ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

С.Г. Михальченко, Е.Ю. Агеев

ЭКСПЛУАТАЦИЯ И РАЗВИТИЕ КОМПЬЮТЕРНЫХ СИСТЕМ И СЕТЕЙ

Руководство к организации самостоятельной работы

ТОМСК — 2007

Федеральное агентство по образованию

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра промышленной электроники

С.Г. Михальченко, Е.Ю. Агеев

ЭКСПЛУАТАЦИЯ И РАЗВИТИЕ КОМПЬЮТЕРНЫХ СИСТЕМ И СЕТЕЙ

Руководство к организации самостоятельной работы

2007

Михальченко С.Г., Агеев Е.Ю.
Эксплуатация и развитие компьютерных систем и сетей: Руководство к организации самостоятельной работы. — Томск: Томский государственный университет систем управления и радиоэлектроники, 2007. — 127 с.

Рассмотрены вопросы доступа к среде передачи информации, способы коммутации и мультиплексирования, методы кодирования и адресации сетевых устройств. Составлен набор лабораторных работ для приобретения навыков сетевого программирования и проектирования коммуникационных сетей. Изучаются вопросы установки, настройки и обслуживания аппаратного и программного обеспечения компьютерных информационных сетей. Проработана рейтинговая система оценки работы студента.

Предназначено для студентов вузов, обучающихся по специальности «Промышленная электроника».

ОГЛАВЛЕНИЕ

Цели и задачи дисциплины, ее место в учебном процессе 5

1 Методы кодирования физического уровня..... 6

ЛАБОРАТОРНАЯ РАБОТА № 1. Полоса пропускания, амплитудно-частотная характеристика и методы кодирования физического уровня 14

2 Методы кодирования физического уровня. Аналоговая модуляция 17

ЛАБОРАТОРНАЯ РАБОТА № 2. Полоса пропускания, амплитудно-частотная характеристика и методы аналоговой модуляции 22

3 Общие сведения об основных протоколах стека TCP/IP..... 25

ЛАБОРАТОРНАЯ РАБОТА № 3. Служебные утилиты стека протоколов TCP/IP..... 45

ЛАБОРАТОРНАЯ РАБОТА № 4. Стек протоколов TCP\IP. Пассивное оборудование ЛВС..... 46

5 Программы-анализаторы пакетов..... 47

ЛАБОРАТОРНАЯ РАБОТА № 5. Анализ сетевого трафика, форматы пакетов протокола TCP\IP, инкапсуляция 50

6 Методы логического кодирования (Канальный уровень). Корректирующие и обнаруживающие коды. Код Хэмминга..... 53

ЛАБОРАТОРНАЯ РАБОТА № 6. Методы логического кодирования (Канальный уровень). Корректирующие и обнаруживающие коды. Код Хэмминга..... 56

7 Кодирование информации. Компрессия. Метод Хаффмана 59

ЛАБОРАТОРНАЯ РАБОТА № 7. Кодирование информации. Компрессия. Метод Хаффмана..... 64

8 Активное сетевое оборудование. Коммутатор..... 67

ЛАБОРАТОРНАЯ РАБОТА № 8. Настройка конфигурации коммутатора 3COM SuperStack 4226T 80

9 Адресация компьютерных сетей. Подсети TCP/IP 86

ЛАБОРАТОРНАЯ РАБОТА № 9. Маршрутизация компьютерных сетей. Подсети TCP/IP 91

10 Метод доступа CSMA/CD и обеспечение надежности Ethernet 94

ЛАБОРАТОРНАЯ РАБОТА № 10. Метод доступа CSMA/CD и обеспечение надежности Ethernet 100

11 Socket-программирование 103

ЛАБОРАТОРНАЯ РАБОТА № 11. Основы socket-программирования под Windows 109

12 Настройка подключения к Wi-Fi сети 114

ЛАБОРАТОРНАЯ РАБОТА № 12. Подключение к открытой беспроводной сети 123

Организация самостоятельной работы. Применение рейтинговой системы 125

Заключение 126

Учебно-методические материалы по дисциплине 127

ЦЕЛИ И ЗАДАЧИ ДИСЦИПЛИНЫ, ЕЕ МЕСТО В УЧЕБНОМ ПРОЦЕССЕ

Основная цель дисциплины — изучение программных и аппаратных средств передачи информации по телекоммуникационным сетям.

В процессе изучения дисциплины изучаются методы доступа к среде передачи, способы коммутации и мультиплексирования, методы кодирования и типы адресации устройств в сети. Изучаются наиболее популярные современные стеки сетевых протоколов и технологии передачи данных. Приобретаются навыки сетевого программирования и проектирования коммуникационных сетей. Изучаются вопросы установки, настройки и обслуживания аппаратного и программного обеспечения компьютерных информационных сетей.

Для изучения дисциплины требуются начальные знания по дисциплинам «Информатика», «Программирование», «Аппаратное и программное обеспечение ЭВМ», «Операционные системы».

1 МЕТОДЫ КОДИРОВАНИЯ ФИЗИЧЕСКОГО УРОВНЯ

Потенциальный код с нулем. Самый очевидный способ кодирования состоит в задании бит двоичного сигнала последовательностью потенциалов (рис. 1, *а*): высокий уровень — для кодирования логической единицы и низкий — для нуля. Такой метод называется потенциальным кодом. Он используется при передаче данных между контроллерами компьютера.

Потенциальный код без возвращения к нулю. Другой метод потенциального кодирования (рис. 1, *б*), называется потенциальным кодированием без возвращения к нулю (***Non Return to Zero, NRZ***) из-за того, что при передаче последовательности единиц сигнал не возвращается к нулю. Метод NRZ из-за двух потенциалов разной полярности обладает хорошей распознаваемостью ошибок, но не обладает свойством самосинхронизации, т.к. при передаче длинной последовательности единиц или нулей сигнал на линии не изменяется. Из-за этого приемник лишен возможности определять по входному сигналу моменты времени, когда нужно в очередной раз считывать данные. Даже при наличии высокоточного тактового генератора приемник может рассинхронизоваться с передатчиком, так как частоты двух генераторов никогда не бывают абсолютно одинаковыми. Поэтому при высоких скоростях обмена данными и длинных последовательностях единиц или нулей некоторое рассогласование тактовых частот может привести к ошибке в целый такт и, следовательно, некорректному считыванию бита

Метод биполярного кодирования с альтернативной инверсией. Модификацией метода NRZ является метод биполярного кодирования с альтернативной инверсией (***Bipolar Alternate Mark Inversion, AMI***). В этом методе (рис. 1, *в*) используются три уровня потенциала — отрицательный, нулевой и положительный. Логический нуль кодируется нулевым потенциалом, а логическая единица — положительным или отрицательным уровнем, при этом потенциал каждой последующей единицы противоположен потенциалу предыдущей.

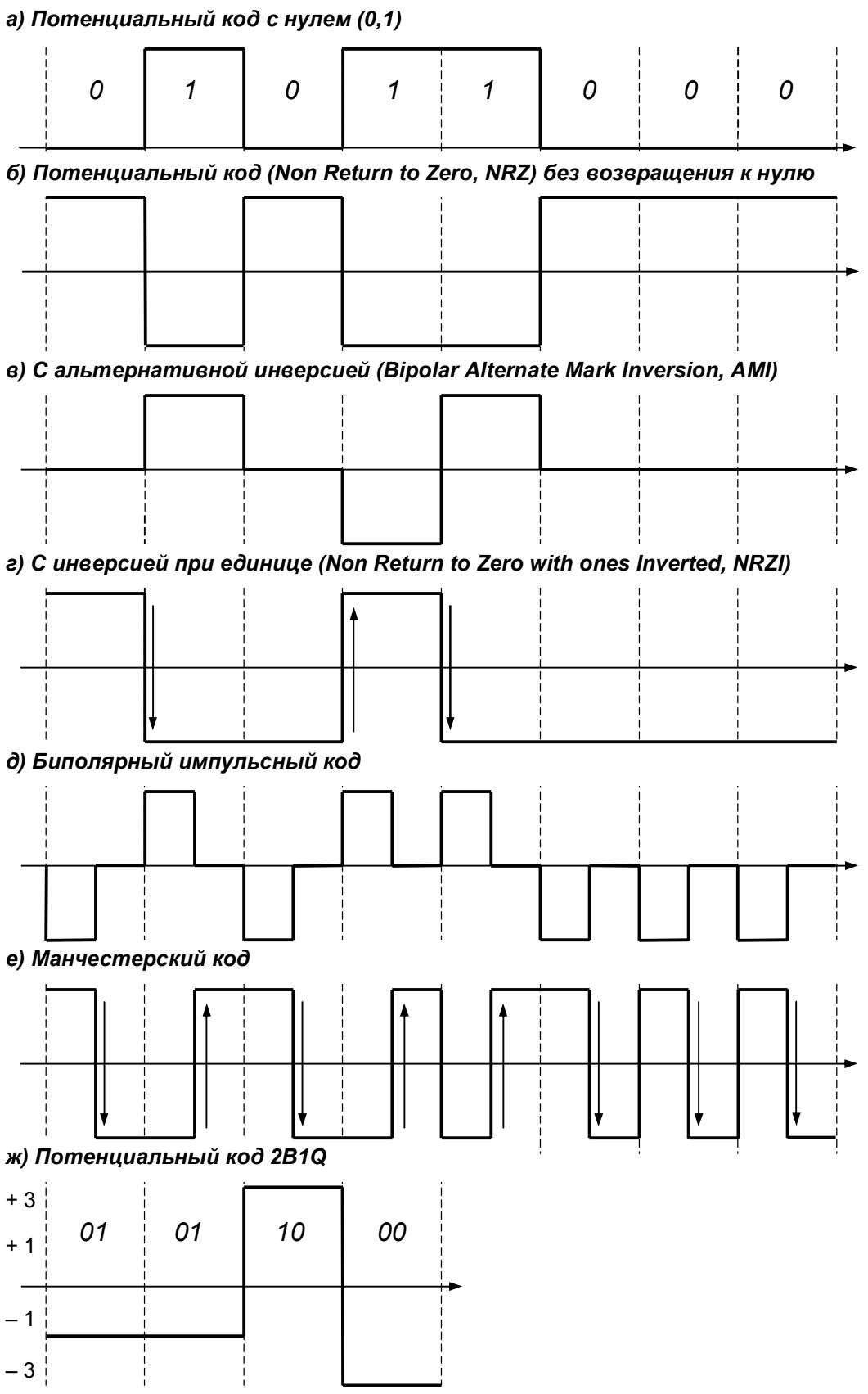


Рис. 1 — Методы кодирования физического уровня

Код AMI ликвидирует проблемы постоянной составляющей и отсутствия самосинхронизации при передаче длинных последовательностей единиц, но не нулей. При передаче единиц сигнал на линии — это последовательность разнополярных импульсов (с тем же спектром, что и у кода NRZ, то есть без постоянной составляющей и с основной гармоникой $N/2$ Гц, где N — битовая скорость передачи данных). В случае длинной последовательности нулей сигнал вырождается в постоянный потенциал нулевой амплитуды.

Потенциальный код с инверсией при единице. Потенциальный код с инверсией при единице (***Non Return to Zero with ones Inverted, NRZI***) похож на AMI, но имеет только два уровня сигнала. При передаче нуля он передает тот же потенциал, который был установлен в предыдущем такте, то есть не меняет его, а при передаче единицы потенциал инвертируется на противоположный (рис. 1, *г*). Этот код удобен в тех случаях, когда использование третьего уровня сигнала весьма нежелательно, например, в оптических кабелях.

Биполярный импульсный код. Кроме потенциальных кодов в сетях используются импульсные коды, в которых данные представлены полным импульсом или же его частью — фронтом. Примером такого подхода является биполярный импульсный код, в котором единица представлена импульсом одной полярности, а ноль — другой (рис. 1, *д*), а каждый импульс длится половину такта. У этого кода отличные самосинхронизирующими свойства, но при передаче длинной последовательности единиц или нулей может присутствовать постоянная составляющая. Из-за слишком широкого спектра биполярный импульсный код используется редко.

Манчестерский код. В локальных сетях до недавнего времени самым распространенным методом кодирования был манчестерский код (рис. 1, *е*), применяемый в технологиях *Ethernet* и *Token Ring*.

В манчестерском коде для кодирования единиц и нулей используется изменение потенциала, то есть *фронт* импульса. При манчестерском кодировании каждый такт делится на две части. Информация кодируется перепадами потенциала, происходящими в середине каждого такта. Единица кодируется перепадом от

низкого уровня сигнала к высокому, а ноль — обратным перепадом. В начале каждого такта может происходить служебный перепад сигнала, если нужно представить несколько единиц или нулей подряд. Так как сигнал изменяется по крайней мере один раз за такт передачи одного бита данных, то манчестерский код обладает хорошими самосинхронизирующими свойствами. Полоса пропускания манчестерского кода уже, чем у биполярного импульсного. У него также нет постоянной составляющей.

Потенциальный код 2B1Q. На (рис. 1, *ж*) показан потенциальный код с четырьмя уровнями сигнала для кодирования данных. Это код ***2B1Q***, название которого отражает его суть — каждые два бита (*2B*) передаются за один такт сигналом, имеющим четыре состояния (*1Q*). Паре бит 00 соответствует потенциал –2.5 В, паре бит 01 соответствует потенциал –0,833 В, паре 11 — потенциал +0,833 В, а паре 10 — потенциал +2,5 В. Этот способ кодирования не защищает от длинных последовательностей одинаковых пар бит, так как при этом сигнал превращается в постоянную составляющую. При случайном чередовании бит спектр сигнала в два раза уже, чем у кода NRZ, так как при той же битовой скорости длительность такта увеличивается в два раза. Таким образом, с помощью кода 2B1Q можно по одной и той же линии передавать данные в два раза быстрее, чем с помощью кода AMI или NRZI. Однако для его реализации мощность передатчика должна быть выше, чтобы четыре уровня четко различались приемником на фоне помех.

Спектр сигнала. Методика построения спектра базируется на разложении исходного сигнала в функциональный ряд, например, в ряд Фурье. Коэффициенты a_j, b_j ряда однозначно определяют гармонические составляющие исследуемого сигнала и могут быть вычислены по формулам:

$$a_j = \frac{2}{T}\int_0^T F(x)\cdot\cos\frac{2\cdot\pi\cdot j\cdot x}{T}dx,\ b_j = \frac{2}{T}\int_0^T F(x)\cdot\sin\frac{2\cdot\pi\cdot j\cdot x}{T}dx,$$

где T — период исходного сигнала $F(x)$. Коэффициенты a_j и b_j определяют амплитуды синусоид (косинусоид) — гармонических составляющих сигнала (*гармоник*), а параметр ***j*** (номер гармоники) задает частоту $f = \frac{2\cdot\pi\cdot j}{T}$.

Теория рядов говорит о том, что любая периодическая, ограниченная непрерывная функция (в нашем случае — $F(x)$) может быть точно представлена рядом Фурье:

$$S(x) = \frac{a_0}{2} + \sum_{j=1}^{n}\left(a_j \cdot \cos\frac{2 \cdot \pi \cdot j \cdot x}{T} + b_j \cdot \sin\frac{2 \cdot \pi \cdot j \cdot x}{T} \right),$$

где $n \to \infty$. В случае непериодической функции, кроме того $T \to \infty$.

Просуммировав ряд Фурье для какого-то конечного n получим некое приближение к исходному сигналу, часть гармоник окажутся отброшенными (не попавшими в АЧХ канала передачи сигнала) — построим сигнал, поступающий на приемник. Говорят, что сигнал восстановлен из набора гармоник a_j, b_j — коэффициентов ряда Фурье.

Данная функция $S(x)$ есть, по сути, арифметическая сумма синусоид и косинусоид определенной частоты f и амплитуды a_j или b_j соответственно. Так, например, на рис. 2 приведен потенциальный код, его спектральное разложение и результирующий сигнал.

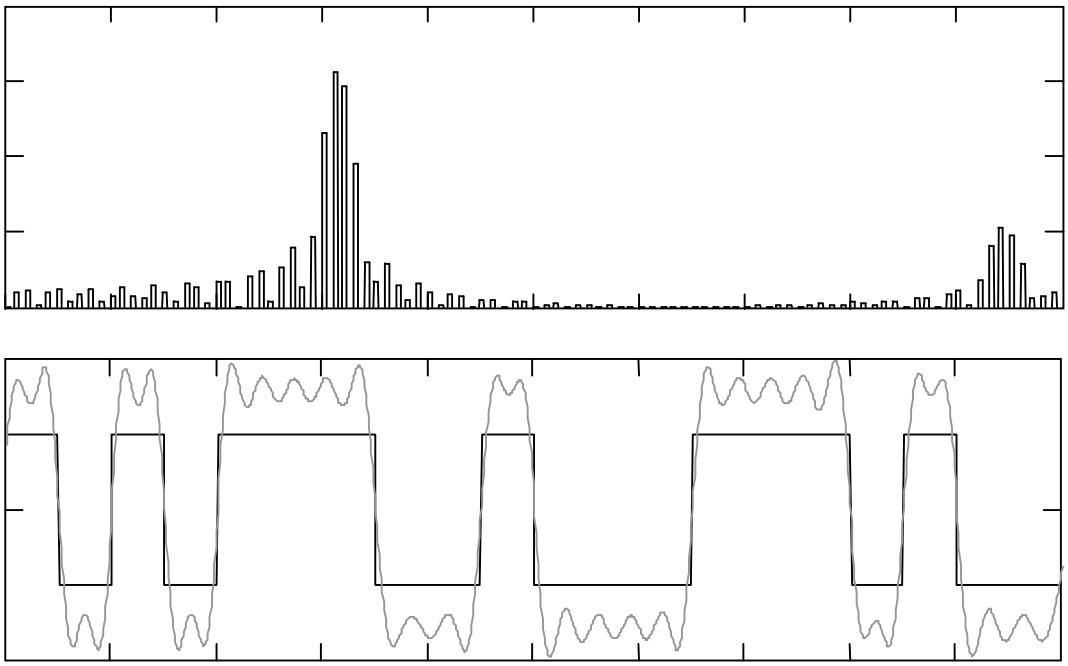


Рис. 2 — Спектр, исходный и результирующий сигналы

Если дискретные данные передаются с битовой скоростью N бит/с, то спектр состоит из постоянной составляющей нулевой частоты и бесконечного ряда гармоник с частотами f, $3 \cdot f$, $5 \cdot f$, $7 \cdot f$,..., где $f = N/2$. Амплитуды этих гармоник убывают достаточно медленно — с коэффициентами 1/3, 1/5, 1/7,... от амплитуды гармоники f. В результате спектр потенциального кода требует для качественной передачи широкую полосу пропускания.

Правило выбора предельного шага при равномерной дискретизации с использованием модели сигнала с ограниченным спектром сформулировано академиком ***В. А. Котельниковым***:

Любая непрерывная функция $F(t)$, спектр которой ограничен частотой $f_{\max}$ полностью определяется последовательностью своих значений F_k в моменты времени, отстоящие друг от друга на интервал

$$\Delta t = \frac{1}{2 \cdot f_{\max}} = \frac{\pi}{\omega_{\max}}.$$

Иными словами, теорема Котельникова определяет, что для адекватного представления непрерывной по времени функции $F(t)$ при помощи дискретных отсчетов F_k частота дискретизации должна быть, по крайней мере, в два раза больше, чем самая высокочастотная гармоническая составляющая непрерывного сигнала $f_{\max}$.

Пропускная способность канала связи зависит не только от его характеристик, таких как амплитудно-частотная характеристика, но и от спектра передаваемых сигналов. Если *значимые* гармоники сигнала (то есть те гармоники, амплитуды которых вносят основной вклад в результирующий сигнал) попадают в полосу пропускания линии, то такой сигнал будет хорошо передаваться данной линией связи и приемник сможет правильно распознать информацию, отправленную по линии передатчиком (рис. 3, *а*). Если же значимые гармоники выходят за границы полосы пропускания линии связи, то сигнал будет значительно искажаться, приемник будет ошибаться при распознавании информации, а значит, информация не сможет передаваться с заданной пропускной способностью (рис. 3, *б*).

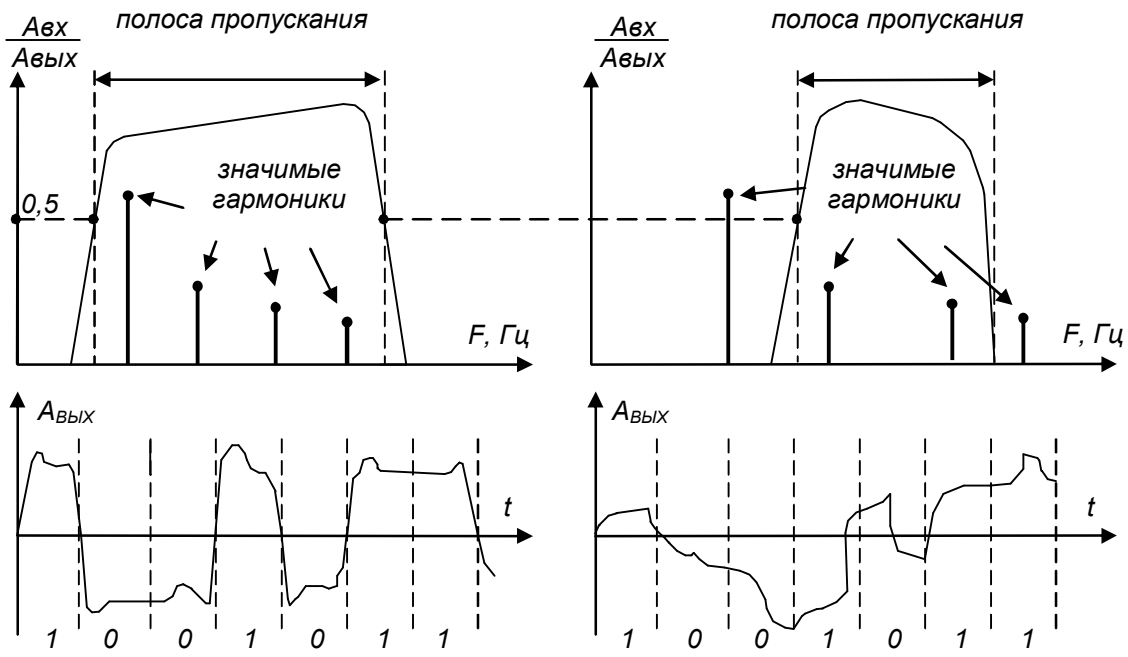


Рис. 3 — Полоса пропускания и спектр сигнала

При построении результирующего сигнала, приведенного на рис. 4, использовалась только та часть спектра, которая выделена серым цветом. Очевидно, что отброшенные гармоники (на рис. 4 выделены черным) не оказывают большого влияния, так как восстановленный сигнал может быть распознан с вероятностью близкой к единице.

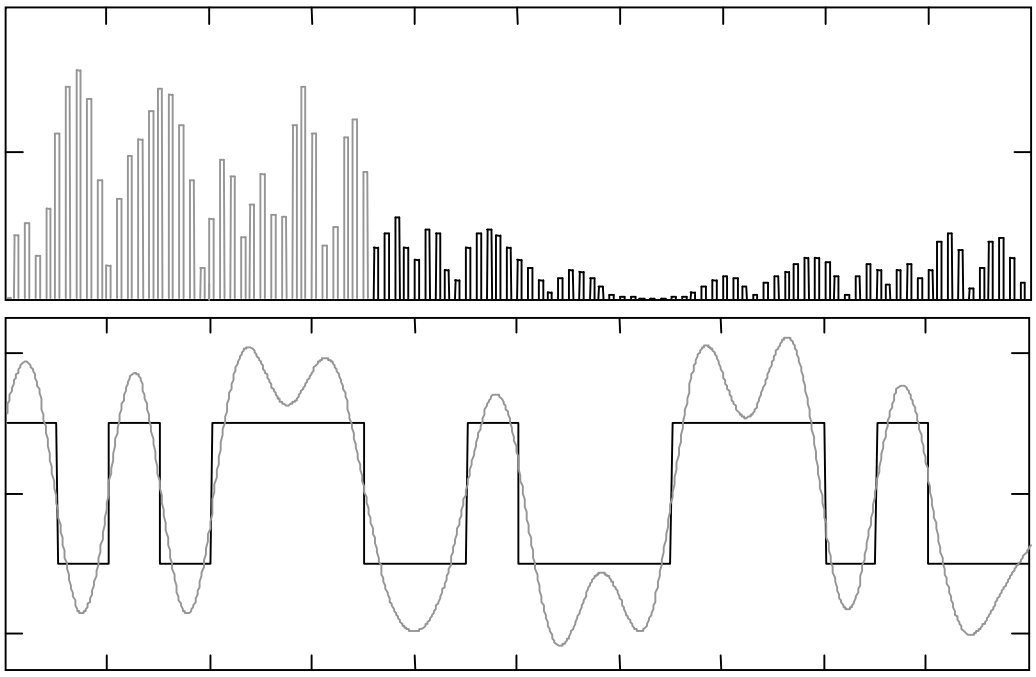


Рис. 4 — Исходный и результирующий сигнал с отброшенными незначащими гармониками

Если же полоса пропускания канала передачи данных сдвинута в сторону, например, так, как показано на рис. 5 (серый цвет), то восстановленный сигнал, очевидно, корректно распознать не удастся.

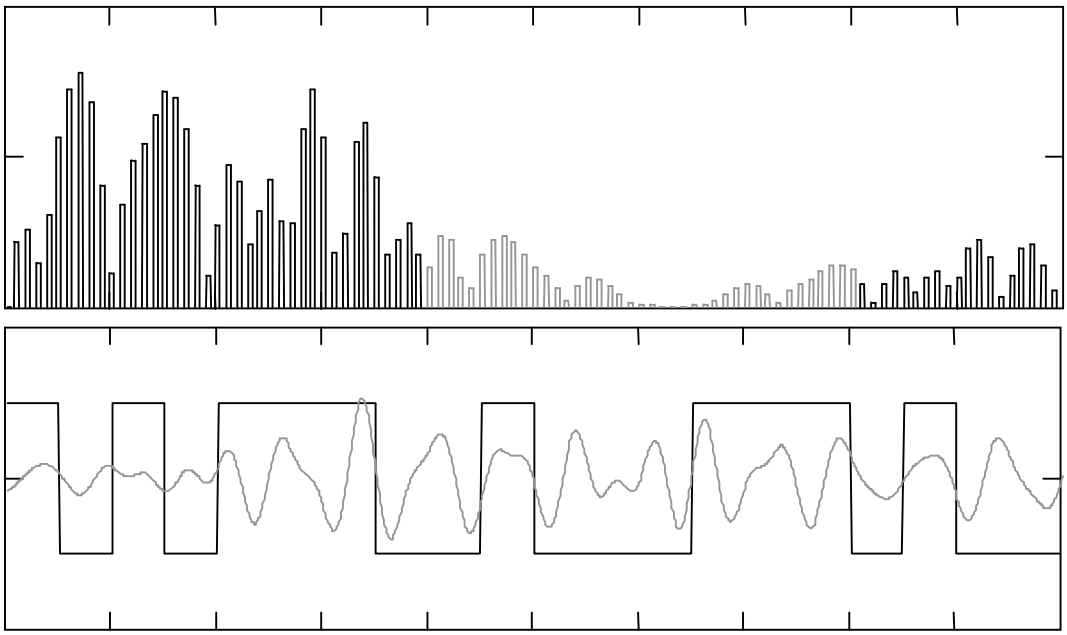


Рис. 5 — Исходный и результирующий сигнал со сдвинутой полосой пропускания канала

ЛАБОРАТОРНАЯ РАБОТА № 1
Полоса пропускания, амплитудно-частотная характеристика и методы кодирования физического уровня

Продолжительность — 4 часа.
Максимальный рейтинг — 8 баллов.

ЦЕЛЬ РАБОТЫ

Познакомиться с методами кодирования физического уровня OSI. Установить связь между амплитудно-частотной характеристикой (АЧХ) и полосой пропускания линии. Повторить методики построения спектра сигнала. Уяснить связь между полосой пропускания линии и спектральными характеристиками сигнала.

ЗАДАНИЕ

1. Построить (на графике) функцию моделирующую цифровой сигнал, передаваемый по линии связи в соответствии со своим индивидуальным заданием.
2. Познакомиться с методами кодирования физического уровня OSI. Метод кодирования сигнала выбирается так же из индивидуального задания.
3. Для заданной функции и выбранного метода кодирования построить спектр сигнала и вывести его на график. Количество разбиений передаваемого сигнала задать в соответствии с теоремой Котельникова.
4. Просуммировав гармоники ряда Фурье построить результирующий (полученный, восстановленный) сигнал на графике.
5. Определить требуемую полосу пропускания, необходимую для передачи исходного сигнала. Соотнести ее с заданной в индивидуальном задании полосой пропускания линии связи.
6. Построить (на графике) переданный по линии связи сигнал, т.е. сигнал, восстановленный из спектра с учетом частотных характеристик линии связи.
7. Провести численный эксперимент по определению полосы пропускания для данного сигнала, варьируя числом гармоник

(коэффициентов ряда Фурье). В отчете привести частотные характеристики полосы пропускания.

ВАРИАНТЫ ИНДИВИДУАЛЬНЫХ ЗАДАНИЙ

1. Закодировать потенциальным кодом с нулем (0,1) и передать по линии связи сигнал 12AF56F4h со скоростью 10 Кбит/с. Полоса пропускания линии: 1—9 гармонические составляющие сигнала.

2. Закодировать потенциальным кодом (Non Return to Zero, NRZ) без возвращения к нулю (–1,1) и передать по линии связи сигнал BD8B7D9Ah со скоростью 32 бит/с. Полоса пропускания линии: 1—16 гармонические составляющие сигнала.

3. Закодировать потенциальным кодом с альтернативной инверсией (Bipolar Alternate Mark Inversion, AMI) и передать по линии связи сигнал 1EDC4562h со скоростью 100 бит/с. Полоса пропускания линии: 1—7 гармонические составляющие сигнала.

4. Закодировать потенциальным кодом с инверсией при единице (Non Return to Zero with ones Inverted, NRZI) и передать по линии связи сигнал 11341DA1h со скоростью 4 Кбит/с. Полоса пропускания линии: 2—30 гармонические составляющие сигнала.

5. Закодировать биполярным импульсным кодом и передать по линии связи сигнал 2452FC2Fh со скоростью 4,800 Кбит/с. Полоса пропускания линии: 2—9 гармонические составляющие сигнала.

6. Закодировать манчестерским кодом и передать по линии связи сигнал 12346AB3h со скоростью 10 Кбит/с. Полоса пропускания линии: 0—5 гармонические составляющие сигнала.

