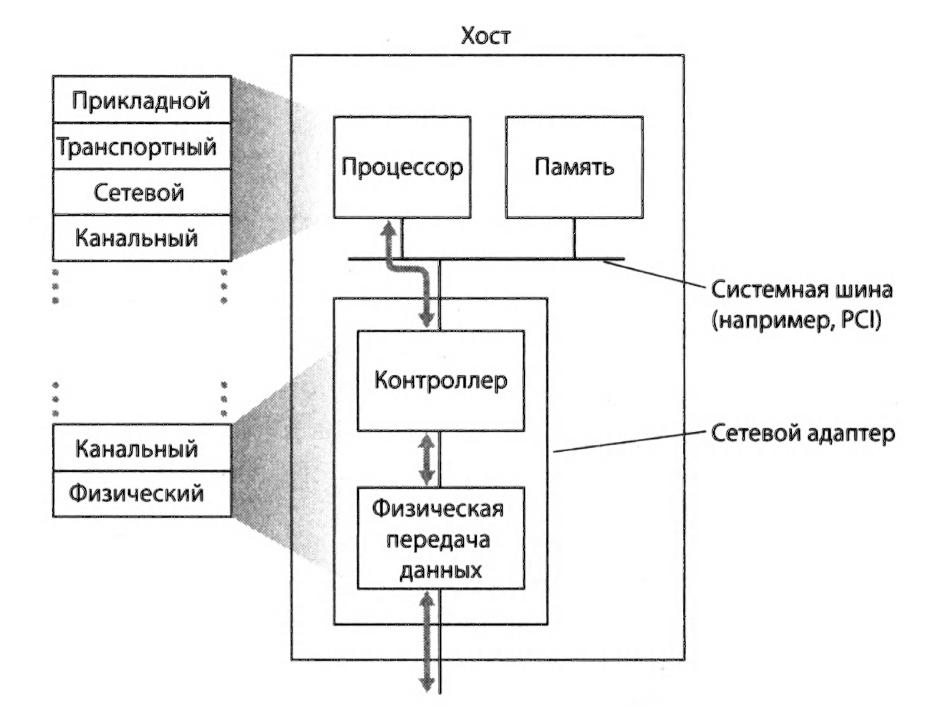
**Протоколы, реализующие канальный уровень**

Здесь мы сосредоточимся на конечной системе, поскольку ранее мы уже выяснили, что в маршрутизаторе канальный уровень реализуется интерфейсной платой. Каким же образом реализо­ван канальный уровень хоста — аппаратно или программно? Отвечает ли за него отдельная плата, как он состыкуется с остальным аппаратным обеспечением хоста и с компонентами операционной системы?

**Рис. 5.2. Сетевой адаптер и его отношение к другим компонентам хоста**

**и к функционалу стека протоколов**

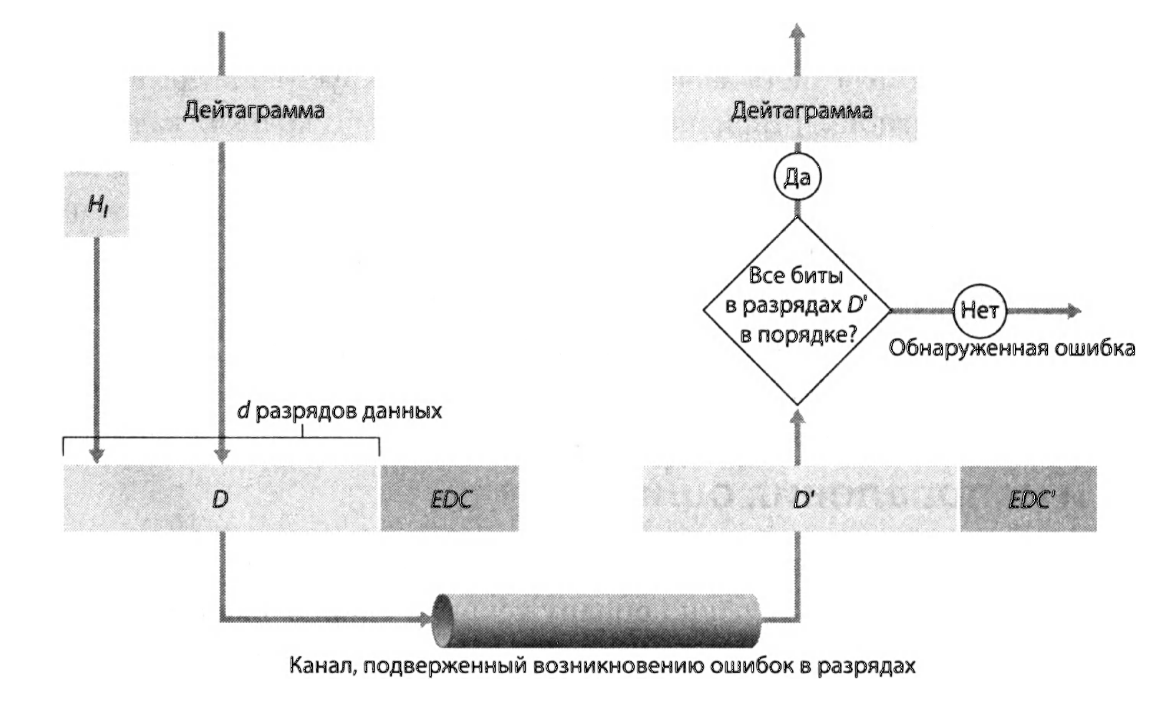
На рис. 5.2 представлена типичная архитектура хоста. Канальный уровень в основном реализован в **сетевом адаптере,** также называемом **сетевой интерфейсной платой** (network interface card, NIC). Централь­ным элементом сетевого адаптера является контроллер канального уров­ня, реализующий многие канальные службы (формирование кадров, до­ступ к каналу, обнаружение ошибок и т. д.). Соответственно, большая часть функционала канального контроллера реализуется аппаратно. Например, контроллер Intel 8254x реализует протоколы Ethernet; контроллер Atheros AR5006 ре­ализует протоколы Wi-Fi 802.11. До конца 1990-х большинство сетевых адаптеров представляли собой физически отдельные карты (например, карта PCMCIA или карта, подключаемая в PCI-разъем ПК). Но со временем сетевые адаптеры стали все чаще встраивать в материнскую плату хоста — такая конфигурация получила название LOМ (LAN-on-motherboard — сетевые решения на материн­ской плате).

Со стороны отправителя контроллер берет дейтаграмму, которая была создана и сохранена в памяти вышестоящими уровнями стека протоколов, инкапсулирует ее в кадр канального уровня (заполняя различные поля ка­дра), а затем передает кадр по линии связи согласно протоколу доступа к каналу. На стороне получателя контроллер принимает целый кадр и из­влекает из него дейтаграмму сетевого уровня. Если канальный уровень выполняет обнаружение ошибок, то именно контроллер на стороне отпра­вителя должен установить биты обнаружения ошибок в заголовке кадра, а обнаружение ошибок выполнит уже контроллер на стороне получателя.

На рис. 5.2 показан сетевой адаптер, подключенный к системной шине хоста (PCI или PCI-X). Он весьма похож на другие устройства ввода-вывода, связанные с другими компонентами хоста. На рис. 5.2 так­же видим, что, хотя большая часть канального уровня реализована аппаратно, в нем есть и программная часть, работающая на процессоре хоста. Программные компоненты канального уровня реализуют высокоуров­невый функционал — в частности, сборку адресной информации каналь­ного уровня и управление аппаратурой контроллера. На стороне полу­чателя программная часть канального уровня реагирует на прерывания контроллера (например, при получении одного или нескольких кадров), обработку ошибок или передачу дейтаграммы на сетевом уровне. Итак, на канальном уровне сочетаются программная и аппаратная часть, имен­но в этой части стека протоколов «железо» и код встречаются друг с дру­гом. На сайте [**www.intel.ru**](http://www.intel.ru)приведен обзор (а также дается подробное описание) контроллера 8254х с программной точки зрения.

**Приемы обнаружения и исправления ошибок**

В предыдущем разделе мы отмечали, что на канальном уровне ча­сто предоставляются услуги по **обнаружению и исправлению ошибок в отдельных разрядах кадра,** передаваемого между двумя физически соединенными узлами. Как было показано ранее, аналогичные услу­ги часто предоставляются также на транспортном уровне. В данном раз­деле мы рассмотрим некоторые простейшие методы обнаружения и ис­правления однобитовых ошибок. Данному вопросу посвящены целые книги, а здесь мы дадим лишь краткое введение в тему. Цель нашего обсуждения — сформировать представление о возможностях этих методов, а также по­казать, как работают наиболее простые из них и как они практически используются для обнаружения и исправления ошибок на канальном уровне.

