Протоколы канального уровня

Функции КУ:

- Формирование кадра
- Контроль ошибок и повышение достоверности
- Обеспечение кодонезависимой передачи
- Восстановление исходной последовательности блоков на приемной стороне
- Управление потоком данных на уровне звена
- Устранение последствий потерь или дублирования кадров

Канальный уровень обеспечивает передачу пакетов данных, поступающих от протоколов верхних уровней, узлу назначения, адрес которого также указывает протокол верхнего уровня.

Протоколы канального уровня оформляют переданные им пакеты в кадры собственного формата, помещая указанный адрес назначения в одно из полей такого кадра, а также сопровождая кадр контрольной суммой. Протокол канального уровня имеет локальный смысл, он предназначен для доставки кадров данных, как правило, в пределах сетей с простой топологией связей и однотипной или близкой технологией, например в односегментных сетях Ethernet или же в многосегментных сетях Ethernet и Token Ring иерархической топологии, разделенных только мостами и коммутаторами.

Во всех этих конфигурациях адрес назначения имеет локальный смысл для данной сети и не изменяется при прохождении кадра от узла-источника к узлу назначения. Возможность передавать данные между локальными сетями разных технологий связана с тем, что в этих технологиях используются адреса одинакового формата, к тому же производители сетевых адаптеров обеспечивают уникальность адресов независимо от технологии.

Другой областью действия протоколов канального уровня являются связи типа «точка-точка» глобальных сетей, когда протокол канального уровня ответственен за доставку кадра непосредственному соседу. Адрес в этом случае не имеет принципиального значения, а на первый план выходит способность протокола восстанавливать искаженные и утерянные кадры, так как плохое качество территориальных каналов, особенно коммутируемых телефонных, часто требует выполнения подобных действий.

Если же перечисленные выше условия не соблюдаются, например связи между сегментами Ethernet имеют петлевидную структуру, либо объединяемые сети используют различные способы адресации, как это имеет место в сетях Ethernet и X.25, то протокол канального уровня не может в одиночку справиться с задачей передачи кадра между узлами и требует помощи протокола сетевого уровня.

Наиболее существенными характеристиками метода передачи, а значит, и протокола, работающего на канальном уровне, являются следующие:

- асинхронный/синхронный;
- символьно-ориентированный/бит-ориентированный;
- с предварительным установлением соединения/дейтаграммный;
- с обнаружением искаженных данных/без обнаружения;
- с обнаружением потерянных данных/без обнаружения;
- с восстановлением искаженных и потерянных данных/без восстановления;
- с поддержкой динамической компрессии данных/без поддержки.

Многие из этих свойств характерны не только для протоколов канального уровня, но и для протоколов более высоких уровней.

5.1. Асинхронные протоколы

Асинхронные протоколы представляют собой наиболее старый способ связи. Эти протоколы оперируют не с кадрами, а с отдельными символами, которые представлены байтами со старт-стоповыми символами. Асинхронные протоколы ведут свое происхождение от тех времен, когда два человека связывались с помощью телетайпов по каналу «точка-точка». С развитием техники асинхронные протоколы стали применяться для связи телетайпов, разного рода клавиатур и дисплеев с вычислительными машинами. Единицей передаваемых данных был не кадр данных, а отдельный символ. Некоторые символы имели управляющий характер, например символ <CR> предписывал телетайпу или дисплею выполнить возврат каретки на начало строки. В этих протоколах существуют управляющие последовательности, обычно начинающиеся с символа <ESC>. Эти последовательности вызывали на управляемом устройстве достаточно сложные действия - например, загрузку нового шрифта на принтер.

В асинхронных протоколах применяются стандартные наборы символов, чаще всего ASCII или EBCDIC. Так как первые 32 или 27 кодов в этих наборах являются специальными кодами, которые не

отображаются на дисплее или принтере, то они использовались асинхронными протоколами для управления режимом обмена данными. В самих пользовательских данных, которые представляли собой буквы, цифры, а также такие знаки, как @, %, $ и т. п., специальные символы никогда не встречались, так что проблемы их отделения от пользовательских данных не существовало.

Постепенно асинхронные протоколы усложнялись и стали наряду с отдельными символами использовать целые блоки данных, то есть кадры. Например, протокол XModem.

5.2. Синхронные символьно-ориентированные и бит-ориентированные протоколы

В синхронных протоколах между пересылаемыми символами (байтами) нет стартовых и стоповых сигналов, поэтому отдельные символы в этих протоколах пересылать нельзя. Все обмены данными осуществляются кадрами, которые имеют в общем случае заголовок, поле данных и концевик (рис. 5.1). Все биты кадра передаются непрерывным синхронным потоком, что значительно ускоряет передачу данных.

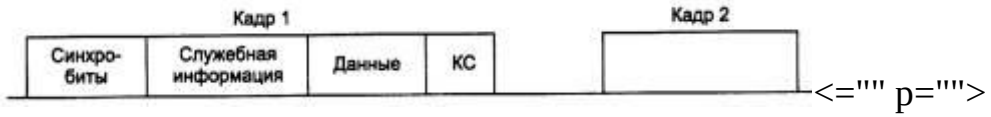


Рис. 5.1. Кадры синхронных протоколов

Так как байты в этих протоколах не отделяются друг от друга служебными сигналами, то одной из первых задач приемника является распознавание границы байт. Затем приемник должен найти начало и конец кадра, а также определить границы каждого поля кадра - адреса назначения, адреса источника, других служебных полей заголовка, поля данных и контрольной суммы, если она имеется.

Большинство протоколов допускает использование в кадре поля данных переменной длины. Иногда и заголовок может иметь переменную длину. Обычно протоколы определяют максимальное значение, которое может иметь длина поля данных. Эта величина называется *максимальной единицей передачи данных (Maximum Transfer Unit, MTU)*. В некоторых протоколах задается также минимальное значение, которое может иметь длина поля данных. Например, протокол Ethernet требует, чтобы поле данных содержало по крайней мере 46 байт данных (если приложение хочет отправить

меньшее количество байт, то оно обязано дополнить их до 46 байт любыми значениями). Другие протоколы разрешают использовать поле данных нулевой длины, например FDDI.

Существуют также протоколы с кадрами фиксированной длины, например, в протоколе ATM кадры фиксированного размера 53 байт, включая служебную информацию. Для таких протоколов необходимо решить только первую часть задачи - распознать начало кадра.

Синхронные протоколы канального уровня бывают двух типов: символьно-ориентированные (байт-ориентированные) и бит-ориентированные. Для обоих характерны одни и те же методы синхронизации бит. Главное различие между ними заключается в методе синхронизации символов и кадров.

Символьно-ориентированные протоколы

Символьно-ориентированные протоколы используются в основном для передачи блоков отображаемых символов, например текстовых файлов. Так как при синхронной передаче нет стоповых и стартовых битов, для синхронизации символов необходим другой метод.

Синхронизация достигается за счет того, что передатчик добавляет два или более управляющих символа, называемых символами SYN, перед каждым блоком символов. В коде ASCII символ SYN имеет двоичное значение 0010110, это несимметричное относительно начала символа значение позволяет легко разграничивать отдельные символы SYN при их последовательном приеме. Символы SYN выполняют две функции: во-первых, они обеспечивают приемнику побитную синхронизацию, во-вторых, как только битовая синхронизация достигается, они позволяют приемнику начать распознавание границ символов SYN. После того как приемник начал отделять один символ от другого, можно задавать границы начала кадра с помощью другого специального символа. Обычно в символьных протоколах для этих целей используется символ STX (Start of TeXt, ASCII 0000010). Другой символ отмечает окончание кадра - ETX (End of TeXt, ASCII 0000011).

Однако такой простой способ выделения начала и конца кадра хорошо работает только в том случае, если внутри кадра нет символов STX и ETX. При подключении к компьютеру алфавитно-цифровых терминалов такая проблема действительно не возникает. Так как синхронные символьно-ориентированные протоколы стали использовать и для связи компьютера с компьютером, а в этом случае данные внутри кадра могут быть любые, если, например, между компьютерами передается программа.

