第 3 章　数据链路层

链路和数据链路意义不同，前者是指一条无源的点对点的物理线路段，中间没有任何其他的交换节点，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，还必须有一些必要的规程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节点（路由器 1），再经过路由器 2 和路由器 3 两个节点，达到目的主机 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 连接到网络可能采用电话网或局域网等接入方式。当然，从实际传输上看，是经过物理层由通信线路到达邻近节点的。

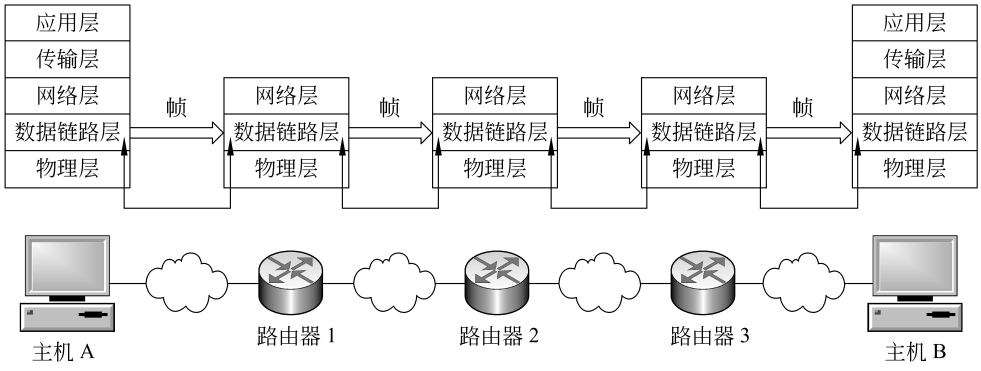


图 3-1　数据链路层传输原理示意

3.1　数据链路层的功能

3.1.1　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，所提供的基本服务有以下三种。

（1）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进行确认。如果由于线路噪声而造成帧丢失，数据链路层不负责重发，留待上层完成。因此，其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。

（2）有确认的无连接服务。这是在上述服务中引入了确认功能。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，则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。

（3）有确认的面向连接服务。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采用这种服务。

3.1.2 组帧

来自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，在接收端即使在物理层传输中出现了错误，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加上帧首部和帧尾部的长度。

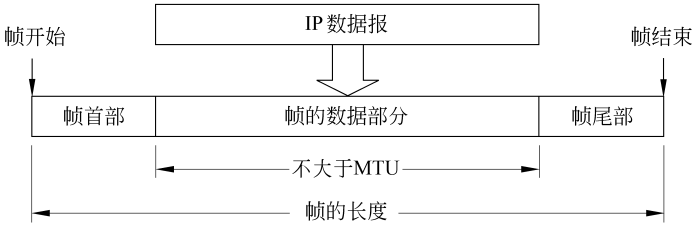


图 3-2 帧的结构模型

3.1.3 差错控制

当一帧到达目的地时，需要校验其正确性。这需要通过差错控制码产生的校验和确定，首先在发送端计算后连同数据一起发送，在接收端进行检错处理。如果是正确的，则数据部分向网络层传送；否则进行丢弃处理。

3.1.4 流量控制

流量控制是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，则会有许多帧丢失。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】 在数据链路层应根据什么原则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；但若使用以太网，则数据链路层使用的是无连接服务。

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，可以是以字节为单位组成帧的各部分字段，称为面向字节的组帧方式；也可以是以任意比特组合成帧的，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，是利用帧首部中的一个域指定该帧中的字节数，其原理如图 3-3 所示。

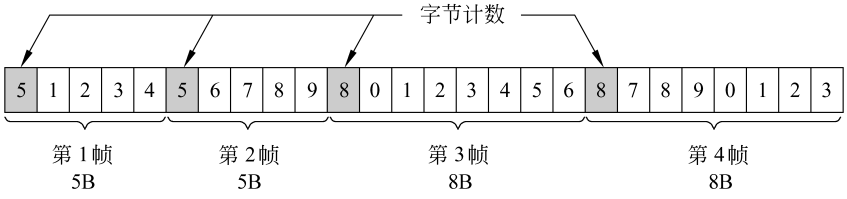


图 3-3　字节计数法示例

图 3-3 标识了 4 个数据帧的帧格式，它们的大小依次为 5B、5B、8B、8B。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，从而可以确定帧的终止位置。

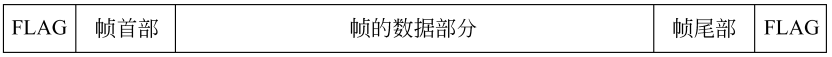
这种方法最大的问题在于如果标识帧大小的字段出错，即失去了帧边界划分的依据，将造成灾难性的后果。如第 2 帧中的计数字节由 5 变为 7，则接收方就会失去帧同步的可能，从而不可能再找到下一帧正确的起始位置。由于第 2 帧的校验和出现了错误，所以即使接收方给发送方请求重传都无济于事。因此，这种字节计数法目前已很少使用。

3.2.2　字符填充法

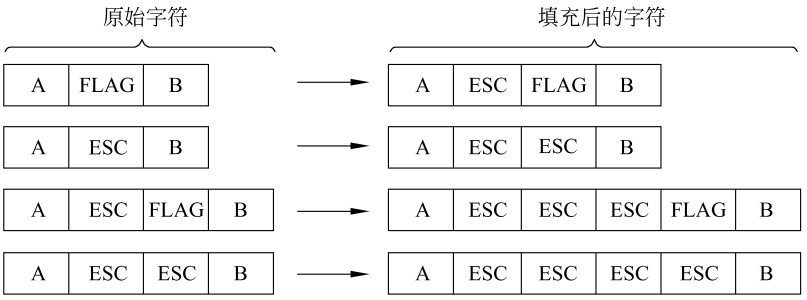
该同步方法是用一些特定的字符作为一个帧的开始和结束标志，同时也采用某些特定的字符作为传输过程中用到的控制字符。在过去，开始和结束字节并不相同，但在最近几年，绝大多数协议倾向于使用相同的字节，称为标志字节，如图 3-4(a)中的 FLAG 所示。因此，接收方如果丢失了同步，也只需要搜索标志字节就能找到当前帧的结束位置。两个连续的标志字节代表了当前帧的结束和下一帧的开始。

如果标志字节出现在数据中，则发送方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。进一步地，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 4 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 16 位字符。所以需要开发新的技术，以便允许任意长度的字符。