На рис. 5.3 представлена схема передачи данных, иллюстрирующая нашу тему. На передающем узле к данным *D* добавляются разряды *EDC* (Error Detection and Correction — обнаружение и исправление оши­бок), позволяющие обнаруживать и исправлять ошибки. Как правило, таким образом защищается не только дейтаграмма, передаваемая сете­вым уровнем для транспортировки по линии связи, но также и адресная информация канального уровня, включая порядковые номера и другие поля заголовка кадра канального уровня. В кадре канального уровня принимающему узлу передаются как данные Д так и биты обнаруже­ния и исправления ошибок *EDC.* Принимающий узел получает разряды *D' и EDC'* соответственно. Обратите внимание, что значения *D' и EDC'* могут отличаться от исходных значений *D* и *EDC* вследствие ошибок при передаче.

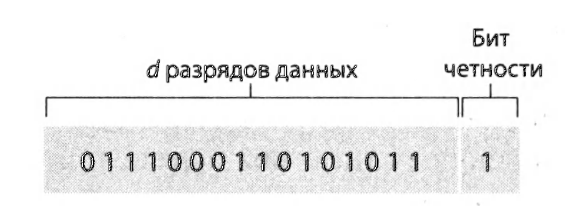
**Рис. 5.3. Сценарий обнаружения и исправления ошибок**

На основании принятых полей *D'* и *EDC'* получатель должен опре­делить, совпадают ли разряды *D'* с разрядами *D.* Обратите внимание, что вопрос заключается в том, обнаружена ли ошибка, а не в том, произошла ли она! Методы обнаружения и исправления ошибок иногда, *но не всегда,* позволяют получателю обнаруживать, что произошла ошиб­ка в одном или нескольких разрядах кадра. Другими словами, даже при добавлении к данным дополнительных бит с этой целью остается вероятность, что **ошибка не будет обнаружена,** а получатель не по­лучит информации об искажении полученных данных. В результате канальный уровень получателя может передать сетевому уровню по­врежденную дейтаграмму или не заметить повреждения какого-либо другого поля в заголовке кадра. Поэтому следует выбирать такую схе­му определения ошибок, при которой вероятность подобных событий мала. Как правило, чем изощреннее методы обнаружения и исправ­ления ошибок (снижающие вероятность появления необнаруженных ошибок), тем больше издержки — требуется больше операций для вы­числения контрольной суммы и больше времени для передачи допол­нительной информации.

Мы рассмотрим три метода обнаружения ошибок в переданных дан­ных: контроль четности (чтобы проиллюстрировать основные идеи, ле­жащие в основе методов обнаружения и исправления ошибок), метод контрольных сумм (больше характерный для транспортного уровня) и коды циклического контроля (как правило, используемые в адаптерах канального уровня).

**Контроль четности**

Возможно, простейшая форма обнаружения ошибок заключается в использовании одного **бита четности.** Предположим, что на рис. 5.3 передаваемые данные *D* имеют длину *d* разрядов. При проверке на чет­ность отправитель просто добавляет к данным один бит, таким образом, чтобы сумма всех единиц в d+1 бит (исходная информация плюс бит четности) оказалась четной. В схемах с проверкой нечетности значение бита выбирается таким образом, чтобы получилось нечетное количество единиц. На рис. 5.4 проиллюстрирован контроль четности, единствен­ный бит четности сохраняется в отдельном поле.

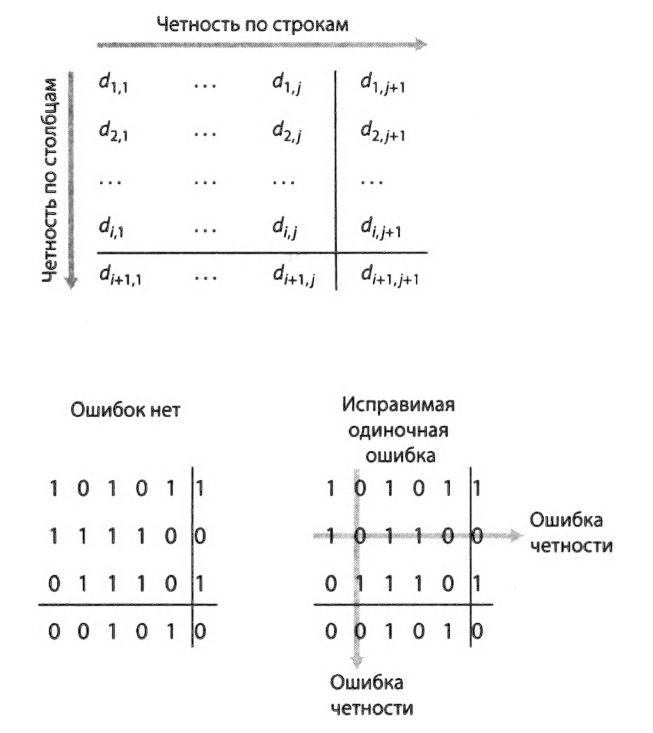
Действия, выполняемые получателем при использовании такой схе­мы, также очень просты. Он должен всего лишь сосчитать количество единиц в полученных d+1 разрядах. Если при проверке на четность по­лучатель обнаруживает, что в принятых им данных нечетное количество единичных разрядов, он понимает, что произошла ошибка по меньшей мере в одном разряде. В общем случае это означает, что в полученных данных инвертировано *нечетное* количество разрядов (произошла ошибка нечетной кратности).

**Рис. 5.4. Контроль четности с добавлением 1 бита.**

Что произойдет, если в прибывшем пакете данных произойдет четное количество однобитовых ошибок? В этом случае получатель не сможет их обнаружить. Если вероятность ошибки в одном разряде мала и можно предположить, что ошибки в отдельных разрядах возникают независи­мо друг от друга, тогда вероятность нескольких ошибок в одном пакете крайне низкая. В таком случае единственного бита четности может быть достаточно. Однако практические наблюдения показали, что в действи­тельности ошибки не являются независимыми, а часто группируются в пакеты ошибок. В случае пакетных ошибок вероятность того, что по­лучатель не обнаружит ошибку в пакете, может приблизиться к величи­не 50%. Очевидно, в такой ситуации требуется более надежная схема обнаружения ошибок (и она действительно применяется на практике). Но прежде чем перейти к изучению схем, применяемых на практике, рассмотрим простой пример, который обобщает предыдущую схему одноразрядного контроля четности и помогает понять принцип работы методов исправления ошибок.

На рис. 5.5 показано двумерное обобщение одноразрядной схе­мы проверки на четность. В данной схеме *d* разрядов пакета данных *D* разделяются на *i* строк и *j* столбцов, образуя прямоугольную матрицу Значение четности вычисляется для каждой строки и каждого столбца. Получающиеся в результате *i+j+1* битов четности образуют разряды об­наружения ошибок кадра канального уровня.

Предположим теперь, что в исходном блоке данных из *d* разрядов происходит однократная ошибка. В такой **двумерной схеме контроля четности** об ошибке будут одновременно сигнализировать контрольные разряды строки и столбца. Таким образом, получатель сможет не только *обнаружить* сам факт ошибки, но и по номерам строки и столбца найти поврежденный бит данных и *исправить* его. На рис. 5.5 показан пример, в котором поврежден бит в позиции (2, 2) — он изменил свое значение с 1 на 0. Такую одиночную ошибку получатель может не только обна­ружить, но и исправить. Хотя нас, в первую очередь, интересует обна­ружение и исправление ошибок в исходных *d* разрядах, данная схема позволяет также обнаруживать и исправлять одиночные ошибки в са­мих битах четности. Кроме того, данная двумерная схема контроля чет­ности позволяет обнаруживать (но не исправлять!) любые комбинации из двух одиночных ошибок (то есть двойные ошибки) в пакете.

**Рис. 5.5. Двумерный контроль четности**

Способность получателя обнаруживать и исправлять ошибки ино­гда называют **прямым исправлением ошибок** (Forward Error Correction, **FEC).** Подобные приемы широко применяются в устройствах хранения и воспроизведения звука, например на лазерных компакт-дисках. В се­тях методы обнаружения и исправления ошибок могут использоваться сами по себе, а также в сочетании с автоматическими запросами на по­вторную передачу, которые мы рассматривали ранее. Методы пря­мого обнаружения и исправления ошибок очень полезны, так как позволяют снизить необходимое количество повторных передач. Кроме того (что, возможно, даже важнее), эти методы позволяют получателю немедленно исправлять ошибки. Таким образом, получателю данных не нужно ждать, пока отправитель примет его сообщение об ошибке и вышлет пакет еще раз, что может быть существенным преимуществом в сетевых приложениях реального времени или при использовании каналов, для которых характерны длительные задержки распростране­ния (например, таковы каналы для передачи информации в космосе). Исследования методов прямого обнаружения и исправления ошибок, представлены в работах Бирсека, Нонненмахера, Байерса и Ша-чема.

