10. Кодирование и контроль правильности при обмене данными
по последовательному цифровому каналу

Основные режимы передачи данных: *симплексный*, *полудуплексный*, *дуплексный*.
Типы синхронизации: а) *битовая*/*тактовая*; б) *символьная*/*байтная*; в) *блочная* (блок передается как цепочка битов). На ***рис.* 10.**1,*а* приведен формат передаваемого символа в *асинхронном* режиме.

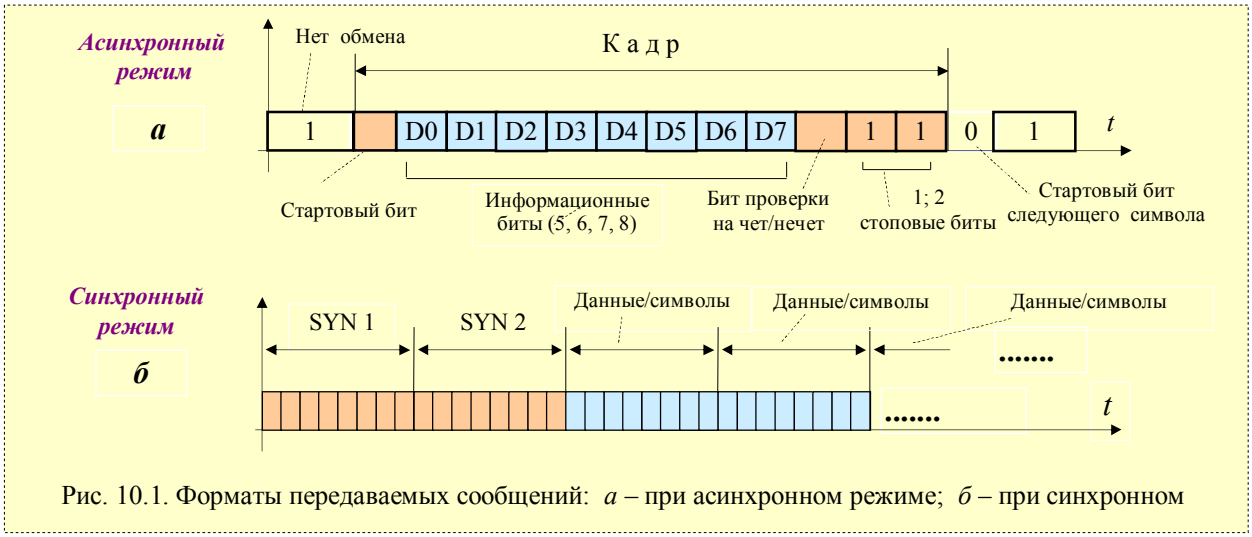


Рис. 10.1. Форматы передаваемых сообщений: *а* – при асинхронном режиме; *б* – при синхронном

При передаче *блоков данных* используется *синхронный режим* (***рис.*** 10.1,*б*). ПД и ПР должны быть синхронизированы в течение длительного промежутка времени, поэтому *не используется* обрамление каждого байта, а границы блока фиксируются с помощью специальных начальной и конечной комбинации байтов SYN 1, SYN 2. Синхронизация может быть достигнута при наличии дополнительной линии, соединяющей 2 узла, однако это дорого. Поэтому чаще используется одна линия данных, а тактовая информация включается в форму передаваемого сигнала. Для этого используют специальные *самосинхронизирующиеся* коды, например, Манчестерский.

Способы контроля правильности передачи данных (*управление ошибками*)

Под *управлением ошибками* понимают полный цикл обнаружения/исправления ошибки. Используются правила, образующие *протокол* связи. При этом возможны *два подхода*: 1. Обнаружение ошибки и **а**втоматическое формирование **з**апроса на **п**овтор сообщения (АЗП); 2. **О**бнаружение и **к**оррекция/исправление **о**шибки (ОКО). При **АЗП** - к каждому передаваемому сообщению/кадру добавляются дополнительные контрольные биты, которые позволяют приемнику обнаружить некоторые конкретные типы ошибок. Если ошибка обнаружена, то используются дополнительные процедуры, *запрашивающие присылку новой копии сообщений* (запрос на повтор сообщения). При **ОКО** - к сообщению добавляется достаточное число битов, позволяющих не только обнаружить наличие в поступившем сообщении одной или нескольких ошибок, но и локализовать место ошибок. Более того, т.к. сообщение поступает в двоичном коде, то *исправление* достигается простым *инвертированием ошибочных битов*.

Если имеется обратный тракт (*дуплекс*), то более эффективен метод на основе АЗП. Однако, например, при получении информации от космических зондов используется только односторонний канал (*симплекс*), при этом задержки передачи от спутников столь велики, что передающая станция успевает передать несколько сотен других сообщений, прежде чем поступит извещение в обратном направлении. Поэтому зачастую совместно с методами АЗП используются метод ОКО, что позволяет сократить число повторных передач.

