第 3 章　数据链路层

“链路”和“数据链路”不同，前者是指一条无源的点对点的物理线路段，中间没有任何交换结点。有时也称为“物理链路”。

在一条线路上传送数据时，除了必须有一条物理线路外，还必须有一些规程来控制这些数据的传输。将实现这些规程的硬件和软件加到链路上就构成了数据链路。当采用复用技术时，一条链路上可以有多条数据链路。

数据链路层位于第 2 层，其传输原理如图 3-1 所示。主机 A 的数据从应用层出发，向下到数据链路层，以帧的方式发送到下一个结点（路由器 1），再经过路由器 2 和路由器 3 两个结点到达目的地主机 B，期间将经过局域网和广域网等不同的网络，主机 A 和主机 B 连接到网络的方式可能是电话网接入或局域网等接入方式。当然，从实际传输上看，其传输是经过物理层由通信线路到达邻近结点。

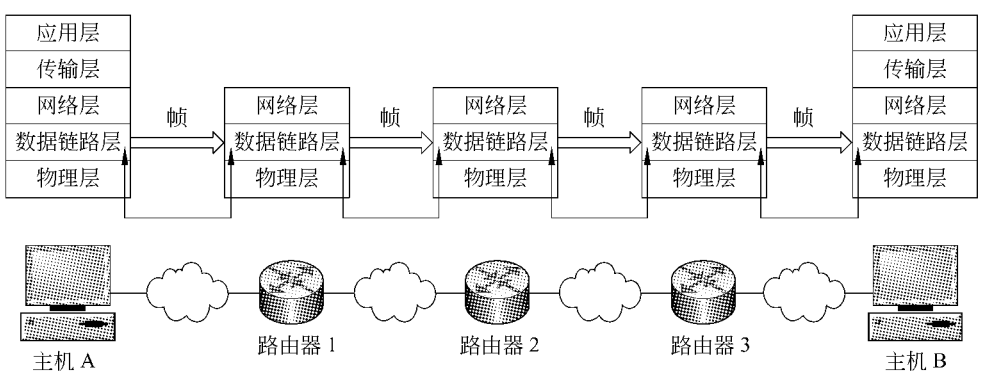


图 3-1 数据链路层的工作示意图

3.1　数据链路层的功能

数据链路层的主要功能如下。

（1）为网络层提供服务

数据链路层的基本任务是将源机器中来自网络层的数据传输到目的机器的网络层，它提供的基本服务有以下三种。

- 无确认的无连接服务。不需要建立链路连接，每个帧上都携带目的地址，形成独立的帧。接收方对收到的帧不作确认。如果由于线路噪声而造成帧丢失，数据链路层不负责重发，留待上层去完成。因此，这种服务主要适用于误码率低、实时性要求较高的传输环境，如局域网。
- 有确认的无连接服务。这是在上述服务中引入了确认的功能。每收到一个帧，接收方都要发回确认信号。如果发送方在规定的时间内没有收到确认信号，该帧就需要重新发送。这类服务适用于可靠性不高的信道，如无线通信系统。
- 有确认的面向连接的服务。具有连接建立、数据传输和连接释放的三个阶段。所有的帧都有序号，每一帧都有确认信号。大多数广域网的通信子网的数据链路层都采用这种服务。

(2) 组帧

网络层的数据加上首部和尾部后，以帧为单位向下传输。这样，即使在物理层传输中出现了错误，接收端也只需要将有错的帧重发，而不必将全部数据重新发送，从而提高了效率。为了明确一个帧的开始和结束，就需要加上一定的标志，实现帧同步或帧定界，如图 3-2 所示。图 3-2 中，MTU 是最大传输单元，是帧的数据部分长度的上限。帧的长度等于数据部分长度加上帧首部和帧尾部的长度。

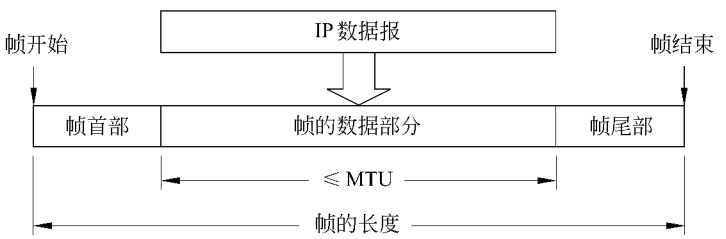


图 3-2 帧的结构模型

(3) 差错控制

当一帧到达目的地时，需要校验其正确性。其正确性是通过差错控制码产生的校验和来确定的，首先在发送端计算校验和后连同数据一起发送，然后在接收端进行检错处理。如果是正确的，则把数据部分向网络层传送。否则，做丢弃处理。