**Методы контрольных сумм**

Методы, основанные на использовании контрольных сумм, об­рабатывают *d* разрядов данных (см. рис. 5.4) как последовательность *k*-разрядных целых чисел. Наиболее простой метод заключается в про­стом суммировании этих *k* -разрядных целых чисел и использовании полученной суммы в качестве битов определения ошибок. Так работает *алгоритм вычисления контрольной суммы, принятый в Интернете, —* байты данных группируются в 16-разрядные целые числа и суммиру­ются. Затем от суммы берется обратное значение (дополнение до 1), которое и помещается в заголовок сегмента. Как уже отмечалось ранее, получатель проверяет контрольную сумму, вычисляя дополне­ние до 1 от суммы полученных данных (включая контрольную сумму), и сравнивает результат с числом, все разряды которого равны 1. Если хотя бы один из разрядов результата равен 0, это означает, что произо­шла ошибка. Принятый в Интернете алгоритм вычисления контрольной суммы и его реализация подробно описываются в RFC 1071. В протоко­лах TCP и UDP контрольная сумма вычисляется по всем полям (вклю­чая поля заголовка и данных). В других протоколах, например ХТР, вычисляются две контрольные суммы, одна для заголовка, а другая для всего пакета.

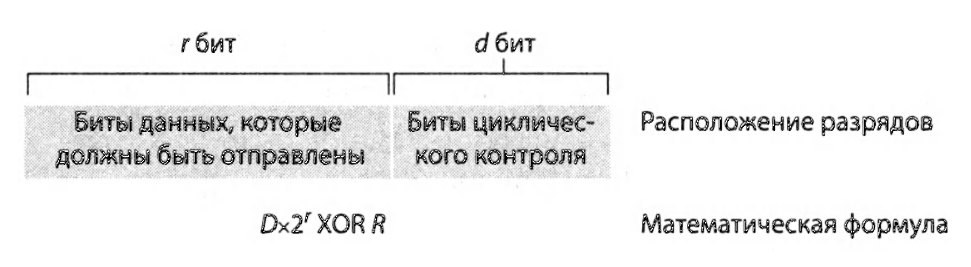
Метод вычисления контрольной суммы для пакетов требует от­носительно небольших накладных расходов. Например, в протоколах TCP и UDP для контрольной суммы используются всего 16 бит. Од­нако подобные методы предоставляют относительно слабую защи­ту от ошибок по сравнению с обсуждаемым далее методом контроля с помощью кода циклического контроля, который часто используется на канальном уровне. Разумеется, возникает вопрос: почему на транспортном уровне применяют контрольные суммы, а на канальном уровне — код циклического контроля? Вспомним, что транспортный уровень, как правило, реализуется на хосте программно как часть опе­рационной системы хоста. Поскольку обнаружение ошибок на транс­портном уровне реализовано программно, важно, чтобы схема обна­ружения ошибок была простой. В то же время обнаружение ошибок на канальном уровне реализуется аппаратно в адаптерах, способных быстро выполнять более сложные операции по вычислению кода ци­клического контроля.

Филдмайер анализирует быстрые методы программной реализа­ции для вычисления не только взвешенных кодов контрольных сумм, но и циклических (см. ниже), а также других кодов.

**Код циклического контроля**

Широко применяемый в современных компьютерных сетях метод обнаружения ошибок основан на использовании кода. **Коды цикличе­ского контроля** (Cyclic Redundancy Check, **CRC)** также называют **по­линомиальными кодами,** так как при их вычислении битовая строка рассматривается как многочлен (полином), коэффициенты которого равны 0 или 1, и операции с этой битовой строкой можно интерпретиро­вать как операции деления и умножения многочленов.

Циклические коды работают следующим образом. Рассмотрим фрагмент данных *D,* которые передающий узел хочет отправить прини­мающему узлу. Отправитель и получатель должны договориться о по­следовательности из *r*+1 бит, называемой **образующим многочленом** (или **генератором),** который мы будем обозначать *G.* Старший (самый левый) бит генератора *G* должен быть равен 1. Ключевую идею кода ци­клического контроля иллюстрирует рис. 5.6. Для заданного фрагмента данных *D* отправитель формирует *r* дополнительных разрядов *R*, кото­рые он добавляет к данным *D* так, что получающееся в результате число, состоящее из *d+r* бит, делится по модулю 2 на генератор (число) *G* без остатка. Таким образом, процесс проверки данных на наличие ошибки относительно прост. Получатель делит полученные *d+r* бит на генера­тор *G.* Если остаток от деления не равен нулю, это означает, что произо­шла ошибка. В противном случае данные считаются верными и прини­маются.

**Рис.** 5.6. **Код циклического контроля**

Все операции по вычислению CRC-кода производятся в арифме­тике по модулю 2 без переносов в соседние разряды. Это означает, что операции сложения и вычитания идентичны друг другу и эквивалентны поразрядной операции исключающего ИЛИ (exclusive OR, XOR). На­пример:

1011 XOR 0101 = 1110 1001 XOR 1101 = 0100

Кроме того, мы получим:

1011 - 0101 = 1110 1001 - 1101 = 0100

Операции умножения и деления аналогичны соответствующим опе­рациям в обычной двоичной арифметике с той разницей, что любая тре­бующаяся при этом операция сложения или вычитания выполняется без переносов в соседний разряд. Как и в обычной арифметике, умноже­ние на 2k означает сдвиг числа на *k* разрядов влево. Таким образом, при заданных значениях *D* и *R* величина D x 2r XOR *R* соответствует после­довательности из *d+r* бит, показанной на рис. 5.6. В нашем дальнейшем обсуждении мы будем использовать именно эту алгебраическую форму для обозначения последовательности из d+r бит.

Вернемся теперь к главному вопросу: как отправитель вычисляет *R?* Вспомним, что нам требуется найти такое значение *R*, чтобы для неко­торого *п* выполнялось следующее равенство:

***Dx2r* XOR *R = nG***

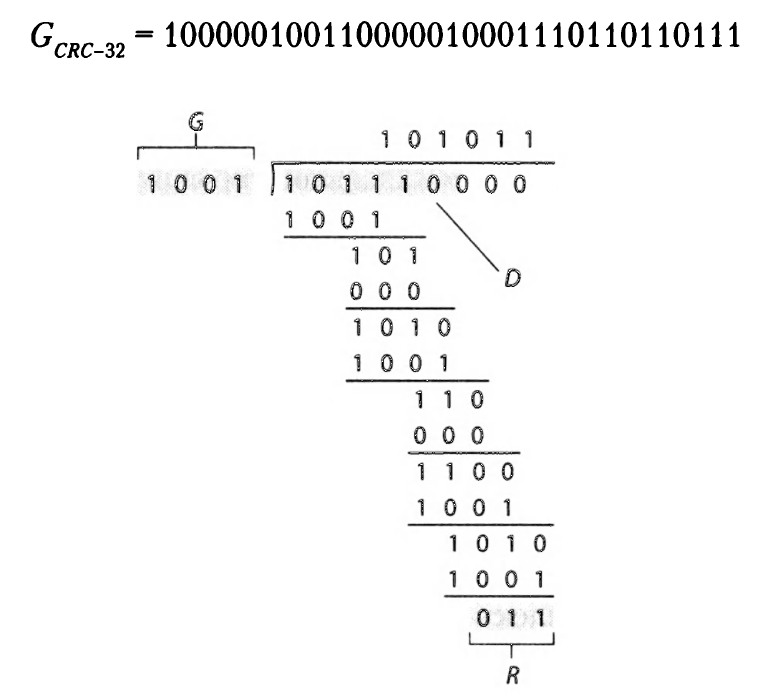
То есть требуется выбрать такое значение *R*, чтобы *Dx2r* XOR *R* де­лилось на *G* без остатка. Если прибавить к каждой части уравнения зна­чение *R* по модулю 2, то мы получим

***Dx2r* = *nG* XOR *R***

Отсюда следует, что если мы разделим Dx2r на G, то значение остат­ка будет равно *R.* Таким образом, мы можем вычислить *R* как

На рис. 5.7 показан пример вычисления R в случае D=101110, d = 6> С=1001и r=3. В этом случае отправитель передает следующие 9 бит: 101110011. Вы можете сами проверить правильность расчетов, а также убедиться, что Dx2r = 101011 х G XOR R*.*