Наиболее популярным протоколом такого типа был протокол BSC компании IBM. Он работал в двух режимах - непрозрачном, в котором некоторые специальные символы внутри кадра запрещались, и прозрачном, в котором разрешалась передачи внутри кадра любых символов, в том числе и ETX. Прозрачность достигалась за счет того, что перед управляющими символами STX и ETX всегда вставлялся символ DLE (Data Link Escape). Такая процедура называется *стаффингом* символов (stuff - всякая всячина, заполнитель). А если в поле данных кадра встречалась последовательность DLE ETX, то передатчик удваивал символ DLE, то есть порождал последовательность DLE DLE ETX. Приемник, встретив подряд два символа DLE DLE, всегда удалял первый, но оставшийся DLE уже не рассматривал как начало управляющей последовательности, то есть оставшиеся символы DLE ETX считал просто пользовательскими данными.

Бит-ориентированные протоколы

Потребность в паре символов в начале и конце каждого кадра вместе с дополнительными символами DLE означает, что символьно-ориентированная передача не эффективна для передачи двоичных данных, так как приходится в поле данных кадра добавлять достаточно много избыточных данных. Кроме того, формат управляющих символов для разных кодировок различен, например, в коде ASCII символ SYN равен 0010110, а в коде EBCDIC - 00110010. Так что этот метод допустим только с определенным типом кодировки, даже если кадр содержит чисто двоичные данные.

Чтобы преодолеть эти проблемы, сегодня почти всегда используется более универсальный метод, называемый бит-ориентированной передачей. Этот метод сейчас применяется при передаче как двоичных, так и символьных данных.

На рис. 5.2 показаны 3 различные схемы бит-ориентированной передачи. Они отличаются способом обозначения начала и конца каждого кадра.

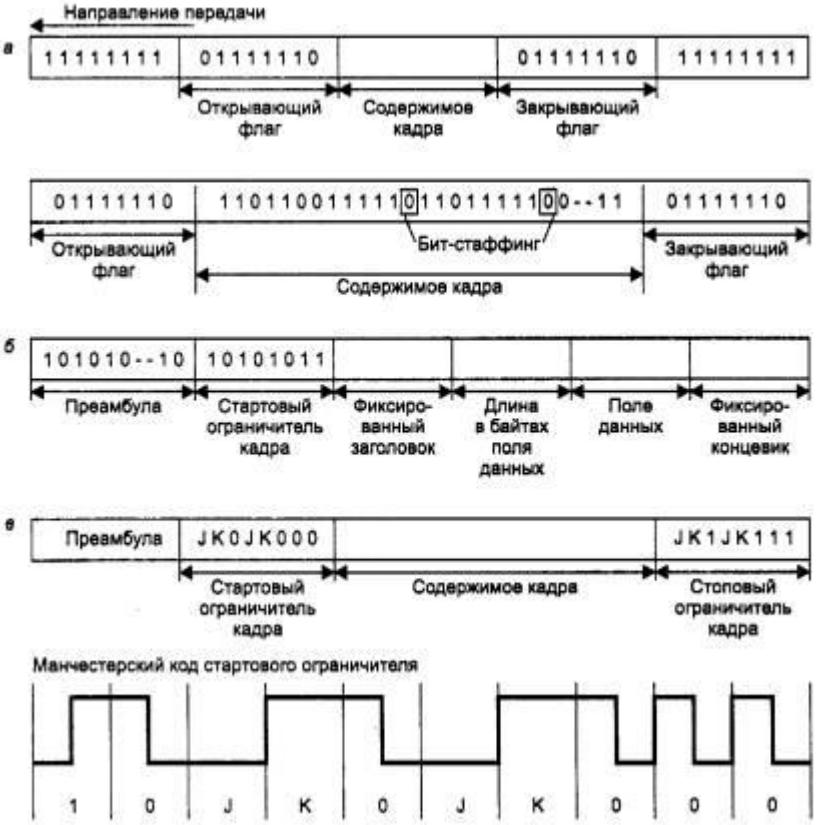


Рис. 5.2. Способы выделения начало и конца кадра при синхронной передаче

Первая схема, показанная на рис. 5.2, а, похожа на схему с символами STX и ETX в символьно-ориентированных протоколах. Начало и конец каждого кадра отмечается одной и той же 8-битовой последовательностью - 01111110, называемой флагом. Термин «бит-ориентированный» используется потому, что принимаемый поток бит сканируется приемником на побитовой основе для обнаружения стартового флага, а затем во время приема для обнаружения стопового флага. Поэтому длина кадра в этом случае не обязательно должна быть кратна 8 бит.

Чтобы обеспечить синхронизацию приемника, передатчик посылает последовательность байтов простоя (каждый состоит из 11111111), предшествующую стартовому флагу.

Для достижения прозрачности данных в этой схеме необходимо, чтобы флаг не присутствовал в поле данных кадра. Это достигается с помощью приема, известного как вставка 0 бита, - *бит-стаффинга*. Схема вставки бита работает только во время передачи поля данных кадра. Если эта схема обнаруживает, что подряд передано пять 1, то она автоматически вставляет дополнительный 0 (даже если после этих пяти 1 шел 0). Поэтому последовательность 01111110 никогда не появится в поле данных кадра. Аналогичная схема работает в

приемнике и выполняет обратную функцию. Когда после пяти 1 обнаруживается 0, он автоматически удаляется из поля данных кадра. Бит-стаффинг гораздо более экономичен, чем байт-стаффинг, так как вместо лишнего байта вставляется один бит, следовательно, скорость передачи пользовательских данных в этом случае замедляется в меньшей степени.

Во второй схеме (рис. 5.2, б) для обозначения начала кадра имеется только стартовый флаг, а для определения конца кадра используется поле длины кадра, которое при фиксированных размерах заголовка и концевика чаще всего имеет смысл длины поля данных кадра. Эта схема наиболее применима в локальных сетях. В этих сетях для обозначения факта незанятости среды в исходном состоянии по среде вообще не передается никаких символов. Чтобы все остальные станции вошли в битовую синхронизацию, посылающая станция предваряет содержимое кадра последовательностью бит, известной как преамбула, которая состоит из чередования единиц и нулей 101010... Войдя в битовую синхронизацию, приемник исследует входной поток на побитовой основе, пока не обнаружит байт начала кадра 10101011, который выполняет роль символа STX. За этим байтом следует заголовок кадра, в котором в определенном месте находится поле длины поля данных. Таким образом, в этой схеме приемник просто отсчитывает заданное количество байт, чтобы определить окончание кадра.

Третья схема (рис. 5.2, в) использует для обозначения начала и конца кадра флаги, которые включают запрещенные для данного кода сигналы (code violations, V). Например, при манчестерском кодировании вместо обязательного изменения полярности сигнала в середине тактового интервала уровень сигнала остается неизменным и низким (запрещенный сигнал J) или неизменным и высоким (запрещенный сигнал К). Начало кадра отмечается последовательностью JKOJKOOO, а конец - последовательностью JK1JK100. Этот способ очень экономичен, так как не требует ни бит-стаффинга, ни поля длины, но его недостаток заключается в зависимости от принятого метода физического кодирования. При использовании избыточных кодов роль сигналов J и К играют запрещенные символы, например, в коде 4B/5B этими символами являются коды 11000 и 10001.

Каждая из трех схем имеет свои преимущества и недостатки. Флаги позволяют отказаться от специального дополнительного поля, но требуют специальных мер: либо по разрешению размещения флага в поле данных за счет бит-стаффинга, либо по использованию в качестве

флага запрещенных сигналов, что делает эту схему зависимой от способа кодирования.

Протоколы с гибким форматом кадра

Для большей части протоколов характерны кадры, состоящие из служебных полей фиксированной длины. Исключение делается только для поля данных, с целью экономной пересылки как небольших квитанций, так и больших файлов. Способ определения окончания кадра путем задания длины поля данных, рассмотренный выше, как раз рассчитан на такие кадры с фиксированной структурой и фиксированными размерами служебных полей.

Однако существует ряд протоколов, в которых кадры имеют гибкую структуру. Например, к таким протоколам относятся очень популярный прикладной протокол управления сетями SNMP, а также протокол канального уровня PPP, используемый для соединений типа «точка-точка». Кадры таких протоколов состоят из неопределенного количества полей, каждое из которых может иметь переменную длину. Начало такого кадра отмечается некоторым стандартным образом, например с помощью флага, а затем протокол последовательно просматривает поля кадра и определяет их количество и размеры. Каждое поле обычно описывается двумя дополнительными полями фиксированного размера. Например, если в кадре встречается поле, содержащее некоторую символьную строку, то в кадр вставляются три поля:

Тип	Длина	Значение
string	6	public

Дополнительные поля «Тип» и «Длина» имеют фиксированный размер в один байт, поэтому протокол легко находит границы поля «Значение». Так как количество таких полей также неизвестно, для определения общей длины кадра используется либо общее поле «Длина», которое помещается в начале кадра и относится ко всем полям данных, либо закрывающий флаг.