7. Закодировать потенциальным кодом 2B1Q и передать по линии связи сигнал CD6682EFh со скоростью 2,400 Кбит/с. Полоса пропускания линии: 1—9 гармонические составляющие сигнала.

8. Закодировать потенциальным кодом с нулем (0,1) и передать по линии связи сигнал G56G7G0Gh со скоростью 56.5 бит/с. Полоса пропускания линии: 1—32 гармонические составляющие сигнала.

9. Закодировать потенциальным кодом (Non Return to Zero, NRZ) без возвращения к нулю (–1,1) и передать по линии связи

сигнал B11B6B9Ah со скоростью 9600 бит/с. Полоса пропускания линии: 0—16 гармонические составляющие сигнала.

10. Закодировать потенциальным кодом с альтернативной инверсией (Bipolar Alternate Mark Inversion, AMI) и передать по линии связи сигнал E069052Eh со скоростью 100 бит/с. Полоса пропускания линии: 1—8 гармонические составляющие сигнала.

11. Закодировать потенциальным кодом с инверсией при единице (Non Return to Zero with ones Inverted, NRZI) и передать по линии связи сигнал F361F1d4h со скоростью 4 Кбит/с. Полоса пропускания линии: 2—10 гармонические составляющие сигнала.

12. Закодировать биполярным импульсным кодом и передать по линии связи сигнал F0F0h со скоростью 90 бит/с. Полоса пропускания линии: 0—5 гармонические составляющие сигнала.

13. Закодировать манчестерским кодом и передать по линии связи сигнал 4325C311h со скоростью 10 Кбит/с. Полоса пропускания линии: 1—15 гармонические составляющие сигнала.

14. Закодировать потенциальным кодом 2B1Q и передать по линии связи сигнал 567439A8h со скоростью 14,400 бит/с. Полоса пропускания линии: 1—8 гармонические составляющие сигнала.

15. Закодировать потенциальным кодом с нулем (0,1) и передать по линии связи сигнал 90AB1261h со скоростью 128 бит/с. Полоса пропускания линии: 0—7 гармонические составляющие сигнала.

16. Закодировать потенциальным кодом (Non Return to Zero, NRZ) без возвращения к нулю (–1,1) и передать по линии связи сигнал C98CC4ACh со скоростью 3,2 Кбит/с. Полоса пропускания линии: 0—50 гармонические составляющие сигнала.

2 МЕТОДЫ КОДИРОВАНИЯ ФИЗИЧЕСКОГО УРОВНЯ. АНАЛОГОВАЯ МОДУЛЯЦИЯ

Аналоговая модуляция является таким способом физического кодирования, при котором информация кодируется изменением амплитуды, частоты или фазы синусоидального сигнала несущей частоты. Основные способы аналоговой модуляции показаны на рис. 6. На диаграмме (рис. 6,*а*) показана последовательность бит исходной информации, представленная *потенциальным кодом*.

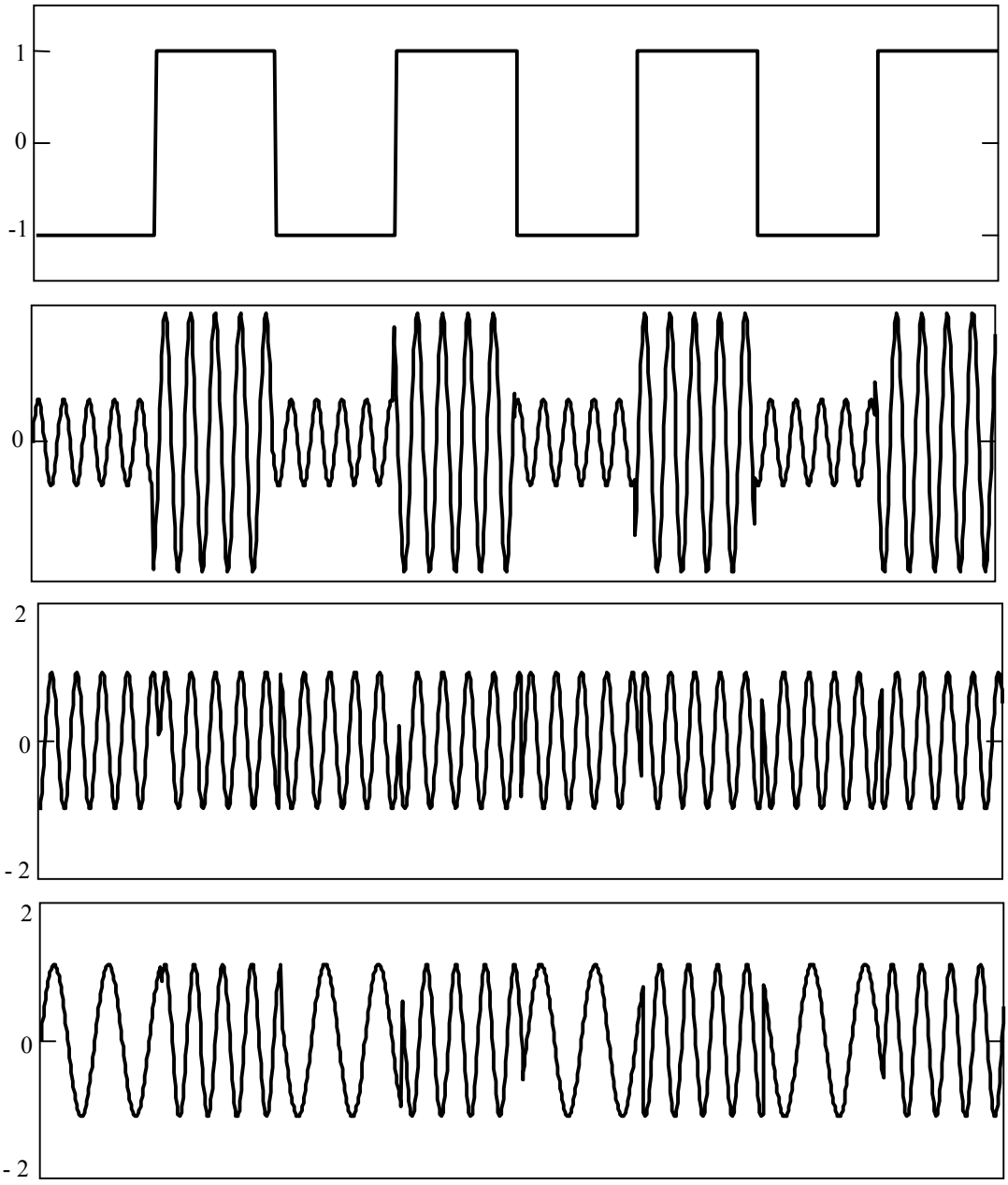


Рис. 6 — Различные виды аналоговой модуляции

Амплитудная модуляция. При амплитудной модуляции на выход модулятора поступает сигнал $V(t) = V_m \cdot \sin(\omega_V \cdot t - \varphi_V)$ и несущая $U(t) = U_m \cdot \sin(\omega_U \cdot t - \varphi_U)$. На выходе нелинейного элемента модулируются колебания:

$$U_{AM}(t) = U_m \cdot (1 + m \cdot \sin(\omega_V \cdot t - \varphi_V)) \cdot \sin(\omega_U \cdot t - \varphi_U),$$

где $m = V_m / U_m$ — коэффициент модуляции. На выходе модулятора в спектре сигнала присутствует несущая частота ω_U и две боковые частоты $\omega_U - \omega_V$ и $\omega_U + \omega_V$.

При амплитудной модуляции во избежание искажений, называемых качанием фронта, необходимо выполнение условия $\omega_U >> \omega_V$. Соблюдение этого условия при стандартной (для среднескоростной аппаратуры передачи данных) несущей частоте 1700 Гц не может обеспечить информационные скорости выше 300 бит/с. Поэтому реально в модемах применяют дополнительное преобразование частоты: сначала производят модуляцию несущей, имеющей повышенную частоту, например $\omega_{НД}$= 10 кГц, затем с помощью фильтра выделяют спектр модулированного сигнала и с помощью преобразователя частоты переносят модулирующие колебания на промежуточную частоту, например ω_U= 1700 Гц. Тогда при боковых полосах до 1400 Гц спектр сигнала согласуется с полосой пропускания телефонных линий.

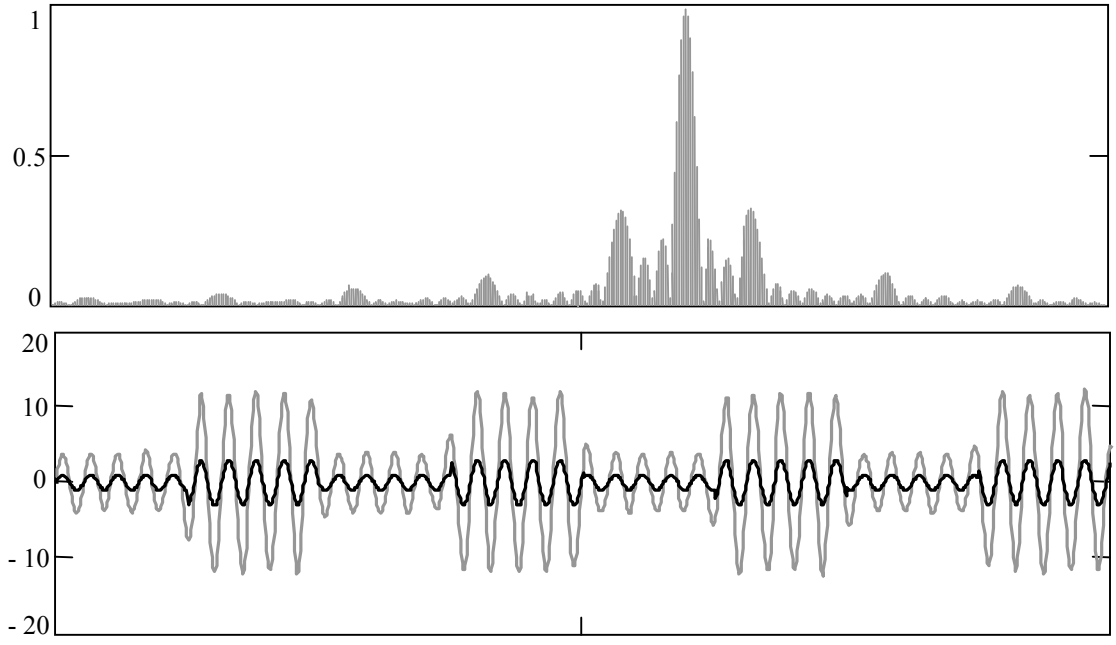


Рис. 7 — Спектр и восстановленный сигнал амплитудной модуляции

В сравнительно простых модемах применяют ***частотную модуляцию*** (***FSK — Frequency Shift Keying***) (рис. 8) со скоростями передачи до 1200 бит/с. Так, если необходима дуплексная связь по двухпроводной линии, то возможно представление логических нуля и единицы в вызывном модеме частотами 980 Гц и 1180 Гц соответственно. При этом скорость передачи составляет 300 бод.

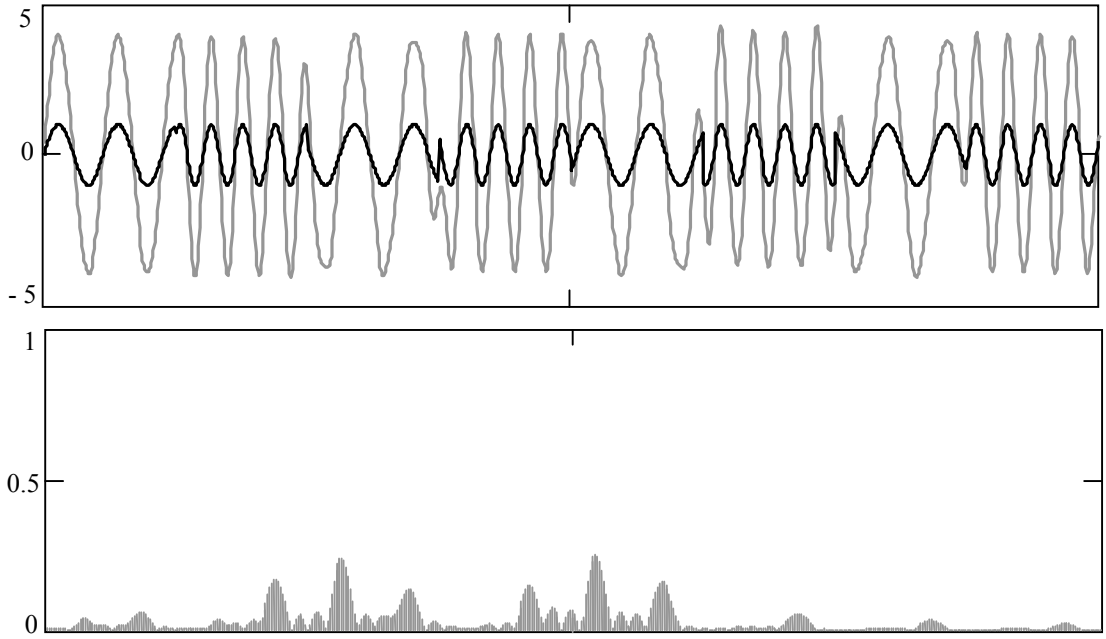


Рис. 8 — Частотная модуляция

Спектр сигнала, модулирующего потенциальный код частотным способом аналоговой модуляции приведен на рис. 8, *б*, а сама форма сигнала при частотной модуляции изображена на рис. 8, *а*.

Обычно для передачи сигнала об ошибке от приемника к передатчику нужен канал обратной связи. При этом требования к скорости передачи данных по обратному каналу могут быть невысокими. Тогда в полосе частот телефонного канала образуют обратный канал с частотной модуляцией, по которому со скоростью 75 бит/с передают единицу частотой 390 Гц и ноль — частотой 450 Гц.

Фазовая модуляция (***PSK — Phase Shift Keying***) двух уровней сигнала (нуля и единицы) осуществляется переключением

между двумя несущими, сдвинутыми на полпериода друг относительно друга (рис. 9,*а*). Другой вариант фазовой модуляции — изменение фазы на $\pi/2$ в каждом такте при передаче нуля и на $3\pi/4$, если передается единица.

Исходный потенциальный код (рис. 9, *а*) после преобразования фазовой модуляции имеет спектр, приведенный на рис. 9, *б*.

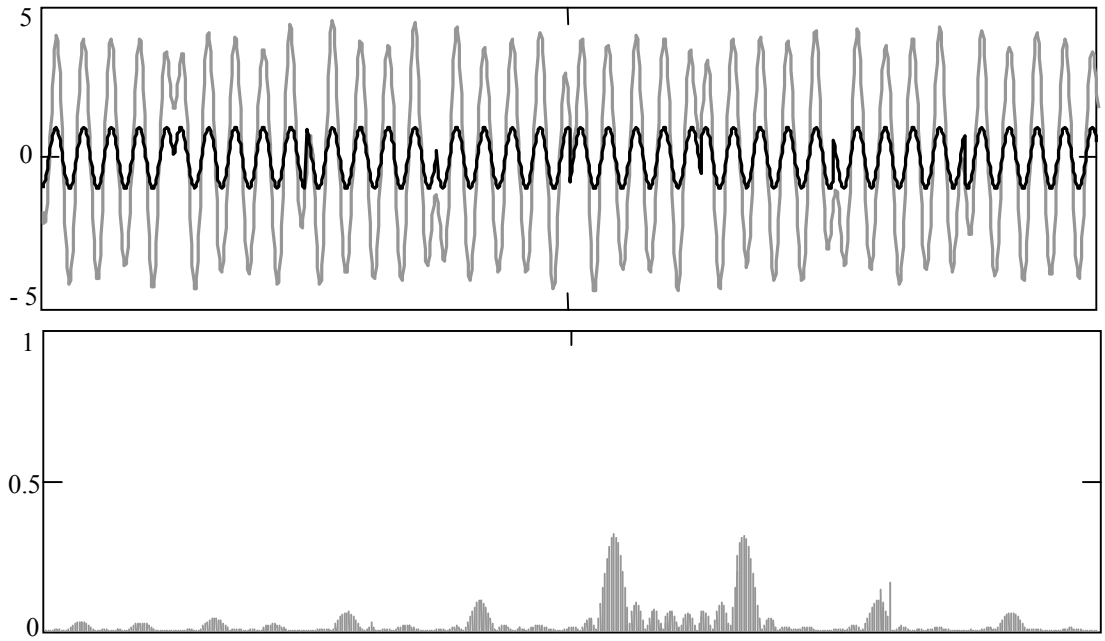


Рис. 9 — Фазовая модуляция

Комбинированные аналоговые методы. Для повышения скорости передачи данных используют комбинированные методы модуляции, сочетающие те или иные методы аналоговой модуляции с методами цифрового кодирования. Наиболее распространенными являются методы ***квадратурной амплитудной модуляции*** (***Quadrature Amplitude Modulation, QAM***). Квадратурно-амплитудная модуляция, которую так же называют ***квадратурно-импульсной*** модуляцией) основанана передаче одним элементом модулированного сигнала n бит информации, где $n = 4…8$ (т.е. используется 16—256 дискретных значений амплитуды). Однако для надежного различения этих значений амплитуды требуется малый уровень помех (отношение сигнал/помеха не мене 12 дБ при $n = 4$).

При меньших отношениях сигнал/помеха лучше применять фазовую модуляцию с четырьмя или восемью дискретными значениями фазы для предоставления соответственно 2 или 3 бит информации. Тогда при скорости модуляции в 1200 бод (т.е. 1200 элементов аналогового сигнала в секунду, где элемент — часть сигнала между возможными сменами фаз) и четырехфазной модуляции скорость передачи данных равна 2400 бит/с. Используются также скорости передачи 4800 бит/с (при скорости модуляции 1600 бод и восьмифазной модуляции), 9600 бит/с и более при комбинации фазовой и амплитудной модуляций.

ЛАБОРАТОРНАЯ РАБОТА № 2 Полоса пропускания, амплитудно-частотная характеристика и методы аналоговой модуляции

Продолжительность — 4 часа.
Максимальный рейтинг — 8 баллов.

ЦЕЛЬ РАБОТЫ

Познакомиться с методами аналоговой модуляции. Установить связь между амплитудно-частотной характеристикой (АЧХ) и полосой пропускания линии. Повторить методики построения спектра сигнала. Уяснить связь между полосой пропускания линии и спектральными характеристиками сигнала.

ЗАДАНИЕ

1. Построить (на графике) функцию моделирующую цифровой сигнал, передаваемый полинии связи в соответствии со своим индивидуальным заданием.

2. Познакомиться с методами аналоговой модуляции. Метод аналоговой модуляции сигнала выбирается так же из индивидуального задания.

3. Построить (на графике) функцию, получаемую в результате применения модуляционного метода к исходному сигналу.

4. Для заданной функции и выбранного метода модуляции построить спектр сигнала и вывести его на график. Количество разбиений передаваемого сигнала задать в соответствии с теоремой Котельникова.

5. Просуммировав гармоники ряда Фурье построить результирующий (полученный, восстановленный) сигнал на графике.

6. Определить требуемую полосу пропускания, необходимую для передачи исходного сигнала. Соотнести ее с заданной в индивидуальном задании полосой пропускания линии связи.

7. Построить (на графике) переданный по линии связи сигнал, т.е. сигнал восстановленный из спектра с учетом частотных характеристик линии связи.

8. Подобрать параметры несущей так, чтобы результирующий сигнал передавался по линии связи без значительных искажений.

9. Определить максимальную пропускную способность канала.

ВАРИАНТЫ ИНДИВИДУАЛЬНЫХ ЗАДАНИЙ

1. Для передачи по линии связи сигнала 12345632h использовать амплитудную модуляцию с двумя уровнями сигнала. Полоса пропускания линии: 300—3400 Гц.

2. Для передачи по линии связи сигнала 65432145h использовать амплитудную модуляцию с четырьмя уровнями сигнала. Полоса пропускания линии: 3.5—7 кГц.

3. Для передачи по линии связи сигнала 34567448h использовать фазовую модуляцию с двумя значениями фазы. Полоса пропускания линии: 7—8 кГц.

4. Для передачи по линии связи сигнала 87654323h использовать фазовую модуляцию с четырьмя значениями фазы. Полоса пропускания линии: 100—400 Гц.

5. Для передачи по линии связи сигнала 56789010h использовать частотную модуляцию с четырьмя значениями частоты. Полоса пропускания линии: 10—103 Гц.

6. Для передачи по линии связи сигнала 98765444h использовать частотную модуляцию с двумя значениями частоты. Полоса пропускания линии: 400—430 Гц.

7. Для передачи по линии связи сигнала 90ABC34Dh использовать амплитудно-частотную модуляцию с двумя уровнями амплитуды и двумя значениями частоты. Полоса пропускания линии: 300—3400 Гц.

8. Для передачи по линии связи сигнала AB23CDEFh использовать амплитудно-фазовую модуляцию с двумя уровнями амплитуды и двумя значениями фазы. Полоса пропускания линии: 3500—7000 Гц.

9. Для передачи по линии связи сигнала FEDCB55Ah использовать фазово-частотную модуляцию с двумя уровнями частоты и двумя значениями фазы. Полоса пропускания линии: 50—101 Гц.

10. Для передачи по линии связи сигнала AB98C012h использовать квадратурно-импульсную модуляцию с двумя уровнями амплитуды и четырьмя значениями фазы. Полоса пропускания линии: 10—102 Гц.

11. Для передачи по линии связи сигнала 210DE44Fh использовать квадратурно-импульсную модуляцию с четырьмя уровнями амплитуды и двумя значениями фазы. Полоса пропускания линии: 10—11 кГц.

12. Для передачи по линии связи сигнала 12356123h использовать амплитудную модуляцию с двумя уровнями сигнала. Полоса пропускания линии: 1300—4000 Гц.

13. Для передачи по линии связи сигнала 654DA654h использовать амплитудную модуляцию с четырьмя уровнями сигнала. Полоса пропускания линии: 3—6 кГц.

14. Для передачи по линии связи сигнала 345AB25Ch использовать фазовую модуляцию с двумя значениями фазы. Полоса пропускания линии: 7—10.1 кГц.

15. Для передачи по линии связи сигнала 87A82A43h использовать фазовую модуляцию с четырьмя значениями фазы. Полоса пропускания линии: 100.5—103.6 Гц.

16. Для передачи по линии связи сигнала AA35AA78h использовать частотную модуляцию с четырьмя значениями частоты. Полоса пропускания линии: 400—401 Гц.

17. Для передачи по линии связи сигнала B245368Bh использовать частотную модуляцию с двумя значениями частоты. Полоса пропускания линии: 430—434 Гц.

18. Для передачи по линии связи сигнала AA3ABB7Bh использовать амплитудно-частотную модуляцию с двумя уровнями амплитуды и двумя значениями частоты. Полоса пропускания линии: 330—660 Гц.

19. Для передачи по линии связи сигнала D5DD1911h использовать амплитудно-фазовую модуляцию с двумя уровнями амплитуды и двумя значениями фазы. Полоса пропускания линии: 70—700 Гц.

20. Для передачи по линии связи сигнала A8AB0BCCh использовать фазово-частотную модуляцию с двумя уровнями частоты и двумя значениями фазы. Полоса пропускания линии: 101—101.3 Гц.

3 ОБЩИЕ СВЕДЕНИЯ ОБ ОСНОВНЫХ ПРОТОКОЛАХ СТЕКА TCP/IP

Протокол межсетевого взаимодействия IP

Основу транспортных средств стека протоколов TCP/IP составляет протокол межсетевого взаимодействия — ***Internet Protocol*** (***IP***). К основным функциям протокола IP относятся:

– перенос между сетями различных типов адресной информации в унифицированной форме,

– сборка и разборка пакетов при передаче их между сетями с различным максимальным значением длины пакета.

Формат пакета IP

Пакет IP состоит из заголовка и поля данных. Заголовок пакета имеет следующие поля:

- ***Поле Номер версии*** (***VERS***) указывает версию протокола IP. Сейчас повсеместно используется версия 4 и готовится переход на версию 6, называемую также IPng (IP next generation).
- ***Поле Длина заголовка*** (***HLEN***) пакета IP занимает 4 бита и указывает значение длины заголовка, измеренное в 32-битовых словах. Обычно заголовок имеет длину в 20 байт (пять 32-битовых слов), но при увеличении объема служебной информации эта длина может быть увеличена за счет использования дополнительных байт в поле Резерв (IP OPTIONS).
- ***Поле Тип сервиса*** (***SERVICE TYPE***) занимает 1 байт и задает приоритетность пакета и вид критерия выбора маршрута. Первые три бита этого поля образуют подполе приоритета пакета (PRECEDENCE). Приоритет может иметь значения от 0 (нормальный пакет) до 7 (пакет управляющей информации). Маршрутизаторы и компьютеры могут принимать во внимание приоритет пакета и обрабатывать более важные пакеты в первую очередь. Поле Тип сервиса содержит также три бита, определяющие критерий выбора маршрута. Установленный бит D (delay) говорит о том, что маршрут должен выбираться для минимизации задержки доставки данного пакета, бит T — для максимизации пропускной способности, а бит R — для максимизации надежности доставки.

- ***Поле Общая длина*** (***TOTAL LENGTH***) занимает 2 байта и указывает общую длину пакета с учетом заголовка и поля данных.
- ***Поле Идентификатор пакета*** (***IDENTIFICATION***) занимает 2 байта и используется для распознавания пакетов, образовавшихся путем фрагментации исходного пакета. Все фрагменты должны иметь одинаковое значение этого поля.
- ***Поле Флаги*** (***FLAGS***) занимает 3 бита, оно указывает на возможность фрагментации пакета (установленный бит Do not Fragment — DF — запрещает маршрутизатору фрагментировать данный пакет), а также на то, является ли данный пакет промежуточным или последним фрагментом исходного пакета (установленный бит More Fragments — MF — говорит о том пакет переносит промежуточный фрагмент).
- ***Поле Смещение фрагмента*** (***FRAGMENT OFFSET***) занимает 13 бит, оно используется для указания в байтах смещения поля данных этого пакета от начала общего поля данных исходного пакета, подвергнутого фрагментации. Используется при сборке/разборке фрагментов пакетов при передачах их между сетями с различными величинами максимальной длины пакета.
- ***Поле Время жизни*** (***TIME TO LIVE***) занимает 1 байт и указывает предельный срок, в течение которого пакет может перемещаться по сети. Время жизни данного пакета измеряется в секундах и задается источником передачи средствами протокола IP. На шлюзах и в других узлах сети по истечении каждой секунды из текущего времени жизни вычитается единица; единица вычитается также при каждой транзитной передаче (даже если не прошла секунда). При истечении времени жизни пакет аннулируется.
- ***Идентификатор Протокола верхнего уровня*** (***PROTOCOL***) занимает 1 байт и указывает, какому протоколу верхнего уровня принадлежит пакет (например, это могут быть протоколы TCP, UDP или RIP).
- ***Контрольная сумма*** (***HEADER CHECKSUM***) занимает 2 байта, она рассчитывается по всему заголовку.
- ***Поля Адрес источника*** (***SOURCE IP ADDRESS***) и ***Адрес назначения*** (***DESTINATION IP ADDRESS***) имеют одинаковую длину — 32 бита, и одинаковую структуру.

- ***Поле Резерв*** (***IP OPTIONS***) является необязательным и используется обычно только при отладке сети. Это поле состоит из нескольких подполей, каждое из которых может быть одного из восьми предопределенных типов. В этих подполях можно указывать точный маршрут прохождения маршрутизаторов, регистрировать проходимые пакетом маршрутизаторы, помещать данные системы безопасности, а также временные отметки. Так как число подполей может быть произвольным, то в конце поля Резерв должно быть добавлено несколько байт для выравнивания заголовка пакета по 32-битной границе.

Максимальная длина поля данных пакета ограничена разрядностью поля, определяющего эту величину, и составляет 65535 байтов, однако при передаче по сетям различного типа длина пакета выбирается с учетом максимальной длины пакета протокола нижнего уровня, несущего IP-пакеты. Если это кадры Ethernet, то выбираются пакеты с максимальной длиной в 1500 байтов, умещающиеся в поле данных кадра Ethernet.

Протоколы транспортного уровня (протоколы TCP или UDP), пользующиеся сетевым уровнем для отправки пакетов, считают, что максимальный размер поля данных IP-пакета равен 65535, и поэтому могут передать ему сообщение такой длины для транспортировки через интерсеть. В функции уровня IP входит разбиение слишком длинного для конкретного типа составляющей сети сообщения на более короткие пакеты с созданием соответствующих служебных полей, нужных для последующей сборки фрагментов в исходное сообщение.

В большинстве типов локальных и глобальных сетей определяется такое понятие как максимальный размер поля данных кадра или пакета, в которые должен инкапсулировать свой пакет протокол IP. Эту величину обычно называют максимальной единицей транспортировки — Maximum Transfer Unit, MTU. Сети Ethernet имеют значение MTU, равное 1500 байт, сети FDDI — 4096 байт, а сети X.25 чаще всего работают с MTU в 128 байт.

Протокол доставки пользовательских дейтаграмм UDP

Задачей протокола транспортного уровня ***UDP*** (***User Datagram Protocol***) является передача данных между прикладны-

ми процессами без гарантий доставки, поэтому его пакеты могут быть потеряны, продублированы или прийти не в том порядке, в котором они были отправлены.

В то время, как задачей сетевого уровня является передача данных между произвольными узлами сети, задача транспортного уровня заключается в передаче данных между любыми прикладными процессами, выполняющимися на любых узлах сети. Действительно, после того, как пакет средствами протокола IP доставлен в компьютер-получатель, данные необходимо направить конкретному процессу-получателю. Каждый компьютер может выполнять несколько процессов, более того, прикладной процесс тоже может иметь несколько точек входа, выступающих в качестве адреса назначения для пакетов данных.

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.*** Таким образом, адресом назначения, который используется на транспортном уровне, является идентификатор (номер) порта прикладного сервиса. Номер порта, задаваемый транспортным уровнем, в совокупности с номером сети и номером компьютера, задаваемыми сетевым уровнем, однозначно определяют прикладной процесс в сети.

Назначение номеров портов прикладным процессам осуществляется либо централизовано, если эти процессы представляют собой популярные общедоступные сервисы, типа сервиса удаленного доступа к файлам TFTP (Trivial FTP) или сервиса удаленного управления telnet, либо локально для тех сервисов, которые еще не стали столь распространенными, чтобы за ними закреплять стандартные (зарезервированные) номера.

Централизованное присвоение сервисам номеров портов выполняется организацией ***Internet Assigned Numbers Authority***. Эти номера затем закрепляются и опубликовываются в стандартах Internet. Например, упомянутому выше сервису удаленного доступа к файлам TFTP присвоен стандартный номер порта 69.

