Министерство образования Республики Беларусь

Учреждение образования
«Гомельский государственный технический
университет имени П. О. Сухого»

Кафедра «Промышленная электроника»

А. В. Ковалев, Д. А. Литвинов, О. М. Ростокина

ИНТЕРФЕЙСЫ И УСТРОЙСТВА ТЕЛЕКОММУНИКАЦИЙ

ПРАКТИКУМ
по выполнению лабораторных работ
по одноименной дисциплине
для студентов специальности 1-53 01 07
«Информационные технологии и управление в технических системах»
дневной формы обучения

Электронный аналог печатного издания

Гомель 2021

УДК 621.391(075.8)
ББК 32.88я73
К56

Рекомендовано к изданию научно-методическим советом автоматизированных и информационных систем ГГТУ им. П. О. Сухого (протокол № 4 от 28.12.2020 г.)

Рецензент: доц. каф. «Информационные технологии» ГГТУ им. П. О. Сухого канд. техн. наук, доц. *В. И. Токочаков*

Ковалев, А. В.

К56 Интерфейсы и устройства телекоммуникаций : практикум по выполнению лабораторных работ по одноим. дисциплине для студентов специальности 1-53 01 07 «Информационные технологии и управление в технических системах» днев. формы обучения / А. В. Ковалев, Д. А. Литвинов, О. М. Ростокина. – Гомель : ГГТУ им. П. О. Сухого, 2021. – 48 с. – Систем. требования: PC не ниже Intel Celeron 300 МГц ; 32 Mb RAM ; свободное место на HDD 16 Mb ; Windows 98 и выше ; Adobe Acrobat Reader. – Режим доступа: https://elib.gstu.by. – Загл. с титул. экрана.

ISBN 978-985-535-462-9.

Рассмотрены аппаратные и программные средства кодирования и защиты информации в интерфейсах и в системах коммуникаций, особенности проектирования кабельных систем передачи информации и кодирования изображений в системах коммуникаций.

Для студентов специальности 1-53 01 07 «Информационные технологии и управление в технических системах» дневной формы обучения.

УДК 621.391(075.8)
ББК 32.88я73

ISBN 978-985-535-462-9

Лабораторная работа № 1

Методы обнаружения ошибок передачи данных. Циклический избыточный код (CRC)

Цель работы: изучить способы обнаружения ошибок передачи данных в системах телекоммуникаций.

1.1. Обнаружение ошибок

При передаче сообщений по низкокачественным каналам связи возможно внесение искажений в эти сообщения, и поэтому необходимо предусмотреть методы, с помощью которых приемник сможет определить, были ли ошибки при передаче. При обнаружении ошибок приемник может запросить повторную передачу сообщения. Для обнаружения применяются методы контрольных сумм. Передатчик подсчитывает некоторое значение, называемое контрольной суммой, являющееся функцией сообщения, и добавляет его в конец сообщения. Приемник использует ту же функцию для подсчета контрольной суммы и сравнивает полученное значение с записанным в конце сообщения. Например, в качестве контрольной суммы можно взять сумму байт сообщения по модулю 256.

Сообщение	06	23	04	
Сообщение с контрольной суммой	06	23	04	33
Сообщение после передачи	06	27	04	33

В этом примере возникла ошибка во втором байте, и вместо числа 23 было передано число 27. Такую ошибку приемник может определить, вычислив контрольную сумму (37) и сравнив ее с полученной (33). Если бы сбой произошел в четвертом байте – в самой контрольной сумме, то верное сообщение было бы забраковано. Но главная опасность в том, что ошибка может произойти как в самом сообщении, так и в контрольной сумме, причем таким образом, что контрольная сумма от неверного сообщения совпадет с неверно переданной контрольной суммой.

1.2. Циклический избыточный код

Циклический избыточный код (англ. *Cyclic Redundancy Code*, *CRC*) – алгоритм вычисления контрольной суммы, предназначенный для проверки целостности передаваемых данных. Алгоритм CRC обнаруживает все одиночные ошибки, двойные ошибки и ошибки в нечетном числе битов. Понятие циклических кодов достаточно широкое, однако на практике его обычно применяют для обозначения только одной разновидности, использующей циклический контроль (проверку) избыточности. В связи с этим в англоязычной литературе CRC часто расшифровывается как *Cyclic Redundancy Check*.

CRC некоторой последовательности вычисляется на основании другой (исходной) битовой последовательности. Главная особенность (и практическая значимость) значения CRC состоит в том, что оно однозначно идентифицирует исходную битовую последовательность и поэтому используется в различных протоколах связи, а также для проверки целостности блоков данных, передаваемых различными устройствами. Благодаря относительной простоте алгоритм вычисления CRC часто реализуется на аппаратном уровне.

Для вычисления CRC используют специальную арифметику по модулю 2. Действия, выполняемые во время вычисления CRC, являются арифметическими операциями без учета переносов (операции XOR).

С порождающим полиномом связан другой параметр – его степень, которая определяет количество битов, используемых для вычисления значения CRC. На практике наиболее распространены 8-, 16- и 32-битовые слова, что является следствием особенностей архитектуры современной вычислительной техники.

1.3. Прямой алгоритм реализации CRC

Прямой алгоритм вычисления контрольной суммы CRC можно представить следующей последовательностью:

1. Создается массив (регистр), заполненный нулями, равный по длине разрядности (степени) полинома.

2. Исходное сообщение дополняется нулями в младших разрядах, в количестве, равном числу разрядов полинома.

3. В младший разряд регистра заносится один старший бит сообщения, а из старшего разряда регистра выдвигается один бит.

4. Если выдвинутый бит равен «1», то производится инверсия битов (операция XOR, исключающее ИЛИ) в тех разрядах регистра, которые соответствуют единицам в полиноме.

5. Если в сообщении еще есть биты, переходим к шагу 3.

6. Когда все биты сообщения поступили в регистр и были обработаны этим алгоритмом, в регистре остается остаток от деления, который и является контрольной суммой CRC.

Назовем этот метод расчета CRC **методом побитового сдвига** или **простым методом**. Рисунок 1.1 иллюстрирует деление исходной последовательности битов на число (1)00000111 или многочлен $x^8 + x^2 + x^1 + x^0$.

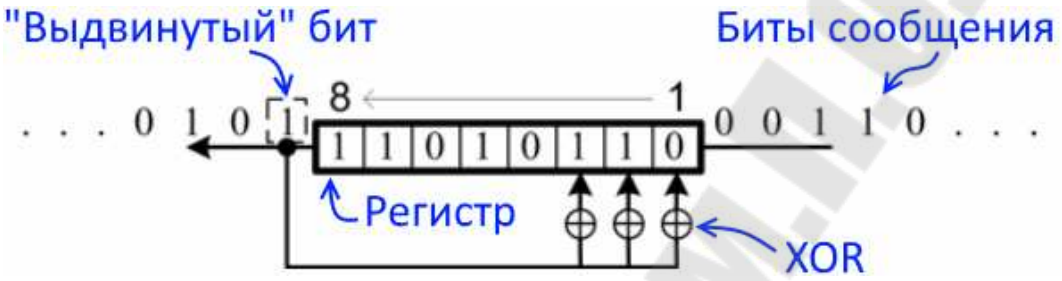


Рис. 1.1. Схематичное представление вычисления CRC на примере деления на многочлен $x^8 + x^2 + x^1 + x^0$

Обобщенный алгоритм получения контрольной суммы будет иметь вид:

Загрузим регистр нулевыми битами

Дополним хвостовую часть сообщения W нулевыми битами

While (пока еще есть необработанные биты)

Begin

Сдвинем регистр на 1 бит влево и поместим очередной

еще не обработанный бит из сообщения в 0 позицию регистра.

If (из регистра был выдвинут бит со значением «1»)

Регистр = Регистр **XOR** Полином

End

Теперь в регистре содержится остаток

Для программной реализации более удобно использовать следующий алгоритм:

CRC-8

CRC8INIT – начальное (стартовое) значение слова

CRC8POLY – значение, задающее полином

описание переменных

int8 crc, i,j, inbyte, hbit

crc = CRC8INIT

```
i=0
while (i<size)
        {
        inbyte=data[i];  взять i байт сообщения
        for(j = 0; j < 8; j++) цикл для перебора все битов в байте данных
        {
        hbit=(crc XOR inbyte) & 0x80; выделяем старший бит
        crc сдвиг влево на 1 разряд
        inbyte сдвиг влево на 1 разряд
        коррекция регистра CRC-регистр по правилам двоичной арифметики без переносов
        if(hbit) crc = crc  XOR CRC8POLY
        }
        i++
}
```

CRC-16

CRC16INIT – начальное (стартовое) значение слова
CRC16POLY – значение, задающее полином
описание переменных

```
int16 crc, inbyte, hbit
int8 i,j
crc = CRC16INIT
i=0
while (i<size)
{
```

взять i байт сообщения и сдвинуть влево на 8 (для того чтобы данные при расчете влияли на все 16 бит)

```
inbyte=(unsigned int)data[i] и сдвиг влево на 8 разрядов;
for(j = 0; j < 8; j++)  цикл для перебора все битов в байте данных
{
        hbit=(crc XOR inbyte)&0x8000;  выделяем старший бит
        crc сдвиг влево на 1 разряд
        inbyte сдвиг влево на 1 разряд
        коррекция регистра (CRC-регистр) по правилам двоичной арифметики без переносов
        if(hbit) crc = crc XOR CRC16POLY;
}
i++;
}
```

Представленные реализации алгоритмов являются не оптимальными и могут быть улучшены при реализации алгоритма. Такая реализация самая простая и понятная для понимания, однако она имеет ряд недостатков: она требует большого числа машинных операций и выполняется достаточно медленно, что существенно ограничивает ее применение в современных системах связи.

1.4. Контрольное задание

Разработать программу, реализующую следующие методы обнаружения ошибок передачи данных:

1) контроль четности – сумма всех битов сообщения по модулю 2;

2) контрольная сумма (размер – 1 байт) – сумма всех байтов сообщения (старшие разряды суммы не учитываются);

3) циклический избыточный код (CRC) согласно варианту (табл. 1.1).

Исходными данными являются:

1) длина сообщения;

2) последовательность байтов сообщения.

Для проверки правильности вычисления CRC необходимо добавить его значение к концу сообщения и вычислить CRC полученной последовательности, которая должна равняться нулю.