Методы обнаружения ошибок при ***асинхронной*** *передаче*

Иногда достаточно использования *одного дополнительного* бита «*чет*/*нечет*» (он вычисляется до передачи данных) *на каждый* передаваемый *символ*. Приемник сам определяет значение бита «*чет*/*нечет*», сравнивает его с полученным, и таким образом выявляет наличие ошибок передачи. Метод позволяет обнаружить ошибку в одном бите. Поэтому, если возможны ошибки в большем числе (в четном числе) бит, то нужно использовать другие методы (с большей избыточностью). Такой метод может быть расширен (***рис.*** 10.2) дополнительным набором битов «*чет*/*нечет*», вычисленных исходя из последовательности *символов*. При этом каждому символу (строке), *синхронный* ПД присоединяет бит «*чет*/*нечет*». Но, кроме того, добавляется *еще по одному биту* «*чет*/*нечет*» на каждый столбец (группы символов).

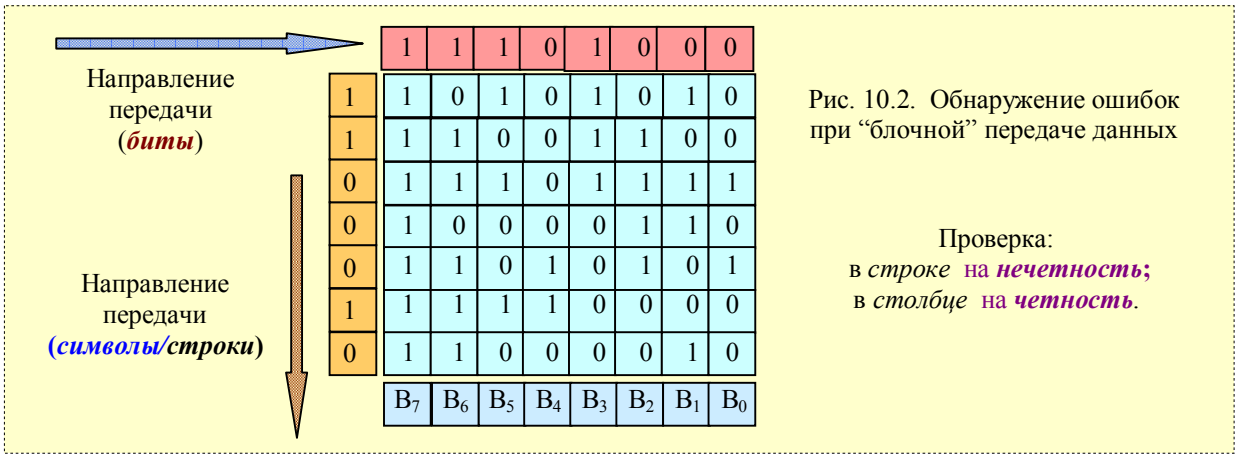


Рис. 10.2. Обнаружение ошибок при “блочной” передаче данных

Проверка:
в *строке* на ***нечетность;***
в *столбце* на ***четность***.

Из рассмотрения этого подхода видно, что 2 ошибки в одном символе, не обнаруженные поперечным битом «*чет*/*нечет*» (в строке), будут обнаружены продольными битами «*чет*/*нечет*» (в столбце). Конечно, это справедливо только при условии, что в одном и том же столбце не возникнут 2 ошибки одновременно. Однако нужно подчеркнуть, что независимо от принятой схемы обнаружения (и исправления) ошибок не существует средств обнаружения всех возможных комбинаций ошибок с полной гарантией. Поэтому на практике *целью различных методов обнаружения и исправления* ошибок является обеспечение *приемлемо низкого уровня вероятности необнаружения* ошибок.

Отметим следующее: а) *помехозащищенность* достигается с помощью введения избыточности; б) *в дуплексных* каналах достаточно применения кодов, *обнаруживающих ошибки*, т.к. сигнализация об ошибке вызывает повторную передачу от источника; в) устранение ошибок с помощью *корректирующих* кодов реализуют *в симплексных* каналах связи; г) проверка на четность или нечетность недостаточно надежна.

Рассмотрим наиболее широко применяемые коды – коды *Хемминга* и *циклические* коды.

