более 412 метров, полнодуплексный оптоволоконный – 2000 метров (при этом задержка сигнала в кабеле не имеет значения, так как метод CSMA/CD не работает).

В случае применения второй конфигурации (с одним концентратором) надо ограничивать длину кабелей A и B сети в соответствии с таблицей 3.

Вид кабеля A	Вид кабеля B	Класс концентратора	Макс. длина кабеля A, м	Макс. длина кабеля B, м	Макс. размер сети, м
TX, T4	TX, T4	I или II	100	100	200
TX	FX	I	100	160,8	260,8
T4	FX	I	100	131	231
FX	FX	I	136	136	272
TX	FX	II	100	208,8	308,8
T4	FX	II	100	204	304
FX	FX	II	160	160	320

Таблица 3. Максимальная длина кабелей в конфигурации с одним концентратором

В случае выбора третьей конфигурации сети (с двумя концентраторами) надо ограничивать длину кабелей A и B в соответствии с таблицей 4. При этом по умолчанию предполагается, что кабель C имеет длину 5 метров.

Вид кабеля A	Вид кабеля B	Макс. длина кабеля A, м	Макс. длина кабеля B, м	Макс. размер сети, м
TX, T4	TX, T4	100	100	205
TX	FX	100	116,2	221,2
T4	FX	136,3	136,3	241,3
FX	FX	114	114	233

Таблица 4. Максимальная длина кабелей в конфигурации с двумя концентраторами

В обеих конфигурациях с концентраторами при использовании одновременно электрического и оптоволоконного кабелей можно за счет уменьшения длины электрического кабеля увеличить длину оптоволоконного. Причем уменьшению длины электрического кабеля на 1 метр соответствует увеличение длины оптоволоконного кабеля на 1,19 метра. Например, уменьшив кабель TX на 10 метров, можно увеличить кабель FX на 11,9 метра, и его предельная длина составит при двух концентраторах 128,1 метра. Немного увеличится и предельный размер сети (в нашем примере на 1,9 метра).

В случае использования двух оптоволоконных кабелей можно уменьшать один из кабелей за счет увеличения другого. При уменьшении одного кабеля на 10 метров можно увеличить другой тоже на 10 метров. Если же используется два электрических кабеля, то увеличивать один из них за счет уменьшения дру-

гого нельзя, так как их длина в принципе не может превышать 100 метров из-за затухания сигнала в кабеле.

Концентратор класса II в принципе не может одновременно поддерживать сегменты с разными методами кодирования TX/FX и T4. Поэтому варианты, соответствующие вторым снизу строкам обеих таблиц 3 и 4 никогда не реализуются на практике, но стандарт все же дает цифры и для них.

Во всех перечисленных случаях под размером сети понимается размер *зоны конфликта* (области коллизии). При этом надо учитывать, что включение в сеть одного коммутатора позволяет увеличить полный размер сети вдвое.

Пример сети максимальной конфигурации в соответствии с первой моделью для витой пары показан на рисунке 5. Здесь максимальный размер *зоны конфликта* складывается из сегментов A, B и C, то есть составляет:

100 + 5 + 100 = 205 метров,

что удовлетворяет условию работоспособности сети (таблица 4, верхняя строка).

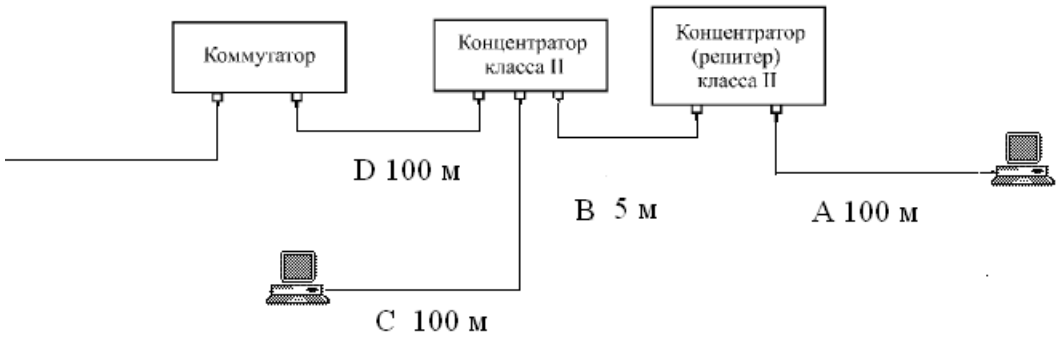


Рисунок 5. Пример максимальной конфигурации сети Fast Ethernet

Сегмент D также входит в *зону конфликта*, так как коммутатор является полноправным передатчиком пакетов сети. Длина сегмента D не может превышать 100 метров, чтобы суммарная длина сегментов A, B и D не была больше все тех же 205 метров. Сегменты, отделенные от рассматриваемой *зоны конфликта* коммутатором, никак не влияют на ее работоспособность.

Расчет по модели 2

Вторая модель для сети Fast Ethernet, как и в случае Ethernet, основана на вычислении суммарного двойного времени прохождения сигнала по сети. В отличие от второй модели, используемой для оценки конфигурации Ethernet, здесь не проводится расчетов величины сокращения межпакетного интервала. Это связано с тем, что даже максимальное количество репитеров и концентраторов, допустимых в Fast Ethernet (два), в принципе не может вызвать недопустимого сокращения межпакетного интервала.

Для расчетов в соответствии со второй моделью сначала надо выделить в сети путь с максимальным двойным временем прохождения и максимальным числом репитеров (концентраторов) между компьютерами, то есть путь максимальной длины. Если таких путей несколько, то расчет должен производиться для каждого из них.

Расчет в данном случае ведется на основании таблицы 5.

Тип сегмента	Задержка на метр	Макс. задержка
Два абонента TX/FX	-	100
Два абонента T4	-	138
Один абонент T4 и один TX/FX	-	127
Сегмент на кабеле категории 3	1,14	114 (100 м)
Сегмент на кабеле категории 4	1,14	114 (100 м)
Сегмент на кабеле категории 5	1,112	111,2 (100 м)
Экранированная витая пара	1,112	111,2 (100 м)
Оптоволоконный кабель	1,0	412 (412 м)
Репитер (концентратор) класса I	-	140
Репитер (концентратор) класса II с портами TX/FX	-	92
Репитер (концентратор) класса II с портами T4	-	67

Таблица 5. Двойные задержки компонентов сети Fast Ethernet

Для вычисления полного двойного (кругового) времени прохождения для сегмента сети необходимо умножить длину сегмента на величину задержки на метр, взятую из второго столбца таблицы. Если сегмент имеет максимальную длину, то можно сразу взять величину максимальной задержки для данного сегмента из третьего столбца таблицы.

Затем задержки сегментов, входящих в путь максимальной длины, надо просуммировать и прибавить к этой сумме величину задержки для приемопередающих узлов двух абонентов (три верхние строчки таблицы) и величины задержек для всех репитеров (концентраторов), входящих в данный путь (три нижние строки таблицы).

Суммарная задержка должна быть меньше, чем 512 битовых интервалов. При этом надо помнить, что стандарт IEEE 802.3u рекомендует оставлять запас в пределах 1 – 4 битовых интервалов для учета кабелей внутри соединительных шкафов и погрешностей измерения. Лучше сравнивать суммарную задержку с величиной 508 битовых интервалов, а не 512 битовых интервалов.

Все задержки, приведенные в таблице, даны для наихудшего случая. Если известны временные характеристики конкретных кабелей, концентраторов и адаптеров, то практически всегда предпочтительнее использовать именно их. В ряде случаев это может дать заметную прибавку к допустимому размеру сети.

Пример расчета по второй модели для сети, показанной на рисунке 5. Здесь существуют два максимальных пути: между компьютерами (сегменты A, B и C) и между верхним (по рисунку) компьютером и коммутатором (сегменты A, B и D). Оба эти пути включают в себя два 100-метровых сегмента и один 5-метровый. Предположим, что все сегменты представляют собой 100BASE-TX и выполнены на кабеле категории 5. Для двух 100-метровых сегментов (макси-

мальной длины) из таблицы следует взять величину задержки 111,2 битовых интервалов.
Для 5-метрового сегмента при расчете задержки, умножается 1,112 (задержка на метр) на длину кабеля (5 метров): 1,112 * 5 = 5,56 битовых интервалов.
Величина задержки для двух абонентов TX из таблицы – 100 битовых интервалов.
Из таблицы величины задержек для двух репитеров класса II – по 92 битовых интервала.
Суммируются все перечисленные задержки:

$$111,2 + 111,2 + 5,56 + 100 + 92 + 92 = 511,96$$

это меньше 512, следовательно, данная сеть будет работоспособна, хотя и на пределе, что не рекомендуется.

Для гарантии лучше несколько уменьшить длину кабелей или взять кабели, имеющие меньшую задержку. Например, при использовании кабеля AT&T 1061 (NVP = 0,7, tз = 0,477) получаются следующие величины задержек для 100-метровых сегментов: (0,477 * 2) * 100 = 95,4 битовых интервалов (умножение на два необходимо, чтобы получить двойное время прохождения), а для 5-метрового сегмента – 4,77 битовых интервалов. Суммарная задержка при этом составит:

95,4 + 95,4 + 4,77 + 100 + 92 + 92 = 483,57,

то есть гораздо меньше 512 и даже 508, что означает полностью работоспособную сеть.

Пользуясь моделью 2, можно обойти некоторые ограничения модели 1, так как модель 1 строится из расчета на наихудший случай. Например, в сети может присутствовать больше двух концентраторов класса II или больше одного концентратора класса I, а кабель, соединяющий концентраторы, может быть длиннее 5 метров.

На рисунке 6 оказана сеть, содержащая три концентратора класса II, соединенных между собой отрезками кабеля длиной по 10 метров. Компьютеры соединены с концентраторами сегментами 100BASE-TX длиной по 50 метров.

Рисунок 6. Пример работоспособной конфигурации сети, нарушающей правила модели 1

Расчет двойного времени прохождения для этого случая.

1. Каждый из трех концентраторов класса II с портами TX даст задержку 92 битовых интервала. Суммарная задержка концентраторов составит 276 битовым интервалам.

2. Для двух соединительных кабелей между концентраторами задержка равна 2 * 1,112 * 10 = 22,24 битовых интервала.

3. Для двух сегментов TX по 50 метров задержка составит 2 * 1,112 * 50 = 111,2 битовых интервала.

4. Для двух абонентов TX задержка будет равна 100 битовым интервалам.

5. Итого суммарная задержка: 276 + 22,24 + 111,2 + 100 = 509,44 битовых интервала.

Данная сеть работоспособна. Но при этом надо учитывать, что каждый дополнительный концентратор класса II уменьшает общую допустимую длину кабеля на 92/1,112 = 82,7 метра. Сеть с четырьмя концентраторами не будет иметь смысла, так как на задержку в кабеле уже не остается почти никакого запаса (четыре концентратора дадут суммарную задержку в 92 * 4 = 368 битовых интервалов).

Какова будет максимальная величина сети Fast Ethernet? Для этого надо взять сеть с одним концентратором класса II и два сегмента 100BASE-FX. Элементарный расчет показывает, что при одинаковых сегментах длина каждого из них может достигать 160 метров (рисунок 7), а общая длина сети составит 320 метров.

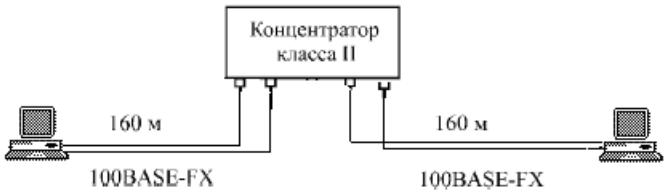


Рисунок 7. Сеть Fast Ethernet максимальной длины

Расчет двойного времени прохождения для этого случая будет выглядеть так:

$$92 + 100 + 2 * 1{,}0 * 160 = 512$$

Получается, что сеть работоспособна, хотя и на пределе. В данном случае важна только суммарная длина обоих кабелей. При уменьшении длины какого-нибудь из сегментов можно без потери работоспособности увеличить на точно такую же величину длину другого сегмента.