(4) 流量控制

流量控制用于解决发送能力大于接收能力的问题，如果接收方来不及接收，就会有许多帧丢失。流量控制实际上是控制发送方的数据流量，使其发送速率不超过接收方所能处理的程度。

【例 3-1】 在数据链路层根据什么原则来确定应当使用面向连接服务还是无连接服务？

解析： 在设计硬件时就能够确定。例如，若采用拨号电路，则数据链路层将使用面向连接服务。但若采用以太网，则数据链路层使用的将是无连接服务。

3.2　组帧技术

组帧技术，也称为帧同步技术。有不同的组帧方式，以字节为单位组成帧的各部分字段的方式称为面向字节的组帧方式；以任意比特组合成帧的方式称为面向比特的组帧方式。下面介绍四种组帧方法。

3.2.1　字节计数法

这种帧同步方法是一种面向字节的同步规程，是利用帧头部中的一个域来指定该帧中的字节数的，其原理如图 3-3 所示。

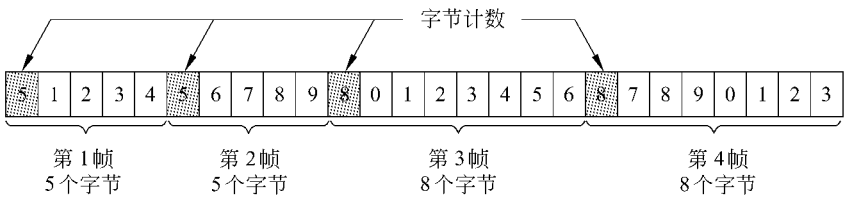


图 3-3　字节计数法示例

图 3-3 标识了四个数据帧的帧格式，它们的大小依次为 5、5、8、8 个字节。接收方可以通过对该特殊字符的识别从比特流中区分出帧的起始，并获知该帧的数据字节数，从而确定出帧的终止位置。

这种方法最大的问题在于如果标识帧大小的字段出错，即失去了帧边界划分的依据，那么将造成灾难性的后果。如第 2 帧中的计数字节由“5”变为“7”，则接收方就会失去帧同步的可能，从而不可能再找到下一帧正确的起始位置。由于第 2 帧的校验和出现了错误，所以即使接收方给发送方请示重传都无济于事。因此，目前已很少使用这种字节计数法。

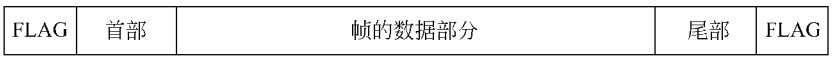
3.2.2　字符填充法

该同步方法是用一些特定的字符来作为一个帧的开始和结束标志，同时也采用某些特定的字符作为传输过程中的控制字符。在过去，开始和结束字节并不相同，但在最近几年，绝大多数协议倾向于使用相同的字节，这些字符称为标志字节，如图 3-4(a)中的 FLAG 所示。因此，接收方如果丢失了同步，只需要搜索标志字节就能找到当前帧的结束位置。两个连续的标志字节代表了当前帧的结束和下一帧的开始。

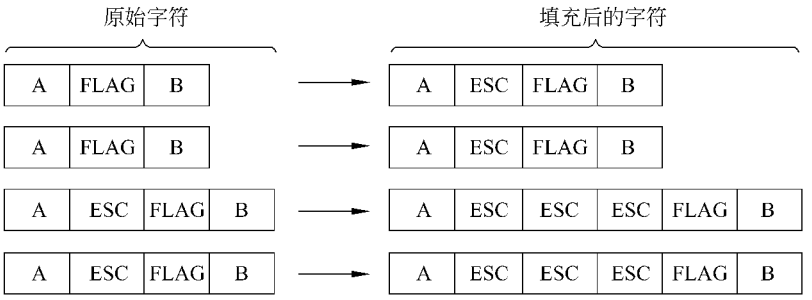
如果标志字节出现在数据中，发送方则在该字节前插入一个特殊的转义字节(ESC)，接收方在删除该 ESC 字节后才将数据送交给网络层，这就是字节填充技术。如果 ESC 字节也出现在数据中，则采用同样的处理方式，在其前面插入一个 ESC。在图 3-4(b)中，给出了四种不同的原始字符序列及其填充结果。