Управление ошибками на основе кода Хемминга

Введено понятие кодового расстояния D (или d_{ij} для отдельной пары двоичных кодовых комбинаций). *Расстояние по Хеммингу* между двумя кодовыми комбинациями равно *числу несовпадающих разрядов*. Если, например, кодовая комбинация 011 0111 принята как 010 0110, то число несовпадающих двоичных разрядов равно 2, и значит расстояние по Хеммингу между ними равно 2, т.е. $d_{ij}=2$. Такое расстояние можно определить как число единиц в сумме этих комбинаций по модулю 2 ($\oplus$):

$$\begin{array}{l} \oplus\ 0\,1\,1\ \ 0\,1\,1\,1 \\ \underline{0\,1\,0\ \ 0\,1\,1\,0} \\ 0\,0\,1\ \ 0\,0\,0\,1 \end{array} \rightarrow (2)_{10},\ \text{т.е. } d_{ij}=2$$

Если расстояние равно 2, то имеет место двукратная ошибка (r=2). В общем случае, когда искажается r разрядов кодовой комбинации, имеет место r-кратная ошибка. Для кодов в целом рассматривают матрицу расстояний D. Например, для кода x1=00, x2=01, x3=10, x4=11 матрица D имеет вид:

	00	01	10	11	
	x1	x2	x3	x4	
$D=$	0	1	1	2	x1 00
	1	0	2	1	x2 01
	1	2	0	1	x3 10
	2	1	1	0	x4 11

По горизонтали (i) и по вертикали (j) отмечены все двухразрядные сравниваемые кодовые комбинации (00, 01, 10, 11), а в соответствующих клетках матрицы (ij) – кодовое расстояние по Хеммингу между ними. В этом 2^x-разрядном коде исчерпываются все кодовые комбинации (*нет избыточности*), поэтому любая ошибка приведет к тому, что получающаяся неверная кодовая комбинация будет воспринята как разрешенная. Как видно из полученной матрицы D, для данного кода имеет место соотношение $d_{ij,min}$=1.

Рассмотрим 2-разрядный код с проверкой на *четность*, в нем к каждой кодовой комбинации добавлен 1 разряд, в котором ставится 0, если число единиц четное, и 1 - если нечетное, т.е. такой код имеет *один избыточный разряд*: x1=00 0, x2=01 1, x3=10 1, x4=11 0.

Матрица D для него примет вид:

	000	011	101	110	
	x1	x2	x3	x4	
D =	0	2	2	2	x1 000
	2	0	2	2	x2 011
	2	2	0	2	x3 101
	2	2	2	0	x4 110

Для этого кода $d_{ij,min}$=2. В методах на основе расстояний по Хеммингу возможности *обнаружения* и *исправления (коррекции)* ошибок кратности r связаны с минимальным кодовым расстоянием $d_{ij,min}$:

- *обнаруживаются* ошибки кратности r, при условии $r = d_{ij,min} - 1$;

- *исправляются* ошибки кратности r, если $r = ent\left(\frac{d_{ij\,min}-1}{2}\right)$,

где *ent* (..) - "целая часть от (..)".

Т.к. при контроле на четность $d_{ij,min}$=2, то *обнаруживаются* только одиночные ошибки, но корректироваться (исправляться) они не могут.

Рассмотрим *код Хемминга* с $d_{ii,min}$=3, в котором *дополнительно* вводится $k = \log_2 n$ разрядов - *дополнительная часть кода* (ДЧК), где n - число информационных разрядов, k – число избыточных разрядов, округляется до ближайшего *большего целого* значения.

Поскольку $d_{ij,min}-1=3-1=2$, а $ent\left(\frac{d_{ij\,min}-1}{2}\right)=\frac{3-1}{2}=1$, то данный код позволяет *обнаруживать* двукратные ошибки, а *исправлять* – *одиночные* ошибки.

Алгоритм *на передающей стороне.* В ПД применение корректирующего кода, в основном, сводится к вычислению ДЧК посредством выполнения следующих *поразрядных* операций:

1) *сложение по модулю 2* *двоичных номеров* тех информационных *разрядов* кодовой комбинации, значения которых равны 1; *нумерация* осуществляется *с единицы*, а не с нуля !!!

2) *инвертирование* полученной двоичной последовательности (инвертирование результата по п.1).

Пример: Передана кодовая комбинация 100**1**10 (n =6), а принята 100**0**10. *Следовательно*, $k = \log_2 6 = 3$ (3 – результат округления).

Определим ДЧК:

$$010 \oplus 011 \oplus 110 = 111 \quad \text{ДЧК} \rightarrow \mathbf{000}$$
$$(2)_{10} \quad (3)_{10} \quad (6)_{10}$$

где $\oplus$ - символ операции *"Сложение по модулю 2"*; после инвертирования, окончательно имеем ДЧК, равную 000. Теперь вместе с основной кодовой комбинацией будет передаваться и ДЧК (000), *т.е.* 100110 000.

Алгоритм** на приемной стороне.* В ПР аналогично (см. выше) вновь рассчитывается ДЧК, которая после инвертирования сравнивается с переданной. Код (результат) сравнения определяется *путем выполнения над ними операции сложения по модулю 2*, и если он отличен от 0, то *его значение* (без инверсии !***) есть *номер ошибочно принятого разряда*.

Так, если *принята кодовая комбинация* 100010, то рассчитанная в приемнике ДЧК равна инверсии от результата выполнения операции:

010 ⊕ 110=100 ДЧК → **011**
$(2)_{10}$ $(6)_{10}$

После его инверсии получаем: 011.

