ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ
ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ЭЛЕКТРОНИКИ И ПРИБОРОСТРОЕНИЯ

Кафедра: «Информационные системы»

А.И. Фролов

СЕТИ ЭВМ И ТЕЛЕКОММУНИКАЦИИ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНОГО ПРАКТИКУМА

Дисциплина – «Сети ЭВМ и телекоммуникации»

Специальность 220400 «Программное обеспечение вычислительной техники и автоматизированных систем»

Печатается по решению редакционно-издательского совета ОрелГТУ

Орел 2006

ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ
ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ЭЛЕКТРОНИКИ И ПРИБОРОСТРОЕНИЯ
КАФЕДРА ИНФОРМАЦИОННЫХ СИСТЕМ

А.И. Фролов

СЕТИ ЭВМ И ТЕЛЕКОММУНИКАЦИИ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНОГО ПРАКТИКУМА

Дисциплина – «Сети ЭВМ и телекоммуникации»

Специальность 220400 «Программное обеспечение вычислительной техники и автоматизированных систем»

Печатается по решению редакционно-издательского совета ОрелГТУ

Орел 2006

Автор: к.т.н., доцент
кафедры «Информационные системы» А.И. Фролов
Рецензент: д.т.н., профессор, зав. кафедрой
«Информационные системы» И.С. Константинов

Методические указания содержат описание шести лабораторных работ по дисциплине «Сети ЭВМ и телекоммуникации». Посвящены выработке у студентов навыков программирования с использованием наиболее распространенных сетевых протоколов и изучению принципов построения компьютерных сетей. Методические указания содержат теоретические сведения по рассматриваемым вопросам, практические примеры и справочную информацию, необходимую для выполнения лабораторных работ.

Методические указания предназначены для студентов очной формы обучения специальности 220400 «Программное обеспечение вычислительной техники и автоматизированных систем».

Редактор Г.А. Константинова
Технический редактор Д.Е.Суханов
Орловский государственный технический университет
Лицензия ИД 00670 от 5.01.2000
АНО «ОрелГТУ-РЦФИО»

Подписано к печати 20 апреля 2006 г. Формат 60×84 1\16
Печать офсетная . Усл. печ. л. 4,5. Тираж _____ экз.
Заказ №_____
Отпечатано с готового оригинал-макета
на полиграфической базе ОрелГТУ

Содержание

1 Расчет конфигурации сети Ethernet

Цель работы: изучение принципов построения сетей по стандарту Ethernet и приобретение практических навыков оценки корректности их конфигурации.

Необходимое оборудование: калькулятор.

1.1 Критерии корректности конфигурации

Соблюдение многочисленных ограничений, установленных для различных стандартов физического уровня сетей Ethernet, гарантирует корректную работу сети (естественно, при исправном состоянии всех элементов физического уровня). Основные характеристики и ограничения технологии Ethernet приведены в таблицах 1.1 и 1.2.

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 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 повторителей 10Base-5 и 5 сегментов максимальный длины 500 м, то окажется, что оно составляет 537 битовых интервала. А так как время передачи кадра минимальной длины (вместе с преамбулой), составляющей 72 байт, равно 575 битовым интервалам, то видно, что разработчики стандарта Ethernet оставили 38 битовых интервала в качестве запаса для обеспечения надежности. Тем не менее в документах комитета IEEE 802.3 утверждается, что и 4 дополнительных битовых интервала создают достаточный запас надежности.

Комитет IEEE 802.3 приводит исходные данные о задержках (таблицы 1.3 и 1.4), 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 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 (Path Variability Value, PVV) при прохождении последовательности кадров через все повторители – 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).

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.

1.2 Методика расчета времени двойного оборота и уменьшения межкадрового интервала

Для упрощения расчетов обычно используются справочные данные IEEE, содержащие значения задержек распространения сигналов в повторителях, приемопередатчиках и различных физических средах (таблица 1.3). Битовый интервал обозначен как bt.

Комитет 802.3 старался максимально упростить выполнение расчетов, поэтому данные, приведенные в таблице, включают сразу несколько этапов прохождения сигнала. Например, задержки, вносимые повторителем, состоят из задержки входного трансивера, задержки блока повторения и задержки выходного трансивера. Тем не менее в таблице все эти задержки представлены одной величиной, названной базой сегмента.

Чтобы не нужно было два раза складывать задержки, вносимые кабелем, в таблице даются удвоенные величины задержек для каждого типа кабеля.

В таблице используются также такие понятия, как левый сегмент, правый сегмент и промежуточный сегмент. Поясним эти термины на примере сети, приведенной на рисунке 1.1. Левым сегментом называется сегмент, в котором начинается путь сигнала от выхода передатчика конечного узла. На рисунке 1.1 это сегмент 1. Затем сигнал проходит через промежуточные сегменты 2-5 и доходит до приемника наиболее удаленного узла наиболее удаленного сегмента 6, который называется правым. Именно здесь в худшем случае происходит столкновение кадров и возникает коллизия.

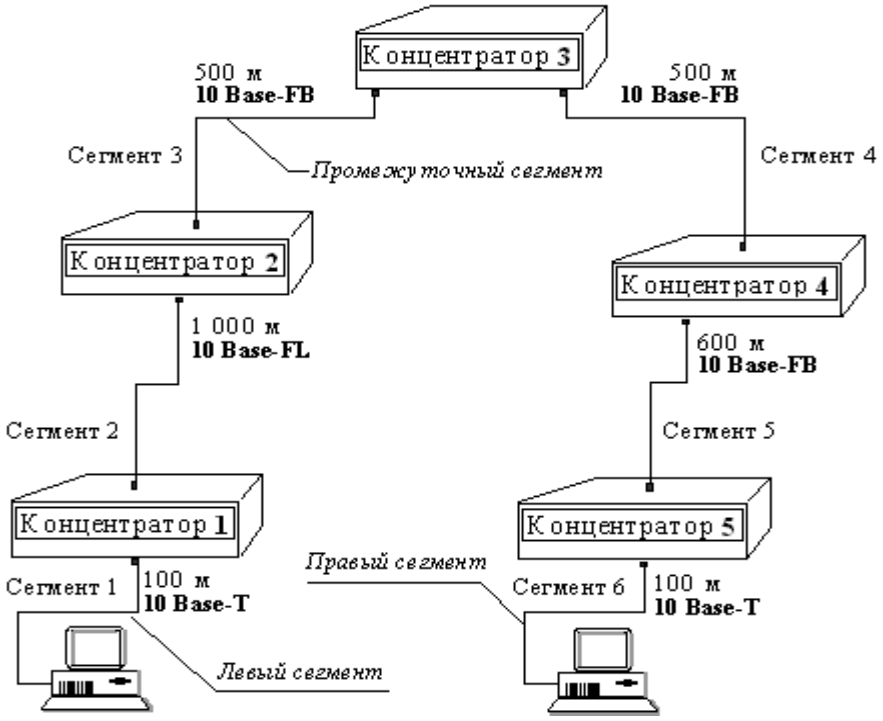


Рисунок 1.1 – Пример сети Ethernet, состоящей из сегментов различных физических стандартов

С каждым сегментом связана постоянная задержка, названная базой, которая зависит только 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.

Кроме этого, с каждым сегментом связана задержка распространения сигнала вдоль кабеля сегмента, которая зависит от длины сегмента и вычисляется путем умножения времени распространения сигнала по одному метру кабеля (в битовых интервалах) на длину кабеля в метрах.

Расчет PDV заключается в вычислении задержек, вносимых каждым отрезком кабеля (приведенная в таблице задержка сигнала на 1 м кабеля умножается на длину сегмента), а затем суммировании этих задержек с базами левого, промежуточных и правого сегментов. Общее значение PDV не должно превышать 575.

Так как левый и правый сегменты имеют разные величины базовой задержки, то в случае различных типов сегментов на удаленных краях сети необходимо выполнить расчеты дважды: один раз принять в качестве левого сегмента сегмент одного типа, а во второй – сегмент другого типа. Результатом можно считать максимальное значение PDV.

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

Для расчета PVV также можно воспользоваться значениями максимальных величин уменьшения межкадрового интервала при прохождении повторителей различных физических сред, рекомендованными IEEE и приведенными в таблице 1.4.

1.3 Пример расчета конфигурации сети

В примере крайние сегменты сети принадлежат к одному типу – стандарту 10Base-T, поэтому двойной расчет не требуется.

Приведенная на рисунке 1.1 сеть в соответствии с правилом «4 хабов» не является корректной – в сети между узлами сегментов 1 и 6 имеются 5 хабов,

хотя не все сегменты являются сегментами 10Base-FB. Кроме того, общая длина сети равна 2800 м, что нарушает правило 2500 м. Рассчитаем значение PDV.

Левый сегмент 1:

15,3 (база) + 100·0,113 = 26,6

Промежуточный сегмент 2:

33,5 + 1000·0,1 = 133,5

Промежуточный сегмент 3:

24 + 500·0,1 = 74,0

Промежуточный сегмент 4:

24 + 500·0,1 = 74,0.

Промежуточный сегмент 5:

24 + 600·0,1 = 84,0

Правый сегмент 6:

165 + 100·0,113 = 176,3.

Сумма всех составляющих дает значение PDV, равное 568,4.

Так как значение PDV меньше максимально допустимой величины 575, то эта сеть проходит по критерию времени двойного оборота сигнала несмотря на то, что ее общая длина превышает 2500 м, а количество повторителей больше 4.

Рассчитаем значение PVV.

Левый сегмент 1 10Base-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 сеть соответствует стандартам Ethernet по всем параметрам.

1.4 Задание на лабораторную работу

1. Ознакомиться с теоретическим материалом.

2. Произвести оценку конфигурации сети в соответствии с вариантом:

– по физическим ограничениям: на длину сегмента, на длину сети, правило «4 хаба» («5 хабов» для 10Base-FB);

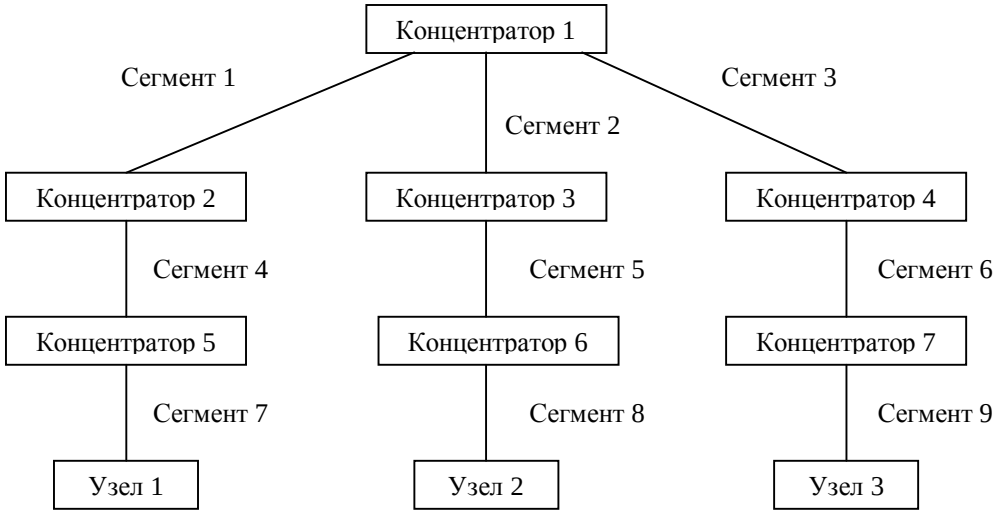
– по времени двойного оборота сигнала в сети;

– по уменьшению межкадрового интервала.

3. По результатам расчетов сделать вывод о корректности конфигурации сети Ethernet.

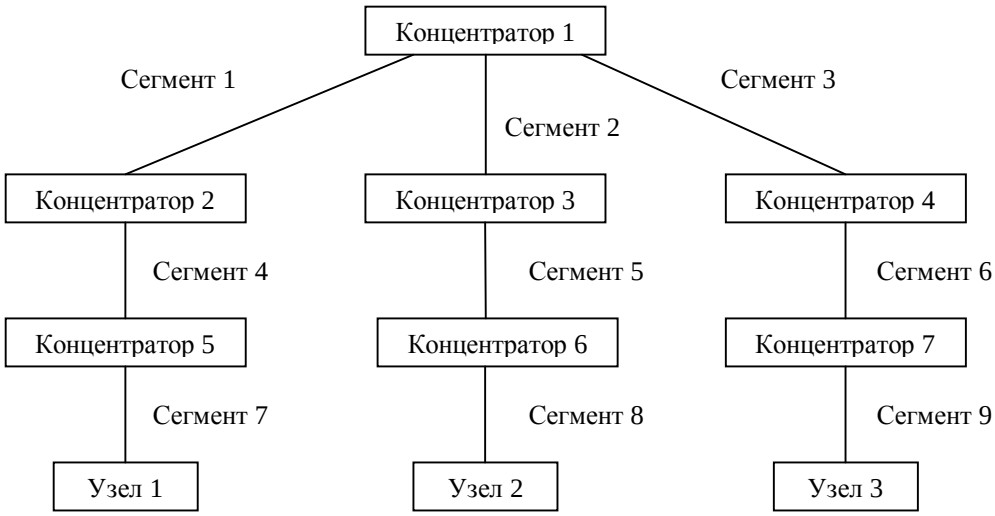
4. По результатам работы оформить отчет. Содержание отчета: исходные данные, расчеты указанных параметров, выводы.