5.3. Передача с установлением соединения и без установления соединения

При передаче кадров данных на канальном уровне используются как дейтаграммные процедуры, работающие без установления

соединения *(connectionless)*, так и процедуры с предварительным установлением логического соединения *(connection-oriented)*.

При дейтаграммной передаче кадр посылается в сеть «без предупреждения», и никакой ответственности за его утерю протокол не несет (рис. 5.3, а). Предполагается, что сеть всегда готова принять кадр от конечного узла. Дейтаграммный метод работает быстро, так как никаких предварительных действий перед отправкой данных не выполняется. Однако при таком методе трудно организовать в рамках протокола отслеживание факта доставки кадра узлу назначения. Этот метод не гарантирует доставку пакета.

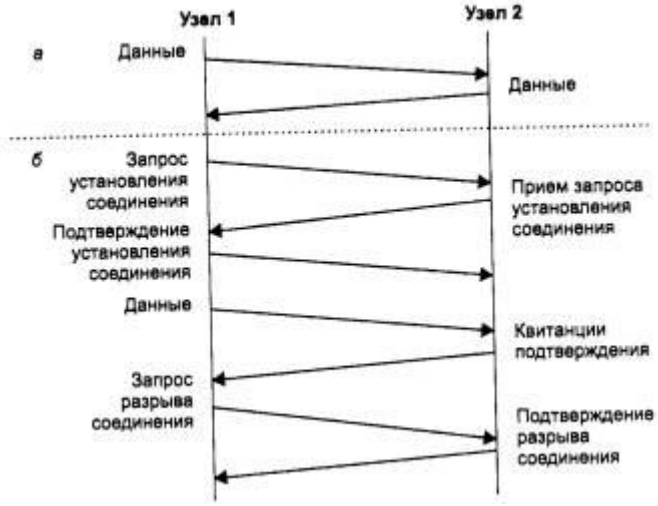


Рис. 5.3. Протоколы без установления соединения (а) и с установлением соединения (б)

Передача с установлением соединения более надежна, но требует больше времени для передачи данных и вычислительных затрат от конечных узлов.

В этом случае узлу-получателю отправляется служебный кадр специального формата с предложением установить соединение (рис. 5.3, б). Если узел-получатель согласен с этим, то он посылает в ответ другой служебный кадр, подтверждающий установление соединения и предлагающий для данного логического соединения некоторые параметры, например идентификатор соединения, максимальное значение поля данных кадров, которые будут использоваться в рамках данного соединения, и т. п.

Узел-инициатор соединения может завершить процесс установления соединения отправкой третьего служебного кадра, в котором сообщит, что предложенные параметры ему подходят. На этом логическое соединение считается установленным, и в его рамках

можно передавать информационные кадры с пользовательскими данными.

После передачи некоторого законченного набора данных, например определенного файла, узел инициирует разрыв данного логического соединения, посылая соответствующий служебный кадр.

В отличие от протоколов дейтаграммного типа, которые поддерживают только один тип кадра - информационный, протоколы, работающие по процедуре с установлением соединения, должны поддерживать несколько типов кадров - служебные, для установления (и разрыва) соединения, и информационные, переносящие собственно пользовательские данные.

Логическое соединение обеспечивает передачу данных как в одном направлении - от инициатора соединения, так и в обоих направлениях.

Процедура установления соединения может использоваться для достижения различных целей.

- Для взаимной аутентификации либо пользователей, либо оборудования (маршрутизаторы тоже могут иметь имена и пароли, которые нужны для уверенности в том, что злоумышленник не подменил корпоративный маршрутизатор и не отвел поток данных в свою сеть для анализа).
- Для согласования изменяемых параметров протокола: MTU, различных тайм-аутов и т. п.
- Для обнаружения и коррекции ошибок. Установление логического соединения дает точку отсчета для задания начальных значений номеров кадров. При потере нумерованного кадра приемник, во-первых, получает возможность обнаружить этот факт, а во-вторых, он может сообщить передатчику, какой в точности кадр нужно передать повторно.
- В некоторых технологиях процедуру установления логического соединения используют при динамической настройке коммутаторов сети для маршрутизации всех последующих кадров, которые будут проходить через сеть в рамках данного логического соединения. Так работают сети технологий X.25, frame relay и ATM.

Как видно из приведенного списка, при установлении соединения могут преследоваться разные цели, в некоторых случаях - несколько одновременно.

Рассмотрим использование логического соединения для обнаружения и коррекции ошибок.

5.4. Обнаружение и коррекция ошибок

Канальный уровень должен обнаруживать ошибки передачи данных, связанные с искажением бит в принятом кадре данных или с потерей кадра, и по возможности их корректировать.

Большая часть протоколов канального уровня выполняет только первую задачу - обнаружение ошибок, считая, что корректировать ошибки, то есть повторно передавать данные, содержавшие искаженную информацию, должны протоколы верхних уровней. Так работают такие популярные протоколы локальных сетей, как Ethernet, Token Ring, FDDI и другие. Однако существуют протоколы канального уровня, например LLC2 или LAP-B, которые самостоятельно решают задачу восстановления искаженных или потерянных кадров.

Очевидно, что протоколы должны работать наиболее эффективно в типичных условиях работы сети. Поэтому для сетей, в которых искажения и потери кадров являются очень редкими событиями, разрабатываются протоколы типа Ethernet, в которых не предусматриваются процедуры устранения ошибок. Действительно, наличие процедур восстановления данных потребовало бы от конечных узлов дополнительных вычислительных затрат, которые в условиях надежной работы сети являлись бы избыточными.

Напротив, если в сети искажения и потери случаются часто, то желательно уже на канальном уровне использовать протокол с коррекцией ошибок, а не оставлять эту работу протоколам верхних уровней. Протоколы верхних уровней, например транспортного или прикладного, работая с большими тайм-аутами, восстановят потерянные данные с большой задержкой. В глобальных сетях первых поколений, например сетях X.25, которые работали через ненадежные каналы связи, протоколы канального уровня всегда выполняли процедуры восстановления потерянных и искаженных кадров.

Поэтому нельзя считать, что один протокол лучше другого потому, что он восстанавливает ошибочные кадры, а другой протокол - нет. Каждый протокол должен работать в тех условиях, для которых он разработан.

5.5. Методы обнаружения ошибок

Все методы обнаружения ошибок основаны на передаче в составе кадра данных служебной избыточной информации, по которой можно судить с некоторой степенью вероятности о достоверности принятых данных. Эту служебную информацию принято называть *контрольной суммой* или (*последовательностью контроля кадра - Frame Check Sequence, FCS*).

Контрольная сумма вычисляется как функция от основной информации, причем необязательно только путем суммирования. Принимающая сторона повторно вычисляет контрольную сумму кадра по известному алгоритму и в случае ее совпадения с контрольной суммой, вычисленной передающей стороной, делает вывод о том, что данные были переданы через сеть корректно.

Существует несколько распространенных алгоритмов вычисления контрольной суммы, отличающихся вычислительной сложностью и способностью обнаруживать ошибки в данных.

Контроль по паритету представляет собой наиболее простой метод контроля данных. В то же время это наименее мощный алгоритм контроля, так как с его помощью можно обнаружить только одиночные ошибки в проверяемых данных. Метод заключается в суммировании по модулю 2 всех бит контролируемой информации. Например, для данных 100101011 результатом контрольного суммирования будет значение 1. Результат суммирования также представляет собой один бит данных, который пересылается вместе с контролируемой информацией. При искажении при пересылке любого одного бита исходных данных (или контрольного разряда) результат суммирования будет отличаться от принятого контрольного разряда, что говорит об ошибке. Однако двойная ошибка, например 110101010, будет неверно принята за корректные данные. Поэтому контроль по паритету применяется к небольшим порциям данных, как правило, к каждому байту, что дает коэффициент избыточности для этого метода 1/8. Метод редко применяется в вычислительных сетях из-за его большой избыточности и невысоких диагностических способностей.

Вертикальный и горизонтальный контроль по паритету представляет собой модификацию описанного выше метода. Его отличие состоит в том, что исходные данные рассматриваются в виде матрицы, строки которой составляют байты данных. Контрольный разряд подсчитывается отдельно для каждой строки и для каждого столбца матрицы. Этот метод обнаруживает большую часть двойных ошибок, однако обладает еще большей избыточностью. На практике сейчас также почти не применяется.

