出了一帧之后就等待肯定的或者否定的确认，那么，若由于硬件故障的原因而丢失了某一帧的话，则发送方就将永远等待下去了。

可以通过在数据链路层中引入定时器来解决这个问题，当发送方送出一帧时，通常还要启动一个定时器。该定时器的过期时间应该设置得足够长，以保证该帧在正常情况下能够到达目标方，并且在目标方进行处理，然后再将确认送回到发送方。一般情况下，在定时器到期之前，该帧将正确地接收到，并且确认报文也会送回来，这时定时器被取消。

然而，如果原始的帧或者确认报文丢失了，则定时器将被触发，从而警告发送方有一个潜在的问题存在。可以重新发送该帧，但多次重复发送可能会导致接收方将两次或者多次接收同一帧，并且多次将它传递给网络层。为了避免发生这样的情形，有必要为送出去的帧分配序列号，这样接收方能够区别原始帧和重传帧。

4. 流控制 flow control

数据链路层中另一个重要的设计问题是，当发送方在一台快速（或者负载较轻）的计算机上运行，而接收方在一台慢速（或者负载较重）的计算机上运行时，接收速率小于发送速率，发送方以很高的速度持续地往外发送帧，直到接收方完全被淹没。即使传输过程不会出错，但到了某一个点上的时候，接收方也将无法再处理持续到来的帧，从这时开始就要丢弃一些帧了。

常用的办法有两种：第一种是基于反馈的流控制（feedback-based flow control），接收方给发送方送回信息，允许它发送更多的数据，或者至少要告诉发送方它的情况；第二种方法是基于速率的流控制（rate-basd flow control），使用这种方法的协议有一种内置的机制，它限制了发送方传输数据的速率，而无需利用接收方的反馈信息。在本章中，将学习基于反馈的流控制方案，数据链路层从来不使用基于速率的流控制方案。

基于反馈的流控制方案有许多种，但是绝大多数使用了同样的基本原理。通常在没有得到接收方许可（隐式或者显式许可）之前，禁止发送方往外发帧。例如，建立连接时，接收方提示发送若干帧之后就禁止发送，等待许可之后再继续发送。

3.2 检错与纠错

不可靠信道：纠错
可靠信道：检错

电话系统有三个部分：交换机、局间干线和本地回路。目前前两部分几乎已经完全数字化，只有本地回路仍然是模拟的双绞线，由于替换掉这些双绞线需要花费大量的资金，所以在接下去的几年中本地回路还是双绞线。虽然在数字部分很少发生传输错误，但是，在本地回路上错误还很常见。而且，无线通信正在普及，它的错误率比光纤干线要高出几个数量级。在接下去的很多年中，传输错误将一直伴随着我们，我们必须要知道该如何处理传输错误。

由于物理过程而产生的错误，在有些介质（比如无线电波）上常常是突发性的连续多位，而不是单个的。突发性的错误与孤立的、单个位的错误相比，既有优点也有缺点。假设数据块的大小为 1000 位，每一位的错误率是 0.001。如果错误是独立的，则大多数数据块将包含一个错误。然而，如果错误是突发性的，发生一次就是连续 100 位，则平均而言，在 100 个数据块中只有 1 个或者 2 个数据块受到影响，突发性错误比单独的错误更加难以纠正。

突发差错 ← 冲击噪声
随机差错 ← 热噪声

1. 纠错码 error-correcting code

网络设计者已经研究出两种用于错误处理过程的基本策略：一是在每一个被发送的数据块中包含足够的冗余信息，以便接收方可以推断出被发送的数据中肯定有哪些内容；另一种方法也是包含一些冗余信息，但是这些信息只能让接收方推断出发生了错误，但推断不出发生了哪个错误，然后请求重传。前一种策略使用了纠错码(error-correcting code)，后一种策略使用了检错码(error-detecting code)，使用纠错码的技术通常也称为前向纠错(forward error correction)。数据后面追加的冗余码：帧检验序列FCS(Frame Check Sequence)

