CHAPTER

第3章 局域网

1.4.2节中将网络按其分布范围分成局域网、城域网和广域网，由于不同的分布范围对实现数据传输的要求不同，因而相应有了适用于实现局域网、城域网和广域网对应分布范围内数据传输的传输网络，本章主要讨论适用于实现局域网对应分布范围内数据传输的传输网络——以太网和令牌环网，重点讨论适用于实现局域网和城域网对应分布范围内数据传输的传输网络——以太网。

3.1 局域网拓扑结构

在讨论具体传输网络时，不时会提到网络拓扑结构，拓扑(Topology)是拓扑学中研究由点、线组成几何图形的一种方法，用此方法可以把计算机网络看作是由一组结点和链路组成的几何图形，这些结点和链路所组成的几何图形就是网络的拓扑结构。由于局域网是一个可以由某个单位单独拥有，且允许自主布线的网络，因此，用户对局域网的拓扑结构有较大的选择空间。局域网常见的拓扑结构有总线形、星形、环形、树形和网状。

1. 总线形拓扑结构

总线形拓扑结构如图3.1所示，通常用同轴电缆作为网络中的总线。为了防止反射信号干扰总线上用于传输数据的基带信号，总线两端必须接匹配阻抗。总线形拓扑结构的优点是简单，缺点是连接在总线上的任何一个终端发生故障，都有可能使总线的阻抗发生变化，导致基带信号传输失败。

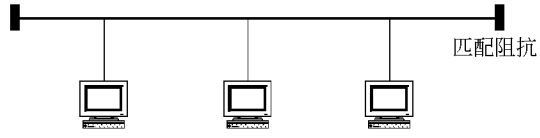

图3.1 总线形拓扑结构

2. 星形拓扑结构

星形拓扑结构如图3.2所示，网络核心设备是物理层的集线器或链路层的交换机，核心设备和终端之间的传输媒体一般为双绞线或光纤，尤其是双绞线作为局域网传输媒体后，其柔软性非常容易满足办公环境下的布线要求，因此出现了一个新的行业——综合布线，并因此将局域网设计和实施分为同等重要的两部分：解决设备之间互连问题的布线系统和实现数据端到端传输及提供应用服务的网络传输系统和应用系统。星形拓扑结构是目前局域网设计中普遍使用的网络结构，当然，实际局域网常常通过级联集线器或交换机将多个星形网络连接在一起。星形拓扑结构的优点是核心设备能够隔离每一个终端，因此，某个终端发生故障或者核心设备用于连接终端的某个端口发生故障，不会影响其他终端之间的通信，这是星形拓扑结构取代总线形拓扑结构的主要原因。

3. 环形拓扑结构

环形拓扑结构如图3.3所示，这种拓扑结构最初因为令牌环网(Token Ring)而得名，目前，令牌环网虽已不是局域网的主流技术，但环形拓扑结构的容错性，即环形物理链路任何一处发生故障，都不会影响终端之间连通性的特性，要求有容错性的局域网往往采用这种拓扑结构。

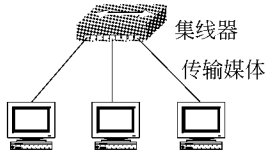

图3.2 星形拓扑结构

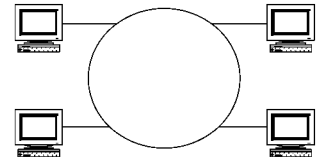

图3.3 环形拓扑结构

4. 树形拓扑结构

树形拓扑结构如图3.4所示，这种拓扑结构实际上就是通过级联交换机或集线器将多个星形拓扑结构连接在一起的网络结构。正常的树形结构要求任何两个终端之间不允许存在环路。