Циклический избыточный контроль (Cyclic Redundancy Check, CRC) является в настоящее время наиболее популярным методом контроля в вычислительных сетях (и не только в сетях, например, этот метод широко применяется при записи данных на диски и дискеты).

Метод основан на рассмотрении исходных данных в виде одного многоразрядного двоичного числа. Например, кадр стандарта Ethernet, состоящий из 1024 байт, будет рассматриваться как одно число, состоящее из 8192 бит. В качестве контрольной информации рассматривается остаток от деления этого числа на известный делитель R. Обычно в качестве делителя выбирается семнадцатиразрядное или тридцати трехразрядное число, чтобы остаток от деления имел длину 16 разрядов (2 байт) или 32 разряда (4 байт).

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

Примечание: Существует несколько модифицированная процедура вычисления остатка, приводящая к получению в случае отсутствия ошибок известного ненулевого остатка, что является более надежным показателем корректности.

Этот метод обладает более высокой вычислительной сложностью, но его диагностические возможности гораздо выше, чем у методов контроля по паритету. Метод CRC обнаруживает все одиночные ошибки, двойные ошибки и ошибки в нечетном числе бит. Метод обладает также невысокой степенью избыточности. Например, для кадра Ethernet размером в 1024 байт контрольная информация длиной в 4 байт составляет только 0,4 %.

5.6. Методы восстановления искаженных и потерянных кадров

Краткое изложение...

Используются **протоколы повторной передачи кадров.**

Цель: надежная доставка кадров по ненадежному каналу.

Реализован механизм автоматического запроса повторной передачи (ARQ -Automatic Repeat Quest).

В качестве характеристики протоколов выступают - корректность и эффективность.

Кадры, не принятые корректно, посылаются повторно.

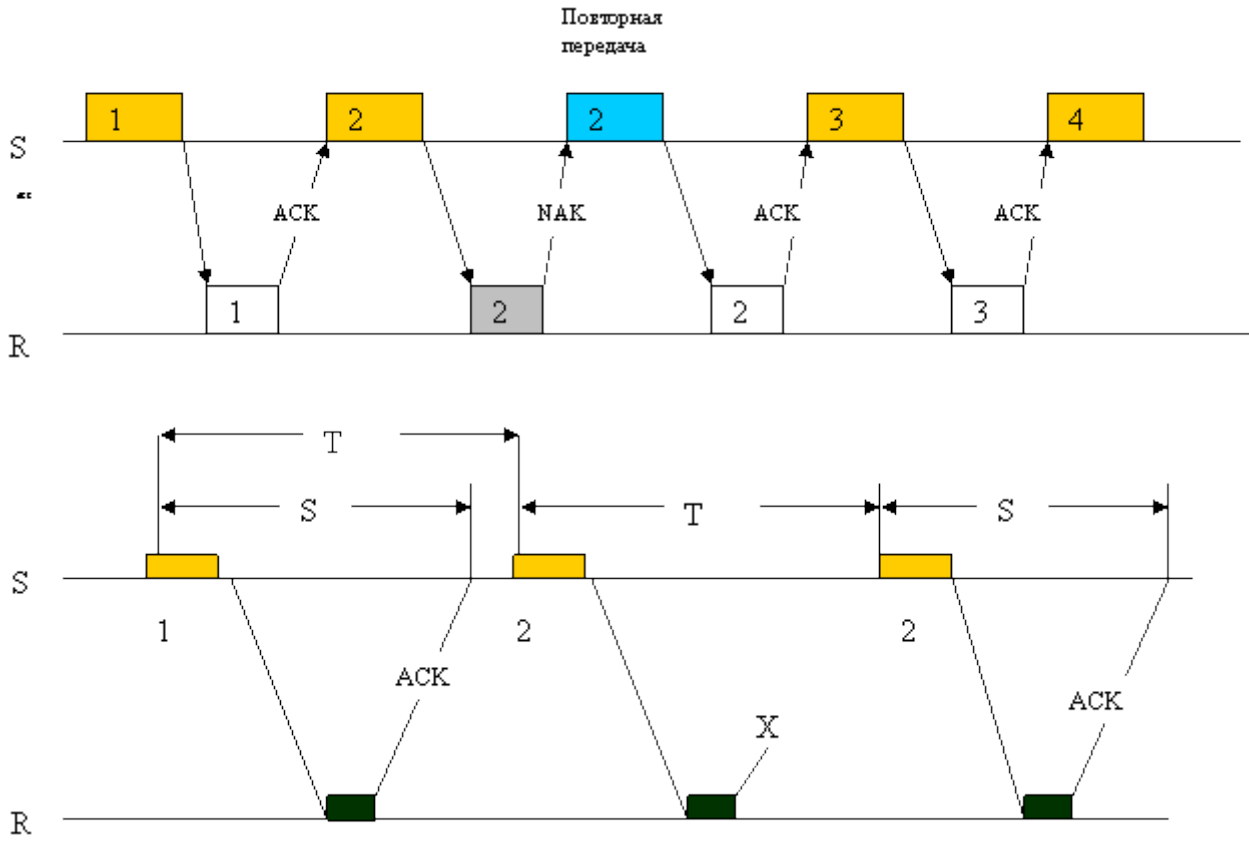
Отправитель информируется об ошибках передачи с помощью

таймера и подтверждений.

Протоколы повторной передачи корректны, если они позволяют получателю принять точно одну правильную копию каждого кадра.

Эффективность протокола повторной передачи равна средней скорости успешной доставки кадров ($R_{эф}$), деленной на скорость передачи в канале.

Протокол с остановкой и ожиданием (SWP - Stop-and-Wait Protocol). Данный протокол реализует алгоритм РОС-ОЖ.



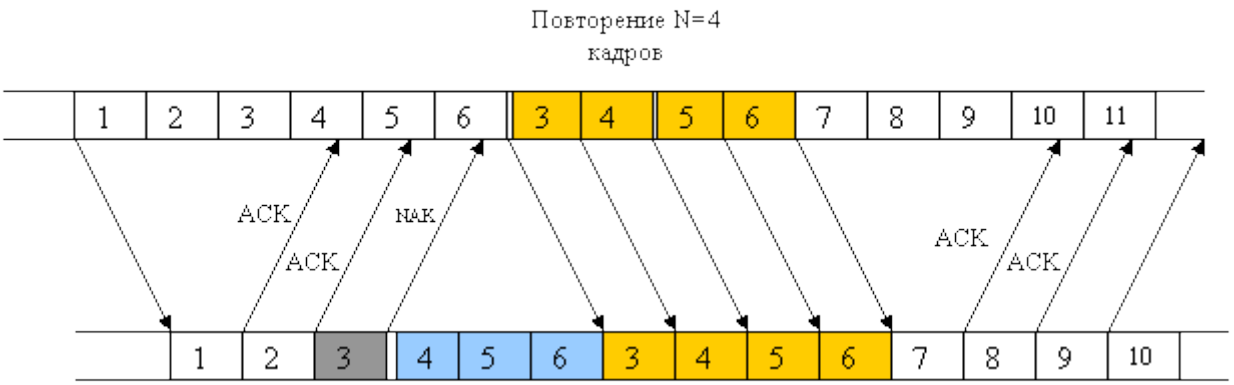
Алгоритм работы протокола **SWP**

Эффективность протокола SWP равна

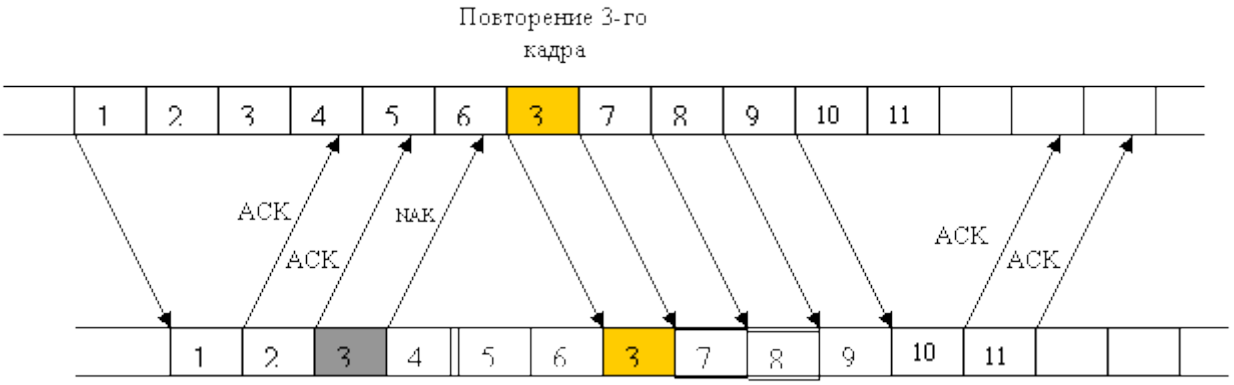
$$\eta = \frac{\tau}{S + T*(1-p)/p}$$

где τ - время передачи кадра;
p - вероятность того, что подтверждение поступает правильно.