Вариант 1



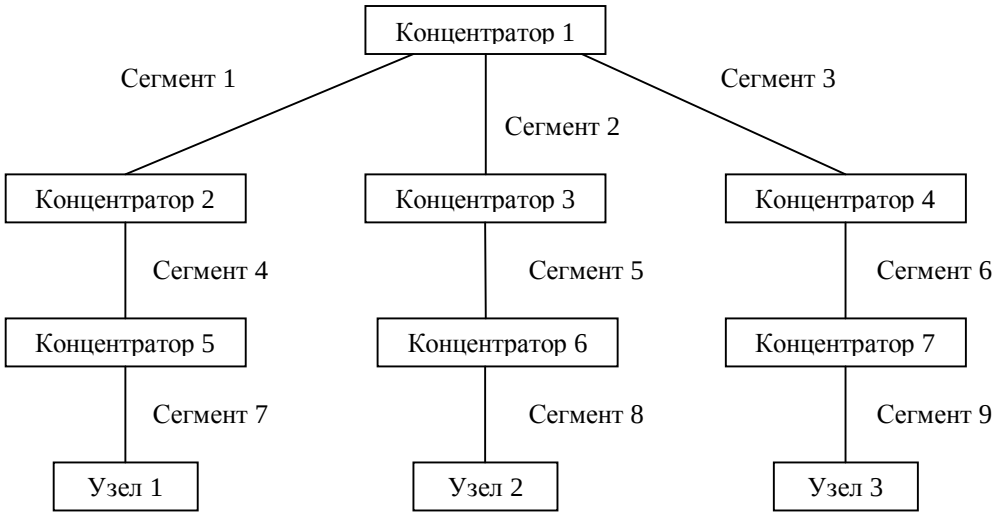
	10 Base-FB	10 Base-FL	10 Base-T	Длина, м
Сегмент 1	+			500
Сегмент 2	+			300
Сегмент 3	+			400
Сегмент 4		+		1000
Сегмент 5		+		300
Сегмент 6		+		400
Сегмент 7			+	100
Сегмент 8			+	50
Сегмент 9			+	100

Вариант 2



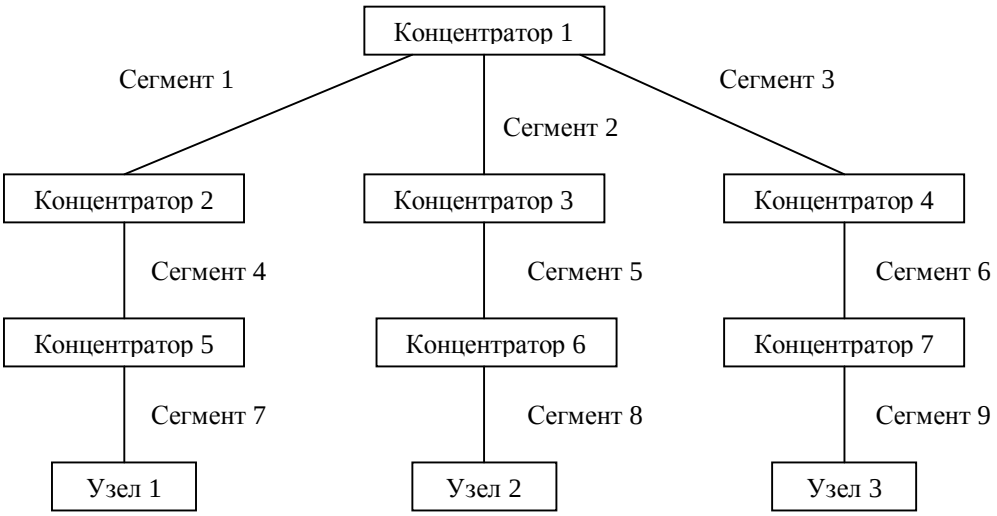
	10 Base-FB	10 Base-FL	10 Base-T	Длина, м
Сегмент 1		+		700
Сегмент 2	+			400
Сегмент 3	+			400
Сегмент 4		+		700
Сегмент 5		+		200
Сегмент 6	+			500
Сегмент 7			+	80
Сегмент 8			+	100
Сегмент 9			+	80

Вариант 3



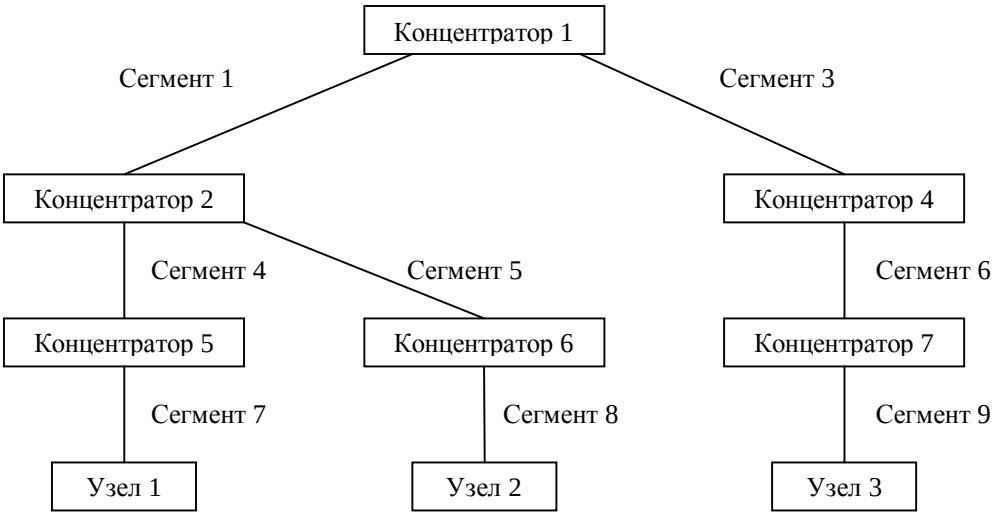
	10 Base-FB	10 Base-FL	10 Base-T	Длина, м
Сегмент 1	+			1000
Сегмент 2		+		200
Сегмент 3		+		200
Сегмент 4		+		400
Сегмент 5	+			300
Сегмент 6		+		200
Сегмент 7			+	100
Сегмент 8			+	100
Сегмент 9			+	40

Вариант 4



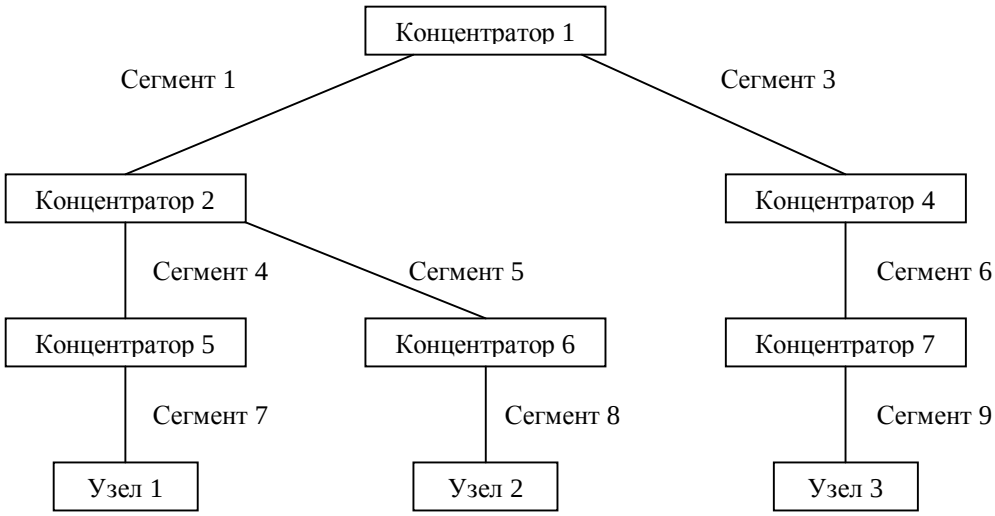
	10 Base-FB	10 Base-FL	10 Base-T	Длина, м
Сегмент 1		+		600
Сегмент 2		+		400
Сегмент 3		+		200
Сегмент 4	+			800
Сегмент 5	+			500
Сегмент 6	+			800
Сегмент 7			+	50
Сегмент 8			+	100
Сегмент 9			+	50

Вариант 5



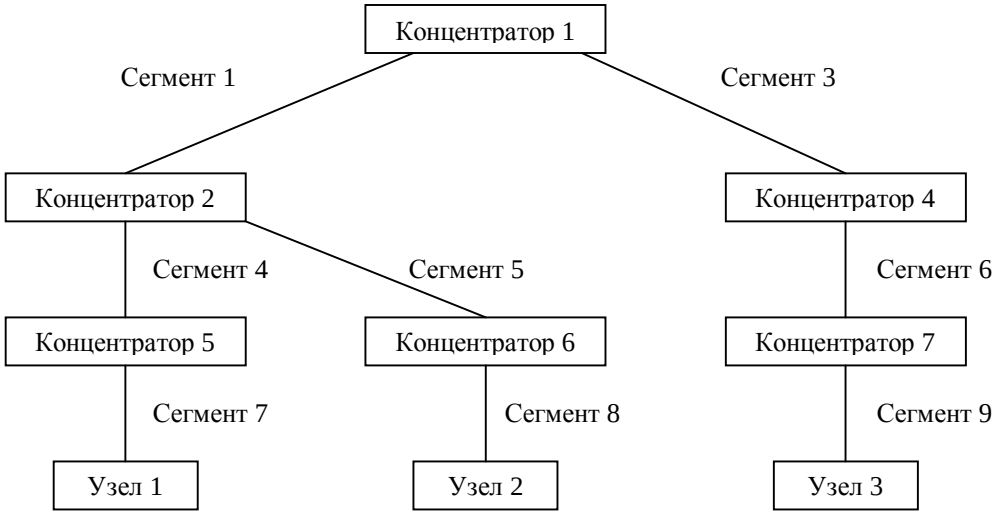
	10 Base-FB	10 Base-FL	10 Base-T	Длина, м
Сегмент 1	+			400
Сегмент 3	+			500
Сегмент 4		+		1100
Сегмент 5		+		1100
Сегмент 6		+		600
Сегмент 7			+	100
Сегмент 8			+	100
Сегмент 9			+	100

Вариант 6



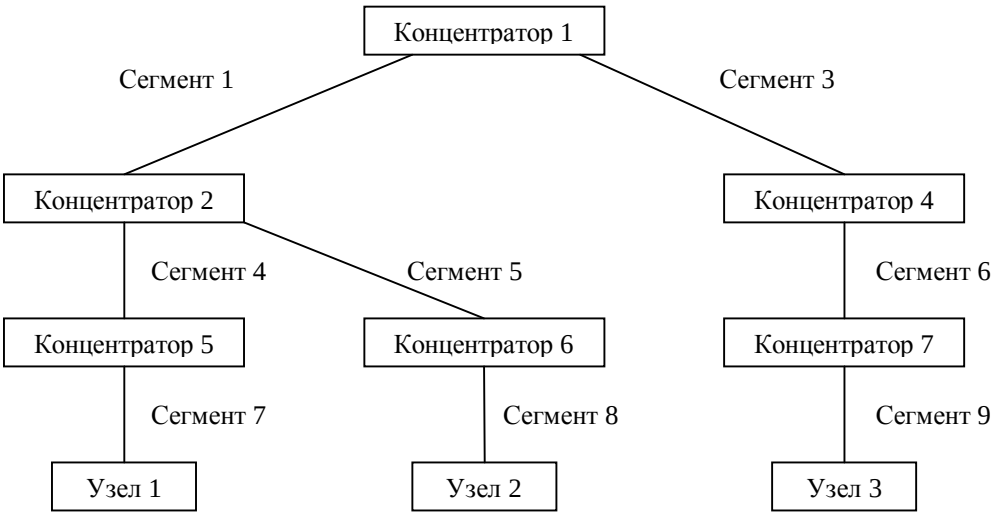
	10 Base-FB	10 Base-FL	10 Base-T	Длина, м
Сегмент 1	+			500
Сегмент 3		+		500
Сегмент 4	+			1000
Сегмент 5	+			1000
Сегмент 6		+		500
Сегмент 7			+	80
Сегмент 8			+	80
Сегмент 9			+	100

Вариант 7



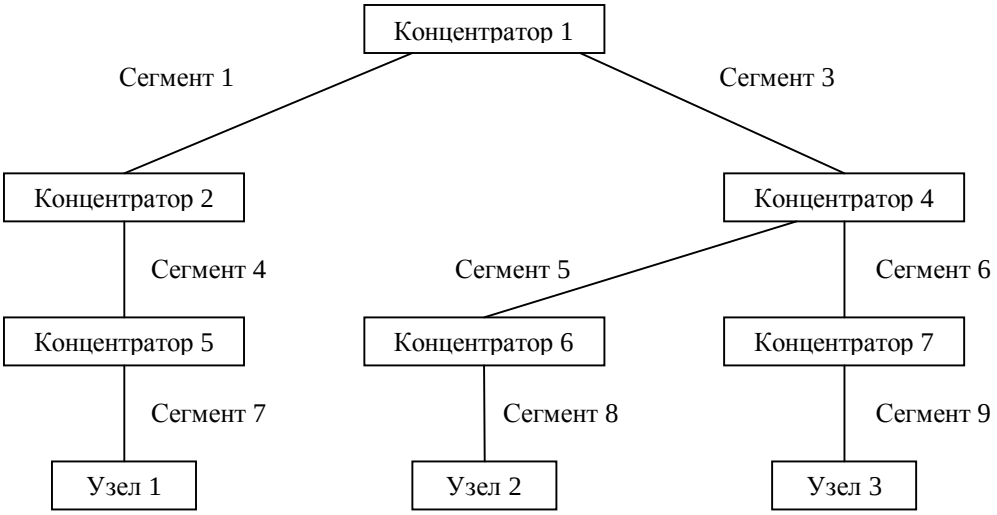
	10 Base-FB	10 Base-FL	10 Base-T	Длина, м
Сегмент 1		+		1000
Сегмент 3	+			1000
Сегмент 4		+		600
Сегмент 5		+		600
Сегмент 6	+			400
Сегмент 7			+	60
Сегмент 8			+	60
Сегмент 9			+	90

Вариант 8



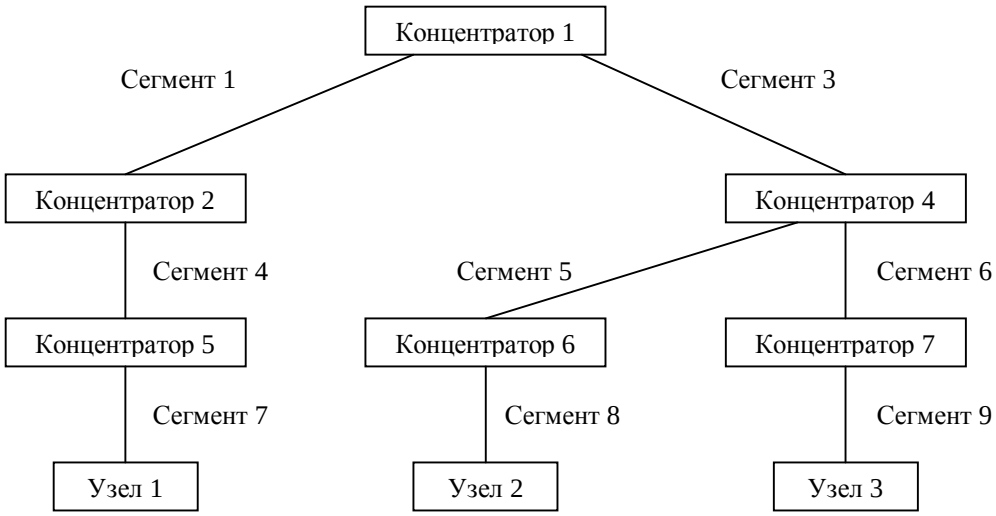
	10 Base-FB	10 Base-FL	10 Base-T	Длина, м
Сегмент 1		+		900
Сегмент 3		+		900
Сегмент 4	+			700
Сегмент 5	+			700
Сегмент 6	+			500
Сегмент 7			+	70
Сегмент 8			+	70
Сегмент 9			+	100