Если в приведенной конфигурации используется концентратор класса I, а не концентратор класса II, то допустимая суммарная длина сегментов сокращается с 320 метров до 272 метров (расчет для этого случая очевиден). А согласно стандарту запаса лучше уменьшить суммарную длину кабеля на 1 – 4 метра, что даст снижение круговой задержки на 1 – 4 битовых интервала.

В заключение следует отметить, что модель 2 целесообразно применять в основном при наличии в сети оптоволоконных сегментов. На электрическом кабеле даже при большом желании довольно трудно создать сеть значительного размера.

Методы преодоления ограничений на размер сети в случае Fast Ethernet те же самые, что и в случае Ethernet: сокращение длины кабелей, уменьшение количества концентраторов, выбор марки кабеля с меньшей задержкой, использо-

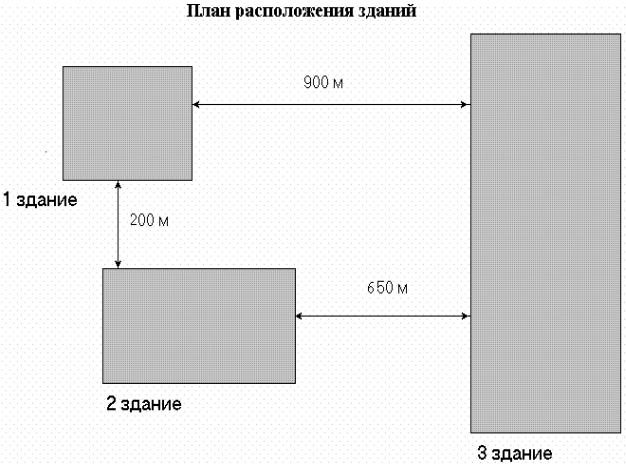
вание коммутаторов, переход на полнодуплексный режим обмена, а также переход на другую сеть (например, FDDI).

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).

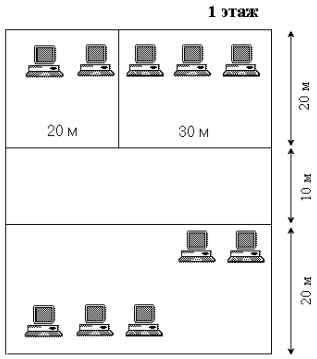
Вариант 1

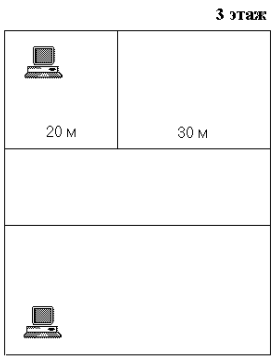


Первое из зданий трехэтажное. Компьютеры расположены в отделах на первом и третьем этажах. Общее число компьютеров – 12 (10 на первом этаже и 2 на втором).

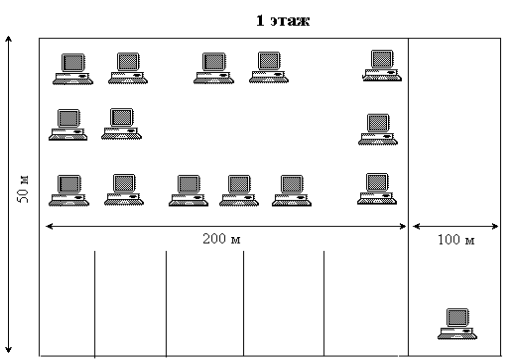
Второе и третье здания одноэтажные. В каждом из них – 15 компьютеров.