Протокол повторной передачи с возвращением на N кадров назад. (GBN - Go Back N).



Протокол повторной передачи с выборочным (селективным) повторением. (SRP - Selective Repeat Protocol).



Подробное изложение...

Методы коррекции ошибок в вычислительных сетях основаны на повторной передаче кадра данных в том случае, если кадр теряется и не доходит до адресата или приемник обнаружил в нем искажение информации.

Чтобы убедиться в необходимости повторной передачи данных, отправитель нумерует отправляемые кадры и для каждого кадра ожидает от приемника так называемой *положительной квитанции* - служебного кадра, извещающего о том, что исходный кадр был получен и данные в нем оказались корректными.

Время этого ожидания ограничено - при отправке каждого кадра передатчик запускает таймер, и, если по его истечении положительная квитанция на получена, кадр считается утерянным. Приемник в случае получения кадра с искаженными данными может отправить *отрицательную квитанцию* - явное указание на то, что данный кадр нужно передать повторно.

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

Метод с простоями (Idle Source) требует, чтобы источник, пославший кадр, ожидал получения квитанции (положительной или отрицательной) от приемника и только после этого посылал следующий кадр (или повторял искаженный).

Если же квитанция не приходит в течение тайм-аута, то кадр (или квитанция) считается утерянным и его передача повторяется. На рис. 5.4, а видно, что в этом случае производительность обмена данными существенно снижается, - хотя передатчик и мог бы послать следующий кадр сразу же после отправки предыдущего, он обязан ждать прихода квитанции. Снижение производительности этого метода коррекции особенно заметно на низкоскоростных каналах связи, то есть в территориальных сетях.

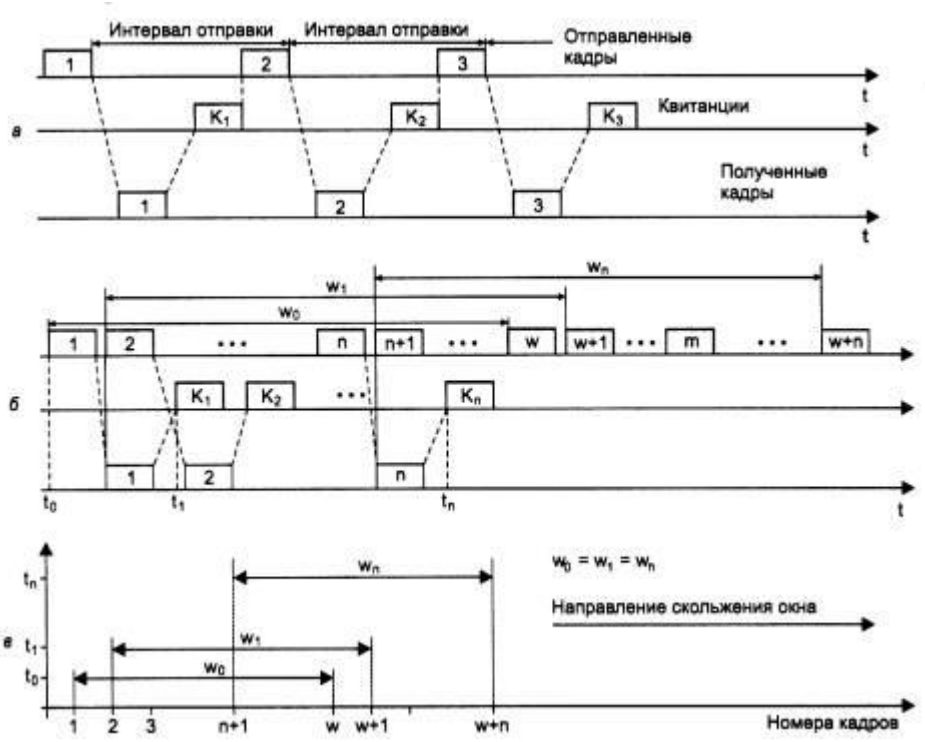


Рис. 5.4. Методы восстановления искаженных и потерянных кадров

Второй метод называется методом «*скользящего* окна» *(sliding window)*. В этом методе для повышения коэффициента использования линии источнику разрешается передать некоторое количество кадров в непрерывном режиме, то есть в максимально возможном для источника темпе, без получения на эти кадры положительных ответных квитанций. (Далее, где это не искажает существо рассматриваемого вопроса, положительные квитанции для краткости будут называться просто «квитанциями».) Количество кадров, которые разрешается

передавать таким образом, называется размером окна. Рисунок 5.4, б иллюстрирует данный метод для окна размером в W кадров.

В начальный момент, когда еще не послано ни одного кадра, окно определяет диапазон кадров с номерами от 1 до W включительно. Источник начинает передавать кадры и получать в ответ квитанции. Для простоты предположим, что квитанции поступают в той же последовательности, что и кадры, которым они соответствуют. В момент t_1 при получении первой квитанции $К_1$ окно сдвигается на одну позицию, определяя новый диапазон от 2 до (W+1).

Процессы отправки кадров и получения квитанций идут достаточно независимо друг от друга. Рассмотрим произвольный момент времени t_n, когда источник получил квитанцию на кадр с номером n. Окно сдвинулось вправо и определило диапазон разрешенных к передаче кадров от (n+1) до (W+n). Все множество кадров, выходящих из источника, можно разделить на перечисленные ниже группы (рис. 5.4, б).

- *Кадры с номерами от 1 до n* уже были отправлены и квитанции на них получены, то есть они находятся за пределами окна слева.
- *Кадры, начиная с номера (n+1) и кончая номером (W+n)*, находятся в пределах окна и потому могут быть отправлены не дожидаясь прихода какой-либо квитанции. Этот диапазон может быть разделен еще на два поддиапазона:
  - кадры с номерами от (n+1) до m, которые уже отправлены, но квитанции на них еще не получены;
  - кадры с номерами от m до (W+n), которые пока не отправлены, хотя запрета на это нет.
- *Все кадры с номерами, большими или равными (W+n)*, находятся за пределами окна справа и поэтому пока не могут быть отправлены.

Перемещение окна вдоль последовательности номеров кадров показано на рис. 5.4, в. Здесь t_0 - исходный момент, t_1 и t_n - моменты прихода квитанций на первый и n-й кадр соответственно. Каждый раз, когда приходит квитанция, окно сдвигается, но его размер при этом не меняется и остается равным W. Заметим, что хотя в данном примере размер окна в процессе передачи остается постоянным, в реальных протоколах (например, TCP) можно встретить варианты данного алгоритма с изменяющимся размером окна.

Итак, при отправке кадра с номером n источнику разрешается передать еще W-1 кадров до получения квитанции на кадр n, так что в сеть последним уйдет кадр с номером (W+n). Если же за это время

квитанция на кадр n так и не пришла, то процесс передачи приостанавливается, и по истечении некоторого тайм-аута кадр n (или квитанция на него) считается утерянным, и он передается снова.

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

Метод скользящего окна более сложен в реализации, чем метод с простоями, так как передатчик должен хранить в буфере все кадры, на которые пока не получены положительные квитанции. Кроме того, требуется отслеживать несколько параметров алгоритма: размер окна W, номер кадра, на который получена квитанция, номер кадра, который еще можно передать до получения новой квитанции.

Приемник может не посылать квитанции на каждый принятый корректный кадр. Если несколько кадров пришли почти одновременно, то приемник может послать квитанцию только на последний кадр. При этом подразумевается, что все предыдущие кадры также дошли благополучно.