Вариант 9



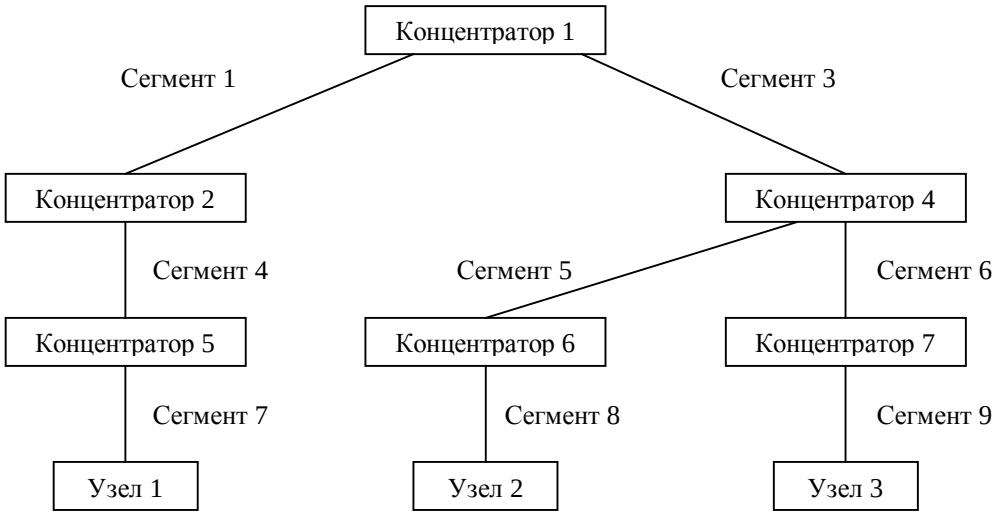
	10 Base-FB	10 Base-FL	10 Base-T	Длина, м
Сегмент 1	+			400
Сегмент 3	+			500
Сегмент 4		+		1100
Сегмент 5		+		1100
Сегмент 6		+		600
Сегмент 7			+	100
Сегмент 8			+	100
Сегмент 9			+	100

Вариант 10



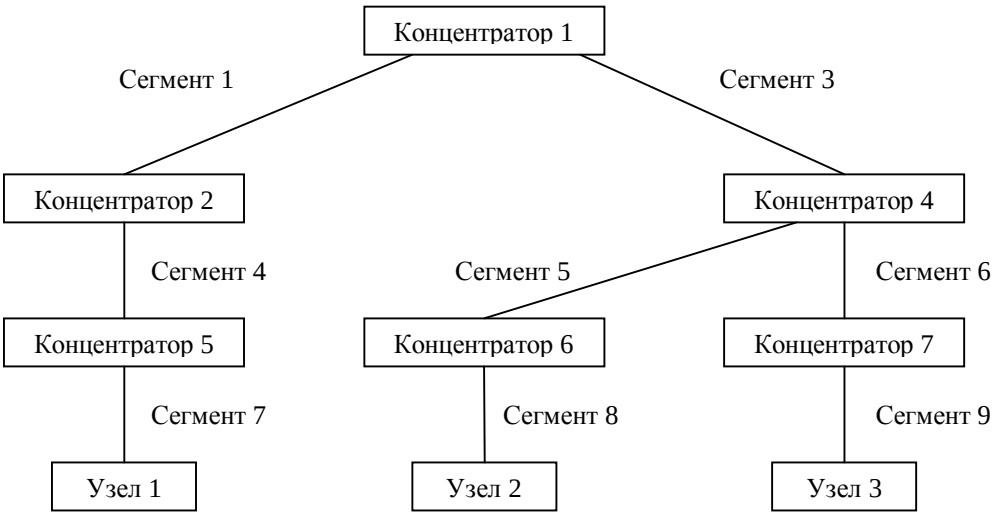
	10 Base-FB	10 Base-FL	10 Base-T	Длина, м
Сегмент 1	+			500
Сегмент 3		+		500
Сегмент 4	+			1000
Сегмент 5	+			1000
Сегмент 6		+		500
Сегмент 7			+	80
Сегмент 8			+	80
Сегмент 9			+	100

Вариант 11



	10 Base-FB	10 Base-FL	10 Base-T	Длина, м
Сегмент 1		+		1000
Сегмент 3	+			1000
Сегмент 4		+		600
Сегмент 5		+		600
Сегмент 6	+			400
Сегмент 7			+	60
Сегмент 8			+	60
Сегмент 9			+	90

Вариант 12



	10 Base-FB	10 Base-FL	10 Base-T	Длина, м
Сегмент 1		+		600
Сегмент 3		+		600
Сегмент 4	+			900
Сегмент 5	+			1000
Сегмент 6	+			500
Сегмент 7			+	70
Сегмент 8			+	80
Сегмент 9			+	90

1.5 Справочные данные IEEE

Таблица 1.1 – Общие ограничения для всех стандартов Ethernet

Характеристика	Значение
Номинальная пропускная способность	10 Мбит/с
Максимальное число станций в сети	1024
Максимальное расстояние между узлами в сети	2500 м (в 10Base-FB – 2750 м)
Максимальное число коаксиальных сегментов в сети	5

Таблица 1.2 – Параметры спецификаций физического уровня для стандарта Ethernet

Параметр	10Base-5	10Base-2	10Base-T	10Base- F
Кабель	Толстый коаксиальный кабель RG-8 или RG-11	Тонкий коаксиальный кабель RG-58	Неэкранированная витая пара категорий 3,4,5	Многомодовый волоконно-оптический кабель
Максимальная длина сегмента, м	500	185	100	2000
Максимальное расстояние между узлами сети (при использовании повторителей), м	2500	925	500	2500 (2740 для 10Base-FB)

Продолжение таблицы 1.2

Параметр	10Base-5	10Base-2	10Base-T	10Base- F
Максимальное число станций в сегменте	100	30	1024	1024
Максимальное число повторителей между любыми станциями сети	4	4	4	4 (5 для 10Base-FB)

Таблица 1.3 – Данные для расчета значения PDV

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
10Base-FL	12,3	33,5	156,5	0,1	2000
FOIRL	7,8	29,0	152,0	0,1	1000
AUI (>2 м)	0	0	0	0,1026	2+48

Таблица 1.4 – Уменьшение межкадрового интервала повторителями

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

1.6 Контрольные вопросы

1. Поясните механизм доступа к разделяемой среде в технологии Ethernet.

2. В каких случаях возможна оценка корректности конфигурации по физическим ограничениям?

3. Сформулируйте условие надежного распознавания коллизий.

4. С какой целью вводится ограничение на уменьшение межкадрового интервала?

5. В каком случае и почему для самого длинного пути проводятся два расчета?

2 Изучение структуры IP-адреса

Цель работы: изучение принципов адресации в сетях TCP/IP и приобретение практических навыков применения и назначения IP-адресов с использованием масок.

Необходимое оборудование: калькулятор.

2.1 Типы адресов стека TCP/IP

В стеке TCP/IP используются три типа адресов: локальные (называемые также аппаратными), IP-адреса и символьные доменные имена.

В терминологии TCP/IP под локальным адресом понимается такой тип адреса, который используется средствами базовой технологии для доставки данных в пределах подсети, являющейся элементом составной интерсети. В разных подсетях допустимы разные сетевые технологии, разные стеки протоколов, поэтому при создании стека TCP/IP предполагалось наличие разных типов локальных адресов. Если подсетью интерсети является локальная сеть, то локальный адрес – это MAC-адрес. Однако протокол IP может работать и над протоколами более высокого уровня, например, над протоколом IPX или X.25. В этом случае локальными адресами для протокола IP соответственно будут адреса IPX и X.25. Компьютер в локальной сети может иметь несколько локальных адресов даже при одном сетевом адаптере. Некоторые сетевые устройства не имеют локальных адресов. Например, к таким устройствам относятся глобальные порты маршрутизаторов, предназначенные для соединений типа «точка-точка».

IP-адреса представляют собой основной тип адресов, на основании которых сетевой уровень передает пакеты между сетями. Эти адреса состоят из 4 байт, например, 109.26.17.100. IP-адрес назначается администратором во время конфигурирования компьютеров и маршрутизаторов. IP-адрес состоит из двух частей: номера сети и номера узла. Номер сети может быть выбран администратором произвольно, либо назначен по рекомендации специального подразделения Internet (Internet Network Information Center, InterNIC), если сеть должна

работать как составная часть Internet. Номер узла в протоколе IP назначается независимо от локального адреса узла. Маршрутизатор по определению входит сразу в несколько сетей. Поэтому каждый порт маршрутизатора имеет собственный IP-адрес. Конечный узел также может входить в несколько IP-сетей. В этом случае компьютер должен иметь несколько IP-адресов, по числу сетевых связей. Таким образом, IP-адрес характеризует не отдельный компьютер или маршрутизатор, а одно сетевое соединение.

Символьные доменные имена. Символьные имена в IP-сетях называются доменными и строятся по иерархическому признаку. Между доменным именем и IP-адресом узла нет никакого алгоритмического соответствия, поэтому необходимо использовать какие-то дополнительные таблицы или службы, чтобы узел сети однозначно определялся как по доменному имени, так и по IP-адресу. В сетях TCP/IP используется специальная распределенная служба Domain Name System (DNS), которая устанавливает это соответствие на основании создаваемых администраторами сети таблиц соответствия. Поэтому доменные имена называют также DNS-именами.

2.2 Классы IP-адресов

IP-адрес имеет длину 4 байта и обычно записывается в виде четырех чисел, представляющих значения каждого байта в десятичной форме и разделенных точками, например:

128.10.2.30 – традиционная десятичная форма представления адреса;

10000000 00001010 00000010 00011110 – двоичная форма представления этого же адреса.

Адрес состоит из двух логических частей – номера сети и номера узла в сети. Какая часть адреса относится к номеру сети, а какая – к номеру узла, определяется значениями первых бит адреса. Значения этих бит являются также признаками того, к какому классу относится тот или иной IP-адрес. На рисунке 2.1 показана структура IP-адресов различных классов.

Если адрес начинается с 0, то сеть относят к классу A и номер сети занимает один байт, остальные 3 байта интерпретируются как номер узла в сети. Сети

класса A имеют номера в диапазоне от 1 до 126. (Номер 0 не используется, а номер 127 зарезервирован для специальных целей, о чем будет сказано ниже.) Количество узлов в сетях класса A может достигать 2^{24}, то есть 16 777 216 узлов.

```
Класс A. 0ccccccc yyyyyyyy yyyyyyyy yyyyyyyy
Класс B. 10cccccc cccccccc yyyyyyyy yyyyyyyy
Класс C. 110ccccc cccccccc cccccccc yyyyyyyy
Класс D. 1110aaaa aaaaaaaa aaaaaaaa aaaaaaaa
Класс E. 11110zzz zzzzzzzz zzzzzzzz zzzzzzzz
```

Рисунок 2.1 – Классы IP-адресов (c – бит, входящий в номер сети; y – бит, входящий в номер узла; a – бит, входящий в адрес группы multicast; z – бит, входящий в зарезервированный адрес)

Если первые два бита адреса равны 10, то сеть относится к классу B. В сетях класса B под номер сети и под номер узла отводится по 16 бит. Таким образом, сеть класса B является сетью средних размеров с максимальным числом узлов 2^{16}, что составляет 65 536 узлов.

Если адрес начинается с последовательности 110, то это сеть класса C. В этом случае под номер сети отводится 24 бита, а под номер узла – 8 бит. Сети этого класса наиболее распространены, число узлов в них ограничено 2^8, то есть 256 узлами.

Если адрес начинается с последовательности 1110, то он является адресом класса D и обозначает особый, групповой адрес – multicast. Если в пакете в качестве адреса назначения указан адрес класса D, то такой пакет должны получить все узлы, которым присвоен данный адрес.

Если адрес начинается с последовательности 11110, то это значит, что данный адрес относится к классу E. Адреса этого класса зарезервированы для будущих применений.

2.3 Особые IP-адреса

В протоколе IP существует несколько соглашений об особой интерпретации IP-адресов:

– Если весь IP-адрес состоит только из двоичных нулей, то он обозначает адрес того узла, который сгенерировал этот пакет; этот режим используется только в некоторых сообщениях ICMP.

– Если в поле номера сети стоят только нули, то по умолчанию узел назначения принадлежит той же самой сети, что и узел, который отправил пакет.

– Если все двоичные разряды IP-адреса равны 1, то пакет с таким адресом назначения должен рассылаться всем узлам, находящимся в той же сети, что и источник этого пакета. Такая рассылка называется ограниченным широковещательным сообщением (limited broadcast).

– Если в поле номера узла назначения стоят только единицы, то пакет с таким адресом рассылается всем узлам сети с заданным номером сети. Например, пакет с адресом 192.190.21.255 доставляется всем узлам сети 192.190.21.0. Такая рассылка называется широковещательным сообщением (broadcast).

При адресации необходимо учитывать те ограничения, которые вносятся особым назначением некоторых IP-адресов. Так, ни номер сети, ни номер узла не может состоять только из одних двоичных единиц или только из одних двоичных нулей. Отсюда следует, что максимальное количество узлов, приведенное для сетей каждого класса, на практике должно быть уменьшено на 2. Например, в сетях класса C под номер узла отводится 8 бит, которые позволяют задавать 256 номеров: от 0 до 255. Однако на практике максимальное число узлов в сети класса C не может превышать 254, так как адреса 0 и 255 имеют специальное назначение. Из этих же соображений следует, что конечный узел не может иметь адрес типа 98.255.255.255, поскольку номер узла в этом адресе класса A состоит из одних двоичных единиц.

Особый смысл имеет IP-адрес, первый октет которого равен 127. Он используется для тестирования программ и взаимодействия процессов в пределах одной машины. Когда программа посылает данные по IP-адресу 127.0.0.1, то

образуется как бы «петля». Данные не передаются по сети, а возвращаются модулям верхнего уровня как только что принятые. Поэтому в IP-сети запрещается присваивать машинам IP-адреса, начинающиеся со 127. Этот адрес имеет название loopback. В протоколе IP нет понятия широковещательности в том смысле, в котором оно используется в протоколах канального уровня локальных сетей, когда данные должны быть доставлены абсолютно всем узлам. Как ограниченный широковещательный IP-адрес, так и широковещательный IP-адрес имеют пределы распространения в интерсети – они ограничены либо сетью, к которой принадлежит узел-источник пакета, либо сетью, номер которой указан в адресе назначения.

Уже упоминавшаяся форма группового IP-адреса – multicast – означает, что данный пакет должен быть доставлен сразу нескольким узлам, которые образуют группу с номером, указанным в поле адреса. Узлы сами идентифицируют себя, то есть определяют, к какой из групп они относятся. Один и тот же узел может входить в несколько групп. Члены какой-либо группы multicast не обязательно должны принадлежать одной сети. Групповой адрес не делится на поля номера сети и узла и обрабатывается маршрутизатором особым образом.

Групповая адресация предназначена для экономичного распространения в Internet или большой корпоративной сети аудио- или видеопрограмм, предназначенных сразу большой аудитории слушателей или зрителей. Если такие средства найдут широкое применение, то Internet сможет создать серьезную конкуренцию радио и телевидению.

2.4 Использование масок в IP-адресации

Важным элементом разбиения адресного пространства Internet являются подсети. Подсеть – это подмножество сети, не пересекающееся с другими подсетями. Это означает, что сеть организации может быть разбита на фрагменты, каждый из которых будет составлять подсеть. Реально, каждая подсеть соответствует физической локальной сети (например, сегменту Ethernet). Подсети используются для того, чтобы обойти ограничения физических сетей на число узлов в них и максимальную длину кабеля в сегменте сети. Например, сегмент

тонкого Ethernet имеет максимальную длину 185 м и может включать до 32 узлов. Самая маленькая сеть класса С может состоять из 254 узлов. Для того чтобы достичь этого значения, необходимо объединить несколько физических сегментов сети. Сделать это можно либо с помощью физических устройств (например, повторителей), либо при помощи машин-шлюзов. В первом случае разбиение на подсети не требуется, так как логически сеть выглядит как одно целое. При использовании шлюза сеть разбивается на подсети.