(a) 有标志字符作为分界的帧



(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)标志一帧的开始和结束，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，每当数据中遇到 5 个连续的比特 1 时，会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中，每当数据中收到 5 个连续的 1 且其后是 0 时，会自动删除该 0 比特。其工作原理如图 3-5 所示，如果要传输的数据帧为 011011111111011111110010，则采用零比特填充后，在网络中传送时表示为 011011111011011111010010。

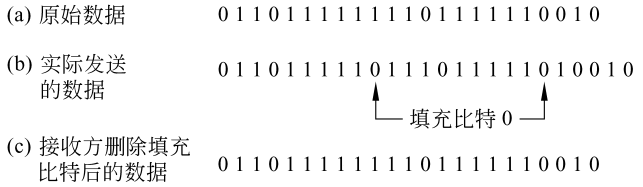


图 3-5　零比特填充法示例

零比特填充帧同步方式很容易由硬件实现，性能优于字符填充方式。所有面向比特的同步控制协议采用统一的帧格式，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

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例的，可以借用这些违例编码序列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。违例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，目前较普遍使用的帧同步法是零比特填充法和违例编码法。

【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，请给出当使用下面的成帧方法时所对应的位序列(用二进制表达)。

(1) 字节计数。

(2) 包含字节填充的标志字节。

(3) 包含零比特填充的开始和结束标志。

解析：

(1) 有 4 个字符和 1 个标识符，二进制表示为 00000101。按照字符计数，这 5 个字符应表示为

00000101 01000111 11100011 11100000 01111110

(2) 首尾各加上一个 FLAG 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，对应的二进制编码为

01111110 01000111 11100011 11100000 11100000 11100000 01111110 01111110

(3) 采用特定的比特模式(01111110)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

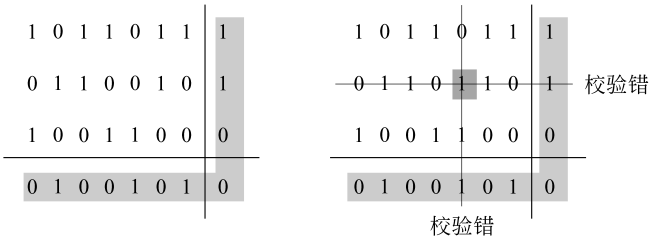
理想的通信系统在现实中是不存在的，信息传输过程中总会出现差错。差错控制是指采用编码技术，在通信过程中发现并检测差错，或对差错进行纠正，从而将差错控制在尽可能小的范围内。能检测差错的编码称为检错码，如奇偶校验码、循环冗余校验码以及校验和；能检测并纠正错误的编码称为纠错码，如汉明码。纠错码的实现过程复杂，在一般的通信场合不适宜采用；检错码的实现较为容易，能够通过重传机制获得正确的帧，因此在网络中广泛使用。

3.3.1　奇偶校验码

奇偶校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，作为奇偶校验位，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。

奇偶校验只能检测出奇数个位发生的错误，校验能力低，不适用于块数据的传输，需要采用垂直水平奇偶校验码(也称为纵横奇偶校验或方阵码)，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校验位，每列的最低位是垂直奇偶校验位。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、奇数个错和大部分偶数个错，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，发现第 2 行和第 5 列的奇偶校验出错，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使用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，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 $n=k+r$ 位的编码即为 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 $G(x)$实现，用 $G(x)$除以 $x^rM(x)$得到的余式就是 $R(x)$。

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

注意，这里的加法和除法都基于模 2 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。

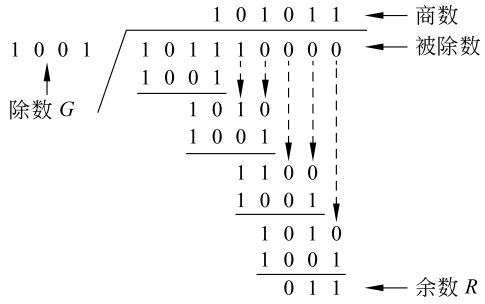


图 3-7　CRC 计算示例

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

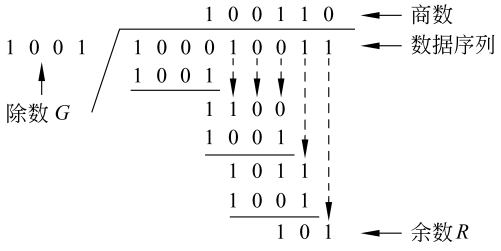


图 3-8　CRC 校验接收的数据

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 $G(x)$ 主要见如下。

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。校验和的校验过程计算简单，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。

3.3.3 汉明码

汉明码由 Richard Hamming 于 1950 年提出，是一种纠错码。其指导思想是，在被校验的数据中，增加几位校验位。当某一数据位出错时，引起几位校验位的值改变，不同代码位出错，得到不同的校验结果（即非法编码）。这样，不仅可发现错误，还能知道错误的位置，进而达到纠正错误的目的。

它是利用在信息位为 k 位，增加 r 位冗余位，构成一个 $n=k+r$ 位的码字，然后用 r 个监督关系式产生的 r 个校正因子区分无错和在码字中的 n 个不同位置的一位错。它必须满足关系式 $2^r \geqslant k+r+1$ 或 $2^r \geqslant n+1$。

汉明码的编码效率为 $R=k/(k+r)$（k 为信息位位数，r 为增加冗余位位数）。

1. 码距

码距是编码系统中两个任意合法码之间的最少二进制位数的差异。可见，ASCII 的码距为 1，奇偶校验码的码距为 2。码距公式为

$$L-1=D+C \tag{3-1}$$

式(3-1)中，L 为码距；D 是可检测的错误位数；C 为可纠正的错误位数。若能在数据码中增加几个校验位，将数据代码的码距均匀拉开，把每个二进制位分配在几个奇偶校验组中，则当某位出错后，会引起几个校验位的值的变化，这样不仅能查错，还能纠错。

为了检测出 d 个比特的错误，需要使用汉明距离为 $d+1$ 的编码。例如：数据后加奇偶校验位，编码后的汉明距离为 2，能检测 1b 的错误。

为了纠正 d 个比特的错误，必须用汉明距离为 $2d+1$ 的编码。例如：一个编码集只有 4 个有效码字：000000，000111，111000，111111。