Некоторые методы используют отрицательные квитанции. Отрицательные квитанции бывают двух типов - групповые и избирательные. Групповая квитанция содержит номер кадра, начиная с которого нужно повторить передачу всех кадров, отправленных передатчиком в сеть. Избирательная отрицательная квитанция требует повторной передачи только одного кадра.

Метод скользящего окна реализован во многих протоколах: LLC2, LAP-B, X.25, TCP, Novell NCP Burst Mode.

Метод с простоями является частным случаем метода скользящего окна, когда размер окна равен единице.

Метод скользящего окна имеет два параметра, которые могут заметно влиять на эффективность передачи данных между передатчиком и приемником, - размер окна и величина тайм-аута ожидания квитанции. В надежных сетях, когда кадры искажаются и теряются редко, для повышения скорости обмена данными размер окна нужно увеличивать, так как при этом передатчик будет посылать кадры с меньшими паузами. В ненадежных сетях размер окна следует уменьшать, так как при частых потерях и искажениях кадров резко возрастает объем вторично передаваемых через сеть кадров, а значит, пропускная способность сети будет расходоваться во многом вхолостую - полезная пропускная способность сети будет падать.

Выбор тайм-аута зависит не от надежности сети, а от задержек передачи кадров сетью.

Во многих реализациях метода скользящего окна величина окна и тайм-аут выбираются адаптивно, в зависимости от текущего состояния сети.

5.7. Компрессия данных

Компрессия (сжатие) данных применяется для сокращения времени их передачи. Так как на компрессию данных передающая сторона тратит дополнительное время, к которому нужно еще прибавить аналогичные затраты времени на декомпрессию этих данных принимающей стороной, то выгоды от сокращения времени на передачу сжатых данных обычно бывают заметны только для низкоскоростных каналов. Этот порог скорости для современной аппаратуры составляет около 64 Кбит/с. Многие программные и аппаратные средства сети способны выполнять *динамическую компрессию* данных в отличие от статической, когда данные предварительно компрессируются (например, с помощью популярных архиваторов типа WinZip), а уже затем отсылаются в сеть.

На практике может использоваться ряд алгоритмов компрессии, каждый из которых применим к определенному типу данных. Некоторые модемы (называемые интеллектуальными) предлагают *адаптивную компрессию*, при которой в зависимости от передаваемых данных выбирается определенный алгоритм компрессии. Рассмотрим некоторые алгоритмы компрессии данных.

Десятичная упаковка. Когда данные состоят только из чисел, значительную экономию можно получить путем уменьшения количества используемых на цифру бит с 7 до 4, используя простое двоичное кодирование десятичных цифр вместо кода ASCII. Просмотр таблицы ASCII показывает, что старшие три бита всех кодов десятичных цифр содержат комбинацию 011. Если все данные в кадре информации состоят из десятичных цифр, то, поместив в заголовок кадра соответствующий управляющий символ, можно существенно сократить длину кадра.

Относительное кодирование. Альтернативой десятичной упаковке при передаче числовых данных с небольшими отклонениями между последовательными цифрами является передача только этих отклонений вместе с известным опорным значением.

Символьное подавление. Часто передаваемые данные содержат большое количество повторяющихся байт. Например, при передаче черно-белого изображения черные поверхности будут порождать большое количество нулевых значений, а максимально освещенные участки изображения - большое количество байт, состоящих из всех единиц. Передатчик сканирует последовательность передаваемых байт и, если обнаруживает последовательность из трех или более одинаковых байт, заменяет ее специальной трехбайтовой последовательностью, в которой указывает значение байта, количество его повторений, а также отмечает начало этой последовательности специальным управляющим символом.

Коды переменной длины. В этом методе кодирования используется тот факт, что не все символы в передаваемом кадре встречаются с одинаковой частотой. Поэтому во многих схемах кодирования коды часто встречающихся символов заменяют кодами меньшей длины, а редко встречающихся - кодами большей длины. Такое кодирование называется также статистическим кодированием. Из-за того, что символы имеют различную длину, для передачи кадра возможна только бит-ориентированная передача.

При *статистическом кодировании* коды выбираются таким образом, чтобы при анализе последовательности бит можно было бы однозначно определить соответствие определенной порции бит тому или иному символу или же запрещенной комбинации бит. Если данная последовательность бит представляет собой запрещенную комбинацию, то необходимо к ней добавить еще один бит и повторить анализ. Например, если при неравномерном кодировании для наиболее часто встречающегося символа «Р» выбран код 1, состоящий из одного бита, то значение 0 однобитного кода будет запрещенным. Иначе мы сможем закодировать только два символа. Для другого часто встречающегося символа «О» можно использовать код 01, а код 00 оставить как запрещенный. Тогда для символа «А» можно выбрать код 001, для символа «П» - код 0001 и т. п.

Вообще, неравномерное кодирование наиболее эффективно, когда неравномерность распределения частот передаваемых символов достаточна велика, как при передаче длинных текстовых строк. Напротив, при передаче двоичных данных, например кодов программ, оно малоэффективно, так как 8-битовые коды при этом распределены почти равномерно.

Одним из наиболее распространенных алгоритмов, на основе которых строятся неравномерные коды, является алгоритм Хафмана, позволяющий строить коды автоматически, на основании известных

частот символов. Существуют адаптивные модификации метода Хафмана, которые позволяют строить дерево кодов «на ходу», по мере поступления данных от источника.

Многие модели коммуникационного оборудования, такие как модемы, мосты, коммутаторы и маршрутизаторы, поддерживают протоколы динамической компрессии, позволяющие сократить объем передаваемой информации в 4, а иногда и в 8 раз. В таких случаях говорят, что протокол обеспечивает коэффициент сжатия 1:4 или 1:8. Существуют стандартные протоколы компрессии, например V.42bis, а также большое количество нестандартных, фирменных протоколов. Реальный коэффициент компрессии зависит от типа передаваемых данных, так, графические и текстовые данные обычно сжимаются хорошо, а коды программ - хуже.

5.8. Протоколы передачи файлов

Протокол передачи файлов представляет собой набор правил передачи файлов. В его задачи входит:

- исправление возникающих при передаче ошибок,
- передача и прием определенных кодов, служащих для орнизации связи (handshaking),
- выполнение некоторых функций передачи файлов и прекращения передачи.

» Протокол XModem
» Протокол XModem-CRC
» Протокол XModem-1k
» Протокол YModem
» Протокол YModem-g
» Протокол ZModem
» Протокол Kermit

Протокол XModem

Среди протоколов передачи файлов между ПК одним из первых появился XModem. Разработанный еще в 1977 году Вардом Христенсеном, он широко использовался в справочных службах, вводился в недорогие связные программы для ПК и фактически стал стандартом для связи между ПК с использовнием модемов.

Последовательность действий, выполняемых при передаче файла с помощью протокола XModem, показана на рисунке.

передатчик		приемник (задание тайм-аута на интервал в 10 с)
	<=====	"nak"
"soh" 01 FE -данные- "xx"	=====>	
	<=====	"ack"
"soh" 02 FD -данные- "xx"	=====>	(помеха в линии связи)
	<=====	"nak"
"soh" 02 FD -данные- "xx"	=====>	
	<=====	"ack"
"soh" 03 FC -данные- "xx" (знак ack искажен)	=====>	
	<=====	"ack"
"soh" 03 FC -данные- "xx"	=====>	
	<=====	"ack"
"eot"	=====>	
	<=====	любой знак кроме "ack"
"eot"	=====>	
	<=====	"ack"
Передача завершена		

Передающий ПК начинает передачу файла только после приема от принимающего ПК знака "nak". Принимающий ПК передает этот знак до тех пор, пока не начинается передача файла. Если передано девять знаков "nak", а передача файла не началась, то процесс возобновляется вручную.

После приема знака "nak" передающий ПК посылает знак начала блока "soh" (start of header), два номера блока (сам номер и его двоичное дополнение по "единицам"), блок данных из 128 байт и контрольную сумму "xx" (блоки нумеруются по модулю 256). Контрольная сумма (1 байт) представляет собой остаток от деления на 255 суммы значений кодов знаков, входящих в блок данных.

В свою очередь принимающий ПК тоже вычисляет контрольную сумму и сравнивает ее с принятой. Если сравниваемые значения различны или если прошло 10 с и не

завершен прием блока, принимающий ПК посылает передатчику знак "nak", означающий запрос на повторную передачу последнего блока. В случае принятия правильного блока приемник передает "ack", а если следующий блок не поступил в течение 10 с, то передача знака "ack" повторяется до тех пор, пока блок не будет правильно принят. После девяти неудачных попыток передачи блока связь прерывается.