Разбиение сети на подсети использует ту часть IP-адреса, которая закреплена за номерами компьютеров. Администратор сети может замаскировать часть IP-адреса и использовать ее для назначения номеров подсетей. Фактически способ разбиения адреса на две части теперь будет применяться к адресу компьютера из IP-адреса сети, в которой организуется разбиение на подсети.

Маска подсети – это четыре байта, которые накладываются на IP-адрес для получения номера подсети. Например, маска 255.255.255.0 позволяет разбить сеть класса В на 254 подсети по 254 узла в каждой. Подсети не только решают, но и создают ряд проблем. Например, происходит потеря адресов, но уже не по причине физических ограничений, а по причине принципа построения адресов подсети. Так, выделение трех битов на адрес подсети приводит к образованию не восьми, а только шести подсетей, так как номера 0 и 7 нельзя использовать в силу специального значения IP-адресов, состоящих из нулей или из единиц.

Для стандартных классов сетей маски имеют следующие значения:

класс А – 11111111. 00000000. 00000000. 00000000 (255.0.0.0);

класс В – 11111111. 11111111. 00000000. 00000000 (255.255.0.0);

класс С – 11111111. 11111111. 11111111. 00000000 (255.255.255.0).

Снабжая каждый IP-адрес маской, можно отказаться от понятий классов адресов и сделать систему адресации более гибкой. Например, адрес 185.23.44.206 попадает в диапазон 128-191, то есть адрес относится к классу В. Следовательно, номером сети являются первые два байта, дополненные двумя нулевыми байтами – 185.23.0.0, а номером узла – 0.0.44.206. Если этот адрес ас-

социировать с маской 255.255.255.0, то номером подсети будет 185.23.44.0, а не 185.23.0.0, как это определено системой классов.

В масках количество единиц в последовательности, определяющей границу номера сети, не обязательно должно быть кратным 8, чтобы повторять деление адреса на байты. Пусть, например, для IP-адреса 129.64.134.5 указана маска 255.255.128.0, то есть в двоичном виде:

IP-адрес 129.64.134.5 – 10000001.01000000.10000110.00000101

Маска 255.255.128.0 – 11111111.11111111.10000000.00000000

Если использовать для определения границы номера сети маску, то 17 последовательных единиц в маске, «наложенные» на IP-адрес, определяют в качестве номера сети в двоичном выражении число:

10000001. 01000000. 10000000. 00000000 или в десятичной форме записи – номер сети 129.64.128.0, а номер узла 0.0.6.5.

2.5 Задание на лабораторную работу

1. Ознакомиться с теоретическим материалом.

2. Разработать приложение, которое по заданному классу (A, B или C), количеству подсетей N и максимальному количеству компьютеров M в подсети определяет маску для разбиения на подсети и список возможных IP-адресов подсетей. Если разбиение на подсети невозможно, приложение должно выдавать соответствующее сообщение об ошибке.

2.6 Контрольные вопросы

1. Назовите типы адресов, используемые в стеке TCP/IP. Охарактеризуйте их назначение и применяемые схемы адресации.

2. Назовите и охарактеризуйте классы IP-адресов.

3. Для каких целей используются договоренности об особых адресах?

4. Для каких целей при назначении адресов используются маски?

5. Опишите вид маски и принцип ее использования.

6. Опишите методику выбора маски по заданным в лабораторной работе параметрам.

3 Взаимодействие прикладных программ с помощью транспортного протокола TCP

Цель работы: изучение принципов организации обмена сообщениями между сетевыми приложениями по протоколу TCP и приобретение практических навыков создания клиент-серверных приложений на основе компонентов TClientSocket и TServerSocket.

Необходимое оборудование: IBM PC-совместимый компьютер, подключенный к локальной сети, с установленным программным обеспечением Borland Delphi версии 5.0 и выше.

3.1 Транспортный протокол TCP

TCP (Transmission Control Protocol) – это один из самых широко распространенных протоколов транспортного уровня. Главная функция TCP заключается в доставке сообщений без потерь, чего не может гарантировать протокол более низкого уровня IP (Internet Protocol). Для доставки сообщений предварительно устанавливается соединение между процессом-отправителем и процессом-получателем. Данное соединение осуществляет надежную доставку пакетов. Протокол TCP производит повторную передачу искаженного или утерянного пакета.

Выделение всех необходимых для надежной доставки сообщений функций в отдельный уровень освобождает разработчиков прикладных программ и утилит от решения задач управления потоком дейтаграмм. Протокол обеспечивает сквозную передачу данных от отправителя к получателю. Поскольку TCP ориентирован на установление соединения, то адресат, получивший дейтаграмму, должен уведомить отправителя об этом. Подразумевается, что между отправителем и получателем устанавливается виртуальный канал, где они обмениваются сообщениями, часть из которых является подтверждениями о получении данных либо кодами ошибок. Виртуальный канал на самом деле может подразумевать несколько реальных физических каналов передачи данных, поскольку сообщение может проходить через один или несколько шлюзов.

Когда некоторое приложение (процесс) прикладного уровня отправляет сообщение другому приложению с помощью TCP, предполагается, что сообщение является потоком, т.е. представляет собой поток байтов, передаваемых асинхронно. TCP получает поток байтов и собирает его в сегменты, добавляя заголовки в начало сегментов. Длина сегмента обычно определяется протоколом или выбирается администратором системы.

Процесс обмена данными начинается с передачи запроса на установление соединения от машины-отправителя к машине-получателю. В запросе содержится специальное целое число, называемое номером сокета (socket). В ответ получатель посылает номер своего сокета. Номера сокетов отправителя и получателя однозначно определяют соединение (конечно, соединение также не возможно без указания IP-адресов отправителя и получателя, но эта задача решается протоколами более низкого уровня – IP).

После установления соединения TCP начинает передавать сегменты сообщения. На более низком IP-уровне отправителя сегменты разбиваются на одну или несколько дейтаграмм. Пройдя через сеть, дейтаграммы поступают к получателю, где IP-уровень снова собирает из них сегменты и передает их TCP. TCP собирает все сегменты в сообщение. От TCP сообщение поступает к процессу-получателю, где обрабатывается протоколом прикладного уровня.

TCP на машине-получателе собирает целое сообщение из сегментов, руководствуясь порядковыми номерами сегментов, которые записаны в их заголовке. Если какой-то сегмент сообщения потерян или поврежден (что проверяется по контрольной сумме в заголовке сегмента), то отправителю посылается сообщение, содержащее номер ошибочного сегмента. В этом случае отправитель повторно передает сегмент. Если сегмент успешно принят, то получатель посылает отправителю подтверждение-квитанцию.

В TCP применяется средство ограничения потока данных, называемое скользящим окном. Оно представляет собой фрагмент сообщения, который адресат готов принять. При установлении соединения отправителю сообщается размер окна (размер окна кратен размеру сегмента). После того, как отправи-

тель передал количество байтов, соответствующее размеру окна, он должен ждать квитанции. Как только будет получена квитанция на переданные сегменты, окно сдвигается вправо на соответствующее число байтов, и новые сегменты могут быть переданы. Отправитель может передать без получения квитанций в сеть максимально столько сегментов, сколько их укладывается в скользящем окне. В процессе обмена данными получатель может присылать квитанции, в которых будет указан новый размер скользящего окна.

Важную роль в протоколе TCP играют таймеры. Сегмент считается потерянным, если квитанция на него не поступила в течение заданного времени ожидания. При этом производится повторная передача сегмента. При получении квитанции таймер останавливается. Если получатель обнаруживает несколько правильных копий одного и того же сегмента, то все лишние копии просто отбрасываются и отправителю передается только одна квитанция.

3.2 Транспортный протокол UDP

Протокол UDP (User Datagram Protocol) является более простым транспортным протоколом, чем протокол TCP. Он предоставляет прикладным процессам услуги транспортного уровня, которые мало чем отличаются от услуг более низкого уровня, предоставляемых протоколом IP. Протокол UDP обеспечивает доставку дейтаграмм, но не требует подтверждения их получения. Поэтому он не требует установления соединения между передающим и принимающим процессами.

Протокол UDP используется в тех случаях, когда требуется передать данные без установления соединения. Такая связь в принципе не надежна, так как отправителю не сообщается, правильно ли принято его сообщение и получено ли оно вообще. Для проверки возникновения ошибок может использоваться контрольная сумма пакета, но ошибки никак не обрабатываются – они либо игнорируются, либо их обработка выполняется уже на более высоком, прикладном уровне.

Данные, отправляемые прикладным процессом через UDP, достигают места назначения как единое целое, не дробясь на части. Например, если процесс-

отправитель передал пять сообщений через порт, то и процесс-получатель должен считать из порта пять сообщений. Размер каждого записанного сообщения должен совпадать с размером каждого прочитанного.

Протокол UDP используется тогда, когда требуется простейший механизм передачи данных. Тогда контроль ошибок либо не выполняется (например, в прикладном протоколе TFTP – Trivial File Transfer Protocol – простейший протокол передачи файлов), либо выполняется на прикладном уровне (например, в управляющем протоколе SNMP – Simple Network Management Protocol или в файловой системе NFS – Network File System).

3.3 Порты, мультиплексирование и демультиплексирование

Пакеты, поступающие на транспортный уровень, организуются операционной системой в виде множества очередей к точкам входа различных прикладных процессов. В терминологии TCP/IP такие системные очереди называются портами. Порт однозначно определяет приложение в пределах компьютера. Порты приложений не стоит путать с портами (интерфейсами) оборудования.

Существуют два способа присвоения порта приложению – централизованный и локальный. За каждым из этих способов закреплен свой диапазон номеров портов: для централизованного - от 0 до 1023, для локального – от 1023 до 65 535.

Если процессы представляют собой популярные общедоступные службы, такие как FTP, telnet, HTTP, TFTP, DNS и т. п., то за ними закрепляются стандартные присвоенные (assigned) номера, иногда также называемые хорошо известными (well-known) номерами. Централизованное присвоение службам номеров портов выполняется организацией Internet Assigned Numbers Authority (IANA). Эти номера закрепляются и публикуются в стандартах Internet (RFC 1700). Примеры хорошо известных портов приведены в таблице 3.1.

Для тех служб, которые еще не стали столь распространенными, чтобы закреплять за ними стандартные номера, номера портов выделяются локально. На каждом компьютере операционная система ведет список занятых и свободных

номеров портов. При поступлении запроса от приложения, выполняемого на данном компьютере, операционная система выделяет ему первый свободный номер. Такие номера называют динамическими (dynamic). В дальнейшем все сетевые приложения должны адресоваться к данному приложению с указанием назначенного ему номера порта. После того как приложение завершит работу, выделенный ему локальный номер порта возвращается в список свободных и может быть назначен другому приложению.

Таблица 3.1 – Примеры номеров портов наиболее популярных служб сети Internet

Номер порта	Служба сети	Описание
0		Зарезервирован
7	echo	Эхо-ответ на входящие сообщения
9	discard	Сброс (поглощение) всех входящих сообщений
11	users	Активные пользователи
13	daytime	Отклик, содержащий время дня
19	chargen	Генератор символов
20	ftp data	Передача данных по протоколу FTP
21	ftp	Передача управляющих команд по протоколу FTP
23	telnet	Порт подключения по протоколу TELNET
25	smtp	Протокол передачи почтовых сообщений SMTP
37	time	Отклик, содержащий время
42	name	Сервер имен
53	domain	Сервер имен доменов
67	boots	Протокол удаленной загрузки сервера
68	bootc	Протокол удаленной загрузки клиента
69	tftp	Упрощенный протокол передачи файлов TFTP
79	finger	Протокол получения информации о пользователях FINGER
80	http	Протокол передачи гипертекста HTTP
109	pop2	Протокол почтового ящика POP2
110	pop3	Протокол почтового ящика POP3
111	rpc	Протокол удаленного вызова процедур RPC
156	sqlserv	Служба SQL
161	snmp	Управляющий протокол SNMP

Все сказанное выше о портах в равной степени относится к обоим протоколам транспортного уровня. Нет никакой зависимости между назначением номеров для приложений, использующих протокол TCP, и приложений, работающих с протоколом UDP. Приложения, которые передают данные на уровень IP, используя протокол UDP, получают номера, называемые портами UDP. Аналогично приложениям, обращающимся к протоколу TCP, выделяются порты TCP. В том и другом случаях это могут быть как назначенные, так и динамические номера. Диапазоны чисел, из которых выделяются номера портов TCP и UDP, совпадают.

Протоколы TCP и UDP ведут для каждого номера порта две очереди: очередь пакетов, поступающих в данный порт из сети, и очередь пакетов, отправляемых данным портом в сеть. Процедура приема данных протоколом TCP (или UDP), поступающих от нескольких прикладных служб, называется мультиплексированием. Обратная процедура распределения протоколом TCP (или UDP) поступающих от сетевого уровня пакетов между набором высокоуровневых служб, идентифицированных номерами портов, называется демультиплексированием.

3.4 Логические соединения

Для надежной передачи данных между двумя прикладными процессами предусматривается установление логического соединения. Номер порта в совокупности с номером сети и номером конечного узла однозначно определяют прикладной процесс в сети. Этот набор идентифицирующих параметров (IP-адрес, номер порта) имеет название сокет (socket).

Каждый взаимодействующий процесс идентифицируется сокетом – парой (IP-адрес интерфейса, номер порта), а каждое соединение – парой сокетов взаимодействующих процессов. Каждый процесс одновременно может участвовать в нескольких соединениях.

Так, например, если (IP1, n1), (IP2, n2), (IP3, n3) – сокеты трех разных процессов, то возможно образование следующих соединений:

– соединение 1 – {(IP1, n1), (IP2, n2)};

– соединение 2 – {(IP1, n1), (IP3, n3)};

– соединение 3 – {(IP2, n2), (IP3, n3)}.

Каждая такая пара однозначно идентифицирует соединение. Сутью же понятия «соединение» является договоренность о параметрах, характеризующих процедуру обмена данными между двумя процессами. В протоколе TCP каждая сторона соединения посылает противоположной стороне следующие параметры:

– максимальный размер сегмента, который она готова принимать;

– максимальный объем данных (возможно несколько сегментов), которые она разрешает другой стороне передавать в свою сторону, даже если та еще не получила подтверждения на предыдущую порцию данных;

– начальный порядковый номер байта, с которого она начинает отсчет потока данных в рамках данного соединения.

После того как в результате переговорного процесса модулей TCP с двух сторон соединения параметры процедуры обмена определены, одни из них остаются постоянными в течение всего сеанса связи, а некоторые адаптивно изменяются.