План первого здания



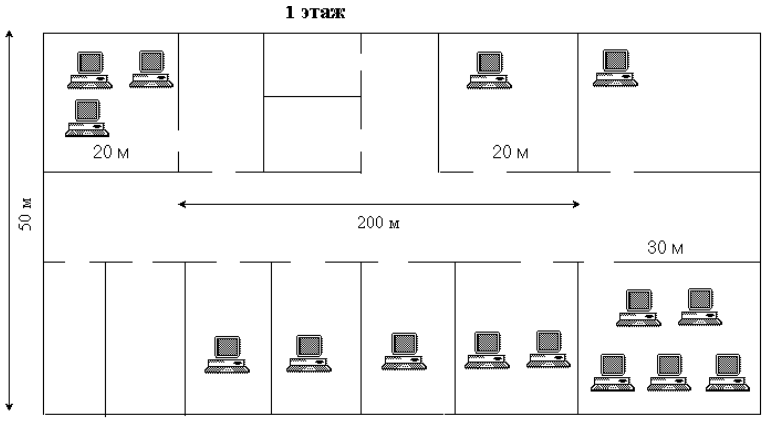


План третьего здания

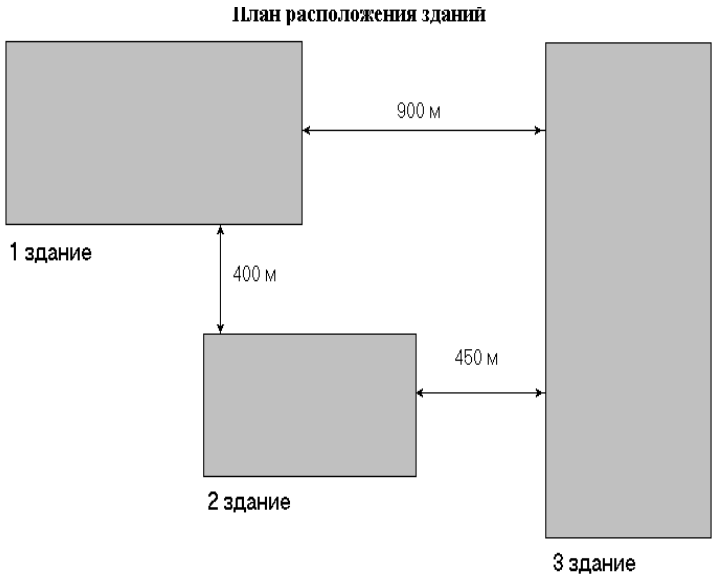


План второго зда-

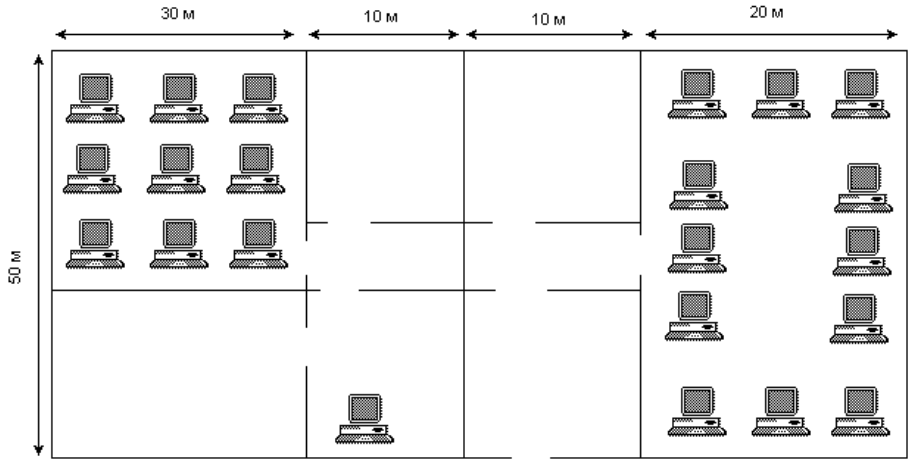
ния



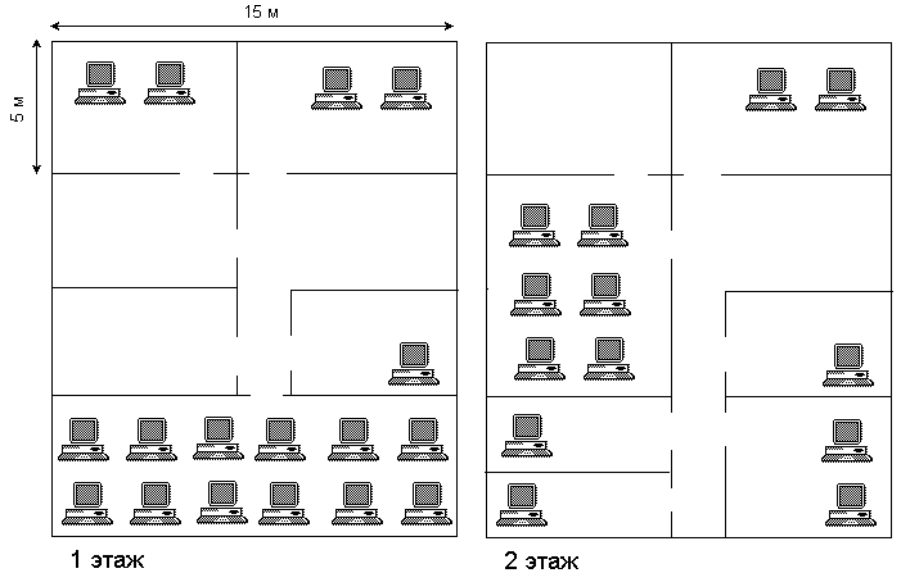
Вариант 2



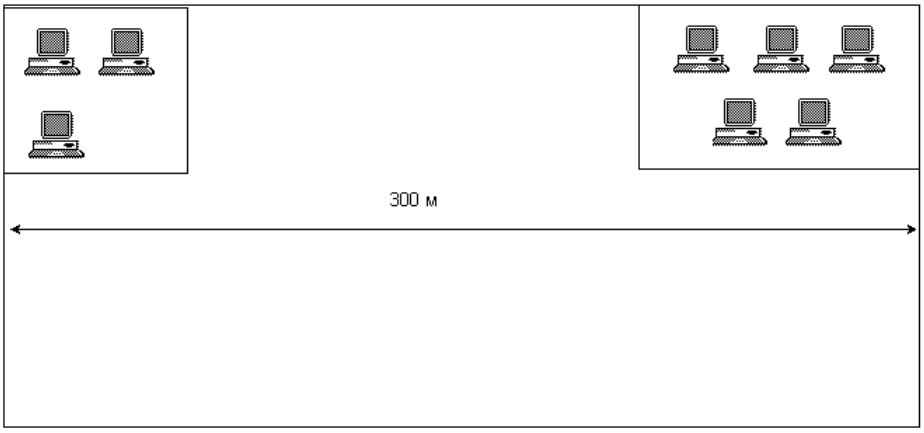
План первого здания



План второго здания

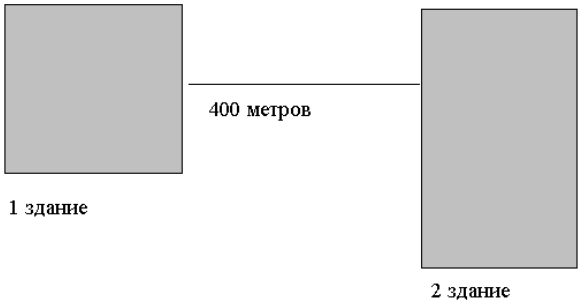


План третьего здания

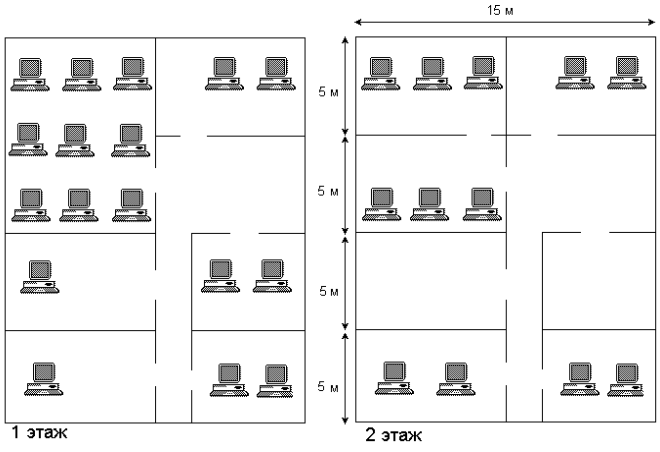


ВАРИАНТ 3

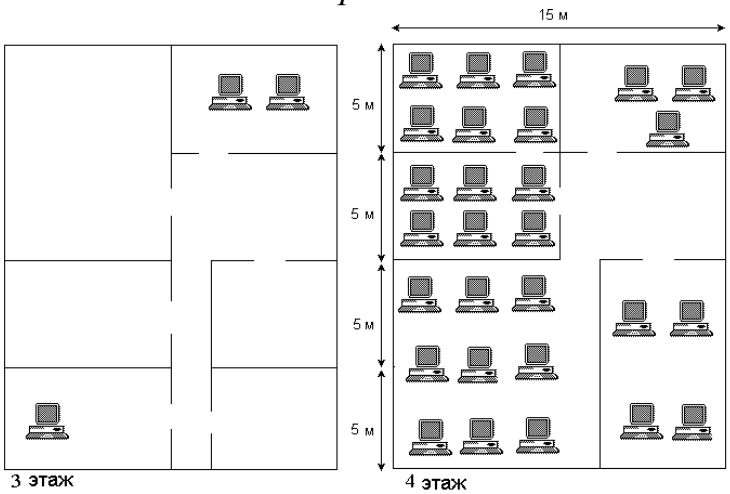
План расположения зданий



План первого здания

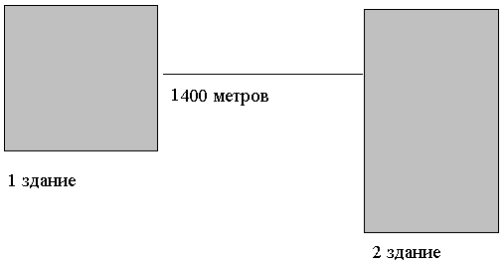


План второго здания

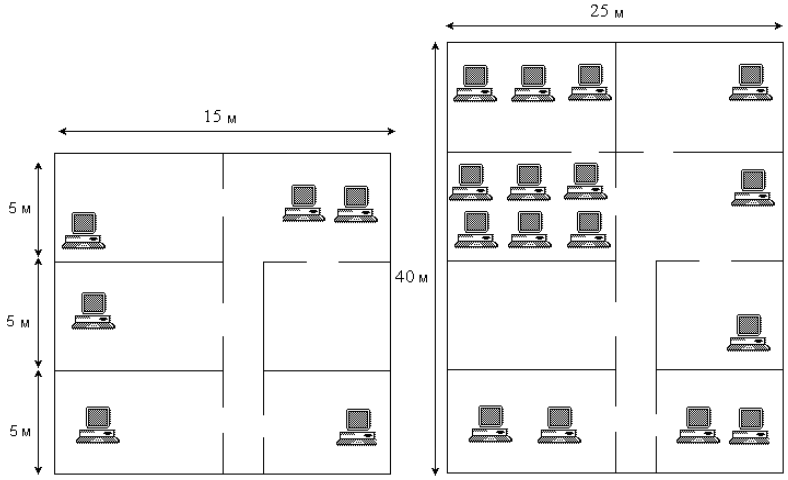


ВАРИАНТ 4

План расположения зданий

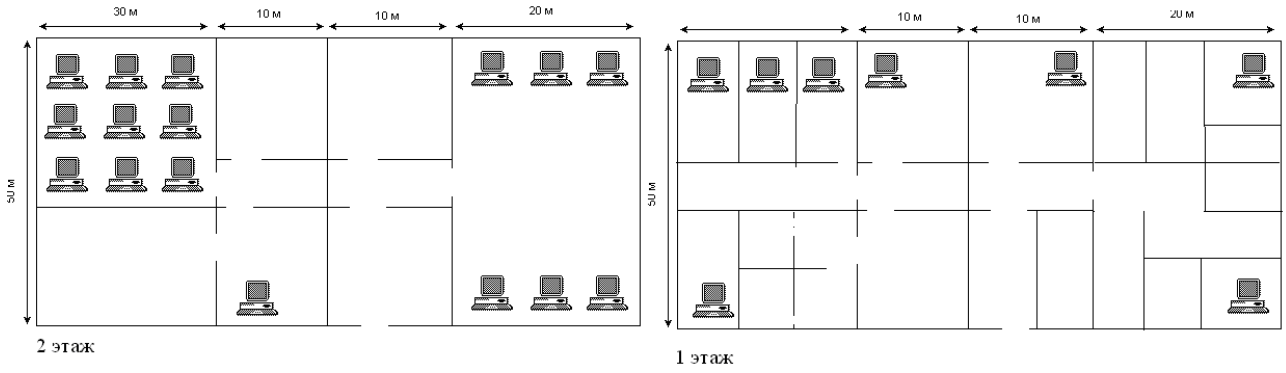


Планы зданий

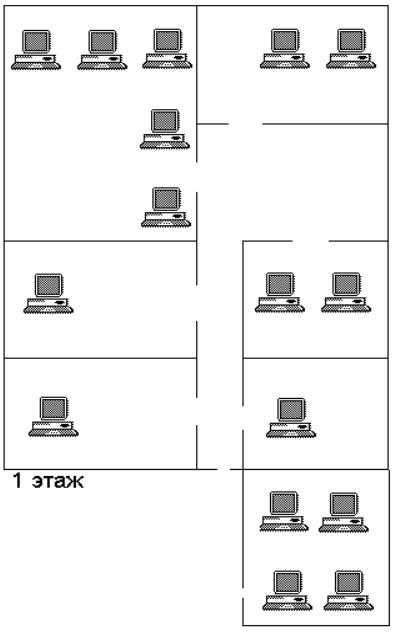


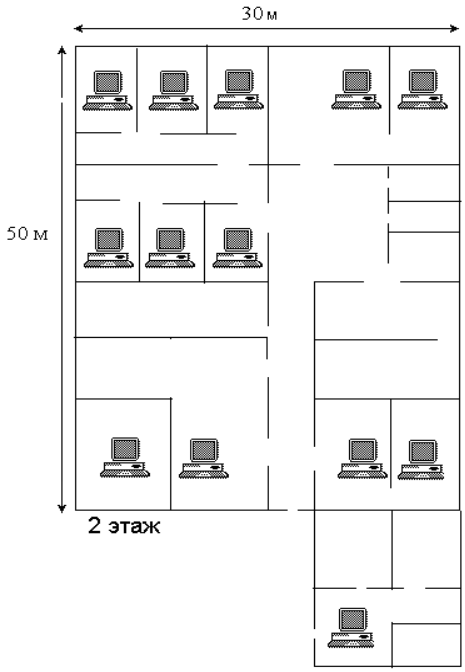
ВАРИАНТ 5

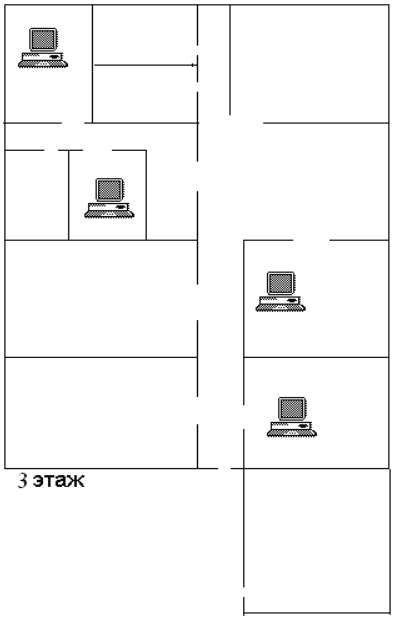
План здания

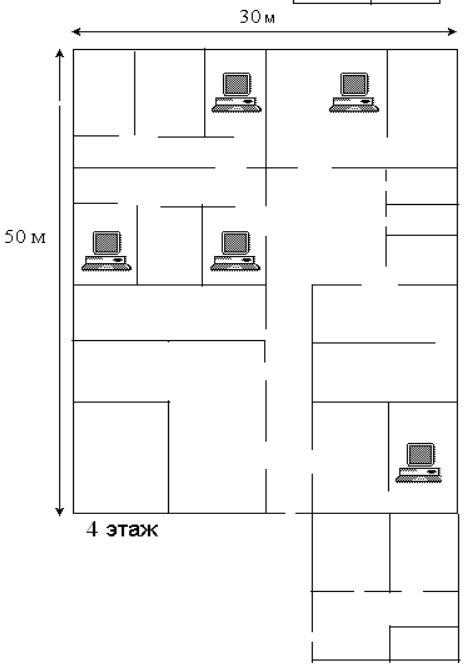


ВАРИАНТ 6









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.

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.

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 и 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 сети 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.

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 приори-

тетами, 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.

ПРИНЦИПЫ ВЗАИМОДЕЙСТВИЯ КОМПЬЮТЕРОВ В СЕТИ

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

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.

ВЗАИМОДЕЙСТВИЕ ДВУХ КОМПЬЮТЕРОВ

В самом простом случае взаимодействие компьютеров может быть реализовано с помощью тех же самых средств, которые используются для взаимо-

действия 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 могут 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.

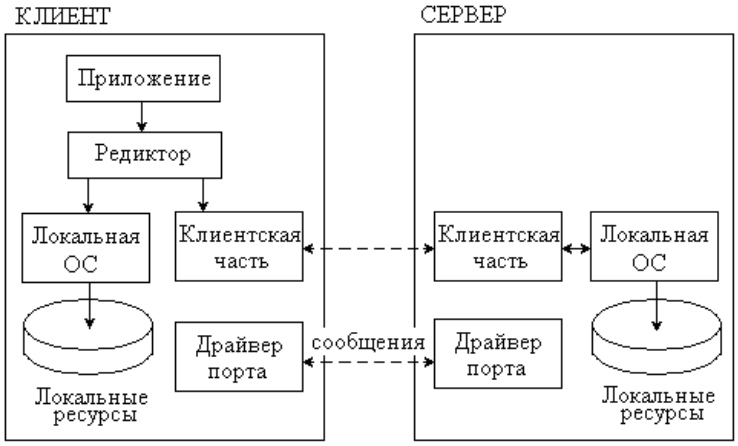


Рис. 1.3. Взаимодействие клиента и сервера

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) за-

прос 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.

ФУНКЦИОНАЛЬНЫЕ ГРУППЫ УСТРОЙСТВ СЕТИ

Узел – любое устройство, подключенное к сети. Это и рабочая станция, и сервер, и повторитель, одним словом все то, что непосредственно подключено к передающей среде сети.

Рабочая станция – это персональный компьютер, подключенные к сети, на котором пользователь сети выполняет свою работу. Каждая рабочая станция обрабатывает свои файлы и использует свою операционную систему. Пользователю на рабочей станции доступны ресурсы сети.

Удаленная рабочая станция – рабочая станция, подключенная к локальной сети через медленную линию связи, отличную от используемой в локальной сети (например, телефонную линию).

Сервер сети – компьютер, подключенный к сети и выполняющий для пользователей сети определенные услуги. Например, хранение данных общего пользования, печать заданий, удаленную обработку заданий. По выполняемым функциям можно выделить следующие группы серверов:

Файловый сервер – компьютер, хранящий данные пользователей сети и обеспечивающий доступ пользователей к этим данным. Как правило, это компьютер с жесткими дисками большой емкости. На таком компьютере обычно используется специальная операционная система, обеспечивающая одновременный доступ пользователей сети к данным, расположенным на нем.