其汉明距离为 3。

如果接收端收到码字 010111 时，判断是由哪个码字错来的？以上 4 个码字分别错 4、1、5 和 2，最有可能（概率最大）是第 2 个：$d=1$。

2. 有效信息位与校验位的关系

校验的位数与有效信息位的位数有关。若被校数据是 k 位、校验位是 r 位，校验位共有 2^r 个状态。在这 2^r 个状态中，应有 k 个状态表示 k 个数据位中是哪位出错。考虑到校验位本身也可能出错，因此有 r 个状态分别表示不同校验位出错，另外还有一个状态表示被校数据位和校验位都不出错。因此，为查出每位错，必须满足：

$$2^r \geqslant k+r+1 \tag{3-2}$$

例如：当 $r=4$ 时，$2^4 \geqslant k+4+1$，即 $16 \geqslant k+5$。于是，k 最大为 11，最小为 5。

因此，当被校数据为 8 位时，r 取 4；被校数据为 16 位时，r 取 5。

3. 编码规则

若被校数据为 D8D7D6D5D4D3D2D1，校验位取 4 位 P1、P2、P3、P4，形成汉明码 H12H11H10H9H8H7H6H5H4H3H2H1。

规则 1：规定校验位 Pi 放在汉明位号为 2^{i-1} 的位置上，则 P1 在 H1 位、P2 在 H2 位、P3 在 H4 位、P4 在 H8 位，其余的用信息数据从高到低插入。传送的数据流为

D8D7D6D5P4D4D3D2P3D1P2P1。

4. 确定校验位的取值

规则 2：被校数据位的汉明位号等于校验该数据位的各校验位汉明位号之和。另外，校验位不需要再被校验。

各数据位与校验位的关系：

D1 放在 H3 上，由 P2P1 校验，满足 3＝2＋1。

D2 放在 H5 上，由 P3P1 校验，满足 5＝4＋1(P3 的汉明位号是 4)。

D3 放在 H6 上，由 P3P2 校验，满足 6＝4＋2。

D4 放在 H7 上，由 P3P2P1 校验，满足 7＝4＋2＋1。

D5 放在 H9 上，由 P4P1 校验，满足 9＝8＋1。

D6 放在 H10 上，由 P4P2 校验，满足 10＝8＋2。

D7 放在 H11 上，由 P4P2P1 校验，满足 11＝8＋2＋1。

D8 放在 H12 上，由 P4P3 校验，满足 12＝8＋4。

注意：上述安排的目的是希望校验的结果能正确反映出错位的位号。

若用偶校验则可得到各位校验位的值：

$$P1：D1,D2,D4,D5,D7 \rightarrow P1 = D1 \oplus D2 \oplus D4 \oplus D5 \oplus D7$$

$$P2：D1,D3,D4,D6,D7 \rightarrow P2 = D1 \oplus D3 \oplus D4 \oplus D6 \oplus D7$$

$$P3：D2,D3,D4,D8 \rightarrow P3 = D2 \oplus D3 \oplus D4 \oplus D8$$

$$P4：D5,D6,D7,D8 \rightarrow P4 = D5 \oplus D6 \oplus D7 \oplus D8$$

若数据 D＝10100110(D8—D1)，则有：

$$P1 = D1 \oplus D2 \oplus D4 \oplus D5 \oplus D7 = 0 \oplus 1 \oplus 0 \oplus 0 \oplus 0 = 1$$

$$P2 = D1 \oplus D3 \oplus D4 \oplus D6 \oplus D7 = 0 \oplus 1 \oplus 0 \oplus 1 \oplus 0 = 0$$

$$P3 = D2 \oplus D3 \oplus D4 \oplus D8 = 1 \oplus 1 \oplus 0 \oplus 1 = 1$$

$$P4 = D5 \oplus D6 \oplus D7 \oplus D8 = 0 \oplus 1 \oplus 0 \oplus 1 = 0$$

5. 汉明校验值

$$C1 = P1 \oplus D1 \oplus D2 \oplus D4 \oplus D5 \oplus D7 = 1 \oplus 1 = 0$$

$$C2 = P2 \oplus D1 \oplus D3 \oplus D4 \oplus D6 \oplus D7 = 0 \oplus 0 = 0$$

$$C3 = P3 \oplus D2 \oplus D3 \oplus D4 \oplus D8 = 1 \oplus 1 = 0$$

$$C4 = P4 \oplus D5 \oplus D6 \oplus D7 \oplus D8 = 0 \oplus 0 = 0$$

6. 校验结论

(1) 若汉明校验值为全 0，即 C4C3C2C1＝0000，表示数据传送无错。

(2) 若汉明校验值 1 位出错，则校验位出错。

当 C4C3C2C1＝0001 时，H1 出错，即校验位 P1 出错。

当 C4C3C2C1＝0010 时，H2 出错，即校验位 P2 出错。

当 C4C3C2C1＝0100 时，H4 出错，即校验位 P3 出错。

当 C4C3C2C1＝1000 时，H8 出错，即校验位 P4 出错。

(3) 若汉明校验值中有 2 位或 2 位以上出错，则是被校验数据位出错。C4C3C2C1

的编码就是出错位的汉明位号。

C4C3C2C1＝0011(汉明位号 3)，H3 出错，即数据位 D1 出错。

C4C3C2C1＝0101 (汉明位号 5)，H5 出错，即数据位 D2 出错。

C4C3C2C1＝0110 (汉明位号 6)，H6 出错，即数据位 D3 出错。

C4C3C2C1＝0111 (汉明位号 7)，H7 出错，即数据位 D4 出错。

C4C3C2C1＝1001 (汉明位号 9)，H9 出错，即数据位 D5 出错。

C4C3C2C1＝1010 (汉明位号 10)，H10 出错，即数据位 D6 出错。

C4C3C2C1＝1011 (汉明位号 11)，H11 出错，即数据位 D7 出错。

C4C3C2C1＝1100 (汉明位号 12)，H12 出错，即数据位 D8 出错。

按上述方法进行汉明码编码后，假设接收到的码字是 101100111001，该如何校验和纠错呢？

已知接收的汉明码共 12 位，从中分离出校验位，得 P1＝1、P2＝0、P3＝1、P4＝0，数据位 D8 到 D1 的排列为 10110110。

接着通过译码公式求解校验结果：

C1＝ P1⊕D1⊕D2⊕D4⊕D5⊕D7＝1⊕0⊕1⊕0⊕1⊕0＝1

C2＝ P2⊕D1⊕D3⊕D4⊕D6⊕D7＝0⊕0⊕1⊕0⊕1⊕0＝0

C3＝ P3⊕D2⊕D3⊕D4⊕D8＝1⊕1⊕1⊕0⊕1＝0

C4＝ P4⊕D5⊕D6⊕D7⊕D8＝0⊕1⊕1⊕0⊕1＝1

则有 C4C3C2C1＝1001，结果非 0，说明接收的码字有错，且汉明位号 9 出错，即数据位 D5 出错。

因此，正确的数据是：10100110。

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。

流量控制用于协调链路中发送方和接收方之间的数据流量，以保证双方的数据发送和接收达到平衡。当来不及接收时，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由于网络中瞬间的大量数据对网络带来的冲击，保障用户的网络运行。