Определяем код сравнения, посредством выполнения операции сложение по модулю 2 (*без дальнейшей его инверсии* !): **000 ⊕ 011=011** → $(\mathbf{3})_{10}$, что означает *ошибку* в 3-м разряде.

Исправляется (корректируется) ошибка посредством инвертирования 3-го разряда в принятой кодовой комбинации.

В общем случае для того, чтобы код позволял *обнаруживать* все ошибки кратности $r \le r_O$ и *корректировать* (*исправлять*) все ошибки кратности $r \le r_к$, его кодовое расстояние должно удовлетворять неравенству $d \ge r_O + r_к + 1$ $(r_O \ge r_к)$

*Управление ошибками на основе **циклических** (полиномиальных) кодов*

К числу эффективных кодов, *обнаруживающих одиночные*, *кратные* ошибки и *пачки* ошибок, относятся *циклические коды* "CRC-коды" (Cyclic Redundance Code). Информация длиной ***n*** бит представляется в виде *полинома* степени, не более *n*-1. Например, данные вида $A=(1011001)_2=(89)_{10}$ представляются в виде полинома $A=1\cdot 2^6+0\cdot 2^5+1\cdot 2^4+1\cdot 2^3+0\cdot 2^2+0\cdot 2^1+1\cdot x^0=2^6+2^4+2^3+1=89$.

Рассмотрим один из алгоритмов формирования циклического кода

Введем обозначения: A - информационный полином степени n, соответствующий передаваемым данным; g – порождающий полином степени k, определяющий число контрольных/избыточных разрядов (корректирующую способность кода); C – полином степени $n+k$, соответствующий передаваемому циклическому коду.

Кодирование информации на передающем узле происходит следующим образом. Информационный полином A умножается на 2^k, т.е. сдвигается на k разрядов влево. Полином $2^k \cdot A$ делится на полином g и определяется остаток B, т.е. полином степени, меньшей чем k, который записывается в k младших разрядах кодового полинома. Далее формируется кодовый полином C, старшие n разрядов которого представляют информационный полином A, т.е. передаваемые данные, а k младших разрядов – остаток B, т.е. контрольные биты. Естественно, что сформированный полином C делится на порождающий полином g без остатка.

Обнаружение ошибки основано на том, что *разрешенными* являются информационные слова, полином которых делится без остатка на порождающий полином (т.е. принятые данные *не содержат* ошибок), а *запрещенными* – имеющие *ненулевой* остаток (*содержатся* ошибки).

Известен алгоритм, в котором образующий двоичный полином $g(2)$ является представлением одного из простых множителей, на которые раскладывается число 2^n-1, где 2^n обозначает единицу в n-м разряде, n равно числу разрядов кодовой комбинации.

Кодирование заключается в *умножении* каждой кодовой комбинации на *образующий полином* $g(2)$, а *декодирование* - *делением* принятой кодовой комбинации на $g(2)$. Обозначим исходные числа/коды как A_i, а закодированные - как $C_i = A_i \cdot g(2)$. При обнаружении ошибки, сигнал об ошибке поступает в ПД, что вызывает повторную передачу.

Пусть, *например*, передаются данные в виде 10-разрядных кодовых комбинаций, т.е. n=10. Тогда 2^{10}-1=1023=11·93. Выберем в качестве образующего полинома $g(2)=(11)_{10}=(1011)_2$.

Один из широко применяемых *алгоритмов кодирования на основе циклических кодов* отличается тем, что операция деления на образующий полином $g(2)$ заменяется операцией ***сложение по модулю 2*** (⊕) с этим полиномом, при этом выполняются следующие операции:

1) ПД формирует новую кодовую комбинацию вида $A\cdot(2^k)$; т.е. к исходному кодируемому числу A справа приписывается k нулей (число битов в образующем полиноме, уменьшенное на 1);

2) над *полученным* числом $A\cdot(2^k)$ на каждом шаге осуществляется поразрядная операция ⊕ с образующим полиномом $g(2)$. Формально это выглядит аналогично делению "*столбиком*", но здесь отсутствует операция вычитания на каждом шаге. При этом на каждом шаге получается остаток, который дополняется до размерности образующего полинома, а затем вновь выполняется поразрядная операция ⊕, полученный остаток дополняется и т.д.

Окончательный остаток B (тоже полином) и представляет собой *дополнительную часть кода* (ДЧК), т.е. избыточные k-разрядов; ими заменяют в кодовой комбинации A приписанные справа k нулей, т.е. формируется передаваемая кодовая комбинация C, $C= A\cdot(2^k)+B$.

На приемном узле выполняются *аналогичные* операции, но ***над кодовой комбинацией C*** ($C=A\cdot 2^4+B$) с *образующим полиномом* $g(2)$.

Если окончательный остаток *не* равен 0, то при передаче данных произошла ошибка и нужна повторная передача кода A, в противном случае ошибок нет.