这种方法依赖于 8 位字符模式，对其他类型的字符码并不适用。例如，UNICODE 使

用的是 16 位字符。所以，需要开发新的技术，以便适用于任意长度的字符。



(a) 有标志字符作分界的帧



(b) 字节填充前后的比较

图 3-4　字符填充法示例

3.2.3　零比特填充法

这是以一组特定的比特模式(01111110)来标志一帧的开始和结束的方法，它允许任意长度的位码，也允许每个字符有任意长度的位。在发送方的数据链路层，当数据中遇到有 5 个连续的比特“1”时，则自动在其输出位流中填充一个比特“0”。在接收方的数据链路层，当数据中收到连续 5 个“1”、且其后是“0”时，则自动删除该“0”比特。其工作原理如图 3-5 所示，如果要传输的数据帧为“011011111111101111110010”，采用零比特填充后，在网络中传送时表示为“0110111110111011111010010”。

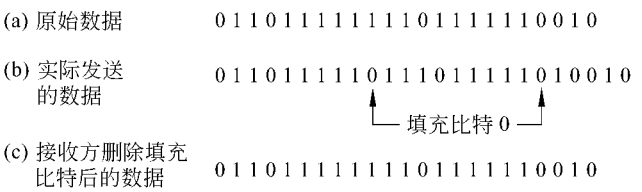


图 3-5　零比特填充法示例

零比特填充帧同步方式很容易由硬件来实现，其性能优于字符填充方式。所有面向比特的同步控制协议都采用统一的帧格式，不论是数据还是单独的控制信息均以帧为单位传送，其典型代表是 HDLC 协议。

3.2.4　违例编码法

这种方法在物理层采用特定的比特编码方法时采用。例如，曼彻斯特编码方法是将数据比特“0”编码成“高—低”电平对，将数据比特“1”编码成“低—高”电平对。而“高—高”电平对和“低—低”电平对在数据比特中是违法的。可以借用这些违法编码序列来界

定帧的开始和结束。局域网 IEEE 802 标准中就采用了这种方法。违法编码法不需要任何填充技术便能实现数据的透明性，但它只适于采用冗余编码的特殊编码环境。

由于字节计数法中计数字段的脆弱性及字符填充实现上的复杂性和不兼容性，目前使用较普遍的帧同步法是零比特填充法和违法编码法。

【例 3-2】 数据链路协议中使用了下面的字符编码。

A：01000111；B：11100011；FLAG：01111110；ESC：11100000。

为了传输一个包含 4 个字符的帧：A B ESC FLAG，请给出使用下面的成帧方法时所对应的位序列(用二进制表达)：

(a) 字节计数；

(b) 包含字节填充的标志字节；

(c) 包含零比特填充的开始和结束标志。

解析：

(a) 有字符 4 个，二进制表示为 00000100。则按照字符计数，这 4 个字符应表示为：

00000100 01000111 11100011 11100000 01111110

(b) 首尾各加上一个 FLAG 来标志一帧的开始和结束，再考虑字节填充法，则结果为：FLAG A B ESC ESC ESC FLAG FLAG，对应的二进制编码是：

01111110 01000111 11100011 11100000 11100000 11100000 01111110 01111110

(c) 采用特定的比特模式(01111110)来标志一帧的开始和结束。再考虑零比特填充法，得到结果为：

01111110 01000111 110100011 111000000 011111010 01111110

3.3　差 错 控 制

理想的通信系统在现实中是不存在的，信息传输过程中总会出现差错。差错控制是指采用编码技术，在通信过程中检测并发现差错，对差错进行纠正，从而将差错控制在尽可能小的范围内。能检测差错的编码称为检错码，如奇偶校验码、循环冗余校验码以及校验和；能检测并纠正错误的编码称为纠错码，如海明码。纠错码的实现过程复杂，在一般的通信场合不适宜采用；检错码的实现较为容易，能够通过重传机制获得正确的帧，因此在网络中广泛使用。