流量控制不仅可以在数据链路层上实现，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对象是不同的，数据链路层上控制的是网络中相邻节点之间的数据传输过程，网络层上控制的是网络源节点和目的节点之间的数据传输，而传输层上控制的是网络中不同节点内发送进程和接收进程之间的数据传输过程。

目前，通信节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，后者又包括了后退 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，发送方已将上层发来的分组装配成帧，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。

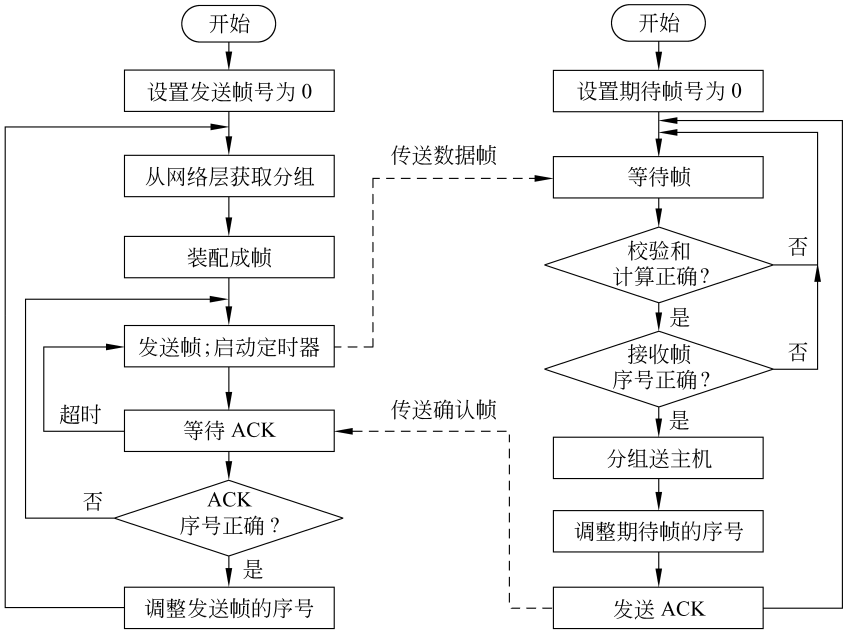


图 3-9　停等协议的工作流程

接收方获得帧后，先计算其校验和，实现差错检测。进一步地，如果正确的帧是接收方所期望的，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。

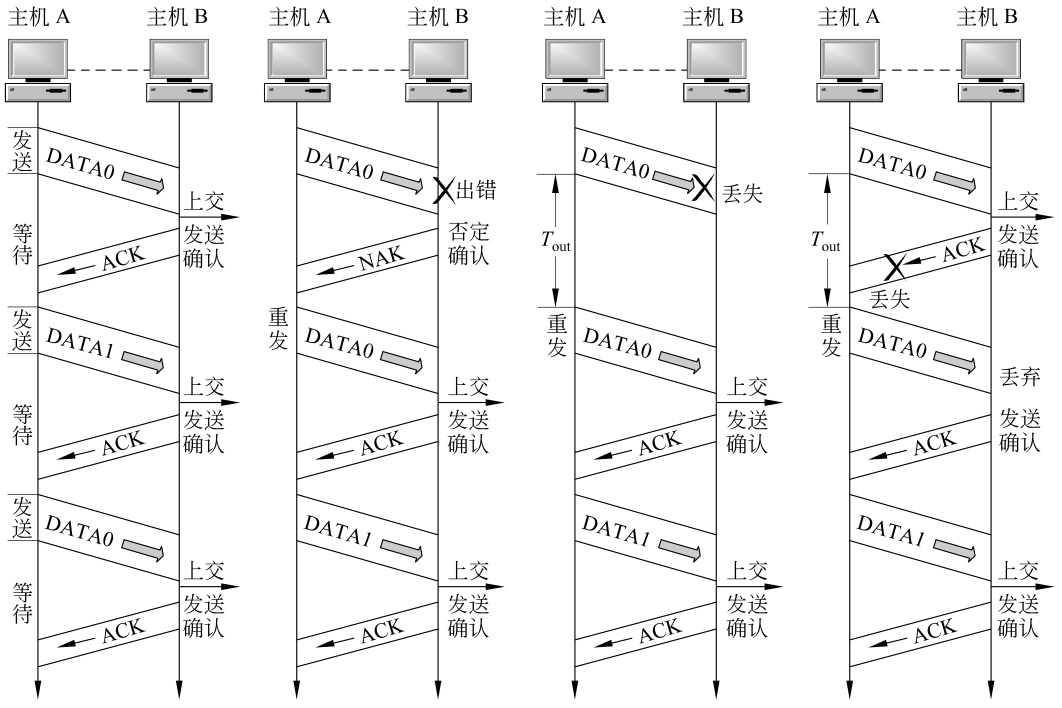
1. 无差错的理想情况

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，收发双方能够实现良好的传输同步。

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。一旦判断其是有差错的数据帧，则丢弃该帧，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。



(a) 无差错　(b) 数据帧出错　(c) 数据帧丢失　(d) 确认帧丢失

图 3-10　在链路上传输帧的情况

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，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。如果定时器超时后仍然没有收到主机 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，超过了一定次数(如 16 次)，则停止发送，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，确认帧在传输中丢失。主机 A 因为收不到该确认帧，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。而接收方控制发送确认帧 ACK 的时间(不超过重发时间)，还完成了流量控制功能。

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，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。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。