Пример (***рис.*** 10.3): Пусть передается кодовая комбинация A=1001 1101. Число разрядов (n) в данной кодовой комбинации n=8, 2^8-1=255, представим это число в виде произведения простых сомножителей 255=3·17·5. Выберем для формирования образующего полинома число 17: $g(2)=(17)_{10}=(10001)_2$, число избыточных разрядов k, т.е. дополнительная часть кода должна содержать число битов в образующем полиноме, уменьшенное на 1; k=5-1=4.

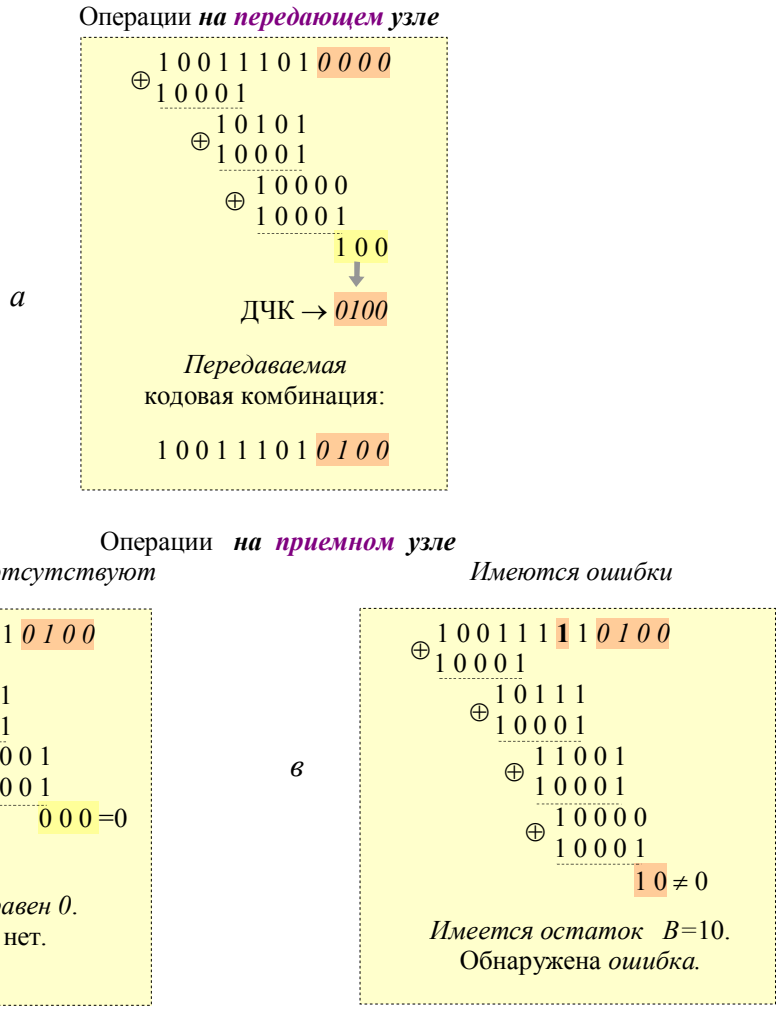


Рис. 10.3. Иллюстрация кодирования/декодирования на основе циклических кодов: *а* – на передающем узле; *б* – в принятых данных не обнаружено ошибок; *в* – в принятых данных обнаружена ошибка.

Выполняемые операции на приемном узле (***рис.*** 10.3,*б*,*в*). Это *аналогичные* операции (см. выше), но уже ***над кодовой комбинацией С*** ($C=A\cdot 2^4+B$=1001 1101 *0100*) с *образующим полиномом* $g(2)$=10001. Если окончательный остаток равен 0 (***рис.*** 10.3,*б*), то при передаче данных ошибок нет, в противном случае (***рис.*** 10.3,*в*), т.е. остаток не равен 0, произошла ошибка и нужна повторная передача данных. *Положительными свойствами* циклических кодов является малая вероятность необнаружения ошибки и сравнительно небольшое число избыточных разрядов.

Практическое применение циклического избыточного контроля

Например, кадр стандарта *Ethernet*, состоящий из 1024 байт, будет рассматриваться как одно число, состоящее из 8192 битов. В качестве контрольной информации рассматривается остаток от деления этого числа на известный делитель R – тоже представленный в виде полинома.

Обычно в качестве делителя выбирается 17- или 33-разрядное число (двоичный полином), чтобы остаток от деления имел длину 16 разрядов (2 байта) или 32 разряда (4 байта).

При получении кадра данных снова вычисляется остаток от деления на тот же делитель, но при этом к данным кадра добавляется и содержащаяся в нем контрольная сумма. Если остаток от деления на R равен 0, то делается вывод об отсутствии ошибок в полученном кадре, в противном случае кадр считается искаженным.

В *протоколе* V.42 для кодирования кодовых групп в 240 разрядов с 2 избыточными байтами (называемыми также контрольной суммой КС=2 байтам) используется полином:

$g(X)=2^{16}+2^{12}+2^5+1$, что эквивалентно коду 1 0001 0000 0010 0001.