5. 网状拓扑结构

有容错性要求的局域网为了保证故障情况下终端之间的连通性，往往使交换机之间构成环路，这种在树形拓扑结构上增加环路的拓扑结构称为网状拓扑结构，如图3.5所示。

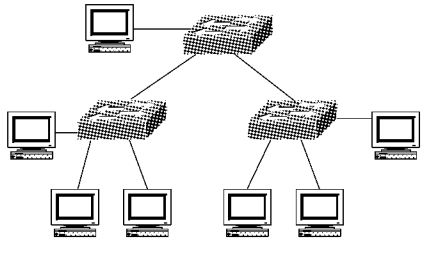

图3.4 树形拓扑结构

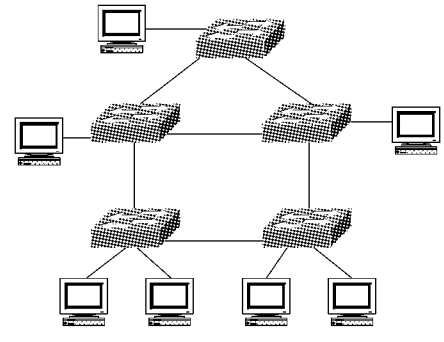

图3.5 网状拓扑结构

3.2 以 太 网

3.2.1 以太网体系结构

以太网是 20 世纪 70 年代初由 Bob Metcalfe 和 David Boggs 发明的，并以历史上表示传播电磁波的以太(Ether)命名。在 20 世纪 70 年代末，Dec、Intel 和 Xerox 这三家公司联合起来开发以太网产品，并在 1980 年 9 月发表了关于以太网规约的第一个版本——DIX V1(DIX 由这三家公司名称的第一个字母组合而成)，1982 年又修改发表了第二个版本——DIX Ethernet V2。电子和电气工程师协会(IEEE)802 委员会在此基础上制定了第一个局域网标准，编号为 802.3。实际上，802.3 标准和 DIX Ethernet V2 还是有点差别的，但目前人们已习惯将符合 802.3 标准的局域网称作以太网。以太网并不是 IEEE 802 委员会制定的唯一局域网标准，在制定 802.3 标准以后，又陆续制定了多个不同的局域网标准，如 3.1 节中讲到的令牌环网，由于不同局域网的链路层标准并不相同，为了给网络层提供统一的局域网功能界面，802 委员会将局域网的链路层分成两个子层：逻辑链路控制(Logical Link Control，LLC)子层和媒体接入控制(Medium Access Control，MAC)子层，因此，可以得出如图 3.6 所示的以以太网为传输网络的 TCP/IP 体系结构。

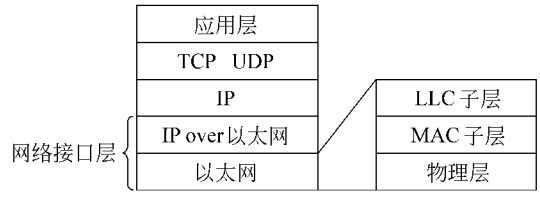

图 3.6　基于以太网的 TCP/IP 体系结构

不同局域网的 MAC 子层是不同的，但 LLC 子层和网际层之间的接口界面是相同的，也就是说 LLC 子层屏蔽了由于多种局域网并存而造成的 MAC 子层的不同，就像 BIOS 屏蔽了主板的差异一样。

以太网的物理层主要解决与传输媒体之间的接口，数字信号 0、1 的表示方式，数字信号的同步等问题。MAC 子层解决与通过以太网传输数据有关的其他一些问题。本章将重点讨论这两层的功能和实现机制。

随着以太网的发展，以太网在局域网市场中已完全取得垄断地位，目前已不存在多种局域网技术并存的问题，而且，LLC 子层是 802 委员会为屏蔽多种局域网之间的差异而提出的，显然不是 DIX Ethernet V2 中的一部分，因此，实际基于以太网的 TCP/IP 体系结构删除了 LLC 子层，如图 3.7 所示。

802.3 标准是将以太网作为适用于实现局域网、城域网对应分布范围内数据传输的传输网络提出的，而且，随着以太网的发展，以太网不仅在局域网市场中完全取得垄断地位，还成为城域网市场中的主流网络，这也是本章着重讨论以太网的原因。