Для 8-, 12-, 16- и 32-разрядных генераторов *G* определены между­народные стандарты. В стандарте CRC-32, принятом в ряде IEEE-протоколов канального уровня, используется генератор вида

****

**Рис. 5.7. Пример вычисления CRC**

Каждый стандарт CRC-кода способен обнаруживать ошибочные па­кеты длиной не более *r* разрядов. Кроме того, при соответствующих до­пущениях ошибочный пакет длиной более чем *r* разрядов будет обнару­жен с вероятностью 1 — 0,5r. Помимо этого каждый стандарт CRC-кода может обнаруживать ошибки нечетной кратности. В работе Вильямсарассматриваются вопросы осуществления проверок при помощи кода циклического контроля

**Протоколы и каналы множественного доступа**

Ранее мы отметили, что существуют два типа сетевых ка­налов: двухточечные и широковещательные. **Двухточечная линия связи** соединяет отправителя на одном конце линии и получателя на другом. Для двухточечных линий связи разработано множество протоколов ка­нального уровня. Ниже будет рассказано о двух таких прото­колах: РРР (Point-to-Point Protocol — протокол передачи от точки к точ­ке) и HDLC (High-level Data Link Control — высокоуровневый протокол управления каналом). Второй тип канала, **широковещательный канал,** может иметь несколько передающих и принимающих узлов, присоеди­ненных к одному и тому же совместно используемому широковещатель­ному каналу. Термин «широковещание» используется здесь потому, что, когда один из узлов передает кадр, этот кадр принимается всеми осталь­ными узлами, присоединенными к каналу. Примерами использования широковещательной технологии канального уровня являются Ethernet-сети и беспроводные локальные сети. В данном случае мы сделаем не­большое отступление от темы протоколов канального уровня и сначала рассмотрим крайне важную проблему, заключающуюся в координации доступа множества передающих и принимающих узлов к общему широко­вещательному каналу, — так называемую **проблему множественного до­ступа.** Широковещательные каналы часто применяют в локальных сетях, то есть в сетях, географически сконцентрированных в одном здании (или комплексе зданий, принадлежащих одной организации, например универ­ситету или компании).

Нам всем известно понятие широковещательной рассылки, так как эта технология передачи данных используется в телевидении с момента его изобретения. Но в традиционном телевидении широковещательная рассылка является односторонней, так как там один фиксированный узел передает информацию множеству получающих узлов. Между тем узлы компьютерной широковещательной сети могут как принимать данные, так и передавать их. Возможно, более близкой к компьютерной широковещательной сети аналогией является вечеринка с коктейлями, где множество людей собираются в большой комнате (средой широ­ковещательного канала при этом является воздух), чтобы поговорить и послушать. Вторая аналогия, хорошо знакомая многим читателям, — аудитория, в которой преподаватель и студенты совместно используют одну общую широковещательную среду передачи. Центральная проблема обоих сценариев заключается в том, чтобы решить, кто и когда полу­чает право говорить (передавать по каналу). Для себя люди разработали сложный набор правил коллективного использования канала:

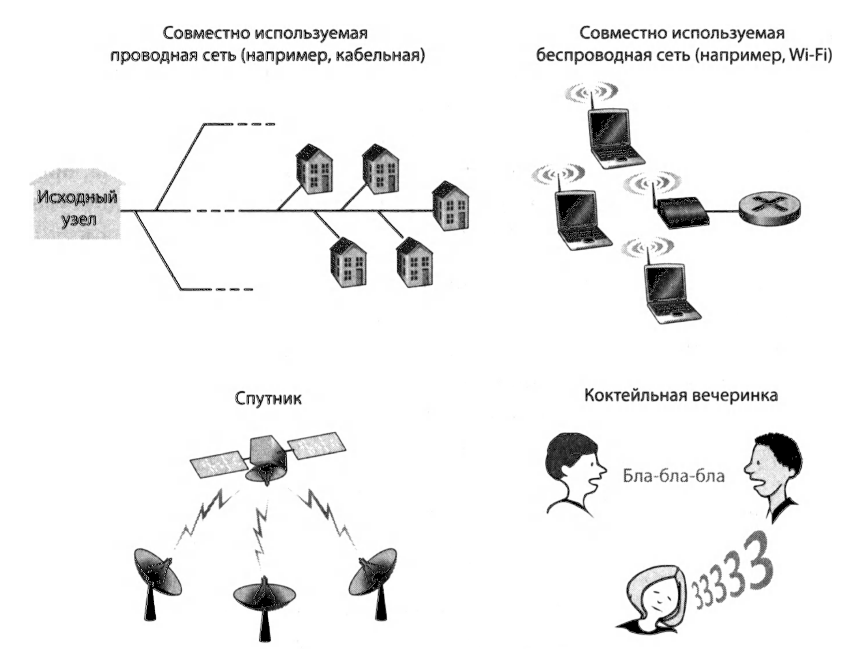
«дайте возможность поговорить каждому»;

«не говорите, пока с вами не заговорят»;

«не монополизируйте беседу»;

«если у вас есть вопрос, поднимите руку»;

«не прерывайте говорящего громким храпом».

Обмен данными в компьютерных сетях также управляется набора­ми правил, составляющих так называемые **протоколы множественного доступа.** Как показано на рис. 5.8, протоколы множественного доступа применяются в сетях самых разных конфигураций, включая кабельные и беспроводные локальные сети, а также спутниковые сети. Хотя техни­чески каждый узел получает доступ к широковещательному каналу че­рез адаптер, в данном разделе и передающее, и принимающее устройства мы будем называть *узлом.* На практике по одному широковещательному каналу могут обмениваться данными сотни или даже тысячи узлов.

**Рис.** 5.8. Различные каналы множественного доступа

Поскольку передавать кадры могут все узлы, возможна ситуация, когда одновременно начнут передачу несколько узлов. Когда такое про­исходит, каждый из узлов одновременно получает несколько кадров, то есть на принимающих узлах имеет место **коллизия** переданных кадров. Как правило, в случае коллизии принимающие узлы не могут коррект­но обрабатывать принятые кадры, так как сигналы таких кадров накла­дываются друг на друга. Таким образом, все вовлеченные в коллизию кадры теряются, а все время, пока они передавались, оказывается по­траченным впустую. Очевидно, что наличие множества узлов, требую­щих частой передачи данных, означает высокую вероятность коллизий и низкий коэффициент использования канала.

Чтобы гарантировать высокую производительность канала при боль­шом количестве активных узлов, необходимо каким-то образом коорди­нировать передачу ими кадров. За эту координацию отвечает протокол коллективного доступа. За последние 40 лет по теме протоколов кол­лективного доступа были написаны тысячи статей и защищены сотни докторских диссертаций. Всесторонний обзор этой работы можно найти в книге Рома. Более того, активные исследования протоколов коллек­тивного доступа продолжаются до сих пор, что вызвано появлением но­вых типов линий связи, например новых беспроводных каналов.

За годы работы с помощью широкого спектра технологий канально­го уровня были реализованы десятки протоколов множественного до­ступа. Тем не менее практически любой из этих протоколов мы можем отнести к одной из трех категорий: **протоколы разделения канала, про­токолы произвольного доступа** и **протоколы поочередного доступа.** Мы обсудим эти категории протоколов коллективного доступа в сле­дующих трех подразделах.

В заключение этого обзора отметим, что в идеальном случае прото­кол множественного доступа для широковещательного канала со скоро­стью передачи данных *R* бит/с должен обладать следующими характе­ристиками.

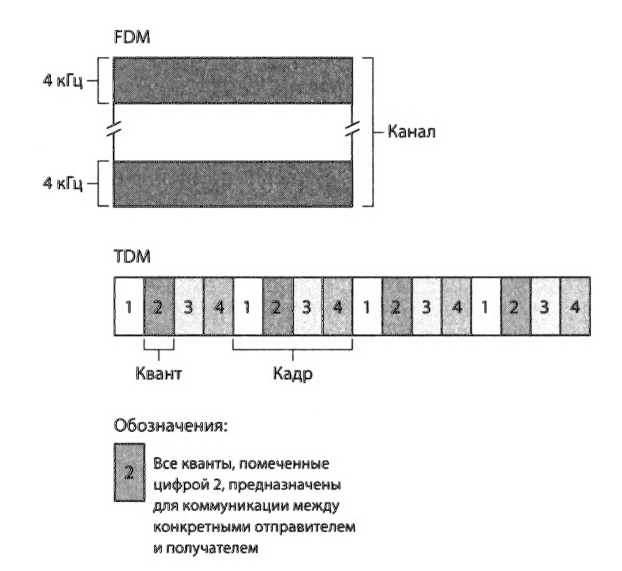
1. Когда данные для передачи есть только у одного узла, этот узел об­ладает пропускной способностью в *R* бит/с.
2. Когда данные для передачи есть у *М* узлов, каждый из этих узлов обладает пропускной способностью в *R/M* бит/с. Это не означает, что каждый из *М* узлов в каждый момент времени может передавать данные со скоростью *R/M* бит/с, — это средняя скорость передачи данных каждого из узлов.
3. Протокол является децентрализованным, то есть не существует управляющих узлов, выход из строя которых может остановить ра­боту всей сети.
4. Протокол прост и дешев в реализации.