Локальное присвоение номера порта заключается в том, что разработчик некоторого приложения просто связывает с ним любой доступный, произвольно выбранный числовой идентифика-

тор, обращая внимание на то, чтобы он не входил в число зарезервированных номеров портов. В дальнейшем все удаленные запросы к данному приложению от других приложений должны адресоваться с указанием назначенного ему номера порта.

Мультиплексирование и демультиплексирование прикладных протоколов с помощью протокола UDP

Протокол UDP ведет для каждого порта две очереди: очередь пакетов, поступающих в данный порт из сети, и очередь пакетов, отправляемых данным портом в сеть.

Процедура обслуживания протоколом UDP запросов, поступающих от нескольких различных прикладных сервисов, называется ***мультиплексированием***.

Распределение протоколом UDP поступающих от сетевого уровня пакетов между набором высокоуровневых сервисов, идентифицированных номерами портов, называется демультиплексированием (рис. 10).

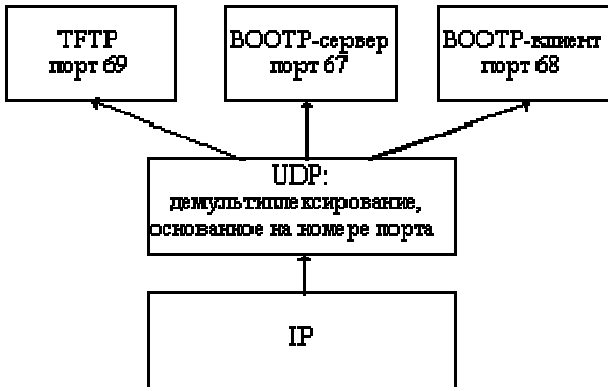


Рис. 10

Хотя к услугам протокола UDP может обратиться любое приложение, многие из них предпочитают иметь дело с другим, более сложным протоколом транспортного уровня TCP. Дело в том, что протокол UDP выступает простым посредником между сетевым уровнем и прикладными сервисами, и, в отличие от TCP, не берет на себя никаких функций по обеспечению надежности передачи. UDP является дейтаграммным протоколом, то есть он не устанавливает логического соединения, не нумерует и не упорядочивает пакеты данных.

С другой стороны, функциональная простота протокола UDP обуславливает простоту его алгоритма, компактность и высокое быстродействие. Поэтому те приложения, в которых реализован собственный, достаточно надежный, механизм обмена сообщениями, основанный на установлении соединения, предпочитают для непосредственной передачи данных по сети использовать менее надежные, но более быстрые средства транспортировки, в качестве которых по отношению к протоколу TCP и выступает протокол UDP. Протокол UDP может быть использован и в том случае, когда хорошее качество каналов связи обеспечивает достаточный уровень надежности и без применения дополнительных приемов типа установления логического соединения и квитирования передаваемых пакетов.

Формат сообщений UDP

Единица данных протокола UDP называется UDP-пакетом или пользовательской дейтаграммой (user datagram). UDP-пакет состоит из заголовка и поля данных, в котором размещается пакет прикладного уровня. Заголовок имеет простой формат и состоит из четырех двухбайтовых полей:

- ***UDP source port*** — номер порта процесса-отправителя,
- ***UDP destination port*** — номер порта процесса-получателя,
- ***UDP message length*** — длина UDP-пакета в байтах,
- ***UDP checksum*** — контрольная сумма UDP-пакета.

Не все поля UDP-пакета обязательно должны быть заполнены. Если посылаемая дейтаграмма не предполагает ответа, то на месте адреса отправителя могут помещаться нули. Можно отказаться и от подсчета контрольной суммы, однако следует учесть, что протокол IP подсчитывает контрольную сумму только для заголовка IP-пакета, игнорируя поле данных.

Протокол надежной доставки сообщений TCP

В стеке протоколов TCP/IP протокол ***TCP*** (***Transmission Control Protocol***) работает так же, как и протокол UDP, на транспортном уровне. Он обеспечивает надежную транспортировку данных между прикладными процессами путем установления логического соединения.

Сегменты TCP

Единицей данных протокола TCP является сегмент. Информация, поступающая к протоколу TCP в рамках логического соединения от протоколов более высокого уровня, рассматривается протоколом TCP как неструктурированный поток байт. Поступающие данные буферизуются средствами TCP. Для передачи на сетевой уровень из буфера «вырезается» некоторая непрерывная часть данных, называемая сегментом.

В протоколе TCP предусмотрен случай, когда приложение обращается с запросом о срочной передаче данных (бит PSH в запросе установлен в 1). В этом случае протокол TCP, не ожидая заполнения буфера до уровня размера сегмента, немедленно передает указанные данные в сеть. О таких данных говорят, что они передаются вне потока — out of band.

Не все сегменты, посланные через соединение, будут одного и того же размера, однако оба участника соединения должны договориться о максимальном размере сегмента, который они будут использовать. Этот размер выбирается таким образом, чтобы при упаковке сегмента в IP-пакет он помещался туда целиком, то есть максимальный размер сегмента не должен превосходить максимального размера поля данных IP-пакета. В противном случае пришлось бы выполнять фрагментацию, то есть делить сегмент на несколько частей, для того, чтобы он вместился в IP-пакет.

Аналогичные проблемы решаются и на сетевом уровне. Для того, чтобы избежать фрагментации, должен быть выбран соответствующий максимальный размер IP-пакета. Однако при этом должны быть приняты во внимание максимальные размеры поля данных кадров (MTU) всех протоколов канального уровня, используемых в сети. Максимальный размер сегмента не должен превышать минимальное значение на множестве всех MTU составной сети.

Порты и установление TCP-соединений

В протоколе TCP также, как и в UDP, для связи с прикладными процессами используются порты. Номера портам присваиваются аналогичным образом: имеются стандартные, зарезервированные номера (например, номер 21 закреплен за сервисом

FTP, 23 — за telnet), а менее известные приложения пользуются произвольно выбранными локальными номерами.

Однако в протоколе TCP порты используются несколько иным способом. Для организации надежной передачи данных предусматривается установление логического соединения между двумя прикладными процессами. В рамках соединения осуществляется обязательное подтверждение правильности приема для всех переданных сообщений, и при необходимости выполняется повторная передача. Соединение в TCP позволяет вести передачу данных одновременно в обе стороны, то есть полнодуплексную передачу.

Соединение в протоколе TCP идентифицируется парой полных адресов обоих взаимодействующих процессов (оконечных точек). Адрес каждой из оконечных точек включает IP-адрес (номер сети и номер компьютера) и номер порта. Одна оконечная точка может участвовать в нескольких соединениях.

Установление соединения выполняется в следующей последовательности:

- При установлении соединения одна из сторон является инициатором. Она посылает запрос к протоколу TCP на открытие порта для передачи (active open).
- После открытия порта протокол TCP на стороне процесса-инициатора посылает запрос процессу, с которым требуется установить соединение.
- Протокол TCP на приемной стороне открывает порт для приема данных (passive open) и возвращает квитанцию, подтверждающую прием запроса.
- Для того чтобы передача могла вестись в обе стороны, протокол на приемной стороне также открывает порт для передачи (active port) и также передает запрос к противоположной стороне.
- Сторона-инициатор открывает порт для приема и возвращает квитанцию. Соединение считается установленным. Далее происходит обмен данными в рамках данного соединения.

Концепция квитирования

В рамках соединения правильность передачи каждого сегмента должна подтверждаться квитанцией получателя. ***Квитиро-***

вание — это один из традиционных методов обеспечения надежной связи. Идея квитирования состоит в следующем.

Для того, чтобы можно было организовать повторную передачу искаженных данных отправитель нумерует отправляемые единицы передаваемых данных (далее для простоты называемые кадрами). Для каждого кадра отправитель ожидает от приемника так называемую положительную квитанцию — служебное сообщение, извещающее о том, что исходный кадр был получен и данные в нем оказались корректными. Время этого ожидания ограничено — при отправке каждого кадра передатчик запускает таймер, и если по его истечению положительная квитанция на получена, то кадр считается утерянным. В некоторых протоколах приемник, в случае получения кадра с искаженными данными должен отправить отрицательную квитанцию — явное указание того, что данный кадр нужно передать повторно.

Существуют два подхода к организации процесса обмена положительными и отрицательными квитанциями: с простоями и с организацией «окна».

Метод с простоями требует, чтобы источник, пославший кадр, ожидал получения квитанции (положительной или отрицательной) от приемника и только после этого посылал следующий кадр (или повторял искаженный). В этом случае производительность обмена данными существенно снижается — хотя передатчик и мог бы послать следующий кадр сразу же после отправки предыдущего, он обязан ждать прихода квитанции. Снижение производительности для этого метода коррекции особенно заметно на низкоскоростных каналах связи, то есть в территориальных сетях.

Скользящее окно. Во втором методе для повышения коэффициента использования линии источнику разрешается передать некоторое количество кадров в непрерывном режиме, то есть в максимально возможном для источника темпе, без получения на эти кадры ответных квитанций. Количество кадров, которые разрешается передавать таким образом, называется ***размером окна***. Обычно кадры при обмене нумеруются циклически, от 1 до N. При отправке кадра с номером 1 источнику разрешается передать еще N – 1 кадров до получения квитанции на кадр 1. Если же за это время квитанция на кадр 1 так и не пришла, то процесс пере-

дачи приостанавливается, и по истечению некоторого тайм-аута кадр 1 считается утерянным (или квитанция на него утеряна) и он передается снова.

Если же поток квитанций поступает более-менее регулярно, в пределах допуска в N кадров, то скорость обмена достигает максимально возможной величины для данного канала и принятого протокола.

Этот алгоритм называют алгоритмом скользящего окна. Действительно, при каждом получении квитанции окно перемещается (скользит), захватывая новые данные, которые разрешается передавать без подтверждения.

Реализация скользящего окна в протоколе TCP

В протоколе TCP реализована разновидность алгоритма квитирования с использованием окна. Особенность этого алгоритма состоит в том, что, хотя единицей передаваемых данных является сегмент, окно определено на множестве нумерованных байт неструктурированного потока данных, поступающих с верхнего уровня и буферизуемых протоколом TCP.

Квитанция посылается только в случае правильного приема данных, отрицательные квитанции не посылаются. Таким образом, отсутствие квитанции означает либо прием искаженного сегмента, либо потерю сегмента, либо потерю квитанции.

В качестве квитанции получатель сегмента отсылает ответное сообщение (сегмент), в которое помещает число, на единицу превышающее максимальный номер байта в полученном сегменте. Если размер окна равен N, а последняя квитанция содержала значение M, то отправитель может посылать новые сегменты до тех пор, пока в очередной сегмент не попадет байт с номером N+M. Этот сегмент выходит за рамки окна, и передачу в таком случае необходимо приостановить до прихода следующей квитанции.

Выбор тайм-аута

Выбор ***времени ожидания*** (***тайм-аута***) очередной квитанции является важной задачей, результат решения которой влияет на производительность протокола TCP.

Тайм-аут не должен быть слишком коротким, чтобы по возможности исключить избыточные повторные передачи, которые снижают полезную пропускную способность системы. Но он не должен быть и слишком большим, чтобы избежать длительных простоев, связанных с ожиданием несуществующей или «заблудившейся» квитанции.

При выборе величины тайм-аута должны учитываться скорость и надежность физических линий связи, их протяженность и многие другие подобные факторы. В протоколе TCP тайм-аут определяется с помощью достаточно сложного адаптивного алгоритма, идея которого состоит в следующем. При каждой передаче засекается время от момента отправки сегмента до прихода квитанции о его приеме (время оборота). Получаемые значения времен оборота усредняются с весовыми коэффициентами, возрастающими от предыдущего замера к последующему. Это делается с тем, чтобы усилить влияние последних замеров. В качестве тайм-аута выбирается среднее время оборота, умноженное на некоторый коэффициент. Практика показывает, что значение этого коэффициента должно превышать 2. В сетях с большим разбросом времени оборота при выборе тайм-аута учитывается и дисперсия этой величины.

Реакция на перегрузку сети

Варьируя величину окна, можно повлиять на загрузку сети. Чем больше окно, тем большую порцию неподтвержденных данных можно послать в сеть. Если сеть не справляется с нагрузкой, то возникают очереди в промежуточных узлах — маршрутизаторах и в конечных узлах-компьютерах.

При переполнении приемного буфера конечного узла «перегруженный» протокол TCP, отправляя квитанцию, помещает в нее новый, уменьшенный размер окна. Если он совсем отказывается от приема, то в квитанции указывается окно нулевого размера. Однако даже после этого приложение может послать сообщение на отказавшийся от приема порт. Для этого, сообщение должно сопровождаться пометкой «срочно» (бит URG в запросе установлен в 1). В такой ситуации порт обязан принять сегмент,

даже если для этого придется вытеснить из буфера уже находящиеся там данные.

После приема квитанции с нулевым значением окна протокол-отправитель время от времени делает контрольные попытки продолжить обмен данными. Если протокол-приемник уже готов принимать информацию, то в ответ на контрольный запрос он посылает квитанцию с указанием ненулевого размера окна.

Другим проявлением перегрузки сети является переполнение буферов в маршрутизаторах. В таких случаях они могут централизовано изменить размер окна, посылая управляющие сообщения некоторым конечным узлам, что позволяет им дифференцировано управлять интенсивностью потока данных в разных частях сети.

Формат сообщений TCP

Сообщения протокола TCP называются сегментами и состоят из заголовка и блока данных. Заголовок сегмента имеет следующие поля:

- ***Порт источника*** (***SOURS PORT***) занимает 2 байта, идентифицирует процесс-отправитель.
- ***Порт назначения*** (***DESTINATION PORT***) занимает 2 байта, идентифицирует процесс-получатель.
- ***Последовательный номер*** (***SEQUENCE NUMBER***) занимает 4 байта, указывает номер байта, который определяет смещение сегмента относительно потока отправляемых данных.
- ***Подтвержденный номер*** (***ACKNOWLEDGEMENT NUMBER***) занимает 4 байта, содержит максимальный номер байта в полученном сегменте, увеличенный на единицу; именно это значение используется в качестве квитанции.
- ***Длина заголовка*** (***HLEN***) занимает 4 бита, указывает длину заголовка сегмента TCP, измеренную в 32-битовых словах. Длина заголовка не фиксирована и может изменяться в зависимости от значений, устанавливаемых в поле Опции.
- ***Резерв*** (***RESERVED***) занимает 6 битов, поле зарезервировано для последующего использования.

- ***Кодовые биты*** (***CODE BITS***) занимают 6 битов, содержат служебную информацию о типе данного сегмента, задаваемую установкой в единицу соответствующих бит этого поля:

– ***URG*** — срочное сообщение;

– ***ACK*** — квитанция на принятый сегмент;

– ***PSH*** — запрос на отправку сообщения без ожидания заполнения буфера;

– ***RST*** — запрос на восстановление соединения;

– ***SYN*** — сообщение используемое для синхронизации счетчиков переданных данных при установлении соединения;

– ***FIN*** — признак достижения передающей стороной последнего байта в потоке передаваемых данных.

- ***Окно*** (***WINDOW***) занимает 2 байта, содержит объявляемое значение размера окна в байтах.
- ***Контрольная сумма*** (***CHECKSUM***) занимает 2 байта, рассчитывается по сегменту.
- ***Указатель срочности*** (***URGENT POINTER***) занимает 2 байта, используется совместно с кодовым битом URG, указывает на конец данных, которые необходимо срочно принять, несмотря на переполнение буфера.
- ***Опции*** (***OPTIONS***) — это поле имеет переменную длину и может вообще отсутствовать, максимальная величина поля 3 байта; используется для решения вспомогательных задач, например, при выборе максимального размера сегмента.
- ***Заполнитель*** (***PADDING***) может иметь переменную длину, представляет собой фиктивное поле, используемое для доведения размера заголовка до целого числа 32-битовых слов.

Протокол обмена управляющими сообщениями ICMP

Протокол обмена управляющими сообщениями ***ICMP*** (***Internet Control Message Protocol***) позволяет маршрутизатору сообщить конечному узлу об ошибках, с которыми машрутизатор столкнулся при передаче какого-либо IP-пакета от данного конечного узла.

Управляющие сообщения ICMP не могут направляться промежуточному маршрутизатору, который участвовал в передаче пакета, с которым возникли проблемы, так как для такой посылки

нет адресной информации — пакет несет в себе только адрес источника и адрес назначения, не фиксируя адреса промежуточных маршрутизаторов.

Протокол ICMP — это протокол сообщения об ошибках, а не протокол коррекции ошибок. Конечный узел может предпринять некоторые действия для того, чтобы ошибка больше не возникала, но эти действия протоколом ICMP не регламентируются.

Каждое сообщение протокола ICMP передается по сети внутри пакета IP. Пакеты IP с сообщениями ICMP маршрутизируются точно так же, как и любые другие пакеты, без приоритетов, поэтому они также могут теряться. Кроме того, в загруженной сети они могут вызывать дополнительную загрузку маршрутизаторов. Для того, чтобы не вызывать лавины сообщения об ошибках, потери пакетов IP, переносящие сообщения ICMP об ошибках, не могут порождать новые сообщения ICMP.

Формат сообщений протокола ICMP

Существует несколько типов сообщений ICMP. Каждый тип сообщения имеет свой формат, при этом все они начинаются с общих трех полей: 8-битного целого числа, обозначающего тип сообщения (TYPE), 8-битного поля кода (CODE), который конкретизирует назначение сообщения, и 16-битного поля контрольной суммы (CHECKSUM). Кроме того, сообщение ICMP всегда содержит заголовок и первые 64 бита данных пакета IP, который вызвал ошибку. Это делается для того, чтобы узел-отправитель смог более точно проанализировать причину ошибки, так как все протоколы прикладного уровня стека TCP/IP содержат наиболее важную информацию для анализа в первых 64 битах своих сообщений.

Поле типа может иметь следующие значения:

- 0 Эхо-ответ (***Echo Replay***).
- *3* Узел назначения недостижим (***Destination Unreachable***).
- 4 Подавление источника (***Source Quench***).
- 5 Перенаправление маршрута (***Redirect***).
- 8 Эхо-запрос (***Echo Request***).
- 11 Истечение времени дейтаграммы (***Time Exceeded for a Datagram***).

- 12 Проблема с параметром пакета (***Parameter Problem on a Datagram***).
- 13 Запрос отметки времени (***Timestamp Request***).
- 14 Ответ отметки времени (***Timestamp Replay***).
- 17 Запрос маски (***Address Mask Request***).
- 18 Ответ маски (***Address Mask Replay***).

Как видно из используемых типов сообщений, протокол ICMP представляет собой некоторое объединение протоколов, решающих свои узкие задачи.

Эхо-протокол

Протокол ICMP предоставляет сетевым администраторам средства для тестирования достижимости узлов сети. Эти средства представляют собой очень простой эхо-протокол, включающий обмен двумя типами сообщений: эхо-запрос и эхо-ответ. Компьютер или маршрутизатор посылают по интерсети эхо-запрос, в котором указывают IP-адрес узла, достижимость которого нужно проверить. Узел, который получает эхо-запрос, формирует и отправляет эхо-ответ и возвращает сообщение узлу — отправителю запроса. В запросе могут содержаться некоторые данные, которые должны быть возвращены в ответе. Так как эхо-запрос и эхо-ответ передаются по сети внутри IP-пакетов, то их успешная доставка означает нормальное функционирование всей транспортной системы интерсети.

Во многих операционных системах используется утилита ping, которая предназначена для тестирования достижимости узлов. Эта утилита обычно посылает серию эхо-запросов к тестируемому узлу и предоставляет пользователю статистику об утерянных эхо-ответах и среднем времени реакции сети на запросы.

Сообщения о недостижимости узла назначения

Когда маршрутизатор не может передать или доставить IP-пакет, он отсылает узлу, отправившему этот пакет, сообщение «Узел назначения недостижим» (тип сообщения — 3). Это сообщение содержит в поле кода значение, уточняющее причину, по которой пакет не был доставлен. Причина кодируется следующим образом:

- 0 Сеть недостижима.
- 1 Узел недостижим.
- 2 Протокол недостижим.
- 3 Порт недостижим.
- 4 Требуется фрагментация, а бит DF установлен.
- 5 Ошибка в маршруте, заданном источником.
- 6 Сеть назначения неизвестна.
- 7 Узел назначения неизвестен.
- 8 Узел-источник изолирован.
- 9 Взаимодействие с сетью назначения административно запрещено.
- 10 Взаимодействие с узлом назначения административно запрещено.
- 11 Сеть недостижима для заданного класса сервиса.
- 12 Узел недостижим для заданного класса сервиса.

Маршрутизатор, обнаруживший по какой-либо причине, что он не может передать IP-пакет далее по сети, должен отправить ICMP-сообщение узлу-источнику, и только потом отбросить пакет. Кроме причины ошибки, ICMP-сообщение включает также заголовок недоставленного пакета и его первые 64 бита поля данных.

Узел или сеть назначения могут быть недостижимы из-за временной неработоспособности аппаратуры, из-за того, что отправитель указал неверный адрес назначения, а также из-за того, что маршрутизатор не имеет данных о маршруте к сети назначения.

Недостижимость протокола и порта означают отсутствие реализации какого-либо протокола прикладного уровня в узле назначения или же отсутствие открытого порта протоколов UDP или TCP в узле назначения.

Ошибка фрагментации возникает тогда, когда отправитель послал в сеть пакет с признаком DF, запрещающим фрагментацию, а маршрутизатор столкнулся с необходимостью передачи этого пакета в сеть со значением MTU меньшим, чем размер пакета.

Перенаправление маршрута

Маршрутные таблицы у компьютеров обычно являются статическими, так как конфигурируются администратором сети, а у

маршрутизаторов — динамическими, формируемыми автоматически с помощью протоколов обмена маршрутной информации. Поэтому с течением времени при изменении топологии сети маршрутные таблицы компьютеров могут устаревать. Кроме того, эти таблицы обычно содержат минимум информации, например, только адреса нескольких маршрутизаторов.

Для корректировки поведения компьютеров маршрутизатор может использовать сообщение протокола ICMP, называемое «Перенаправление маршрута» (Redirect).

Это сообщение посылается в том случае, когда маршрутизатор видит, что компьютер отправляет пакет некоторой сети назначения нерациональным образом, то есть не тому маршрутизатору локальной сети, от которого начинается более короткий маршрут к сети назначения.

Механизм перенаправления протокола ICMP позволяет компьютерам содержать в конфигурационном файле только IP-адреса его локальных маршрутизаторов. С помощью сообщений о перенаправлении маршрутизаторы будут сообщать компьютеру всю необходимую ему информацию о том, какому маршрутизатору следует отправлять пакеты для той или иной сети назначения. То есть маршрутизаторы передадут компьютеру нужную ему часть их таблиц маршрутизации.

В сообщении «Перенаправление маршрута» маршрутизатор помещает IP-адрес маршрутизатора, которым нужно пользоваться в дальнейшем, и заголовок исходного пакета с первыми 64 битами его поля данных. Из заголовка пакета узел узнает, для какой сети необходимо пользоваться указанным маршрутизатором.

Служебные утилиты стека протоколов TCP/IP

Утилита ipconfig

Утилита ***ipconfig*** используется для проверки параметров конфигурации узла, включая IP-адрес, маску подсети и шлюз по умолчанию. Это полезно при выяснении, успешно ли прошла инициализация TCP/IP и не дублируется ли IP-адрес, указанный в конфигурации. Синтаксис команды:

```
ipconfig /all
```

Если протокол инициализировался успешно с заданной конфигурацией, то на экране отобразятся IP-адрес, маска подсети и шлюз по умолчанию. Если адрес, заданный в конфигурации уже используется другим узлом в сети на том же сегменте, то отобразится заданный IP-адрес, но маска подсети будет равна 0.0.0.0.

Утилита ping

Утилита ***ping*** (*Packet InterNet Groper*) предназначена для тестирования соединений. Это диагностическое средство тестирует конфигурации TCP/IP и позволяет определить неисправности соединения. Утилита Ping использует пакеты эхо-запроса (*echo request*) и эхо-ответа (*echo reply*) протокола ***ICMP*** (*Internet Control Message Protocol*) для проверки доступности и работоспособности определенного узла TCP/IP. Синтаксис команды:

```
ping 212.192.122.65
```

Если проверка прошла успешно, то отобразится сообщение типа:

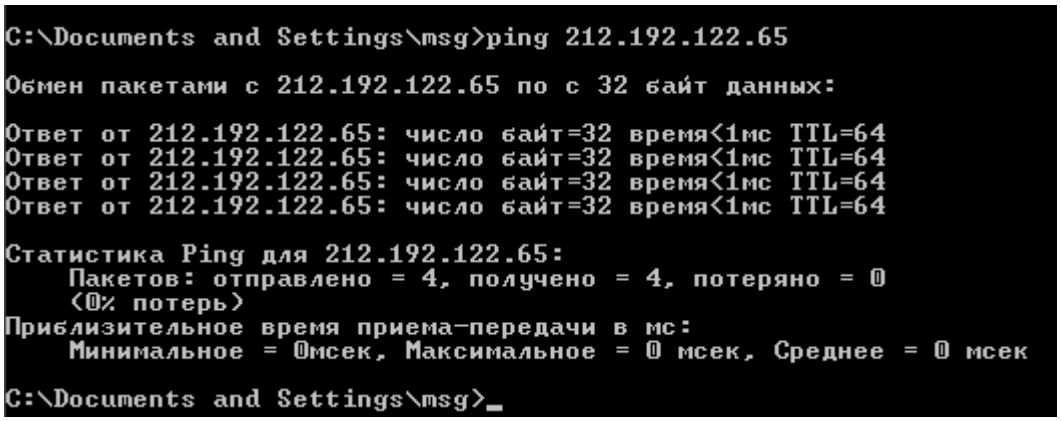


Рис. 11

Помимо этого, с помощью Ping можно оценить время возврата пакета от хоста, что дает представление о том, насколько далеко находится хост. Когда принимается ICMP эхо отклик, печатается номер последовательности, затем параметр время жизни (TTL) и рассчитанное время возврата. (TTL это поле времени жизни в IP заголовке. Программа ping печатает полученное TTL каждый раз, когда принимается эхо отклик.

Ping может рассчитать время возврата, так как она сохраняет время, когда был отправлен эхо запрос, в разделе данных ICMP сообщения. Когда отклик возвращается, эти данные извлекаются и сравниваются с текущим временем.

Программа traceroute

Программа ***traceroute*** — отладочное средство, которое позволяет лучше понять устройство протоколов TCP/IP. Traceroute позволяет просмотреть маршрут, по которому двигаются IP дейтаграммы от одного хоста к другому. С помощью traceroute можно воспользоваться IP опцией маршрутизации от источника.

Traceroute использует ICMP и поле TTL в IP заголовке. Поле TTL (время жизни) это 8-битное поле, которое отправитель устанавливает в какое-либо значение.

Каждый маршрутизатор, который обрабатывает дейтаграмму, уменьшает значение TTL на единицу или на количество секунд, в течение которых маршрутизатор обрабатывал дейтаграмму. Так как большинство маршрутизаторов задерживает дейтаграмму меньше чем секунду, поле TTL, как правило, уменьшается на единицу и довольно точно соответствует количеству пересылок.

Логика работы программы traceroute

На хост назначения отправляется IP дейтаграмма с TTL, установленным в единицу. Первый маршрутизатор, который должен обработать дейтаграмму, уничтожает ее (так как TTL равно 1) и отправляет ICMP сообщение об истечении времени (time exceeded). Таким образом, определяется первый маршрутизатор в маршруте. Затем Traceroute отправляет дейтаграмму с TTL равным 2, что позволяет получить IP адрес второго маршрутизатора. Это продолжается до тех пор, пока дейтаграмма не достигнет хоста назначения.

Для определения хоста назначения в исходящих UDP дейтаграммах, которые посылает traceroute, устанавливается несуществующий номер UDP порта (больше чем 30000), что делает невозможным обработку этой дейтаграммы каким-либо приложением. Поэтому когда прибывает подобная дейтаграмма, UDP модуль

хоста назначения генерирует ICMP сообщение «порт недоступен» (port unreachable). Ttaceroute определяет тип вернувшегося ICMP сообщения — если это сообщение об истечении времени, то это сообщение с промежуточного узла, а если сообщение о недоступности порта — то это узел назначения.

В операционных системах семейства Windows эта команда имеет транскрипцию ***tracert*** вместо traceroute.

ЛАБОРАТОРНАЯ РАБОТА № 3
Служебные утилиты стека протоколов TCP/IP

Продолжительность — 2 часа.
Максимальный рейтинг — 7 баллов.

ЦЕЛЬ РАБОТЫ

Изучить форматы основных протоколов стека TCP/IP и утилит настройки параметров и получения предварительной информации.

ЗАДАНИЕ

1. Изучить работу и основные ключи следующих команд:
1.1. hostname;
1.2. ipconfig;
1.2. ping;
1.3. route/ tracert;
1.4. netstat;
1.5. arp.
2. Протестировать все указанные утилиты на вашем компьютере в вашей сети. Результаты тестирования привести в отчете. Отчет должен содержать, для каждой команды:
2.1. назначение команды;
2.2. описание основных ключей;
2.3. примеры использования.
3. Привести краткие разъяснения для всех данных и всех настроек протоколов стека TCP/IP, полученные в результате тестирования.

ЛАБОРАТОРНАЯ РАБОТА № 4
Стек протоколов TCP\IP. Пассивное оборудование ЛВС

Продолжительность — 4 часа.
Максимальный рейтинг — 7 баллов.

ЦЕЛЬ РАБОТЫ

Изучить стек протоколов TCP\IP. Научиться настраивать пассивное оборудование ЛВС: сетевую карту и концентратор (hub).

ЗАДАНИЕ

1. Отформатировать HDD на компьютере и установить ОС Windows 98 или ОС Windows XP. Установить драйвера сетевого адаптера с настройками «по умолчанию».

2. Подключить ПК в одноранговую (без сервера) ЛВС, организованную при помощи концентратора (hub). Установить все необходимые настройки сетевого адаптера. Проверить, что все сетевые устройства ЛВС доступны через папку «Сетевое окружение» на рабочем столе Windows.

3. Подключить свой ПК в учебную ЛВС (домен Windows NT) ***SMBIE***. Настройки выделения IP адреса сделать динамическими (DHCP). Проверить, что все сетевые устройства ЛВС SMBIE доступны.

4. Настроить IP адрес шлюза. Настроить Internet браузер. Проверить доступ к ресурсам Inernet.

ПРИМЕЧАНИЯ

1. Для определения коннекта между устройствами сети и для определения параметров шлюза, назначенными вашему ПК сервером DHCP пользоваться сетевыми утилитами, изученными в Лабораторной работе № 3.