假设在传输过程中没有数据帧的差错发生，则总时延为

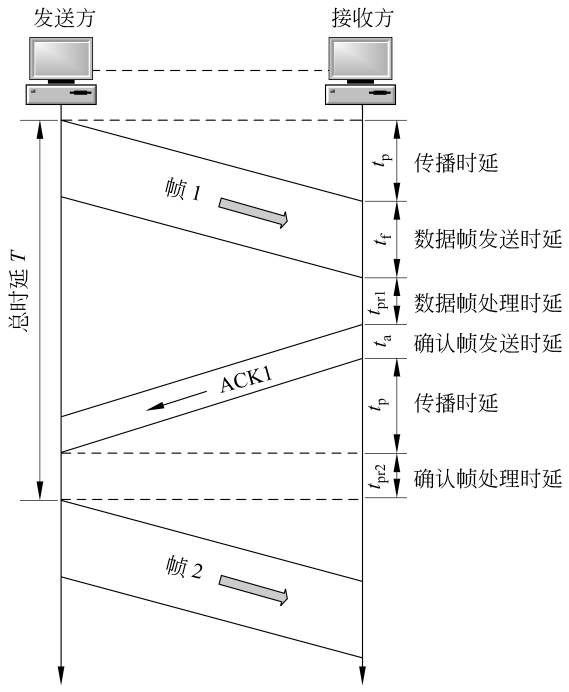


图 3-11　停等协议的信道利用率计算示意

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

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

$$T \approx t_f + 2t_p \tag{3-4}$$

在无差错的数据链路中，数据传输效率 U 表示为

$$U = \frac{t_f}{t_f + 2t_p} \tag{3-5}$$

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 8kb/s，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(3-5)可得

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

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

3.4.2　滑动窗口机制

滑动窗口机制是网络中控制流量最常用的技术方案，发送方不必等待接收方的应答就可以连续发送数据帧，但对发送方在收到确认帧之前可以发送的数据帧数目加以限制。如果希望发送方停止发送数据，就停止发送确认信息，使发送方发送缓冲区中未被确认的

数据帧很快达到极限而停止发送新的数据帧，直到再次收到接收方的确认帧。

所有帧都进行统一编号，既要正确区分不同的帧，又要减少控制开销，提高传输效率。如果用 n 比特表示帧的序号，则帧的序号范围是 0～(2^n-1)。例如，在传播时延较小的链路上，通常设置 $n=3$，序号空间为 0～7，共 8 个序号，称为“模 8”编码。待发送完序号为 0～7 的帧后，下一帧又从 0 开始。而在传播时延较大的链路上，如卫星链路上，通常使用 $n=7$ 的编码方案，序号空间为 0～127，共 128 个序号，称为“模 128”编码。

1. 发送窗口

发送缓冲区由两部分组成：一是已经发送出去、但未接收到确认的数据帧，保留这部分的目的是预备其中某些帧需要重发；二是还未被发送的数据帧，也是发送方能够继续发送的数据帧。

发送方把未得到确认而允许连续发送的一组帧的序号集合称为发送窗口，即允许发送的帧的序号表。

图 3-12 是发送窗口的滑动控制示意图，帧的序号范围是 0～7。

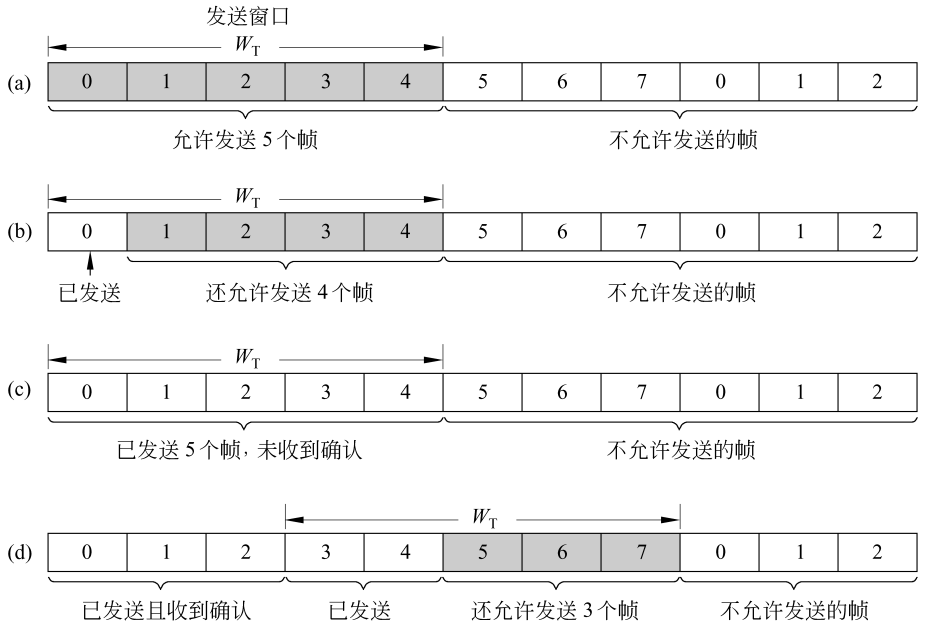


图 3-12　发送窗口的控制过程

发送方未得到确认而允许连续发送的帧的最大数目称为发送窗口尺寸。发送方每发送一个新帧，都要先检查该帧序号是否落在发送窗口内。发送方最早发送但还未收到确认的帧序号，称为发送窗口的后沿；发送窗口后沿加上窗口尺寸再减 1，称为发送窗口的前沿，表示发送方在收到确认前最后允许发送的帧序号。如图 3-12(a)所示的窗口尺寸为 5，后沿为 0，则前沿为 0＋5－1＝4。

发送方每发送一个新的数据帧，窗口的后沿就向前滑动一个序号，窗口尺寸减 1，即可以连续发送的帧数减 1。而当发送窗口尺寸为 0 时，窗口关闭。如图 3-12(b)所示，窗

口尺寸由 5 减小到 4；而在图 3-12(c)中，窗口尺寸为 0，不可发送。

在收到了发送窗口后沿所对应的帧的确认应答后，就将发送窗口前沿向前滑动一个序号，且窗口尺寸加 1，表示可以继续发送的帧数加 1，并从发送缓冲区中将已确认的数据帧的副本删除。如图 3-12(d)所示，窗口尺寸变为 3，可以发送 3 个新的数据帧。

因此，接收方就可以通过发送确认帧控制发送窗口的滑动，从而达到流量控制的目的。

2. 接收窗口

同样，在接收方将允许接收的一组帧的序号集合称为接收窗口，即允许接收的帧的序号表。接收方最多允许接收的帧数目称为接收窗口尺寸。接收窗口的上下界分别称为接收窗口的前沿和后沿。如图 3-13 所示，接收窗口尺寸为 1。图 3-13(b)中的前沿和后沿都是 1。