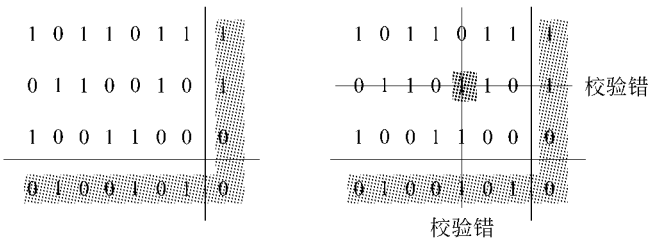
3.3.1　奇偶检验码

奇偶检验码是最常见的一种检错码，主要用于以字符为传输单位的通信系统中。其工作原理很简单，即在原始数据字节的最高位或最低位增加一位，作为奇偶检验位，以保证所传输的每个字符中 1 的个数为奇数(奇校验)或偶数(偶校验)。

国际标准规定在同步传输中使用奇校验，而在异步传输中使用偶校验。

奇偶检验只能检测出奇数个位发生的错误，校验能力低，不适合用于块数据的传输，块数据的传输需要采用垂直水平奇偶检验码(也称为纵横奇偶检验或方阵码)，其工作原

理如图 3-6 所示。



(a) 发送方的方阵偶校验　　(b) 接收方的方阵偶校验

图 3-6　方阵校验码应用示例

图 3-6 中有一组由 3 行 7 列的数据组成的块，每行的最右位是水平奇偶检验位，每列的最低位是垂直奇偶检验位。由于增加了纵向的校验，所以其检错能力得到了提高，能够检测出所有 3 位或 3 位以下的错误以及奇数个错和大部分偶数个错，它可以使误码率下降到原误码率的百分之一到万分之一。

该方阵码的另一个特点是可以纠正部分差错。如图 3-6(b)所示中第 2 行和第 5 列的奇偶检验出错，则可以判定当前位置的数据位发生了传输错误，将之取反即可纠正差错。

3.3.2　循环冗余校验码

循环冗余校验(Cyclic Redundancy Check，CRC)编码是局域网和广域网的数据链路层通信中用得最多也是最有效的检错方式。其基本思想是在数据后面添加一组与数据相关的冗余码，冗余码的位数越多，则其检错能力越强，但传输的额外开销也越大。

CRC 码又称为多项式码，任何一个由二进制数位串组成的代码都可以和一个只含有 0 和 1 两个系数的多项式建立一一对应的关系，例如，代码 1011011 对应的多项式为 $x^6+x^4+x^3+x+1$。k 位要发送的信息位可对应于一个$(k-1)$次多项式 $M(x)$，r 位冗余位对应于一个$(r-1)$次多项式 $R(x)$。由 k 位信息位后面加上 r 位冗余位组成的 $n=k+r$ 位的编码即 CRC 码，对应于一个$(n-1)$次多项式 $F(x)=x^rM(x)\oplus R(x)$。

由信息位产生冗余位的编码过程，就是已知 $M(x)$求 $R(x)$的过程。在 CRC 码中，可以通过找到一个特定的 r 次多项式 $G(x)$来实现，用 $G(x)$去除 $x^rM(x)$得到的余式就是 $R(x)$。