5 ПРОГРАММЫ-АНАЛИЗАТОРЫ ПАКЕТОВ

Использование программ-анализаторов сетевых пакетов помогает лучше понять процессы, происходящие при работе сетевого оборудования и обычных компьютеров в локальной сети. Наглядно показывая механизм инкапсуляции информационных пакетов уровня приложения в служебные кадры нижележащих уровней, это позволяет увидеть взаимодействие сетевых протоколов разного уровня и осознать необходимость использования многоуровневых моделей сетевого взаимодействия.

В настоящей лабораторной работе исследование сетевого трафика выполняется с помощью программы Packetyzer 5.0.0. Данная программа является фактически Windows-версией широко известной программы-анализатора сетевых пакетов Ethereal для UNIX-платформ. Эта программа разработана компанией Network Chemistry и распространяется с лицензией GNU GPL. При запуске программы автоматически отображается окно выбора сетевого адаптера и режима его работы, рис. 12.

Рис. 12 — Окно выбора
сетевого адаптера

Обычно драйвер сетевого адаптера настроен таким образом, что адаптер захватывает только пакеты, отправленные по MAC-адресу, соответствующему адресу адаптера. Установив флажок «*Capture packets in promiscuous mode*» можно перевести адаптер в режим захвата всех пакетов, распространяющихся по сети. Если работа выполняется в сети Ethernet, перевод адаптера в такой режим эффективен только в том случае, когда сегмент сети работа-

ет, как среда общего доступа для нескольких рабочих станций (например в беспроводных сетях или сетях, построенных на концентраторах). Если же сеть Ethernet построена на коммутаторах (switch), перевод сетевого адаптера в этот режим не изменит количество захватываемых пакетов.

Рабочее пространство программы разделено на три окна рис. 13. В левом окне отображается структура пакета, выделенного в правом верхнем окне. Правое нижнее окно разделено на два подокна, в одном из которых показывается содержимое пакета в шестнадцатеричном виде, а в другом — преобразованное в текст в соответствии с ASCII кодом.

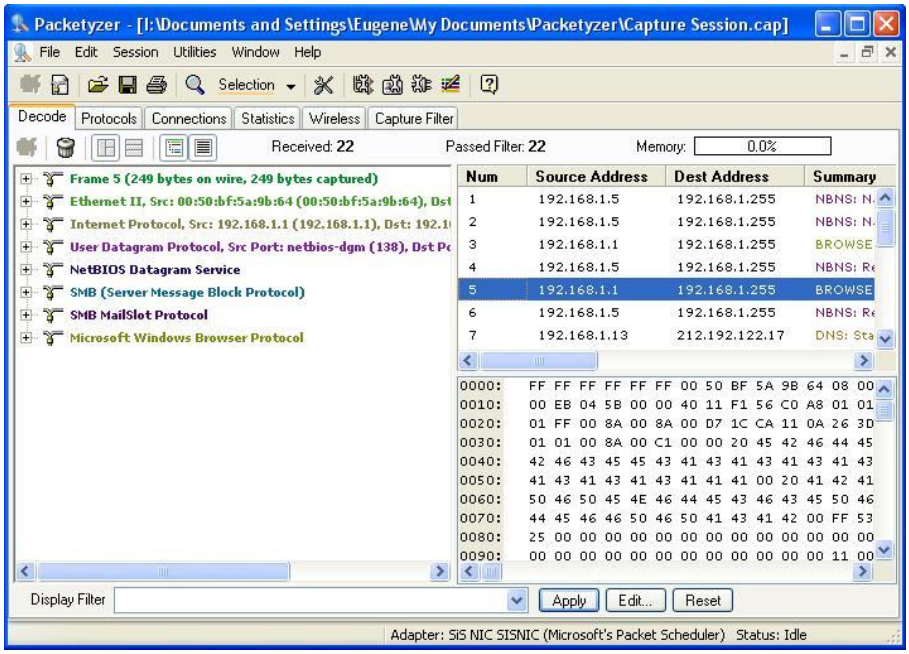


Рис. 13 — Главное окно программы Packetyzer

Захват пакетов программа начинает выполнять при нажатии клавиши F5 или клике мышкой на значке  на панели инструментов. Вид значка в процессе захвата изменяется  . При этом последовательно пронумерованные захваченные пакеты начинают появляться в правом верхнем окне. Нажатие клавиши **F6** или повторный клик мышкой на том же значке останавливает процесс захвата пакетов.

Захваченные пакеты могут быть отфильтрованы с помощью команды контекстного меню «Watch», например, по текущей сес-

сии, рис. 14. Кроме того, в программе есть широкие возможности для гибкой настройки параметров фильтрации и большое число часто используемых фильтров, увидеть которые можно раскры список «*Display Filter*» внизу основного окна программы.

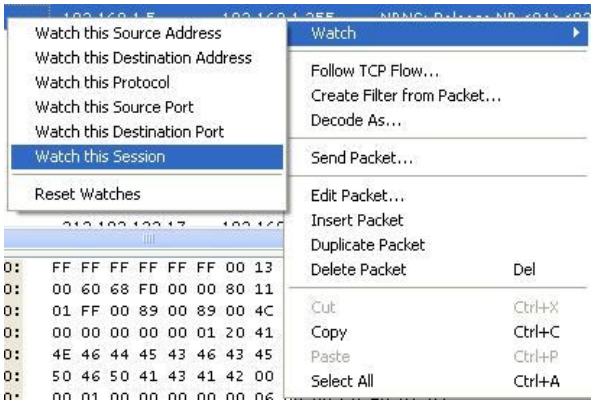


Рис. 14 — Фильтрация захваченных пакетов
по текущей сессии

Клавиши «*Apply*», «*Edit…*» и «*Reset*» управляют режимом фильтрации, например, нажатие клавиши «Reset» отменяет примененный ранее фильтр.

ЛАБОРАТОРНАЯ РАБОТА № 5
Анализ сетевого трафика, форматы пакетов протокола TCP\IP, инкапсуляция

Продолжительность — 4 часа.
Максимальный рейтинг — 8 баллов.

ЦЕЛЬ РАБОТЫ

Проанализировать сетевой трафик рабочей станции при помощи программы ***Packetyzer 5.0.0.*** Изучить форматы пакетов стека сетевых протоколов TCP\IP. Проследить и описать последовательность инкапсуляции пакетов.

ЗАДАНИЕ

При выполнении задания, выяснить и отразить в отчете следующие вопросы:

1. Какой протокол, используется программой-приложением.
2. Какова структура кадра этого протокола (Названия, размер и содержание служебных полей кадра.
3. Каким протоколом переносятся данные. Размер и расположение поля содержащего данные (при наличии такого поля)).
4. Какие еще протоколы участвуют в передаче информационных пакетов прикладной программы.
5. На каких уровнях модели сетевого взаимодействия работает каждый из протоколов.
6. Какие еще функции, наряду с основной задачей — передать информацию по сети, выполняются этими протоколами.

ВАРИАНТЫ ЗАДАНИЙ

№ варианта	Задание
1	Проанализировать сетевой трафик, возникающий при отправке на реально существующий удаленный компьютер серии эхо-пакетов командой ping по IP-адресу удаленного компьютера

№ варианта	Задание
2	Проанализировать сетевой трафик, возникающий при отправке на реально существующий удаленный компьютер серии эхо-пакетов командой ping(запрос по NetBIOS-имени компьютера)
3	Проанализировать сетевой трафик, возникающий при отправке на несуществующий удаленный компьютер серии эхо-пакетов командой ping по IP-адресу
4	Проанализировать сетевой трафик, возникающий при отправке на несуществующий удаленный компьютер серии эхо-пакетов командой ping по доменному имени (например – mail.ru)
5	Проанализировать сетевой трафик, возникающий при отправке на удаленный компьютер текстового сообщения командой net send по IP-адресу компьютера
6	Проанализировать сетевой трафик, возникающий при отправке на удаленный компьютер текстового сообщения командой net send по NetBIOS-имени компьютера
7	Проанализировать сетевой трафик, возникающий при установлении с удаленным компьютером сетевого соединения с помощью программы NetMeeting
8	Проанализировать сетевой трафик, возникающий при запуске сетевого пейджера chat в программе NetMeeting
9	Проанализировать сетевой трафик, возникающий при передаче на удаленный компьютер файла в программе NetMeeting
10	Проанализировать сетевой трафик, возникающий при установлении с удаленным компьютером(switch) Telnet-соединения
11	Проанализировать сетевой трафик, возникающий при установлении с удаленным компьютером (switch) соединения с помощью программы HyperTerminal
12	Проанализировать сетевой трафик, возникающий при установлении с удаленным компьютером (switch) со-

№ варианта	Задание
	единения с помощью программы-браузера
13	Проанализировать сетевой трафик, возникающий в процессе авторизации на коммутаторе с помощью программы Telnet
14	Проанализировать сетевой трафик, возникающий в процессе авторизации на коммутаторе с помощью программы Hyper Terminal
15	Проанализировать сетевой трафик, возникающий в процессе авторизации на коммутаторе с помощью программы-браузера
16	Проанализировать сетевой трафик, возникающий при просмотре на локальном компьютере состава локальной сети через оснастку «Сетевое окружение»
17	Проанализировать сетевой трафик, возникающий при просмотре на локальном компьютере состава локальной сети с помощью команды Net view
18	Проанализировать сетевой трафик, возникающий при запуске программы Windows Messenger
19	Проанализировать сетевой трафик, возникающий при запуске программы соединения с удаленным рабочим столом (Remote Desktop Connection)
20	Проанализировать сетевой трафик, возникающий при запуске программы ftp, – соединения с удаленным компьютером по протоколу ftp
21	Проанализировать сетевой трафик, возникающий при отправке задания на несуществующий сетевой принтер
22	Проанализировать сетевой трафик, возникающий при отправке сообщения электронной почты по несуществующему адресу

6 МЕТОДЫ ЛОГИЧЕСКОГО КОДИРОВАНИЯ (КАНАЛЬНЫЙ УРОВЕНЬ). КОРРЕКТИРУЮЩИЕ И ОБНАРУЖИВАЮЩИЕ КОДЫ. КОД ХЭММИНГА

Наиболее известные из корректирующих кодов — коды Хемминга. В них исправляются ошибки кратности $r = [d - 1]/2$ и обнаруживаются ошибки кратности $d - 1$.

Коды Хемминга, применительно к двоичной системе счисления, предназначены либо для исправления одиночных ошибок (при $d = 3$), либо для исправления одиночных и обнаружения без исправления двойных ошибок ($d = 4$).

Исправление ошибок возможно благодаря избыточности кода Хемминга — к m информационным битам добавлено k контрольных, благодаря которым возможно определить и/или исправить ошибки. Количество необходимых контрольных бит можно определить из следующего неравенства: $m + k + 1 \leq 2^k$. В отличие от других методов коррекции ошибки, где контрольные биты дописываются в конец или начало блока данных (либо вообще в другом пакете данных), биты кода Хэмминга записываются вместе с данными в строго определённых позициях.

В таком коде n-значное число имеет m информационных и k контрольных разрядов. Каждый из контрольных битов является знаком четности для определенной группы информационных знаков слова. При декодировании производится k групповых проверок на четность. В результате каждой проверки в соответствующий разряд регистра ошибки записывается **0**, если проверка была успешной, или **1**, если была обнаружена нечетность.

Группы для проверки формируются таким образом, что в регистре ошибки после окончания проверки получается k-разрядное двоичное число, показывающее номер позиции ошибочного двоичного разряда. Изменение этого разряда — исправление ошибки. Позиция i-го контрольного знака имеет номер 2^{i-1}. При этом каждый контрольный знак входит лишь в одну группу проверки на четность.

Рассмотрим код Хемминга, предназначенный для исправления одиночных ошибок, т.е. код с минимальным кодовым расстоянием $d = 3$. Ошибка возможна в одной из n позиций. Следо-

вательно, число контрольных знаков, а значит, и число разрядов регистра ошибок должно удовлетворять условию:

$$k \geq \log_2(n+1),$$

тогда под информационные знаки остается m разрядов:

$$m \leq n - \log_2(n+1).$$

Пример 1

Рассмотри механизм работы кода Хэмминга на примере передачи 7-битового кода {**1110011**}. Для контроля целостности блока данных такой длины, нам необходимо 4 бита кода Хэмминга, которые записываются в позициях 1, 2, 4 и 8:

Позиция бита	11	10	9	8	7	6	5	4	3	2	1
Значение бита	1	1	1	**x**	0	0	1	**x**	1	**x**	**x**

Контрольная сумма формируется путем выполнения операции «*исключающее ИЛИ*» над кодами позиций ненулевых битов. В данном случае это 11, 10, 9, 5 и 3:

11	1011
10	1010
09	1001
05	0101
03	0011
сумма	**1110**

Полученная контрольная сумма записывается в соответствующие разряды блока данных — младший бит в младший разряд:

Позиция бита	11	10	9	8	7	6	5	4	3	2	1
Значение бита	1	1	1	**1**	0	0	1	**1**	1	**1**	**0**

Код Хэмминга сформирован: {**11110011110**}.

Теперь рассмотрим два случая ошибки:

1) ошибка в бите 7 — бит **0** заменён на бит **1**, таким образом, принят следующий код: {1111**1**011110}. Просуммировав номера позиций ненулевых битов получим сумму:

11	1011
10	1010
09	1001
08	1000
07	0111
05	0101
04	0100
03	0011
02	0010
сумма	**0111 =7**

2) Ошибка в бите 5 — бит **1** заменён на бит **0**, принят следующий код: {111100**0**1110}. Просуммируем коды позиций с ненулевыми битами:

11	1011
10	1010
09	1001
08	1000
04	0100
03	0011
02	0010
сумма	**0101 =5**

Найдена контрольная сумма в кодовых последовательностях, содержащих ошибку не равна 0. В обоих случаях контрольная сумма равна позиции бита, переданного с ошибкой. Теперь для исправления ошибки достаточно инвертировать бит, номер которого указан в контрольной сумме.

ЛАБОРАТОРНАЯ РАБОТА № 6
Методы логического кодирования (Канальный уровень). Корректирующие и обнаруживающие коды. Код Хэмминга

Продолжительность — 4 часа.
Максимальный рейтинг — 7 баллов.

ЦЕЛЬ РАБОТЫ

Изучить методы логического кодирования, используемые на канальном уровне (OSI) сетевых устройств. Изучить корректирующие и обнаруживающие коды на примере кода Хэмминга.

ЗАДАНИЕ

1. Составить корректирующий код Хэмминга для передачи битовой последовательности длины *m* (длина информационной последовательности и сама битовая последовательность выбирается из вариантов индивидуального задания). Определить число контрольных битов. Вычислить позиции контрольных битов.
2. Закодировать полученным кодом Хэмминга заданную в индивидуальном задании последовательность.
3. Внести единичную ошибку в кодовую последовательность. Обнаружить и исправить ошибку по алгоритму Хэмминга.
4. Внести двойную ошибку. Обнаружить ее по методу Хэмминга.
5. В отчете привести полный расчет кода, обосновать выбор длины контрольной суммы.

ВАРИАНТЫ ИНДИВИДУАЛЬНЫХ ЗАДАНИЙ

№ варианта	m	S
1	17	00000111000011000
2	18	111110101011011110
3	19	0111010000100111010
4	20	00100111010011011110
5	21	110000000011000101111

№ варианта	m	S
6	22	1100111110111010000001
7	23	10001011000001010101000
8	24	100011110011100100101100
9	25	1001110000011100111100111
10	26	10010011011010110111111100
11	17	00111000011100100
12	18	000001110001001011
13	19	0001011101011001010
14	20	00011001111011011011
15	21	011010000110010111110
16	22	0010110100100111100101
17	23	00110000001001010100000
18	24	110110011111010010111100
19	25	0111101111011100101000000
20	26	00101010010110000011110011
21	17	01100011110001011
22	18	000110111111110011
23	19	1111100011110010100
24	20	10010110110010110001
25	21	000110101100100111101
26	22	1010011011011010001011
27	23	11110001111100011110011
28	24	100000111110001111000110
29	25	1001111001110110101010100
30	26	01010010100000010110011101

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Основные функции методов логического кодирования.
2. Корректирующие и обнаруживающие коды. Кодовое расстояние.
3. Обнаруживающая способность кода. Корректирующая способность.
4. Линейные коды. Систематические линейные коды. Совершенные и квазисовершенные коды.

5. Циклические коды. CRC. Порждающий многочлен. Алгоритм кодирования\декодирования.

6. Код Хэмминга. Размер кодовой последовательности. Позиции кодовых знаков.

7 КОДИРОВАНИЕ ИНФОРМАЦИИ. КОМПРЕССИЯ. МЕТОД ХАФФМАНА

Метод Хаффмана (***Huffman code***) или минимально-избыточный префиксный код (*minimum-redundancy prefix code*) относится к статистическим методам кодирования.

Обычно для хранения данных и передачи сообщений используются коды фиксированной длины, например, код ASCII. Множество символов представляется некоторым количеством кодовых слов равной длины, которая для кода ASCII равна 8 битам. При этом для всех сообщений с одинаковым количеством символов требуется одинаковое количество битов при хранении и одинаковая ширина полосы пропускания при передаче.

Метод Хаффмана основан на кодировании более короткими кодовыми словами часто встречающиеся символы, а символы встречающиеся редко — более длинными. Подбирая кодовые последовательности таким образом, можно получить код с длиной, очень близкой к его энтропии (то есть информационной насыщенности). Кодовые слова при этом должны быть выбраны так, чтобы никакое из них не было префиксом другого кодового слова. Благодаря этому условию гарантируется возможность однозначного декодирования определенного закодированного текста.

В своей статье, опубликованной в 1952 г., *Дэвид Хаффман* описал алгоритм поиска множества кодов, которые минимизируют ожидаемую длину сообщений при условии, что известны вероятности появления каждого символа. В этом методе символам, имеющим меньшую вероятность появления, ставятся в соответствие более длинные кодовые слова.

Пусть $A = \{a_1, a_2, ..., a_n\}$ — алфавит из n различных символов, $P = \{p_1, p_2, ..., p_n\}$ — соответствующий ему набор положительных весов (вероятностей).

Тогда набор бинарных кодов $C = \{c_1, c_2, ..., c_n\}$, такой что:

(1) c_i не является префиксом для c_j, при $i \neq j$;

(2) $\sum_{i=1}^{n} p_i \cdot |c_i|$ — минимальна

называется минимально-избыточным префиксным кодом (*minimum-redundancy prefix code*) или иначе кодом Хаффмана.

Для создания кода Хаффмана должны быть известны, или рассчитаны, вероятности $P = \{p_1, p_2, ..., p_n\}$ появления символов исходного алфавита $A = \{a_1, a_2, ..., a_n\}$ в кодируемом сообщении *S*. В этом случае либо пользуются стандартизованными таблицами вероятностей, составленными для каждого языка, либо используют в качестве вероятностей частоту появления символа в исходном сообщении. Рассмотрим алгоритм Хаффмана на примере кодирования сообщения *S* = {DDABDECBCDBE}.

Составим список всех символов, встречающихся в исходном тексте, и определим количество появлений каждого символа в нем: символ A встречается 1 раз, B – 3, C – 2, D – 4 и E – 2 раза.

Алгоритм Хаффмана состоит из двух этапов — построения дерева вероятностей и построения кодов символов. На первом этапе составляется двоичный граф — дерево вероятностей, узлы которого строятся по простому правилу, а листьями являются символы, нуждающиеся в кодировании. На втором этапе ребра графа нумеруются символами **1** и **0**, а затем выписываются коды вершин графа.

- Составим список кодируемых символов (при этом будем рассматривать каждый символ как одноэлементное бинарное дерево, вес которого равен весу символа) как на рис. 15.

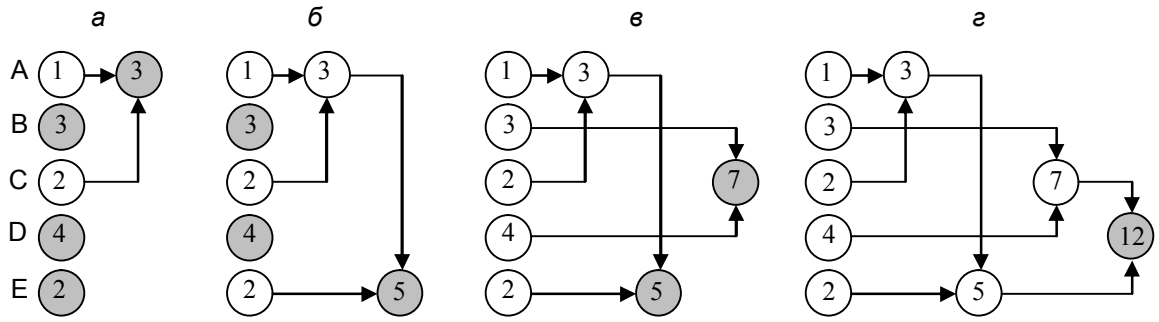


Рис. 15 —тапы построения дерева Хаффмана

- Из списка выберем два узла с наименьшим весом (на рис. 6.1(*а*) это листья **A** и **C** с вероятностями 1 и 2 соответственно).

- Сформируем новый узел и присоединим к нему, в качестве дочерних, два узла выбранных из списка. При этом вес сформированного узла положим равным сумме весов дочерних узлов (в нашем случае — 3).
- Теперь два использованных узла объединены в дерево с общим весом — 3. Отныне не будем принимать во внимание при поиске наименьших вероятностей два объединенных узла, но будем рассматривать новое дерево как полноценную структуру с частотой появления, равной сумме частот появления двух соединившихся вершин (на рис. 15 вершины выпавшие из дальнейшего анализа — белого цвета, а рассматриваемые вершины — серого).
- Будем повторять операцию объединения вершин до тех пор, пока не придем к корню дерева.
- На этом построение дерева вероятностей закончено и веса его вершин нам больше не нужны. Занумеруем теперь ребра полученного графа, начиная с вершины и присваивая **1** верхнему (правому) ребру и **0** — нижнему (левому), как на рис. 16,*а*.

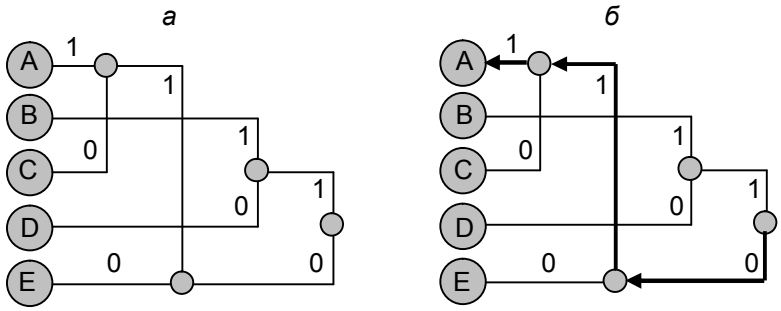


Рис. 16 — Составление кода Хаффмана

- Листовые узлы дерева Хаффмана соответствуют символам кодируемого алфавита (рис. 16). Глубина листовых узлов равна длине кода соответствующих символов.
- Код Хаффмана для каждого символа — это путь от корня дерева к соответствующему листовому узлу (на рис. 16,*б* стрелками обозначен путь для символа A). Его можно представить в виде битовой строки, в которой **0** соответствует выбору левого поддерева, а **1** — правого.

Выпишем коды $C = \{c_1, c_2, ..., c_n\}$ для всех символов в нашем примере:

E – **00,** D – **10,** B – **11,** C – **010,** A – **011.**

- Теперь у нас есть всё необходимое, для того чтобы закодировать сообщение *S*. Достаточно просто заменить каждый символ соответствующим ему кодом, получим строку
S′ = { 10100111110000101101010 1100 }.
- Декодируем сообщение *S′*. Начиная с корня дерева, будем двигаться вниз, выбирая левое поддерево, если очередной символ в битовом потоке равен **0**, и правое — если **1**. Дойдя до листового узла, мы декодируем соответствующий ему символ. Следуя этому алгоритму, мы в точности получим исходное сообщение *S*.

```
10-10011111000010110101011 00
10-10-0111110000101101010110 0
10-10-011-1110000101101010110 0
***
10-10-011-11-10-00-010-11-010-10-11-00
```

S = {DDABDECBCDBE}.

В теории кодирования информации доказывается, что код Хаффмана является префиксным, то есть код никакого символа не является началом кода какого-либо другого символа. Проверьте это на нашем примере. Из этого следует, что код Хаффмана однозначно восстановим получателем, даже если не сообщается длина кода каждого переданного символа. Получателю пересылают только дерево Хаффмана в компактном виде, а затем входная последовательность кодов символов декодируется им самостоятельно без какой-либо дополнительной информации.

Оценим теперь ***степень сжатия***. В исходном сообщении *S* было 12 символов, на каждый из которых отводилось бы по $\lceil \log_2 |A| \rceil = \lceil \log_2 5 \rceil = 3$ бита минимального равномерного кода (или по 8 бит ASCII), таким образом, размер *S* равнялся бы 12×3 = 36 бит (или 96 бит ASCII). Размер закодированного сообщения *S′* можно получить воспользовавшись формулой $\sum_{i=1}^{n} p_i \cdot |c_i|$ из определения, или непосредственно, подсчитав количество бит в *S′*, получим 27 бит.

Передача кодового дерева. Для того чтобы закодированное сообщение удалось декодировать, декодеру необходимо иметь

такое же кодовое дерево (в той или иной форме), какое использовалось при кодировании. Поэтому вместе с закодированными данными мы вынуждены сохранять соответствующее кодовое дерево. Ясно, что чем компактнее оно будет, тем лучше. Решить эту задачу можно несколькими способами. Самое очевидное решение — сохранить дерево в явном виде (т.е. как упорядоченное множество узлов и указателей того или иного вида). Это, пожалуй, самый расточительный и неэффективный способ. На практике он не используется.

Можно сохранить список частот символов (т. е. частотный словарь). С его помощью декодер без труда сможет реконструировать кодовое дерево. Хотя этот способ и менее расточителен, чем предыдущий, он не является наилучшим.

Наконец, можно использовать одно из свойств канонических кодов. Как уже было отмечено ранее, канонические коды вполне определяются своими длинами. Другими словами, все что необходимо декодеру — это список длин кодов символов. Учитывая, что в среднем длину одного кода для n-символьного алфавита можно закодировать $[\log_2(\log_2 n)]$ битами, получим очень эффективный алгоритм.

ЛАБОРАТОРНАЯ РАБОТА № 7
Кодирование информации. Компрессия. Метод Хаффмана

Продолжительность — 4 часа.
Максимальный рейтинг — 7 баллов.

ЦЕЛЬ РАБОТЫ

Целью работы является изучение методов обратимой компрессии на основе кода Хаффмана.

ЗАДАНИЕ

1. Изучить методы сжатия. Разобраться с понятиями: обратимая\необратимая компрессия; сжатие с потерями\без потерь; словарные\статистические методы архивации; степень сжатия, скорость сжатия, коэффициент сжатия; потеря качества при необратимой компрессии; оценки потери качества.

2. Изучить метод обратимой компрессии Хаффмана. Выполнить индивидуальное задание соответствующее вашему варианту. (Вариант определяет преподаватель).

3. Для каждого символа Si кодируемого сообщения S определить его статистическую вероятность Pi — (частоту появления этого символа в сообщении S).

4. Построить дерево Хаффмана для полученного алфавита {Si, Pi}.

5. Построить неканонический код Хаффмана {Si, Hi} занумеровав символами «0» и «1» ребра дерева Хаффмана и построив таким образом соответствие между символами Si и кодовыми комбинациями Hi.

6. Закодировать сообщение S полученным кодом {Si, Hi}, полученную кодовую последовательность H привести в отчете.

7. Декодировать полученную последовательность H и в отчете привести результат декодирования — символьную последовательность Q. Будет ли она всегда точно соответствовать исходному сообщению S?

8. Оценить степень сжатия полученного кода Хаффмана {Si, Hi} и сравнить его по этому показателю с кодом ASCII.

9. Подготовить отчет, включающий в себя: исходную (S) и закодированную (H) последовательности, дерево Хаффмана с весами узлов {Si, Pi}, код Хаффмана {Si, Hi}, декодированное сообщение Q. В отчете обязательно привести рассчет коэффициента сжатия полученного кода. Выполнить свой (!) индивидуальный вариант задания.

ВАРИАНТЫ ИНДИВИДУАЛЬНЫХ ЗАДАНИЙ

№ варианта	Последовательность S
1	qwyiweqriiwryiuruquitqeerwewrwuyrqiuruquitqeerwewrwuyrq
2	stnnrrtopsqprrpqmqrmmmrpqrssomoqtsqppmrpqrssomoqtsq
3	fhfheggcfbhhbffehbfbghbdcbcfbgfagdbdcbcfbgfagd
4	utrqyyquiqieyryurtutrqyyquutrqyyquutrqyyqu
5	psmqtrnmttqrsnrsmtroorrnnmpospqmprrnrmpospqmprrnr
6	egfbcghfbacfhehabgefadfddafaabacfhehabgebacfhehabgebfhehab
7	ytqtwuuwtyrrquiiwyywieieyqeuwtyrrquiiwyywieiey
8	optntptqpmomrnnnnopttpnottqtostqrorpttpnnnnnnottqtostnnnqror
9	geaggdffgacdhcgccbfccgcgedhddebef
10	twreiwiriyitqryrewtuwqeiituieetyrewtuwqeiituieetyrewtuwqeiiiiii
11	ntostrntorpopnrspmtmmmprpnomqnnptoosqnnptoosqnnptoos
12	cefeeghcdegaaahgbhbggaaahfeghcdegaaahgbhbggaaahf
13	wqerityqweeuirttiiwyyyqyreeuirttiiwyyyqyreeuirttiiwyyyqyr
14	nqqqomsmtmrsrqrsssrnnsotmtmoqoqr
15	dbgeecfehaahgbcghdcbfbhccfehaahgbcghdcbfbhc
16	yqtqwwteeyitteertereqryurrqettrweyywwt
17	qmqqomsqooprrtrrpttsoroporsrprpttsoroporsrprpttsoroporsrp
18	afbbddeehaghfbdbfgcgeaeafbebeghcgeaeafbebeghcgeaeafbebegh
19	wriqtutewtqqiuyirqyyewuqrqrrireirqyyewuqrqrreirqyyewuqrqrri
20	tpmssnmrppqsrqtmmonspmmoqsqomqsqrnnotno
21	bfdaffcfdgcfdgdgbefggahfcegbbbddaha
22	yiuewtryiiwrrruyiwyuqurieryeryyrwtiu
23	qoosmstnpsmsrtrtrmmmpnnqtqrqmqssnprmo
24	dceeffbaagebdeaddhfgeffbaagebdeaddhfgeffbaagebdeaddhfg
25	weeiuureyywtuewqquuiqywtuuurwewqquuiqywtuuurw
26	mqrntotrnprmotmtmtpqppqomnnssotmtmtpqppqomnnss
27	ccadefhgfecgcgedehhhcebeahddfbaehdfafb
28	rwiewuieuryeierreuitquiwtewqquwuyiwrrwtewqquwuyiwrr
29	oontttqponntppsnrntqspnomqsonntppsnrntqspnomqs
30	hbhagfgdgdagaebbecegdbegcefgdgdagaebbecegdbegce

ПРИМЕЧАНИЯ

1. Дополнительные 10 баллов выставляются студенту, написавшему программу, реализующую данную лабораторную работу.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Охарактеризовать методы необратимой компресии. Привести примеры.
2. Дать определение методам сжатия без потерь. Примеры.
3. Чем отличаются словарные алгоритмы сжатия от статистических методов компресии?
4. Каким (обратимым\необратимым, словарным\статистическим) методом компресии является метод Хаффмана? Почему?
5. Как вычисляется коэффициент сжатия? Что он характеризует?
6. Каким (обратимым\необратимым, словарным\статистическим) методом компресии является метод JPEG? Почему?
7. Объясните суть арифметического метода кодирования. В чем достоинства и недостатки этого алгоритма?
8. Каким (обратимым\необратимым, словарным\статистическим) методом компресии является метод RLE? Почему?
9. Расскажите о принципе архивации по алгоритму Лемпеля-Зива.
10. Каким (обратимым\необратимым, словарным\статистическим) методом компресии является метод LZW? Почему?
11. Как происходит сжатие по методу Шеннона-Фано? Какие у этого метода недостатки? Достоинства?
12. Каким (обратимым\необратимым, словарным\статистическим) методом компресии является метод MPEG? Почему?
13. Расскажите об известных критериях оценки потерь качества при необратимой компрессии.
14. Каким (обратимым\необратимым, словарным\статистическим) методом компресии является метод Шеннона? Почему?