Файловый сервер выполняет следующие функции:

1) хранение данных;
2) архивирование данных;
3) согласование изменений данных, выполняемых разными пользователями;
4) передача данных.

Для многих задач использование в сети только файлового сервера бывает недостаточно.

Сервер прикладных программ – компьютер, который используется для решения прикладных программ пользователей.

Сервер баз данных (SQL-Server) компьютер, выполняющий функции хранения, обработки и управления файлами баз данных. Он выполняет следующие функции:

1) хранение, поиск и обновление записей баз данных;
2) обеспечение секретности данных;
3) согласование изменений данных, выполняемых разными пользователями;
4) взаимодействие с другими серверами баз данных, расположенными в другом месте.

Коммуникационный сервер – устройство или компьютер, который предоставляет пользователям локальной сети прозрачный доступ к последовательным портам ввода/вывода. С помощью коммуникационного сервера можно создать разделяемый модем, подключив его к одному из портов сервера.

Сервер доступа – устройство или компьютер, позволяющий выполнять удаленную обработку заданий. Это стойка, в которую устанавливаются системные платы с модемами и сетевыми платами по количеству пользователей, работающих в сети на удаленных рабочих станциях. Либо компьютер, к которому подключены модемы для связи с удаленными рабочими станциями и на котором выполняется многопользовательская операционная система.

Сервер печати – устройство или компьютер, к которому подключены устройства печати, доступные пользователям сети.

Факс-сервер – устройство или компьютер, который выполняет рассылку и прием факсимильных сообщений для пользователей локальной сети.

Сервер резервного копирования данных, который решает задачи создания, хранения и восстановления копий данных, расположенных на файловых серверах и рабочих станциях. В качестве такого сервера могут использоваться один из файловых серверов сети, рабочая станция либо специализированный модуль, подключаемый непосредственно к локальной сети.

Протяженность сети, расстояние между станциями, в первую очередь определяется физическими характеристиками передающей среды. При передаче данных в любой среде происходит затухание сигнала, что и приводит к ограничению расстояния. Установив специальное коммуникационное устройство, усиливающее сигнал, можно значительно расширить сеть. Такими устройствами являются повторители, мосты, коммутаторы, маршрутизаторы и шлюзы. Часть сети, в которую не входит устройство расширения, принято называть *сегментом* сети.

Повторитель (repeater) - устройство, позволяющее расширить сеть подключением дополнительных сегментов кабеля. Повторитель, приняв пакет из одного сегмента, передает его во все остальные. При этом происходит как бы "усиление" сигнала. Устройства, выполняющие функции повторителя, часто называют хабами (hub) или концентраторами (concentrator).

Мост (bridge) - устройство, позволяющее объединить несколько сегментов, так что передача данных между станциями внутри одного сегмента не будет влиять на передачу данных в других сегментах. Это обеспечивается фильтрацией передаваемых данных.

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.

Мультиплексор (multiplexor или MUX) - устройство, позволяющее мультиплексировать данные, приходящие одновременно от различных станций или

сегментов и передавать их через передающую среду, например телефонную линию, другим станциям, сегментам или мультиплексорам. Мультиплексоры позволяют более эффективно использовать пропускную способность линий связи.

В случае, когда необходимо объединить несколько сетей, используются устройства межсетевого взаимодействия. К ним относятся маршрутизаторы и шлюзы.

Маршрутизатор (router) - устройство, соединяющее сети разного типа, но использующие одну сетевую операционную систему или протокол обмена данными.

Шлюз (gateway) устройство, позволяющее организовать обмен данными между сетевыми объектами, использующими различные протоколы обмена данными. Шлюз выполняет свои функции на уровнях выше сетевого. Он не зависит от используемой передающей среды, но зависит от используемых протоколов обмена данными. Как правило, шлюз выполняет преобразования между какими-либо двумя протоколами.

ТОПОЛОГИИ ПОДКЛЮЧЕНИЯ УСТРОЙСТВ

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.

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

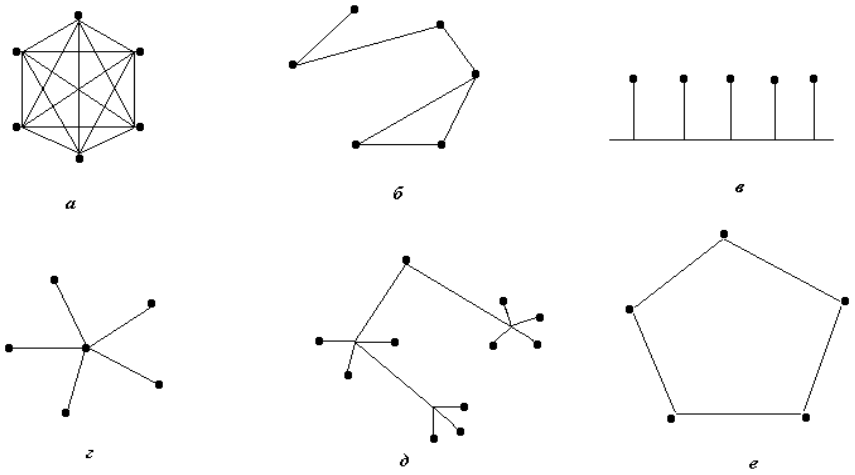


Рис. 1.4. Типовые топологии сетей

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. 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.

Выделяют также сотовую (концентрическую) топологию. Она рассматривается как топология подключения устройств в беспроводных сетях. Сотовая топология метод разделения географической области на зоны, в каждой из которых обеспечивается обмен информацией между станциями сети, находящимися внутри зоны (рис.1.5). В каждой зоне могут устанавливаться ретрансляторы, позволяющие обмениваться информацией станциям, 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.

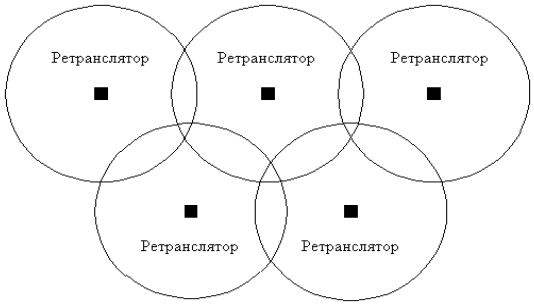


Рис.1.5. Сотовая, концентрическая топология

ЛОГИЧЕСКАЯ И ФИЗИЧЕСКАЯ СТРУКТУРИЗАЦИЯ СЕТИ

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.

Физическая структуризация сети

Простейшее из коммуникационных устройств – повторитель – 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.

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 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 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. 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.

Кроме перечисленных устройств отдельные части сети может соединять шлюз. 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 структу-

ризации. 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.

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 (Open System Interconnection – OSI) (1977 г). Одна из основных ее идей – обеспечение относительно легкого и простого обмена 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.

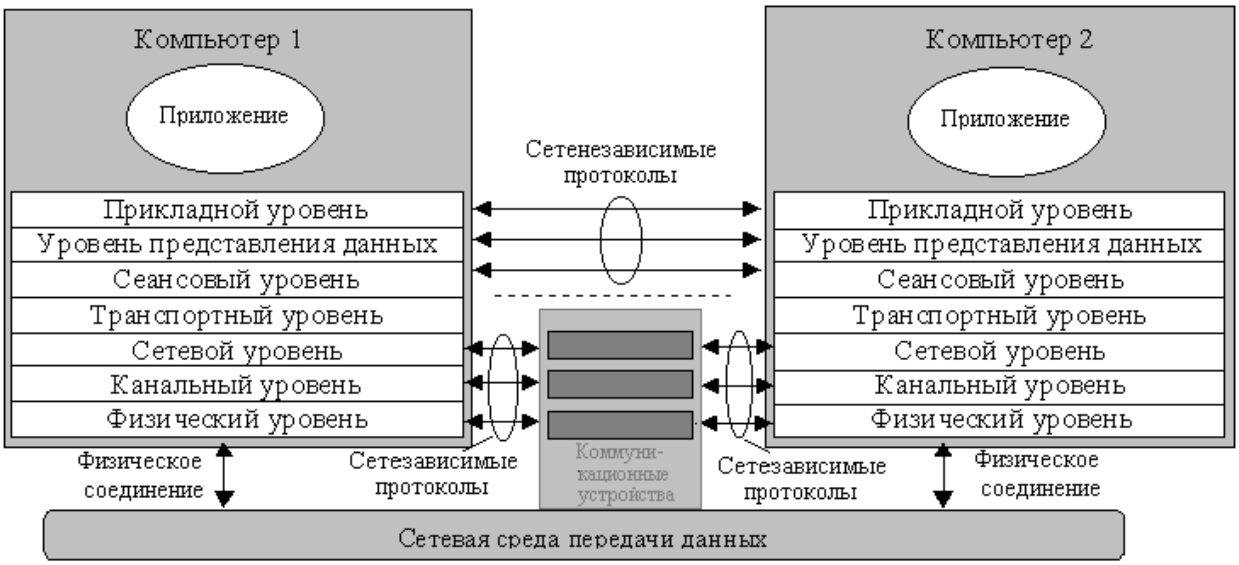


Рис. 1. 7. Модель взаимосвязи открытых систем

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 стан-

дартного 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 служебную 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

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. 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

Уровень представления данных отвечает за физическое отображение (представление) информации, не меняя при этом ее содержание. Уровень выполняет следующие функции: преобразование форматов данных и кодирование/декодирование данных, в том числе компрессию и декомпрессию данных.

Форматы представления данных могут различаться по следующим признакам:

порядок следования битов и размерность символа в битах;

порядок следования байтов;

представление или кодировка символов;

структура и синтаксис файлов.

СЕАНСОВЫЙ УРОВЕНЬ

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

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

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

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

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 характеристик. Не всякая спецификация является стандартом. Открытые специ-

фикации – 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.

БАЗОВЫЕ ТЕХНОЛОГИИ ЛОКАЛЬНЫХ СЕТЕЙ

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.

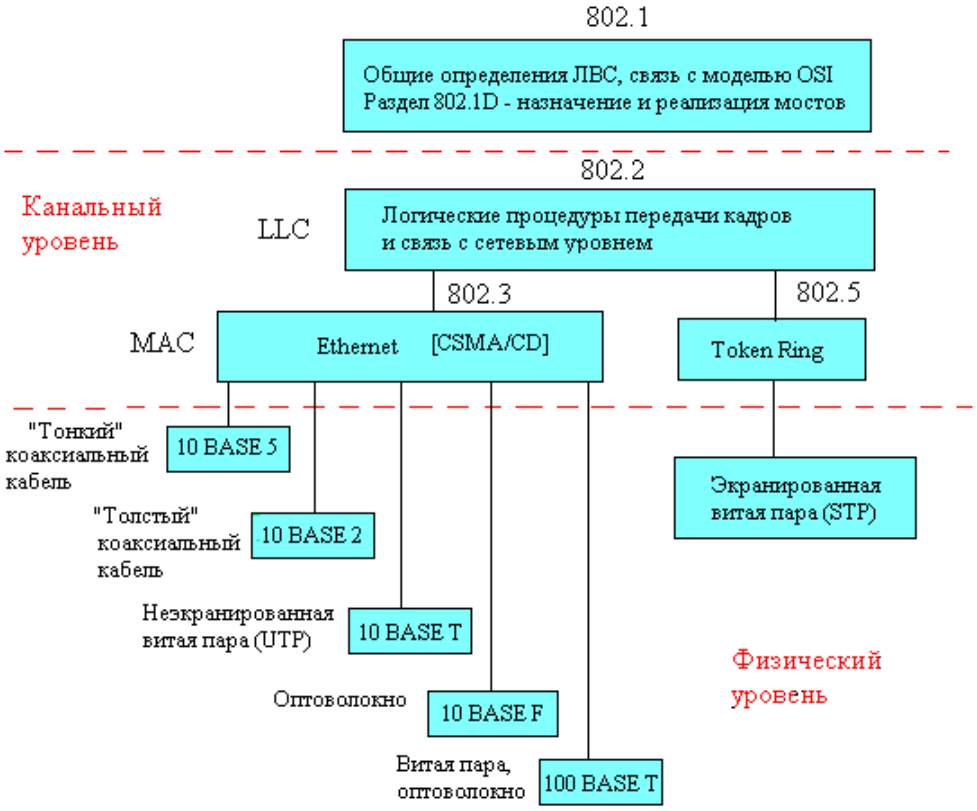


Рис. 3.1 Структура стандартов 802.x

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.