Когда устанавливается несколько соединений, то может случиться, что несколько машин пошлют запросы на соединение, в которых указаны одинаковые порты источники и получатели. Однако путаницы с соединениями не возникает, потому что IP-адреса у всех машин разные, следовательно, каждое соединение будет однозначно определено своим сокетом.

3.5 Программирование обмена данными на основе транспортных протоколов

TCP должен взаимодействовать не только с протоколами нижележащего уровня, но и с протоколами и приложениями прикладного уровня. Связь с прикладным уровнем осуществляется с помощью набора сервисных примитивов. Сервисные примитивы определены в стандарте протокола, а для прикладных программ они доступны в форме библиотек работы с сокетами.

При установлении соединения каждая из сторон выполняет некоторые операции, называемые открытием соединения. Открытие может быть пассивным или активным. Как правило, одна из сторон производит активное открытие соединения, а другая – пассивное, тогда соединение устанавливается. Оба режима подчиняются четким правилам. Пассивное соединение еще иногда называют серверным, а активное – клиентским.

При активном соединении процесс прикладного уровня передает программному обеспечению TCP на той же ЭВМ сервисный примитив запроса на установление соединения с номером сокета, после чего TCP отправляет получателю запрос на установление соединения, затем ждет ответа. После установления соединения активный процесс (клиент) может инициировать прием или передачу данных.

При пассивном соединении прикладная программа переводит программное обеспечение TCP в режим ожидания запроса на соединение от удаленной системы. Когда поступает запрос, программное обеспечение TCP осуществляет установку соединения, после чего пассивный процесс (сервер) готов принимать и передавать данные.

Программный интерфейс сокетов изначально был разработан для операционной системы (ОС) UNIX. Библиотека функций, поддерживающих этот интерфейс, входит в ядро всех ОС типа UNIX и Linux. Однако принципы работы с этим программным интерфейсом применимы к большинству ОС, поддерживающих TCP/IP (например, в семействе ОС и оболочек типа Windows программный интерфейс сокетов реализован в динамической библиотеке Winsock.dll).

Для протокола TCP пассивное (на стороне сервера) соединение с сокетом приводит к выполнению следующих функций:

– создание сокета и установление его типа (в ОС типа UNIX функция socket);

– настройка сокета на конкретное соединение (указывает адрес и номер порта – в ОС типа UNIX – функция bind);

– создание очередей клиентов (в ОС типа UNIX – функция listen);

– ожидание приходящего запроса на соединение с сокетом (в ОС типа UNIX – функция accept);

– прием и передача данных от клиента (в ОС типа UNIX – функции read, write, send, recv и их модификации);

– закрытие соединения с клиентом (в ОС типа UNIX – функция close).

Получив входящий запрос на соединение, сервер должен решать как бы две задачи одновременно: обслуживать уже установленное с клиентом соединение в соответствии с прикладным протоколом (принимать и отдавать данные клиенту) и ожидать поступления новых запросов на соединение от других клиентов. Обычно в развитых ОС (к ним относятся все современные ОС) эта проблема решается за счет возможностей параллельного выполнения нескольких процессов. Сервер может породить новый процесс (или новую цепочку выполнения – thread), который и должен будет заняться обслуживанием уже установленного соединения, а основной процесс сервера может закрыть текущее соединение и вновь вернуться к ожиданию запросов на соединение от других клиентов. В ОС типа UNIX создание нового процесса решается с помощью функции fork, при этом за вновь созданным процессом сохраняются все соединения, сделанные в основном процессе.

Для протокола TCP активное (на стороне клиента) соединение с сокетом приводит к выполнению следующих функций:

– создание сокета и установление его типа (в ОС типа UNIX функция socket);

– установление соединения с сервером (указывает адрес и номер порта – в ОС типа UNIX – функция connect);

– прием и передача данных (в ОС типа UNIX – функции read, write, send, recv и их модификации);

– закрытие соединения с сервером (в ОС типа UNIX – функция close).

Клиент, как правило, не требует для своей работы параллельного выполнения нескольких процессов.

В среде программирования Borland Delphi существуют специальные классы, которые позволяют выполнять те же действия, что и библиотека сокетов в ОС UNIX. Они взаимодействуют с библиотекой Winsock.dll на основе специальных технологий ОС (ActiveX технологии и COM-объекты). В среде Borland Delphi версии 3.0 для целей клиентского и серверного соединений служит класс объектов TTCP; а в среде Borland Delphi версии 5.0 и выше для клиентского соединения существует класс объектов TClientSocket, а для серверного – TServerSocket. Естественно, пользователь может на основе базовых классов разрабатывать свои собственные классы, которые будут поддерживать соединения по определенным им самим прикладным протоколам.

Для того чтобы создать сокет, достаточно создать экземпляр объекта выбранного класса (TTCP – в среде Borland Delphi версии 3.0 на страничке компонент «Internet», TClientSocket или TServerSocket – в среде Borland Delphi версии 5.0 и выше также на страничке компонент «Internet»). Это можно выполнить при проектировании приложения в среде разработки или же средствами языка программирования при выполнении приложения. Чтобы специфицировать (настроить) сокет, необходимо созданному экземпляру объекта присвоить нужные значения в указанные свойства (properties) – как правило, это свойства с именами вида «Port» и «Host» (имена и состав свойств зависят от версии среды разработки). Это тоже можно сделать как в режиме проектирования приложения, так и командами присвоения свойств объекта в тексте программы. После этого сокет инициализирован и с ним можно работать.

Для работы сокета клиента необходимо открыть сокет (процедура Open устанавливает в Thru свойство Active), затем использовать процедуры установления соединения, передачи и приема данных, а в конце работы закрыть сокет (процедура Close). При удалении экземпляра объекта автоматически прекратит существование и связанный с ним сокет. Краткое описание основных свойств, событий и методов объектов приведено в пункте 3.8 «Справочные данные».

По установлению соединения наступит событие, которое программист должен соответствующим образом обработать. В среде Borland Delphi версии

3.0 программист сам должен создавать потоки выполнения для обслуживания соединения (для выполнения потоков служат экземпляры класса объектов TThread), а в среде Borland Delphi версии 5.0 и выше это можно сделать в автоматическом режиме.

Более подробную информацию о функциях программного интерфейса с сокетами можно получить в справке соответствующей среды разработки или специализированной литературе.

3.6 Пример реализации простейшего клиент-серверного приложения на основе сокетов

Запустить Delphi. На пустой форме (возможен вариант совмещения функций клиента и сервера в одном приложении) разместить два компонента с закладки Internet: TClientSocket и TServerSocket (в 6 и 7 версии Delphi они могут отсутствовать; для установки надо выполнить шаги: Component – Install Packages – Add – dclsockets60.bpl или dclsockets70.bpl).

Для клиентской части программы понадобятся два компонента TEdit (один для ввода ip-адреса или dns-имени сервера, другой – для ввода передаваемой информации) и одна кнопка «Послать» для соединения с сервером и отправки ему сообщения.

Для серверной части понадобится один компонент TMemo для выдачи служебной информации о подключениях и отображения принимаемых сообщений).

В свойствах TServerSocket: установить Port в 4000, затем – Active в true.

В обработчике Button1Click написать код для соединения с сервером:

```
ClientSocket1.Host:=Edit1.Text;
ClientSocket1.Port:=4000;
ClientSocket1.Active:=true;
```

Теперь, при нажатии на Button1 произойдет соединение с сервером, адрес которого был указан в Edit1. При установлении соединения произойдет событие ClientSocket1.OnConnect, которое, например, надо обработать так, чтобы на сервер передавалась строка из Edit2 и происходило отсоединение:

```
Socket.SendText(Edit2.Text);
ClientSocket1.Active:=False;
```

При подключении клиента к ServerSocket1 произойдет событие ServerSocket1.OnClientConnect, в котором нужно вывести следующую информацию:

```
Memo1.Lines.Add('Client connected from: '+Socket.RemoteHost);
```

При попытке передать клиентом на сервер сообщение на сервере произойдет событие ServerSocket1.OnClientRead, в котором текст считывается и выводится в Memo1:

```
ReceivedString:=''; {локальная переменная типа string}
while Socket.ReceiveLength>0 do
        ReceivedString:=ReceivedString+Socket.ReceiveText;
Memo1.Lines.Add('Received string: '+ReceivedString);
```

При отсоединении клиента произойдет событие OnClientDisconnect, в котором мы добавим в Memo1 соответствующее сообщение:

```
Memo1.Lines.Add('Client '+Socket.RemoteHost+' disconnected.');
```

Запустите проект и протестируйте его. В качестве ip-адреса можно использовать адреса соседних компьютеров, на которых запущены аналогичные программы или адрес 127.0.0.1 для тестирования на локальном компьютере.

Примечание: Оправка сообщений с сервера клиентам осуществляется с использованием тех же методов и событий соответственно на стороне сервера и клиентов. Отличие заключается в необходимости точного указания номера соединения в свойстве Connections[Index: Integer]: TcustomWinSocket. Например:

```
ServerSocket1.Socket.Connections[i].SendText('Test');
```

Здесь i = (0..N-1) – номер соединения, N – общее количество соединений.

3.7 Задание на лабораторную работу

1. Ознакомиться с теоретическим и справочным материалом.

2. Реализовать приведенное в пункте 2 простейшее приложение и ознакомиться с работой компонентов Socket.

3. Разработать приложение сервера и приложение клиента, позволяющее общаться между клиентами в режиме online (один сервер обслуживает несколько клиентов).

Приложение сервера постоянно опрашивает входящие соединения, хранит историю сообщений определенной глубины (например, 20), список зарегистрированных пользователей и другую информацию, необходимую для решения задачи.

Приложение клиента должно обеспечивать: регистрацию нового пользователя и его авторизацию по имени пользователя и паролю на сервере, передачу сообщений на сервер и прием сообщений от сервера, переданных на сервер другими клиентами.

3.8 Справочные данные

Основные свойства компонента ServerSocket:

Active – инициализирует соединение (открывает для прослушивания socket) ;

Name – имя экземпляра класса для использования в программе;

Port – номер порта, на котором ведется прослушивание входящих соединений.

Основные события компонента ServerSocket:

OnAccept – зафиксировано входящее соединение;

OnClientConnect – установлено соединение с клиентом;

OnClientDisconnect – разорвано соединение с клиентом;

OnClientError – ошибка при работе с клиентом;

OnClientRead – чтение данных, получаемых от клиента, только в этом событии разрешается принимать данные от клиента;

OnClientWrite – запись данных, передаваемых клиенту, только в этом событии разрешается передавать данные клиенту;

OnListen – сервер успешно проинициализирован для прослушивания socket.

Основные свойства компонента ClientSocket:

Active – инициализирует соединение;

Host – IP-адрес сервера, с которым устанавливается соединение;

Name – имя экземпляра класса для использования в программе;

Port – номер порта, на котором производится соединение с сервером.

Основные события компонента ClientSocket:

OnConnect – установлено соединение с сервером;

OnDisconnect – разорвано соединение с сервером;

OnError – ошибка при работе с сервером;

OnRead – чтение данных, получаемых от сервера, только в этом событии разрешается принимать данные от сервера;

OnWrite – запись данных, передаваемых серверу, только в этом событии разрешается передавать данные серверу.

Основные методы класса CustomWinSocket:

ReceiveText – принимает данные в виде строки символов;

SendText – отправляет данные в виде строки символов.

3.9 Контрольные вопросы

1. Назначение протокола TCP и принцип его работы.
2. Назначение протокола UDP и принцип его работы.
3. Назовите отличия протоколов TCP и UDP.
4. С какой целью используются порты?
5. В чем заключается сущность мультиплексирования и демультиплексирования?
6. Поясните сущность понятия «логическое соединение». Каким образом оно определяется в сети?
7. Поясните сущность активного режима работы сетевого приложения.
8. Поясните сущность пассивного режима работы сетевого приложения.

4 Взаимодействие прикладных программ с помощью протокола электронной почты SMTP

Цель работы: изучение принципов организации взаимодействия прикладных программ с помощью протокола электронной почты SMTP и приобретение практических навыков создания клиентских почтовых приложений, использующих протокол SMTP.

Необходимое оборудование: IBM PC-совместимый компьютер, подключенный к глобальной сети Internet, с установленным программным обеспечением Borland Delphi версии 5.0 и выше.

4.1 Модель протокола, команды и коды ответов SMTP

Для передачи сообщений по TCP-соединению большинство почтовых агентов пользуются протоколом SMTP (Simple Mail Transfer Protocol – простой протокол электронной почты).

SMTP принят в качестве стандартного метода передачи электронной почты в сети Internet. В качестве транспортного протокола SMTP использует TCP, соединение устанавливается через порт с номером 25. Для обслуживания этого соединения используется специальная программа, которая именуется почтовым сервером. Для формирования сообщения и установления соединения используется почтовая программа (утилита) пользователя.

Главной целью протокола SMTP является надежная и эффективная доставка электронных почтовых сообщений. Для реализации протокола требуется только надежный канал связи. Средой для SMTP может служить отдельная локальная сеть, система сетей или же вся всемирная сеть Internet. Если между отправителем и получателем письма имеется непосредственная связь, адрес пользователя имеет вид *имя_пользователя@адрес_ЭВМ*. Когда получатель находится на ЭВМ, которая не поддерживает соединение по протоколу SMTP, и передача происходит через промежуточный сервер, то адрес получателя письма может иметь иной вид, например:

имя_пользователя%имя_сервера@адрес_ЭВМ.

Адреса ЭВМ в сети Internet представляют собой имя домена и преобразуются в IP-адреса согласно протоколу DNS.

Протокол SMTP базируется на следующей модели коммуникаций: в ответ на запрос пользователя почтовая программа-отправитель сообщения устанавливает двустороннюю связь с программой-приемником (почтовым сервером). Получателем может быть оконечный или промежуточный адресат. Если необходимо, почтовый сервер может установить соединение с другим сервером и передать сообщение дальше. SMTP-команды генерируются отправителем и посылаются получателю. На каждую команду должен быть получен отклик.

Перечень команд протокола SMTP, определенный спецификацией RFC 821, приведен в таблице 4.1. Это внутренние команды протокола. Если пользователь использует для работы с электронной почтой некоторую утилиту, то эти команды ему недоступны. Они представляют интерес только при программировании взаимодействия программ на основе этого протокола.

В соответствии со спецификацией, помеченные крестиком команды обязаны присутствовать в любой реализации SMTP. Остальные команды SMTP могут быть реализованы дополнительно. Каждая SMTP-команда должна заканчиваться либо пробелом (если у нее есть аргумент), либо комбинацией CRLF (Carriage-Return, Line-Feed – возврат каретки, перевод строки; коды ASCII – 13 и 10 соответственно). В описании команд употребляется слово <данные>, а не <сообщение>. Этим подчеркивается, что, кроме текста, SMTP позволяет передавать и двоичную информацию, например, графические или звуковые файлы.

Команды состоят из ключевых слов, за которыми следует один или более параметров. Ключевое слово состоит из 4-х символов и отделено от аргумента одним или несколькими пробелами. Каждая командная строка заканчивается символами CRLF. Синтаксис команд протокола SMTP приведен в пункте 4.6 «Справочные данные».