3.2.2 总线形以太网

802.3 标准定义的以太网是一种采用总线形拓扑结构，物理层采用曼彻斯特编码，

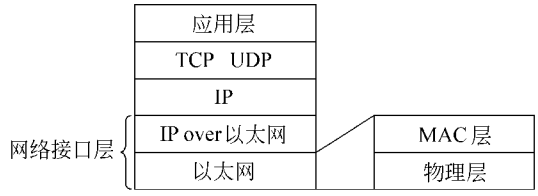

图 3.7 基于以太网的 TCP/IP 体系结构

MAC 层用 CSMA/CD 算法解决终端争用总线问题的网络。但随着以太网的发展，无论物理层的编码技术，还是网络拓扑结构都发生了很大变化。本节从总线形以太网开始，循序渐进，完整讨论以太网的发展历程。

1. 基带传输与曼彻斯特编码

1）基带传输和时钟同步

总线形以太网如图 3.8 所示，图中多台 PC 共同争用一根总线，总线两端连接匹配阻抗。图中 PC 可以称为终端，有时也称为站点。在双绞线出现之前，人们见到的以太网就是这种采用同轴电缆作为总线的总线形以太网结构。以太网物理层采用基带传输方式，所谓基带传输就是直接在同轴电缆上传输表示二进制数 0、1 的数字信号。这种传输方式下，以太网必须解决基带信号的同步问题，即确定二进制数 0 和 1 的表示方式及每一位二进制数的时间间隔。说到基带传输，很容易想到用两种不同幅度的电压来表示 0 和 1，为了简单起见，往往用 0V 电压表示二进制数 0，5V 电压表示二进制数 1（这里为讨论方便，基带信号采用 TTL 电平，实际上，为了消除直流电压，往往用对称的正负电压表示二进制数 0 和 1，如用 1.5V 表示二进制数 0，用－1.5V 表示二进制数 1）。每一位二进制数的时间间隔由传输速率确定，实际上它就是传输速率的倒数，如果以太网的传输速率为 10Mbps，则每一位二进制数的时间间隔＝10^{-7}s＝100ns。发送数据的计算机用时钟精确控制每一位二进制数的时间间隔，图 3.9 给出了发送端用时钟控制每一位二进制数时间间隔的过程。

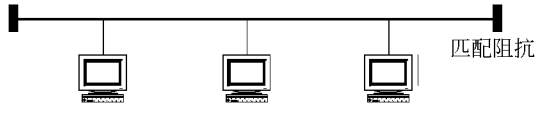

图 3.8 总线形以太网结构

接收端的时钟必须与发送端同步，即接收端时钟和数据中体现的发送端时钟同频同相。为了提高数据接收的可靠性，接收端在每一位数据的中间用时钟的正跳变来锁存数据。在接收端时钟和发送端时钟完全同步的情况下，这种传输过程可以正确进行，如图 3.9 中同步接收时钟和正确接收数据这部分内容。但要使两块网卡上的时钟严格一致是不可能的，它们之间肯定存在误差，图 3.9 中不同步接收时钟的波形表明接收端时钟周期小于发送端时钟周期（接收端时钟频率大于发送端时钟频率）。当然，两个时钟虽有误差，肯定不会像图 3.9 所示的那样严重，但时钟误差如果累积，出现图 3.9 中错误接收数据这样的情况只是时间早晚而已。实际上，两块网卡的时钟误差还是比较小的，只要这种误差不累积，就不会出现图 3.9 中错误接收数据这样的情况，但必须经常重新使接收时钟和发送时钟同步。重新同步是指即使接收时钟和发送时钟的频率不同，也强迫接收时钟的下降沿（或上升沿）和

数据中体现的发送时钟的下降沿(或上升沿)一致的过程,这样的时钟同步过程可以发生在每一个时钟周期,在保证累积时钟误差不会导致发生数据接收错误的情况下,也可以间隔若干个时钟周期发生一次,图 3.9 中重新同步的接收时钟是每一个时钟周期都重新同步接收时钟的情况。