**Протоколы разделения канала**

Раньше шла речь о двух методах разделения про­пускной способности широковещательного канала между всеми узла­ми, использующими этот канал. Мы говорили о мультиплексирова­нии с временным разделением (Time-Division Multiplexing, TDM) и мультиплексировании с частотным разделением (Frequent-Division Multiplexing, FDM). Предположим для примера, что канал поддержи­вает *N* узлов и скорость передачи данных в канале равна *R* бит/с. При временном разделении канала время делится на интервалы, называемые **кадрами,** каждый из которых делится на *N* элементарных интервалов времени, называемых **квантами.** (Не путайте кадр времени в контексте TDM с *кадром* как минимальной единицей передачи данных на каналь­ном уровне между исходным и конечным адаптерами. Чтобы не допу­скать излишней путаницы, в этом разделе мы будем называть кадр об­мена данными на канальном уровне словом «пакет».) Затем каждому из N узлов назначается один временной квант. Когда у узла есть кадр для отправки, он передает биты этого кадра в течение назначенного ему эле­ментарного интервала времени. Как правило, длина кванта выбирается таким образом, чтобы за этот интервал можно было передать один кадр. На рис. 5.9 показан простой пример мультиплексирования с временным разделением для канала с четырьмя узлами. Если вернуться к нашей аналогии с вечеринкой, то при такой схеме каждый участник вечеринки сможет говорить в течение некоторого фиксированного интервала вре­мени, после чего такое же право получает другой участник, и т. д. После того как возможность поговорить получат все участники вечеринки, она снова переходит к первому участнику, и все повторяется.

Привлекательность временного разделения канала заключается в том, что такая схема полностью устраняет конфликты (коллизии) и обладает идеальной справедливостью: каждый узел получает выде­ленную скорость передачи данных, равную *R/N* бит/с в течение каждого временного кадра. Однако у такой схемы есть два существенных недо­статка. Во-первых, каждый узел ограничен средней скоростью передачи данных в *R/N* бит/с даже в том случае, когда этот узел единственный, кому нужно отсылать данные в этот момент. Во-вторых, при передаче узел всегда должен ждать своей очереди, даже когда кроме него никто не отправляет данные. Сложно представить себе вечеринку, где лишь одно­му гостю есть что сказать (и тем более маловероятно, что все остальные гости готовы его слушать). Таким образом, очевидно, что временное раз­деление канала плохо подходит для протокола коллективного доступа.

Метод мультиплексирования с частотным разделением делит канал с пропускной способностью *R* бит/с на частотные диапазоны с полосой пропускания *R/N* бит/с, при этом каждому узлу выделяется собствен­ный частотный диапазон. Таким образом, при методе частотного разде­ления из одного канала с пропускной способностью *R* бит/с создается *N* каналов с пропускной способностью *R/N* бит/с. Мультиплексирова­ние с частотным разделением канала обладает теми же преимущества­ми и недостатками, что и с временным. Устраняются коллизии и обеспе­чивается справедливое распределение пропускной способности между узлами, но пропускная способность каждого узла ограничена значением *R/N* бит/с, независимо от текущей загруженности канала.

****

**Рис. 5.9. Примеры мультиплексирования с временным и частотным разделением**

**для системы с четырьмя узлами**

Еще один метод совместного использования общего канала, который мы рассмотрим, предлагает протокол **множественного доступа с кодовым разделением** (Code Division Multiple Access, **CDMA).** В отличие от схем мультиплексирования с частотным и временным разделением канала, предоставляющих узлам частотные диапазоны или временные интервалы, протокол CDMA назначает каждому узлу собственный *код.* Затем каждый узел использует этот уникальный код для кодирования передаваемых данных. Как мы увидим, протокол CDMA позволяет не­скольким узлам передавать данные *одновременно*, при этом получатели могут корректно их принимать (при условии, что получателю изве­стен код передатчика), даже при воздействии помех со стороны других узлов. Протокол CDMA в течение некоторого времени использовался в военных системах связи (благодаря своей устойчивости к попыткам подавления сигнала), а в настоящее время получает все более широкое распространение в гражданских беспроводных средствах связи коллек­тивного доступа. Поскольку использование CDMA так тесно связано с темой беспроводных каналов, мы отложим его подробное рассмотре­ние. Пока будет достаточно сказать, что коды CDMA, подоб­но квантам времени в TDM и частотам в FDM, могут выделяться сразу многим пользователям канала коллективного доступа.

**Протоколы произвольного доступа**

Второй широкий класс протоколов коллективного доступа состав­ляют так называемые протоколы произвольного доступа. В протоколе произвольного доступа передающий узел всегда передает данные в ка­нал с максимальной скоростью, то есть *R* бит/с. Когда возникает кол­лизия, каждый вовлеченный в нее узел отправляет свой кадр (то есть пакет) повторно до тех пор, пока ему не удастся осуществить передачу без коллизий. Однако, претерпев коллизию, узел, как правило, не по­вторяет **передачу сразу же, а *выжидает в течение случайного интер­вала*** *времени.* Благодаря разной длительности случайных интервалов времени существует ненулевая вероятность того, что интервал, вы­бранный одним из узлов, окажется меньше, чем у других вовлеченных в коллизию узлов, и он успеет вытолкнуть свой кадр в канал беспре­пятственно.

В литературе описаны десятки, если не сотни, протоколов произвольного доступа.

**Дискретный протокол ALOHA**

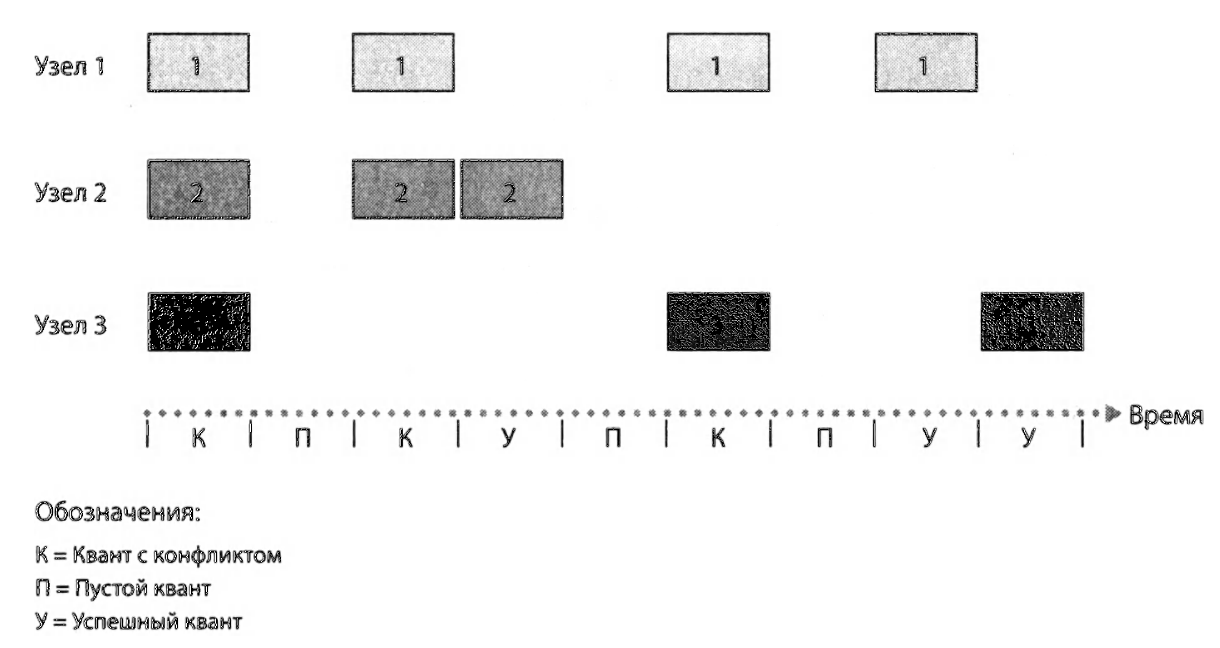
Начнем наше изучение протоколов произвольного доступа с одного из наиболее простых, так называемого дискретного протокола ALOHA. В нашем описании мы будем предполагать следующее:

* все кадры состоят ровно из *L* бит;
* время разделено на интервалы времени (кванты) длительностью *L/R* с (это время, за которое передается один кадр);
* узлы начинают передачу кадров только в момент начала очередного кванта;
* узлы синхронизируются так, что каждый из них знает, когда начина­ется квант;
* если в течение данного временного кванта сталкиваются несколько кадров, тогда все узлы обнаруживают факт коллизии, прежде чем за­кончится данный квант.