В спецификации SMTP требуется, чтобы сервер отвечал на каждую команду SMTP-клиента. Сервер отвечает трехзначной комбинацией цифр, называе-

мой кодом ответа. Вместе с кодом ответа, как правило, передается одна или несколько строк текстовой информации.

Примечание: Несколько строк текста, как правило, сопровождают только команды EXPN и HELP. В спецификации SMTP, однако, ответ на любую команду может состоять из нескольких строк текста.

Каждая цифра в коде ответа имеет определенный смысл. Первая цифра означает, было ли выполнение команды успешным (2), неуспешным (5) или еще не закончилось (3). Как указано в приложении Е документа RFC 821, простой SMTP-клиент может анализировать только первую цифру в ответе сервера и на основании ее продолжать свои действия. Вторая и третья цифры кода ответа разъясняют значение первой. В лабораторной работе допускается не анализировать коды ответов. Можно использовать готовые коды, определенные RFC 821 и приведенные в пункте 4.6 «Справочные данные».

Через систему электронной почты передаются сообщения, которые должны иметь строго определенный формат. Любое почтовое сообщение можно разделить на три части: «конверт», заголовки и собственно текст. «Конверт» используется почтовым сервером, он содержит две команды – MAIL и RCPT. Заголовок используется почтовой программой пользователя. Он может содержать несколько специальных полей: From, To, Date, Subject и др. Каждое из этих полей содержит имя, за которым после двоеточия идет его значения. При желании, пользователь может ознакомиться с содержимым всех полей заголовка письма.

4.2 Кодировка сообщений

Текст сообщения должен передаваться в виде 7-разрядных символов ASCII. Конец сообщения представляет собой строку, содержащую только символы точки «.» и перевода строки. Если по каким-то причинам такая строка непосредственно встречается внутри текста сообщения, то передающая сторона автоматически дублирует точку, чтобы принимающая сторона не приняла ее за конец сообщения (на принимающей стороне дублированный символ точки в начале строки заменяется одной точкой).

С 7-битной кодировкой сообщений связана проблема пересылки писем, написанных на национальных языках, в т.ч. и на русском, поскольку для русского алфавита требуется 8-битная кодировка. Большинство современных почтовых серверов поддерживают не только 7-битные, но и 8-битные кодировки, однако для совместимости со старыми почтовыми системами может использоваться специальная схема преобразования 8-битных кодов в 7-битные (естественно, при этом увеличивается объем сообщения, поскольку один 8-битный символ заменяется на специальную последовательность 7-битных).

Более сложная проблема заключается в том, что нет единого стандарта на кодировку русского алфавита, и поэтому на сторонах отправителя и получателя могут использоваться различные кодировки. Для того чтобы сообщение, посланное отправителем в одной кодировке, было понятно адресату, который применяет другую кодировку, почтовый сервер использует специальные таблицы смены кодировок. С помощью них текст сообщения, посланного в одной кодировке, преобразуется в другую кодировку. Чтобы сервер знал, какую кодировку использовать, информация об исходной кодировке включается почтовой системой отправителя в текст или в заголовок сообщения (в зависимости от используемой системы). Одно сообщение в процессе своего прохождения через почтовые сервера может претерпевать несколько перекодировок. В случае, когда у сервера нет информации об используемой кодировке или тип кодировки указан неверно (непонятен серверу), текст сообщения может исказиться.

4.3 Процесс передачи сообщений

Передача сообщения по протоколу SMTP происходит следующим образом: после установления соединения стороны обмениваются кодами аутентификации (с помощью команд HELO), затем одна из них посылает команду MAIL, в которой указан адрес отправителя и сведения о письме. Если получатель готов к приему сообщения, он посылает положительный отклик. Далее отправитель посылает одну или несколько команд RCPT, в которых идентифицирует адресатов сообщения. Если получатель может принять сообщение для указанного адресата, то он снова выдает положительный отклик на каждую команду. После

этого передается команда DATA, за которой следует само письмо (сколько бы не было адресатов, само письмо передается только один раз). В конце сеанса отправитель дает команду QUIT.

Конечно, маловероятно, что при приеме почтовым сервером сообщения, адресованному некоторому пользователю ЭВМ, этот пользователь установит с сервером непосредственную связь по протоколу SMTP. Обычно сервер размещает сообщение в почтовом ящике пользователя. Почтовый ящик – это некоторое промежуточное хранилище электронных сообщений, позволяющее почтовому серверу осуществить их передачу адресату не сразу, а в тот момент, когда он установит связь с сервером. Обычно сообщения хранятся в почтовом ящике не вечно – по истечении некоторого времени почтовый сервер удаляет их оттуда. Политика обслуживания почтовых ящиков регламентируется администратором системы.

Для того чтобы получить сообщение из своего почтового ящика, почтовая программа пользователя соединяется с сервером уже не по протоколу SMTP, а по специальному почтовому протоколу получения сообщений. Такой протокол позволяет работать с почтовым ящиком: забирать сообщения, удалять сообщения, сортировать их и выполнять другие операции. Самым популярным в настоящее время протоколом такого рода является протокол POP3 (Post Office Protocol v.3). Он предусматривает соединение с почтовым сервером на основе транспортного протокола TCP через порт 110. Другой, более сложный, известный протокол – IMAP4.

Во многих версиях ОС UNIX протокол SMTP, а также другие почтовые протоколы, реализованы в простейших утилитах типа sendmail или mmdf. Первая из них может служить как почтовым клиентом, так и почтовым сервером. Обычно она работает в фоновом режиме и ждет поступления сообщений от пользователя или извне. В ОС типа Windows NT ту же роль играет утилита, именуемая mdaemon (mail daemon).

Для непосредственного взаимодействия с пользователем существует простейшая программа, именуемая mail или mailx. Однако в современных ОС она

используется редко, поскольку существует большое количество прикладных программ с развитым интерфейсом, которые предоставляют пользователю возможности работы с электронной почтой. Примерами таких программ могут служить широко известные пакеты Outlook Express (производство фирмы Microsoft) и Netscape Communicator (производство фирмы Netscape). Внешний интерфейс этих программных продуктов чрезвычайно прост, так что использовать их может даже неквалифицированный пользователь. Все эти программы формируют сообщения в требуемом формате и передают их утилитам ОС, работающим в фоновом режиме, для отправки. К ним же они обращаются и для приема сообщений.

4.4 Пример последовательности команд почтовой транзакции

Каждой строке присвоен номер и обозначено, кому они принадлежат – передатчику (C) или приемнику (S). Текст справа от двоеточия содержит действительно передаваемые данные. Трехзначные цифровые комбинации в начале передаваемых строк обозначают коды ответа. Ответ SMTP похож на сообщения-подтверждения о доставке, поскольку появляется лишь в том случае, когда приемник получил данные.

1. S: 220 mail.ru ESMTP Wed, 07 Sep 2005 09:21:58 +0400
2. C: Helo XName
3. S: 250 mx1.mail.ru Hello XName [62.76.36.11]
4. C: mail from: klimov@mail.ru
5. S: 250 OK
6. C: rcpt to: ivanov@yandex.ru
7. S: 250 Accepted
8. C: rcpt to: petrov@mail.ru
9. S: 250 Accepted
10. C: data
11. S: 354 Enter message, ending with "." on a line by itself
12. C: From: klimov@mail.ru
13. C: To: ivanov@yandex.ru
14. C: Subject: Тема
15. C: Текст сообщения
16. C: простой
17. C: .
18. S: 250 OK id=1ECsNi-000ATp-00
19. C: quit

20. S: 221 mx1.mail.ru closing connection

Как видно из строки 1, когда SMTP-клиент устанавливает TCP-соединение с портом протокола 25, SMTP-сервер отвечает кодом 220. Это означает, что соединение успешно установлено.

После того как почтовые агенты компьютеров установили соединение и обменялись приветствиями, первой командой, согласно спецификации, должна быть команда HELO. Как указано в строке 2, SMTP-клиент передает HELO, указывая имя своего компьютера в качестве аргумента (имя может быть вымышленным).

В ответ на HELO приемник выдает код 250, сообщая передатчику о том, что команда принята и обработана.

После установления TCP-соединения и идентификации (при помощи HELO) SMTP-клиент приступает к почтовой транзакции. Для начала он выполняет одну из следующих команд: MAIL, SEND, SOML или SAML. В нашем примере использована команда MAIL.

После того как сервер выдал код ответа 250 (строка 5), согласившись обработать сообщение от klimov@mail.ru, необходимо указать получателя сообщения. Это делается при помощи команды RCPT. Команда RCPT имеет аргумент – имя получателя. На одну команду приходится только одно имя, поэтому, если получателей несколько, команда RCPT выдается несколько раз. В нашем примере команды RCPT выполняются в строках 6 и 8. Выдав команду RCPT, клиент ожидает получить ответ с кодом 250.

После того как посланы все команды RCPT, клиент начинает передачу данных при помощи команды DATA. В строке 10 показано, как клиент (передатчик) высылает команду DATA, в строке 11 – как сервер отвечает кодом 354. Этот код означает, что передача данных разрешена и должна заканчиваться комбинацией CRLF-точка-CRLF (новой строкой, содержащей только точку).

После того как получен код 354, клиент может начать передачу данных. Сервер, в свою очередь, помещает принятые данные в очереди входящих сооб-

щений. Сервер не высылает никаких ответов до тех пор, пока не получит комбинацию CRLF-точка-CRLF от клиента, означающую конец передачи данных. Как показано в строках 17 и 18, в ответ на полученную комбинацию CRLF-точка-CRLF сервер выдает код 250, который означает успешное окончание операции.

Для того чтобы закончить почтовую транзакцию, клиент, по правилам SMTP, обязан послать команду QUIT. Сервер, в свою очередь, отвечает кодом 221. Этот код подтверждает клиенту, что соединение будет закрыто, после чего соединение действительно закрывается.

В любой момент во время транзакции клиент может использовать команды NOOP, HELP, EXPN и VRFY. В ответ на каждую команду сервер высылает клиенту определенную информацию. В зависимости от ответа, клиент может предпринять определенные действия, однако в спецификации SMTP это не оговаривается.

4.5 Задание на лабораторную работу

1. Ознакомиться с теоретическим и справочным материалом.

2. Получить у преподавателя доменное имя или IP-адрес SMTP-сервера, через который возможна отправка сообщений из учебных аудиторий.

3. Разработать приложение почтового клиента, позволяющее выполнять отправку текстовых сообщений по протоколу SMTP указанным адресатам.

Приложение использует для обмена с сервером транспортный протокол TCP/IP. Сообщение должно включать поля From, To, Subject. Пользователь должен иметь возможность заполнить необходимые поля и дать команду на отправку сообщения. Процесс обмена командами и ответами должен происходить в автоматическом режиме и отображаться на экране в формате, приведенном в примере.

В случае возвращения сервером кода ответа, отличающегося от требуемого, необходимо вывести сообщение об этом и разорвать соединение.

4.6 Справочные данные

Таблица 4.1 – Команды протокола SMTP

Команда	Обязательна	Описание
HELO	X	Идентифицирует модуль-передатчик для модуля-приемника (hello).
MAIL	X	Начинает почтовую транзакцию, которая завершается передачей данных в один или несколько почтовых ящиков (mail).
RCPT	X	Идентифицирует получателя почтового сообщения (recipient).
DATA		Строки, следующие за этой командой, рассматриваются получателем как данные почтового сообщения. В случае SMTP, почтовое сообщение заканчивается комбинацией символов: CRLF-точка-CRLF.
RSET		Прерывает текущую почтовую транзакцию (reset).
NOOP		Требует от получателя не предпринимать никаких действий, а только выдать ответ ОК. Используется для тестирования. (No operation).
QUIT		Требует выдать ответ ОК и закрыть текущее соединение.
VRFY		Требует от приемника подтвердить, что ее аргумент является действительным именем пользователя.
SEND		Начинает почтовую транзакцию, доставляющую данные на один или несколько терминалов (а не в почтовый ящик).
SOML		Начинает транзакцию MAIL или SEND, доставляющую данные на один или несколько терминалов или в почтовые ящики.
SAML		Начинает транзакцию MAIL и SEND, доставляющие данные на один или несколько терминалов и в почтовые ящики.
EXPN		Команда SMTP-приемнику подтвердить, действительно ли аргумент является адресом почтовой рассылки, и если да, вернуть адрес получателя сообщения (expand).
HELP		Команда SMTP-приемнику вернуть сообщение-справку о его командах.

Таблица 4.2 – Коды ответа протокола SMTP

Код	Описание
211	Ответ о состоянии системы или помощь.
214	Сообщение-подсказка (помощь).
220 <имя_домена>	Служба готова к работе.
221 <имя_домена>	Служба закрывает канал связи.
250	Запрошенное действие почтовой транзакции успешно завершилось.
251	Данный адресат не является местным; сообщение будет передано по маршруту <forward-path>.
354	Начинай передачу сообщения. Сообщение заканчивается комбинацией CRLF-точка-CRLF.
421 <имя_домена>	Служба недоступна; соединение закрывается.
450	Запрошенная команда почтовой транзакции не выполнена, так как почтовый ящик недоступен.
451	Запрошенная команда не выполнена; произошла локальная ошибка при обработке сообщения.
452	Запрошенная команда не выполнена; системе не хватило ресурсов.
500	Синтаксическая ошибка в тексте команды; команда не опознана.
501	Синтаксическая ошибка в аргументах или параметрах команды.
502	Данная команда не реализована.
503	Неверная последовательность команд.
504	У данной команды не может быть аргументов.
550	Запрошенная команда не выполнена, так как почтовый ящик недоступен.
551	Данный адресат не является местным; попробуйте передать сообщение по маршруту <forward-path>.
552	Запрошенная команда почтовой транзакции прервана; дисковое пространство, доступное системе, переполнилось.
553	Запрошенная команда не выполнена; указано недопустимое имя почтового ящика.
554	Транзакция не выполнена.

Синтаксис команд протокола SMTP (SP – пробел):

HELO <SP> <domain> <CRLF>

MAIL <SP> FROM:<reverse-path> <CRLF>

RCPT <SP> TO:<forward-path> <CRLF>

DATA <CRLF>

RSET <CRLF>

SEND <SP> FROM:<reverse-path> <CRLF>

SOML <SP> FROM:<reverse-path> <CRLF>

SAML <SP> FROM:<reverse-path> <CRLF>

VRFY <SP> <string> <CRLF>

EXPN <SP> <string> <CRLF>

HELP <SP> <string> <CRLF>

NOOP <CRLF>

QUIT <CRLF>

4.7 Контрольные вопросы

1. Назначение протокола SMTP.

2. Опишите модель работы протокола SMTP.

3. Каковы особенности кодировки почтовых сообщений?

4. Опишите типовую последовательность действий почтового клиента при отправке сообщения.

5. Какие операции должны выполняться во время почтовой транзакции? Какими командами она открывается и закрывается?

6. По каким принципам формируются команды и коды ответов протокола SMTP?

5 Взаимодействие прикладных программ с помощью протокола электронной почты POP3

Цель работы: изучение принципов организации взаимодействия прикладных программ с помощью протокола электронной почты POP3 и приобретение практических навыков создания клиентских почтовых приложений, использующих протокол POP3.