这里的每一项技术都有不同的适用环境，在高度可靠的信道，比如光纤，使用检错码是比较合理的做法，偶尔有错误发生时，只需重传整个数据块即可。但是在错误发生很频繁的信道上，比如无线链路，最好的做法是在每一个数据块中加入足够的冗余信息，以便接收方能够计算出原始的数据块是什么，而不是依靠重传来解决问题，因为重传的数据块本身也可能是错误的。

为了理解错误发生之后是如何被处理的，先看一看错误到底是什么样的。通常，一帧包含 m 个数据位(即报文)和 r 个冗余位(校验位)。假设总长度为 n，即 $n=m+r$。包含数据和校验位的 n 位单元通常也称为 n 位码字(code word)。

给定两个码字，例如 10001001 和 10110001，我们可以确定它们之间有多少位不同。在这个例子中，有 3 位不同。为了确定有多少位不同，只要对这两个码字进行异或(XOR)运算，然后计算出异或结果中 1 的个数，例如：

$$\begin{array}{r} 10001001 \\ 10110001 \\ \hline 00111000 \end{array}$$

两个码字中不相同的位的个数称为海明距离(Hamming distance)。其意义在于，如果两个码字的海明距离为 d，则需要 d 个 1 位错误才能将一个码字转变成另一个码字。

在大多数数据传输应用中，所有 2^m 种可能的数据报文都是合法的，但是，根据检验位的计算方法，并非所有 2^m 种可能的码字都被用到了。给定了计算校验位的算法以后，有可能构造出完整的合法码字列表，并且从这个列表中找到海明距离最小的两个码字。此距离是整个编码方案的海明距离。

一种编码方案的检错和纠错特性与它的海明距离有关，为了检测 d 个错误，需要一个距离为 $d+1$ 的编码方案，因为在这样的编码方案中，d 个 1 位错误不可能将一个有效码字改变成另一个有效码字。当接收方看到一个无效码字时，它就知道已经发生了传输错误。类似地，为了纠正 d 个错误，需要一个距离为 $2d+1$ 的编码方案，因为在这样的编码方案中，合法码字之间的距离足够远，因而即使发生了 d 位变化，则还是原来的码字离它最近，从而可以唯一确定原来的码字，达到纠错的目的。包括校验位

下面给出一个简单的检错编码例子。请考虑这样一个编码：在数据后面加上一个奇偶位(parity bit)。奇偶位是这样选择的：保证码字中“1”位的数目是偶数(或者奇数)。

例如，当 1011010 以偶数位发送时，后面加上一位变成了 10110100。如果是按奇数位发送，则 1011010 变成 10110101。只加上单个奇偶位的编码方案的距离为 2，因为任何 1 位错误所产生的码字，其奇偶位一定是错误的。因此，它可以用来检测单个错误。

(奇数个)

作为纠错编码的简单例子，请考虑下面只有 4 个有效码字的编码：

0000000000，0000011111，1111100000，1111111111

以上编码的距离为 5，这意味着它可以纠正 2 个错误。如果码字 0000000111 到达，则接收方知道原始的码字一定是 0000011111。然而，如果发生了三个错误，0000000000 变成了 0000000111，则以上编码就不能够正确地纠正错误了。

设想我们要设计一种编码方案，每个码字有 m 个报文位和 r 个校验位，并且能够纠正所有的单个错误。对于 2^m 个合法报文，任一个报文都对应有 n 个非法的码字，它们与该报文的距离为 1。这些非法的码字可以这样构成：将该报文对应的合法码字的 n 位逐个取反，可以得到 n 个距离为 1 的非法码字。因此，每个合法的报文都要求 $n+1$ 个位模式，专门供它使用。由于总共只有 2^n 个位模式，所以，必须有 $(n+1)2^m \leqslant 2^n$。利用 $n=m+r$，这个要求变成了 $(m+r+1) \leqslant 2^r$。在给定 m 的情况下，这个条件给出了用于纠正单个错误所需要的校验位数目的下界。