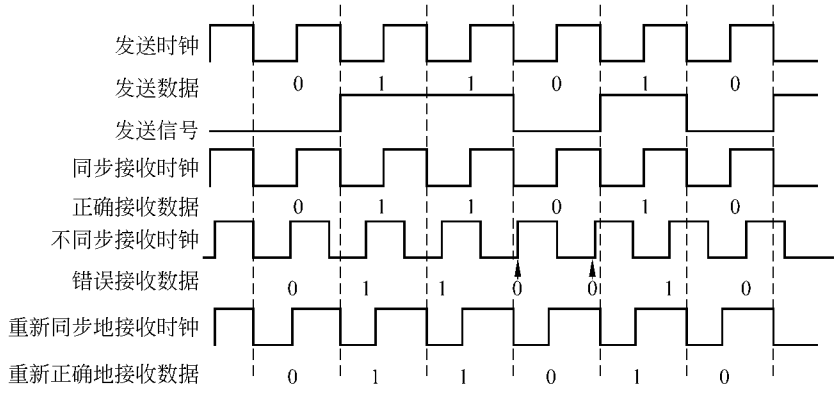

图 3.9　发送和接收数据过程

2) 用曼彻斯特编码传输发送时钟信息

由于发送端只发送数据给接收端,接收端为了使自己的时钟和发送端同步,必须能够从接收到的数据中提取发送端的时钟信息。实际上,图 3.9 所示的发送信号中包含着发送端的时钟信息,因为信号从 0 到 1 跳变或从 1 到 0 跳变恰好和发送时钟同步,接收端可以利用这种跳变来同步自己的时钟,但问题是如果基带信号中出现连续多位 0 或 1 的情况,接收时钟就很难和发送时钟同步了。

另外,多台计算机争用同一条总线,意味着任何时候只有一台计算机能够经过总线传输数据,在已经有计算机经过总线传输数据的情况下,别的计算机如何通过判别总线的状态来确定总线是闲还是忙? 能否有一种总线状态只有总线空闲时才具有,发送任何二进制数位流模式都不可能与此总线状态相同? 由于数字信号只用两种总线状态分别表示二进制数 0 和 1,针对当前用 0V 电压表示二进制数 0,5V 电压表示二进制数 1 的信号表示模式,可以肯定地说没有。

为了解决在数据中提取出发送端时钟信息的问题,另外也为了能够找到一种表示总线空闲的总线状态,以太网标准提出了曼彻斯特编码。

曼彻斯特编码将每一位二进制数的时间间隔分成两部分,对于二进制数 0,前半部分为高电平,而后半部分为低电平。对于二进制数 1,恰好相反,前半部分为低电平,后半部分为高电平(也可以采用相反约定,即 1 是先高后低,0 是先低后高)。一旦采用曼彻斯特编码,如果数据是连续 1 或连续 0,则同轴电缆上的电信号在每一位数据位的开始和中间都发生跳变。如果数据是 0、1 交替出现,则同轴电缆上的电信号在每一位数据的中间发生跳变。因此,无论何种二进制数位流模式,每一位数据至少发生一次跳变,接收端可以用该跳变来同步接收时钟。图 3.10 给出数据的曼彻斯特编码过程和用每一位数据中间的跳变来同步接收端时钟的情况。