Таблица 1.1

Варианты заданий

Название	**Образующий полином**	**Представление: нормальное/ реверсивное**
CRC-8-CCITT	$x^8 + x^2 + x^1 + x + 1$ (ATM HEC), ISDN Header Error Control and Cell Delineation ITU-T I.432.1 (02/99)	0x07 / 0xE0
CRC-8-Dallas/Maxim	$x^8 + x^5 + x^4 + 1$ (1-Wire bus)	0x31 / 0x8C
CRC-8	$x^8 + x^7 + x^6 + x^4 + x^2 + 1$	0xD5 / 0xAB
CRC-8-SAE J1850	$x^8 + x^4 + x^3 + x^2 + 1$	0x1D / 0xB8
CRC-8-WCDMA	$x^8 + x^7 + x^4 + x^3 + x + 1$	0x9B / 0xD9
CRC-16-IBM	$x^{16} + x^{15} + x^1 + 1$ (Modbus, USB, ANSI X3.28, many others, also known as CRC-16 and CRC-16-ANSI)	0x8005 / 0xA001

Окончание табл. 1.1

Название	**Образующий полином**	**Представление: нормальное/ реверсивное**
CRC-16-CCITT	$x^{16}+x^{12}+x^5+1$ (X.25, V.41, HDLC, XMODEM, Bluetooth, SD, many others, known as CRC-CCITT)	0x1021 / 0x8408
CRC-16-T10-DIF	$x^{16}+x^{15}+x^{11}+x^9+x^8+x^7+x^5+x^4+x^2+1$ (SCSI DIF)	0x8BB7 / 0xEDD1
CRC-16-DNP	$x^{16}+x^{13}+x^{12}+x^{11}+x^{10}+x^8+x^6+x^5+x^2+1$ (DNP, IEC 870, M-Bus)	0x3D65 / 0xA6BC
CRC-16-DECT	$x^{16}+x^{10}+x^8+x^7+x^3+1$ (cordless telephones)	0x0589 / 0x91A0
CRC-32-IEEE 802.3	$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+$ $+x^7+x^5+x^4+x^2+x+1$ (V.42, MPEG-2, PNG, POSIX cksum)	0x04C11DB7 / 0xEDB88320
CRC-32C (Castagnoli)	$x^{32}+x^{28}+x^{27}+x^{26}+x^{25}+x^{23}+x^{22}+x^{20}+x^{19}+$ $+x^{18}+x^{14}+x^{13}+x^{11}+x^{10}+x^9+x^8+x^6+1$ (iSCSI, G.hn payload)	0x1EDC6F41 / 0x82F63B78
CRC-32K (Koopman)	$x^{32}+x^{30}+x^{29}+x^{28}+x^{26}+x^{20}+x^{19}+x^{17}+x^{16}+$ $+x^{15}+x^{11}+x^{10}+x^7+x^6+x^4+x^2+x+1$	0x741B8CD7 / 0xEB31D82E
CRC-32Q	$x^{32}+x^{31}+x^{24}+x^{22}+x^{16}+x^{14}+x^8+x^7+x^5+$ $+x^3+x+1$ (aviation, AIXM)	0x814141AB / 0xD5828281

1.5. Контрольные вопросы

1. Способы обнаружения ошибок передачи данных.
2. Посимвольный контроль четности.
3. Поблочный контроль четности.
4. Вычисление контрольных сумм.
5. Контроль циклическим избыточным кодом CRC.
6. Общий алгоритм расчета CRC.
7. В чем принципиальная особенность CRC-арифметики?
8. Основные параметры CRC. Порождающий полином.
9. Прямой алгоритм вычисления контрольной суммы CRC.

Лабораторная работа № 2

Исследование скремблеров и дескремблеров

Цель работы: изучить алгоритмы построения скремблеров и дескремблеров.

2.1. Скремблирование

Скремблирование – это обратимое преобразование цифрового потока без изменения скорости передачи с целью получения свойств, близких к свойствам случайной последовательности. Исходное сообщение можно восстановить, применив обратный алгоритм.

Применительно к телекоммуникационным системам скремблирование повышает надежность синхронизации устройств, подключенных к линии связи, и уменьшает уровень помех, излучаемых на соседние линии многожильного кабеля. Есть и иная область применения скремблеров – защита передаваемой информации от несанкционированного доступа.

Для синхронной передачи двоичный сигнал должен удовлетворять двум основным требованиям:

– частота смены символов (1, 0) должна обеспечивать надежное выделение тактовой частоты непосредственно из принимаемого сигнала;

– спектральная плотность мощности передаваемого сигнала должна быть по возможности постоянной и сосредоточенной в заданной области частот с целью снижения взаимного влияния каналов.

Одним из способов обработки двоичных посылок, удовлетворяющим данным требованиям, является скремблирование (*scramble* – перемешивание). После скремблирования появление «1» и «0» в выходной последовательности примерно равновероятно. Скремблирование также может использоваться для определенной защиты передаваемой информации и для идентификации абонентов.

Скремблирование широко применяется во многих видах систем связи для улучшения статистических свойств сигнала. Обычно скремблирование осуществляется на последнем этапе цифровой обработки непосредственно перед модуляцией (рис. 2.1).

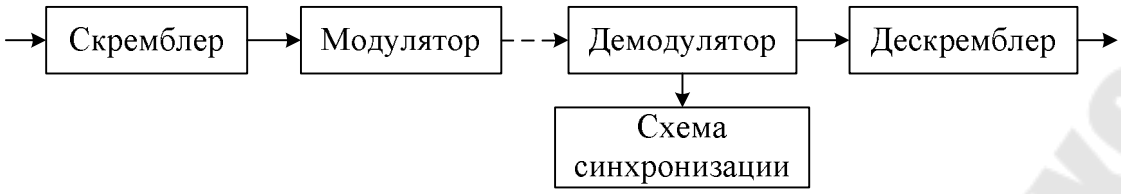


Рис. 2.1. Схема включения скремблера и дескремблера в канал связи

Скремблирование производится на передающей стороне с помощью устройства – скремблера, реализующего логическую операцию суммирования по модулю 2 исходного и преобразующего псевдослучайного двоичного сигнала. На приемной стороне осуществляется обратная операция – восстановление устройством, называемым дескремблером. Дескремблер выделяет из принятой последовательности исходную последовательность.

Основной частью скремблера является генератор псевдослучайной последовательности (ПСП) в виде линейного n-каскадного регистра с обратными связями, формирующий последовательность максимальной длины $2^n - 1$.

Различают два основных типа скремблеров и дескремблеров – самосинхронизирующиеся (СС) и с установкой (аддитивные). В литературе также можно встретить другие названия – скремблеры с неизолированным и изолированным от линии связи генераторами псевдослучайных последовательностей.

Особенностью самосинхронизирующегося скремблера (СС-скремблера) (рис. 2.2) является то, что он управляется скремблированной последовательностью, т. е. той, которая передается в канал. Поэтому при данном виде скремблирования не требуется специальной установки состояний скремблера и дескремблера; скремблированная последовательность записывается в регистры сдвига скремблера и дескремблера, устанавливая их в идентичное состояние. При потере синхронизма между скремблером и дескремблером время восстановления синхронизма не превышает числа тактов, равного числу ячеек регистра скремблера.

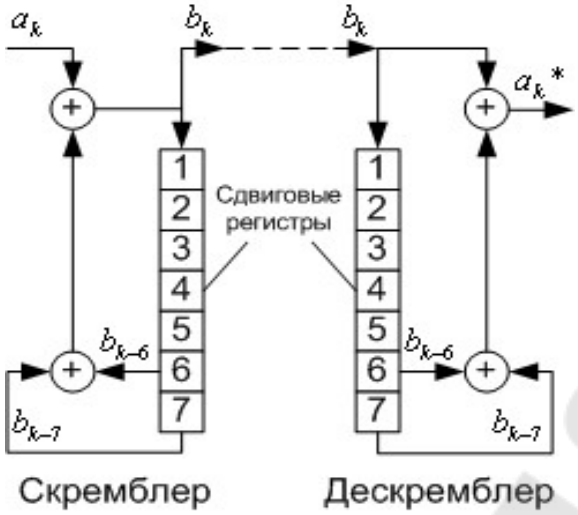


Рис. 2.2. Самосинхронизирующиеся скремблер и дескремблер

На приемной стороне выделение исходной последовательности происходит путем сложения по модулю 2 принятой скремблированной последовательности с последовательностью на выходе сдвигового регистра. Например, для схемы на рис. 2.2 входная последовательность a_k с помощью скремблера в соответствии с соотношением $b_k = a_k XOR\left(b_{k-6} XOR b_{k-7}\right)$ преобразуется в посылаемую двоичную последовательность b_k. В приемнике из этой последовательности таким же регистром сдвига, как на приеме, формируется последовательность $a_k^* = b_k XOR\left(b_{k-6} XOR b_{k-7}\right)$. Эта последовательность на выходе дескремблера идентична первоначальной последовательности a_k.

Как следует из принципа действия схемы, при одной ошибке в последовательности b_k ошибочными получаются также последующие шестой и седьмой символы (в данном примере). В общем случае влияние ошибочно принятого бита будет сказываться α раз, где α – число обратных связей. Таким образом, СС скремблер-дескремблер обладает свойством размножения ошибок. Данный недостаток СС скремблера-дескремблера ограничивает число обратных связей в регистре сдвига; практически это число не превышает $\alpha = 2$.

Недостатки, присущие СС скремблеру-дескремблеру, практически отсутствуют при аддитивном скремблировании (рис. 2.3), однако этот тип скремблеров-дескремблеров требует предварительной идентичной установки состояний регистров скремблера и дескремблера. В скремблере с установкой (АД-скремблере), как и в СС-скремблере, производится суммирование входного сигнала и ПСП, но результи-

рующий сигнал не поступает на вход регистра. В дескремблере скремблированный сигнал также не проходит через регистр сдвига, поэтому размножения ошибок не происходит.

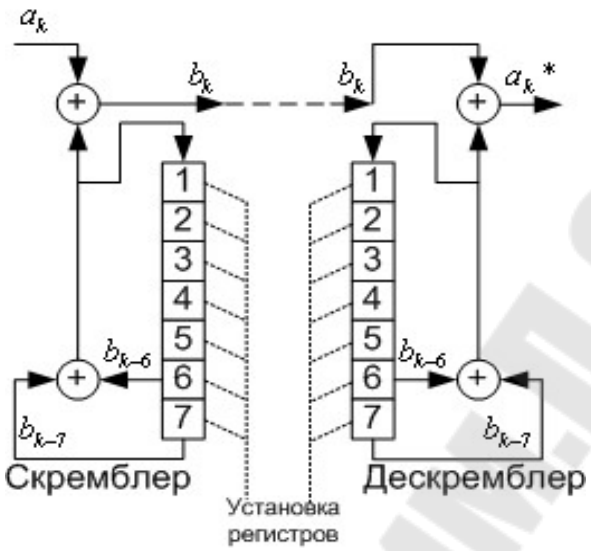


Рис. 2.3. Аддитивные скремблер и дескремблер

Суммируемые в скремблере последовательности независимы, поэтому их период всегда равен наименьшему общему кратному величин периодов входной последовательности и ПСП и критическое состояние отсутствует. Отсутствие эффекта размножения ошибок и необходимости в специальной логике защиты от нежелательных ситуаций делают способ аддитивного скремблирования предпочтительнее, если не учитывать затрат на решение задачи фазирования скремблера и дескремблера. В качестве сигнала установки в ПСП используют сигнал цикловой синхронизации.

2.2. Контрольное задание

Разработать программу, реализующую работу скремблера и дескремблера согласно варианту (табл. 2.1). Исходными данными являются:

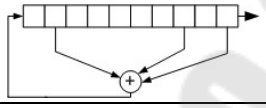
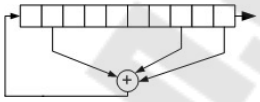
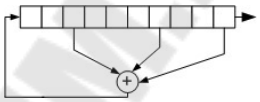
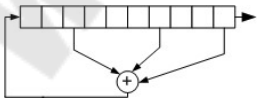
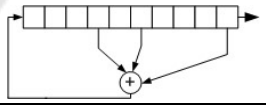
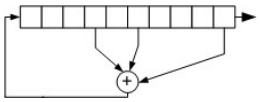
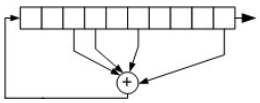
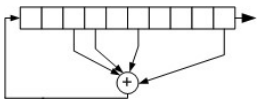
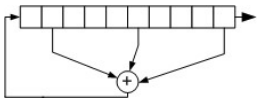
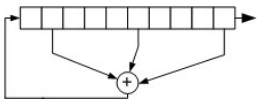
1) длина сообщения;

2) последовательность байтов сообщения.