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.

ЛОКАЛЬНАЯ СЕТЬ Ethernet

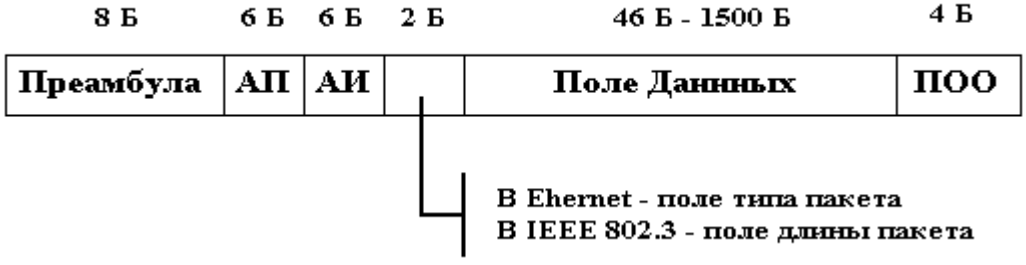
Ethernet – самый распространенный в настоящее время стандарт локальных сетей.

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 СЕТИ 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).

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.

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	3600
Максимальное число станций на сегменте	100	30	1024	1024	100
Максим. Число повторителей Между любыми станциями сети	4	4	4	4	4
Тип кабеля	50 Ом-ный	50 Ом-ный	Неэкрани-рованная	Многомодо-вый 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 (Attechment Unit Interface) длиной до 50 метров, состоящий из 4 витых пар. Трансивер устанавливается непосредственно на кабеле и питается от сетевого адаптера компьютер. 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.

На концах сегмента должны устанавливаться заглушки – терминаторы с сопротивлением 50 Ом, поглощающие распространяющиеся по кабелю сигналы и препятствующие возникновению отраженных сигналов.

Различные компоненты сети, состоящей из трех сегментов, соединенных повторителями, показаны на рис. 3.4.

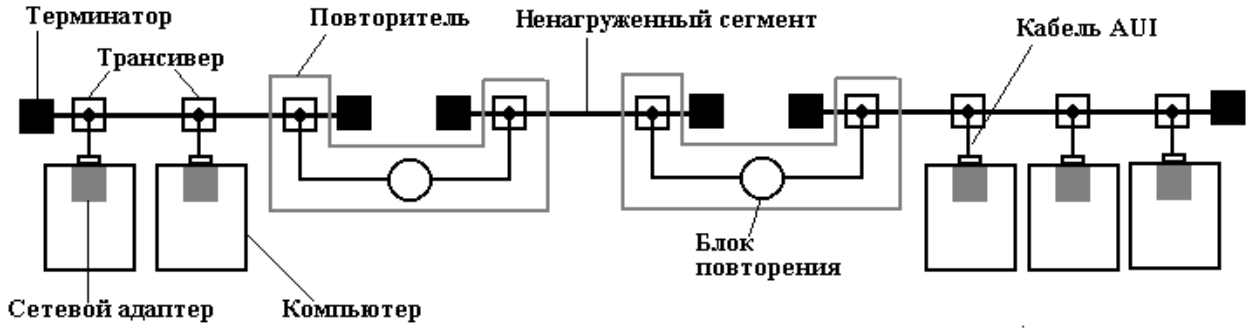


Рис. 3.4. компоненты физического уровня сети стандарта 10Base-5

К достоинствам 10Base-5 относятся:
- хорошая защищенность кабеля от внешних воздействий;

- сравнительно большое расстояние между узлами;
- возможность простого перемещения рабочей станции в пределах длины кабеля AUI.

Недостатками 10Base-5 являются:

- высокая стоимость кабеля;
- сложность его прокладки из-за большой жесткости;
- потребность в специальном инструменте для заделки кабеля;
- останов работы всей сети при повреждении кабеля или плохом соединении.

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.

На практике для реализации данного стандарта требутся только сетевые адаптеры, Т-коннекторы и терминаторы (рис. 3.5.)

Рис. 3.5. Сеть стандарта 10Base-2

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».

Для обеспечения максимального числа станций в сети (1024) создают двухуровневую иерархию, расположив на нижнем уровне достаточно большое число концентраторов с общим числом портов 1024. Правило 4-х хабов при этом выполняется (рис. 3.6.).

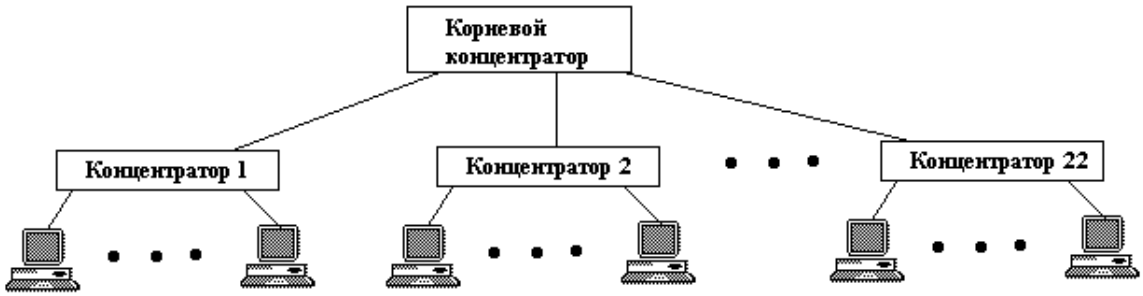


Рис. 3.6. Иерархия концентраторов Ethernet

Данный стандарт обладает по сравнению с коаксиальными вариантами многими преимуществами. Физическое разделение станций позволяет контролировать их состояние и отключать в случае обрыва, что повышает эксплуатацию больших сетей.

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.

10BROAD36 — широкого распространения не получил. Один из первых стандартов, позволяющий работать на больших расстояниях. Использовал технологию широкополосной модуляции, похожей на ту, что используется в кабельных модемах. В качестве среды передачи данных использовался коаксиальный кабель.

Спецификация 10Broad36 использовала такую же схему подключения абонентов к среде передачи, какая была использована в 10Base5. Длина кабеля AUI в данном случае могла достигать 50 метров.

Применение технологии 10Broad36 обеспечивало следующие преимущества:
использование существующих коммуникаций;
увеличение дальности информационного взаимодействия.

С появлением и широким внедрением оптоволоконных линий эти преимущества были нивелированы и технология 10Broad36 не получила дальнейшего развития.

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. Первой эту концепцию внедрила фирма Kalpana, за ней последовали Alantec Artel. 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. А поскольку связь клиент-сервер в большинстве случаев является односторонней, то чаще всего общая производительность оказывается ниже ожидаемой. Данную технологию внедрила фирма Kalpana в конце 1993 года.

100BaseVG AnyLAN. Скорость передачи – 100 Мбит/с. 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. Хотя данная технология и поддерживает форматы кадры двух форматов Ethernet и Token Ring.

100BaseX или *Fast Ethernet.* Предложение фирмы Grand Junction реализуется с помощью дуплексной передачи сигналов по традиционной для Ethernet схеме доступа с контролем несущей и обнаружением коллизий в комбинации с уровнем зависимости от физической среды.

Технологию *Fast Ethernet* называют быстрый Ethernet. Она принята в 1995 году в качестве дополнения 802.3u к существующему стандарту 802.3

Более сложная структура физического уровня технологии Fast Ethernet вызвана тем, что в ней используется три варианта кабельных систем:

Волоконно-оптический кабель многомодовый кабель (используются 2 волокна),

2-х парная витая пара категории 5;

4-х парная витая пара категории 3.

Fast Ethernet всегда имеет иерархическую древовидную структуру, построенную на концентраторах, подобно *10BaseT* и *10BaseF*.

Диаметр сети сокращен до 200 метров, что объясняется уменьшением времени передачи кадра минимальной длины.

При использовании коммутаторов протокол *Fast Ethernet* может работать в полнодуплексном режиме, в котором нет ограничения на длину сети, а существуют лишь ограничения на длину физических сегментов.

Формат кадра не отличается от формата 10-мегабитного *Ethernet*.

По сравнению с вариантами физической реализации классической Ethernet здесь отличия каждого варианта от других глубже, поскольку меняется и количество проводников, и методы кодирования.

100BASE-T - общий термин для обозначения стандартов, использующих в качестве среды передачи данных витую пару. Длина сегмента до 100 метров. Включает в себя стандарты 100BASE-TX, 100BASE-T4 и 100BASE-T2.

100BASE-TX, IEEE 802.3u – развитие стандарта 10BASE-T для использования в сетях топологии "звезда". Задействована витая пара категории 5, фактически используются только две неэкранированные пары проводников, поддерживается дуплексная передача данных, расстояние до 100 м.

100BASE-T4 – стандарт, использующий витую пару категории 3. Задействованы все четыре пары проводников, передача данных идёт в полудуплексном режиме. Максимальная длина сегмента 100 м. Практически не используется.

100BASE-T2 - стандарт, использующий витую пару категории 3. Задействованы только две пары проводников. Поддерживается полный дуплекс, когда сигналы распространяются в противоположных направления по каждой паре. Скорость передачи в одном направлении — 50 Мбит/с. Практически не используется.

100BASE-SX – стандарт, использующий многомодовый волоконно-оптический кабель. Максимальная длина сегмента 400 метров в полудуплексном режиме (для гарантированного обнаружения коллизий) или 2 км в полнодуплексном режиме.

100BASE-FX – стандарт, использующий одномодовый волоконно-оптический кабель. Максимальная длина ограничена только величиной затухания в оптоволоконном кабеле и мощностью передатчиков.

100BASE-FX WDM – стандарт, использующий одномодовый волоконно-оптический кабель. Максимальная длина ограничена только величиной затухания в оптоволоконном кабеле и мощностью передатчиков. Интерфейсы бывают двух видов, отличаются длиной волны передатчика и маркируются либо цифрами (длина волны) либо одной латинской буквой A(1310) или B(1550). В паре могут работать только парные интерфейсы: с одной стороны передатчик на 1310 нм, а с другой — на 1550 нм.

Более высокая скорость работы в сетях Fast Ethernet обусловлена более быстрыми средами передачи данных, а не значительными изменениями стандартов кадров или методов их передачи, применение как полудуплексной передачи данных, так и дуплексной.

При определение корректности конфигурации сети можно не руководствоваться правилами использования хабов, а рассчитать время двойного оборота сети.

Гигабит Ethernet – поддерживает скорость обмена до 1 Гбит/с. Сеть Gigabit Ethernet представляет собой ЛВС, в которой применяется многомодовый оптоволоконный кабель, но IEEE изучает методы ее применения на сетях с UTP 5 категории.

Топология звездообразная. Оптоволоконный кабель используется для магистрали, коаксиальный кабель для отводов к концентраторам.