如果采用曼彻斯特编码,一旦传输数据,总线上的电信号在每一位二进制数间隔内至少

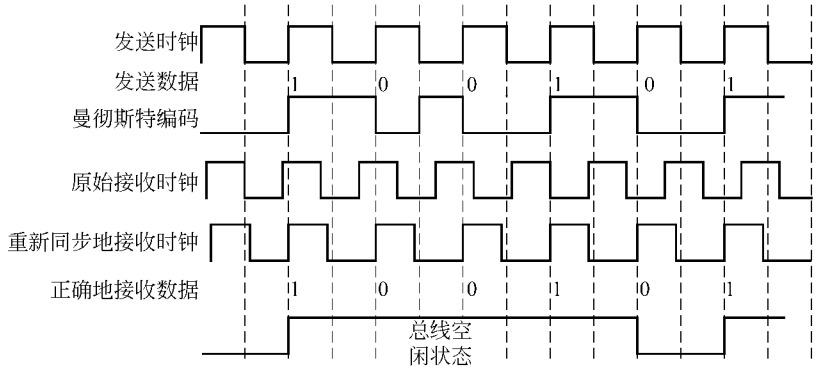

图 3.10　曼彻斯特编码和接收端时钟同步过程

发生一次跳变，因此可用一段时间内检测不到电信号跳变的总线信号状态来表示总线空闲。

2. MAC 帧结构

按照 OSI 网络体系结构，总线形以太网两端接匹配阻抗的单根同轴电缆就是物理链路，在物理链路上传输的是称为 MAC 帧的信息格式，它由数据和用于实现数据传输的地址信息及检测传输过程中出错情况的检错码等组成，总线形以太网的链路层称为 MAC 层，除了必须实现帧定界、寻址、差错控制这些基本链路层功能外，还需实现多点接入网络的接入控制功能。总线形以太网的多点接入方式和 HDLC 中的多点接入方式不同，HDLC 中由主站控制数据传输过程，数据传输只能在主站和从站间进行，但总线形以太网中每一个接入终端都是平等的，任何一个终端可以通过总线向另一个终端传输数据。总线只能进行半双工通信，另外，任何一个终端通过总线发送的数据可以被其他所有终端接收，总线的这一特性使其被称为广播信道。

1）帧定界

数据通过总线进行传输时，必须先对数据进行封装，由于总线形以太网的链路层为 MAC 层，因此，将总线形以太网的链路层封装形式称为 MAC 帧。由于在总线上通过基带信号传输的是一串二进制数位流，从一串二进制数位流中分离出每一帧，即确定每一帧的起始和结束字节的过程就是帧定界。

MAC 帧结构如图 3.11 所示，先导码和帧开始分界符并不是 MAC 帧的一部分，它们的作用是帮助接收终端完成帧定界的功能。

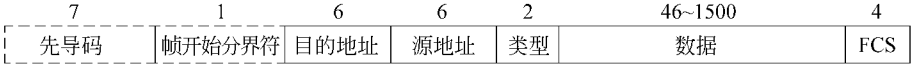

图 3.11　MAC 帧结构

先导码是由 7 个二进制数位流模式为 10101010 的字节组成的一组编码，它的作用是让连接在总线上的终端进行位同步（让接收端能够正确接收 MAC 帧中的每一位二进制数）。

帧开始分界符为 1 字节二进制数位流模式为 10101011 的编码，用于通知接收端该编码后面是 MAC 帧。这意味着连接在总线上的每一个终端，都必须通过先导码和帧开始分界符从经过总线传输的一串数字信号中正确定位 MAC 帧的起始字节（目的地址字段的第一个字节）。

如果连接在总线上的每一个终端都能够正确监测到先导码和帧开始分界符，实现帧定界是没有问题的。由于总线形以太网规定在传输完每一帧后，必须让总线空闲一段时间，而曼彻斯特编码很容易让终端监测到总线从空闲状态转变为发送先导码状态，并因此实现帧定界功能。所以，严格地说，总线形以太网的帧定界并不是由 MAC 层实现，而是由物理层实现的。

2）寻址

由于总线形以太网在同一总线上连接多个终端，通过总线传输的每一帧寻找接收终端的过程就是寻址过程。

MAC 帧中的目的地址和源地址字段给出该 MAC 帧的接收端和发送端的地址，由于这些地址在总线形以太网的 MAC 层识别和处理，被称为 MAC 地址，MAC 地址由 6 个字节组成。48 位 MAC 地址的最低位是 I/G 位，该位为 0，表示该 MAC 地址对应单个终端，该位为 1，表示该 MAC 地址对应一组终端，由于源地址表示发送 MAC 帧的终端地址，因此，源地址的 I/G 位为 0。48 位 MAC 地址的次低位是 G/L 位，该位为 0，表示该 MAC 地址是局部地址，该位为 1，表示该 MAC 地址是全局地址，全局地址是指该 MAC 地址全球范围内唯一。MAC 帧携带 MAC 地址的原因是总线形以太网是一个多点接入网络，允许同时有多个（大于 2 个）终端或网络互连设备接入总线形以太网，这种情况下，只有携带用于标识接收端的目的地址信息的 MAC 帧，才能确定它的接收终端，并因此实现寻址功能。