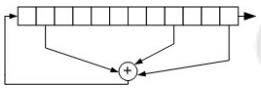
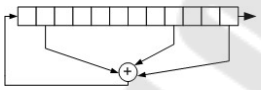
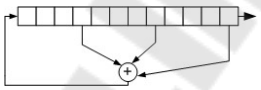
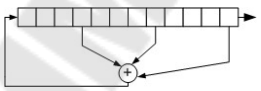
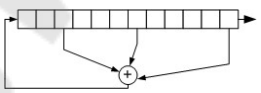
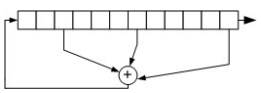
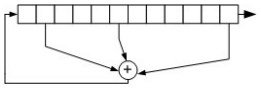
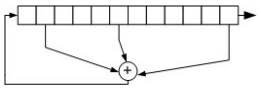
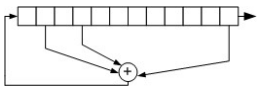
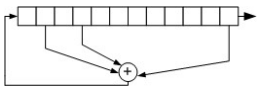
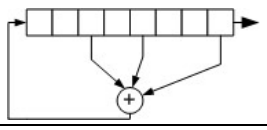
Проверить правильность функционирования написанных алгоритмов путем скремблирования исходной последовательности и дескремблирования полученной последовательности.

Таблица 2.1

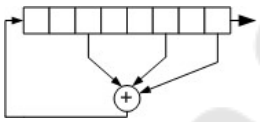
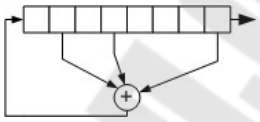
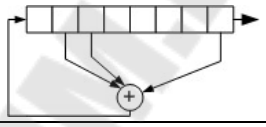
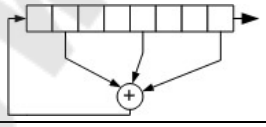
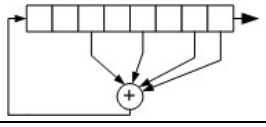
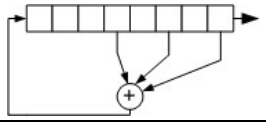
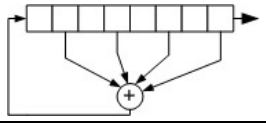
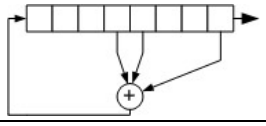
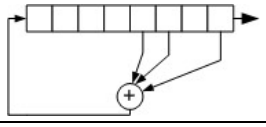
Варианты заданий

Номер варианта	**Тип скремблера-дескремблера**	**Разряд-ность**	**Сдвиговый регистр**	**Исходное значение сдвигового регистра**
1	Самосинхронизирующийся	10	10100000101 	–
2	Аддитивный	10	10100000101 	0010010110
3	Самосинхронизирующийся	10	10010001001 	–
4	Аддитивный	10	10010001001 	0000100100
5	Самосинхронизирующийся	10	10001010001 	–
6	Аддитивный	10	10001010001 	1111111111
7	Самосинхронизирующийся	10	10011010001 	–
8	Аддитивный	10	10011010001 	1111111111
9	Самосинхронизирующийся	10	10100010001 	–
10	Аддитивный	10	10100010001 	1111111111

Продолжение табл. 2.1

Номер варианта	**Тип скремблера-дескремблера**	**Разряд-ность**	**Сдвиговый регистр**	**Исходное значение сдвигового регистра**
11	Самосинхронизирующийся	12	1010000001001 	–
12	Аддитивный	12	1010000001001 	000101001101
13	Самосинхронизирующийся	12	1000100010001 	–
14	Аддитивный	12	1000100010001 	110101001001
15	Самосинхронизирующийся	12	1001000100001 	–
16	Аддитивный	12	1001000100001 	110001010001
17	Самосинхронизирующийся	12	1010001000001 	–
18	Аддитивный	12	1010001000001 	001011010100
19	Самосинхронизирующийся	12	1010100000001 	–
20	Аддитивный	12	1010100000001 	100010101101
21	Самосинхронизирующийся	8	100101001 	–

Окончание табл. 2.1

Номер варианта	**Тип скремблера-дескремблера**	**Разряд-ность**	**Сдвиговый регистр**	**Исходное значение сдвигового регистра**
22	Аддитивный	8	100100101 	00110010
23	Самосинхронизирующийся	8	101010001 	–
24	Аддитивный	8	101100001 	00101010
25	Самосинхронизирующийся	8	101001001 	–
26	Аддитивный	8	100101011 	00100011
27	Самосинхронизирующийся	8	100010101 	–
28	Аддитивный	8	101010101 	10010110
29	Самосинхронизирующийся	8	100011001 	–
30	Аддитивный	8	100001101 	11100111

2.3. Контрольные вопросы

1. Что такое скремблер и дескремблер?
2. Для каких целей используют скремблеры и дескремблеры?
3. Какие типы скремблеров и дескремблеров вам известны?
4. Какие преимущества и недостатки самосинхронизирующихся скремблеров и дескремблеров вам известны?
5. Какие преимущества и недостатки аддитивных скремблеров и дескремблеров вам известны?

Лабораторная работа № 3

Изучение интерливеров и деинтерливеров

Цель работы: получение навыков построения интерливеров и деинтерливеров.

3.1. Интерливеры и деинтерливеры

На пути распространения информации в радиоинтерфейсах обычно случаются различные препятствия, которые ведут к отражениям сигнала и изменению его траектории. В результате может сложиться ситуация, когда к приемнику будут поступать не одна, а сразу несколько сдвинутых по времени копий исходного сигнала с разными амплитудами. Само по себе это явление не ведет к большим проблемам, так как существуют достаточно эффективные методы борьбы. Однако может сложиться ситуация, когда две копии сигнала придут в противофазе. Это означает, что копия сигнала может задержаться на промежуток времени кратный периоду сигнала. В таком случае два луча сигнала могут сложиться в приемнике и нейтрализовать друг друга (рис. 3.1).

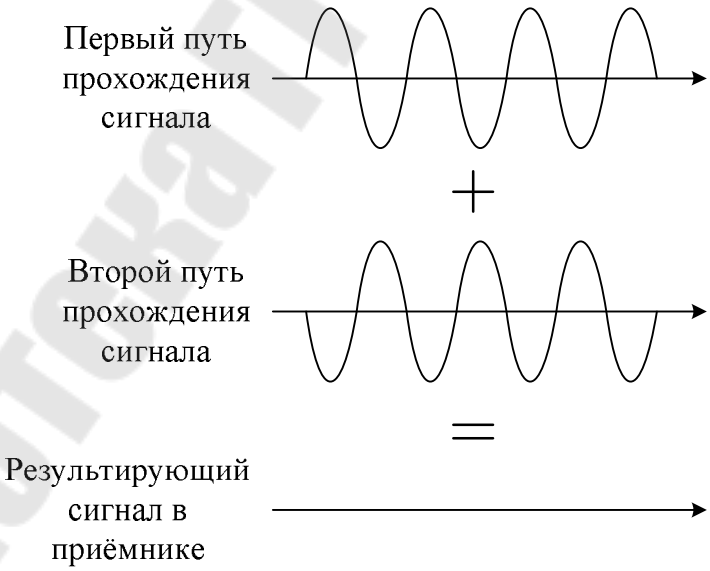


Рис. 3.1. Эффект замирания сигнала

Если окажется, что эти два луча в сумме несли весомую энергию сигнала, то это может привести к увеличению числа ошибок и снижению качества канала связи. Это явление получило название

«замирание» сигнала, т. е. сигнал вроде как перестает на время поступать между источником и приемником.

Выделяют две основные разновидности замираний в зависимости от эффекта, оказываемого ими, и их причины: быстрые и медленные замирания.

Некоторые схемы помехоустойчивого кодирования, такие как сверточный код, обеспечивают хорошую защиту против разрозненных ошибок. В случае достаточно медленных замираний сигнала возможны непрерывные ошибки передачи данных (пакеты ошибок). Для борьбы с пакетами ошибок используется интерливер (перемежитель), который осуществляет перестановку входных бит по заранее заданному закону.

Интерливер (перемежитель, от англ. *Interleaver*) – блок, реализующий перемежение – один из способов борьбы с ошибками. Предназначен для борьбы с пакетированием ошибок путем их разнесения во времени и (или) частоте. Использует перемешивание (перемежение) символов передаваемой последовательности на передаче и восстановление ее исходной структуры на приеме. Может использоваться как самостоятельно, так и вместе с помехоустойчивым кодом, являясь в таком случае его составным компонентом.

Благодаря перемежению на входе декодера ошибки равномерно распределяются во времени и (или) частоте, в идеале образуя поток независимых ошибок.

Существует несколько типов устройств интерливеров:

1. *Периодические интерливеры*. Относительно просты и используются в большинстве случаев. Подразделяются на блоковые и сверточные:

а) блоковые устройства являются двумерным массивом, запись данных в который осуществляется по столбцам, а считывание – по строкам. На приеме запись и считывание осуществляются в обратном порядке. Математически это соответствует транспонированию матрицы;

б) сверточные устройства обычно реализуются в виде N регистров сдвига разной длины, в которые с помощью коммутатора последовательно записываются данные.

2. *Псевдослучайные*. Сложнее в реализации, но обладают лучшими характеристиками. Используются в турбокодах, в них цель перемежителя состоит в том, чтобы предложить каждому кодеру некоррелированную или случайную версию информации, в результате чего проверочные биты каждого кодера становятся независимыми. Сте-

пень независимости этих проверочных битов является по существу функцией типа длины/глубины интерливера.

Принцип работы блочного интерливера изображен на рис. 3.2. Входные биты записываются в матрицу размерности $N \times M$ строка за строкой. Считывание данных из этой матрицы выполняется по очереди столбец за столбцом, как это изображено на рис. 3.2.