1000BASE-T, IEEE 802.3ab — стандарт, использующий витую пару категорий 5e. В передаче данных участвуют все 4 пары. Скорость передачи данных — 250 Мбит/с по одной паре. Используется метод кодирования PAM5, частота основной гармоники 62,5 МГц.

1000BASE-TX был создан Ассоциацией Телекоммуникационной Промышленности. Стандарт, использует раздельные приём и передачу (1 пару на передачу, 1 пару на приём, по каждой паре данные передаются со скоростью 500 Мбит/с), что существенно упрощает конструкцию приёмопередатчиков. Но, как следствие, для стабильной работы по такой технологии требуется кабельная система высокого качества, поэтому 1000BASE-TX может использовать только кабель 6 категории. Ещё одним существенным отличием 1000BASE-TX является отсутствие схемы цифровой компенсации наводок и возвратных помех, в результате чего сложность, уровень энергопотребления и цена процессоров становится ниже, чем у процессоров стандарта 1000BASE-T. На основе данного стандарта практически не было создано продуктов, хотя 1000BASE-TX использует более простой протокол, чем стандарт 1000BASE-T, и поэтому может использовать более простую электронику.

1000BASE-X — общий термин для обозначения стандартов со сменными приёмопередатчиками GBIC или SFP.

1000BASE-SX, IEEE 802.3z — стандарт, использующий многомодовое оптоволокно. Дальность прохождения сигнала без повторителя до 550 метров.

1000BASE-LX, IEEE 802.3z — стандарт, использующий одномодовое оптоволокно. Дальность прохождения сигнала без повторителя до 80 километров.

1000BASE-CX — стандарт для коротких расстояний (до 25 метров), использующий твинаксиальный кабель с волновым сопротивлением 150 Ом. Данные посылаются одновременно по паре проводников, каждый из которых окружен экранирующей оплеткой. Полудуплесная передача. Максимальная длина твинкоаксиального сегмента всего 25м, т.е. для оборудования, расположенного в одной комнате. Заменён стандартом 1000BASE-T и сейчас не используется.

1000BASE-LH (Long Haul) — стандарт, использующий одномодовое оптоволокно. Дальность прохождения сигнала без повторителя до 100 километров.

Новый стандарт 10 Гигабит Ethernet включает в себя семь стандартов физической среды для LAN, MAN и WAN. В настоящее время он описывается поправкой IEEE 802.3ae и должен войти в следующую ревизию стандарта IEEE 802.3.

10GBASE-CX4 — Технология 10 Гигабит Ethernet для коротких расстояний (до 15 метров), используется медный кабель CX4 и коннекторы InfiniBand.

10GBASE-SR — Технология 10 Гигабит Ethernet для коротких расстояний (до 26 или 82 метров, в зависимости от типа кабеля), используется многомодовое оптоволокно. Он также поддерживает расстояния до 300 метров с использованием нового многомодового оптоволокна (2000 МГц/км).

10GBASE-LX4 — использует уплотнение по длине волны для поддержки расстояний от 240 до 300 метров по многомодовому оптоволокну. Также поддерживает расстояния до 10 километров при использовании одномодового оптоволокна.

10GBASE-LR и 10GBASE-ER — эти стандарты поддерживают расстояния до 10 и 40 километров соответственно.

10GBASE-SW, 10GBASE-LW и 10GBASE-EW – эти стандарты используют физический интерфейс, совместимый по скорости и формату данных с интерфейсом OC-

192 / STM-64 SONET/SDH. Они подобны стандартам 10GBASE-SR, 10GBASE-LR и 10GBASE-ER соответственно, так как используют те же самые типы кабелей и расстояния передачи.

10GBASE-T, IEEE 802.3an-2006 — принят в июне 2006 года после 4 лет разработки. Использует экранированную витую пару. Расстояния — до 100 метров.

Стандарт 10 Гигабит Ethernet ещё слишком молод, поэтому потребуется время, чтобы понять, какие из вышеперечисленных стандартов передающих сред будут реально востребованы на рынке.

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).

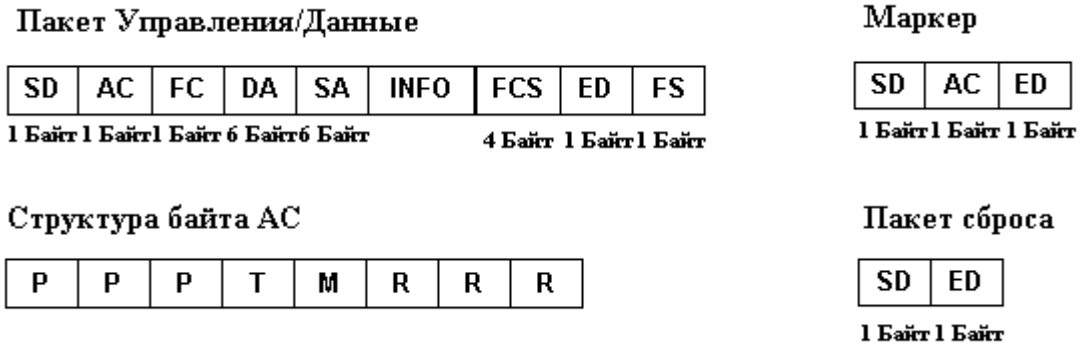


Рис. 3.4. Типы пакетов, используемые в IBM Token Ring

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.

РАСПРЕДЕЛЕННЫЙ ВОЛОКОННО-ОПТИЧЕСКИЙ ИНТЕРФЕЙС ПЕРЕДАЧИ ДАННЫХ (FDDI)

Предложен 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.

СЕТЕВЫЕ ОПЕРАЦИОННЫЕ СИТСТЕМЫ

Сетевая операционная система – операционная система, обладающая встроенными возможностями для работы в компьютерных сетях. Она обеспечивает поддержку сетевого оборудования, сетевых протоколов, протоколов маршрутизации, фильтрации сетевого трафика, доступа к удалённым ресурсам и пр.

Примерами сетевых операционных систем служат
Novell NetWare, Microsoft Windows (95, NT и более поздние), различные UNIX системы, такие как Solaris, FreeBSD, различные GNU/Linux системы и др.

Главными задачами сетевых ОС являются разделение ресурсов сети и администрирование сети. С помощью сетевых функций системный администратор определяет разделяемые ресурсы, задаёт пароли, определяет права доступа для каждого пользователя или группы пользователей. Различают сетевые ОС для серверов и сетевые ОС для пользователей.

Существуют специальные сетевые ОС, которым приданы функции обычных систем (Windows NT) и обычные ОС (Windows XP), которым приданы сетевые функции. Сегодня практически все современные ОС имеют встроенные сетевые функции.

VI. МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ К ПРОВЕДЕНИЮ ПРАКТИЧЕСКИХ ЗАНЯТИЙ

Типы и характеристики 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.

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

проводные (воздушные),

кабельные (медные и волоконно-оптические),

радиоканалы наземной и спутниковой связи.

ВИТАЯ ПАРА

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.

КОАКСИАЛЬНЫЙ КАБЕЛЬ

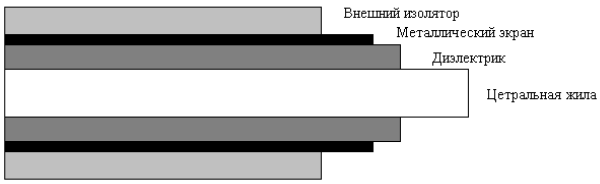


Рис. Коаксиальный кабель

ВОЛОКОННО-ОПТИЧЕСКИЙ КАБЕЛЬ

Состоит из тонких 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.

В компьютерных сетях применяются практически все описанные типы передающей среды, но наиболее перспективными являются волоконно-оптические. На их основе строятся как магистрали крупных территориальных сетей, так и высокоскоростные линии связи локальных сетей. Популярной является и витая пара за счет

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.

Затухание определяется как относительное уменьшение амплитуды или мощности сигнала при передаче определенной частоты. Таким образом, затухание представляет собой одну точку из АЧХ. При эксплуатации линии заранее известна частота передаваемого сигнала, поэтому достаточно знать затухание на этой частоте, чтобы оценит искажение сигнала.

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.

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.

Соотношение Найквиста 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. Помехоустойчивость зависит от типа используемой физической среды. Наименее помехоустойчивыми являются радиолинии, наиболее – оптоволокно.

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. Если ко второй паре будет подключен приемник, то он может принять наведенную внутреннюю помеху за полезный сигнал. Показатель NEXT, выраженный в децибелах, равен 10 10 lg ($P_{вых}/P_{нав}$), где $P_{вых}$ - мощность выходного сигнала, $P_{нав}$ – мощность наведенного сигнала.

Стандарты кабелей

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.

Эти стандарты близки между собой и по многим позициям имеют идентичные требования. Однако имеются и отличия, так в международный и европейский стандарты вошли типы кабелей, отсутствующие в EIA/TIA.

Кроме этих открытых стандартов, многие компании разработали свои фирменные стандарты, из которых только один в настоящее время имеет практическое значение – стандарт компании IBM.

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.

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.

Наиболее важные электромагнитные характеристики кабеля 5 категории имеют следующие значения:

- полное волновое сопротивление равно 100 Ом (в стандарте iso 11801 допускается кабель с волновым сопротивлением 120 Ом)

- NEXT в зависимости от частоты сигнала принимает значения не менее 74 дБ на частоте 150 кГц и не менее 32 дБ на частоте 100 МГц

- Затухание имеет предельные значения от 0,8 дБ (на частоте 64 кГц) до 22 дБ (на частоте 100 МГц)

- Емкость кабеля не должна превышать 5,6 на 100м

Каждая из категорий имеет свою граничную частоту полосы пропускания сигнала, что и определяет возможность использования того или иного типа кабеля в качестве среды передачи данных для локальной сети. Для уровня 3 - это 10 MHz, уровня 4 -20 MHz и уровня 5 - 100 MHz. Таким образом, для локальной сети со скоростью передачи данных 10 Мбит/с можно использовать

типы кабеля: 3, 4, 5, но для сети со скоростью передачи данных 100 Мбит/с только витую пару уровня 5.

Все кабели UTP независимо от категории выпускаются в 4-парном исполнении. Каждая из четырех пар кабеля имеет определенный цвет и шаг скрутки. Обычно 2 пары предназначены для передачи данных, 2 – для передачи голоса.

Недостатками витой пары являются высокий коэффициент затухания сигнала и высокая чувствительность к электромагнитным помехам, поэтому максимальное расстояние между активными устройствами в локальной сети при использовании витой пары до 100 метров.

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.

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.

Основными типами коаксиального кабеля 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).

- 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.

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:

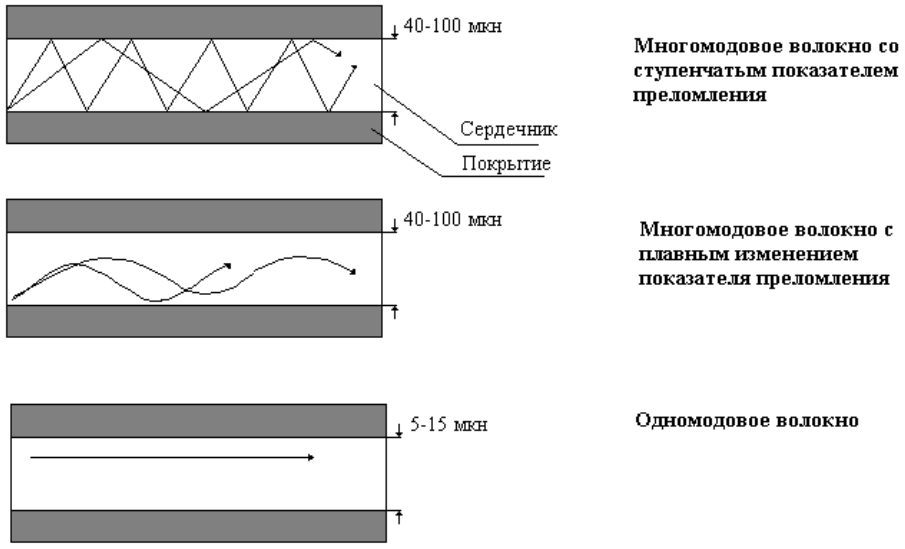


Рис. Типы оптического кабеля

Задание

1. Изучить стандарты передающей среды
2. Определите пропускную способность канала связи для каждого из направлений дуплексного режима, если известно, что его полоса пропускания равна 600 кГц, а в методе кодирования используется 10 состояний сигнала.
3. Каким будет теоретический предел скорости передачи данных по каналу с шириной пропускания в 20кГц, если мощность передатчика составляет 0,01 мВт, а мощность шума 0,0001 мВт?