在接收方，校验的方法是用生成多项式 $G(x)$去除接收到的 $x^rM(x)\oplus R(x)$，若不能整除，表明传输中出错，但无法指明错误位置。

注意，这里的加法和除法都是基于模 2 的运算，其特点是不考虑进位和借位的运算，相当于异或运算。

【例 3-3】 假设要发送的数据为 101110，采用 CRC 的生成多项式是 $G(x)=x^3+1$，请问：

(1) 冗余码和发送的码字分别是什么？

(2) 若收到的数据序列是 100010011，请判断是否有错？

解析： 已知发送的信息 $M=101110$，生成多项式对应的除数 $G=1001$。

(1) 经过除法运算，如图 3-7 所示，得到冗余码为 $R=011$，所以发送的码字

是 101110011。

(2) 如图 3-8 所示，用除数 G 去除收到的数据序列 100010011，得到结果是 101，不是全零。所以，收到的数据序列有错。

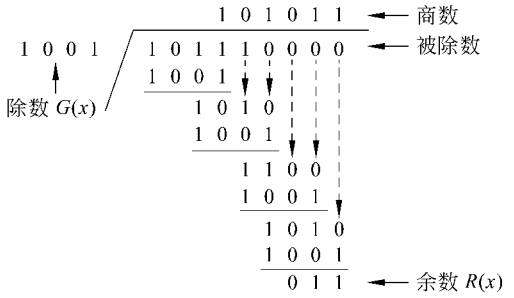


图 3-7　CRC 计算示例

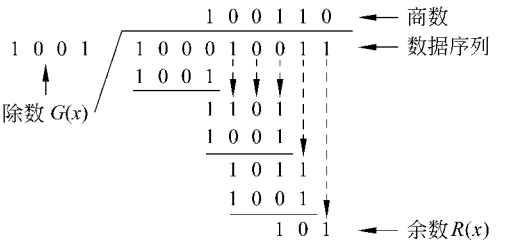


图 3-8　CRC 校验接收的数据

需要注意的是，余数为 0 并不能断定传输中一定无错。在某些非常特殊的位差错组合下，CRC 完全有可能使余数为 0，所以其检错率并非 100%。经过精心设计和实际检验，目前国际上已被标准化的生成多项式 $G(x)$ 主要有以下几种。

CRC-8：x^8+x^2+x+1

CRC-12：$x^{12}+x^{11}+x^3+x^2+x+1$

CRC-16：$x^{16}+x^{15}+x^2+1$

CRC_CCITT：$x^{16}+x^{12}+x^5+1$

CRC-32：$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$

它们在实际通信中得到广泛的应用，如 CRC-8 被用于 ATM 信元头差错校验中，CRC-16 被用于二进制同步传输规程中，CRC_CCITT 被用于 HDLC 通信规程中，CRC-32 被用于 IEEE 802.3 以太网的数据链路层通信中。

除了以上两种差错校验方法外，还有校验和校验方法。校验和的校验计算过程简单，但其校验差错的能力远不如 CRC 高。用 CRC 检测方法无论是生成校验码还是进行差错检测计算都比较复杂，在数据链路层一般都是使用硬件技术生成校验码和实现数据校验操作。因此，CRC 技术常用在数据链路层，而校验和技术一般用在链路层之上的高层。另外，设置高层校验也是为了避免因数据链路层漏检或硬件设备及软件系统异常而误收错误分组。

3.4　流量控制

在链路的两个站点之间建立了链路连接后，就进入了数据传输阶段。为了保证数据传输的正确性和完整性，必须建立一套通信协议。

流量控制用于控制协调链路中发送方和接收方之间的数据流量，保证双方的数据发送和接收达到平衡。当接收方来不及接收时，就必须及时控制发送方的发送速率。流量控制可以有效地防止网络中由于瞬间的大量数据对网络带来的冲击，保证用户的网络运行。

流量控制不仅可以在数据链路层上实现，而且在其他高层，如网络层和传输层上也有相应的控制机制。不同功能层的流量控制的对象是不同的，数据链路层上控制的是网络中相邻结点之间的数据传输，网络层上控制的是网络源结点和目的结点之间的数据传输，而传输层上控制的是网络中不同结点内发送进程和接收进程之间的数据传输。

目前，通信结点之间常用的流量控制技术有停止—等待方式（简称为停等方式）和滑动窗口方式，分别对应有停止—等待协议（简称为停等协议）和滑动窗口协议，后者包括后退 N 帧协议和选择重传协议。

3.4.1 停等协议

停等协议是最简单的流量控制协议。在数据传输之前，发送方已将上层来的分组装配成帧，分为数据帧和确认帧。每当发送完一个数据帧，就主动停止，等待接收方的确认。如果收到的是肯定应答，则接着发送下一个帧；如果收到否定应答或在规定的时间内没有收到任何应答，则重发该帧。停等协议的工作流程如图 3-9 所示。