8 АКТИВНОЕ СЕТЕВОЕ ОБОРУДОВАНИЕ. КОММУТАТОР

Общие сведения

Работа коммутатора основана на определении ***MAC-адреса*** компьютера, подключенного к порту коммутатора. При включении коммутатора в локальную сеть и передаче какой-либо рабочей станцией информационного кадра коммутатор анализирует содержащееся в нем поле MAC-адреса узла-источника и заполняет на основе этой информации таблицу соответствия порт-MAC-адрес сетевого компьютера. Впоследствии, при получении нового кадра, коммутатор ищет в таблице MAC-адресов адрес, совпадающий с адресом в поле узла-назначения, и передает кадр только на тот порт, к которому подключена соответствующая рабочая станция. Таким образом, вся работа коммутатора сосредоточена на 2 уровне модели OSI — уровне передачи данных или, как его еще называют, на канальном уровне.

Если в локальной сети, построенной на концентраторах все рабочие станции находятся в одном, общем ***домене коллизий***, поскольку используют общую разделяемую среду доступа, то в сети, построенной на коммутаторах отдельный домен коллизий образуется каждым портом коммутатора. Таким образом коллизии в коммутируемой сети могут возникать только между сетевым интерфейсом рабочей станции и тем портом коммутатора, к которому эта станция подключена. Очевидно, что вероятность таких коллизий очень невысокая. Домен коллизий при этом уменьшается до сегмента кабеля, соединяющего компьютер и коммутатор, поэтому он называется микросегментом. А локальная сеть, построенная на коммутаторах — коммутируемой.

Коммутация может быть симметричной и асиметричной (иногда говорят: синхронной и асинхронной). При симметричной или синхронной коммутации все порты коммутатора работают на одной скорости. При асимметричной (асинхронной) коммутации некоторые порты коммутатора могут работать на одной скорости, например, 10 Мбит/с, а другие на более высокой скорости — 100 Мбит/с или 1000 Мбит/с. Очевидно, что для соединения между портами, работающими на разных скоростях применяется

буферизация, т.е. коммутаторы, поддерживающие асинхронный режим имеют собственную память для временного хранения данных. Асинхронная коммутация позволяет подключать к скоростным каналам большое число устройств с более низким быстродействием.

Одним из способов повышения пропускной способности сетевого соединения является механизм, основанный на протоколе ***IEEE 802.3ad LACP*** (***Link Aggregation Control Protocol***) и носящий название ***Aggregated Link***. Группирование кабельных линий позволяет увеличить пропускную способность соединения (обычно магистрального) во столько раз, сколько линий объединяется.

Порты современного коммутатора поддерживают несколько режимов работы. Различают дуплексный режим — ***duplex*** или ***full duplex***, в котором порт одновременно работает и на прием и на передачу данных. Такой режим возможен только тогда, когда порт «умеет» вычитать из получаемого сигнала передаваемый. Скорость передачи при этом удваивается. Дуплексный режим разрешается если оба интерфейса с двух сторон кабельной линии его поддерживают. Другими словами, этот режим должен поддерживаться не только портами коммутатора, но и сетевыми адаптерами рабочих станций. В полудуплексном режиме (***half duplex***) передача и прием осуществляются попеременно. Для подключения некоторых типов сетевого оборудования (например, для соединения нескольких коммутаторов между собой) требуется изменение порядка соединений выводов порта. Такое изменение достигается применением специального сетевого кабеля с перекрестным соединением выводов — ***crossover,*** вместо обычного типа кабеля с прямыми соединениями — ***straight-through***. Аналогичное изменение коммутации выводов может быть получено с помощью электронной схемы внутри коммутатора, при этом порт переключается в режим ***MDIX*** (***MDI crossover***) вместо обычного ***MDI*** (***medium dependent interface***). В таком случае отпадает необходимость использовать специальные сетевые кабели. Как правило, порт любого современного коммутатора может автоматически определить тип подключенного оборудования и переключиться в режим MDIX, если это необходимо. Однако, такое автоматическое определение режима работы

порта возможно только при активизации на нем конфигурационного параметра ***Auto Negotiation***. При активизации этого параметра автоматически выбирается необходимая скорость работы (обычно поддерживаются несколько скоростей передачи данных), дуплексный или полудуплексный режим работы и схема соединения выводов MDI или MDIX.

Методы коммутации, применяемые в коммутаторах

Для коммутации сетевых кадров могут использоваться различные методы. По способу обработки кадров их можно разделить на две группы:

1. Store and forward — метод в котором сетевой кадр полностью сохраняется в буферной памяти коммутатора и уже затем направляется по адресу назначения. При использовании такого метода коммутатор может проверить полученный кадр на целостность и отсутствие ошибок, применить фильтрацию, если она задана и т.п. Надежность передачи по сети информации при использовании этого метода возрастает. Только при использовании метода **Store and forward** возможна асимметричная коммутация. Недостатком метода **Store and forward** является увеличение времени задержки, вносимое коммутатором в общее время транспортировки кадра по сети. При увеличении размера передаваемых пакетов задержки, вносимые коммутатором, тоже возрастают.

2. Cut-through — метод, при использовании которого кадр начинает передаваться по адресу назначения еще до того, как он полностью получен. Этот метод снижает задержки, вносимые коммутатором, но и возможности обнаружения ошибок при передаче кадров также уменьшаются. Другим недостатком этого метода коммутации является то, что он может применяться только при симметричной коммутации, когда источник и приемник сообщения работают на одной частоте. Метод **Cut-through** реализуется двумя способами:

2.1. Fast-forward — Эта разновидность метода **Cut-through** нацелена на достижение максимально малой задержки при передаче кадра. Кадр начинает передаваться по адресу назначения как только в принимаемом кадре этот адрес прочитан. Так как адрес назначения содержится в самом начале кадра, задержка, вноси-

мая коммутатором, существенно уменьшается. Именно этот метод коммутации применяется в коммутаторах по умолчанию. Искаженный, по каким-либо причинам, кадр не будет отброшен коммутатором, т.к. он просто не проверяется на отсутствие ошибок. Он будет передан по адресу назначения и только при его получении станцией назначения кадр будет проверен, отброшен и будет затребована повторная передача этого кадра. В методе **Fast-forward** задержка измеряется временем между первым принятым битом и первым переданным битом и составляет 14 байт.

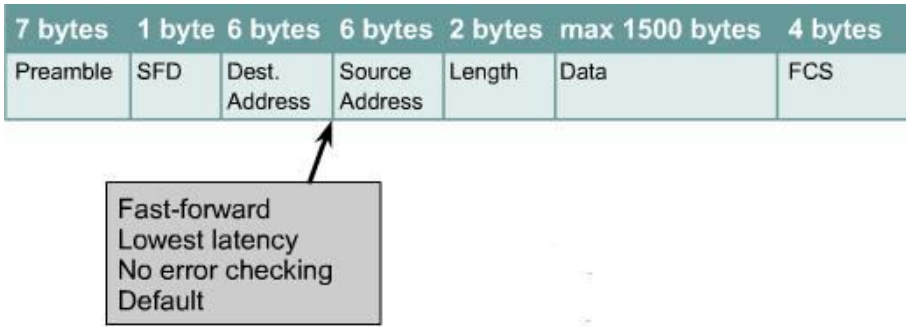


Рис. 17 — Величина задержки, вносимой коммутатором при передаче кадра при использовании метода Fast-forward

2.2. Fragmet-free — Этот метод основан на том, что в буферной памяти коммутатора сохраняются первые 64 байта кадра. Это минимальный размер кадра (без преамбулы) в соответствии со стандартом Ethernet. Кадры меньшего размера, как правило, — результат возникновения коллизий в сети Ethernet, если используется среда общего доступа. Все эти кадры, очевидно ошибочные, не передаются коммутатором, а проверяются и запрашивается их повторная передача. Таким образом устраняется львиная доля ошибочных кадров и количество переданных коммутатором кадров, содержащих ошибки, значительно сокращается. Однако в полностью коммутируемой сети этот метод не дает никакого выигрыша.

На рис. 18 представлен сравнительный вид задержек при различных методах коммутации. Применяются и более сложные методы коммутации, представляющие собой некоторую комбинацию уже перечисленных методов. Например, метод ***Adaptive switching*** работает по алгоритму ***Fast-forward*** пока количество ошибочно переданных кадров не достигнет некоторого заранее

установленного предела, после чего переключается на алгоритм ***Store and forward***.

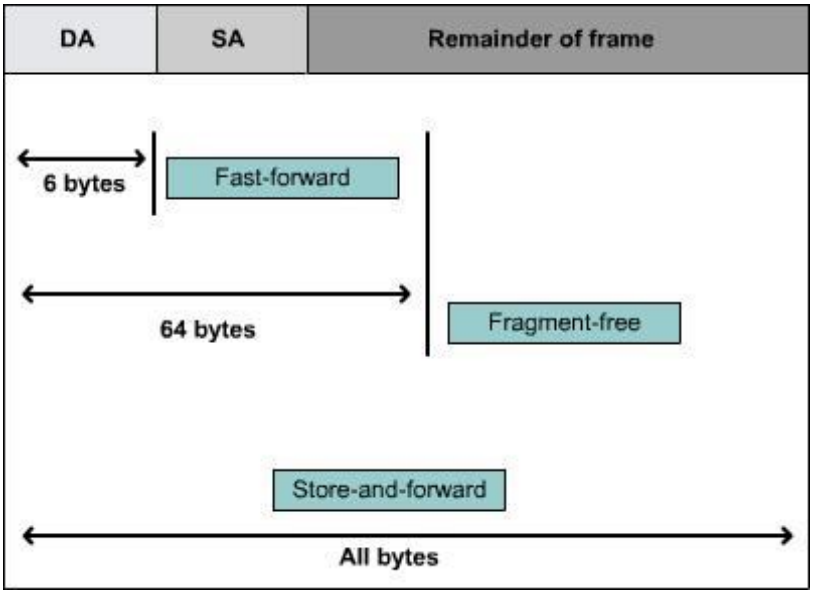


Рис. 18 — Сравнение величины задержки
при использовании различных методов коммутации

Проблемы коммутируемых локальной сетей и методы их решения

В локальной сети передаются не только сообщения, направляемые конкретному компьютеру-рабочей станции, но и групповые, и широковещательные сообщения (рис. 19). Широковещательные сообщения, полученные на один из портов коммутатора передаются на все остальные порты.

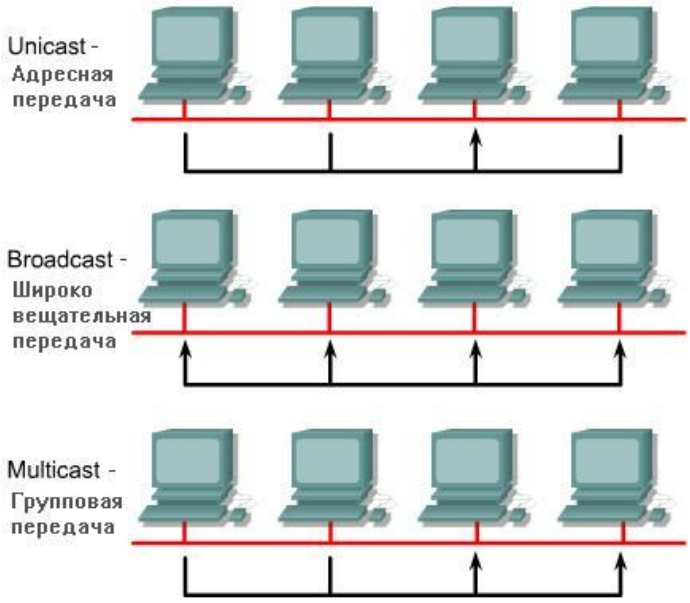


Рис. 19 — Типы сообщений, передаваемых в локальной сети

Если в локальной сети несколько коммутаторов и для повышения надежности используются избыточные соединения между ними, то может возникнуть ситуация получившая название «широковещательный шторм» — ***broadcast storm*** (рис. 20). Получив широковещательное сообщение от одного из узлов сети, коммутаторы будут рассылать его снова и снова. Непрерывная рассылка широковещательных сообщений может полностью блокировать работу сети.

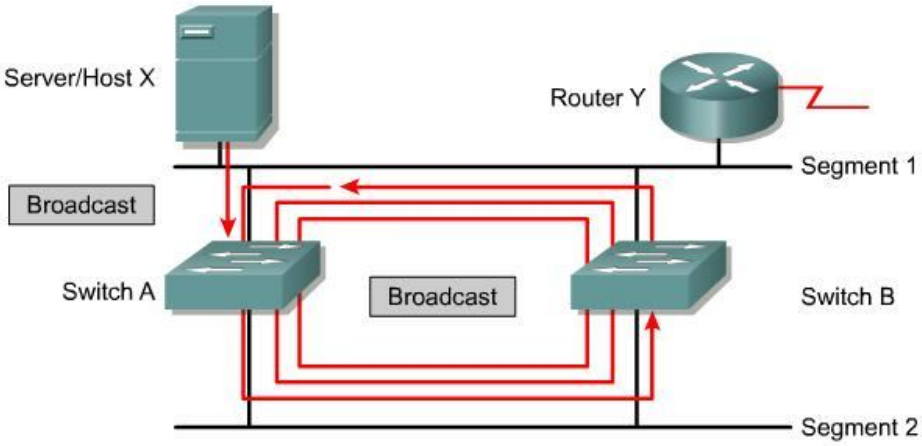


Рис. 20 — Ситуация возникновения широковещательного шторма в локальной сети

Чтобы избежать такой ситуации несколько коммутаторов в сети автоматически создают логическую структуру, свободную от замкнутых петель. Для создания такой структуры используется определенный стандартом ***IEEE 802.1D*** протокол ***STP*** (Spanning-Tree Protocol) — протокол покрывающего дерева. С помощью этого протокола в сети определяется «главный» коммутатор, всем остальным коммутаторам назначаются уникальные идентификаторы и все сетевые маршруты ранжируются по «весу» в зависимости от скорости передачи данных. Чем скорость выше, тем меньший вес имеет маршрут. Все коммутаторы в сети каждые 2 сек. рассылают служебные пакеты ***BPDU*** (Bridge Protocol Data Unit), содержащие идентификатор коммутатора, идентификатор главного коммутатора, идентификатор порта и вес маршрута.

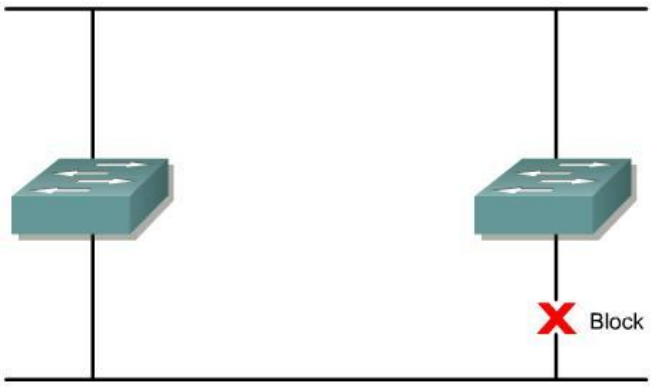


Рис. 21 — Блокирование петлевых соединений
при создании логической структуры «дерева»

Избыточные маршруты, приводящие к петлевым соединениям блокируются для передачи данных (рис. 21), но принимают пакеты BPDU. Как только изменится конфигурация сети, эти порты также будут участвовали в новой процедуре построения логической структуры «дерева».

Виртуальные локальные сети VLAN

Другим способом ограничения широковещательного трафика в коммутируемых сетях является создание виртуальных локальных сетей — ***VLAN***, основанных на использовании протокола ***IEEE 802.1Q***. Виртуальные локальные сети — удобный и гибкий механизм для создания логической структуры в локальной сети. Применение коммутаторов дает возможность объединять в локальной сети группы компьютеров по таким признакам, например, как характер работы, принадлежность к одному подразделению предприятия и т.п. Несмотря на то, что это логически созданная структура, трафик локализуется внутри виртуальной локальной сети, принадлежность к которой задается на уровне портов коммутатора. Компьютеры, подключенные к портам, которые не входят в виртуальную локальную сеть, не смогут получить к ней доступ. Сама процедура изменения конфигурации коммутатора и привязка его портов к той или иной локальной сети выполняется значительно проще настройки маршрутизатора и создания реальных, а не виртуальных подсетей.

Каждая виртуальная сеть имеет имя, носящее описательный характер и идентификатор или номер сети. Идентификатор сети

переносится в пакетах IEEE 802.1Q, если используется механизм «маркировки» (tagged) этих пакетов. По умолчанию в коммутаторе всегда присутствует виртуальная сеть с номером 1 — VLAN 1 или Default VLAN. Удалить эту виртуальную локальную сеть невозможно. Все порты коммутатора исходно находятся в этой виртуальной сети.

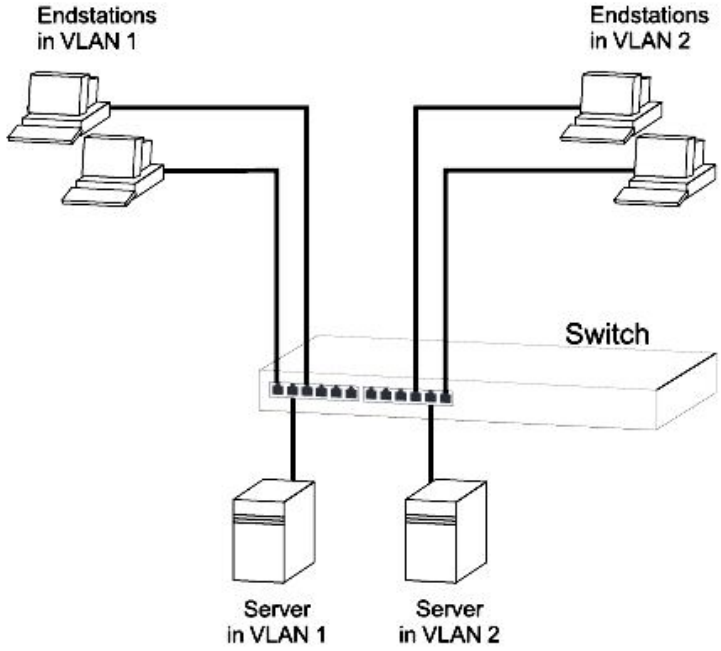


Рис 22 — Создание двух VLAN на базе одного коммутатора

Предположим, что виртуальные локальные сети создаются в небольшой локальной сети, содержащей только один коммутатор, к портам которого подключены рабочие станции. Создаваемые виртуальные сети должны изолировать трафик между группами рабочих станций. В этом случае любой порт коммутатора будет принадлежать только одной VLAN. Тогда необходимо использовать механизм привязки портов к создаваемым VLAN ***без маркировки — untagged*** (рис. 22).

В случае примера, приведенного на рис. 22, должна быть создана, очевидно, только VLAN 2, т.к. VLAN 1 уже присутствует в коммутаторе. После чего в эту созданную виртуальную сеть должны быть (без маркировки) включены определенные порты коммутатора. Немаркированный порт может быть членом только одной виртуальной сети, поэтому порты, включенные в состав

VLAN 2, автоматически исчезнут из состава VLAN 1 и виртуальные сети будут изолированы.

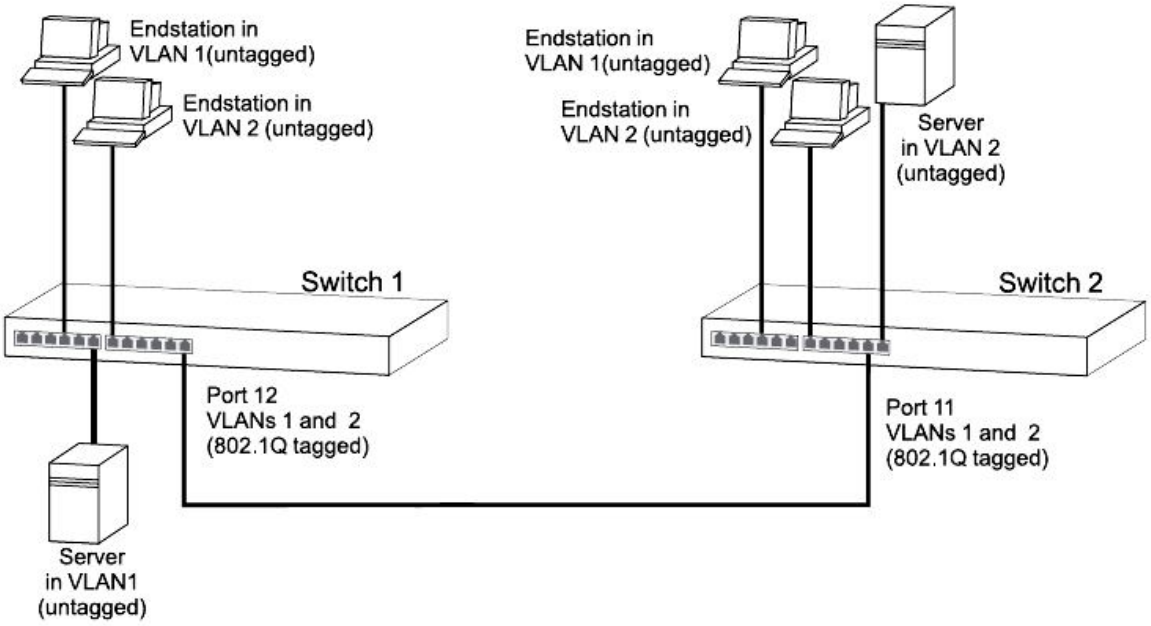


Рис. 23 — По одному физическому кабелю может перноситься трафик нескольких виртуальных сетей

Маркированные (***tagged***) порты могут быть включены в состав нескольких виртуальных локальных сетей. Этот механизм используется в более сложных локальных сетях для переноса трафика нескольких виртуальных сетей по одному физическому кабелю. Для того чтобы осуществить такой перенос, с обеих сторон кабеля должно работать устройство, поддерживающее стандарт IEEE 802.1Q. Другими словами, через маркированные (tagged) порты соединяются между собой два коммутатора (рис. 23).

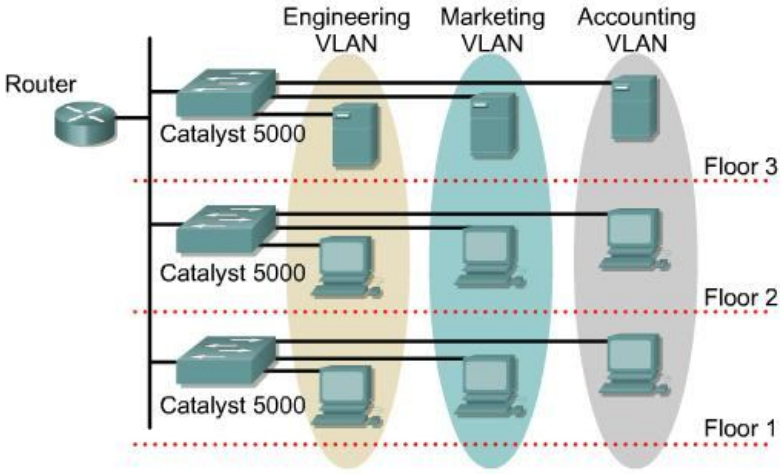


Рис. 24 — Для организации связи между несколькими VLAN необходим маршрутизатор

При этом маркированный порт одновременно входит в состав нескольких виртуальных локальных сетей. Конечные же станции всегда подключаются через немаркированные порты.

Для организации соединения между виртуальными локальными сетями необходимо оборудование, работающее на сетевом уровне (3 уровень модели OSI) — ***маршрутизатор*** (***router***) (рис. 24) или специальный коммутатор, поддерживающий функции маршрутизации. Такие коммутаторы называются коммутаторами 3-го уровня.

Настройка конфигурации коммутатора 3COM SuperStack 4226T

Коммутатор ***3COM SuperStack 4226T*** имеет 24 порта, способные работать на скорости 10/100 Мбит/с по витой паре категории 5 и два порта, поддерживающие, кроме того, гигабитную скорость на таком же кабеле (рис. 25).

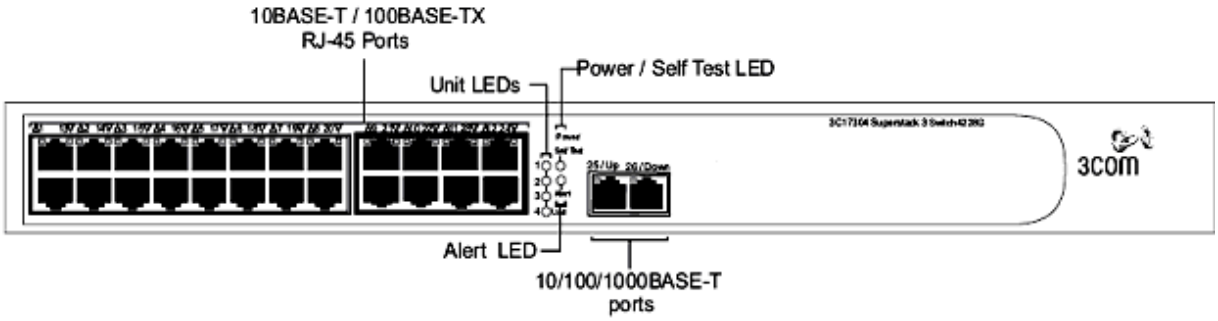


Рис. 25 — Внешний вид коммутатора 3COM SuperStack 4226T

Управление коммутатором может осуществляется при подключении с компьютера — рабочей станции через программу терминального доступа (например ***Telnet*** или ***HyperTerminal***, входящие в состав ОС Windows), через веб-браузер или с помощью протокола ***SNMP*** и специального программного обеспечения, например, поставляемой вместе с коммутатором программой ***3Com Network Supervisor***. Подключение может осуществляться как через последовательный порт, так и через локальную сеть по протоколу TCP/IP (очевидно, что программа веб-браузера поддерживает только соединение по протоколу TCP/IP).

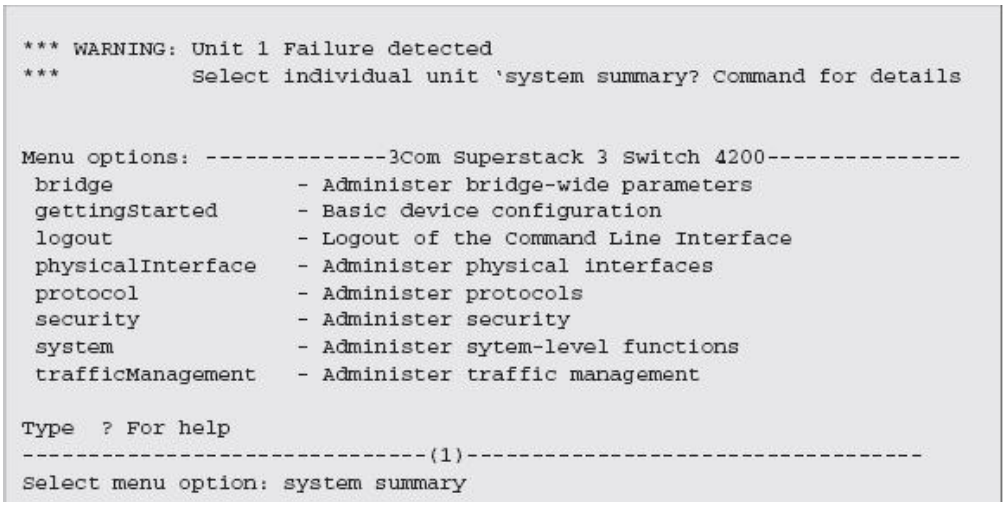


Рис. 26 — Главное меню коммутатора

Настройки подключения при соединении через последовательный порт:

- Скорость передачи данных 19200 бод.
- Группировка бит данных — по 8 бит.
- Четность — нет.
- Стоповые биты — 1.
- Управление потоком — нет.

Рассмотрим кратко систему команд коммутатора, при этом будем основываться на режиме командной строки, используемом при подключении к коммутатору программой терминального доступа. При работе через веб-браузер отображение командного меню в целом повторяет режим командной строки, однако сами команды задаются нажатием клавиши мышки и выбором в активизируемых окнах конфигурации. Главное меню коммутатора (рис. 26) содержит команды, перечисленные в таблице:

КОМАНДА	ОПИСАНИЕ
Bridge	Эта команда содержит основные «рабочие» подкоманды управления настройками коммутатора, настройка фильтров, создание виртуальных сетей и т.п.
gettingStarted	Команда запускает мастер, автоматически предлагающий вариант первичной настройки коммутатора. Выполнять эту команду на коммутаторе уже работающем в сети не ре-

	комендуется
Logout	Команда завершения сеанса связи с коммутатором
physicalInterface	Команда настройки параметров Ethernet портов коммутатора
Protocol	Команда настройки протокола IP
Security	Команда настройки режима безопасности и изменения пароля пользователя
system	Команда содержит системные подкоманды конфигурации коммутатора, например задание имени коммутатора, задание автоматических действий происходящих при определенных условиях и т.п.
trafficManagement	Команда позволяет настраивать параметры приоритетов для трафика различного типа (Quality of Service)

Общая информация о системе команд

Введение команды с последующим нажатием клавиши Enter переводит в подменю соответствующей команды. При этом команду не обязательно вводить полностью, достаточно ввести несколько начальных символов. Например, команда **br** <Enter> переведет вас в подменю команды ***bridge,*** отобразив доступные в нем подкоманды. Команда **br vl** <Enter> переведет в подменю команды ***bridge vlan***, а команда **br vl sum**, аналогичная команде ***bridge vlan summary,*** выведет информацию о созданных на коммутаторе виртуальных сетях. Список подкоманд, доступных в той или иной команде и небольшую подсказку по формату команды можно вывести с помощью символа **?** в конце команды, например **br vl ?** Перейти в предыдущее меню можно с помощью команды quit, в главное меню из меню любого уровня можно попасть, нажав клавишу Escape или стрелку «вверх» на клавишах управления курсором.