Возможен также образующий полином для КС=4 байта:

$g(X)=2^{32}+2^{26}+2^{23}+2^{22}+2^{16}+2^{12}+2^{11}+2^{10}+2^8+2^7+2^5+2^4+2^2+2^1+2^0$, который используется в локальной сети *Ethernet* для передачи сообщений объемом 1024 байт=8192 бита.

Контроль с помощью сторожевых таймеров. Контроль с помощью сторожевых таймеров (*watchdog times*) является простым и широко употребляемым средством контроля, дополняющим средства на основе кода Хемминга и циклических кодов. Сторожевой таймер обнаруживает ошибки процессора путем слежения за его активностью. Процессор периодически запускает внешний по отношению к нему таймер (счетчик обратного счета), который начинает отсчитывать время *таймаута* (например, 500 мс). Если процессор исправен, он подает строб на таймер до истечения этого времени. Если за время таймаута процессор не запустит таймер, фиксируется ошибка и выдается прерывание.

Сжатие (компрессия) данных

На современном этапе развития информационных технологий требования к пропускной способности и объему памяти сетевых средств довольно жесткие. Например, для показа 5-минут-ного фрагмента видеофильма в окне 352x288 пикселов при частоте смены кадров 25 Гц и при представлении пиксела 24-разрядным кодом требуется скорость передачи данных 7,6 Мбайт/с и память 2,3 Гбайт. Поэтому необходимо сжатие информации.

Т.о., при стремлении *сократить время* передачи данных и *сэкономить объем внутренней памяти* устройств возникает необходимость ставить вопрос о сжатии (компрессии) данных, т.е. о передаче того же количества информации с помощью последовательностей символов *меньшей* длины. Это особенно актуально, если вспомнить, что определение и коррекция ошибок при передаче данных требуют *избыточности* кодов.

Для целей сжатия данных используются специальные алгоритмы, *уменьшающие избыточность*. Эффект сжатия оценивают *коэффициентом сжатия* $K=n/q$, где n - число минимально необходимых символов для передачи сообщения (число символов на выходе эталонного алгоритма сжатия); q - число символов в сообщении, сжатом данным алгоритмом.

Часто степень сжатия оценивают отношением длин кодов на входе и выходе алгоритма сжатия. Так, при двоичном кодировании n равно энтропии источника информации.

Компрессия/сжатие и декомпрессия данных осуществляется либо на прикладном *программном* уровне, либо с помощью *аппаратных* средств. Устройства или программы, применяемые для компрессии и декомпрессии, называют *кодеками* ("***ко****дирование*" и "***дек****одирование*").

Т.к. на предварительное сжатие данных ПД тратит дополнительное время, к которому нужно добавить аналогичные затраты времени на разворачивание этих данных в ПР, то выгоды от сокращения времени на передачу сжатых данных обычно бывают заметны только для *низкоскоростных* каналов – около 64 Кбит/с.

По способу сжатия относительно текущего времени существующие методы сжатия можно разделить на 2 группы:

Статические, обеспечивающие предварительное сжатие данных (например, с помощью архиваторов типа ARJ, RAR, WinZip), после чего они отсылаются в сеть. Речь идет о сжатии данных с помощью программ сжатия.

Динамические, обеспечивающие компрессию совместно с передачей данных.

По способу сжатия относительно сохранности исходной информации существующие методы сжатия можно разделить тоже на 2 группы:

- алгоритмы, *не уменьшающие количество информации в сообщении*, т.е. *сжатие без потерь* (алгоритмы Лемпеля-Зива, RLE, кодирование Хаффмена и др.);
- алгоритмы *сжатия с потерями* (JPEG, M-JPEG, MPEG).

Алгоритмы сжатия без потерь

Алгоритм Лемпеля-Зива – лежит в основе некоторых архиваторов (pkzip, arj, lha) и программ динамического сжатия. *Основная идея* – второе и последующие *вхождения* некоторой строки символов в сообщении заменяются ссылкой на ее первое появление в сообщении. Используется для сжатия *текстов* и *графики*.

Алгоритм символьного подавления - наиболее эффективен, когда передаваемые данные содержат большое количество *повторяющихся* байтов. *Например*, при передаче черно-белого изображения черные поверхности будут порождать большое количество *нулевых* значений, а максимально освещенные участки – большое количество *байтов, состоящих из единиц*.

Передатчик сканирует последовательность передаваемых байтов и, если обнаруживает последовательность из *3 или более* одинаковых байт, то заменяет ее специальной 3-*байтовой* последовательностью, в которой указывается *значение* байта, *число его повторений*, а также отмечает *начало* этой последовательности специальным управляющим символом.