实际上，利用海明 1950 年提出的方法，这个理论下界是可以达到的。码字中的每一位连续编号，从最左边位 1 开始，它的右边是位 2，等等。编号为 2 的幂次方的位（1，2，4，8，16，…）为校验位，剩下的位（3，5，6，7，9，…）用 m 个数据位来填充。每一个检验位都迫使某一组位（包括它自己）的奇偶值为偶数（或奇数），一个位可能包含在几次奇偶值计算中。为了看清楚位置 k 上的数据位对哪些校验位有影响，将 k 重写成 2 的幂次方的和。例如 $11=1+G+8$，$29=1+4+8+16$，只有出现在 k 的展开式中的校验位才校验位置 k 上的数据位。

当一个码字到来时，接收方将一个计数器初始化为 0。然后，它检查每一个校验位 $k(k=1,2,4,8,\cdots)$，看它是否有正确的奇偶性。如果没有，则接收方将 k 加到计数器上。如果在所有的校验位都被检查过之后，计数器为 0，即这些校验位都是正确的，则该码字被作为有效码字而接收。如果计数器不为 0，则它包含了不正确位的编号。例如，如果校验位 1、2 和 8 是错误的，则变反的位（即错误的位）是 11，因为只有它才被 1+2 和 8 位校验。图 3.7 显示了一些 7 位 ASCII 字符，利用海明码将它们编成了 11 位的码字。注意，数据位出现在位置 3，5，6，7，9，10 和 11 上。

通常，海明码只能纠正单个错误，但通过一个技巧就能使海明码也能够纠正突发性的错误。k 个连续的码字被排列成一个矩阵，每行一个码字。通常情况下，传输数据时每次一个码字，从左向右。为了纠正突发性的错误，传输数据时每次发送一列，从最左边的列开始。当第 1 列所有的 k 位都被发送出去以后，再发送第 2 列，以此类推，如图 3.7 所示。当这一帧到达接收方时，接收方重构同样的矩阵，每次 1 列。如果一个突发性错误的长度为 k 位，则在 k 个码字中，每个码字至多只有 1 位受到影响，但是利用海明码，每个码字可以纠正一个错误，所以整个数据块也可以恢复出来。这种方法利用 kr 个校验位，使 km 个数据位能够抵抗长度等于或小于 k 的单个突发性错误。

2. 检错码

纠错码广泛应用于无线链路，因为无线链路相比铜线或者光纤有更多的噪声，也更容易出错。如果不使用纠错码，则几乎任何数据都难以通过。然而，在铜线或者光纤上错误率非常低，对于这些链路上偶尔出现的错误，利用错误检测和重传机制往往更加有效。

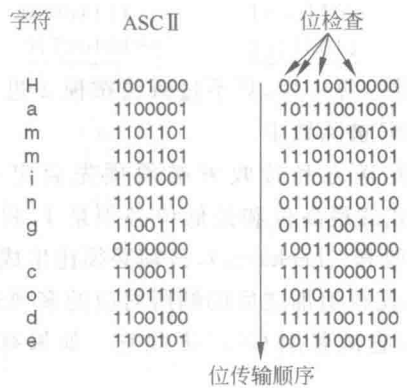

图 3.7 利用海明码来纠正突发性错误

举一个简单的例子，考虑这样一个信道：错误是孤立的，错误率为每位 10^{-6}。对于 1000 位的数据块，为了提供纠错功能，需要 10 个校验位；1 兆位的数据将需要 10000 个校验位，如果仅仅为了检测数据块中的单个 1 位错误，则每个数据块 1 个奇偶位就足够了。每 1000 个数据块将需要额外传输一个块(1001 位)。利用“错误检测＋重传”的方法，总的开销是每 1 兆位数据只需 2001 位。相比之下，如果利用海明码，则需要 10000 位。