Для исключения повторной передачи одного и того же блока из-за потери подтверждающего сообщения в протоколе используется двукратная передача номера. Принимающий ПК контролирует неповторяемость принятого блока, и если блок ошибочно передан повторно, то он сбрасывается. После успешной передачи всех данных передающий ПК посылает знак завершения "eot" (end of transmission), сообщающий об окончании передачи файла.

Перерыв в передаче блока свыше 1 с считается разрывом связи.

Преимуществами данного протокола по сравнению с другими являются:

- его доступность для разработчиков программных средств;
- простота реализации на языках высокого уровня;
- малый объем приемного буфера (256 байт);
- возможность передачи не только символьных (коды ASCII), но и исполняеиых файлов (с расширением .COM и .EXE). Последнее возможно вследствии того, что конец файла определяется подсчетом переданных байтов и вместо знака файлового маркера (Ctrl-Z) используется специальный сигнал завершения. Эффективность обнаружения ошибок данным протоколом составляет 99,6% - выше, чем при обычной асинхронной проверке четности (95%).

К основным недостаткам этого протокола можно отнести:

- низкое быстродействие;
- большая вероятность необнаруженных ошибок;
- необходимость задания имени файла при приеме;
- относительно большой объем передаваемой служебной информации.

Последующие модифакации протокола XModem были направлены на устранение этих и некоторых других его недостанков.

Протокол XModem-CRC

В протоколе XModem-CRC передается два проверочных знака, образующих проверочную комбинацию CRC-16, вместо одного знака арифметической контрольной суммы, используемого в исходном протоколе XModem-CRC и в большинстве коммерческих приложений. Проверка CRC-16 гарантирует обнаружение всех одиночных и двойных ошибок, всех нечетных ошибок, всех пакетов ошибок длиной до 16 знаков; 99,9969% 17-битовых пакетов ошибок и 99,9984% более длинных пакетов ошибок.

В начале соединения вместо знаков "nak", как в протоколе XModem, приемник передает знаки "c". Если передатчик не поддерживает протокол XModem-CRC, он игнорирует эти знаки. Не получив ответа на передачу трех знаков "c", приемник переходит на работу по протоколу XModem и передает знаки "nak".

Протокол XModem-1k

Данный протокол представляет собой модификацию XModem-CRC с блоками длиной 1024 байт. При такой длине блоков значительно снижается влияние задержек в системах с разделением времени, модемах и сетях с коммутацией пакетов. Кроме того, по сравнению с обычным протоколом XModem уменьшается относительная доля заголовка в общем объеме передаваемой информации.

Для сообщения приемнику об увеличении длины передаваемого блока вместо знака "soh" (01) в его начале ставится знак "stx" (02). Номер блока, передаваемый во втором и третьем байтах, увеличивается на единицу независимо от его длины.

Передатчик не должен изменять длину блока между 128 и 1024 байт, если не был принят знак "ack" для текущего блока. Игнорирование этого ограничения может привести к необнаружению ошибок.

При использовании блоков по 1024 байт возможно увеличение длины передаваемого файла до значения, кратного 1024.

Блоки длиной 1024 байт могут применяться при групповой или одиночной передаче файлов. Для сохранения целостности данных, передаваемых по телефонному каналу, с этим вариантом следует использовать проверку CRC-16.

Протокол YModem

Протокол YModem - это протокол XModem-CRC, в котором реализована групповая передача файлов. Его появление было вызвано необходимостью устранения недостатков протокола XModem.

Все программы, реализующие протокол YModem, должны выполнять следующие функции:

- передачу имени и пути файла в блоке 0 в виде строки знаков кода ASCII, завершающейся знаком "нуль";
- их использование на приемном конце файла в качестве имени и пути принятого файла, если иная реализация не оговорена специально;
- применение проверки CRC-16 при приеме знаков "c", в противном случае - использование 8-битовой контрольной суммы;
- прием любой комбинации из 128- и 1024-байтовых блоков внутри каждого принимаего файла; возможность переключения длины блоков в конце файла (файлов) и/или в случае частых повторных передач;
- исключение изменения длины неподтвержденного блока на передающем конце канала;
- передачу в конце каждого файла знака "eot" до десяти раз, пока не будет принят знак "ack" (часть спецификации протокола XModem);
- обозначение конца сеанса связи нулевым (пустым) именем пути.

Связные программы, в которых не реализованы все перечисленные функции, с протоколом YModem не совместимы.

Выполнение этих минимальных требований не гарантирует надежной передачи файлов при помехах.

Расширения протокола XModem и протокол YModem устраняют некоторые недостатки протокола XModem, сохраняя в основном его простоту.

Как и в случае передачи одного файла, приемник инициирует групповую передачу путем посылки знака "c" (для режма CRC-16).

Передатчик открывает файл и передает номер 0 блока и последующую информацию.

Для групповой передачи требуются только имена файлов.

Для обеспечения совместимости "вверх" все неиспользуемые байты блока 0 должны иметь значение 0.

Имя файла (возможно с указанием пути) передается как строка кодов ASCII, завершаемая знаком "нуль". Этот формат имени файла используется в функциях, ориентированных на операционную систему MS-DOS, или в функциях fopen библиотеке языка Си. В имя файла не включаются пробелы. Обычно передается только само имя (без префекса справочника). Название привода источника (например, A:, B: и т.д.) не передается. Если в имя файла включен каталог, его название должно ограничиваться знаком "/".

Обозначение длины файла каждого последующего поля произвольно. Длина файла представлена в блоке как десятичная строка, обозначающая количество байтов в файле. В нее не должны входить знаки CPMEOF (^Z) или другие знаки (garbage characters), используемые для заполнения последнего блока.

Если передаваемый файл увеличивается во время передачи, параметр "длина файла" должен представлять свое максимально ожидаемое значение или не передаваться вовсе.

Дата модификации является необязательным параметром, имя файла и его длина могут передаваться без передачи даты модификации.

Протокол YModem допускает возможность введения других полей заголовка без нарушения совместимости со своими прежними программами. Оставшаяся часть блока устанавливается в 0. Это существенно для сохранения совместимости "вверх".

Если блок имени файла принят с ошибкой по коду, необходимо запросить повторную передачу. Прием блока с именем файла, успешно открытого для записи, подтверждается

знаком "ack". Если файл не может быть открыт для записи, приемник прерывает передачу с помощью знака "can".

Затем приемник инициирует передачу содержимого файлов в соответствии с протоколом XModem-CRC.

После того как содержимое файла передано, приемник запрашивает имя следующего файла.

Передача нулевого имени файла может означать, во-первых, что групповая передача завершена, и, во-вторых, что запрошенные у передатчика файлы не могут быть открыты для чтения.

По умолчанию приемник запрашивает CRC-16.

Протокол YModem поддерживается большинством связных программ общего пользования.

Протокол YModem-g

В настоящее время разработаны методы, обеспечивающие передачу данных с очень высокими скоростями и малой вероятностью ошибок. Эти методы реализованы в высокоскоростных модемах и ряде связных программ; при их применении достигается скорость передачи, близкая к предельной, но при существенном увеличении времени задержки. Эти задержки уменьшают пропускную способность большинства протоколов передачи данных, исправляющих ошибки.

Вариант g протокола YModem (YModem-g) обеспечивает высокую эффективность передачи в таких условиях. Протокол YModem-g используется приемником, который инициирует групповую передачу путем посылки знака "g" вместо "c". Передатчик, распознавший этот знак, прекращает ожидание обычных подтверждений по каждому переданному блоку и передает последовательные блоки на полной скорости с использованием метода управления потоком, например XON/XOFF.

Прежде чем передавать файл, передатчик ожидает первоначальных знаков "g", а в конце передачи каждого файла - подтверждающего знака "ack". Такой способ синхронизации

позволяет приемнику открывать и закрывать файлы в нужное время.

При обнаружение ошибки в случае использования протокола YModem-g приемник прерывает передачу, посылая последовательность, состоящую из многих знаков "can".

Расширение YModem-g протокола YModem позволяет значительно увеличить скорость передачи в каналах с защитой от ошибок (при использовании модемов со встроенным протоколом защиты от ошибок). Это достигается за счет отказа от переспроса принятых с ошибками блоков: при обнаружении ошибки передача файла просто прерывается. Для повышения быстродействия в последующих модификациях протокола XModem (например, в протоколе ZModem) был применен так называемый "оконный" алгоритм, при котором последующие блоки передаются подряд, без ожидания подтверждения правильного приема блока.