Обозначим через *р* вероятность — то есть *р* будет иметь значение 0 или 1. Работа дискретного протокола ALOHA на каждом узле проста:

* Когда у узла появляется новый кадр для отправки, он дожидается начала следующего кванта времени и за этот квант передает весь кадр.
* Если коллизия не возникает, это значит, что узел успешно передал свой кадр и повторная передача не требуется. (Если у узла есть но­вый кадр, то он может подготовить его к отправке.)
* Если коллизия возникает, то узел замечает ее до окончания кванта времени. Узел будет пытаться передать свой кадр в течение после­дующих квантов времени, что рано или поздно удастся ему с вероят­ностью ***р.***

Под повторной передачей кадра с вероятностью *р* мы подразумеваем, что узел как бы подбрасывает монету При этом кадр передается повтор­но, только если выпадает решка, что происходит с вероятностью *р.* При выпадении орла, что происходит с вероятностью (1 — *р),* узел пропускает данный временной интервал и подбрасывает монетку еще раз. Все узлы, вовлеченные в коллизию, подбрасывают свои монетки независимо друг от друга. Казалось бы, дискретный протокол ALOHA обладает рядом до­стоинств. В отличие от протоколов мультиплексирования с разделением канала, он позволяет единственному активному в сети узлу передавать кадры без перерыва с максимальной скоростью *R* (узел называется актив­ным, если у него есть кадр для передачи). Дискретный протокол ALOHA является в большой степени децентрализованным, так как каждый узел сам обнаруживает факт коллизии и независимо от других узлов прини­мает решение о времени повторной передачи. (Однако для использова­ния дискретного протокола ALOHA требуется синхронизация узлов. Да­лее мы кратко обсудим непрерывную версию протокола АЮН А, а также протоколы CSMA, не требующие подобной синхронизации.) Кроме того, ALOHA — чрезвычайно простой протокол.

Рис. 5.10. **Между узлами 1, 2 и 3 возникает коллизия в первый квант времени. Узел 2 наконец срабатывает в четвертом кванте времени, узел 1 — в восьмом,**

**а узел 3 — в девятом**

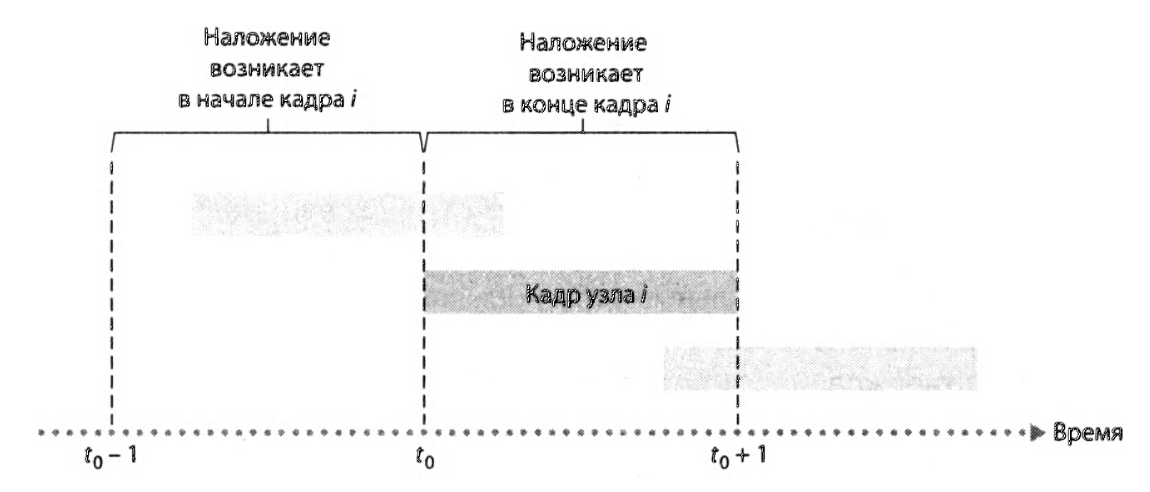
Дискретный протокол ALOHA хорошо работает в тех ситуациях, когда имеется только один активный узел, но какова его эффективность при наличии нескольких активных узлов? Ее снижают два фактора. Во-первых, как показано на рис. 5.10, когда в сети несколько активных узлов, определенная доля квантов тратится впустую из-за коллизий. Во-вторых, другая доля квантов тратится напрасно, когда все активные узлы одновременно отказываются от передачи в силу ее вероятностного механизма. Дискретный протокол ALOHA работает продуктивно толь­ко в те кванты, когда передавать требуется ровно одному узлу. Квант, на протяжении которого передает только один узел, называется **успеш­ным квантом. Эффективность** дискретного протокола коллективного доступа определяется долей успешных квантов при наличии большого количества активных узлов, у каждого из которых всегда есть большое количество кадров для передачи. Обратите внимание, что, если вообще не использовать протоколы коллективного доступа и после коллизии немедленно повторять передачу каждым из узлов, эффективность сети была бы равна нулю. Дискретный протокол ALOHA очевидно увеличи­вает эффективность сети, но насколько?

Попытаемся определить максимальную эффективность дискретно­го протокола ALOHA. Чтобы упростить наши вычисления, немного из­меним протокол, предположив, что каждый узел с вероятностью *р* пы­тается передать кадр с наступлением каждого нового кванта. То есть мы предполагаем, что у каждого узла всегда есть кадр для передачи и узел всегда с вероятностью *р* пытается передать его независимо от того, но­вый это кадр или передаваемый повторно. Пусть в сети будет *N* узлов. В этом случае квант оказывается успешным, если один из узлов пере­дает, a *N-1* узлов воздерживаются от передачи. Вероятность того, что некий конкретный узел будет передавать, равна *p*. Вероятность того, что остальные *N-1* узлов не будут передавать, равна *(1-p)N-1.* Таким обра­зом, вероятность того, что данному узлу удастся успешно передать кадр, равна *p(1-p)N-1.* Поскольку существует *N* узлов, вероятность того, что повезет одному (любому) из них, равна *Np(1-p)N-1.*

Таким образом, при наличии *N* активных узлов эффективность дис­кретного протокола ALOHA равна *Np(1-p)N-1.* Чтобы определить *мак­симальную* эффективность протокола при *N* активных узлах, нам нужно найти такое значение вероятности р\*, при котором данное выражение достигает максимума. А чтобы получить максимальную эффективность протокола для большого количества активных узлов, мы можем найти предел значения *Np\*(1-p\*)N-1* для значения *N,* стремящегося к беско­нечности. Выполнив все эти вычисления, мы обнаружим, что максимальная эффективность прото­кола составляет *1/е =* 0,37. Таким образом, когда у большого количества узлов имеется много кадров для передачи, то (в лучшем случае) толь­ко около 37% квантов канал будет тратить на полезную работу. То есть эффективная пропускная способность канала равна не *R* бит/с, а всего лишь 0,37 *R* бит/с! Оказывается, что около 37% времени канал будет простаивать и еще около 26% времени тратить на обработку коллизий. Представьте себе несчастного администратора вычислительной сети, приобретшего систему с дискретным ALOHA с пропускной способно­стью 100 Мбит/с и собирающегося использовать ее для обслуживания трафика между большим количеством пользователей с суммарной пропускной способностью около 80 Мбит/с! Несмотря на то, что мгновен­ная пропускная способность канала равна 100 Мбит/с, его эффективная пропускная способность составит менее 37 Мбит/с.