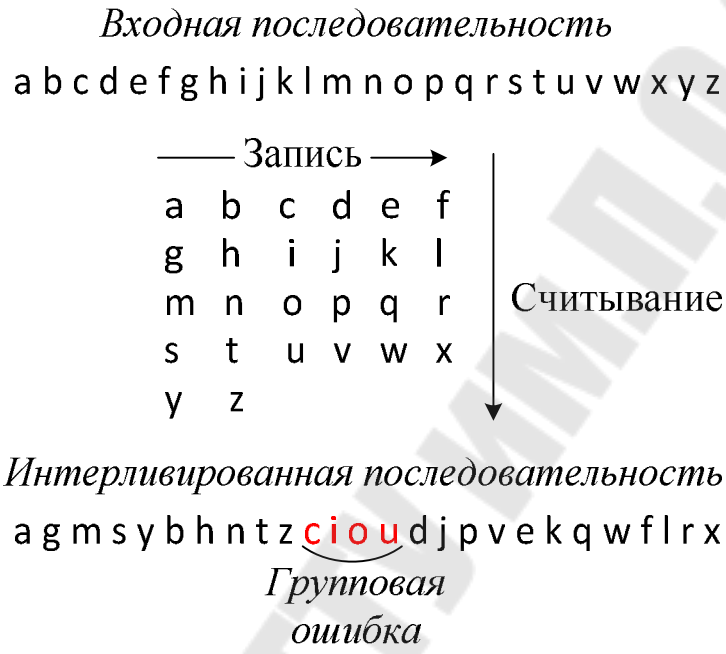


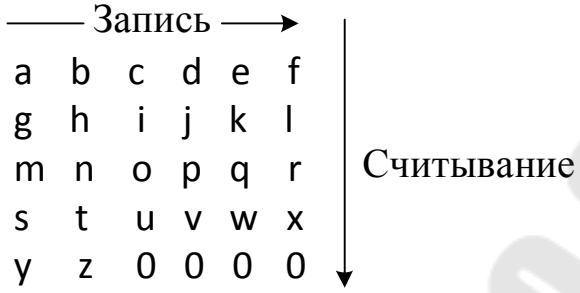
Рис. 3.2. Принцип работы блочного интерливера

В некоторых случаях, если длина интерливируемой последовательности K не кратна длине строки интерливирующей матрицы, для удобства оставшиеся пустые места могут заполняться нулями (филлерами). В этом случае после интерливера получается последовательность длиной $K + d$, где $d = N \times M - K$.

Принцип работы блочного интерливера с филлерами изображен на рис. 3.3.

Входная последовательность

a b c d e f g h i j k l m n o p q r s t u v w x y z



Интерливированная последовательность

a g m s y b h n t z c i o u d j p v 0 e k q w 0 f l r x 0

Групповая ошибка

Ошибки в деинтерливированной последовательности

a b c d e f g h i j k l m n o p q r s t u v w x y z

Рис. 3.3. Принцип работы блочного интерливера с филлерами

Иногда выполняют еще и дополнительное перемежение столбцов матрицы. Пусть дана некоторая последовательность y_0, y_1, …, $y_{N\times M-1}$, которую надо интерливировать. В этом случае матрица интервирования будет иметь вид:

$$\begin{bmatrix} y_0 & y_1 & y_2 & \cdots & y_{M-1} \\ y_M & y_{M+1} & y_{M+2} & \cdots & y_{2M-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ y_{(N-1)\times M} & y_{(N-1)\times M+1} & y_{(N-1)\times M+2} & \cdots & y_{(N\times M-1)} \end{bmatrix}.$$

Тогда если определен некоторый закон $\left[P(0), P(1), \ldots, P(M-1)\right]$ перемежения, то матрица интерливирования будет иметь вид:

$$\begin{bmatrix} y_{P(0)} & y_{P(1)} & y_{P(2)} & \cdots & y_{P(M-1)} \\ y_{P(0)+M} & y_{P(1)+M} & y_{P(2)+M} & \cdots & y_{P(M-1)+M} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ y_{P(0)+(N-1)\times M} & y_{P(1)+(N-1)\times M} & y_{P(M-1)+(N-1)\times M} & \cdots & y_{P(M-1)+(N-1)\times M} \end{bmatrix}.$$

И интерливированная последовательность запишется следующим образом: $y_{P(0)}$, $y_{P(0)+M}$, …, $y_{P(0)+(N-1)\times M}$, $y_{P(1)}$, $y_{P(1)+M}$, …, $y_{P(1)+(N-1)\times M}$, $y_{P(2)}$, $y_{P(2)+M}$, …, $y_{P(2)+(N-1)\times M}$, …, $y_{P(M-1)}$, $y_{P(M-1)+M}$, …, $y_{P(M-1)+(N-1)\times M}$.

Пример такого перемежения изображен на рис. 3.4.

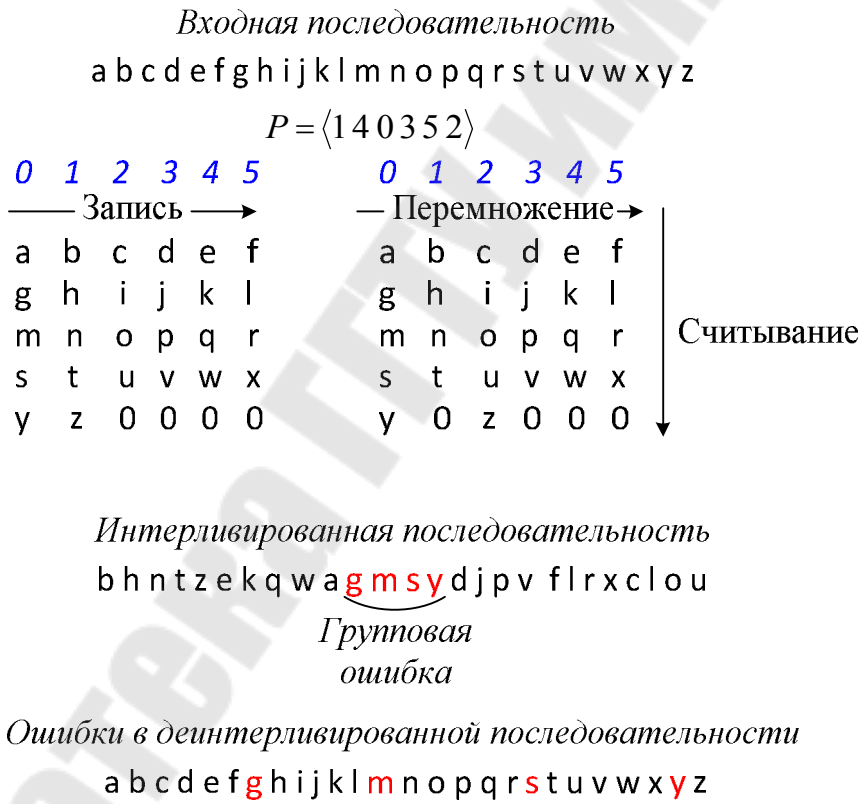


Рис. 3.4. Принцип работы блочного интерливера с филлерами и перемежение столбцов

3.2. Контрольное задание

Разработать программу, реализующую работу интерливера и деинтерливера согласно варианту (табл. 3.1). Исходными данными являются:

– длина сообщения в байтах $K = (N \times M) / 8 \times P$, где P – число пакетов, N – число строк в пакете, M – число столбцов в пакете;

– последовательность байтов сообщения;

– последовательность $\left[P(0), P(1), \ldots, P(M-1)\right]$, задает закон перемежения столбцов в интерливирующей матрице.

Проверить правильность функционирования написанных алгоритмов путем интерливирования исходной последовательности и деинтерливирования полученной ранее последовательности.

Рекомендации к реализации алгоритма работы интерливера:

1. Один интервелированный пакет формируется из $L = (N \times M) / 8$, где $L = (N \times M) / 8$ – байт исходного сообщения.

2. Для удобства работы с битами использовать битовую структуру:

```
typedef union{
    unsigned short data16;
    struct{
        unsigned b0     :1;
        unsigned b1     :1;
        unsigned b2     :1;
        unsigned b3     :1;
        unsigned b4     :1;
        unsigned b5     :1;
        unsigned b6     :1;
        unsigned b7     :1;
        unsigned b8     :1;
        unsigned b9     :1;
        unsigned b10    :1;
        unsigned b11    :1;
        unsigned b12    :1;
        unsigned b13    :1;
        unsigned b14    :1;
        unsigned b15    :1;
    };
    byte16_type;
```

3. Пример алгоритма интерливера без перемежения столбцов для $N = 4$, $M = 10$:

```
#define M 10
#define N 4
#define P 2
#define L ((N*M)/8)
#define K (L*P)

void interliver(unsigned char *in_data, short int *out_data)
{
    unsigned char i, j, row, col;
    byte16_type data;
    row = col = 0;
    data.data16 = 0x0000;
    for(i = 0; i <L; i++)
    {
       for(j = 0; j < 8; j++)
       {
          // формирование строки матрицы
          data.data16>>=1;
          data.b9 = in_data[i]&0x01;
          in_data[i]>>=1;
          col++;
          if(col==M)  // если сформирована строка матрицы
          {
           col=0;
           out_data[row] = data.data16;  // сохраняем ее
           row++;     // и переходим к формированию следующей
           data.data16 = 0x0000;
           }
       }
     }
    }
```

Пример вызова функции:

```
short int out_paket[N];
unsigned char  msg[K];
unsigned char  i; // номер пакета;

interliver(&msg[L*i], out_paket);
```

Таблица 3.1

Варианты заданий

Номер задания	***N***	***M***	$\left[P(0), P(1), \ldots, P(M-1)\right]$
1	4	10	<0 2 3 1 4 5 7 8 9 6>
2	4	10	<0 2 1 4 3 5 7 6 8 9>
3	4	10	<0 9 1 8 2 7 3 6 4 5 >
4	4	10	<8 1 3 4 6 7 9 5 2 0>
5	4	10	<7 2 9 5 0 6 3 1 8 4>
6	4	10	<9 8 2 7 3 6 4 0 5 1>
7	4	10	<5 6 4 7 3 8 2 9 1 0>
8	4	10	<2 9 1 4 8 6 7 0 5>
9	4	10	<4 8 2 6 0 9 3 7 1 5>
10	4	10	<9 8 7 6 5 4 3 2 1 0>
11	2	12	<9 7 5 3 1 0 2 11 4 6 8 10>
12	2	12	<5 2 6 0 11 7 3 8 1 9 4 10>
13	2	12	<8 1 7 0 6 3 10 5 2 11 4 9>
14	4	12	<11 0 6 3 5 1 7 10 4 8 2 9>
15	4	12	<7 2 6 8 3 0 9 4 1 10 5 11>
16	4	12	<5 0 7 1 9 2 11 3 10 8 4 6>
17	4	12	<6 2 4 10 3 8 0 9 11 5 1 7>
18	6	12	<10 3 8 5 7 11 2 6 1 4 0 9>
19	6	12	<2 6 3 7 1 8 11 4 9 0 10 5>
20	6	12	<11 10 9 8 7 6 5 4 3 2 1 0>
21	3	8	<0 5 1 4 2 6 3 7>
22	3	8	<7 5 3 1 0 2 4 6>
23	3	8	<5 2 4 0 7 6 3 1>
24	4	8	<2 7 5 1 4 0 3 6>
25	4	8	<0 7 3 5 6 2 4 1>
26	4	8	<6 0 3 7 1 4 2 5>
27	4	8	<2 0 3 5 7 1 4 6>
28	5	8	<3 0 4 6 2 7 1 5>
29	5	8	<4 7 1 3 6 0 2 5>
30	5	8	<7 6 5 4 3 2 1 0>

3.3. Контрольные вопросы

1. Что такое эффект замирания сигнала? Медленные и быстрые замирания.
2. Что такое интерливер?
3. Какие существуют типы устройств интерливеров?
4. Принцип работы блочного интерливера.
5. Принцип работы блочного интерливера с перемежением столбцов.

Лабораторная работа № 4

Алгоритмы обработки звуковых и видеосигналов для передачи их в системах телекоммуникаций. Форматы изображений

Цель работы: освоение основных приемов работы с изображениями в системах телекоммуникаций.