Протокол ZModem

Этот протокол, введенный в большинство связных программ, получил сейчас самое широкое применение. Представляя собой фактически развитие протоколов XModem и YModem, протокол ZModem устраняет их недостатки и, будучи совместимым с ними, имеет ряд преимуществ:

- высокое быстродействие благодаря использованию "оконного" алгоритма;
- динамическая адаптация к качеству канала связи посредством изменения в широких пределах размера блока;
- защита управляющей информации, доступа к передаче и защита от имитации управляющих сигналов;
- возможность возобновления прерванной передачи файла с того места, на котором произошло прерывание;
- повышенная достоверность передачи благодаря использованию 32-разрядной проверочной комбинации;
- возможность оптимального применения как в канале с высокой вероятностью ошибок, так и в каналах, работающих практически без ошибок (в которых уже реализован протокол, исправляющий ошибки).

Протокол ZModem разрабатывался для следующих областей применения:

- работа в сетях с большим временем задержки (по сравнению с временем передачи знака) и малой вероятностью ошибок;
- передача с помощью модемов с временным разделением и буферизацией, характеризующихся значительными задержками и быстрым снижением пропускной способности при росте обмена по обратному каналу;
- обеспечение прямой связи между двумя модемами при высокой вероятности ошибок в канале.

Протокол ZModem может быть использован либо самостоятельно, либо в сочетании с защитой от ошибок канального уровня, реализованной протоколами X.25, V.42, MNP, Fastlink и др. В случае сочетания с протоколами канального уровня протокол ZModem обеспечивает обнаружение и исправление ошибок в интерфейсах между средой, в которой исправляется ошибка, и остальной частью канала связи.

Можно сказать, что протокол ZModem - это результат технического компромисса между рядом противоречивых требований: простотой реализации и использования, обеспечением высокой пропускной способности, сохранением целостности информации и высокой надежностью передачи.

Протокол позволяет программно инициировать передачу файлов либо передавать команды и/или модификаторы другим программам. Имена файлов вводятся только один раз. При групповых передачах допускается общее задание файлов (например, "*.txt"). Общее требование: сведение к минимуму количества команд, вводимых с клавиатуры.

Вместе с файлами передается специальный служебный кадр, инициирующий автоматический прием. Если на противоположном конце не поддерживается протокол ZModem, то он может перейти в режим протокола YModem.

Протокол ZModem экономно использует пропускную способность обратного канала. Благодаря этому свойству модемы, которые динамически распределяют передачу в двух направлениях, осуществляют передачу с оптимальными скоростями; специальные свойства протокола ZModem обеспечивают простую и эффективную реализацию широкого спектра задач главных ЭВМ, работающих с разделением во времени.

Передача файлов осуществляется без начального таймаута независимо от того, какая из программ запущена первой, в отличие от протокола XModem, требующего задержки в 10 с.

С самого начала сеанса по протоколу ZModem вся передаваемая информация защищается циклическим кодом с проверочной комбинацией CRC из 16 или 32 битов. Кроме того, в протоколе реализован механизм защиты информации против сообщений типа "троянский конь", имитирующих разрешенные команды или передачу файлов.

Хотя протокол ZModem имеет широкий набор возможностей, он не предназначен для замены протоколов канального уровня - таких, как, например, протокол X.25.

Протокол адаптируется к задержкам сети и систем с временным разделением путем непрерывной передачи данных, продолжающейся до тех пор, пока приемник не прервет передатчик запросом на повторную передачу искаженных данных. Фактически протокол ZModem использует отдельный файл как "окно". Такое упрощение управления буфером исключает переполнение "окна", которому подвержены протоколы MEGAlink, Super Kermit и др.

ZModem разработан в качестве протокола передачи файлов общего назначения компанией Telenet (США). Его описание и исходная программа Unix rz / sz общедоступны. Лицензирование, торговые марки и ограничение копирования на применение этого протокола и исходной программы Unix rz /sz не распространяются.

Протокол Kermit

Протокол Kermit, реализованный практически во всех связных программах, предназначен в основном для передачи файлов между ЭВМ разных типов, включая большие и мини ЭВМ. Он оптимизирован для работы как при сильных помехах, так и при больших задержках сигнала. В отличие от протоколов XModem и YModem в нем используются пакеты переменной длины с максимальным значением 94 байт. Аналогично протоколу YModem протокол Kermit обеспечивает групповую передачу файлов.

Наряду со стандартным протоколом Kermit в ряде связных программ реализован более эффективный протокол Super Kermit, предусматривающий для уменьшения задержек при передаче

переменное "окно" передачи, в котором может содержаться от 1 до 32 пакетов. На приемном конце канала осуществляется обнаружение ошибок, но повторная передача не запрашивается до тех пор, пока не будут переданы все пакеты "окна". Кроме того, в протоколе реализован простой метод сжатия данных, также позволяющий сократить время передачи. Если на противоположном конце канала поддерживается протокол Kermit, то происходит автоматическое переключение на работу с ним.

5.9. Протоколы сжатия данных

Протокол V.42bis.

Этот протокол обеспечивает коэффициент сжатия 4:1, протокол V.42bis основан на алгоритме Лемпела-Зива-Уэлча (LZW-алгоритм).

Рассмотрим работу кодера LZW на примере трёхсимвольного алгоритма (а, б, в), а - код 1, б - код 2, в - код 3.

Символ	wK	w	Выход	Строка, добавляемая в словарь
а	а	а	-	
б	аб	б	код "а"=1	аб - код4
а	ба	а	код "б"=2	ба - код5
б	аб	аб	-	
в	абв	в	код "аб"=4	абв - код6
б	вб	б	код "в"=3	вб - код7
а	ба	ба	-	
б	баб	б	код "ба"=5	баб - код8
а	ба	ба	-	
б	баб	баб	-	
а	баба	а	код "баб"=8	баба - код9
а	аа	а	код "а"=1	аа - код10
а	аа	аа	-	
а	ааа	а	код "аа"=10	ааа - код11
а	аа	аа	-	
а	ааа	ааа	-	
а	аааа	а	код "ааа"=11	аааа - код12

Протокол V.44.

Коэффициент сжатия 6:1. Эффективен при работе с гипертекстом. В основе протокола лежит модификация алгоритма Лемпела-Зива.

Выводы

- Основной задачей протоколов канального уровня является доставка кадра узлу назначения в сети определенной технологии и достаточно простой топологии.
- Асинхронные протоколы разрабатывались для обмена данными между низкоскоростными старт-стопными устройствами: телетайпами, алфавитно-цифровыми терминалами и т. п. В этих протоколах для управления обменом данными используются не кадры, а отдельные символы из нижней части кодовых таблиц ASCII или EBCDIC. Пользовательские данные могут оформляться в кадры, но байты в таких кадрах всегда отделяются друг от друга стартовыми и стоповыми сигналами.
- Синхронные протоколы посылают кадры как для отправки пользовательских данных, так и для управления обменом.
- В зависимости от способа выделения начала и конца кадра синхронные протоколы делятся на символьно-ориентированные и бит-ориентированные. В первых для этой цели используются символы кодов ASCII или EBCDIC, а в последних - специальный набор бит, называемый флагом. Бит-ориентированные протоколы более рационально расходуют поле данных кадра, так как для исключения из него значения, совпадающего с флагом, добавляют к нему только один дополнительный бит, а символьно-ориентированные протоколы добавляют целый символ.
- В дейтаграммных протоколах отсутствует процедура предварительного установления соединения, и за счет этого срочные данные отправляются в сеть без задержек.
- Протоколы с установлением соединения могут обладать многими дополнительными свойствами, отсутствующими у дейтаграммных протоколов. Наиболее часто в них реализуется такое свойство, как способность восстанавливать искаженные и потерянные кадры.
- Для обнаружения искажений наиболее популярны методы, основанные на циклических избыточных кодах (CRC), которые выявляют многократные ошибки.
- Для восстановления кадров используется метод повторной передачи на основе квитанций. Этот метод работает по

алгоритму с простоями источника, а также по алгоритму скользящего окна.

- Для повышения полезной скорости передачи данных в сетях применяется динамическая компрессия данных на основе различных алгоритмов. Коэффициент сжатия зависит от типа данных и применяемого алгоритма и может колебаться в пределах от 1:2 до 1:8.