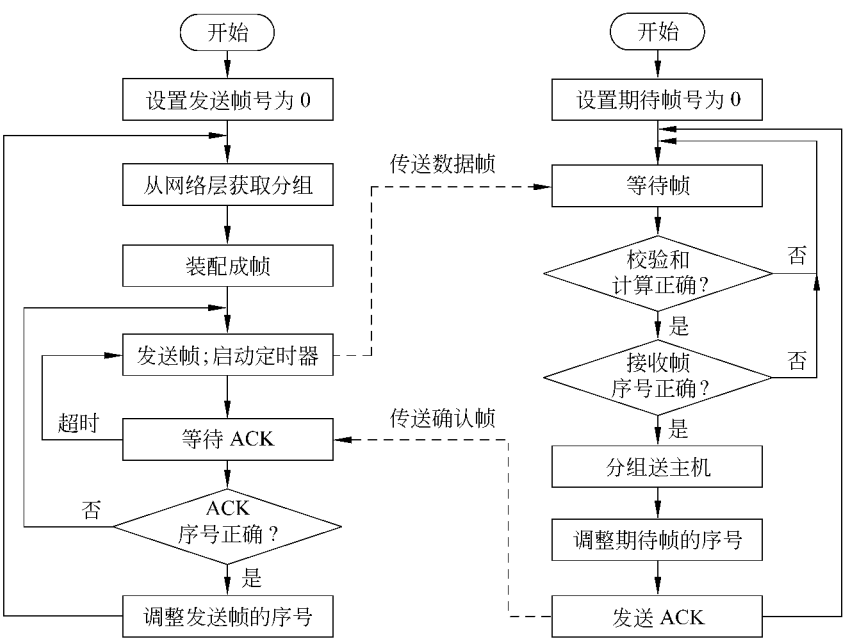


图 3-9 停等协议的流程图

接收方获得了帧后，先计算其校验和，实现差错检测。如果收到的帧正是接收方期望的，则给发送方返回确认帧，并上交分组给网络层；否则丢弃该帧，继续等待，或者发回一个否定帧（对于选择重传协议）。

为了说明停等协议的具体原理，下面分别讨论几种可能的数据传输现象，如图 3-10 所示。

(1) 无差错的理想情况

如图 3-10(a)所示，指主机 A 到主机 B 的传输信道没有差错。接收方的缓存只需要

装下一个数据帧，收发双方就能够实现很好的传输同步。

实际的传输信道是不理想的，差错不可避免，可能出现数据错误或丢失的情况。

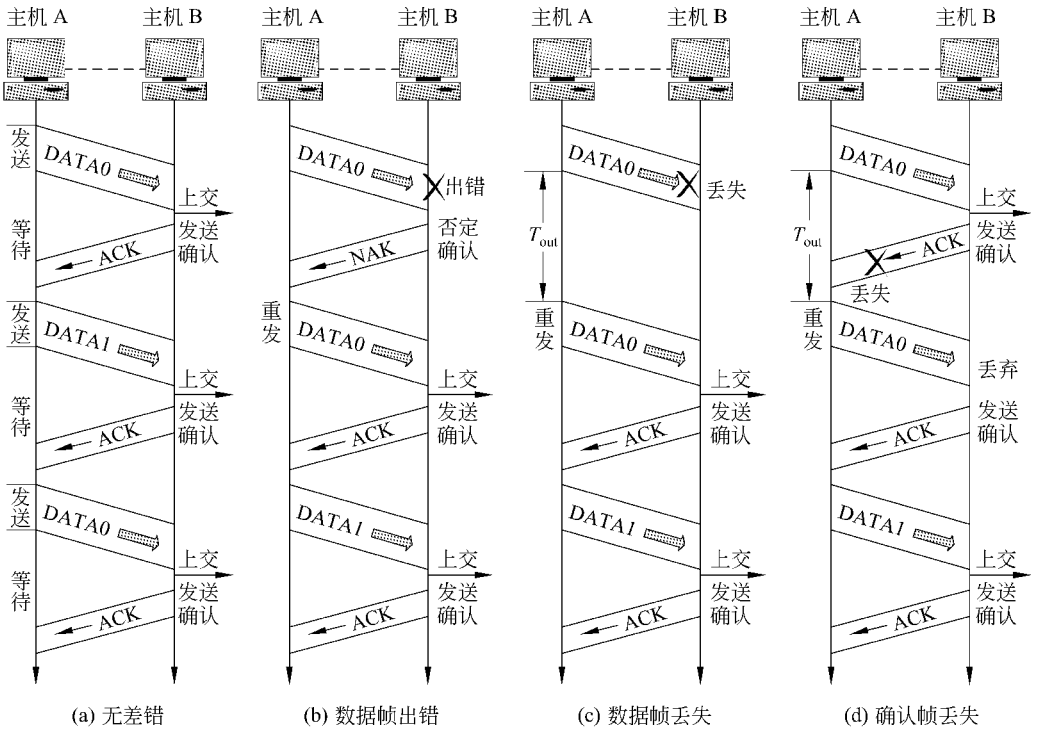


图 3-10　在链路上传输帧的情况

(2) 数据帧传输出错情况

如图 3-10(b)所示，主机 A 向主机 B 发送一个数据帧 DATA0，但在传输中出现了差错。如果该帧的结构仍然完整，则接收方能够识别此帧，并进行差错校验。一旦判断此帧是有差错的数据帧，就丢弃该帧，并向对方发回一个否定的帧(Negative Acknowledgement，NAK)，要求对方对 NAK 中指定的帧进行重传。