Необходимое оборудование: IBM PC-совместимый компьютер, подключенный к глобальной сети Internet, с установленным программным обеспечением Borland Delphi версии 5.0 и выше.

5.1 Модель протокола POP3, его назначение и стадии POP3-сессии

Post Office Protocol (POP) – протокол доставки почты пользователю из почтового ящика почтового сервера POP. Многие концепции, принципы и понятия протокола POP выглядят и функционируют подобно SMTP. Команды POP практически идентичны командам SMTP, отличаясь в некоторых деталях. Сервер POP находится между агентом пользователя и почтовыми ящиками.

В настоящее время существуют две версии протокола POP – POP2 и POP3, обладающие примерно одинаковыми возможностями, однако несовместимые друг с другом. У POP2 и POP3 разные номера портов протокола (109 и 110 соответственно). Протокол POP3 не является расширением или модификацией POP2 – это совершенно другой протокол. POP2 определен в документе RFC 937 (Post Office Protocol-Version 2, Butler, et al, 1985), а POP3 – в RFC 1225 (Post Office Protocol-Version 3, Rose, 1991). POP3 разработан с учетом специфики доставки почты на персональные компьютеры и имеет соответствующие операции для этого.

Конструкция протокола POP3 обеспечивает возможность пользователю обратиться к своему почтовому серверу и изъять накопившуюся для него почту. Пользователь может получить доступ к POP-серверу из любой точки доступа к Internet. При этом он должен запустить специальный почтовый агент, работающий по протоколу POP3, и настроить его для работы со своим почтовым

сервером. Сообщения доставляются клиенту по протоколу POP, а посылаются при помощи SMTP. То есть на компьютере пользователя существуют два отдельных агента-интерфейса к почтовой системе – доставки (POP) и отправки (SMTP).

В протоколе POP3 оговорены три стадии процесса получения почты: авторизация, транзакция и обновление. После того как сервер и клиент POP3 установили соединение, начинается стадия авторизации. На стадии авторизации клиент идентифицирует себя для сервера. Если авторизация прошла успешно, сервер открывает почтовый ящик клиента и начинается стадия транзакции. В ней клиент либо запрашивает у сервера информацию (например, список почтовых сообщений), либо просит его совершить определенное действие (например, выдать почтовое сообщение). На стадии обновления сеанс связи заканчивается.

В этом режиме POP3 сервер освобождает все занятые ресурсы и завершает работу. После этого TCP соединение закрывается. У POP3 сервера может быть INACTIVITY AUTOLOGOUT таймер. Этот таймер должен быть, по крайней мере, с интервалом 10 минут. Это значит, что если клиент и сервер не взаимодействуют друг с другом, сервер автоматически прерывает соединение и при этом не переходит в режим обновления.

5.2 Формат сообщений

Почтовое сообщение состоит из двух частей: заголовка и тела письма. Между ними расположена пустая строка. Каждое поле заголовка состоит в свою очередь из имени поля и значения, которые разделяются двоеточием. Полей заголовка может быть множество, но наиболее часто встречаются только некоторые из них. Поля **To:** и **From:** указывают на получателя и отправителя письма соответственно, поле **Subject:** используется для передачи темы письма. **Reply-To:** применяется для указания обратного адреса, а **CC:** – копии сообщения. **Received:** – такую строку добавляет каждый почтовый сервис, через который прошло письмо. По нему легко отследить путь письма. **Date:** – означает дату создания письма. Используется специальный стандарт с указанием часового пояса отправителя относительно Гринвича.

Существует множество не стандартизованных полей заголовка. Они начинаются с символа X. Это могут быть такие поля, как **X-Mailer:** – почтовая программа отправителя; **X-MSMail-Priority:**, **X-Priority:** – приоритет (важность) письма и т.д. Разработчик также может создавать и использовать собственные поля, начинающиеся с **X-**.

Дополнением к традиционной электронной почте является ее расширение MIME (Multupurpose Internet Mail Extentions). Оно не требует каких-либо переделок в почтовых серверах, но позволяет снять с электронной почты привычные ограничения и предоставляет возможность пересылать любую информацию. Например, MIME позволяет создавать многосекционные письма. В каждой секции описывается тип информации, находящейся в ней (**Content-Type:**) и вид кодирования (**Content-Transfer-Encoding:**). Секции могут содержать как простой текст (**Content-Type: text/plain**) и текст в формате HTML (**Content-Type: text/html**), так и произвольные файлы в прикреплениях письма (**attach**).

Так как электронная почта изначально разрабатывалась как служба по пересылке обычного текста, это наложило ограничение на передаваемую информацию. Для передачи двоичной информации требуется специальное кодирование. Используется несколько методов кодирования: для текста с использованием только латинских букв, цифр и некоторых спецсимволов обычно используется **Content-Transfer-Encoding: 7bit**. Если сюда еще добавить русские буквы, то используется **Content-Transfer-Encoding: 8bit**. Для передачи HTML-кода применяют **Content-Transfer-Encoding: quoted-printable**. Чтобы отличить код от текста, перед ним ставится символ «=». Например, фраза «Привет!» будет выглядеть в закодированном виде как «=CF=F0=E8=E2=E5=F2!». Для двоичных файлов (архивы, изображения и т.д.) используют **Content-Transfer-Encoding: base64**, когда каждые 6 байт преобразуются в 8 печатных символов.

5.3 Процесс получения сообщений. Команды и ответы протокола POP3

Команды POP3 состоят из ключевых слов, за некоторыми следует один или более аргументов. Все команды заканчиваются парой CRLF. Ключевые

слова и аргументы состоят из ASCII символов. Ключевое слово и аргументы разделены одиночным пробелом. Ключевое слово состоит от 3-х до 4-х символов, а аргумент может быть длиной до 40 символов.

Ответы в POP3 состоят из индикатора состояния и ключевого слова, за которым может следовать дополнительная информация. Ответ заканчивается парой CRLF. Существует только два индикатора состояния: «+OK» – положительный и «-ERR» – отрицательный. Синтаксис и описание команд и ответов протокола POP3 приведены в пункте 5.5 «Справочные данные»

Стадия авторизации

Как только будет установлено TCP соединение с POP3 сервером, он отправляет приглашение, заканчивающееся парой CRLF, например:

S: +OK POP3 server ready

Теперь POP3 сессия находится в режиме авторизации. Клиент должен идентифицировать себя на сервере, используя команды USER и PASS. Сначала необходимо отправить команду USER, после которой в качестве аргумента следует имя пользователя. Если сервер отвечает положительно, то необходимо отправить команду PASS, аргументом которой является пароль. Если после отправки команды USER или PASS сервер отвечает негативно, то можно попытаться авторизироваться снова или выйти из сессии с помощью команды QUIT. После успешной авторизации сервер открывает и блокирует maildrop (почтовый ящик). В ответе на команду PASS сервер информирует, сколько сообщений находится в почтовом ящике, и передает их общий размер. Синтаксис, описание и возможные ответы на команды приведены в справочном разделе.

Стадия транзакции

После успешной идентификации пользователя на сервере POP3 сессия переходит в режим транзакции, где пользователь может передавать соответствующие команды (см. справочный раздел). После каждой из таких команд следует ответ сервера. В режиме транзакции доступны действия: получение статистической информации о корреспонденции, получение списка писем, чтение выбранного письма, чтение определенного количества строк текста выбранного

письма, чтение идентификаторов всех или выбранного сообщения, пометка сообщения как удаленного, сброс отметок об удалении и др.

Стадия обновления

Когда клиент передает команду QUIT в режиме транзакции, то сессия переходит в режим обновления. В этом режиме сервер удаляет все сообщения, помеченные для удаления. После этого TCP соединение закрывается с соответствующим ответом сервера.

5.4 Задание на лабораторную работу

1. Ознакомиться с теоретическим и справочным материалом.

2. Получить у преподавателя доменное имя или IP-адрес POP3-сервера, с которого возможно получение почтовых сообщений в учебных аудиториях, имя почтового ящика и пароль.

3. Разработать приложение почтового клиента, позволяющее выполнять получение текстовых сообщений по протоколу POP3 из почтового ящика.

Приложение должно поддерживать все приведенные в справочном разделе команды, кроме UIDL. Приложение использует для обмена с сервером транспортный протокол TCP/IP. Процесс обмена командами и ответами должен отображаться на экране.

Предполагается, что в заголовке и тексте сообщения не используются коды русских букв. То есть в приложении допускается отсутствие функций работы с другими методами кодирования, кроме Content-Transfer-Encoding: 7bit.

5.5 Справочные данные

Ниже приведено описание команд и ответов протокола POP3.

USER <SP> <name> <CRLF>

Возможные ответы:

+OK name is a valid mailbox

-ERR never heard of mailbox name

Описание. Первый шаг авторизации на сервере. Если ответом на эту команду является строка индикатора «+OK», клиент может отправлять команду

PASS – ввод пароля или QUIT – завершить сессию. Если ответом является строка «-ERR», клиент может либо повторить команду USER, либо закрыть сессию. Аргумент: name - строка, идентифицирующая почтовый ящик системы.

PASS <SP> <passw> <CRLF>

Возможные ответы:

+OK maildrop locked and ready

-ERR invalid password

-ERR unable to lock maildrop

Описание. Второй шаг авторизации на сервере. Если ответом на эту команду является строка индикатора «+OK», авторизация прошла успешно и сессия переходит в состояние транзакции. Если ответом является строка «-ERR», то пользователь либо ввел неправильный пароль, либо неверно указал почтовый ящик. Аргумент: passw – строка-пароль.

QUIT <CRLF>

Возможные ответы:

+OK

Описание. Завершение сессии. При появлении команды QUIT в состоянии транзакции сессия переходит в состояние обновления и осуществляется удаление помеченных сообщений.

STAT <CRLF>

Возможные ответы:

+OK <n> <s>

Описание. Команда STAT используется для просмотра текущего состояния почтового ящика: n – количество сообщений, s – их общий объем. Сообщения, помеченные как удаленные, не учитываются.

LIST [<SP> <mes>] <CRLF>

Возможные ответы:

+OK scan listing follows

<n1> <s1>

<n2> <s2>…

-ERR no such message

Описание. Команда LIST может передаваться как с аргументом mes – номером сообщения, так и без аргумента. Если команда содержит аргумент и сообщение с указанным номером существует, ответом на нее будет «информационная строка», которая содержит номер сообщения n1 и размер сообщения s1 в байтах. Если аргумент не указан – ответом будет список информационных строк ni si обо всех сообщениях в данном почтовом ящике. Сообщения, помеченные как удаленные, в этом списке не приводятся.

RETR <SP> <mes> <CRLF>

Возможные ответы:

+OK message follows

<заголовок и тело сообщения>

-ERR no such message

Описание. Используется для передачи клиенту запрашиваемого сообщения. Аргумент команды: mes – номер сообщения.

DELE <SP> <mes> <CRLF>

Возможные ответы:

+OK message deleted

-ERR no such message

Описание. По команде DELE сообщение с указанным в аргументе mes номером помечается как удаленное. Перенумерация остальных сообщений не производится. Физическое удаление помеченных сообщений происходит только при переходе сессии в стадию обновления.

NOOP <CRLF>

Возможные ответы:

+OK

Описание. Используется для проверки состояния сервера. Никаких действий не производится. Ответ всегда положительный.

RSET <CRLF>

Возможные ответы:

+OK

Описание. По команде RSET со всех сообщений снимаются пометки об удалении.

TOP <SP> <mes> <SP> <n> <CRLF>

Возможные ответы:

+OK

< заголовок сообщения + первые n строк из тела сообщения>

-ERR no such message

Описание. По команде TOP передается заголовок и n первых строк сообщения с номером mes. Если количество строк в сообщении меньше указанного в параметре n, пользователю передается все сообщение. Если n=0, то передается только заголовок.

UIDL [<SP> <mes>] <CRLF>

Возможные ответы:

+OK unique-id listing follows

<№ сообщения1 + ID1, № сообщения2 + ID2, …>

-ERR no such message

Описание. Если указан номер сообщения mes, то передается уникальный идентификатор для этого сообщения. Если аргумент не был передан, то передаются идентификаторы всех сообщений, кроме помеченных для удаления.

5.6 Контрольные вопросы

1. Охарактеризуйте модель протокола POP3.

2. Назовите и поясните назначение стадий POP3-сессии.

3. Опишите формат почтового сообщения.

4. Назовите особенности и методы кодирования информации в почтовых сообщениях.

5. По каким принципам формируются команды и коды ответов протокола POP3?

6. Какие операции выполняются на стадиях авторизации и обновления?

7. Какие операции выполняются на стадии транзакции?

6 Взаимодействие прикладных программ с помощью протокола передачи данных FTP

Цель работы: изучение принципов организации взаимодействия прикладных программ с помощью протокола передачи данных FTP и приобретение практических навыков создания клиентских приложений, использующих протокол FTP.

Необходимое оборудование: IBM PC-совместимый компьютер с установленным программным обеспечением: Borland Delphi версии 5.0 и выше, FTP-сервер Babyftp 1.0.

6.1 Назначение и модели работы протокола FTP

FTP (File Transfer Protocol – протокол передачи данных) – один из старейших протоколов в Internet и входит в его стандарты. Первые спецификации FTP относятся к 1971 году. С тех пор FTP претерпел множество модификаций и значительно расширил свои возможности. FTP может использоваться как в программах пользователей, так и в виде специальной утилиты операционной системы.

FTP предназначен для решения задач разделения доступа к файлам на удаленных узлах, прямого или косвенного использования ресурсов удаленных компьютеров, обеспечения независимости клиента от файловых систем удаленных узлов, эффективной и надежной передачи данных.

Обмен данными в FTP происходит по TCP-каналу. Обмен построен на технологии «клиент-сервер». FTP не может использоваться для передачи конфиденциальных данных, поскольку не обеспечивает защиты передаваемой информации и передает между сервером и клиентом открытый текст. FTP-сервер может потребовать от FTP-клиента аутентификации. Однако пароль и идентификатор пользователя будут переданы от клиента на сервер открытым текстом.

Простейшая модель работы протокола FTP представлена на рисунке 6.1. FTP соединение инициируется интерпретатором протокола пользователя. Управление обменом осуществляется по каналу управления в стандарте прото-

кола TELNET. Команды FTP генерируются интерпретатором протокола пользователя и передаются на сервер. Ответы сервера отправляются пользователю также по каналу управления.