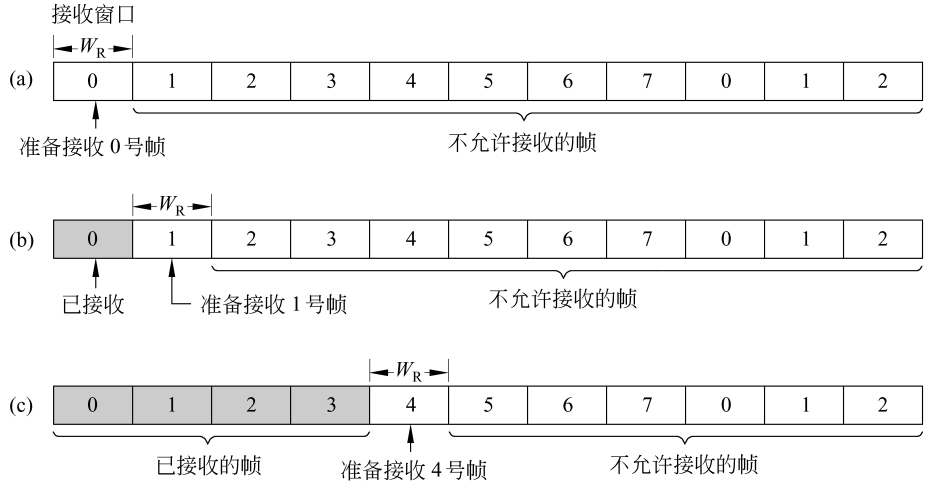


图 3-13　接收窗口的控制过程

接收方每收到一帧，都要判断该帧是否落在接收窗口之内。如果帧的序号正好等于接收窗口的后沿，且经过检验正确，则将该帧的数据部分上交给网络层，并向发送方返回一个确认帧，同时使接收窗口向前滑动一个序号。

如果收到的帧序号落在接收窗口之外，则直接丢弃该帧，不进行其他处理。

如果收到的帧序号落在接收窗口但不等于接收窗口后沿，则该帧经过检验正确后，暂时保留在接收缓冲区中。然后，继续等待序号为接收窗口后沿的帧，直到正确地收到后，才将其连同先前保留在接收缓冲区中的帧按顺序送交上层，并发出应答，同时向前滑动接收窗口。

下面，以饼图方式表示滑动窗口协议的基本原理，其发送窗口尺寸为 2、接收窗口尺寸为 1，如图 3-14 所示，从初态开始，展示了从发送 0 号帧到接收 1 号确认帧的窗口滑动过程。从中可以清晰地看到，在发送进行中，无论是发送窗口还是接收窗口，其位置都一直在顺时针转动。

在滑动窗口机制中，既要发挥流量控制的作用，又要尽可能地提高传输信道的利用率。如果发送窗口太小，则会造成传输信道的浪费；但如果发送窗口太大，又起不到流量控制的作用。理想的情况是，当刚刚发完发送窗口中允许发送的最后一帧时，就收到了窗

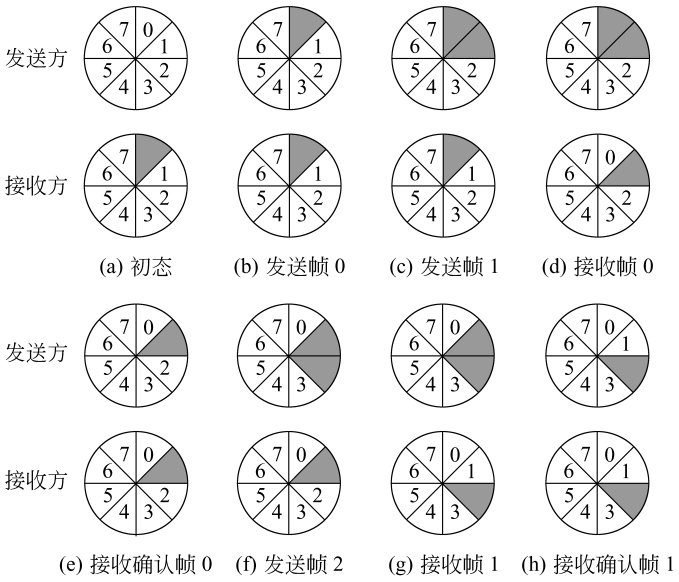


图 3-14　滑动窗口协议的滑动示意

口中最先发送帧的确认。这样，发送窗口向前滑动，又可以继续发送；同时，信道的利用率几乎没有浪费。

在发送窗口大于 1 的滑动窗口协议中，如果传输中出现差错，协议则会自动要求发送端重传出错的数据帧，这种滑动窗口控制机制称为自动重传请求（Automatic Repeat reQuest，ARQ），或称为自动请求重传。

3.4.3　后退 *N* 帧协议

后退 N 帧协议也称为连续 ARQ 协议，是指发送方可以连续发送多个数据帧，而接收方只能按顺序接收指定序号的帧。如果该帧被正确接收，则接收窗口向前滑动一帧，并开始接收下一帧。但是，如果某个发送的数据帧出错或丢失，接收方无法接收到该帧，则其后到达的 N 帧也都只能丢弃。所以，等到发送方定时器超时，就必须重发这一帧及其后面的所有帧，因此称这种协议为后退 N 帧（Go-Back-N）协议，简称为 GBN 协议。

图 3-15 是后退 N 帧协议的示意图，当 2 号数据帧出错并被丢弃后，后面到达的 3～8 号帧都被丢弃，也不发应答。等到发送方超时后，从第 2 帧开始全部重发直到确认后，才能继续发送新的数据帧。可见，本协议一方面因为能够连续发送多个数据帧而提高了效率，另一方面却因为后退 N 帧的重传方式而降低了效率。所以，如果信道的传输质量很差且误码率较大时，后退 N 帧协议不一定优于停等协议。

【例 3-5】 对于后退 N 帧协议，其接收窗口尺寸固定为 1，即 $W_R=1$。若用 n 个比特对数据帧进行编码，试证明：只有当发送窗口尺寸 $W_T \leqslant 2^n-1$ 时，本协议才能正确运行。