Таблица MAC-адресов и связанных с ними портов коммутатора может быть выведена по команде ***bridge addressDatabase summary***. Заполнение этой таблицы выполняется несколькими способами:

1. В режиме ***Learned*** коммутатор заполняет таблицу MAC-адресов при получении кадров от компьютеров-рабочих станций. В этом режиме записи в таблице храняться только в течение определенного времени (***aging time***) и автоматически удаляются, если коммутатор больше не получает кадров от рабочей станции. Это позволяет коммутатору постоянно иметь актуальную базу данных адресов. Все записи, созданные в режиме Learned, удаляются из таблицы MAC-адресов при перезагрузке или выключении коммутатора.

2. Если параметр «время жизни» установлен в 0, то запись в таблице MAC-адресов больше не обновляется и будет хранится до следующей перезагрузки коммутатора.

3. Наконец, при внесении записи в таблицу MAC-адресов вручную, с помощью команды ***bridge addressDatabase add***, она приобретает статус постоянной (***Permanent***) и может быть удалена из нее также только вручную командой ***bridge addressDatabase remove***.

Коммутатор позволяет защитить локальную сеть, введя ограничения на подключение к сети компьютеров по их MAC-адресам. Команда ***Security Network Access Port Security*** запрещает подключение неразрешенных устройств.

ЛАБОРАТОРНАЯ РАБОТА № 8
Настройка конфигурации коммутатора 3COM SuperStack 4226T

Продолжительность — 4 часа.
Максимальный рейтинг — 8 баллов.

ЦЕЛЬ РАБОТЫ

Целью работы является ознакомление со способами настройки конфигурации активного сетевого оборудования на примере коммутатора 3COM SuperStack 4226T. Изучение способов терминального подключения к сетевому оборудованию. Приобретение навыков создания виртуальных локальных сетей (VLAN).

ЗАДАНИЕ

Лабораторная работа выполняется группой из 2—3 человек после допуска у преподавателя. Группы работают поочередно друг за другом.

№	Описание
1	1. Убедится в том, что все компьютеры локальной сети отображаются в сетевом окружении. 2. Создать две VLAN с именами 338 CLASS и 338 GPO, включив в них по две рабочие станции — в 338 CLASS ws338-02 и ws338-03, а в 338 GPO ws338-06 и ws338-10. 3. Проверить с помощью команды ping и через сетевое окружение, что трафик локализован внутри созданных виртуальных сетей. 4. Удалить созданные виртуальные сети, перенести все порты коммутатора в виртуальную сеть по умолчанию. 5. Повторить пункт 1 задания
2	1. Убедится в том, что все компьютеры локальной сети отображаются в сетевом окружении. 2. Создать три VLAN с именами VLAN 1, Second и Third, включив в них по три рабочие станции — в VLAN 1 ws338-01, ws338-02 и ws338-03, в Second ws338-04, ws338-05 и ws338-06,

№	Описание
	в Third ws338-09, ws338-10 и ws338-11. 3. Проверить с помощью команды ping и через сетевое окружение, что трафик локализован внутри созданных виртуальных сетей. 4. Удалить созданные виртуальные сети, перенести все порты коммутатора в виртуальную сеть по умолчанию. 5. Повторить пункт 1 задания
3	1. Убедится в том, что все компьютеры локальной сети отображаются в сетевом окружении. 2. Создать две VLAN с именами TUSUR и PromElectronics, включив в них: a. в TUSUR три рабочих станции — ws338-02, ws338-03 и ws338-10; b. в PromElectronics четыре рабочих станции — ws338-01, ws338-04, ws338-05 и ws338-06. 3. Проверить с помощью команды ping и через сетевое окружение, что трафик локализован внутри созданных виртуальных сетей. 4. Удалить созданные виртуальные сети, перенести все порты коммутатора в виртуальную сеть по умолчанию. 5. Повторить пункт 1 задания
4	1. Убедится в том, что все компьютеры локальной сети отображаются в сетевом окружении. 2. Создать четыре VLAN с именами VLAN 1, VLAN 2, VLAN 3 и VLAN 4, включив в VLAN 2 — VLAN 4 по две рабочие станции, а в VLAN 1 — четыре рабочих станции: a. VLAN 1 ws338-01 — ws338-04; b. VLAN 2 ws338-05 и ws338-06; c. VLAN 3 ws338-08 и ws338-09; d. VLAN 4 ws338-10 и ws338-11. 3. Проверить с помощью команды ping и через сетевое окружение, что трафик локализован внутри созданных виртуальных сетей. 4. Удалить созданные виртуальные сети, перенести все порты коммутатора в виртуальную сеть по умолчанию. 5. Повторить пункт 1 задания

№	Описание
5	1. Убедится в том, что все компьютеры локальной сети отображаются в сетевом окружении. 2. Создать три VLAN с именами TUSUR 1, TUSUR 2 и TUSUR 3, включив в них a. TUSUR 1 ws338-01, ws338-02 и ws338-03; b. TUSUR 2 ws338-05 и ws338-10; c. TUSUR 3 ws338-06 и ws338-11. 3. Проверить с помощью команды ping и через сетевое окружение, что трафик локализован внутри созданных виртуальных сетей. 4. Удалить созданные виртуальные сети, перенести все порты коммутатора в виртуальную сеть по умолчанию. 5. Повторить пункт 1 задания
6	1. Убедится в том, что все компьютеры локальной сети отображаются в сетевом окружении. 2. Создать две VLAN с именами TIME и UPTICK, включив в них a. TIME ws338-01, ws338-05, ws338-10; b. UPTICK ws338-06 и ws338-09. 3. Проверить с помощью команды ping и через сетевое окружение, что трафик локализован внутри созданных виртуальных сетей. 4. Удалить созданные виртуальные сети, перенести все порты коммутатора в виртуальную сеть по умолчанию. 5. Повторить пункт 1 задания
7	1. Убедится в том, что все компьютеры локальной сети отображаются в сетевом окружении. 2. Создать две VLAN с именами RULE и 3COM, включив в них: a. RULE ws338-02 и ws338-10; b. 3COM ws338-06 и ws338-3. 3. Проверить с помощью команды ping и через сетевое окружение, что трафик локализован внутри созданных виртуальных сетей. 4. Удалить созданные виртуальные сети, перенести все порты коммутатора в виртуальную сеть по умолчанию

№	Описание
	5. Повторить пункт 1 задания
8	1. Убедится в том, что все компьютеры локальной сети отображаются в сетевом окружении. 2. Создать три VLAN с именами SWITCH, SWAB и SPACE, включив в них a. SWITCH ws338-01, ws338-03, ws338-11; b. SWAB ws338-05 и ws338-04; c. SPACE ws338-06, ws338-08 и ws338-09. 3. Проверить с помощью команды ping и через сетевое окружение, что трафик локализован внутри созданных виртуальных сетей. 4. Удалить созданные виртуальные сети, перенести все порты коммутатора в виртуальную сеть по умолчанию. 5. Повторить пункт 1 задания
9	1. Убедится в том, что все компьютеры локальной сети отображаются в сетевом окружении. 2. Создать одну VLAN с именем NETWORKING, включить в нее рабочие станции ws338-08, ws338-09 и ws338-10. 3. Проверить с помощью команды ping и через сетевое окружение, что трафик локализован внутри созданной виртуальной сети и внутри виртуальной сети по умолчанию. 4. Удалить созданную виртуальную сеть, перенести все порты коммутатора в виртуальную сеть по умолчанию. 5. Повторить пункт 1 задания
10	1. Убедится в том, что все компьютеры локальной сети отображаются в сетевом окружении. 2. Создать две VLAN с именами ETHERNET и MANAGEMENT, включив в них a. ETHERNET все рабочие станции с ws338-02 по ws338-06; b. MANAGEMENT ws338-10 и ws338-11. 3. Проверить с помощью команды ping и через сетевое окружение, что трафик локализован внутри созданных виртуальных сетей. 4. Удалить созданные виртуальные сети, перенести все порты коммутатора в виртуальную сеть по умолчанию. 5. Повторить пункт 1 задания

ПРИМЕЧАНИЯ

1. MAC-адрес удаленной рабочей станции в локальной сети можно посмотреть в режиме командной строки с помощью команды ***arp eth_addr IP-адрес***. После выполнения этой команды искомый адрес будет занесен в локальную таблицу MAC-адресов, просмотреть которую можно командой ***arp -a*** или ***arp -g***.

2. Определить IP-адрес удаленной рабочей станции можно с помощью команды ***ping***, указав в качестве параметра ***NetBIOS_имя*** удаленного компьютера.

3. При работе с программой терминального доступа по протоколу TCP/IP полезно включить поддержку завершения ввода автоматической передачей символов перевода строки. В программе ***Microsoft Telnet*** это команда: *SET CRLF*. В программе ***HyperTerminal*** в меню Файл/Свойства (File/Properties) вкладка Дополнительно(Settings), клавиша «настройка ASCII (ASCII Setup)» необходимо установить флажок (***Send line ends with line feeds***). В этом случае программа терминального доступа не будет «проглатывать» введенные с клавиатуры символы.

4. Для подключения к коммутатору необходимо пройти авторизацию. Предусмотрено несколько уровней доступа с различными возможностями по изменению конфигурации коммутатора. По умолчанию самыми ограниченными возможностями обладает пользователь monitor, заводской пароль для этого пользователя совпадает с его именем — monitor. Пользователь ***manager*** имеет право изменять все конфигурационные параметры коммутатора, кроме настройки некоторых параметров безопасности, например, он не может создать новых пользователей. Пароль по умолчанию для этого пользователя также совпадает с его именем.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое дуплексный режим работы?

2. Какой стандарт позволяет объединить несколько кабельных линий для увеличения пропускной способности?

3. Перечислите известные вам режимы коммутации, поясните их особенности.

4. Поясните назначение кабеля типа crossover, в каких случаях его применение не обязательно.

5. Какой стандарт описывает протокол STP, поясните назначение этого протокола.

6. Что такое виртуальная локальная сеть, какой стандарт описывает работу таких сетей?

7. Может ли имя виртуальной сети содержать пробелы?

8. Поясните назначение механизма маркировки портов коммутатора при включении их в виртуальную сеть.

9. Как осуществить передачу информации между двумя виртуальными локальными сетями?

10. Как заполняется таблица MAC-адресов в коммутаторе?

9 АДРЕСАЦИЯ КОМПЬЮТЕРНЫХ СЕТЕЙ. ПОДСЕТИ TCP/IP

Как известно, адрес IP состоит из 4х октетов, разделенных точками, и представляется в формате 200.200.200.5. Однако этот адрес сам по себе недостаточен и требует маски подсети для того, чтобы показать, какая часть IP-адреса является идентификатором сети (***Network ID***), а какая — идентификатором хоста (***Host ID***).

Смысл маски подсети (***Subnet Mask***), например, 255.255.255.0 легко понять, записав ее октеты в двоичном виде: 11111111.11111111.11111111.00000000, те биты IP-адреса, которым соответствуют *единицы* маски относятся к *Network ID*, а биты, помеченные *нулями* в маске подсети — *Host ID*. Например, для адреса 200.200.200.5 и маски подсети 255.255.255.0 идентификатор сети будет 200.200.200.0, а хост будет иметь идентификатор 0.0.0.5:

IP-address	11001000	.11001000	.11001000	.00000101
Subnet Mask	11111111	.11111111	.11111111	.00000000
Network ID	11001000	.11001000	.11001000	.00000000
Host ID	00000000	.00000000	.00000000	.00000101

Результат получается побитным AND между *IP-адресом* и маской подсети.

Интернет — глобальная сеть TCP/IP — это совокупность сетей, где обмен пакетами данных между сетями обеспечивают маршрутизаторы (***router***), не знающие впрочем, точного расположения узла. Из *IP-адреса* назначения пакета маршрутизатор узнает только, к какой сети принадлежит узел (*Network ID*), и чтобы доставить пакет в сеть узла назначения, он использует сведения, хранящиеся в своей таблице маршрутизации. Как только пакет доставлен в необходимую сеть, он доставляется в соответствующий узел (по *Host-ID* пакету назначается нужный *MAC-адрес*).

Иначе говоря, когда пакет с конечным адресом 192.168.123.132 доставляется в сеть 192.168.123.0 (из локальной подсети или удаленной сети), компьютер 0.0.0.132 получит его из сети и обработает.

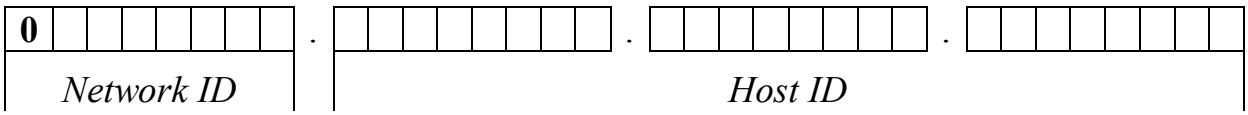
Необходимо отметить, что сетевые устройства с несколькими сетевыми интерфейсами (***multihomed***) имеют несколько *IP-адресов*: по одному на каждый интерфейс.

Так как каждый интерфейс, подключенный к сети, должен иметь уникальный адрес, встает вопрос распределения *IP-адресов* в глобальной сети Интернет. Этим занимается сетевой информационный центр (Internet Network Information Center или InterNIC) http://www.internic.net. InterNIC назначает только сетевые идентификаторы, назначением идентификаторов хостов в сети занимаются системные администраторы.

Существует три типа *IP-адресов:* персональный адрес (***unicast***) — указывает на один хост. Широковещательный адрес (***broadcast***) — указывает на все хосты в указанной подсети (*Host ID* состоит из одних единиц). Групповой адрес (***multicast***) указывает на группу хостов, принадлежащих к группе адресации (передача по протоколу IGMP).

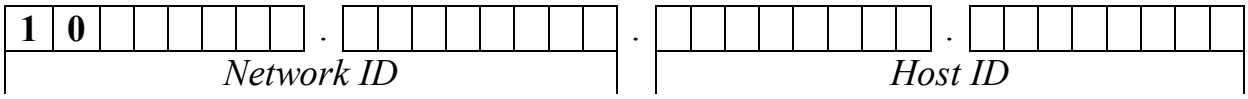
Классы сетей. *IP-адреса* распределяются организацией *InterNIC* по пяти классам. Наиболее распространены классы A, B и C, классы D и E существуют, но обычно не используются конечными пользователями. Каждый из классов адресов имеет свою маску подсети по умолчанию. Определить класс *IP-адреса* можно по его первому октету:

- Сети ***класса A*** имеют в первом октете значение от 0 до 127, что позволяет адресовать $2^7 - 1 = 127$ сетей и 16 777 216 узлов:



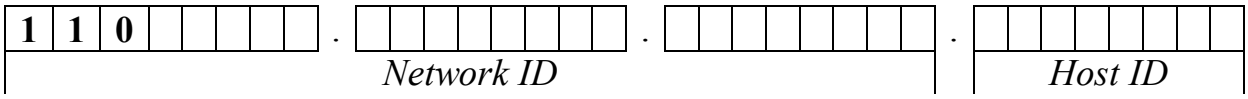
IP-адреса: 0.0.0.1 — 126.255.255.254
Subnet Mask: 255.0.0.0
Broadcast IP: 0-126.x.x.255

- Сети ***класса B*** имеют в первом октете значение от 128 до 191. Что определяет 14 бит для адреса сети и позволяет задавать адреса $2^{14} = 16\,384$ сетей и 65 535 хостов:



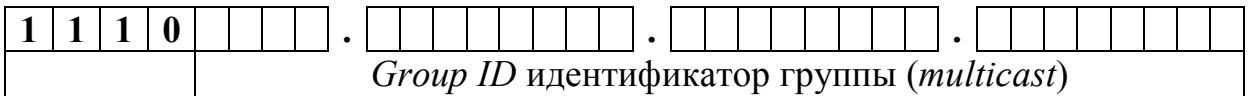
IP-адреса: 128.0.0.0 — 191.255.255.254
Subnet Mask: 255.255.0.0
Broadcast IP: 128-191.x.255.255

- Сети ***класса C*** имеют в первом октете значение от 192 до 223. В адресе сети класса C 21 бит отводится под адрес сети и 8 бит — под адрес хоста. Сетей этого класса может быть 2^{21} = = 2 097 152, а узлов — всего 254:



IP-адреса: 192.0.0.0 — 223.255.255.254
Subnet Mask: 255.255.255.0
Broadcast IP: 191-223.255.255.255

- Сети ***класса D*** (224.0.0.0 — 239.255.255.255) планировалось использовать для групповой адресации, они практически не используются, и имеют специфический формат:



- Сети ***класса E*** (240.0.0.0 — 247.255.255.255) зарезервированы и нигде не используются.

Итак, *IP-адрес* при помощи *Subnet Mask* разделяется на две части: адрес сети и адрес узла. Для упрощения записи маски подсети применяется префикс сети, показывающий, сколько бит в *IP-адресе* отводится под *Network ID*, следующие записи эквивалентны: 192.168.123.132 (255.255.255.0) и 192.168.123.132 /24.

В некоторых случаях значения маски подсети того или иного класса по умолчанию не соответствует потребностям организации из-за физической топологии сети или потому, что количество сетей (или узлов) не соответствует ограничениям маски подсети. Этого неудобства можно избежать, разделяя сети на подсети (***subnet***) с помощью масок подсети.

TCP/IP-сеть класса A, B или C может еще быть разбита на подсети системным администратором. Для этого можно «занять» несколько бит, из адреса узла и использовать их для адресации *подсети* в адресе. Например, использование маски 255.255.255.**192** (1111111.11111111.1111111.**11**000000) преобразует сеть 192.168.123.0 в четыре сети:

- сеть 192.168.123.0, с узлами из диапазона 192.168.123.1 — 62;
- сеть 192.168.123.64, с узлами 192.168.123.65 — 126;
- сеть 192.168.123.128, с узлами 192.168.123.129 — 190;
- сеть 192.168.123.192, с узлами 192.168.123.193 — 254.

Первые две цифры последнего октета становятся адресами сети, поэтому появляются дополнительные сети, в которых последние 6 двоичных цифр можно использовать в качестве адресов узлов. Маска подсети 255.255.255.192 позволяет создать четыре сети с 62-мя узлами в каждой.

Обратите внимание на следующие два адреса узлов: 192.168.123.71 и 192.168.123.133. Если использовать по умолчанию маску подсети класса C 255.255.255.0, оба адреса будут в сети 192.168.123.0. Однако если использовать маску подсети 255.255.255.192, они окажутся в разных сетях; 192.168.123.71 в сети 192.168.123.64, в то время как 192.168.123.133 в сети 192.168.123.128.

Не забывайте, что адреса узлов с одними только единицами (широковещательные адреса *broadcast*) и одними нолями (адрес сети или подсети *Network ID*) заняты, поэтому для адресации хостов нельзя использовать адреса со следующими числами в последнем октете: 0, 63, 64, 127, 128, 191, 192, или 255.

Связь между узлами из различных TCP/IP сетей осуществляется через маршрутизатор, его адрес называется ***основным шлюзом.*** Сопоставляя *Network ID* назначения (из передаваемого пакета) со своим *Network ID*, протокол TCP/IP определяет, отправлять данный пакеты на основной шлюз (который переправит его в нужную ЛВС) или узел назначения находится внутри собственной ЛВС.

Если в результате этого сопоставления назначением является локальный узел, то компьютер просто отправляет пакет в локальную подсеть (где передача осуществляется по *MAC-адресу*).

Если в результате сопоставления выясняется, что назначением является узел в удаленной сети, компьютер направляет пакет на основной шлюз, определенный в настройках протокола TCP/IP (пакету присваивается *IP-адрес* удаленного хоста, но *MAC-адрес* шлюза). Таким образом, именно маршрутизатор (или proxi-сервер) отвечает за отправку пакета в правильную подсеть, передавая его от маршрутизатора к маршрутизатору. После того, как пакет достиг ЛВС назначения (т.е. достиг основного шлюза с соответствующим *Network ID*), шлюз назначает ему *MAC-адрес* искомого хоста — и он доставляется внутри ЛВС.

ЛАБОРАТОРНАЯ РАБОТА № 9
Маршрутизация компьютерных сетей. Подсети TCP/IP

Продолжительность — 2 часа.
Максимальный рейтинг — 7 баллов.

ЦЕЛЬ РАБОТЫ

Изучить методы адресации в Интернет. Научиться разбивать сеть на подсети с корректным назначением диапазонов IP-адресов.

ЗАДАНИЕ

1. Освоить понятия: *IP-адрес*, доменные имена, *MAC-адрес*, маска подсети, префикс подсети, подсети, основные шлюзы.

2. Научиться определять класс IP-адреса, рассчитывать для него диапазон адресов, число сетей и число узлов.

3. Научиться различать *unicast*, *broadcast* и *multicast* адреса. Научиться записывать их для различных классов *IP-адресов*.

4. Для выбранного в соответствии с вариантом индивидуального задания *IP-адреса* определить класс сети, и для этого класса в отчете записать (в двоичном и в десятичном виде):

- диапазон адресов (минимальный и максимальный адрес хоста);
- broadcast адрес;
- вычислить число сетей данного класса;
- вычислить число хостов в сети данного класса.

5. Спроектировать разбиение заданной сети на N диапазонов адресов подсетей.

6. В отчете выписать (в двоичном и в десятичном виде):

- Host IP-address;
- Network ID;
- Host ID;
- Subnet Mask, Subnet Prefix;
- broadcast адрес.

7. Построить диапазоны IP для каждой из N подсетей:

- Network ID подсети;
- Subnet Mask, Subnet Prefix;

- вычислить число хостов в данной подсети;
- диапазон адресов (минимальный и максимальный адрес хоста);
- broadcast адрес для каждой подсети.

ПРИМЕЧАНИЯ

1. Построить подсети так, чтобы исходный хост входил в одну из них.

ВАРИАНТЫ ИНДИВИДУАЛЬНЫХ ЗАДАНИЙ

№ варианта	IP-адрес хоста	Число подсетей N
1	172.183.211.75	10
2	135.202.233.13	2
3	146.179.250.1	3
4	193.212.179.1	2
5	152.102.242.11	4
6	171.2.253.145	5
7	188.12.100.131	6
8	212.144.131.12	3
9	191.101.123.123	7
10	129.98.98.91	8
11	222.225.115.222	6
12	169.76.210.87	9
13	202.123.103.145	4
14	194.252.215.201	7
15	133.23.249.43	11
16	197.171.109.190	5
17	148.12.234.61	12
18	157.45.228.49	13
19	155.156.217.51	14
20	128.222.201.27	15

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Различия протоколов *IPv4, IPv6.*
2. Как разделить *IP-адрес* на *Network ID* и *Host ID*?
3. Для чего используется *Subnet Mask*?

4. Продемонстрировать эквивалентные записи маски и префикса сети

5. Классы *IP-адресов*. Как определить класс по адресу?

6. Как определить число сетей и узлов в определенном классе IP-адресов?

7. Смысл понятий *unicast* и *multicast* адресов.

8. Для чего используется *broadcast* адресация?

9. Какой класс используется для групповой адресации?

10. Как используется *Subnet mask* для доставки пакета в локальную или удаленную сеть?

11. Что такое основные шлюзы?

12. Система DNS — предназначение, основные функции.

13. Определить понятия: IP-адрес, доменные имена, MAC-адрес.

10 МЕТОД ДОСТУПА CSMA/CD И ОБЕСПЕЧЕНИЕ НАДЕЖНОСТИ ETHERNET

В технологии Ethernet, независимо от применяемого стандарта физического уровня, существует понятие домена коллизий.

Домен коллизий (*collision domain*) — это часть сети Ethernet, все узлы которой распознают коллизию независимо от того, в какой части этой сети коллизия возникла. Сеть Ethernet, построенная на повторителях, всегда образует один домен коллизий. Домен коллизий соответствует одной разделяемой среде. Мосты, коммутаторы и маршрутизаторы делят сеть Ethernet на несколько доменов коллизий.

Узлы, образующие один домен коллизий, работают синхронно, как единая распределенная электронная схема.

Правило 4-х повторителей. При описании топологии сети стандарта 10Base-5 приводились ограничения на длину одного непрерывного отрезка коаксиального кабеля, используемого в качестве общей шины передачи данных для всех станций сети. Отрезок кабеля, завершающийся на обоих концах терминаторами и имеющий общую длину не более 500 м называется физическим сегментом сети. Однако при расчете окна коллизий общая максимальная длина сети 10Base-5 считалась равной 2500 м. Противоречия здесь нет, так как стандарт 10Base-5 (впрочем как и остальные стандарты физического уровня Ethernet) допускает соединение нескольких сегментов коаксиального кабеля с помощью повторителей, которые обеспечивают увеличение общей длины сети. Повторитель соединяет два сегмента коаксиального кабеля и выполняет функции регенерации электрической формы сигналов и их *синхронизации* (*retiming*). Повторитель прозрачен для станций, он обязан передавать кадры без искажений, модификации, потери или дублирования. Имеются ограничения на максимально допустимые величины дополнительных задержек распространения битов нормального кадра через повторитель, а также битов jam-последовательности, которую повторитель обязан передать на все подключенные к нему сегменты при обнаружении коллизии на одном из них. Воспроизведение коллизии на всех подключенных к повторителю сегментах — одна из его основных функций.

В общем случае стандарт 10Base-5 допускает использование *до 4-х повторителей*, соединяющих в этом случае *5 сегментов* длиной до 500 м каждый, если используемые повторители удовлетворяют ограничениям на допустимые величины задержек сигналов. При этом общая длина сети будет составлять 2500 м, и такая конфигурация гарантирует правильное обнаружение коллизии крайними станциями сети. Только 3 сегмента из 5 могут быть нагруженными, то есть сегментами с подключенными к ним трансиверами конечных станций.

Правила 4-х повторителей и максимальной длины каждого из сегментов легко использовать на практике для определения корректности конфигурации сети. Однако эти правила применимы только тогда, когда все соединяемые сегменты представляют собой одну физическую среду, то есть в нашем случае толстый коаксиальный кабель, а все повторители также удовлетворяют требованиям физического стандарта 10Base-5. Аналогичные простые правила существуют и для сетей, все сегменты которых удовлетворяют требованиям другого физического стандарта, например, 10Base-T или 10Base-F. Однако для смешанных случаев, когда в одной сети Ethernet присутствуют сегменты различных физических стандартов, правила, основанные только на количестве повторителей и максимальных длинных сегментов становятся более запутанными.

Поэтому более надежно рассчитывать время полного оборота сигнала (PDV) по смешанной сети с учетом задержек в каждом типе сегментов и в каждом типе повторителей.

Расчет PDV и PVV. Для того чтобы сеть *Ethernet*, состоящая из сегментов различной физической природы, работала корректно, необходимо, чтобы выполнялись три основных условия:

- ***Количество станций*** в сети не превышает ***1024*** (с учетом ограничений для коаксиальных сегментов).
- Максимальная длина каждого физического сегмента не более величины, определенной в соответствующем стандарте физического уровня;
- Удвоенная задержка распространения сигнала (***Path Delay Value***, ***PDV***) между двумя самыми удаленными друг от друга станциями сети не превышает ***575 bt*** (количество битовых интервалов в пакете минимальной длины с учетом преамбулы).

- Сокращение межкадрового расстояния (***Path Variability Value***, ***PVV***) при прохождении последовательности кадров через все повторители не более чем на ***49 bt*** (напомним, что при отправке кадров станция обеспечивает начальное межкадровое расстояние в 96 битовых интервалов, значит после прохождения сегмента оно должно быть не меньше, чем 96 – 49 = 47 bt).

Соблюдение этих требований обеспечивает корректность работы сети даже в случаях, когда нарушаются простые правила конфигурирования, определяющие максимальное количество повторителей и максимальную длину сегментов каждого типа.

Физический смысл ограничения задержки распространения сигнала по сети (*PDV*) уже пояснялся — соблюдение этого требования обеспечивает своевременное обнаружение коллизий.

Требование на минимальное межкадровое расстояние связано с тем, что при прохождении кадра через повторитель это расстояние уменьшается. Каждый пакет, принимаемый повторителем, ресинхронизируется для исключения дрожания сигналов, накопленного при прохождении последовательности импульсов по кабелю и через интерфейсные схемы. Процесс ресинхронизации обычно увеличивает длину преамбулы, что уменьшает межкадровый интервал. При прохождении кадров через несколько повторителей межкадровый интервал может уменьшиться настолько, что сетевым адаптерам в последнем сегменте не хватит времени на обработку предыдущего кадра, в результате чего кадр будет просто потерян. Поэтому не допускается суммарное уменьшение межкадрового интервала более чем на 49 битовых интервалов. Величину уменьшения межкадрового расстояния при переходе между соседними сегментами обычно называют в англоязычной литературе *Segment Variability Value*, ***SVV***, а суммарную величину уменьшения межкадрового интервала при прохождении всех повторителей — *Path Variability Value,* ***PVV***. Очевидно, что величина *PVV* равна сумме *SVV* всех сегментов, кроме последнего.

Расчет PDV. Для упрощения расчетов обычно используются справочные данные, содержащие значения задержек распространения сигналов в повторителях, приемопередатчиках и в различных физических средах.

Таблица 2

Тип сегмента	*База левого сегмента*	*База промежуточного сегмента*	*База правого сегмента*	*Задержка среды на 1 м*	*Max длина сегмента*
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

Поясним терминологию, использованную в этой таблице, на примере сети, изображенной на рис. 27.