4.1. Теоретические сведения

Каждый пиксель растрового изображения содержит информацию о цвете. Для кодирования рисунок разбивают на небольшие одноцветные части. Все цвета, использованные в изображении, нумеруют и для каждой части записывают номер ее цвета. Запомнив последовательность расположения частей и номер цвета для каждой части, можно однозначно описать любой рисунок. Цветовая информация может занимать от одного до тридцати двух битов, в зависимости от глубины цвета. Цветовые модели могут быть различны.

Модель *CMYK* – используется для подготовки печатных изображений. Отличается тем, что изображения видят в отраженном свете, т. е. чем больше краски положено, тем больше цвета они поглощают. Цветовыми компонентами такой модели являются не основные цвета, а те, которые получают вычитанием основных цветов из белого.

Например: Белый – Синий = Красный + Зеленый.

Цветовая модель *HSB* – наиболее удобна для человека. В ней компонентами цвета являются: *Hue* – тон – характеризует конкретный оттенок цвета; *Saturation* – насыщенность (интенсивность) цвета; *Brightness* – характеризует примесь черного цвета в цвете. Эта модель используется в редакторах, направленных на создание изображений.

Цветовая модель *LAB* – принята в качестве международного цветового стандарта. Эта модель использует три компонента: *Lightness* – яркость, *Chroma* – интенсивность, которые вместе составляют информацию об освещенности *Luminance*, в изображении, содержащуюся в канале *L*. Канал *A* хранит информацию о тонах от зеленого до пурпурного, а информация о тонах от голубого до желтого хранится в канале *B*.

Цветовая модель *RGB* – цвета формируются посредством смешивания красного, синего и зеленого цветов. Любой цвет в этой модели представлен тремя числами, описывающими величину каждой цветовой

составляющей. Черный цвет образуется, когда интенсивность всех трех составляющих равна 0, а белый – когда она максимальна.

Формат *BMP* – собственный формат системы Windows, используется для хранения данных без потери качества, глубина цвета изображения – до 24 бит. Не использует систему цветокоррекции.

Формат *TIF* (*TIFF*) – фактически стандарт для подготовки изображений в полиграфии. Глубина цвета изображения – до 24 бит. В файле формата изображение может храниться в моделях *CMYK*, *RGB*, *LAB*. Разрешение – количество пикселов растрового изображения, приходящихся на дюйм. Значение разрешения сохраняется в файле при экспорте и используется графическими пакетами, чтобы изображение имело те же размеры, что и в исходном документе.

Формат *JPEG* – основная особенность формата является высокая степень сжатия данных, достигаемая за счет сжатия с потерями (теряются мелкие детали, появляется муар).

Формат *GIF* – уплотненный формат с глубиной цвета – 256. Сжатие включено постоянно, так как он предназначен для размещения изображений в сети Internet. Особенность формата – возможность создания анимированных изображений.

Формат *PSD* – собственный формат Photoshop. Он позволяет записывать готовое растровое изображение со многими слоями, дополнительными цветовыми каналами, масками и др.

Системы сжатия изображений можно разделить на два больших класса: сжатие изображений без потери качества и сжатие с потерей качества.

При сжатии без потерь восстановленные на приемной стороне данные в точности совпадают с входным сигналом кодера. Для сжатия изображений без потерь можно использовать известные из теории информации алгоритмы кодирования источников.

Эффективность кодирования обычно оценивается в битах на отсчет или в битах в секунду. Распространенным критерием качества при кодировании с потерями является отношение *сигнал*/*шум* (*SNR*) на выходе декодера

$$SNR = 10\log_{10}\left(\frac{E_{ipn}}{E_n}\right), \text{ дБ},$$

где E_{ipn} – энергия входного сигнала кодера; E_n – энергия шумового сигнала.

В случае обработки изображений вместо *SNR* часто используется пиковое отношение *сигнал*/*шум*, которое определяется по формуле

$$SNR = 10\log_{10}\left(\frac{255}{E_n}\right),$$

где 255 – это максимальное десятичное значение беззнакового 8-битового числа, соответствующего одной точке черно-белого изображения.

Рассмотрим стандартные форматы, используемые для представления цветных изображений. В цифровой форме любое изображение представляется в виде двумерного поля отсчетов (точек), называемых пикселями. Таким образом, изображение описывается тремя цветовыми компонентами, каждая из которых представляет собой прямоугольный массив чисел. Значение интенсивности компоненты в одной точке характеризуется целым числом из интервала 0–255 и для ее хранения отводится 1 байт. Всего изображение размера $M \times N$ в формате *RGB* занимает $3MN$ байт в памяти компьютера.

В типичных изображениях в формате *RGB* имеется существенная корреляция между цветными компонентами, и с точки зрения сжатия изображений формат *RGB* является заведомо избыточным. Как известно, в стандартах телевизионного вещания используется другое представление изображений, при котором также применяются три компоненты сигнала, но эти три компоненты почти некоррелированы друг с другом. Компоненты *R*, *G* и *B* преобразуются в яркостную компоненту *Y* и две цветоразностных компоненты *U* и *V* формата *YUV*.

Преобразование формата *RGB* в формат *YUV* выполняется по формулам:

$$Y = 0{,}299R + 0{,}578G + 0{,}114B;\ U = (B - Y)0{,}5673 + 128;$$

$$V = (R - Y)0{,}7132 + 128.$$

Обратное преобразование выполняется по формулам:

$$G = Y - 0{,}714\ (V - 128) - 0{,}334\ (U - 128);$$

$$R = Y + 1{,}402\ (V - 128);\ B = Y + 1{,}772\ (U - 128).$$

В формате *YUV* компоненты слабо коррелированы. Более того, так как большая часть информации сосредоточена в яркостной компоненте, то мы теряем мало информации, если выполним децимацию (прореживание) компонент *U* и *V* с коэффициентом 2. При таком прореживании четыре соседние точки изображения, образующие квадрат размера 2×2, описываются четырьмя значениями яркостных компо-

нент *Y*, одним значением компоненты *U* и одним значением компоненты *V*. Каждая из цветоразностных компонент вычисляется как округленное до ближайшего целого среднее арифметическое соответствующих четырех значений рассматриваемого квадрата. Результатом является стандартный формат *YUV* 4 : 1 : 1, который, как правило, является входным для большинства видеокодеров. Нетрудно подсчитать, что на квадрат 2 × 2 будет затрачено уже не 12, а 6 байт. Таким образом, получается сжатие в 2 раза без сколько-нибудь заметного искажения изображения.

Описанное выше представление цветных изображений в виде *RGB* является фактически вариантом более общей конструкции формата *BMP* (от *Bitmap*). В формате *BMP* изображение может храниться как без сжатия, так и со сжатием без потерь с использованием метода кодирования длинами серий. Изображения могут быть монохромными (1 бит/пиксел) или цветными (4, 8, 16, 24 или 32 бита/пиксел).

Файл в формате *BMP* состоит из четырех частей:

BITMAPFILEHEAD
BITMAPINFOHEAD
RGBQUADS
Pixels

В *BITMAPFILEHEADER* и *BITMAPINFOHEADER* содержатся параметры файла и изображения, в *RGBQUADS* записывается цветовая палитра, а затем хранятся собственно пикселы изображения (как индексы палитры или как величины красной, зеленой и голубой составляющей цвета).

В настоящее время в файлах *BMP* изображения обычно хранятся без сжатия в формате либо 8 бит/пиксел (с палитрой), либо 24 бит/пиксел.

4.2. Контрольное задание

1. Создать файл с изображением размером 800 × 600 точек в формате *BMP*. Преобразовать его в форматы *PNG*, *GIF*, *JPEG*, *TIFF* с минимальной компрессией. Сравнить качество изображений. Сделать оценку качества изображений в различных форматах. Оценить

размер полученных изображений. Выбрать оптимальный тип по соотношению *объем файла / качество изображения*.

2. Преобразовать его в форматы *PNG*, *GIF*, *JPEG*, *TIFF* с максимальной компрессией. Сравнить качество изображений. Сделать оценку качества изображений в различных форматах. Оценить размер полученных изображений. Выбрать оптимальный тип по соотношению *объем файла* / *качество изображения*.

3. Создать файл в формате *BMP*, 8 бит/пиксел. Расшифровать полученный файл, вывести размеры, число цветов и другие поля в заголовке изображения, выделить и напечатать массив цветов – палитру.

4. Создать файл в формате *BMP*, 24 бит/пиксел. Прочитать исходное изображение, растянуть его в два раза по высоте и ширине и записать в новый *BMP*-файл. Оценить полученный размер файла и сравнить его с оригиналом по объему и качеству изображения.

5. Создать файл с изображением размером 40 × 30 точек в формате *RGB*. Преобразовать его в формат *YUV*.

6. Выполнить обратное преобразование *YUV* в *RGB*.

4.3. Содержание отчета

В отчете описать исходные данные, ход работы, алгоритмы или листинг программ в соответствии с заданием, результаты выполнения заданий.

4.4. Контрольные вопросы

1. Основные форматы изображений.
2. Характеристики основных форматов изображений.
3. Способы компрессии изображений.
4. Параметры, влияющие на коэффициент компрессии.
5. Параметры файла класса *изображение*.
6. Что такое отношение *сигнал*/*шум* в изображении?
7. Что такое разрешение изображения?
8. Как вычислить объем изображения в битах?

Лабораторная работа № 5

Изучение принципов построения и алгоритмов функционирования терминальных устройств телематических служб. Проектирование сетей абонентского кабеля

Цель работы: изучение методов проектирования сетей абонентского кабеля.

5.1. Арифметический метод

Пусть абонентская матрица имеет размер $n \times m$. Каждая ячейка соответствует некоторой территории, а содержимое – числу источников нагрузки q_{ij} на этой территории.

Место расположения столбца (опорной станции) определяется таким образом, чтобы как слева, так и справа от него было примерно одинаковое количество источников нагрузки (рис. 5.1) [1].

											Qz_i	$\sum_{i=1}^{l} Qz_i$	$\sum_{i=l}^{m} Qz_i$	Dz_l
	16		15	15				9	9	19	83	83	581	498
	10		17	15	12	9	9	12	9		93	176	498	322
	10	15	20	12	15	9	12	9	9		111	287	405	118
		24	15	16	19	9	9		15		107	394	294	100
			13	15	24	10		5			67	461	187	274
			15	15			10		20	21	81	542	120	422
		7	10			11			11		39	581	39	542
Qs_j	36	46	105	88	70	48	40	35	73	40				
$\sum_{j=1}^{k} Qs_j$	36	82	187	275	345	393	433	468	541	581				
$\sum_{j=k}^{n} Qs_j$	581	545	499	394	306	236	188	148	113	40				
Ds_k	545	463	312	119	39	157	245	320	428	541				

Рис. 5.1. Определение места расположения станции арифметическим методом

Если обозначить сумму источников нагрузки в некотором столбце через Qs_j, тогда для каждого столбца s_k можно записать модуль разности сумм источников нагрузки, суммируемых по столбцам, расположенным от него слева и справа, в следующем виде:

$$Ds_k = \left| \sum_{j=1}^{k} Qs_j - \sum_{j=k}^{n} Qs_j \right|. \quad (5.1)$$

Критерий выбора места расположения столбца имеет вид:

$$Ds_k \to \min. \quad (5.2)$$

Для выбора места расположения строки z_l необходимо найти такую строку, для которой сверху и снизу будет примерно одинаковое количество источников нагрузки.