安装在终端中的每一块以太网卡（也称网络适配器）都有唯一的 48 位 MAC 地址。网卡 MAC 地址在出厂时已经设定，不可更改。当终端 A 发送 MAC 帧给终端 B 时，用终端 A 网卡的 MAC 地址作为 MAC 帧的源 MAC 地址，用终端 B 网卡的 MAC 地址作为 MAC 帧的目的 MAC 地址，当终端 A 通过总线发送该 MAC 帧时，连接在总线上的所有终端都接收到该 MAC 帧，但都用自己网卡的 MAC 地址和 MAC 帧中的目的 MAC 地址比较，如果相符，则继续处理，否则将该 MAC 帧丢弃。因此，当一个终端想要给另一个终端发送 MAC 帧时，它必须先获取另一个终端的 MAC 地址，否则，只能以广播方式发送 MAC 帧。

MAC 地址分为单播地址、广播地址和组播地址。

广播地址是 48 位全 1 的地址，用十六进制数表示是 ff:ff:ff:ff:ff:ff（6 个用冒号分隔的全 1 字节）。

组播地址范围是：01:00:5e:00:00:00～01:00:5e:7f:ff:ff。

单播地址是广播和组播地址以外且 I/G 位为 0 的 MAC 地址。

MAC 帧的源 MAC 地址必须是单播地址，目的 MAC 地址可以是单播地址、组播地址或广播地址。对于图 3.8 所示的总线形结构，目的 MAC 地址类型对 MAC 帧的发送方式没有影响，无论目的 MAC 地址是单播、组播或广播地址，都同样将 MAC 帧发送到总线上，被连接在总线上的所有其他终端所接收。

目的 MAC 地址类型对接收到 MAC 帧的终端的处理方式有极大关系，如果是单播 MAC 地址，则只有目的 MAC 地址和其网卡 MAC 地址相符的单个终端接收并处理该 MAC 帧。如果是广播地址，则连接在总线上的所有终端均需接收并处理该 MAC 帧。如果是组播地址，只有属于组播地址所指定的组播组的终端才需要接收并处理该 MAC 帧。组播地址前 25 位是固定不变的，为 01:00:5e:0x（第 4 个字节中只有最高位固定为 0），后 23 位用于标识组播组。组播是一种非常有用的技术，主要用于视频点播（VOD）、视频会议

和远程教育等。

3）差错控制功能

帧检验序列(FCS)字段用于接收端检验MAC帧在传输过程中是否出错。总线形以太网采用循环冗余检验(CRC)码对MAC帧进行检错，使用的生成多项式如下：

$$G(x)=x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^{8}+x^{7}+x^{5}+x^{4}+x^{2}+x+1$$

32位帧检验序列(FCS)就是用生成多项式除以由目的地址、源地址、类型、数据和填充字段组合成的二进制数位流后得到的余数。由于CRC对检测连续多位二进制数出错非常有效，而且无论用同轴电缆、双绞线还是光纤传输基带信号，有效传输距离内的误码率都很低，因此，总线形以太网在MAC层通过帧检验序列(FCS)字段能够检测出大多数的传输错误。

接收端一旦通过FCS字段检测出MAC帧传输过程中发生的错误，将丢弃该MAC帧。由于MAC层只有检错功能，没有重传机制，因此，一旦发生MAC帧传输出错的情况，MAC层并没有相应的补救措施。

4）MAC帧其他字段功能

类型字段用于标明数据类型，MAC帧所封装的数据可以是IP分组，也可以是ARP请求及其他类型的数据，包含不同类型数据的MAC帧需要提交给不同的进程进行处理，类型字段就用于接收端选择和数据的类型相对应的进程。

数据字段用于传输数据，和其他字段不同，数据字段才真正承载高层协议要求传输的数据，而其他字段只是用于保证数据的正确传输，因此，把数据字段称作MAC帧的净荷字段(也称载荷字段)，数据字段的长度是可变的。MAC帧的这种组织结构很容易让终端从MAC帧中分离出数据。

MAC帧有着严格的长度限制，它的长度必须在64字节和1518字节之间，由于其他字段占用了18个字节(6个字节源MAC地址＋6个字节目的MAC地址＋2个字节类型＋4个字节帧检验序列)，数据字段长度应该在46字节和1500字节之间，但高层协议要求传输的数据的长度应该是任意的，一旦数据的字节数不足46字节，就需要用填充字段将MAC帧的长度延长到64字节，由此可以推出填充字段的长度在0字节和46字节之间。