(3) 数据帧丢失情况

如图 3-10(c)所示，主机 A 发往主机 B 的数据帧在传输中丢失，使得主机 B 始终处于等待状态。此时，主机 A 也在一直等待确认的到来，这样就会出现死锁现象。为了解决这个问题，需要引入定时器，设置重发时间 T_{out}。每当一个数据帧发出之后，就立即启动一个定时器。如果定时器超时了，主机 A 仍然没有收到主机 B 的应答，则重发该帧。这种方法称为超时重发。如果连续多次重传都出现差错，重传超过了一定次数(如 16 次)，就停止发送，向上一级报告故障情况。

(4) 确认帧丢失情况

如图 3-10(d)所示，主机 B 发送送的确认帧在传输之中丢失。主机 A 因收不到该确认帧，执行超时重发。结果是主机 B 收到了一个重复帧。为了分清是新帧还是重复帧，需要对每个帧都设置序号，以示区别。因此，如果两个序号相同，就认为是重复帧。

可见，使用以上方法可以避免帧的重复和丢失，实现了一定的差错控制功能。而接收方不仅控制了发送确认帧 ACK 的时间(不超过重发时间)，还完成了对流量的控制。

停等协议的优点是控制比较简单，但由于一次只能发送一帧，所以在信号传播过程中发送方必须处于等待状态，这对于短信道来说是合适的。但对于长信道而言，效率很低。下面分析停等协议的传输效率，具体过程如图 3-11 所示。

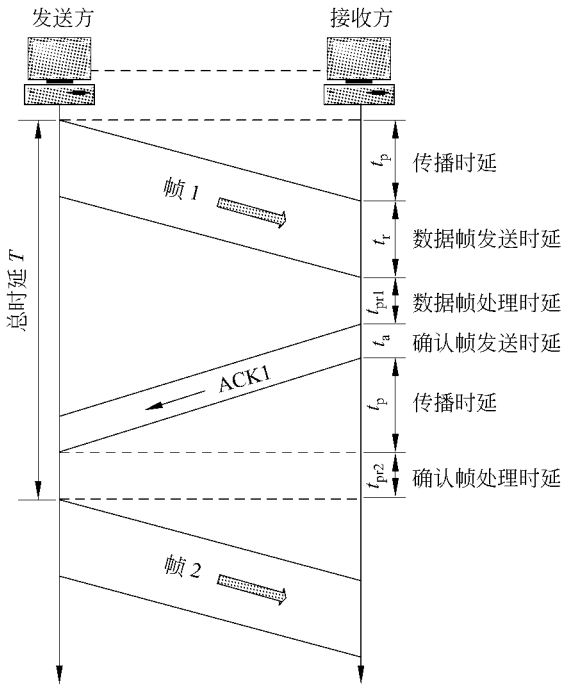


图 3-11 停等协议的信道利用率计算示意图

假设在传输过程中没有数据帧的差错发生，则总时延为：

$$T = t_p + t_f + t_{pr1} + t_a + t_p + t_{pr2} \tag{3-1}$$

进一步假设计算机对数据帧的确认帧的处理时间可以忽略不计。同时，由于确认帧比数据帧小得多，其传输时延也忽略，则有：

$$T \approx t_f + 2t_p \tag{3-2}$$

在无差错的数据链路中，数据传输效率 U 表示为：

$$U = \frac{t_f}{t_f + 2t_p} \tag{3-3}$$

可见，如果在一个总时延内能够连续发送多个数据帧，就会使传输效率成倍增加。

【例 3-4】 已知信道速率为 8kbps，传播时延为 20ms，确认帧长度和处理时间均可忽略。如果采用停等协议，请问帧长是多少时才能使信道利用率至少达到 50%？

解析：已知 t_p=20ms。设帧长为 L(b)，则有：t_f=(L/8)ms。由式(3-3)可得：

$$\frac{t_f}{t_f + 2t_p} \geqslant 50\%$$

当 $t_f \geqslant$40ms 时，不等式成立。因此，帧长 $L \geqslant$320(b)。