Модуль разности сумм источников нагрузки, суммируемых по строкам, расположенным сверху и снизу от строки z_l:

$$Dz_l = \left| \sum_{i=1}^{l} Qz_i - \sum_{i=l}^{m} Qz_i \right|. \quad (5.3)$$

Критерий выбора места расположения строки имеет вид:

$$Dz_l \to \min. \quad (5.4)$$

Полученные суммы по столбцам и строкам должны быть равны:

$$\sum_{j=1}^{n} Qs_j = \sum_{j=n}^{1} Qs_j = \sum_{i=1}^{m} Qz_i = \sum_{i=m}^{1} Qz_i. \quad (5.5)$$

5.2. Проектирование сетей абонентского кабеля путем минимизации затрат на линии (геометрический метод)

Геометрический метод учитывает не только множество источников и приемников информации, но и расходы на линию [1].

Для всех источников нагрузки, расположенных слева от столбца s_k определяется «горизонтальный» компонент расходов:

$$As_{k\,\text{сл}} = As_{k-1\,\text{сл}} + \sum_{j=1}^{k-1} Qs_j. \quad (5.6)$$

«Горизонтальный» компонент расходов для всех источников нагрузки, расположенных справа от столбца s_k:

$$As_{k\,\text{сп}} = As_{k+1\,\text{сп}} + \sum_{j=k+1}^{n} Qs_j \,. \qquad (5.7)$$

Общие расходы на линию As_k получаются в виде сумм обоих компонентов:

$$As_k = As_{k\,\text{сл}} + As_{k\,\text{сп}} \,. \qquad (5.8)$$

Аналогичные вычисления расходов на линию выполняются для источников нагрузки, расположенных сверху и снизу от строки z_l.

Критерий выбора места расположения столбца и строки тот же, что и в первом случае (рис. 5.2).

											Qz_i	$Az_{k\,св}$	$Az_{k\,сн}$	Az_k
	16		15	15				9	9	19	83	0	1543	1543
	10		17	15	12	9	9	12	9		93	83	1045	1128
	10	15	20	12	15	9	12	9	9		111	259	640	899
		24	15	16	19	9	9		15		107	546	346	892
			13	15	24	10		5			67	940	159	1099
			15	15			10		20	21	81	1401	39	1440
		7	10			11			11		39	1943	0	1943
Qs_j	36	46	105	88	70	48	40	35	73	40				
$As_{k\,сл}$	0	36	118	305	580	925	1318	1751	2219	2760				
$As_{k\,сп}$	2469	1924	1425	1031	725	489	301	153	40	0				
As_k	2469	1960	1543	1336	1305	1414	1619	1904	2259	2760				

Рис. 5.2. Определение места расположения станции геометрическим методом

5.3. Влияние местных условий на оптимальное место расположения станции

Рассмотрим правила вычисления перерасхода средств при обходе препятствия.

Порядок расчета следующий. Сначала определяют места расположения строк и столбцов при отсутствии препятствия. После этого вновь рассчитывают совокупные расходы на линию («горизонтальные» и «вертикальные» компоненты) для первоначальной ячейки

абонентской матрицы и для всех смежных ячеек, учитывающих необходимые направления трасс [1].

Если на абонентской матрице существует участок действия препятствия, то очевидно, что горизонтально расположенные препятствия могут вызывать только «горизонтальный» компонент, а вертикально расположенные – «вертикальный» компонент перерасхода затрачиваемых средств.

Если препятствие занимает нечетное количество столбцов $(1+2p)$ и ожидаемое место расположения опорной станции лежит выше препятствия, то получается перерасход на линию M_j, который может вычисляться по суммам источников нагрузки для столбцов ниже препятствия $Qs_{j\text{н}}$. Если ожидаемое место расположения узла лежит ниже препятствия, то определяющую роль играют суммы источников нагрузки для столбцов выше препятствия $Qs_{j\text{в}}$.

Если препятствие размещается на линии растровой сетки между строками h и $h+1$, то для перерасхода M_j имеем:

$$M_{j\text{в}} = 2\sum_{i=h+1}^{n} q_{ij}; \tag{5.9}$$

$$M_{j\text{н}} = 2\sum_{i=1}^{h} q_{ij}. \tag{5.10}$$

Тогда общий перерасход можно описать следующим выражением:

$$MA_j = \sum_{k=l-p}^{l+p} f_k M_k. \tag{5.11}$$

Таблица 5.1

Значения множителя f_k при условии, что длины препятствий соответствуют (1 + 2 p) длинам стороны квадрата растровой сетки и препятствие не длиннее 5 квадратов

Обозначение столбца	$l-2$	$l-1$	l	$l+1$	$l+2$
$l-2$	$p-1$	$p-1$	$p-1$	$p-1$	$p-1$
$l-1$	$p-1$	p	p	p	$p-1$
l	$p-1$	p	$p+1$	p	$p-1$
$l+1$	$p-1$	p	p	p	$p-1$
$l+2$	$p-1$	$p-1$	$p-1$	$p-1$	$p-1$

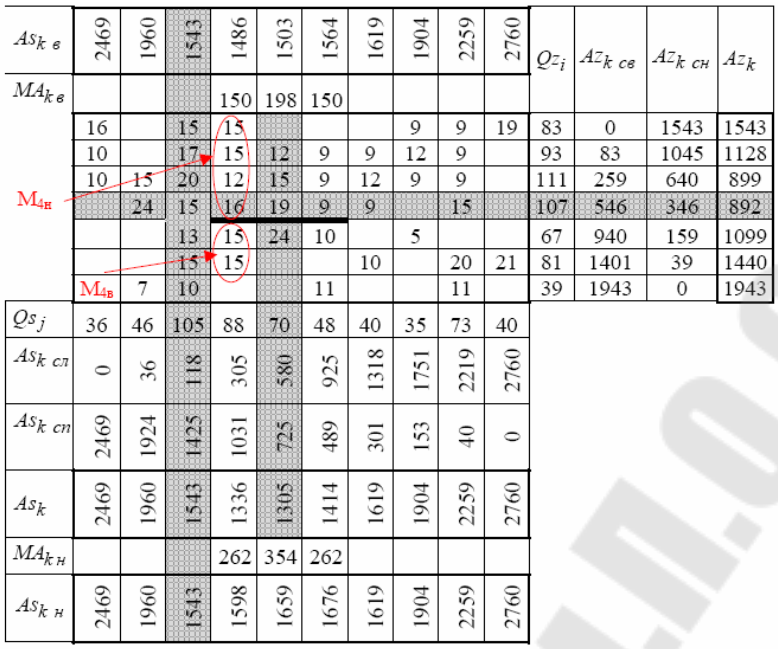


Рис. 5.3. Определение места расположения станции при наличии препятствия

Таблица 5.2

Значения множителя f_k при условии, что длины препятствий соответствуют 2 p длинам стороны квадрата растровой сетки и препятствие не длиннее 5 квадратов

Обозначение столбца	$l-2$	$l-1$	l	$l+1$	$l+2$
$l-2$	$p-2$	$p-2$	$p-2$	$p-2$	$p-2$
$l-1$	$p-2$	$p-1$	$p-1$	$p-1$	$p-1$
l	$p-2$	$p-1$	p	p	$p-1$
$l+1$	$p-2$	$p-1$	p	p	$p-1$
$l+2$	$p-2$	$p-1$	$p-1$	$p-1$	$p-1$

5.4. Пример

Рассмотрим исходную абонентскую матрицу. Отмеченное на рис. 5.3 препятствие содержит горизонтальный компонент перерасхода на линию по столбцам 4–6. Поскольку схема должна ориентироваться на середину препятствия, то $l=5$ (центральный столбец) и $p=1$ $(1+2p=3$, где 3 – длина препятствия). Поскольку препятствие занимает нечетное число ячеек, то используем табл. 5.1 в ином случае используется табл. 5.2. По формулам (5.9) и (5.10) для столбцов 4–6 и для $h=4$ находим перерасход на линию. Результаты расчета занесем в табл. 5.3.

Таблица 5.3

Перерасход на линию

Перерасход	Номер столбца		
	$k = 4$	$k = 5$	$k = 6$
$M_{j\text{В}}$	60	48	42
$M_{j\text{Н}}$	116	92	54

$$MA_{4\text{в}} = pM_{4\text{в}} + pM_{5\text{в}} + pM_{6\text{в}} = 150;$$
$$MA_{5\text{в}} = pM_{4\text{в}} + (p+1)M_{5\text{в}} + pM_{6\text{в}} = 198;$$
$$MA_{6\text{в}} = pM_{4\text{в}} + pM_{5\text{в}} + pM_{6\text{в}} = 150;$$
$$MA_{4\text{н}} = pM_{4\text{н}} + pM_{5\text{н}} + pM_{6\text{н}} = 262;$$
$$MA_{5\text{н}} = pM_{4\text{н}} + (p+1)M_{5\text{н}} + pM_{6\text{н}} = 354;$$
$$MA_{6\text{н}} = pM_{4\text{н}} + pM_{5\text{н}} + pM_{6\text{н}} = 262.$$

Наличие препятствия приводит к тому, что оптимальное место расположения узла смещается в квадрат $A_{4,3}$.

Если препятствие занимает часть вертикальной линии растровой сетки, то аналогичные соображения справедливы для сумм строк, находящихся слева или справа от препятствия.

5.5. Контрольное задание

Самостоятельно сформировать абонентскую матрицу 7×10. Выполнить расчет по пунктам 5.1–5.3. Препятствие расположить по горизонтали, согласовав его с преподавателем. Размер препятствия принять равным пяти ячейкам.

5.6. Содержание отчета

В отчете должны быть отображены следующие пункты:

1. Задание.
2. Теория (при необходимости).
3. Исходные данные.
4. Ход решения поставленной задачи с комментариями.
5. Выводы.

5.7. Контрольные вопросы

1. Как формируется абонентская матрица?
2. В чем заключается задача определения места районной АТС?
3. Перечислите и поясните факторы, влияющие на окончательный выбор места районной АТС.
4. Дайте классификацию сетей связи.
5. Сформулируйте задачи сети доступа и транспортной сети.

Лабораторная работа № 6

Изучение принципов построения и алгоритмов функционирования терминальных устройств телематических служб. Сети Ethernet и Fast Ethernet

Цель работы: проектирование сетей Ethernet и Fast Ethernet.

6.1. Проводные линии связи

Сегодня как для внутренней (кабели зданий), так и для внешней проводки чаще всего применяются три класса проводных линий связи:

– витая пара;

– коаксиальные кабели;

– волоконно-оптические кабели.

Основные особенности конструкции кабелей схематично показаны на рис. 6.1.

Кабели на основе витой пары являются симметричными, т. е. они состоят из двух одинаковых в конструктивном отношении проводников. Симметричный кабель на основе витой пары может быть как экранированным, так и неэкранированным.