对MAC帧规定长度上限可以理解，因为一是每个接收终端的缓冲器空间有限，二是不能允许某个终端通过发送无限长的MAC帧独占总线，三是一旦长度很长的MAC帧传输出错，重新传输的代价太大。但对MAC帧规定长度下限却不好理解，下面章节将详细讨论对MAC帧规定长度下限的意义。

3. CSMA/CD操作过程

1）CSMA/CD工作原理

如果图3.8中某一个终端想要发送数据，通过CSMA/CD算法解决和其他终端争用总线的问题。CSMA/CD的中文是载波侦听(Carrier Sense，CS)、多点接入(Multiple Access，MA)/冲突检测(Collision Detection，CD)，它用于解决多个终端争用总线的机制如下：

(1) 先听再讲

想发送数据的终端必须确定总线上没有其他终端正在发送数据后，才能开始往总线上发送数据，即先要侦听总线上是否有载波(这里的载波指其他终端发送MAC帧时产生的高

低电平有规律跳变的电信号，不是调制过程中使用的特定频率的正弦信号），在确定总线空闲（无载波出现）的情况下，才能开始发送数据。一旦开始发送数据，随着电信号在总线上传播，总线上所有其他终端都能侦听到载波存在，这就是先听（侦听总线载波）再讲（发送数据）。

（2）等待帧间最小间隔

并不是一侦听到总线空闲就立即发送数据，而必须侦听到总线空闲一段时间（称为帧间最小间隔：IFG，10Mbps 以太网的帧间最小间隔为 9.6μs）后，才能开始发送数据。这样做的目的有三：一是如果接连两帧 MAC 帧的接收终端相同，必须在两帧之间给接收终端一点用于腾出缓冲器空间的时间。二是一个想连续发送数据的终端，在发送完当前帧后，不允许接着发送下一帧，必须和其他终端公平争用发送下一帧的机会。三是总线在发送完一帧 MAC 帧后，必须回到空闲状态，以便在发送下一帧 MAC 帧时，能够让连接在总线上的终端正确监测到先导码和帧开始分界符。

（3）边讲边听

一旦某个终端开始发送数据，其他终端都能侦听到载波，即使这些终端中存在想发送数据的终端，它也必须等待，直到总线空闲一段时间（由 IFG 确定）后，才能开始发送数据。但可能存在这样一种情况，两个终端都想发送数据，因此都开始侦听总线，当当前帧发送完毕时，两个终端同时侦听到总线空闲，并在总线空闲状态持续一段时间后，同时发送数据，这样，两个终端发送的电信号就会叠加在总线上，导致冲突发生。其实，由于电信号经过总线传播需要时间，如果两个终端相隔较远，即使一个终端开始发送数据，在电信号传播到另一个终端前，另一个终端仍然认为总线空闲，因此，即使不是同时开始侦听总线，只要两个终端开始侦听总线的时间差在电信号传播时延内，仍然可能发生冲突。因此，某个终端开始发送数据后，必须一直检测总线上是否发生冲突，如果检测到冲突发生，停止正常的数据传输，发送 4 字节或 6 字节长度的阻塞信号（也称干扰信号），加重冲突情况，使所有发送数据的终端都能检测到冲突情况的发生。这就是边讲（发送数据）边听（检测冲突是否发生）。检测冲突是否发生的方法很多，其中比较简单的一种是边发送边接收，并将接收到的数据和发送的数据进行比较，一旦发现不相符的情况，表明冲突发生。

（4）退后再讲

一旦检测到冲突发生，停止数据发送过程，延迟一段时间后，再开始侦听总线。两个终端的延迟时间必须不同，否则可能进入发送→冲突→延迟→侦听→发送→冲突这样的循环中。如果两个终端的延迟时间不同，延迟时间短的终端先开始侦听总线，在侦听到总线空闲并持续空闲一段时间后，开始发送数据，当延迟时间长的终端开始侦听总线时，另一个终端已经开始发送数据，它必须等待总线空闲后，才可以开始发送过程，CSMA/CD 操作过程如图 3.12 所示。