К этой же группе примыкают *алгоритмы RLE* (Run Length Encoding). В них вместо передачи *непрерывной* цепочки из одинаковых символов передаются *символ* и *значение длины цепочки* – *два байта* (в первом – *символ*, во втором – *счетчик*, т.е. число, которое показывает, сколько таких символов идет подряд). Метод эффективен при передаче *растровых изображений* (сжатие *графики*: файлы формата PCX и *видео*), но малополезен при передаче *текста*.

Алгоритм Хаффмена реализует замену информационных символов кодовыми последовательностями различной длины (*коды переменной длины*).

Идея метода - часто повторяющиеся символы нужно кодировать более короткими цепочками битов, чем цепочки редких символов. Строится двоичное дерево, листья соответствуют кодируемым символам, код символа представляется последовательностью значений ребер (эти значения в двоичном дереве суть 1 и 0), ведущих от корня к листу. Листья символов с высокой вероятностью появления находятся ближе к корню, чем листья маловероятных символов.

Распознавание кода, сжатого по методу Хаффмена, выполняется по алгоритму, аналогичному алгоритмам восходящего грамматического разбора.

Такое кодирование называют также *статистическим* кодированием.

Алгоритм позволяет строить *неравномерные* коды *автоматически*, на основании известных частот появления символов.

Например, пусть набор из 8 символов (A,B,C,D,E,F,G,H) имеет следующие правила кодирования: A ⇒ 10; E ⇒ 0001; B ⇒ 01; F ⇒ 0000; C ⇒ 111; G ⇒ 0011; D ⇒ 110; H ⇒ 0010.

Тогда при распознавании входного потока 1 0 1 1 0 0 0 0 0 1 1 0 в стек распознавателя заносится 1, но 1 не совпадает с правой частью ни одного из правил. Поэтому в стек добавляется следующий символ 0. Полученная комбинация 10 распознается и заменяется на A (есть правило A ⇒ 10). В стек поступает следующий символ 1, затем 1, затем 0. Сочетание 110 совпадает с правой частью правила для D (D ⇒ 110). Теперь в стеке AD, заносятся следующие символы 0000 и т.д. В итоге получаем: 1 0 1 1 0 0 0 0 0 1 1 0 ⇒ A D F D.

Недостаток метода заключается в необходимости знать вероятности символов. Если заранее они неизвестны, то *требуется 2 прохода*: на одном в передатчике подсчитываются вероятности, на другом эти вероятности и сжатый поток символов передаются в приемник. Однако двухпроходность не всегда возможна. Этот недостаток устраняется в однопроходных алгоритмах адаптивного сжатия, в которых схема кодирования есть схема приспособления к текущим особенностям передаваемого потока символов. Так как схема кодирования известна как кодеру, так и декодеру, сжатое сообщение будет восстановлено приемником.

Обобщением этого способа является алгоритм, основанный на *словаре сжатия данных*. В нем происходит выделение и запоминание в словаре повторяющихся цепочек символов, которые кодируются цепочками меньшей длины.

Интересен *алгоритм "Стопка книг",* когда код символа равен его порядковому номеру в списке. Появление символа в кодируемом потоке вызывает его перемещение в начало списка. Очевидно, что часто встречающиеся символы будут тяготеть к малым номерам, а они кодируются более короткими цепочками 1 и 0.

Среди форматов представления текстовой информации наиболее популярными являются форматы: *doc, pdf, html, xml.*

Алгоритм десятичной упаковки - очевидный способ сжатия *числовой* информации, представленной в коде ASCII, заключается в использовании сокращенного кода с 4 битами на символ вместо 8 (7), если передается набор символов, включающий только 10 цифр, символы "точка", "запятая" и "пробел". Просмотр таблицы ASCII показывает, что старшие 3 бита всех кодов десятичных цифр содержат комбинацию 011. Таким образом, поместив в заголовок кадра соответствующий *управляющий символ*, можно существенно сократить длину кадра.

0 – $(011\ 0000)_2$ – $(30)_{16}$ – $(48)_{10}$
1 – $(011\ 0001)_2$ – $(31)_{16}$ – $(49)_{10}$
……
9 – $(011\ 1001)_2$ – $(39)_{16}$ – $(57)_{10}$

Из сказанного выше следует, что: **а)** нет алгоритма, одинаково эффективного для данных разной природы; **б)** приведенные алгоритмы рассчитаны на сжатие данных, в которых есть последовательности *одинаковых символов* **или** *одни символы встречаются чаще других*.

Алгоритмы сжатия с потерями

Алгоритмы JPEG - разработаны группой **J**oint **P**hotographic **E**xpert **G**roup и ориентированы на сжатие *неподвижных изображений*. Основаны на потере *малосущественной информации* (не различимые для глаза *оттенки* кодируются одинаково, коды могут стать короче). Передаваемая последовательность данных делится на *непересекающиеся блоки изображений* по 8x8 пикселов, в каждом блоке производится дискретное косинусное преобразование (DCT) неподвижного изображения, устраняются высокие частоты, передаются коэффициенты разложения для оставшихся частот, по ним в приемнике изображение восстанавливается.