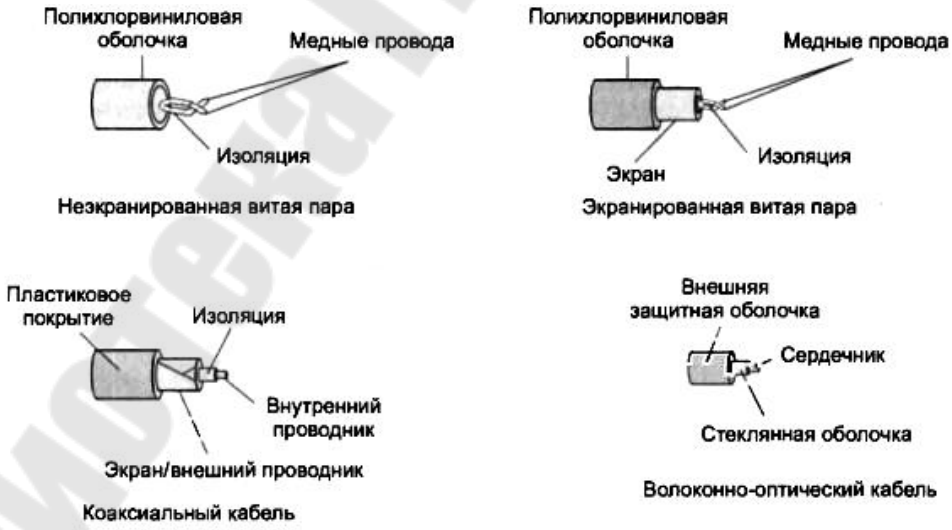


Рис. 6.1. Структура проводных линий связи

В качестве источников света в волоконно-оптических кабелях применяются:

– светодиоды, или светоизлучающие диоды (*Light Emitted Diode*, *LED*);

– полупроводниковые лазеры, или лазерные диоды.

Стоимость волоконно-оптических кабелей ненамного превышает стоимость кабелей на витой паре, но проведение монтажных работ с оптоволокном обходится намного дороже из-за трудоемкости операций и высокой стоимости применяемого монтажного оборудования.

6.2. Расчет сети Ethernet

Задача анализа сети Ethernet возникает при большой протяженности сети (диаметр > 2,5 км) и числе последовательно установленных повторителей больше двух [9].

Рассматривают две модели проектирования: ***Модель 1*** и ***Модель 2***.

Если сеть удовлетворяет ***Модели 1***, то сеть спроектирована верно. Если сеть не удовлетворяет ***Модели 1***, то следует применить ***Модель 2***, и если сеть будет удовлетворять ***Модели 2***, то считают, что сеть спроектирована верно.

В ***Модели 1*** выделяют три условия, которым должна удовлетворять проектируемая сеть. Согласно *первому условию* (рис. 6.2) путь между двумя узлами может содержать:

– до пяти сегментов;

– до четырех повторителей;

– два трансивера;

– два трансиверных кабеля.

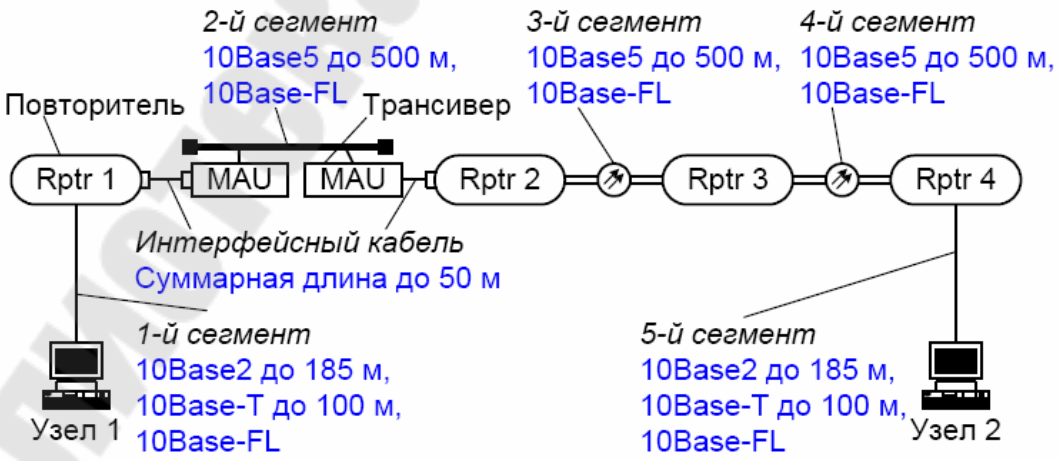


Рис. 6.2. Графическая интерпретация первого условия по ***Модели 1***

Второе условие: если путь содержит пять сегментов и четыре повторителя, то в нем может быть до трех коаксиальных сегментов. Тогда длина волоконно-оптических сегментов может достигать 500 м.

Третье условие: если путь между двумя узлами состоит из трех повторителей и четырех сегментов, то длина волоконно-оптического участка между повторителями не должна превышать 1000 м (рис. 6.3, *а*), а между повторителем и узлом – 400 м (рис. 6.3, *б*). При этом число коаксиальных сегментов может достигать четырех.

а) *б*)

Рис. 6.3. Максимальная длина волоконно-оптического участка по третьему условию

Повторитель (*RPTR*) необходим для объединения сегментов сети, восстановления формы сигналов, временных характеристик и регенерации преамбулы.

Трансивер (*MAU*) служит для подключения толстого коаксиального кабеля или волоконно-оптического кабеля.

При использовании ***Модели 2*** проверяются:

1. Задержка распространения сигнала на двойном пробеге *RTD*, которая не должна превышать максимально допустимой величины $RTD_{\max} = 575$ BT при запасе надежности $SF = 5$ BT. Расчет *RTD* выполняют отдельно от узла *i* к узлу *j* и от узла *j* к узлу *i* по формуле

$$RTD = base + RTDM \cdot L, \qquad (6.1)$$

где *base* – задержка в сетевом элементе (узле, повторителе); *RTDM* – задержка на двойном пробеге в кабельном сегменте в 1 м; *L* – длина сегмента (следует отметить, что длина трансиверного кабеля уменьшается на 2 м от действительного значения).

2. Уменьшение межкадрового интервала *SVV*, которое должно быть не более 49 BT. Уменьшение межкадрового интервала происходит в повторителях в процессе регенерации преамбулы, ретрансляции кадров. Уменьшение задержки учитывается на начальном и среднем сегментах сети. Максимально допустимые задержки на устройствах Ethernet и кабельных сегментах указаны в табл. 6.1.

Таблица 6.1

Максимально допустимые задержки на устройствах Ethernet и кабельных сегментах

Тип сегмента	Максимальная длина, м	*base*, BT			*RTDM*, BT/м
		Начальный сегмент	Средний сегмент	Конечный сегмент	
10*Base*5	500	11,75	46,5	169,5	0,0866
10*Base*2	185	11,75	46,5	169,5	0,1026
10*Base–T*	100	15,25	42	165	0,113
10*Base–FL*	2000	12,25	33,5	156,5	0,1
Трансиверный кабель	48 (+2)	–	–	–	0,102

Методику расчета рассмотрим на примере сети, приведенной на рис. 6.4. Для участка между первым и вторым узлами имеем:

− число повторителей – четыре;

− число сегментов – пять;

− число трансиверов – два;

− длина трансиверного кабеля – 50 м;

− смешанных сегментов – три.

Следовательно, данный участок удовлетворяет *Модели 1*.

Для участка между первым и третьим узлами число трансиверов больше двух, следовательно, этот участок не удовлетворяет *Модели 1*. Поэтому необходимо воспользоваться *Моделью 2* для принятия решения о правильности проектирования сети.

Выполним расчет задержки на двойном пробеге *RTD* между первым и вторым узлами. Результирующая формула будет содержать пять слагаемых (по числу сегментов), вычисляемых по формуле (6.1).

Для первого сегмента $base = 15,25$ BT (поскольку он начальный, а тип сегмента 10*Base–T*), $RTDM = 0,113$ BT/м и $L = 100$.

Используя формулу (6.1), следует выполнить аналогичный расчет для остальных четырех сегментов. Результаты вычислений всех возможных значений *RTD* приведены ниже:

$$RTD(1,2) = (15,25 + 0,113 \cdot 100) + ((50 - 2) \cdot 0,1026 + 33,5 + 0,1 \cdot 500) +$$
$$+ (33,5 + 0,1 \cdot 500) + (33,5 + 0,1 \cdot 500) + (165 + 0,113 \cdot 100) = 458 \text{ BT};$$

$$RTD(2,1) = (15,25 + 0,113 \cdot 100) + (33,5 + 0,1 \cdot 500) + (33,5 + 0,1 \cdot 500) +$$
$$+ ((50 - 2) \cdot 0,1026 + 33,5 + 0,1 \cdot 500) + (165 + 0,113 \cdot 100) = 458 \text{ BT};$$

$$RTD(1,3) = (15,25 + 0,113 \cdot 100) + ((50 - 2) \cdot 0,1026 + 33,5 + 0,1 \cdot 500) +$$
$$+ (33,5 + 0,1 \cdot 500) + ((19 - 2) \cdot 0,1026 + 156,5 + 0,1 \cdot 500) = 407 \text{ BT};$$

$$RTD(3,1) = ((19 - 2) \cdot 0,1026 + 12,25 + 0,1 \cdot 500) + (33,5 + 0,1 \cdot 500) +$$
$$+ ((50 - 2) \cdot 0,1026 + 33,5 + 0,1 \cdot 500) + (165 + 0,113 \cdot 100) = 412 \text{ BT};$$

$$RTD(2,3) = (15,25 + 0,113 \cdot 100) + (33,5 + 0,1 \cdot 500) +$$
$$+ ((19 - 2) \cdot 0,1026 + 156,5 + 0,1 \cdot 500) = 318 \text{ BT};$$

$$RTD(3,2) = ((19 - 2) \cdot 0,1026 + 12,25 + 0,1 \cdot 500) + (33,5 + 0,1 \cdot 500) +$$
$$+ (165 + 0,113 \cdot 100) = 324 \text{ BT}.$$

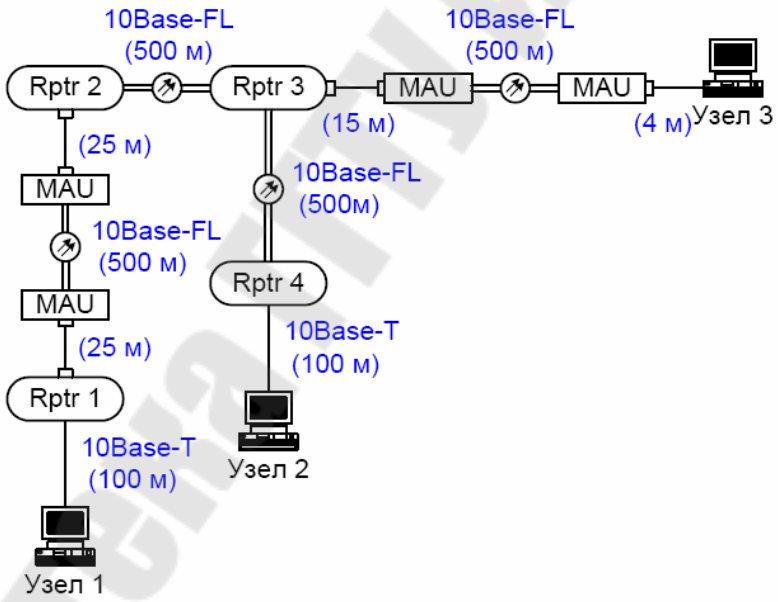


Рис. 6.4. Проектируемая сеть Ethernet

Видно, что *RTD* не превышает 575 BT. Для окончательного принятия решения о правильности проектирования сети рассчитаем величину уменьшения межкадрового интервала *SVV* с учетом табл. 6.2.