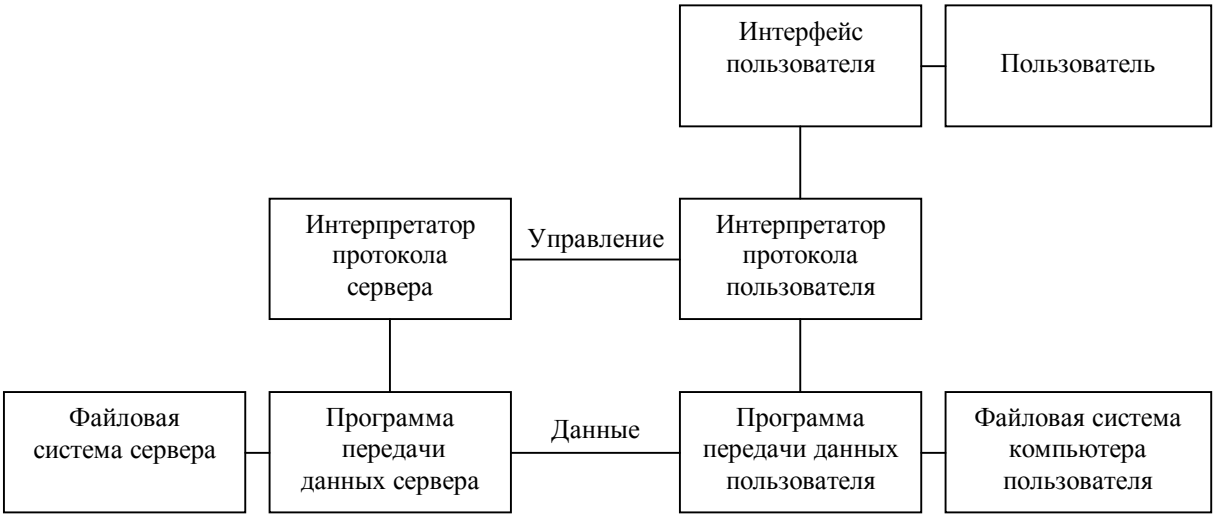


Рисунок 6.1 – Модель работы протокола FTP

Команды FTP определяют параметры канала передачи данных и самого процесса передачи. Они также определяют и характер работы с удаленной и локальной файловыми системами.

Сессия управления инициализирует канал передачи данных. При организации канала передачи данных последовательность действий другая, отличная от организации канала управления. В этом случае сервер инициирует обмен данными в соответствии с согласованными в сессии управления параметрами.

Канал данных устанавливается для того же узла, что и канал управления, через который ведется настройка канала данных. Канал данных может быть использован как для приема, так и для передачи данных.

Алгоритм работы протокола FTP состоит в следующем:

1. Сервер FTP использует в качестве управляющего соединение на TCP порт 21, который всегда находится в состоянии ожидания соединения со стороны пользователя FTP.

2. После того как устанавливается управляющее соединение модуля «Интерпретатор протокола пользователя» с модулем сервера – «Интерпретатор

протокола сервера», пользователь (клиент) может отправлять на сервер команды. FTP-команды определяют параметры соединения передачи данных: роль участников соединения (активный или пассивный), порт соединения (как для модуля «Программа передачи данных пользователя», так и для модуля «Программа передачи данных сервера»), тип передачи, тип передаваемых данных, структуру данных и управляющие директивы, обозначающие действия, которые пользователь хочет совершить (например, сохранить, считать, добавить или удалить данные или файл и другие).

3. После того как согласованы все параметры канала передачи данных, один из участников соединения (пассивный модуль) становится в режим ожидания открытия соединения на определенный порт. После этого другая сторона (активный модуль) открывает соединение на указанный порт и начинается обмен данными.

4. После окончания передачи данных соединение между «Программой передачи данных сервера» и «Программой передачи данных пользователя» закрывается, но управляющее соединение «Интерпретатора протокола сервера» и «Интерпретатора протокола пользователя» остается открытым. Пользователь, не закрывая сессии FTP, может еще раз открыть канал передачи данных.

Возможна ситуация, когда данные должны передаваться на третий узел. В этом случае имеет место другая модель работы протокола FTP, при которой пользователь организует канал управления с двумя серверами и прямой канал данных между ними. Команды управления идут через пользователя, а данные – напрямую между серверами. Подробное рассмотрение этой модели работы протокола FTP выходит за рамки данной лабораторной работы.

6.2 Особенности управления процессом обмена данными

Основу передачи данных FTP составляет механизм установления соединения между соответствующими портами и выбора параметров передачи. Каждый участник FTP-соединения должен поддерживать порт передачи данных по умолчанию. По умолчанию «Программа передачи данных пользователя» использует тот же порт, что и для передачи команд, а «Программа передачи дан-

ных сервера» использует порт L-1, где L – управляющий порт. Однако для ускорения процессов обмена данными участники соединения часто используют другие порты передачи данных.

Передача данных может вестись в активном или пассивном режиме. Если доступ к ftp-серверу осуществляется через proxy-сервер, то возможна работа только в пассивном режиме.

Установление соединения передачи данных при активном режиме работы происходит следующим образом:

1. Клиент создает сокет на выбранном им порте P и активизирует его (переводит в режим ожидания соединения).

2. Клиент направляет серверу по управляющему соединению команду PORT (описание приведено ниже), в которой указывает свой IP-адрес и выбранный для передачи данных порт P.

3. В случае успешного получения и обработки команды PORT сервер по управляющему соединению отправляет клиенту положительный отклик. Затем сервер пытается соединиться с портом P клиента со своего локального порта данных A (может быть выбран сервером произвольно).

4. Клиент обнаруживает событие соединения на порт P и начинает процесс передачи данных с использованием управляющего соединения для отправки команд FTP-сервиса и соединения передачи данных для получения либо отправки данных.

Установление соединения передачи данных при пассивном режиме работы происходит следующим образом:

1. Клиент отправляет по управляющему соединению команду PASV, указывающую серверу о намерении клиента работать в пассивном режиме.

2. В случае успешного получения и обработки команды PASV сервер создает сокет на выбранном им порте P, активизирует его (переводит его в режим ожидания соединения) и отправляет клиенту по управляющему соединению отклик, в котором указываются параметры соединения (IP-адрес сервера и порт P).

3. Клиент, получив положительный отклик с параметрами соединения, создает на произвольном порте A сокет и активизирует его, то есть пытается соединиться с портом P сервера.

4. В случае успешного установления соединения клиент начинает процесс передачи данных с использованием управляющего соединения для отправки команд FTP-сервиса и соединения передачи данных для получения либо отправки данных.

Одновременно с передачей данных по установленному соединению в обоих режимах по каналу «Интерпретатор протокола сервера» – «Интерпретатор протокола пользователя» могут передаваться уведомления о получении данных. Протокол FTP требует, чтобы управляющее соединение было открыто, пока по каналу обмена данными идет передача.

Как правило, сервер FTP ответственен за открытие и закрытие канала передачи данных. Сервер FTP должен самостоятельно закрыть канал передачи данных в следующих случаях:

1. Сервер закончил передачу данных в формате, который требует закрытия соединения.
2. Сервер получил от пользователя команду «прервать соединение».
3. Пользователь изменил параметры порта передачи данных.
4. Было закрыто управляющее соединение.
5. Возникли ошибки, при которых невозможно возобновить передачу данных.

6.3 Команды и ответы протокола FTP

Все команды протокола FTP отправляются «Интерпретатором протокола пользователя» в текстовом виде – по одной команде в строке. Каждая строка команды – идентификатор и аргументы – заканчиваются символами CRLF. Имя команды отделяется от аргумента символом пробела.

Обработчик команд возвращает код обработки каждой команды, состоящий из трех цифр. Коды обработки составляют определенную иерархическую структуру и, как правило, определенная команда может возвратить только оп-

ределенный набор кодов. За кодом обработки команды следует символ пробела и текст пояснения. Описание команд и основных кодов ответов приведено в пункте 6.5 «Справочные данные».

Команды протокола FTP, которыми обмениваются «Интерпретатор протокола сервера» и «Интерпретатор протокола пользователя», можно разделить на три группы.

1. **Команды управления доступом к системе** обеспечивают авторизацию пользователя в системе, выход из нее и настройку некоторых текущих параметров соединения.

2. **Команды управления потоком данных** устанавливают параметры передачи данных. Все параметры, описываемые этими командами, имеют значение по умолчанию, поэтому команды управления потоком используются только тогда, когда необходимо изменить значение параметров передачи, используемых по умолчанию. Команды управления потоком могут подаваться в любом порядке, но все они должны предшествовать командам FTP-сервиса.

3. **Команды FTP-сервиса** определяют действия, которые необходимо произвести с указанными файлами. Как правило, аргументом команд этой группы является имя файла.

6.4 Задание на лабораторную работу

1. Ознакомиться с теоретическим и справочным материалом.

2. Запустить приложение простейшего ftp-сервера «babyftp.exe». Сервер поддерживает работу только с анонимными пользователями (имя пользователя – «anonymous», пароль – любой адрес электронного почтового ящика). Параметры доступа настраиваются в окне «Settings».

3. Разработать приложение ftp-клиента, реализующие следующие функции:

- создание и удаление папок;
- перемещение по папкам;
- чтение списка содержимого папки;

– получение, отправка, удаление и переименование файлов.

Для получения списка содержимого папки необходимо использовать активный режим. Для получения и отправки файлов – пассивный. Приложение использует для обмена с сервером транспортный протокол TCP/IP. Процесс обмена командами и ответами должен отображаться на экране.

Примечание: Для отправки файлов на сервер необходимо использовать метод SendStreamThenDrop класса TCustomWinSocket. Данный метод обеспечивает автоматическое закрытие соединения после завершения передачи файла. В этом случае для связывания метода с файлом необходимо создать файловый поток (класс TFileStream).

6.5 Справочные данные

Команды протокола FTP.

1. Команды управления доступом к системе

USER. Как правило, эта команда открывает сессию FTP между клиентом и сервером. Аргументом команды является имя (идентификатор) пользователя для работы с файловой системой. Эта команда может подаваться не только в начале, но и в середине сессии, если, например, пользователь желает изменить идентификатор, от имени которого будут проводиться действия. При этом все переменные, относящиеся к старому идентификатору, освобождаются. Если во время изменения идентификатора происходит обмен данными, обмен завершается со старым идентификатором пользователя.

PASS. Данная команда подается после ввода идентификатора пользователя и содержит в качестве аргумента пароль пользователя.

CWD. Команда обеспечивает возможность работы с различными каталогами удаленной файловой системы. Аргументом команды является строка, указывающая путь каталога удаленной файловой системы, в котором желает работать пользователь.

REIN. Команда реинициализации. Эта команда очищает все переменные текущего пользователя, сбрасывает параметры соединения. Если в момент по-

дачи команды происходит передача данных, передача продолжается и завершается с прежними параметрами.

QUIT. Команда закрывает управляющий канал. Если в момент подачи команды происходит передача данных, канал закрывается после окончания передачи данных.

2. Команды управления потоком данных

PORT. Команда указывает серверу адрес и порт, которые будут использоваться клиентом для прослушивания соединения. Синтаксис команды: «PORT<SP>h1,h2,h3,h4,p1,p2». Аргументами команды являются 32-битный IP адрес и 16-битный номер порта соединения. Эти значения разбиты на шесть 8-битных полей и представлены в десятичном виде, где hN – байты адреса (от старшего к младшему), а pN – байты порта (от старшего к младшему).

PASV. Эта команда отправляется серверу для указания, что он должен «слушать» соединение. Ответом на данную команду является строка, содержащая адрес и порт узла, находящегося в режиме ожидания соединения. Формат представления данных соответствует формату команды PORT.

Команды **TYPE**, **STRU**, **MODE** определяют, соответственно, тип передаваемых данных (ASCII, Image и другие), структуру или формат передачи данных (File, Record, Page), способ передачи (Stream, Block и другие). Использование этих команд очень важно при построении взаимодействия в гетерогенных средах и весьма отличающихся операционных и файловых систем взаимодействующих узлов.

3. Команды FTP-сервиса

RETR. Эта команда указывает модулю «Программа передачи данных сервера» передать копию файла, заданного параметром этой команды, модулю передачи данных на другом конце соединения.

STOR. Команда указывает модулю «Программа передачи данных сервера» принять данные по каналу передачи данных и сохранить их как файл, имя которого задано параметром этой команды. Если такой файл уже существует, он будет перезаписан, если нет, будет создан новый.

Таблица 6.1 – Основные коды ответов на команды протокола FTP

Код	Описание
110	Комментарий
125	Канал открыт, обмен данными начат
150	Статус файла правилен, подготавливается открытие канала
200	Команда корректна
220	Слишком много подключений к FTP-серверу (можете попробовать позднее). В некоторых версиях указывает на успешное завершение промежуточной процедуры
221	Успешное завершение по команде quit
225	Канал сформирован, но информационный обмен отсутствует
226	Закрытие канала, обмен завершен успешно
230	Пользователь идентифицирован, продолжайте
250	Запрос прошел успешно
331	Имя пользователя корректно, нужен пароль
332	Для входа в систему необходима аутентификация
421	Процедура невозможна, канал закрывается
425	Открытие информационного канала невозможно
426	Канал закрыт, обмен прерван
450	Запрошенная функция не реализована, файл недоступен, например, занят
451	Локальная ошибка, операция прервана
452	Ошибка при записи файла (недостаточно места)
500	Синтаксическая ошибка, команда не может быть интерпретирована (возможно, она слишком длинна)
501	Синтаксическая ошибка (неверный параметр или аргумент)
503	Неудачная последовательность команд
504	Команда не применима для такого параметра

RNFR и **RNTO.** Команды должны следовать одна за другой. Первая команда содержит в качестве аргумента старое имя файла, вторая – новое. Последовательное применение этих команд переименовывает файл.

ABOR. Команда предписывает серверу прервать выполнение предшествующей сервисной команды (например, передачу файла) и закрыть канал передачи данных.

DELE. Удаление указанного файла.

MKD. Создание указанного в аргументе каталога.

RMD. Удаление указанного в аргументе каталога.

LIST. Получение списка файлов в указанном каталоге. Передача списка осуществляется по соединению «Программа передачи данных сервера» – «Программа передачи данных клиента».

6.6 Контрольные вопросы

1. Сколько логических соединений необходимо для работы протокола FTP? Каким образом они используются?

2. Какие стадии можно выделить в FTP-сессии?

3. В чем заключается особенность работы протокола FTP в активном режиме?

4. В чем заключается особенность работы протокола FTP в пассивном режиме?

5. Какая сторона соединения отвечает за закрытие канала передачи данных? В каких случаях это происходит?

6. Опишите структуру команды и ответа протокола FTP.

7. На какие группы можно разбить команды протокола FTP? Каково назначение команд каждой из групп?

Список рекомендуемой литературы

1. Олифер, В.Г. Компьютерные сети. Принципы, технологии, протоколы: Учебник для вузов [Текст] / В.Г. Олифер, Н.А. Олифер. – СПб: Издательство «Питер», 2005, – 864 с.

2. Блэк, Ю. Сети ЭВМ: Протоколы, стандарты, интерфейсы [Текст] / Ю. Блэк. – М.: Мир, 1990. – 506 с.

3. Золотов, С. Протоколы в Internet: Руководство для профессионалов [Текст] / С. Золотов. – СПб.: BHV, 1998, – 304 с.

Приложение А

(обязательное)

Пример оформления титульного листа отчета по лабораторной работе

ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра «Информационные системы»

О Т Ч Е Т

о выполнении лабораторной работы

на тему: «_____»
по дисциплине «Сети ЭВМ и телекоммуникации»

Выполнил(и):_____ Шифр:_____
Ф. И. О.

Факультет:_____

Специальность:_____

Группа:_____

Орел 200_