证明：为讨论方便，取 $n=3$，则数据帧的序号范围为 0～7，最大窗口尺寸为 8。

采用反证法，令 $W_T=8$，即发送窗口尺寸为 8 时的协议工作情况。

考虑以下极端状态下的传输过程（如图 3-16 所示）。

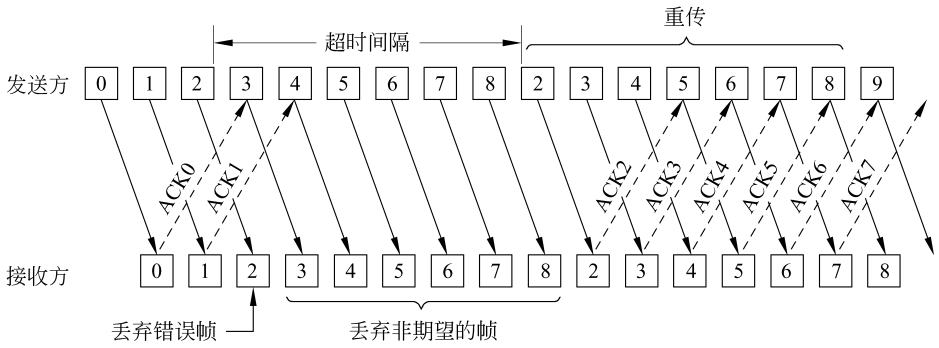


图 3-15　后退 N 帧协议

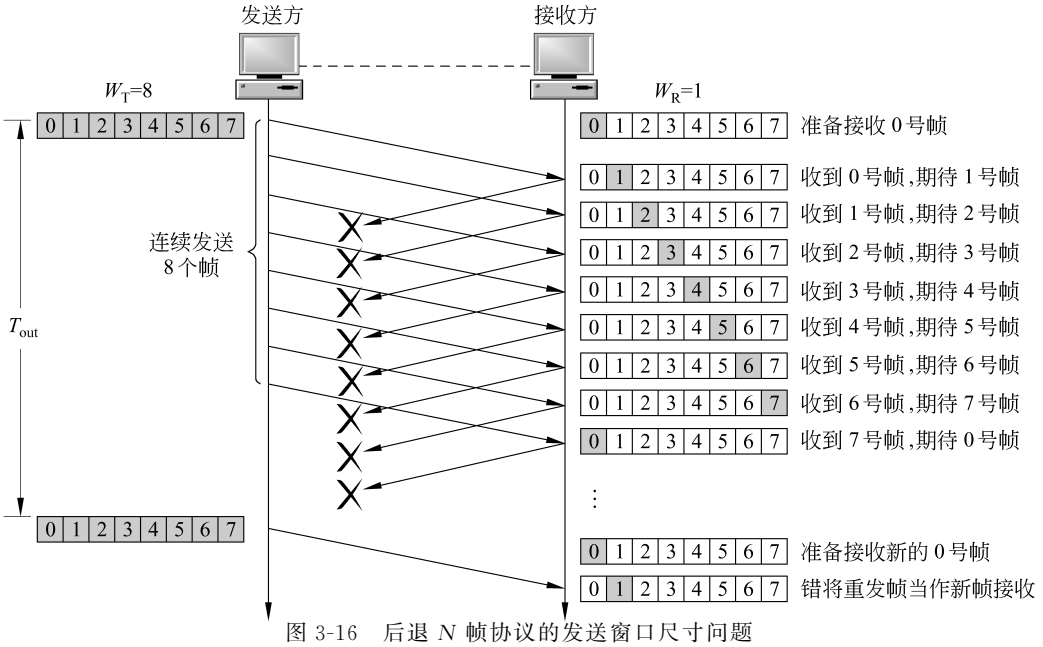


图 3-16　后退 N 帧协议的发送窗口尺寸问题

(1) 初始状态,接收窗口位于 0 号时,发送方开始发送 0～7 号的数据帧。

(2) 接收方在正确收到了 0～7 号共 8 个帧后,立即上交到网络层,返回对所有帧的确认后,窗口滑动到 0 号,并准备接收新一轮的 0～7 号数据帧。

(3) 所有的应答帧都丢失。

(4) 发送方超时,重发 0～7 号数据帧。

(5) 接收方按顺序滑动,先后收到 0～7 号数据帧,并错误地将其当作新的数据帧送交网络层,于是协议失效。

因此,发送窗口尺寸必须小于 8,即 $W_T<8$。

那么,$W_T=7$ 时协议能否正确运行呢?

再次考虑上述极端状态的传输过程。

（1）初始状态，接收窗口位于 0 号时，发送方开始发送 0～6 号的数据帧。

（2）接收方在正确收到了 0～6 号共 7 个帧后，立即上交到网络层，返回对所有帧的确认后，窗口滑动到 7 号，并准备接收新一轮的 7 号和 0～5 号数据帧。

（3）所有的应答帧都丢失。

（4）发送方超时，重发 0～6 号数据帧。

（5）接收方先后收到 0～6 号重发的数据帧，由于不是期待的 7 号，所以直接丢弃，然后重新发送对 0～6 号帧的应答，表示希望接收序号从 7 开始的帧。

（6）发送方收到应答后，发送序号为 7 号和 0～5 号的新数据帧。这样就保证了协议的正常实现。

因此，有 $W_T \leqslant 7$ 成立。证毕。

3.4.4 选择重传协议

在后退 N 帧协议的重发方案中，可能将已经正确传送到目的方的帧再次重发，这显然是一种浪费。另一种效率更高的策略是，当接收方发现某帧出错后，其后继续送来的正确帧虽然不能立即递交给接收方的高层，但接收方仍可收下来并存放在一个缓冲区中。同时，要求发送方只重传出错的数据帧或者定时器超时的数据帧。一旦收到正确的重传数据帧后，就可以与原来已存于缓冲区中的其余帧一并按正确的顺序递交高层。这就是选择重传(Selective Repeat)协议，简称为 SP 协议。

选择重传协议的工作原理如图 3-17 所示，其发送窗口尺寸和接收窗口尺寸都大于 1。接收方收到出错的 2 号帧后，立即丢弃该帧并返回一个否认帧 NAK2，要求发送方选择重发 2 号帧。随后正确接收的 3～4 号数据帧被保存在缓冲区中。在接收到正确的 2 号数据帧后，将 2～4 号数据帧一起提交给高层，且返回 ACK4。

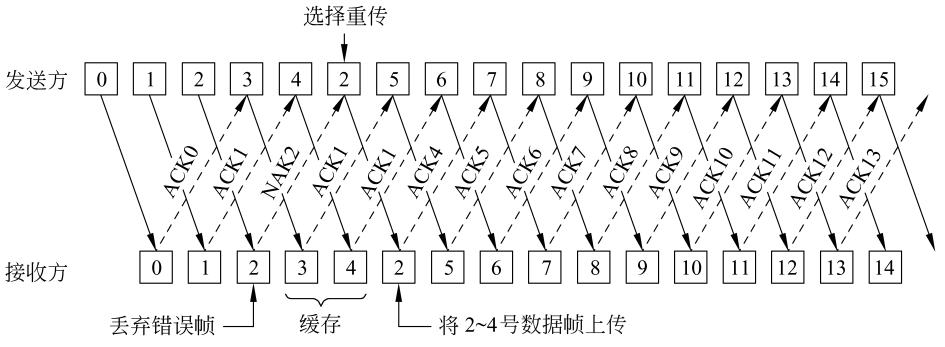


图 3-17 选择重传协议的工作原理

选择重传协议具有两个明显的特点。

（1）采用累计确认技术。接收方收到了连续且正确的数据帧后，为了节省链路带宽资源，并不对每一个数据帧返回确认，而是只返回对最高序号的帧进行确认。发送方收到这个确认帧号后，认为该帧号之前的其他帧都已经被正确接收。