如果在一块数据中只增加了一个奇偶位，但是在传输这一块数据时发生了一个很长的突发性错误，那么该错误能够被检测到的概率只有 0.5，这是难以接受的。如果每一块数据被当作一个矩阵(n 位宽，k 位高)来发送，则检测到错误的几率会明显增加。为每一列单独计算一个奇偶位，并将它附在矩阵的下边作为最后一行，然后按每次一行发送该矩阵数据，当数据块到达接收方时，接收方检查所有的奇偶位。如果任何一位有错误，则接收方请求重传该数据块。根据需要可以再次请求重传，直到接收到的整个数据块没有任何奇偶错误为止。

这种方法可以检测出长度为 n 的单个突发性错误，因为每列只有 1 位被改变了。然而，对于长度为 $n+1$ 的突发性错误，如果第 1 位变反，第 $n+1$ 位变反，所有其他的 $n-1$ 位都是正确的，则这种方法无法检测出这样的错误(突发性错误并不意味着所有的位都是错误的，它只意味着至少第 1 位和最后 1 位是错误的)。

如果数据块被一个长的突发性错误或者多个短一点的突发性错误影响了，则在 n 列中，任何一列有正确的奇偶位的概率是 0.5，所以，一个坏块被当前正确数据块接收的概率为 2^{-n}。

尽管上述方案有时已经足够了，但是在实践中，广泛使用的是另外一种方法：多项式编码(polynomial code)，也称为 CRC(Cyclic Redundancy Check，循环冗余校验码)。多项式编码的基本思想是：将位串看成是系数为 0 或 3 的多项式。一个 k 位的帧看作一个 $k-1$ 次多项式的系数列表，该多项式共有 k 项，从 x^{k-1} 到 x^0。这样的多项式认为是 $k-1$ 阶多项式。高次(最左边)位是 X^{k1} 项的系数；接下去的位是 X^{k2} 项的系数；依次类推。例如 110001 有 6 位，因此代表了一个共有 6 项的多项式，其系数为 1、1、0、0、0 和 1，即 $x^8+x^4+x^0$ 多项式的算术运算采用代数域理论的规则，以 2 为模来完成。加法没有进位，减法没有借位，加法和减法都等同于异或。例如：

$$\begin{array}{rrrr} 10011011 & 00110011 & 11110000 & 01010101 \\ +11001010 & +11001101 & -10100110 & -10101111 \end{array}$$

长除法与二进制中的长除运算一样，只不过减法按模 2 进行。如果被除数与除数有一样多的位，则称该除数“进入到”被除数中。

当使用多项式编码时，发送方和接收方必须预先商定一个生成多项式（generator polynomia）。$G(x)$生成多项式的最高位和最低位必须是 1，假设一帧有 m 位，它对应于多项式 $M(x)$，为了计算它的校验和（checksum），该帧必须比生成器多项式长。基本的思想是在帧的尾部追加一个校验和，使得追加之后的帧所对应的多项式能够被 $G(x)$除尽。当接收方收到了带校验和的帧之后，它试着用 $G(x)$去除它。如果有余数，则表明传输过程中有错误。

计算校验和的算法如下：

(1) 假设 $G(x)$的阶为 m，在帧的低位端加上 r 个 0 位，所以该帧现在包含 $m+r$ 位，对应多项式为 $x^rM(x)$。

(2) 利用模 2 除法，用对应于 $G(x)$的位串去除对应于 $x^rM(x)$的位串。

(3) 利用模 2 减法，从对应于 $x^rM(x)$的位串中减去余数（总是小于等于 r 位）。

结果就是将被传输的带校验和的帧，它的多项式不妨设为 $T(x)$。图 3.8 显示了当帧为 1101011011，生成器多项式为 x^4+x+1 时计算校验和的情形。

显然，$T(x)$可以被 $G(x)$除尽（模 2）。在任何一种除法中，将被除数减去余数，剩下的差值一定可以被除数除尽。例如，在十进制中，如果用 210278 除以 10941，则余数为 2399。从 210278 中减去 2399，得到 207879，它可以被 10941 除尽。