Алгоритм M-JPEG – кодек M-JPEG – стандартный алгоритм сжатия потока MPEG-данных (*движущихся изображений*), предложенный группой M-JPEG; используется для компрессии *видео*, в котором каждый отдельный кадр сжимается по методу JPEG.

Алгоритмы MPEG (Moving Pictures Experts Group) – представляет собой открытый (т.е. не требующий оплаты за использование) *стандарт* на сжатие и воспроизведение *движущихся изображений*, разработанный группой MPEG, а также формат хранения сжатого (до 1:200) файла; ориентированы на обработку *видео*. При формировании потока данных исходят из предположения о том, что 2 соседних кадра в видеопоследовательности мало отличаются. Опорные кадры сжимают по методу JPEG и передают относительно редко. В основном передаются изменения между соседними кадрами.

Сжатая информация упаковывается в файлы в форматах: *avi, mov, mpg, wmv и др.*

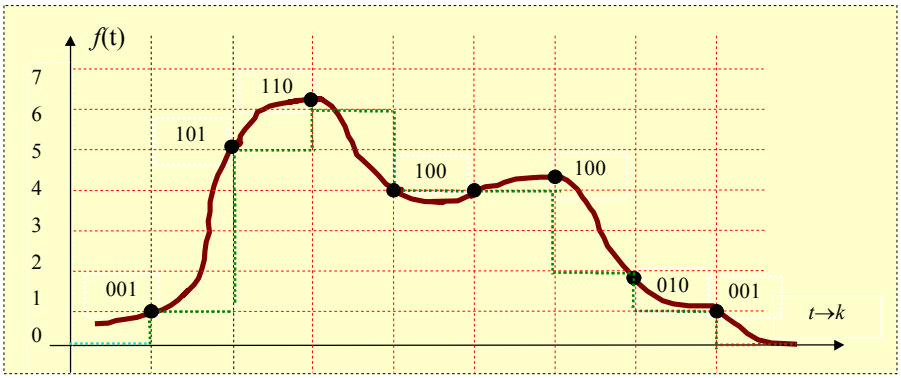
Методы *MPEG* стали мировыми стандартами для цифрового телевидения.

Для представления звуковых данных в цифровой форме используется импульсно-кодовая модуляция (ИКМ). Для высококачественного воспроизведения звука выделяют полосу 20 кГц. Следовательно, частота дискретизации должна быть не менее 40 кГц. Если каждый отсчет кодировать двумя байтами, то информационная скорость воспроизведения будет равна 640 кбит/с и для записи звука продолжительностью 1 мин потребуется память около 4,8 Мбайт.

Поэтому используются специальные алгоритмы сжатия аудио-информации с ее представлением в форматах: *mp3, wave, wma, vqf и др.*

В некоторых разновидностях метода ИКМ используется *алгоритм относительного кодирования*, когда при передаче *числовых* данных с небольшими отклонениями между последовательными отсчетами (числами) осуществляется передача только этих отклонений вместе с известным опорным значением.

При *дифференциальной* (разностной) ИКМ (ДИКМ) вместо кодирования отсчетов кодируются *разности* между соседними отсчетами (они по величине меньше самих отсчетов).



Адаптивная ДИКМ (АДИКМ) - система ДИКМ с адаптацией квантователя (АЦП и ЦАП) и предсказателя. При АДИКМ оцифровывается не сам сигнал, а его отклонение от предсказанного значения.

Т.о., к методам сжатия относят также *методы разностного кодирования*, так как разности амплитуд отсчетов представляются меньшим числом разрядов, чем сами амплитуды.

Разностное кодирование реализовано в методах *дельта-модуляции* и ее разновидностях, а также *предсказывающие* (*предикативные*) *методы*, которые основаны на экстраполяции значений отсчетов, и если выполнено условие $A_р$-$A_п$>d, то отсчет должен быть передан, иначе он является избыточным; здесь $A_р$ и $A_п$ - амплитуды реального и предсказанного отсчетов, d - допуск (допустимая погрешность представления амплитуд).

На практике используют ряд алгоритмов сжатия, каждый из которых применим к определенному типу данных. Некоторые *модемы* (называемые *интеллектуальными*) предлагают адаптивное сжатие, при котором в зависимости от передаваемых данных выбирается определенный алгоритм сжатия.

Многие модели коммуникационного оборудования, такие как *модемы*, *мосты*, *коммутаторы* и *маршрутизаторы*, поддерживают протоколы *динамической* компрессии, позволяющие обеспечить коэффициент сжатия 1:4 или 1:8. Реальный коэффициент компрессии зависит от типа передаваемых данных, так графические и текстовые данные обычно сжимаются хорошо, а коды программ – хуже.

Сжатие данных увеличивает пропускную способность линии связи. На передающем узле данные автоматически сжимаются, а принимающий узел их восстанавливает.