**«Чистый» протокол ALOHA**

Дискретный протокол ALOHA требует, чтобы все узлы синхрони­зировали время начала передачи кадров. Собственно, первый протокол ALOHА не был дискретным и *являлся* полностью децентрализованным. В так называемом чистом протоколе ALOHA, когда прибывает первый кадр (то есть дейтаграмма сетевого уровня передается на более низкий уровень передающего узла), узел немедленно передает весь кадр цели­ком в широковещательный канал. Если переданный кадр сталкивается с одним или несколькими другими кадрами (происходит коллизия), с вероятностью *р* узел немедленно передает кадр повторно. В против­ном случае узел выжидает в течение времени, необходимого для переда­чи одного кадра, после чего опять с вероятностью *р* передает кадр либо пережидает еще один интервал времени с вероятностью *1-р.*

****

**Рис. 5.11. Наложение передачи данных в чистом протоколе ALOHA**

Чтобы определить максимальную эффективность чистого протокола АЮНА, обсудим отдельный узел. Мы будем использовать те же допуще­ния, что и в случае дискретного протокола АЮНА, и примем за едини­цу времени интервал (квант), требующийся для передачи одного кадра. В любой момент времени вероятность того, что узел передает кадр, равна *р.* Предположим, передача этого кадра началась в момент времени *t0.* Как видно из рис. 5.11, чтобы этот кадр был передан успешно, никакой другой узел не должен начать свою передачу во временном интервале [£0-1,£0], так как иначе она совпадет по времени с началом отправки нашего узла *i*. Вероятность того, что остальные узлы не начнут передачу в течение этого интервала времени, равна *(1-p)N-1.* Аналогично никакой другой узел не должен начать свою передачу, пока передает наш узел г, так как такая пе­редача также приведет к коллизии, но уже с концом нашего кадра *L* Веро­ятность этого события также равна *(1-p)N-1*. Таким образом, вероятность успешной передачи кадра данным узлом становится равной *p(1-p)2(N-1)* При стремлении количества узлов к бесконечности максимальная эф­фективность чистого протокола АЮНА будет равна всего лишь 1/(2е), то есть половине от максимальной эффективности дискретного протоко­ла ALOHA. Такова плата за полную децентрализацию.

**ИСТОРИЯ**

**Норм Абрамсон и ALOHAnet**

Доктор философии Норм Абрамсон любил серфинг и интересовался коммутацией пакетов. Это сочетание увлечений привело его в 1969 го­ду в Гавайский университет. Гавайи представляют собой множество гористых островков, на которых трудно установить традиционную ло­кальную сеть. В свободное от серфинга время Абрамсон размышлял о том, как разработать сеть с коммутацией пакетов, передаваемых по радио. У спроектированной им сети был один центральный хост и не­сколько второстепенных узлов, разбросанных по Гавайским островам. У сети было два канала, для каждого из которых использовался свой частотный диапазон. По нисходящему широковещательному каналу пакеты рассылались от центрального хоста остальным узлам. По вос­ходящему каналу остальные узлы посылали пакеты центральному хо­сту. Помимо информационных пакетов центральный хост также посы­лал подтверждения для каждого успешно принятого пакета.

Поскольку второстепенные узлы передавали пакеты децентрализо­ванно, в восходящем канале неизбежно возникали коллизии. Это на­блюдение натолкнуло Абрамсона на идею протокола ALOHA (чистого), описанного ранее. В 1970 году Абрамсон при финансовой под­держке управления ARPA соединил сеть ALOHA net с сетью ARPA net. Эта работа не только привела к рождению первой беспроводной сети с коммутацией пакетов, но еще и вдохновила Боба Меткалфа на созда­ние на основе ALOHA протокола CSMA/CD и локальной Ethernet-сети.

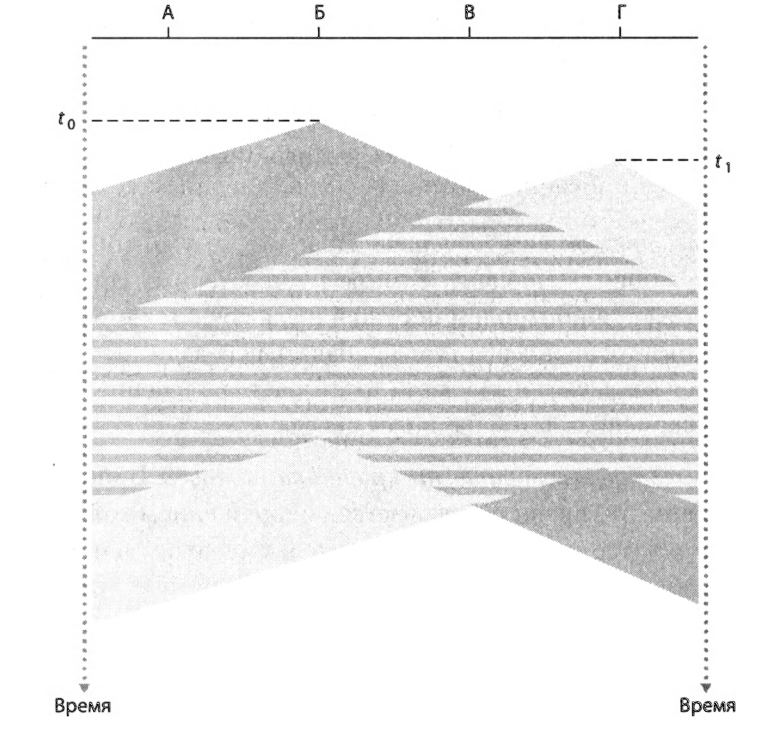
**Протокол множественного доступа с контролем несущей (CSMA)**

В обоих вариантах протокола ALOHA, дискретном и чистом, узел принимает решение о передаче кадра независимо от активности остальных узлов, присоединенных к широковещательному каналу. В част­ности, узел не обращает внимания на то, ведется ли в данный момент передача другими узлами, и не прекращает ее в случае коллизий. Если вспомнить нашу аналогию с вечеринкой, протоколы ALOHA подобны невоспитанным людям, прерывающим чужую беседу и продолжающим говорить, несмотря на то, что в нее вступили другие участники вечерин­ки. У людей также есть свои протоколы, позволяющие им не только ве­сти себя более цивилизованно, но и тратить меньше времени на «колли­зии» друг с другом и, таким образом, повышать «производительность» беседы. В частности, существуют два важных правила вежливого раз­говора.

* *Слушайте, прежде чем говорить.* Если кто-то уже говорит, подо­ждите, пока он не закончит. В мире компьютерных сетей это пра­вило называется **контролем несущей** и заключается в том, что узел прослушивает канал перед тем, как начать передачу. Если по каналу передается кадр, узел выжидает («отступает») в течение случайного периода времени, а затем начинает передачу.
* ***Если кто-то начал говорить, прекращайте разговор.* В мире компью­терных** сетей это правило называется **обнаружением коллизий.** Оно заключается в том, что во время передачи узел прослушивает канал. Если он обнаруживает, что другой узел в этот момент времени тоже ведет передачу, он прекращает свою и выжидает в течение случайно­го периода времени, после чего начинает новый цикл проверки кана­ла и передачи, если канал оказывается не занят.

Эти два правила реализованы в семействе протоколов CSM A (Carrier Sense Multiple Access — **множественный доступ с контролем несущей)** и CSMA/CD (CSMA with Collision Detection — **множественный доступ с контролем несущей и обнаружением коллизий).** Было разработано множество вариантов протоколов CSMA и CSMA/CD. Да­лее будут рассмотрены наиболее важные и фундаментальные свойства CSMA и CSMA/CD.

Первый вопрос о протоколе CSMA, который может возникнуть, состоит в том, как вообще могут возникать коллизии, если все узлы прослушивают несущую? В самом деле, ведь узел воздерживается от передачи, если он замечает, что канал занят. Ответ на этот вопрос луч­ше всего проиллюстрировать с помощью пространственно-временной диаграммы. На рис. 5.12 показана пространственно-временная диа­грамма работы четырех узлов (А, Б, В, Г), присоединенных к линейной широковещательной шине. На горизонтальной оси фиксируется поло­жение каждого узла в пространстве, вдоль вертикальной оси изменяет­ся время.