Адресация узлов сети

Адреса могут быть числовыми (например, 129.26.255.255) и символьными (site.domain.ru). Один и тот же адрес может быть записан в разных форматах, скажем, числовой адрес в предыдущем примере 129.26.255.255 может быть записан и в шестнадцатеричном формате цифрами — 81.1a.ff.ff.

Адреса могут использоваться для идентификации не только отдельных интерфейсов, но и их групп (групповые адреса). С помощью групповых адресов данные могут направляться сразу нескольким узлам. Во многих технологиях компьютерных сетей поддерживаются так называемые широковещательные адреса.

Множество всех адресов, которые являются допустимыми в рамках некоторой схемы адресации, называется адресным пространством. Адресное пространство может иметь плоскую (линейную) или иерархическую организацию. В первом случае множество адресов никак не структурировано.

При иерархической схеме адресации оно организовано в виде вложенных друг в друга подгрупп, которые, последовательно сужая адресуемую область, определяют отдельный сетевой интерфейс.

Трехуровневая структура адресного пространства, при которой адрес конечного узла задается тремя составляющими: идентификатором группы (K), в которую входит данный узел, идентификатором подгруппы (L) и, наконец, идентификатором узла (n), однозначно определяющим его в подгруппе. Иерархическая адресация во многих случаях оказывается более рациональной, чем плоская. В больших сетях, состоящих из многих тысяч узлов, использование плоских адресов может привести к большим издержкам — конечным узлам и коммуникационному оборудованию придется работать с таблицами адресов, состоящими из тысяч записей. А иерархическая система адресации позволяет при перемещении данных до определенного момента пользоваться только старшей составляющей адреса, затем для дальнейшей локализации адресата следующей по старшинству частью, и в конечном счете — младшей частью.

Примером плоского числового адреса является MAC-адрес, используемый для однозначной идентификации сетевых интерфейсов в локальных сетях. Такой адрес обычно применяется только аппаратурой, поэтому его стараются сделать по возможности компактным и записывают в виде двоичного или шестнадцатеричного значения, например 0081005e24a8. Когда задаются MAC-адреса, вручную ничего делать не нужно, так как они обычно встраиваются в аппаратуру компанией-изготовителем; их называют еще аппаратными (hardware) адресами. Использование плоских адресов является жестким решением — при замене аппаратуры, например сетевого адаптера, изменяется и адрес сетевого интерфейса компьютера.

Типичными представителями иерархических числовых адресов являются сетевые IP- и IPX-адреса. В них поддерживается двухуровневая иерархия, адрес делится на старшую часть — номер сети — и младшую — номер узла. Такое разделение позволяет передавать сообщения между сетями только на основа-

нии номера сети, а номер узла используется после доставки сообщения в нужную сеть; точно так же, как название улицы используется почтальоном только после того, как письмо доставлено в нужный город. В последнее время, чтобы сделать маршрутизацию в крупных сетях более эффективной, предлагаются более сложные варианты числовой адресации, в соответствии с которыми адрес имеет три и более составляющих. Такой подход, в частности, реализован в новой версии протокола IPv6, предназначенного для работы в Internet.

Символьные адреса или имена предназначены для запоминания людьми и поэтому обычно несут смысловую нагрузку. Символьные адреса можно использовать как в небольших, так и в крупных сетях. Для работы в больших сетях символьное имя может иметь иерархическую структуру, например ftp-arch1.ucl.ac.uk. Этот адрес говорит о том, что данный компьютер поддерживает FTP-архив в сети одного из колледжей Лондонского университета (University College London — ucl), и данная сеть относится к академической ветви (ac) Internet Великобритании (United Kingdom — uk). При работе в пределах сети Лондонского университета такое длинное символьное имя явно избыточно и вместо него можно пользоваться кратким символьным именем, на роль которого хорошо подходит самая младшая составляющая полного имени, то есть ftp-arch1.

В современных сетях для адресации узлов, как правило, применяются все три приведенные выше схемы одновременно. Пользователи адресуют компьютеры символьными именами, которые автоматически заменяются в сообщениях, передаваемых по сети, на числовые номера. С помощью этих числовых номеров сообщения передаются из одной сети в другую, а после доставки сообщения в сеть назначения вместо числового номера используется аппаратный адрес компьютера.

IP- адрес имеет длину 4 байта и обычно записывается в виде четырех чисел, представляющих значения каждого байта в десятичной форме и разделенных точками. Существует 5 классов IP- адреса.

Класс	Первые биты	Наименьший номер сети	Наибольший номер сети	Максимальное количество узлов в сети
A	0	1.0.0.0	126.0.0.0	2^{24}
B	10	128.0.0.0	191.255.0.0	2^{16}
C	110	192.0.1.0	223.255.255.0	2^{8}
D	1110	224.0.0.0	239.255.255.255	Multicast
E	11110	240.0.0.0	247.255.255.255	Зарезервирован

Большие сети получают адреса класса A, средние – класса B, а маленькие класса C. Класс D – особый групповой адрес. Адреса класса E зарезервированы для будущих применений.

Иногда встречается запись IP-адресов вида 10.96.0.0/11. Данный вид записи заменяет собой указание диапазона IP-адресов. Число после косой черты означает количество единичных разрядов в маске подсети. Для приведённого примера маска подсети будет иметь двоичный вид 11111111 11100000 00000000 00000000 или то же самое в десятичном виде: 255.224.0.0. 11 разрядов IP-адреса

отводятся под номер сети, а остальные 32 − 11 = 21 разрядов полного адреса — под локальный адрес в этой сети. Итого, 10.96.0.0/11 означает диапазон адресов от 10.96.0.0 до 10.111.255.255

Значение маски	1	2	3	4
1	128	0	0	0
2	192	0	0	0
3	224	0	0	0
4	240	0	0	0
5	248	0	0	0
6	252	0	0	0
7	254	0	0	0
8	255	0	0	0
9	255	128	0	0
…				
32	255	255	255	255

Запись IP-адресов с указанием через слэш маски подсети переменной длины также называют CIDR-адресом в противоположность обычной записи без указания маски, в операционных системах типа UNIX также именуемой INET-адресом.

Задание

1. К какому типу можно отнести следующие адреса:
a) www.intuit.ru
b) 192.168.0.205
c) 2C-45-00-FE-5B-90
2. Какие из приведенных адресов могут использованы в качестве IP-адреса конечного узла сети, подключенной к Интернету?
a) 127.0.0.1
b) 201.13.123.245
c) 226.4.37.105
d) 204.0.1.1.
e) 12.13.14.15
f) 198.65.255.0
3. Определите номер подсети, минимальный, максимальный и широковещательный адреса, максимальное количество узлов в подсети по заданному адресу и маске:
IP-адрес 192.168.101.20 Маска 255.255.254.0, 255.255.255.128
IP-адрес 110.111.43.56 Маска 255.255.254.0, 255.255.255.192
IP-адрес 195.134.23.43 Маска 255.255.254.0, 255.255.255.224
IP-адрес 132.154.32.211 Маска 255.255.254.0, 255.255.255.192

Расчет характеристик сети ETHERNET PDV, PVV

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. 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.

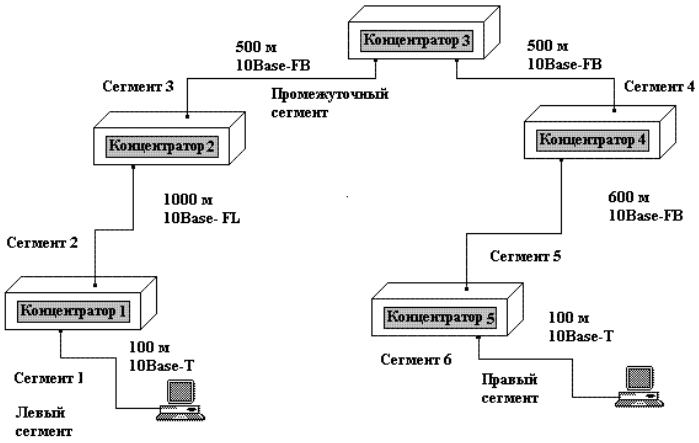
Левым сегментом называется сегмент, в котором начинается путь сигнала от выхода передатчика конечного узла. На рисунке это сегмент 1. Затем сигнал проходит через промежуточные сегменты 2 – 5 и приходит до наиболее удаленного узла сегмента 6, который называется правым. Именно здесь в худшем случае происходит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

жуточных и правого сегментов. Общее значение PDV не должно превышать 575.



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.

Алгоритмы маршрутизации

Алгоритмы маршрутизации применяются для определения наилучшего пути пакетов от источника к приёмнику и являются основой любого протокола маршрутизации. Для формулирования алгоритмов маршрутизации сеть рассматривается как граф. При этом маршрутизаторы являются узлами, а физические линии между маршрутизаторами — рёбрами соответствующего графа. Каждой грани графа присваивается определённое число — стоимость, зависящая от физической длины линии, скорости передачи данных по линии или финансовой стоимости линии.

Алгоритмы маршрутизации можно разделить на адаптивные и неадаптивные; глобальные и децентрализованные; статические и динамические.

Протокол TCP/IP

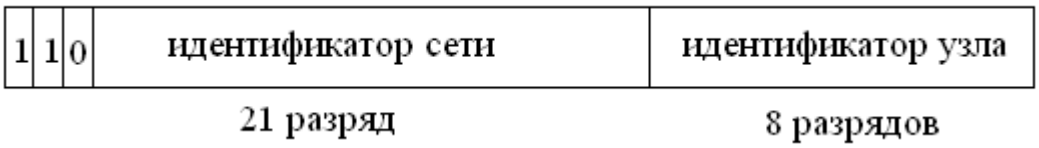
Протокол TCP/IP предназначен для соединения сетей с разнородным оборудованием. Это единственное средство коммуникации, позволяющее связываться рабочим станциям всех типов – PC, Macintosh, Unix. Он необходим для выхода в Internet. Фактически TCP/IP является набором протоколов. Наиболее известные из них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

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 C, формат которого показан на рисунке 1.



Формат IP-адреса класса C.

По соглашению IP-адреса пишутся в десятичном формате с разделением точками. Четыре части адреса соответствуют отдельным байтам. Например, IP-адрес класса C 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.

Датаграмма IP		
	Датаграмма UDP	
Заголовок IP (20 байтов)	Заголовок UDP (8 байтов)	Данные UDP

Датаграмма UDP внутри датаграммы IP.