2）后退算法

发生冲突后，两个终端的延迟时间必须不同，否则将再次发生冲突，如何使两个终端产生不同的延迟时间呢？以太网采用称为截断二进制指数类型的后退算法，算法的基本思路如下：

① 确定参数 K。开始时 $K=0$，每发生一次冲突，K 就加 1，但 K 不能超过 10，因此，$K=\text{MIN}[\text{冲突次数},10]$。

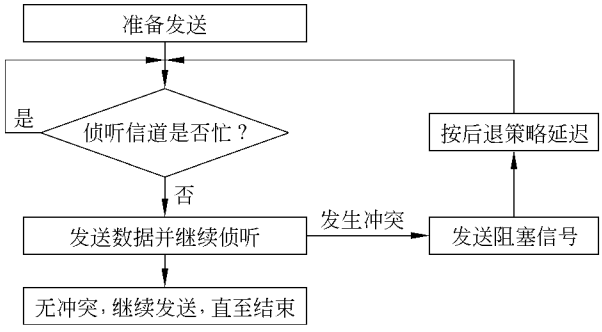

图 3.12 CSMA/CD 操作过程

② 从整数集合[0,1,…,2^K-1]中随机选择某个整数 r。

③ 根据 r，计算出后退时间 $T=r\times t$（t 是最大往返时延，对于 10Mbps 以太网，$t=51.2\mu s$）。

④ 如果连续重传了 16 次都检测到冲突发生，则终止传输，并向高层协议报告。

一旦两个终端发生冲突，每一个终端单独执行后退算法，在计算延迟时间时，对于第一次冲突，$K=1$，两个终端各自在[0,1]中随机挑选一个整数，由于只有 2 种挑选结果，两个终端挑选相同整数的概率为 50%。如果两个终端在第一次发生冲突后挑选了相同整数，则将再一次发生冲突。当检测到第二次冲突发生时，两个终端各自在[0,1,2,3]中随机挑选整数，由于选择余地增大，两个终端挑选到相同整数的概率降为 25%。随着冲突次数不断增加，两个终端产生相同延迟时间的概率不断降低。当两个终端的延迟时间不同时，选择较小延迟时间的终端先成功发送数据。

截断二进制指数类型的后退算法是一种自适应后退算法，为了提高总线的利用率，在发生冲突时，最好做到：①同时参与争用总线行动的终端的延迟时间不能相同；②最小的且与其他终端的延迟时间不同的延迟时间最好为 0。当参与争用总线行动的终端少时（最少为 2 个），有 50%的可能是一个终端选择 0 延迟时间，另一个终端选择 51.2μs 的延迟时间，选择 0 延迟时间的终端可以立即侦听总线，并在总线空闲时发送数据，这对提高总线利用率当然有益。但当有多个终端（假定为 100 台）参与争用总线的行动时，在第一次冲突发生时，其中一个终端选择 0 延迟时间，其余 99 个终端选择 51.2μs 延迟时间的概率实在太小。但随着冲突次数的不断增多，整数集合的不断扩大，很有可能在发生 16 次冲突前，有一个终端选择了整数 r，它和所有其他终端选择的整数不同，且小于所有其他终端选择的整数。

3）捕获效应

截断二进制指数类型的后退算法在两个终端都想连续发送数据的情况下，有可能导致一个终端长时间内一直争到总线发送数据，而另一个终端长时间内一直争不到总线发送数据，即所谓的捕获效应。

如图 3.13 所示，当两个终端同时想长时间发送数据时，都去侦听总线，当总线空闲后（总线持续空闲 IFG 规定的时间），两个终端同时向总线发送数据，导致冲突发生，用截断二进制指数类型算法求出后退时间，假定终端 A 选择的后退时间为 $0\times t$，而终端 B 选择的后退时间为 $1\times t$，终端 A 的第 1 帧数据发送成功。由于终端 A 有大量数据需要发送，在发送完第 1 帧数据后，紧接着发送第 2 帧数据，但必须通过争用总线过程才能获得发送第 2 帧数据的机会。当终端 A 和终端 B 又侦听到总线空闲，并又同时发送数据，导致冲突再次发生