（2）否认帧。在收到的数据帧经校验计算发现有错后，接收方会返回一个否定帧，以

便通知发送方重发该帧。这样，发送方如果收到了否认帧，则可以在定时器的超时到来之前，就能够立即重发数据帧。

显然，选择重传协议减少了浪费，但要求接收方有足够大的缓冲区空间，这在许多情况下是不够经济的，因此，该协议目前还没有后退 N 帧协议使用得广泛。随着存储技术的发展，选择重传协议应该会得到重视，如在传输层的 TCP 协议中，就使用了类似选择重传的传输控制方法。

在选择重传协议中，发送窗口尺寸和接收窗口尺寸往往相同，其最大窗口尺寸为 2^{n-1}（n 为帧的编码序号）。

【例 3-6】 假设帧的序号长度为 3 位，发送窗口尺寸和接收窗口尺寸都是 2，采用选择重传协议发送数据帧。请画出由初始状态出发，下列事件依次发生时的发送窗口和接收窗口示意图：发送帧 0、发送帧 1、接收帧 0、接收确认帧 0、发送帧 2、接收帧 2、重发帧 1、接收帧 1、接收确认帧 2。

解析：采用饼图描述这个过程，如图 3-18 所示。

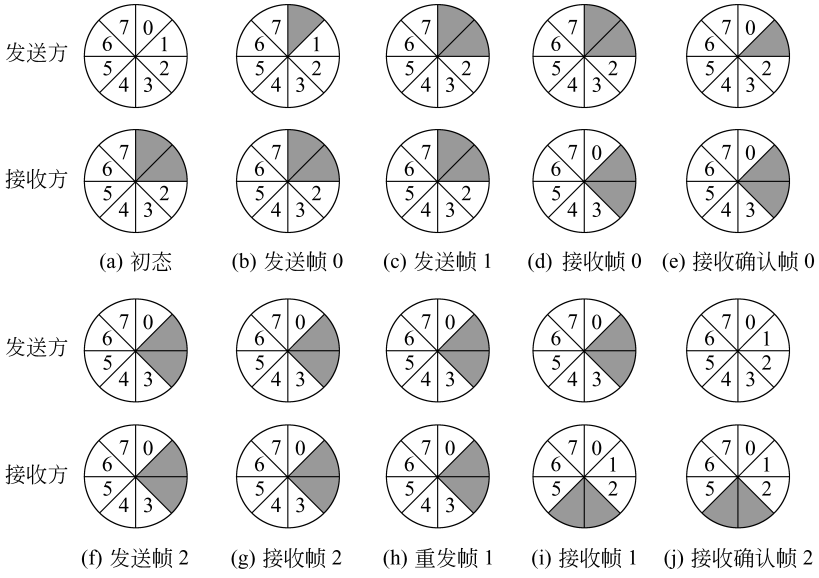


图 3-18　选择重传协议的滑动窗口控制示例

3.5　高级数据链路控制（HDLC）协议

数据链路层协议基本可以分为两类：面向字符型和面向比特型。面向字符型协议出现得最早，其特点是利用已经定义好的一种字符编码的一个子集执行通信控制功能，典型的有 IBM 公司的 BSC 协议。但是，这类协议具有明显的缺点，一是使用不同字符集的计算机无法利用该协议进行通信；二是控制字符的编码不能出现在用户数据中；三是控制功能的扩展性差，每增加一项功能就需要添加和定义相应的控制字符。而面向比特型协议可以克服这些缺点。尽管如此，目前面向字符型的点到点协议（Point-to-Point Protocol，

PPP)在 Internet 中仍然得到了广泛的应用。

1974 年，IBM 公司推出了著名的系统网络体系结构(System Network Architecture，SNA)，用于 IBM 公司的大型机(ES/9000 和 S/390 等)和中型机(AS/400)之间的联网。SNA 中的数据链路层协议采用了面向比特的 SDLC(Synchronous Data Link Control)协议。面向比特指帧首部中的控制信息是由比特组合而成，而不是由几种特殊的控制字符而定，因此，控制信息可以具有很多功能，使 SDLC 协议能够满足不同用户的需求。

此后，ISO 把 SDLC 修改为高级数据链路控制 HDLC(High-level Data Link Control)，作为国际标准 ISO 3309。相应地，我国的国家标准原为 GB7496，现已被 GB/T 7421—2008 代替，于 2009 年 1 月 1 日起实施。原 CCITT 则将 HDLC 再修改后并称为链路接入规程(Link Access Procedure，LAP)，并作为 X.25 建议书的一部分。不久，HDLC 的新版本又把 LAP 修改为 LAPB，称为链路接入规程(平衡型)。

下面详细介绍 HDLC 的特点、帧结构和帧类型。

3.5.1 HDLC 的基本特点

HDLC 定义了三种类型的站，两种配置和三种数据传输模式。

1. 站

三种类型的站分别是主站、从站和复合站。主站负责控制链路的操作，其发出的帧称为命令帧。从站受控于主站，向主站发出响应帧。复合站是指具有主站和从站的双重功能，可以发出命令帧和响应帧。

2. 两种配置

两种配置包括平衡配置和非平衡配置。在平衡配置中，每一端都是复合站，所以这种配置只能工作于点对点方式中。而非平衡配置可用于点到点链路或多点链路，由一个主站和一个从站或多个从站组成。

3. 三种数据传输模式

(1) 正常响应模式(NRM)，属于非平衡配置，主站向从站传输数据，从站进行响应。但是，从站只有在收到主站的许可后，才能响应。

(2) 异步平衡模式(ABM)，属于平衡配置，每个复合站都可以发送、接收命令/响应帧。

(3) 异步响应模式(ARM)，属于非平衡配置，从站在没有接收到主站的允许时就可以发起数据传输，但主站仍负责全程的初始化和差错恢复等工作。

3.5.2 HDLC 的帧结构

所有面向比特的协议都使用如图 3-19 所示的帧结构。

1. 标志字段 F

每帧的首尾都采用 01111110(0x7E)作为边界。当连续传输一些帧时，前帧的结束标志 F 可以兼作为下一帧的起始标志。在组帧方式中，HDLC 规定采用 3.2.3 节介绍的比