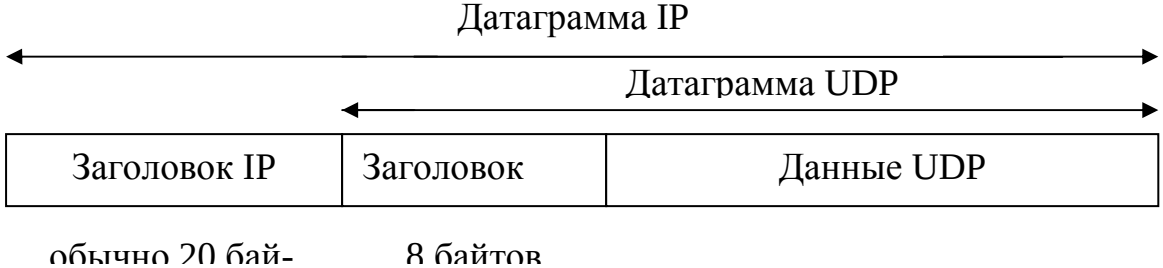
Подобно IP, UDP не сообщает отправителю о доставке датаграммы. Решение оставлено на усмотрение приложения. Например, отправитель может потребовать, чтобы получатель прислал ответ, и может заново отправить датаграмму, если ответ не пришел в течение, скажем, 20 с. UDP хорош для простых однократных сообщений — 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)	

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.



обычно 20 бай- 8 байтов

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="3">16-разрядный номер порта источника</td><td>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="3">16 разрядная контрольная сумма (заголовок TCP + данные)</td><td></td></tr>
<tr><td colspan="4">Параметры (если есть)</td></tr>
<tr><td colspan="4">Данные (если есть)</td></tr>
</table>

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 объединены в правильной последовательности. Программа-отправитель заново пе-

ресылает 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.

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.

VII. МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ К ВЫПОЛНЕНИЮ И КОНТРОЛЮ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

На самостоятельную работу студентов отводится 84 часа. 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. Алгоритмы маршрутизации (1 – С.349 – 360)
2. Особенности технологий Frame Relay, ATM, SDH (1 – С.252–253, 540–564, 530–539; 3 – С. 167–185)
3. Беспроводные сети (3 – С.105 – 113)
4. Сети с коммутацией пакетов (1 – С.520 – 530; 2 – С. 198 – 203)
5. Организация корпоративных сетей (1 – С.597- 600, 2 – С. 369–378)
6. Спутниковые каналы (1 – С.111)
7. Сотовые системы связи (1 – С.479 – 486)
8. Виды конференц-связи (4)

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.

VIII. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ОРГАНИЗАЦИИ МЕЖСЕССИОННОГО КОНТРОЛЯ ЗНАНИЙ СТУДЕНТОВ

Межсессионный контроль осуществляется на лабораторных работах.

IX. ПРИМЕР ЭКЗАМЕНАЦИОННОГО БИЛЕТА

АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Утверждено на заседании кафедры — Кафедра ИУС
«15 » декабря 2009 г. протокол №7 — Факультет МиИ
Заведующий кафедрой ИУС
Утверждаю А.В. Бушманов

Тестовые задания по дисциплине
«Сети ЭВМ и телекоммуникации» для магистрантов по направлению
230100 – Информатика и вычислительная техника
Тест содержит 25 вопросов.
Время выполнения заданий 60 минут

Фамилия Имя студента_____ номер группы_____

Вариант 1

7. Какие функции выполняются на транспортном уровне эталонной модели взаимодействия открытых систем (OSI)?

а) определение начала и окончания сеанса связи; определение времени, длительности и режима сеанса связи; определение точек синхронизации для промежуточного контроля и восстановления при передаче данных;

б) контроль последовательности передачи данных; установка соответствия между транспортными (логическими) и сетевыми адресами абонентов; обнаружение и обработка ошибок передачи данных;

в) определение маршрутизации в сети и связь между сетями; обеспечение независимости высших уровней от используемой для передачи информации физической среды; обнаружение и обработка ошибок передачи данных;

г) определение метода доступа к среде передачи данных; определение логической топологии сети передачи данных; определение физической адресации.

8. 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.

9. Какую из ситуаций называют коллизией?

а) когда станция, желающая передать пакет, обнаруживает, что в данный момент другая станция уже заняла передающую среду;

б) когда в данных, переданных станцией, обнаружена ошибка;

в) когда две рабочие станции одновременно передают данные в разделяемую передаю-

щую среду;

г) когда передача данных завершена станцией успешно.

10. Какое название носит сети метод доступа станций к передающей среде в сети Ethernet?

а) маркерное кольцо

б) множественный доступ с контролем несущей и обнаружением коллизий

в) маркерная шина

г) раннее освобождение маркера

11. При каком способе обмена информацией между узлами сети данные передаются в одном направлении?

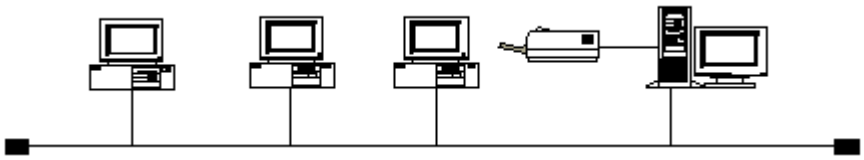
а) при симплексной передаче; **б)** при дуплексной передаче;

в) при полудуплексной передаче; **г)** при полусимплексной передаче

12. Как называется характеристика сети, предполагающая скрытие особенностей работы сети от конечного пользователя?

а) интегрируемость; **б)** прозрачность;

в) надежность; **с)** масштабируемость.



7. ***По какой топологии изображено подключение устройств на рисунке?***

а) звезда; **б)** шина; **в)** ячеистая топология; **г)** кольцо.

***8. Какой из перечисленных вариантов характеристик соответствует технологии* 10BASE-F?**

	а	б	в	г
Длина сегмента кабеля, м	500	185	100	2000
Тип кабеля	50 Омный коаксиальный, «толстый»	50 Омный коаксиальный, «тонкий»	Витая пара (UTP)	Оптоволокно
Топология подключения устройства	Шина	Шина	Звезда	Звезда

9. Какое из утверждений о логическим адресе узла сетиверное?

а) не определяется используемым протоколом обмена данными;

б) определяется стандартом локальной сети;

в) не может быть изменен после подключения устройств к сети;

г) может быть изменен в процессе работы.

10. Поставьте соответствия между типом коммуникационного устройства и уровнями моделиOSI, на котором оно может работать

1) повторитель **а)** физический, канальный, сетевой, транспортный

2) мост **б)** физический, канальный

3) шлюз
4) маршрутизатор

в) физический
г) над уровнями модели OSI

11. Какое устройство изолирует трафик одной подсети от трафика другой подсети, но не может быть использован в сети с замкнутым контуром?

а) мост **б)** шлюз **в)** маршрутизатор **г)** повторитель

12. Что используются в одномодовом кабеле в качестве источника излучения света?

а) светодиоды
б) полупроводниковые лазеры
с) светодиоды и полупроводниковые лазеры
г) разные виды лазеров

13. Какие из перечисленных пар сетевых технологий совместимы по форматам кадров и позволяют обрабатывать составную сеть без необходимости транслирования кадров?

а) Token Ring – Fast Ethernet
б) FDDI – Ethernet
в) Ethernet – Fast Ethernet
г) Token Ring - FDDI

14. Поставьте соответствия между названиями стандартов и технологий, которые они описывают:

а) 802.1
б) 802.3
в) 802.5

1) технология Token Ring
2) технология Ethernet
3) общие определения ЛВС, связь с моделью OSI

15. Где располагается горизонтальная подсистема иерархической структурированной кабельной системы?

а) в пределах этажа
б) в пределах здания
в) в пределах одной территории с несколькими зданиями
г) в пределах предприятия

16. Какими характеристиками оценивается степень искажения синусоидальных сигналов линиями связи?

а) затухание, мощность
б) пропускной способностью, полоса пропускания
в) помехоустойчивость, достоверность
г) амплитудно-частотная характеристика, полоса пропускания

17. Какие уровни модели OSI являются сетенезависимыми?

а) прикладной, представления данных
б) канальный, сетевой
в) транспортный, сеансовый
г) физический, сеансовый

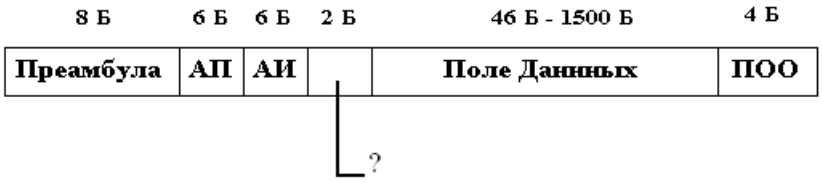
18. В каком году был принят первый стандарт сети Ethernet?

а) 1990 **б)** 1980 **в)**1970 **г)** 2000

19. Любая вычислительная система, построенная в соответствии с общедоступными спецификациями, соответствующими стандартам и принятые в результате публичного обсуждения, называется_____системой.

20. Стек коммуникационных протоколов _____ является лидером среди протоколов вычислительных сетей и использует гибкую систему адресацию, позволяющую более просто включать в интерсеть сети различных технологий.

21. На рисунке схематично изображен формат пакета стандарта IEEE 802.3. Каково название поля, отмеченного вопросительным знаком?



поле_____

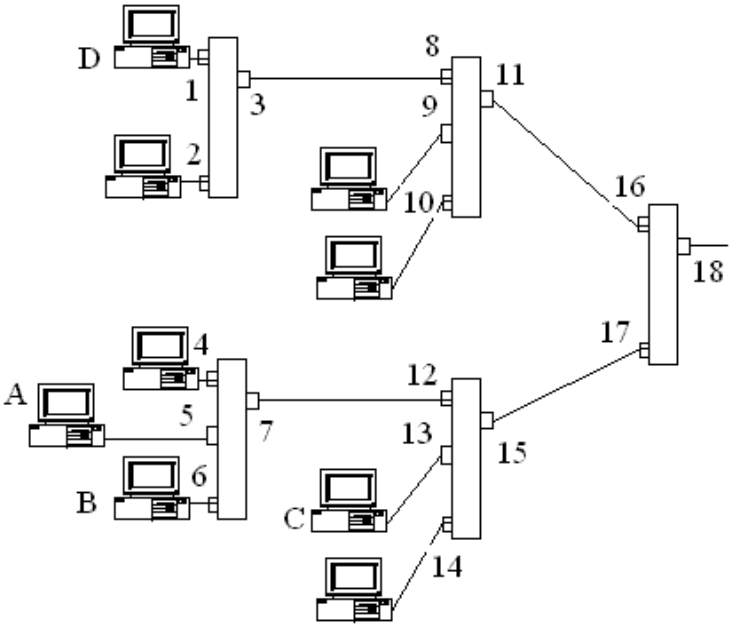
22. Какое из перечисленных названий стандартов является общим термином для обозначения стандартов, использующих в качестве среды передачи данных витую пару?

а)100BASE-T **б)**100BASE-TX **в)** 100BASE-T4 **г)**100BASE-T2

23. Каким будет теоретический предел скорости передачи данных в битах в секунду по каналу с шириной полосы пропускания в 20 кГц, если мощность передатчика составляет 0,01 мВт, а мощность шума в канале 0,0001 мВт

24. Если все коммуникационные устройства на схеме сети являются концентраторами, то на каких портах появится кадр, если его отправить с компьютера A компьютеру D?

а) на всех портах **б)** 5,6, 7 **в)** 4,5,6,7 **г)** 4,5,6,7,12, 13,14,15, 17,18



25. Если в предыдущем примере считать, что все устройства являются коммутаторами, то на каких портах появится кадр, если его отправить с компьютера A компьютеру C? _____

X. КАРТА ОБЕСПЕЧЕННОСТИ ДИСЦИПЛИНЫ КАДРАМИ ПРОФЕССОРСКО-ПРЕПОДАВАТЕЛЬСКОГО СОСТАВА

Все виды занятий по данной дисциплине ведет канд. техн. наук, доцент Галаган Т.А.

СОДЕРАНИЕ

Татьяна Алексеевна Галаган (составитель),
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