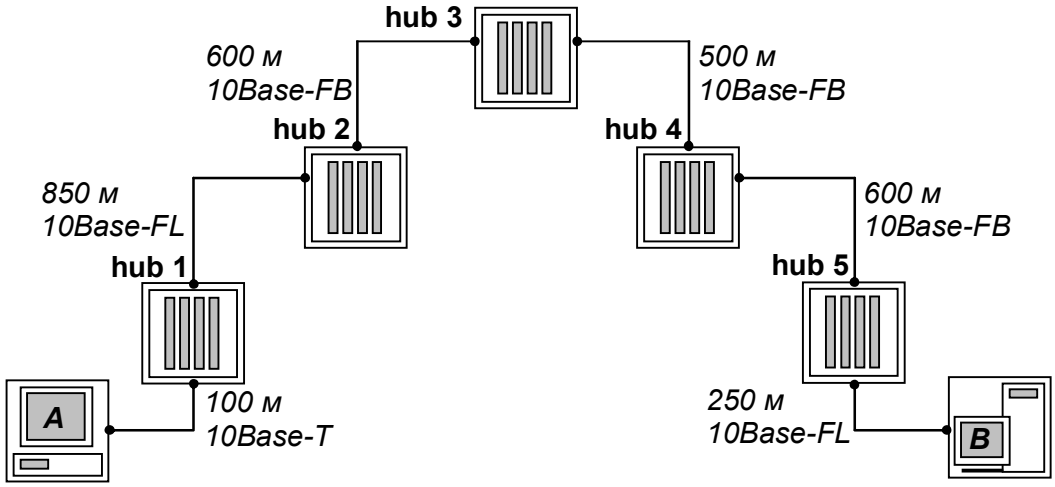


Рис. 27 — Пример сети *Ethernet*

Л*евым* сегментом называется сегмент, в котором начинается путь сигнала от выхода передатчика (выход *Tx*) конечного узла. Затем сигнал проходит через промежуточные сегменты и доходит до приемника (вход *Rx*) наиболее удаленного узла наиболее удаленного сегмента, который называется *правым*. С каждым сегментом связана постоянная задержка $PDV_i^{БАЗА}$, названная базой, которая зависит только от типа сегмента и от положения сегмента на пути сигнала (левый, промежуточный или правый).

Кроме этого, с каждым сегментом связана задержка распространения сигнала вдоль кабеля сегмента, которая зависит от длины сегмента L_i и вычисляется путем умножения времени распространения сигнала по одному метру кабеля $t_i^{1\,м}$ (в битовых интервалах) на длину кабеля в метрах:

$$PDV_i = PDV_i^{БАЗА} + L_i \times t_i^{1м}.$$

Общее значение *PDV* равно сумме базовых и переменных задержек всех сегментов сети:

$$PDV = \sum_i PDV_i.$$

Значения констант в табл. 2 даны с учетом удвоения величины задержки при круговом обходе сети сигналом, поэтому удваивать полученную сумму не нужно.

Так как левый и правый сегмент имеют различные величины базовой задержки, то в случае различных типов сегментов на удаленных краях сети необходимо выполнить расчеты дважды: один раз принять в качестве левого сегмента сегмент одного типа (*A–B*), а во второй раз — сегмент другого типа (*B–A*), а результатом считать максимальное значение *PDV*.

Таблица 3

A–B	***Ethernet***	***Длина, м***	***Расчет***	***PDV***
A–1 (левый)	10Base-T	100	15.3 + 100 × 0.113	26.6
1–2	10Base-FL	850	33.5 + 850 × 0.1	118.5
2–3	10Base-FB	600	24 + 600 × 0.1	84
3–4	10Base-FB	500	24 + 500 × 0.1	74
4–5	10Base-FB	600	24 + 600 × 0.1	84
5–B (правый)	10Base-FL	250	156 + 250 × 0.1	181
		2900		**568.1**
B–A	***Ethernet***	***Длина, м***	***Расчет***	***PDV***
B–5 (левый)	10Base-FL	250	12.3 + 250 × 0.1	37.3
5–4	10Base-FB	600	24 + 600 × 0.1	84
4–3	10Base-FB	500	24 + 500 × 0.1	74
3–2	10Base-FB	600	24 + 600 × 0.1	84
2–1	10Base-FL	850	33.5 + 850 × 0.1	118.5
1–A (правый)	10Base-T	100	165 + 100 × 0.113	176.3
		2900		**574.1**

В табл. 3 приведены расчеты PDV для каждого сегмента (рис. 27) в обоих направлениях, в результате задержка распространения сигнала получилась равной 574.1 bt. Так как значение PDV меньше максимально допустимой величины 575, то эта сеть проходит по величине максимально возможной задержки оборота сигнала. Несмотря на то, что ее общая длина существенно больше 2 500 м и используется целых 5 повторителей.

Расчет PVV. Для расчета PVV также можно воспользоваться табличными значениями максимальных величин уменьшения межкадрового интервала при прохождении повторителей различных физических сред (табл. 4).

Таблица 4

Тип сегмента	***Передающий сегмент***	***Промежуточный сегмент***
10Base-5 или 10Base-2	16	11
10Base-FB	–	2
10Base-FL	10.5	8
10Base-T	10.5	8

В соответствии с этими данными рассчитаем значение PVV для нашего примера.

Таблица 5

A–B	***Ethernet***	***PVV***
A–1 (левый)	10Base-T	10.5
1–2	10Base-FL	8
2–3	10Base-FB	2
3–4	10Base-FB	2
4–5	10Base-FB	2
5–B (правый)	10Base-FL	8
		32.5

B–A	***Ethernet***	***PVV***
B–5 (левый)	10Base-FL	10.5
5–4	10Base-FB	2
4–3	10Base-FB	2
3–2	10Base-FB	2
2–1	10Base-FL	8
1–A (правый)	10Base-T	8
		32.5

Сумма этих величин дает значение PVV, равное 32.5 bt, что меньше предельного значения в 49 битовых интервалов.

В результате, приведенная в примере сеть по всем параметрам соответствует стандартам Ethernet.

ЛАБОРАТОРНАЯ РАБОТА № 10
Метод доступа CSMA/CD и обеспечение надежности Ethernet

Продолжительность — 2 часа.
Максимальный рейтинг — 8 баллов.

ЦЕЛЬ РАБОТЫ

Освоить методы расчета характеристик сети, гарантирующих корректность передачи кадров и доступа к среде в технологии Ethernet.

ЗАДАНИЕ

1. Повторить темы, связанные с передачей данных в каналах связи, методы доступа, форматы пакетов и характеристики передачи. Повторить методы Ethernet доступа и способы расчета PDV и PVV.

2. В соответствии с индивидуальным заданием спроектировать ЛВС.

3. Обязательно произвести оценочный расчет следующих параметров:

– Vmax максимальный объем передаваемых данных для каждого домена сети;

– Cbt_max Cbt_min требуемую max и min битовую скорость передачи данных (при передаче максимальными 1508б и минимальными 72б пакетами);

– Ckdr_min Ckdr_max требуемую max и min кадровую скорость передачи данных;

– t_pr время реакции сети (при передаче максимальными и минимальными пакетами).

4. Доказать работоспособность сегментов сети с точки зрения методов доступа. Для технологии типа Ethernet произвести расчет PDV и PVV.

5. Для сетевого сервера рассчитать:

– необходимое быстродействие;

– объем ОЗУ и сетевого буфера;

– объем HDD.

6. Спроектировать адресацию компьютеров в сети. Разработать схему подсетей.

Выписать маски подсетей для всех сегментов и диапазоны адресов хостов.

Допускается использование VLAN.

ВАРИАНТЫ ТОПОЛОГИЧЕСКИХ СХЕМ СЕТИ

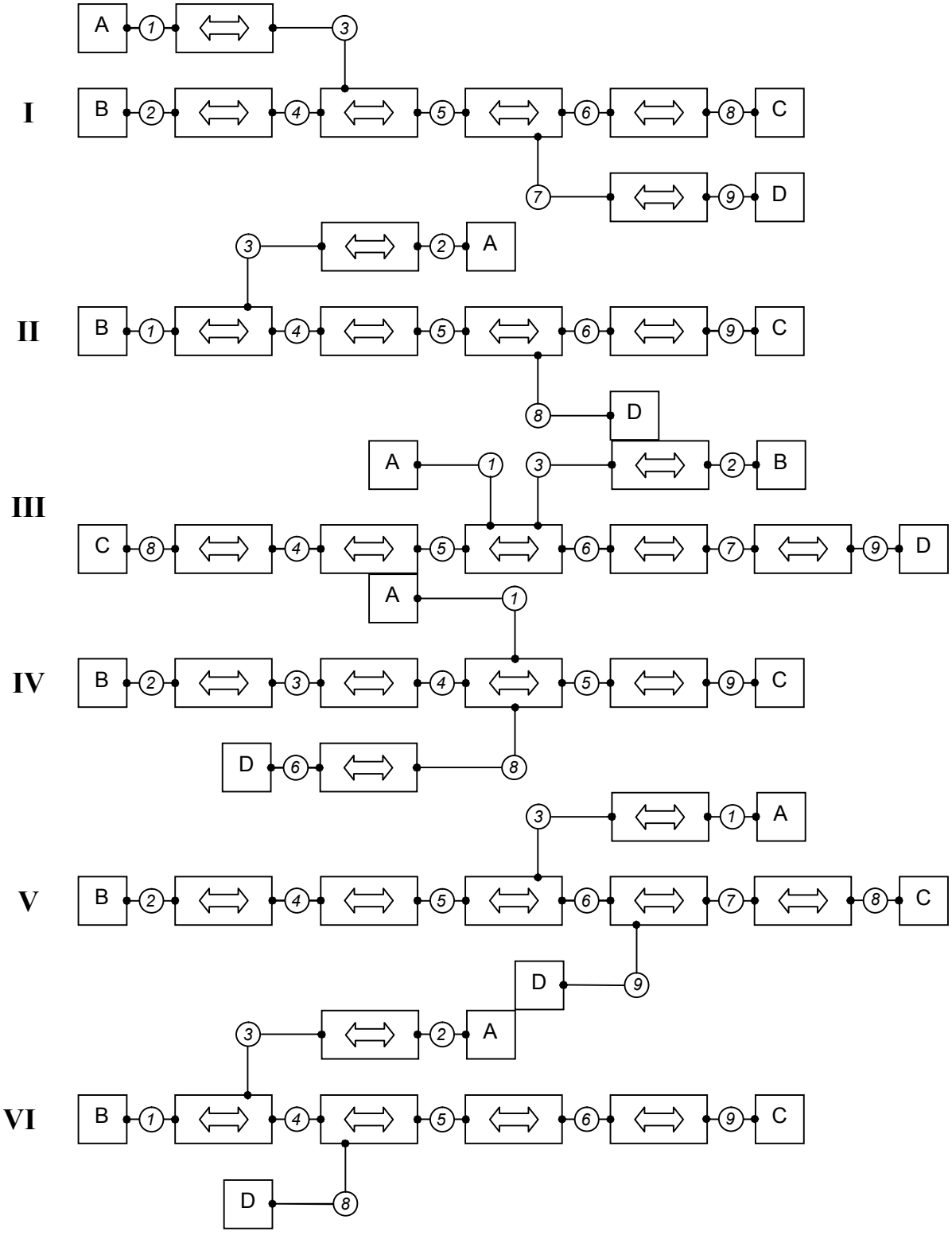


Таблица 1 — **Протоколы передачи и длины сегментов для топологических схем сети**

Вариант	1 сегмент	2 сегмент	3 сегмент	4 сегмент	5 сегмент	6 сегмент	7 сегмент	8 сегмент	9 сегмент
1	10Base-T 60	10Base-5 150	10Base-5 100	10Base-FB 700	10Base-FL 200	10Base-5 200	FOIRL 900	10Base-T 100	10Base-5 200
2	10Base-2 50	10Base-2 100	FOIRL 850	10Base-FL 1600	FOIRL 300	FOIRL 700	10Base-FL 1900	10Base-2 160	10Base-2 100
3	10Base-T 50	10Base-T 100	10Base-5 50	10Base-5 200	10Base-FL 800	10Base-FL 900	FOIRL 900	10Base-T 800	10Base-T 80
4	10Base-T 100	10Base-2 125	FOIRL 800	FOIRL 900	FOIRL 900	FOIRL 600	10Base-FL 1400	10Base-T 60	10Base-T 70
5	10Base-T 70	10Base-FL 1500	10Base-FB 600	10Base-FB 1800	10Base-FL 700	10Base-FL 1500	FOIRL 800	10Base-T 40	10Base-FL 1750
6	10Base-2 110	10Base-2 125	10Base-5 500	10Base-FL 1400	FOIRL 1000	FOIRL 800	10Base-FB 1800	10Base-5 340	FOIRL 400
7	10Base-T 80	10Base-5 125	FOIRL 900	FOIRL 700	10Base-FB 1500	10Base-FB 800	10Base-FL 1200	10Base-5 200	10Base-T 40
8	10Base-2 50	10Base-2 90	10Base-5 400	10Base-FL 1700	10Base-FL 1200	10Base-FL 1300	FOIRL 650	10Base-2 100	10Base-2 150
9	10Base-T 60	10Base-T 90	10Base-FL 700	10Base-5 200	FOIRL 550	FOIRL 650	10Base-FB 1250	10Base-T 80	10Base-T 90
10	10Base-T 70	10Base-T 30	10Base-FB 500	10Base-5 350	10Base-FB 1700	10Base-FB 1500	10Base-5 340	10Base-2 125	10Base-T 100
11	10Base-2 75	10Base-FL 1300	10Base-FL 500	10Base-5 300	10Base-5 100	10Base-5 200	10Base-FB 750	10Base-FL 1800	10Base-2 135
12	10Base-T 90	FOIRL 750	10Base-5 200	FOIRL 800	10Base-FL 1500	10Base-FB 500	10Base-FL 800	FOIRL 200	10Base-2 150
13	10Base-T 60	10Base-2 125	FOIRL 700	10Base-FB 700	10Base-5 200	10Base-FB 1500	10Base-FL 1000	10Base-T 40	10Base-T 100
14	10Base-2 50	10Base-5 125	10Base-FL 1700	10Base-FL 1600	FOIRL 700	10Base-FL 1200	FOIRL 650	10Base-2 150	10Base-2 160
15	10Base-T 50	10Base-2 90	10Base-5 200	10Base-5 200	10Base-FL 900	FOIRL 550	10Base-FB 1250	10Base-T 90	10Base-T 800
16	10Base-T 100	10Base-T 90	10Base-5 350	FOIRL 900	FOIRL 600	10Base-FB 1700	10Base-5 340	10Base-T 100	10Base-T 60
17	10Base-T 70	10Base-T 30	10Base-5 300	10Base-FB 1800	10Base-FL 1500	10Base-5 100	10Base-FB 750	10Base-2 135	10Base-T 40
18	10Base-2 110	10Base-FL 1300	FOIRL 800	10Base-FL 1400	FOIRL 800	10Base-FL 1500	10Base-FL 800	10Base-2 150	10Base-5 340
19	10Base-T 80	FOIRL 750	FOIRL 900	10Base-5 100	10Base-FL 200	10Base-FB 800	FOIRL 900	10Base-5 200	10Base-5 200
20	10Base-2 50	10Base-5 150	10Base-5 400	FOIRL 850	FOIRL 300	10Base-FL 1300	10Base-FL 1900	10Base-2 100	10Base-2 100
21	10Base-T 60	10Base-2 100	10Base-FL 700	10Base-5 50	10Base-FL 800	FOIRL 650	FOIRL 900	10Base-T 80	10Base-T 80
22	10Base-T 70	10Base-T 100	10Base-FB 500	FOIRL 800	FOIRL 900	10Base-FB 1500	10Base-FL 1400	10Base-T 70	10Base-2 125
23	10Base-2 75	10Base-2 125	10Base-FL 500	10Base-FB 600	10Base-FL 700	10Base-5 200	FOIRL 800	10Base-FL 1750	10Base-FL 1800
24	10Base-T 90	10Base-FL 1500	10Base-5 200	10Base-5 500	FOIRL 1000	10Base-FB 500	10Base-FB 1800	FOIRL 400	FOIRL 200

11 SOCKET-ПРОГРАММИРОВАНИЕ

Протокол IP не имеет никакой обратной связи и не гарантирует целостности передаваемых данных, то для этого применяются протоколы более высокого уровня — TCP и UDP. Пакеты данных, сформированные этими протоколами, не могут в чистом виде передаваться по сети, они вкладываются в IP-пакеты и передаются в их составе. Протокол UDP обеспечивает быструю, но ненадежную передачу данных, без организации канала и без подтверждения доставки. Протокол TCP — существенно более надежный, но и не такой быстрый. При отправке данных ожидается некоторое время ответ получателя, если ответа нет или пришел ответ о получении испорченных данных, то происходит повторная передача.

При работе с компонентами C++Builder или Delphi, предназначенными для передачи данных по сетям всегда необходимо заполнять свойство ***Host*** компонента. Это свойство содержит имя (DNS, Wins) или IP-адрес и используется для формирования пакетов при передаче по сети. Другое свойство каждого сетевого компонента — это свойство ***Port***. Речь идет не об аппаратных портах компьютера, а о программных портах Windows. Порт характеризуется своим номером, многие номера закреплены за определенными протоколами и службами Интернета. При самостоятельном определении номера порта необходимо следить за тем, чтобы задаваемый номер порта не совпал с используемым (зарезервированным) номером. Какие именно порты вашей системы заняты, можно узнать, посмотрев файл ***Services*** без расширения, расположенный в каталоге ***Windows*** (для Windows 9.x) или в каталоге ***System32\drivers\etc*** (для Windows NT/2000/XP).

Сокеты (socket) — это некие программные коммутаторы, обслуживающие соединения в сети. По своему назначению они разделяются на 3 типа: клиентские, слушающие и серверные. Клиентский сокет инициирует соединение, указывая имя или IP-адрес удаленного сервера и порт, используемый сервером. На сервере запрос клиента об установлении связи получает слушающий сокет, его задача — формировать очередь запросов. А серверный сокет, последовательно обрабатывая эту очередь и дойдя до запроса клиента, устанавливает соединение с клиентским сокетом. Он от-

правляет ответ на клиентский сокет, который получает описание серверного сокета с которым он установил связь.

Таким образом, сокеты позволяют организовать обмен информацией в сетях, когда к одному серверу может одновременно обращаться множество клиентов. Функции клиентского сокета в C++Builder (или Delphi) выполняет компонент ***TClientSocke***t, а за серверный сокет отвечает компонент ***TServerSocket***. Рассмотрим пример использования компонентов TClientSocket и TServerSocket.

Сервер

1. Для создания серверного приложения нужно использовать компонент ***TServerSocket*****,** расположенный на вкладке ***Internet***.

2. Для компонента ***TServerSocket*** установить свойство Port равным 3000. Можно установить и другой номер TCP-порта — это не принципиально. Важно только, чтобы порт не оказался занятым каким-нибудь приложением. Этот же номер порта мы будем использовать и для ***TClientSocket.***

3. Необходимо активировать серверное приложение — дать команду на прослушивание соответствующего порта.

ServerSocket1->Active= true;

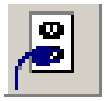
4. Написать обработчик для получения данных. Для объекта ***ServerSocket1*** обрабатываем событие ***OnClientRead***. Получить принятые от клиента данные можно при помощи метода ***ReceiveText***(), для этого используется передаваемый в обработчик параметр ***Socket*** типа ***TCustomWinSocket***.

5. Полезно визуализировать признак того, что клиент подсоединился к нашему серверу. Для этого создаются для объекта ***ServerSocket1*** обработчики событий ***OnClientConnect*** и ***OnClientDisconnect***, в которых отражается информация о признаке присоединения (отсоединения) клиента.

Написание серверного приложения закончено.

Клиент

1. Для создания клиентского приложения необходимо использовать компонент ***TClientSocket***.



2. Свойство ***Address*** для ***TClientSocket*** установить в ***127.0.0.1***, свойство ***Host*** задать равным ***localhost***, свойство ***Port*** — такое же, как и для ***TServerSocket***. Эти параметры задают компьютер, к которому будет подключаться клиент. Задавать его можно либо по IP-адресу (127.0.0.1 — это собственный IP-адрес компьютера), либо по имени (localhost — это универсальное собственное имя компьютера). Для TCP-порта требуется указать такое же значение, которое задано для TCP-порта сервера.

3. Для передачи текста на сервер требуется активировать соединение:

ClientSocket1->Active= true;

Этот код активирует ***Socket***. Так как у него установлены свойства ***Address***, ***Host*** и ***Port***, то произойдет соединение с соответствующим сервером.

4. Передать текст можно, например, так:

ClientSocket1->Socket->SendText(«текст»);

Клиентское приложение написано.

Можно запустить сервер и активировать его. После этого запустить клиента и активировать его, в серверном приложении признак присоединения должен поменяется. Дать команду на передачу текста. Проследить, что данные получены сервером.

Ниже приводится список компонентов с кратким описанием, наравне с сокетами используемых в написании разнообразных сетевых приложений.

Компоненты из набора FastNet

- Компонент ***TNMDayTime*** (вкладка ***FastNet***) предназначен для получения от серверов даты и времени в Интернет и интранет в соответствии со стандартом RFC 867. Используется свойство ***DayTimeStr***, в котором содержатся значения даты и времени, полученные с сервера. В свойстве ***Host*** нужно задать ссылку на соответствующий сервер даты и времени в Интернет.

- Компонент ***TNMMsg*** (вкладка ***FastNet***) предназначен для обмена простыми текстовыми сообщениями в кодах ASCII по Интернет и интранет с использованием протокола TCP/IP. При этом на хост-компьютере, на который по-

сылается сообщение, должен быть запущен сервер, использующий компонент ***TNMGeneralServer***. В свойстве ***Host*** задается имя удаленного компьютера или его IP, в свойстве ***Port*** указывается TCP-порт, который слушает сервер. Так же нужно задать свойство ***FromName***, а после этого можно вызывать метод ***PostIt***().

- Компонент ***TNMMsgServ*** (вкладка ***FastNet***) используется для получения простых текстовых сообщений в кодах ASCII, распространяемых по Интернет и интранет с использованием протокола TCP/IP. При этом на посылающий сообщение компьютер должен использоват компонент ***TNMMsg***. В свойстве ***Port*** указывается TCP-порт, который слушает сервер. Для обработки полученных сообщений используется обработчик события ***OnMSG***.

- Компонент ***TNMEcho*** (вкладка ***FastNet***) предназначен для отсылки текстовых сообщений на эхо-сервер и получения этого сообщения обратно в соответствии со стандартом RFC 862. Этот компонент применяется для тестирования и настройки ЛВС, т.к. позволяет оценить время ответа сервера, сохраняемое в свойстве ***ElapsedTime***.

- Компонент ***TNMFTP*** (вкладка ***FastNet***) применяется для обмена файлами между FTP-сервером и клиентской машиной по протоколу FTP. Для подключения к FTP-серверу необходимо задать свойства ***Host*** и ***Port***, а так же в свойствах ***UserId*** и ***Password*** необходимые имя пользователя и пароль. Метод ***Connect***() применяется для установления связи с сервером. Имеет большое число собственных свойств и методов, если метод компонента завершается успешно, то генерируется событие ***OnSuccess***, иначе — происходит событие ***OnFailure***. В обоих случаях через параметр ***Trans_Type*** обработчиков этих событий передается имя команды типа ***TCmdType***. Компонент позволяет на удаленном компьютере: определять содержимое каталога, изменять текущий каталог, считывать файлы, записывать файлы, удалять файлы или каталоги.

- Компонент ***TNMHTTP*** (вкладка ***FastNet***) используется для передачи через Интернет или интранет гипертекста по протоколу HTTP (версии HTTP 1.1). Для подключения к серверу необходимо задать свойства ***Host*** и ***Port***. Использует методы ***Get***() (для получения гипертекстовых докумен-

тов), ***Head***() (для считывания заголовка), ***Options***() (для получения справочной информации), Trace (для отладки), Put (для создания документа на сервере), ***Post***() (для редактирования гипертекста на сервере) и ***Delete***() (для удаления документа). Имеет ряд свойств, позволяющих работать с HTML-документами. Если метод компонента завершается успешно, то генерируется событие ***OnSuccess***, иначе — происходит событие ***OnFailure***.

- Компонент ***TNMNNTP*** (вкладка ***FastNet***) предназначен для чтения и публикации сообщений на NEWS-серверах в Интернет в соответствии с протоколом NNTP (RFC 977). Для подключения к серверу новостей необходимо задать его имя (или IP-адрес) в свойстве ***Host*** и вызвать метод ***Connect***(). Компонент выполняет следующие основные задачи: получение списка групп новостей — метод ***GetGroupList***(), выбор активной группы новостей ***SetGroup***(), чтение статей — метод ***GetArticle***(), и свойства ***Header*** и ***Body*** и отправка статей в выбранную группу — свойства ***PostHeader***, ***PostBody*** и метод ***PostArticle***().

- Компонент ***TNMPOP3*** (вкладка ***FastNet***) используется для получения электронных писем (e-mail) от почтового сервера POP3. Для подключения к серверу необходимо задать свойства ***Host*** и ***Port***, а так же в свойствах ***UserId*** и ***Password*** прописать необходимые имя пользователя и пароль, затем вызвать метод ***Connect***() для установления связи с сервером. Для получения почты необходимо использовать метод ***GetMailMessage***(), полученное сообщение попадает в свойство ***MailMessage***. При успешном или ошибочном завершении методов, генерируются события ***OnSuccess*** или ***OnFailure*** соответственно.

- Компонент ***TNMSMTP*** (вкладка ***FastNet***) позволяет отправить почту через почтовый сервер Интернет по протоколу SMTP (RFC 821). Необходимо задать свойства ***Host*** и ***Port***, а затем вызвать метод ***Connect***(). Для разрыва соединения используется метод ***Disconnect***(). Для отправления почты необходимо заполнить свойство ***PostMessage*** необходимыми элементами электронного письма и вызвать метод ***SendMail***(). Метод ***Verify***() применяется для проверки существо-

вания данного имени на сервере, а метод ***ExpandList***() возвращает список рассылки сервера.

- Компонент ***TNMUDP*** (вкладка ***FastNet***) применяется для отправки пакетов по сети Интернет или интранет с использованием протокола UDP (RFC 768). Имя удаленного компьютера и порт задаются в свойствах ***RemoteHost*** и ***RemotePort***. Для получения данных по протоколу UDP нужно задать значение свойству ***LocalPort***. Передача может вестись двумя способами: Передача двоичного потока — ***SendStream***() и передача массива символов (буфер) — ***SendBuffer***(). При корректном поступлении данных происходит событие ***OnDataAvailable***, в обработчике этого события можно считывать данные в буфер — ***ReadBuffer***() или в поток — ***ReadStream***().

ЛАБОРАТОРНАЯ РАБОТА № 11
Основы socket-программирования под Windows

Продолжительность — 4 часа.
Максимальный рейтинг — 8 баллов.

ЦЕЛЬ РАБОТЫ

Целью данной лабораторной работы является получение основ программирования Windows-приложений непосредственно для локальной вычислительной сети. В ходе выполнения студенты получат навыки в осуществлении связи компьютеров через ЛВС, передачи информации и удаленным управлением компьютера с помощью средств C++Builder (Delphi).

ЗАДАНИЕ

1. Повторить понятия: протокол, интерфейс, пакет, кадр, адрес, IP-адрес. Ознакомиться с сетевыми службами Windows: DNS, MSClient, Wins, DHCP. Вспомнить навыки работы с сетевыми утилитами: ipconfig, ping, netstat.

2. Изучить вопросы программирования в среде визуальной разработки C++Builder или Delphi (Borland). Освоить методы программирования Windows-приложений. Изучить литературу и Интернет-ресурсы по socket-программированию.

4. Изучить протоколы стека TCP\IP: IP, TCP, UDP, ICMP, IGMP, ARP, RARP, RIP, DHCP, Telnet, FTP, TFTP, SNMP, SMTP, POP3, HTTP и т.п.

5. Написать программы, соответствующие своему индивидуальному заданию (варианты индивидуальных заданий определяет преподаватель). Отладить, протестировать их.

6. Снабдить программы необходимыми комментариями, спроектировать удобный визуальный интерфейс.

7. В отчете привести полные блок-схемы алгоритмов клиентской и серверной программ, листинг программы, снабженные комментариями и результаты работы программ (листинги, скриншоты, файлы с данными и т.п.).

ВАРИАНТЫ ИНДИВИДУАЛЬНЫХ ЗАДАНИЙ

1. Написать программу, копирующую определенный файл с удаленного компьютера. С помощью TClientSocket осуществить перенос файла с сервера на компьютер-клиент.

2. Написать программу, копирующую определенный файл с удаленного компьютера. С помощью TServerSocket осуществить перенос файла с сервера на компьютер-клиент.

3. Написать программу, копирующую определенный файл на удаленный компьютер. С помощью TClientSocket осуществить перенос файла с клиента на сервера.

4. Написать программу, копирующую определенный файл на удаленный компьютер. С помощью TServerSocket осуществить перенос файла с клиента на сервера.

5. С помощью компонент TClientSocket и TServerSocket, написать подпрограмму позволяющую открывать или закрывать определенный порт (использовать порт от 33333) на удаленном компьютере.

6. Создать программу с использованием TServerSocket, позволяющую осуществлять пингование IP-адреса на наличие открытых портов.

7. Создать программу с использованием TClientSocket, позволяющую осуществлять пингование IP-адреса на наличие открытых портов.

8. Создать программу ЧАТ с помощью средств C++Builder (Delphi), используя TClientSocket и TServerSocket.

9. Используя TClientSocket и TServerSocket обеспечить связь между двумя компьютерами с возможностью пересылки сообщений как с клиента, так и с сервера.

10. Создать программу удаленного управления компьютером, состоящую из двух подпрограмм клиент-сервер. Программа клиента должна осуществлять подключение к серверу, пересылку текстовых команд (koff, moff, close). Программа сервера должна открывать порт, ждать подключения клиента, при поступлении команды (koff) произвести отключение клавиатуры на компьютере, при получении следующей команды (moff) произвести отключение мыши. При получении команды close осуществить отсоединение от клиента и закрывает себя.

11. Написать программу, определяющую с помощью TClientSocket и TServerSocket дескриптор сокета Windows (который необходим для возова API-функций сокетов).

12. Использовать TNMDayTime и TNMTime для получения даты и времени с удаленного сервера.

13. Создать программу ЧАТ с помощью средств C++Builder (Delphi), используя TNMMsg и TNMMsgServ.

14. Создать программу, позволяющую осуществлять пингование IP-адреса с определением времени ответа сервера, использовать TNMEcho.

15. Использовать компонент TNMFTP для определения и вывода на экран содержимого каталогов FTP-сервера, считывания, записи и удаления указанного файла или каталога.

16. Использовать компонент TNMFTP для определения и вывода на экран содержимого каталогов FTP-сервера, передвижения по дереву каталогов, считывания и удаления файлов и каталогов

17. Создать программу, использующую компонент TNMHTTP для работы с гипертекстовыми документами: загрузить гипертекст с определенного URL, получить доступную справочную информацию о типе сервера, предусмотреть индикацию успешного и ошибочного выполнения операций.