****

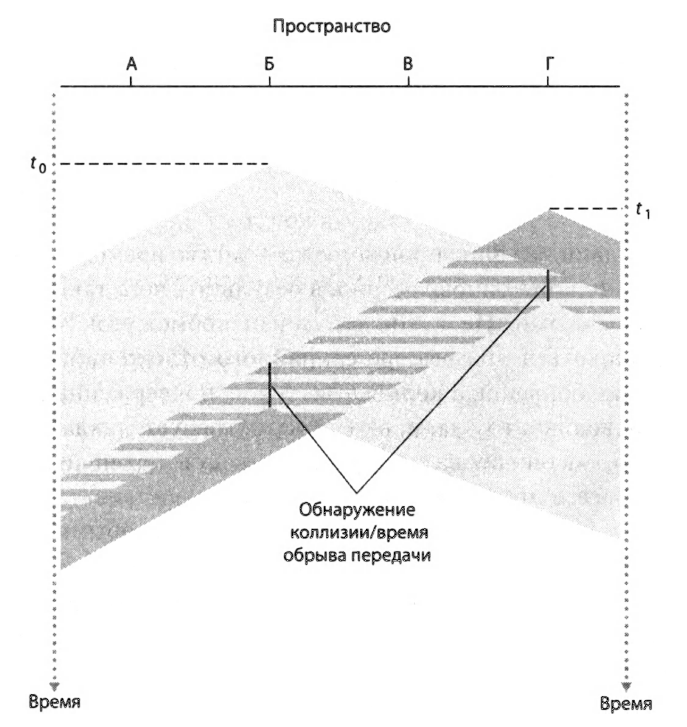
**Рис. 5.12. Пространственно-временная диаграмма с двумя узлами CSMA, между**

**которыми возникает коллизия при передаче**

В момент времени *t0* узел Б опрашивает канал и приходит к выводу, что канал свободен, так как никакой другой узел в этот момент не ведет передачу Поэтому узел Б начинает передачу, и передаваемый им сигнал распространяется по широковещательному носителю в обоих направ­лениях со скоростью, близкой к скорости света, но конечной. В момент времени *t1(t1> t0)* у узла Г появляется кадр для передачи. Хотя узел Б в этот момент *t1* уже отправляет данные, передаваемый им сигнал еще не достиг узла Г, таким образом, узел Г полагает, что канал в момент *t1* свободен. Поэтому в соответствии с протоколом CSMA узел Г начинает передачу своего кадра. Немного позднее сигнал, отправляемый узлом Б, смешивается с сигналом узла Г — возникает коллизия. Из рисунка видно, что производительность широковещательного канала в значительной степени зависит от **времени прохождения сигнала** по каналу от одного узла до другого. Чем больше это время, тем выше вероятность коллизий, вызванных тем, что сигнал уже начавшего передачу узла не успел достичь другого узла, готового к передаче.

**Множественный доступ с контролем несущей и обнаружением коллизий (CSMA/CD)**

Изображенные на рис. 5.12 узлы не обнаруживают коллизию. Оба узла (Б и Г) продолжают передавать свои кадры целиком и после кол­лизии. При использовании протокола с обнаружением коллизий узел прекращает передачу, как только обнаруживает коллизию. На рис. 5.13 показан тот же сценарий, что и на рис. 5.12, но в этом случае узлы прекра­щают передачу, обнаружив коллизию. Разумеется, добавление функции обнаружения коллизий к протоколу множественного доступа положи­тельно повлияет на производительность протокола, так как поврежден­ный (из-за наложения на кадр другого узла), а значит — бесполезный кадр не будет передаваться по нему целиком.

****

**Рис. 5.13. Протокол CSMA с обнаружением коллизий**

Прежде чем приступать к анализу протокола CSMA/CD, давайте рассмотрим, как он работает с точки зрения адаптера (на узле), подклю­ченного к широковещательному каналу.

1. Адаптер получает дейтаграмму с сетевого уровня, готовит кадр ка­нального уровня и заполняет буфер кадра в адаптере.
2. Если адаптер обнаруживает, что канал не занят (то есть на адаптер из этого канала не поступает никаких сигналов), он начинает пере­дачу кадра. Иначе, если адаптер узнает, что канал занят, то он до­жидается исчезновения сигналов в этом канале, после чего начинает передачу.
3. При передаче адаптер следит за тем, не пойдет ли какая-нибудь сиг­нальная энергия от других адаптеров, использующих этот широко­вещательный канал.
4. Если адаптер передает полный кадр, не зафиксировав встречных сигналов от других адаптеров, то обработка (передача) этого кадра завершена. В противном случае передача кадра обрывается (то есть не оканчивается).
5. После такого обрыва передачи адаптер выжидает в течение случай­ного интервала времени, а затем возвращается к шагу 2.

Длительность ожидания должна быть именно случайной (а не фик­сированной) по понятным причинам: если два узла сначала одновре­менно пытались передать по кадру, а когда это не удалось — перешли в режим ожидания на фиксированное количество времени — то и вый­дут из ожидания они одновременно, в результате чего такие коллизии будут бесконечными. Но каков разумный промежуток, от которого стоит отталкиваться при выборе **случайного отложенного времени?** Если интервал большой, а количество узлов, подверженных коллизи­ям, не столь велико, то узлам, вероятно, потребуется ждать слишком долго до того, как они снова смогут проверить, свободен ли канал (и на протяжении всего этого времени канал будет простаивать). С другой стороны, если интервал невелик, а узлов, между которыми возникают коллизии, очень много, то вполне вероятно, что многие значения, вы­бранные случайным образом, будут практически идентичны, и между передающими узлами вновь возникнет коллизия. Нам нужно, чтобы интервал был кратким при малом количестве узлов, но большим, если в сети окажется много узлов, между которыми могут возникать кол­лизии.

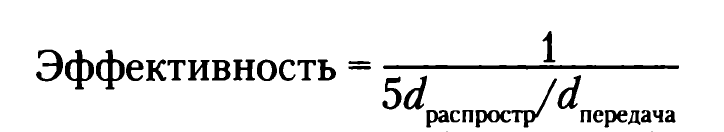
**Экспоненциальный двоичный алгоритм выдержки,** используемый в Ethernet, а также в протоколах множественного доступа кабельных се­тей DOCSIS, красиво решает эту проблему. Так, при передаче кадра, с которым уже произошло *п* коллизий, узел случайным образом выби­рает значение *К* из интервала {0,1,2,...*2п-*1}. Соответственно, чем больше коллизий испытал кадр, тем больше интервал, из которого выбирается *К.* В случае с Ethernet точная длительность ожидания для узла равна Хх512 бит раз (то есть *К —* кратное истечение того промежутка време­ни, который требуется для отправки 512 байт по сети Ethernet). Макси­мальное значение, которое может принимать *К,* жестко ограничено 10.

Рассмотрим пример. Предположим, что узел впервые пытается пе­редать кадр и при попытке передачи возникает коллизия. Тогда узел выбирает *К* = 0 с вероятностью 0,5 или *К* = 1, также с вероятностью 0,5. При выборе *К* = 0 узел сразу же начинает опрашивать канал. При вы­боре *К* = 1 узел выжидает, пока пройдет 512 бит (в Ethernet 100 Мбит/с это происходит примерно за 5,12 мс), после чего начинает новый цикл опроса и передает информацию, если обнаруживает, что канал про­стаивает. После второй коллизии значение *К* с равной вероятностью выбирается из интервала {0,1,2,3}. После трех коллизий этот интервал увеличивается до {0,1,2,3,4,5,6,7}. После 10 и более коллизий *К* с равной вероятностью выбирается из диапазона {0,1,2,..., 1023}. Соответственно, размер множества, из которого выбирается значение *К,* возрастает по экспоненте при увеличении количества коллизий. Именно поэтому данный алгоритм называется «экспоненциальным двоичным алгорит­мом выдержки».

Также отметим, что всякий раз, когда узел подготавливает кадр для передачи, он запускает алгоритм CSMA/CD. При этом не учитывается, сколько коллизий могло произойти в недавнем прошлом. Итак, сохра­няется вероятность, что новый кадр сможет проскользнуть сразу же (по­пасть в успешную передачу), в то время как другие кадры по-прежнему будут оставаться в состоянии экспоненциальной двоичной выдержки.

**Эффективность алгоритма CSMA/CD**

Когда кадр для отправки имеется только у одного узла, узел может задействовать для его передачи всю пропускную способность канала. Типичные значения пропускной способности в сетях Ethernet равны 10 Мбит/с, 100 Мбит/с или 1 Гбит/с. Однако если таких узлов много, то фактическая скорость передачи данных в канале может быть гораздо ниже. **Эффективность алгоритма CSMA/CD** на достаточно длитель­ном промежутке времени определяется как количество времени, за ко­торое кадры передаются по каналу без коллизий при наличии большого количества активных узлов, где каждый имеет множество кадров для отправки. Чтобы записать аналитическое выражение для приблизи­тельной оценки эффективности сетей Ethernet, обозначим с помощью *d*распростр максимальное время, требуемое сигналу для распространения между двумя адаптерами. Через *dneредача* обозначим время, требуемое для передачи максимально крупного кадра (для Ethernet со скоростью 10 Мбит/с это значение составляет около 1,2 мкс). Здесь мы просто укажем следующее выражение:



Из формулы видно, что по мере приближения *d*распростр к 0 эффективность приближается к 1. Это понятно и на интуитивном уровне: если на распространение тратится нулевое время, то конфликтующие узлы будут мгновенно отменять передачу, не тратя пространство канала впу­стую. Кроме того, когда значение *dne* ача чрезмерно увеличивается, эф­фективность стремится к 1. Также понятно, что, когда кадр занимает канал, он задерживается там на достаточно долгое время, но на всем его протяжении канал будет осуществлять полезную работу.