Таблица 6.2

Вносимое уменьшение межкадрового интервала

Тип сегмента	Начальный сегмент, BT	Средний сегмент, BT
10*Base*2, 10*Base*5	16	11
10*Base–FL*, 10*Base–T*	10,5	8

Расчет *SVV* выполняется путем суммирования значений вносимого уменьшения межкадрового интервала от начального и средних сегментов. При этом учитываются тип сегмента и его местоположение. Для участка сети между первым и вторым узлами имеется начальный сегмент 10*Base–FL*, который вносит уменьшение межкадрового интервала в 10,5 BT и три сегмента 10*Base–FL*, дающие вклад по 8 BT. Поэтому результирующее значение

$$SVV(1,2) = 10,5 + 8 + 8 + 8 = 34,5 \text{ BT}.$$

Аналогично выполним расчет для остальных сегментов.

$$SVV(2,1) = 10,5 + 8 + 8 + 8 = 34,5 \text{ BT};$$

$$SVV(1,3) = 10,5 + 8 + 8 = 26,5 \text{ BT}; \quad SVV(3,1) = 10,5 + 8 + 8 = 26,5 \text{ BT};$$

$$SVV(2,3) = 10,5 + 8 = 18,5 \text{ BT}; \quad SVV(3,2) = 10,5 + 8 = 18,5 \text{ BT}.$$

Анализ показывает, что *SVV* не превышает 49 BT. Следовательно, рассматриваемая сеть спроектирована верно.

6.3. Расчет сети Fast Ethernet

Для сети Fast Ethernet также приняты две модели, которым должна соответствовать проектируемая сеть [9], [10].

Модель 1 определяет четыре типовых схемы (рис. 6.5), на которых указываются типы сегментов и их предельные длины. Следует отметить, что все приведенные длины сегментов предельны. При установке на удаленном конце коммутатора с подключенными к нему рабочими станциями суммарная длина кабелей должна соответствовать приведенной в *Модели 1*. Использование Модели 2 заключается в вычислении задержки распространения сигнала на двойном пробеге *RTD*, которая не должна превышать 512 BT. Уменьшение межкадрового интервала *SSV* не рассчитывается, поскольку в сети имеется небольшое число повторителей. Максимально допустимые задержки на двойном пробеге $RTD_{\max}$ приведены в табл. 6.3.

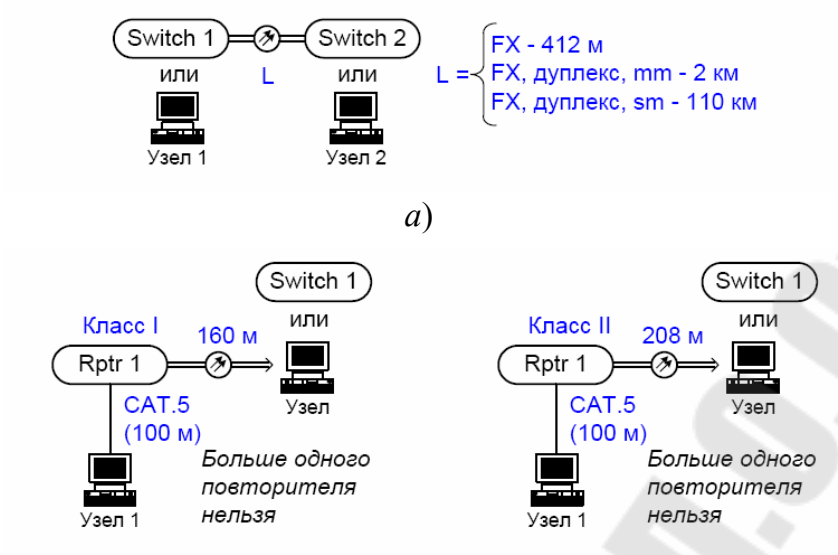


Рис. 6.5. Графическая интерпретация первого условия по ***Модели 1*** для сети Fast Ethernet:
а – Модель 1A; *б – Модель 1B*; *в – Модель 1C*

Таблица 6.3

Максимально допустимые задержки на устройствах Fast Ethernet и кабельных сегментах

Устройство / кабельный сегмент	**Максимальная задержка на двойном пробеге RTD_{max}, BT**
Узел / коммутатор	100 (суммарно для двух оконечных устройств)
Повторитель класса I	140
Повторитель класса II	92
Витая пара *CAT*.5 для 1 м	1,112
Оптоволокно для 1 м	1,0

Проверим с помощью *Модели 2* справедливость *Модели 1*.

Так в *Модели 1A* (рис. 6.5, *а*) имеется два узла с суммарной задержкой в 100 BT и оптоволоконный сегмент длиной 412 м с удельной задержкой в 1 BT. Кроме того, $RTD(1,2) = RTD(2,1) = 100 + 412 \cdot 1 = 512$ BT. Аналогично для остальных моделей:

Модель 1B: $RTD = 100 + 100 \cdot 1{,}112 + 140 + 160 \cdot 1 = 511{,}2$ BT.

Модель 1C: $RTD = 100 + 100 \cdot 1{,}112 + 92 + 208 \cdot 1 = 511{,}2$ BT.

Модель 1D:RTD(1, 2) = 100 + (100 + 100 + 5) · 1,112 + 2 · 92 = = 511,96 BT,

$$RTD(1,3) = 100 + 100 \cdot 1{,}112 + 92 + 111 \cdot 1 = 414{,}2 \text{ BT},$$

$$RTD(2,3) = 100 + (100 + 5) \cdot 1{,}112 + 2 \cdot 92 + 111 \cdot 1 = 511{,}76 \text{ BT}.$$

Из приведенных расчетов видно, что значение *RTD* не превышает 512 BT.

Рассмотрим пример вычисления максимальной длины волоконно-оптического кабеля для сети по рис. 6.6.

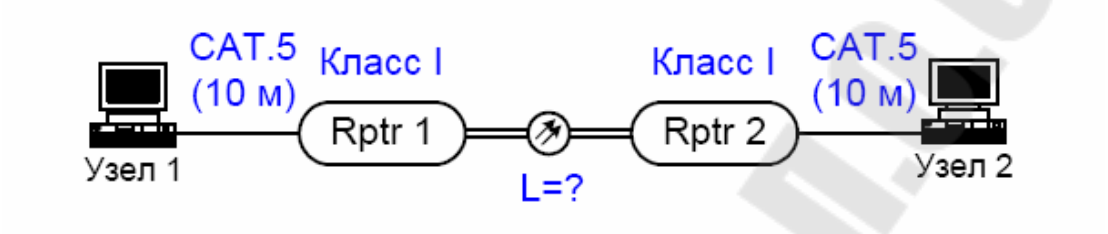


Рис. 6.6. Сеть Fast Ethernet

Составим по формуле (6.1) выражение для вычисления максимальной задержки распространения сигнала на двойном пробеге с учетом всех элементов сети, приведенной на рис. 6.6. Имеем:

$$RTD_{\max} = 100 + (10 + 10) \cdot 1{,}112 + 2 \cdot 140 + L \cdot 1 = 512 \text{ BT}.$$

Из данного уравнения находим максимальную длину волоконно-оптического участка L , которая будет равна 109,8 м.

6.4. Контрольное задание

Самостоятельно составить сеть Ethernet по варианту, предложенному преподавателем, в которой должно быть не меньше пяти сегментов. Выполнить анализ сети по *Модели 1* и *Модели 2*.

6.5. Требование к отчету

В отчете должны быть отображены следующие пункты:

1. Задание.
2. Теория (при необходимости).
3. Исходные данные.
4. Ход решения поставленной задачи с комментариями.
5. Выводы.

6.6. Контрольные вопросы

1. Дайте понятие коллизионного домена.
2. Объясните принцип протокола *CSMA*/*CD*.
3. Объясните назначение концентратора.
4. Объясните назначение коммутатора.
5. Приведите и поясните формат кадра Ethernet.
6. Особенности Модели 1.
7. Особенности Модели 2.
8. Классификация и особенности применения витой пары.
9. Классификация и особенности применения коаксиальной линии связи.
10. Разновидности волоконно-оптических линий связи.
11. Объясните, для чего при проектировании сети следует учитывать задержки.

Литература

1. Бесслер, Р. Проектирование сетей связи : справочник : пер. с нем. / Р. Бесслер, А. Дойч. – М. : Радио и связь, 1988. – 267 с.

2. Убайдуллаев, Р. Р. Волоконно-оптические сети / Р. Р. Убайдуллаев. – М. : Эко-трендз, 1998. – 336 с.

3. Гольдштейн, Б. С. Протоколы сети доступа связи. – в 2 т. / Б. С. Гольдштейн. – М. : Радио и связь, 2001. – Т. 2. – 290 с.

4. Семенов, Ю. В. Проектирование сетей следующего поколения / Ю. В. Семенов. – СПб. : Наука и Техника, 2005. – 240 с.

5. Сэломон. Сжатие данных, изображений и звука / Сэломон. – М. : Техносфера, 2004. – 368 с.

6. Чемпен, Н. Цифровые технологии мультимедиа / Н. Чемпен, Д. Чемпен. – М. : Вильямс, 2006. – 624 с.

7. Чемпен, Н. Цифровые графические инструменты / Н. Чемпен, Д. Чемпен. – М. : Вильямс, 2006. – 656 с.

8. Прэтт, У. Цифровая обработка изображений / У. Прэтт. – М. : Мир, 1982. – 790 с.

9. Вудс, Р. Цифровая обработка изображений / Р. Вудс, Р. Гонсалес. – М. : Техносфера, 2005. – 1072 с.

10. Цифровая обработка изображений в информационных системах : учеб. пособие. / Н. С. Грузман [и др.]. – Новосибирск : НГТУ, 2000. – 440 с.

11. Уэлстид, С. Фракталы и вейвлеты для сжатия изображений в действии / С. Уэлстид. – М. : Триумф, 2003. – 320 с.

12. Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео / Д. Ватолин [и др.]. – М. : Диалог-МИФИ, 2003. – 384 с.

Содержание

Учебное электронное издание комбинированного распространения

Учебное издание

Ковалев Алексей Викторович
Литвинов Дмитрий Александрович
Ростокина Ольга Михайловна

ИНТЕРФЕЙСЫ И УСТРОЙСТВА ТЕЛЕКОММУНИКАЦИЙ

Практикум по выполнению лабораторных работ по одноименной дисциплине для студентов специальности 1-53 01 07 «Информационные технологии и управление в технических системах» дневной формы обучения

Электронный аналог печатного издания

Редактор Н. В. Гладкова
Компьютерная верстка И. П. Минина

Подписано в печать 27.05.21.
Формат 60x84/16. Бумага офсетная. Гарнитура «Таймс».
Ризография. Усл. печ. л. 2,79. Уч.-изд. л. 2,71.
Изд. № 2.
http://www.gstu.by

Издатель и полиграфическое исполнение
Гомельский государственный
технический университет имени П. О. Сухого.
Свидетельство о гос. регистрации в качестве издателя
печатных изданий за № 1/273 от 04.04.2014 г.
пр. Октября, 48, 246746, г. Гомель