18. Создать программу, использующую компонент TNMHTTP для работы с гипертекстовыми документами: считать заголовок гипертекстового документа с определенным URL, создать новый документ на данном сервере, получить справочную информацию об используемой версии протокола HTTP. Предусмотреть индикацию успешного и ошибочного выполнения операций.

19. Создать программу, использующую компонент TNMHTTP для работы с HTTP-документами: изменить определенный гипертекстовый документ на заданном сервере и получить справочную информацию о сервере, браузере и HTTP-протоколе. Предусмотреть индикацию успешного и ошибочного выполнения операций.

20. Используя компонент TNMNNTP написать программу для подключения к серверу новостей: получить список групп но-

востей, выбрать активную группу и прочитать заголовок и содержание выбранной статьи.

21. Используя компонент TNMNNTP написать программу для подключения к серверу новостей: получить список групп новостей, выбрать активную группу и отправить статью в выбранную группу.

22. Создать программу, использующую компонент TNMPOP3 для получения электронных писем (e-mail) от почтового сервера POP3. Предусмотреть индикацию успешного и ошибочного выполнения операций.

23. Компонент TNMSMTP использовать в программе для отправки почтового сообщения на почтовый сервер Интернет по протоколу SMTP. Предварительно проверив методом Verify() существование данного имени на сервере.

24. Использовать компонент TNMSMTP в написании программы для получения списка рассылки SMTP-сервера Интернет. Предусмотреть индикацию успешного и ошибочного выполнения операций.

25. Написать программу для отправки UDP-пакетами по сети Интернет двоичных данных, генерируемых потоком (Stream) — задать генератор случайных чисел. Программа-клиент должна получать эти данные в свой асинхронный поток и выводить на экран. Использовать компонент TNMUDP.

26. Написать программу для отправки из заданного буфера UDP-пакетами двумерных массивов данных (матриц 10x10), генерируемых случайным образом и помещаемых в буфер. Программа-клиент должна получать эти данные в свой буфер и выводить на экран. Использовать компонент TNMUDP.

27. Написать программу, которая с помощью TClientSocket принимает асинхронный поток (Stream) десятичных чисел, генерируемых потоком сервера и выводит эти данные на экран. Использовать методы SendStream(), ReceiveStream().

28. Написать программу, которая с помощью TClientSocket принимает последовательность текстовых данных, случайным образом генерируемых сервером и выводит эти данные на экран. Использовать методы SendText(), ReceiveText().

29. Написать программу, которая с помощью TClientSocket принимает в свой буфер, из буфера сервера текстовые данные, и

выводит этот текст на экран. Использовать методы SendBuf(), ReceiveBuf().

30. Написать программу, которая с помощью TClientSocket принимает в свой буфер, из буфера сервера двумерные массивы данных (4 матрицы 5x5), и выводит эти матрицы на экран. Использовать методы SendBuf(), ReceiveBuf().

ПРИМЕЧАНИЯ

1. Необходимые для выполнения индивидуального задания параметры (Address, Host, Port, RemoteHost, RemotePort, UserId и Password) на этапе разработки, отладки и тестирования программы задавать самостоятельно, при сдаче лабораторной работы — получить у преподавателя.

2. Для проверки работы программ №№ 22—24 создать тестовый почтовый ящик на открытых почтовых серверах типа *mail.ru* или *yandex.ru.*

12 НАСТРОЙКА ПОДКЛЮЧЕНИЯ К WI-FI СЕТИ

Несмотря на присущие локальным беспроводным компьютерным сетям (***сети Wi-Fi***) недостатки, прежде всего, — невысокую скорость передачи данных, они получают все большее распространение. Это связано с реальной свободой перемещения, предоставляемой Wi-Fi сетью пользователю. Кроме того, такие сети могут быть очень быстро развёрнуты, при минимальных затратах. Ограничение на радиус действия Wi-Fi сети преодолевается за счёт увеличения количества точек доступа или использования направленных антенн. Делаются попытки построения маршрутизируемых и масштабируемых Wi-Fi сетей. Поддержка стандарта Wi-Fi производителями мобильного оборудования: ноутбуков, карманных компьютеров и даже сотовых телефонов, также способствует популярности этого вида компьютерной связи.

Режимы функционирования Wi-Fi сетей

К оборудованию, которое нужно для развёртывания беспроводной сети, относят беспроводную ***точку доступа*** (***Access Point, AP***), выполняющую функции беспроводного концентратора, и беспроводные адаптеры. Однако в простейшем случае для развёртывания беспроводной сети не требуется даже точка доступа. Метод доступа в сети Wi-Fi используется такой же, как в сетях Ethernet, — ***CSMA/CD*** (***Carrier Sense Multiply Access with Collision Detection***), — множественного доступа с определением несущей частоты и распознаванием коллизий. Поэтому логическая топология сети Wi-Fi — общая шина, при этом существует три режима работы Wi-Fi сети:

- ***Специальный,*** (***Ad Hoc***), его также называют независимым основным режимом — ***Independent Basic Service Set*** (***IBSS***) или режимом одноранговой сети — Peer to Peer. В этом режиме два (или более) устройства, имеющие Wi-Fi адаптеры могут установить соединение друг с другом. Дальность работы беспроводного соединения в специальном режиме не превышает нескольких метров, т.к. мощность излучения Wi-Fi адаптеров невелика. При наличии преград — стен, потолочных перекрытий и т.п., такое соединение установить не удастся.

- ***Инфраструктуры***, (***Infrastructure***) или основной режим ***Basic Service Set*** (***BSS***). В этом режиме в Wi-Fi сети существует точка доступа, через которую осуществляется соединение всех устройств с беспроводными адаптерами. Мощность излучения точки доступа может достигать 100 мВт (это максимальное значение, допускаемое стандартом, т.к. микроволновое излучение опасно для живых организмов), поэтому дальность связи может достигать десятков метров даже при наличии преград.
- ***Инфраструктуры, расширенный, Extended Basic Service Set*** (***EBSS***), это режим распределенной сети, основной, расширенный. В этом режиме в единой сети Wi-Fi работает несколько точек доступа, что позволяет обеспечить покрытие сетью больших территорий (обычно офисных зданий, гостиниц, аэропортов и т.п.).

Стандарты беспроводной связи

В настоящее время в сетях Wi-Fi распространены три стандарта беспроводной связи:

- ***IEEE 802.11a***, используется диапазон 5 ГГц (не разрешен в России), максимальная скорость передачи данных 54 Мбит/с. Примерная дальность связи 35—75 метров;
- ***IEEE 802.11b***, используется диапазон 2,4 ГГц, максимальная скорость передачи данных 11 Мбит/с. Примерная дальность связи 40—80 метров;
- ***IEEE 802.11g***, используется диапазон 2,4 ГГц, максимальная скорость передачи данных 54 Мбит/с. Совместим с IEEE 802.11b, т.е. поддерживает «смешанные» сети в которых работают устройства обоих стандартов. При этом более «быстрые» устройства стандарта IEEE 802.11g начинают работать как устройства стандарта IEEE 802.11b, т.е. вся сеть становится более «медленной» (даже более «медленной», чем однородная сеть стандарта IEEE 802.11b, т.к. точка доступа выполняет дополнительные операции). Примерная дальность связи 40—90 метров.

Настройка нового беспроводного соединения

Для создания нового беспроводного соединения необходимо настроить его параметры:

- SSID, Service Set Identifier — уникальный идентификатор (имя) создаваемой беспроводной сети.
- Параметры шифрования:
  - Ключи шифрования, если используется режим предварительно заданных ключей.
  - Параметры аутентификации (смарт-карта или сертификат) при использовании протокола 802.1x и RADIUS-сервера.

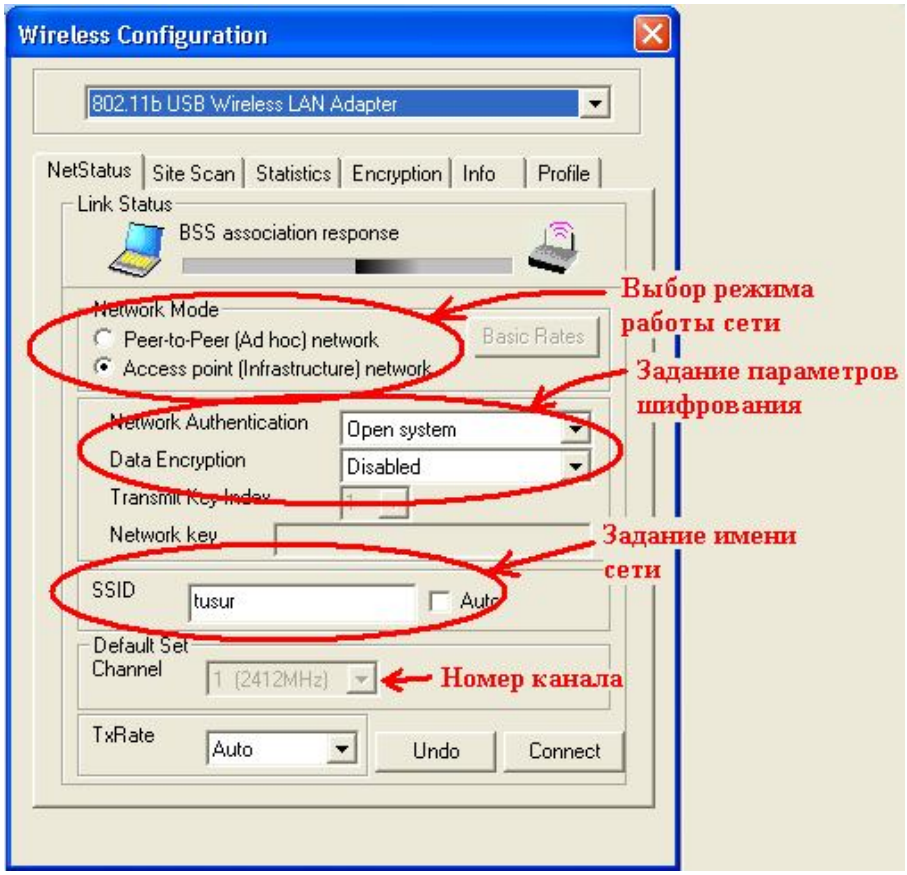


Рис. 28 — Настройка параметров беспроводного соединения в утилите ASUS Wireless Configuration

При создании одноранговой сети (без точки доступа) настройки выполняются на одном из компьютеров и, затем, повторяются на другом. На начальном этапе, для исключения возможных проблем, допустима настройка соединения ***без шифрования*** (***Open System***) с указанием только имени сети (***SSID***). На рис. 28 показано окно настройки беспроводного соединения на компьютере с помощью фирменной утилиты ASUS, а на рис. 29 — средства, встроенного в Windows. Фирменные утилиты позволяют

выполнить более тонкую настройку, например, выбрать номер канала, на котором будет работать соединение.

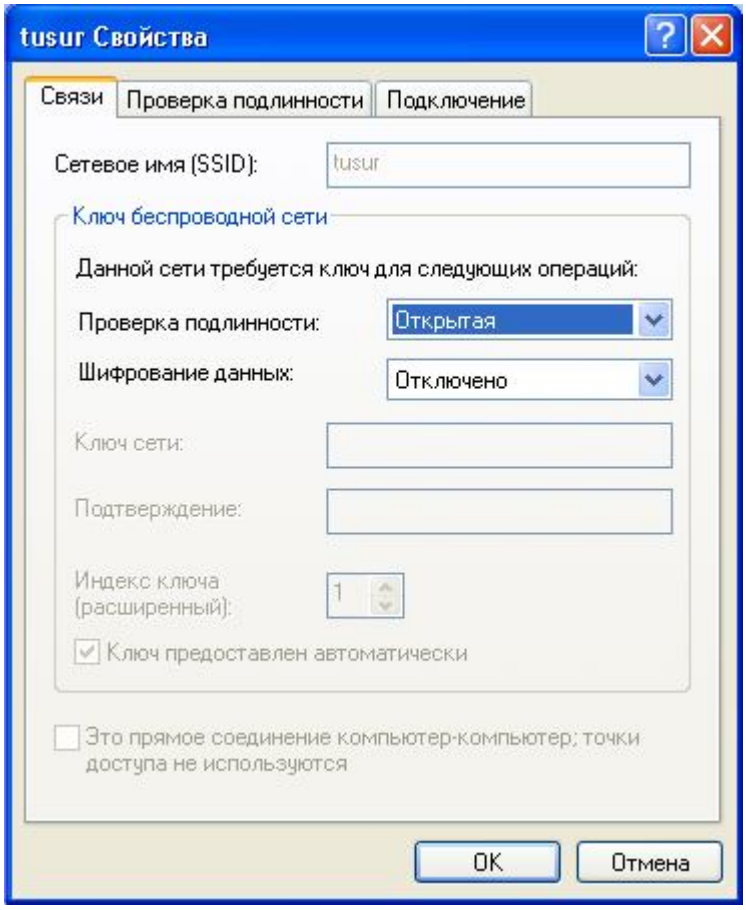


Рис. 29 — Настройка параметров беспроводного соединения с помощью встроенных средств Windows

Безопасность в беспроводных сетях

В беспроводных сетях Wi-Fi используется несколько стандартов шифрования. Первым из них был стандарт ***WEP*** (***Wired Equivalent Privacy — эквивалент проводной безопасности***). В соответствии со стандартом WEP шифрование осуществляется с помощью 64 или 128-битного ключа (некоторые модели современных точек доступа и беспроводных адаптеров поддерживают и более длинные ключи), а сам ключ представляет собой набор ASCII-символов длиной 5 (40-бит) или 13 (104-бит) символов, составляющих статическую часть ключа, к которой добавляется 24-битный вектор инициализации.

Вектор инициализации выбирается случайным образом и динамически меняется во время работы. Допускается вместо набора ASCII-символов использовать их шестнадцатеричные коды. Протокол WEP-шифрования, даже со 128-битным ключом, считается не очень стойким, он использовался в выпускавшихся ранее устройствах стандарта IEEE 802.11b. В устройствах стандарта IEEE 802.11g поддерживается улучшенный алгоритм шифрования ***WPA — Wi-Fi Protected Access***, который включает протоколы ***TKIP*** и ***MIC***, а также поддержку стандарта ***IEEE 802.1x***. Стандарт IEEE 802.1x описывает механизм защищенной аутентификации пользователей. Данный стандарт основан на работе в сети сервера аутентификации, его называют RADIUS-сервер, т.к. аутентификация выполняется по протоколу ***RADIUS*** (***Remote Authentication Dial-In User Service — Служба дистанционной аутентификации пользователей по коммутируемым линиям***). Стандарт IEEE 802.1x обеспечивает на сегодняшний день наиболее сильную защиту беспроводного соединения.

Протокол ***TKIP*** (***Temporal Key Integrity Protocol***) — реализует динамическую замену ключей шифрования. Ключи шифрования имеют длину 128 бит и генерируются по сложному алгоритму, а общее количество возможных вариантов ключей достигает сотни миллиардов, и меняются они очень часто.

Протокол ***MIC*** (***Message Integrity Check***) — это протокол проверки целостности пакетов. Протокол позволяет отбрасывать пакеты, которые были «вставлены» в канал связи. Обычно беспроводное оборудование поддерживает шифрование по протоколу TKIP одновременно с MIC.

Многие производители беспроводного оборудования встраивают в свои решения поддержку стандарта ***AES*** (***Advanced Encryption Standard***), который приходит на замену TKIP.

Выбор вариантов шифрования сильно зависит от конкретного беспроводного устройства, старые адаптеры стандарта IEEE 802.11b «знают» только WEP-шифрование. Однако использование протокола IEEE 802.1x возможно и с этими адаптерами.

Операционная система Windows имеет встроенную службу ***Wireless Zero Configuration,*** которая позволяет почти полностью автоматизировать процесс создания беспроводного соединения. Какое из средств настройки будет использоваться определяется

снятием или установкой флажка «Использовать Windows для настройки сети» в окне свойств беспроводного соединения, рис. 30.

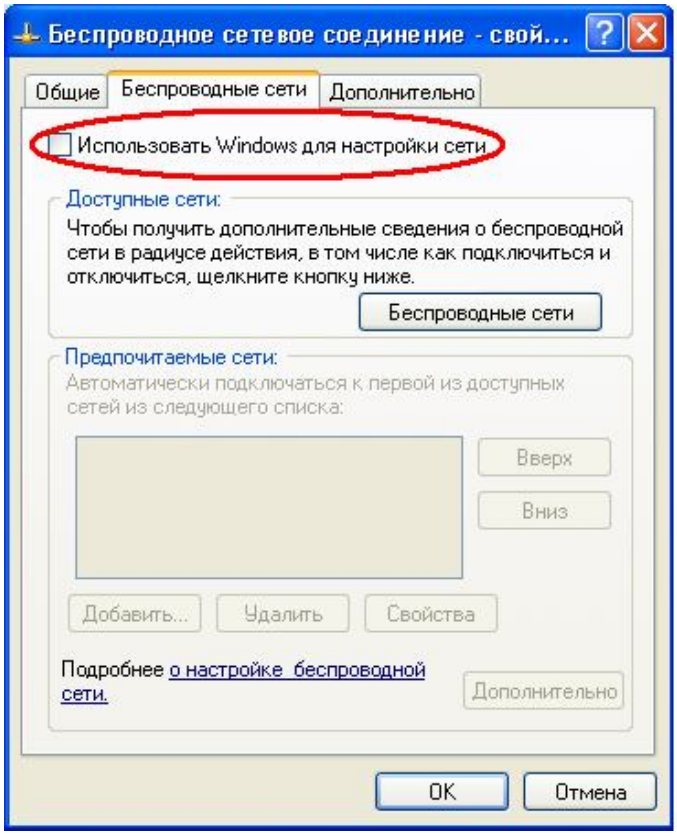


Рис. 30 — Выбор способа настройки беспроводного адаптера

Средства настройки Windows главным образом ориентированы на подключение к сетям на основе точки доступа, а при создании соединений типа Ad Hoc могут работать некорректно. Если беспроводная сеть строится на основе точки доступа, то все настройки соединения выполняются сначала на точке доступа. Точка доступа всегда сопровождается программным обеспечением, позволяющим задать кроме основных и дополнительные параметры соединения. Например, точка доступа может выполнять фильтрацию по MAC-адресам для подключающихся к беспроводной сети рабочих станций и разрешать подключение компьютеров только с указанными в фильтре физическими адресами, другая распространенная функция точек доступа — запрет широковещательной рассылки имени сети.

Беспроводные сети Wi-Fi не имеют средств маршрутизации, поэтому, при настройке протокола TCP/IP на беспроводном адап-

тере, сетевой адрес и маска подсети для всех клиентов сети выбирается из одной подсети.

Все остальные настройки беспроводного соединения (единая рабочая группа (домен), создание совместных сетевых ресурсов и определение прав доступа к ним) не отличаются от настройки обычных проводных соединений. С этой точки зрения клиенты беспроводной и проводной сети совершенно равноправны, а передача информации между сетями осуществляется абсолютно «прозрачно», т.е. невозможно прямо определить каким способом подключен к сети тот или другой компьютер.

Подключение к существующему беспроводному соединению

Для подключения к существующему беспроводному соединению клиенту необходимо знать уже указанные параметры: SSID и ключ (ключи) шифрования. В Windows XP после выхода обновления Service Pack 2 появился мастер подключения к беспроводной сети. Выполнив обзор доступных беспроводных сетей, рис. 31, можно выбрать сеть без шифрования (сеть tusur на рис. 31). Для подключения к такой сети достаточно нажать клавишу «Подключить» в правом нижнем углу окна мастера.

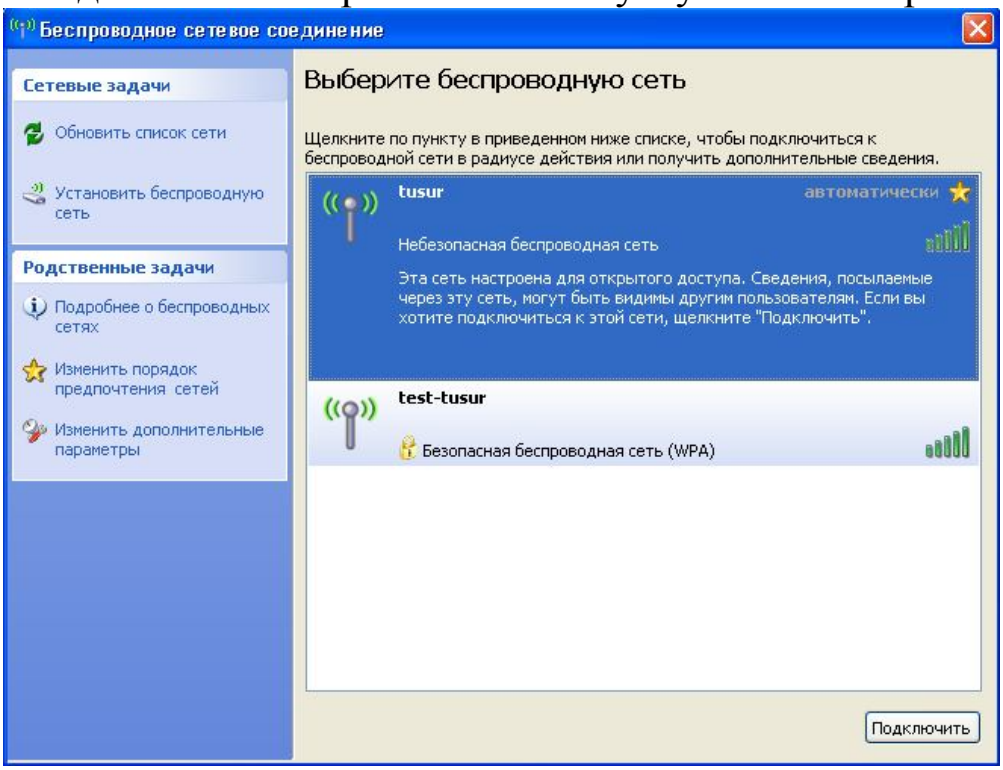


Рис. 31 — Мастер Windows подключения к беспроводной сети

При использовании шифрования в поле отображения информации о беспроводной сети выводится знак замка (сеть test-tusur на рис. 31). Для успешного подключения к такой сети необходимо, чтобы используемые в ней параметры шифрования совпали с заданными в свойствах соединения. На рис. 32 показаны значки Windows, отображающие тип беспроводного соединения.

Рис. 32 — Значок сети с точкой дотупа (слева)
и Ad hoc (справа)

Кроме указанных настроек необходимо, чтобы IP-адрес подключаемого компьютера лежал в той же подсети, что и адрес точки доступа. В беспроводных сетях общего доступа задача получения клиентом нужного IP-адреса решается с помощью сервера DHCP. Чтобы компьютер направил запрос DHCP-серверу необходимо в свойствах протокола Интернета задать режим автоматического получения IP-адреса и адреса сервера имен, рис. 33. Ключи шифрования в таких сетях обычно выдаются при покупке карточки пользования сетью.

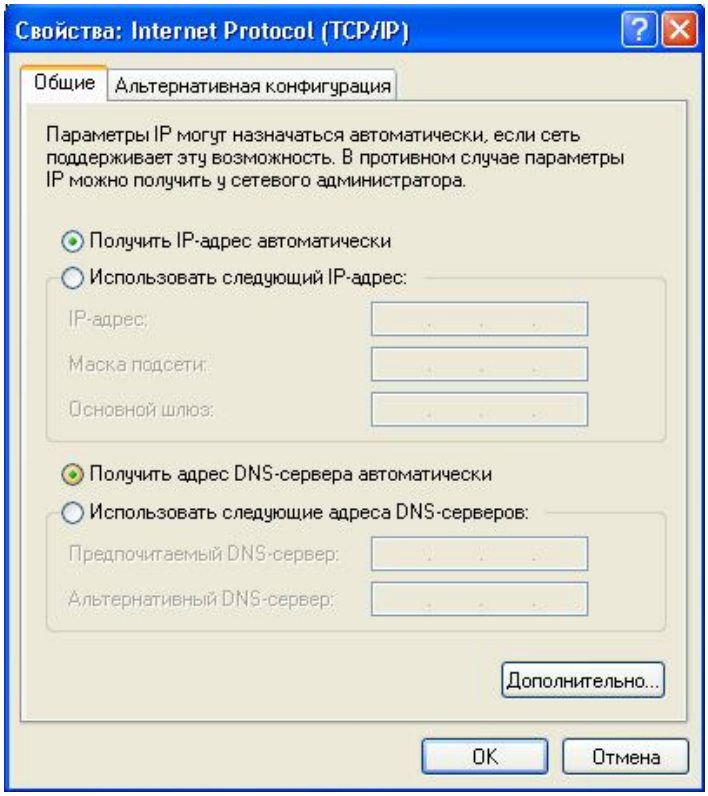


Рис. 33 — Настройка компьютера на поддержку
протокола DHCP

Протокол DHCP незначительно увеличивает время подключения к сети, поэтому его применяют только в сетях с постоянно меняющейся конфигурацией. Если конфигурация сети неизменна, то лучше использовать назначение компьютерам постоянных IP-адресов.

ЛАБОРАТОРНАЯ РАБОТА № 12
Подключение к открытой беспроводной сети

Продолжительность — 2 часа.
Максимальный рейтинг — 7 баллов.

ЦЕЛЬ РАБОТЫ

Целью данной лабораторной работы является получение навыков подключения оконечного устройства к открытой беспроводной сети, форматирование ее настроек. Изучение правил сетевой безопасности.

ЗАДАНИЕ

1. Произведите поиск доступных беспроводных сетей, запишите результаты поиска в таблицу вида:

SSID сети	Тип беспроводной сети	Способ шифрования	Уровень сигнала

2. Подключитесь к открытой беспроводной сети tusur, определите какие ресурсы этой сети вам доступны.

3. Создайте собственную беспроводную сеть типа Ad hoc. Ответьте на следующие вопросы:

- Как изменяется уровень сигнала в этой сети при изменении расстояния между компьютерами, при появлении препятствий между компьютерами?
- Изменяется ли скорость работы сети.

Выполните в созданной сети передачу файла (файлов) с одного компьютера на другой.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие способы защиты беспроводных сетей вы знаете.
2. Чем отличается беспроводной маршрутизатор от точки доступа.
3. По какому протоколу передается информация в беспроводных сетях.

4. Для чего предназначен протокол RADIUS.

5. Почему в беспроводных сетях общего доступа применяется протокол DHCP.

6. Что такое SSID.

7. Можно ли средствами Windows определить канал, на котором передается информация в беспроводной сети. Как часто происходит смена канала?

8. Какие вопросы рассмотрены в стандарте IEEE 802.1x.

9. На какой частоте работают адаптеры стандарта IEEE 802.11g, какова максимальная скорость передачи данных этих адаптеров.

10. Какой может быть логическая топология беспроводной сети.

11. Какой метод доступа применяется в сетях Wi-Fi.

12. Каков максимальный диаметр сети Wi-Fi, от чего он зависит.

13. Почему ограничена мощность точек доступа, какова максимально допустимая мощность.

ОРГАНИЗАЦИЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ. ПРИМЕНЕНИЕ РЕЙТИНГОВОЙ СИСТЕМЫ

Самостоятельная работа распределяется по времени следующим образом: 32 часа — на изучение лекционного курса; оформление отчетов по лабораторным работам — 40 часов; выполнение творческого задания — 10 часов; подготовка к экзамену — 10 часов.

Лабораторные занятия выполняются согласно расписанию занятий. Текущий рейтинг за лабораторные работы, сдаваемые с опозданием, не выставляется согласно положению о рейтинге. Собеседование и прием экзамена проводится во время экзаменационной сессии.

Максимальный рейтинг по дисциплине составляет 120 баллов и определяется по табл. 6. В соответствии с положением, для получения оценки «отлично» требуется набрать не менее 100 баллов, «хорошо» — 80 баллов. Для допуска к экзамену требуется набрать не менее 60 баллов.

Таблица 6 — **Распределение рейтинга по элементам контроля**

Отчетные этапы	***Элементы контроля***	***Рейтинг***
1 контрольная точка	Лабораторные работы №№ 1 — 6	45
2 контрольная точка	Лабораторные работы №№ 7—12	45
Зачетная неделя		
	Посещение лекций	10
	Творческое задание	20
Всего		***120 мах***

Творческое задание выполняется по желанию самостоятельно. Тема творческого задания выбирается студентом самостоятельно по тематике курса. Тема творческого задания должна быть обязательно согласована с преподавателем.

ЗАКЛЮЧЕНИЕ

Описания к лабораторным работам находятся в учебной лаборатории и выдаются студенту для оформления отчета и подготовки к следующей работе.

Все лабораторные работы проводятся в вычислительной лаборатории с использованием ЭВМ. Операционные системы и соответствующее программное обеспечение установлено.

Индивидуальные задания по лабораторным работам выдаются студентам автоматически и помещаются в персональные каталоги студентов в ЛВС учебных классов. Имеются в свободном доступе электронные версии лабораторных работ и электронная документация по ним.

В учебно-методическом пособии на примерах рассмотрены основные вопросы передачи информации, способы коммутации, мультиплексирования, кодирования и адресации сетевых устройств. Практически изучаются вопросы установки, настройки и обслуживания сетевого оборудования.

УЧЕБНО-МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ

Основная литература

1. Норенков И.П., Трудоношин В.А. Телекоммуникационные технологии и сети. — 2-е изд., испр. и доп. — М.: Изд-во МГТУ им. Н.Э. Баумана, 2000. — 248 с.

2. Обрусник П.В. Эксплуатация и развитие компьютерных сетей и систем. Часть 2: Учебно-методическое пособие. — Томск: Томский межвузовский центр дистанционного образования, 2001. — 21 с.

Дополнительная литература

1. Танненбаум Э. Компьютерные сети: Пер. с англ. — 4-е изд. — СПб.: Питер, 2003. — 991 с.

2. Соловьева Л.Ф. Сетевые технологии: учебник-практикум. — СПб.: БХВ-Перербург, 2004. — 397 с.

3. Нанс Б. Компьютерные сети: Пер. с англ. — М.: Бином, 1995. — 400 с.

4. Вишневский В.М. Теоретические основы проектирования компьютерных сетей: Монография / РАН. Институт проблем передачи информации. — М.: Техносфера, 2003. — 506 с.

5. Корнеев В.В. Вычислительные системы. — М.: Гелиос АРВ, 2004. — 510 с.

6. Компьютерные сети. Принципы, технологии, протоколы. Учебник / В.Г. Олифер, Н.А. Олифер. — СПб.: Изд-во «Питер», 2000. — 672 с.