现在来分析一下这种方法的功能，什么样的错误可以被检测到呢？想象一下在传输过程中发生了一个错误，所以接收方收到的不是 $T(x)$，而是 $T(x)+E(x)$，$E(x)$中的每一个“1”位都对应于有一位变反了。如果 $E(x)$中有 k 个“1”位，则表明有 k 个“1”位错误发生了。一个突发性错误可以这样来描述：首先是 1，然后是 0 和 1 的混合，最后也是 1，所有其他的位都是 0。

接收方在收到了带校验和的帧之后，用 $G(x)$来除它，接收方计算$[T(x)+E(x)]/G(x)$。$T(x)/G(x)$是 0，所以计算的结果是 $E(x)/G(x)$。如果错误多项式 $E(x)$恰好包含$G(x)$作为它的一个因子，这样的错误检测不到，其他的错误都能够检测得到。

如果只有一位发生错误，$E(x)=x^i$，这里 i 决定了错误发生在哪一位上。如果 $G(x)$包含两项或者更多项，则它永远也不会除尽 $E(x)$，可见，所有的一位错误都将被检测到。

如果有两个独立的 1 位错误，则 $E(x)=x^i+x^j$，这里 $i>j$。换一种写法，$E(x)$可以写成 $E(x)=x^j(x^{i-j}+1)$。假定 $G(x)$不能被 x 除尽，则所有的双位错误都能够被检测到的充分条件是，对于任何小于等于 $i-j$ 最大值（即小于等于最大的帧长度）的 k 值，$G(x)$都不能除尽 x^k+1。简而言之，低阶的多项式可以保护长的帧。例如，对于任何 $k<32768$，$x^{13}+x^{14}+1$ 都不能除尽 x^k+1。

如果有奇数个位发生了错误，则 $E(x)$包含奇数项（比如 x^5+x^2+1，但不是 x^2+1）。有意思的是，在模 2 的系统中，没有一个奇数项多项式包含 $x+1$ 作为因子。因此，以 $x+1$ 作为 $G(x)$的一个因子，就可以捕捉到所有包含奇数个位变反的错误情形。

为了理解奇数项多项式不可能被 $x+1$ 除尽，用反证法，假设 $E(x)$有奇数项，并且可以

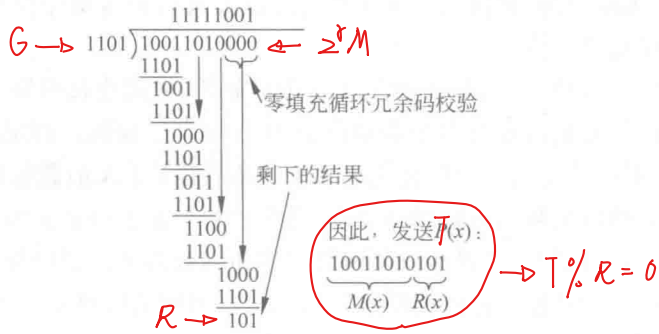

图 3.8 多项式编码校验和的计算过程

被 $x+1$ 除尽。于是，将 $E(x)$ 分解成 $(x+1)Q(x)$。现在令 $x=1$，则得到 $E(1)=(1+1)Q(1)$。由于 $1+1=0 \pmod 2$，所以 $E(1)=0$。如果 $E(x)$ 有奇数项，则用 1 代替所有的 x，结果总是 1。因此，奇数项多项式不可能被 $x+1$ 除尽。

最后要说明的是，带 r 个校验位的多项式编码可以检测到所有长度小于等于 r 的突发性错误。长度为 k 的突发性错误可以用 $x^i(x^{k-1}+\cdots+1)$ 来表示，这里 i 决定了突发性错误的位置离帧的最右端的距离有多远。如果 $G(x)$ 包含一个 x 项，那么它不可能有 x^i 作为因子，所以，如果括号内表达式的阶小于 $G(x)$ 的阶，那么余数永远不可能为 0。

如果突发性错误的长度为 $r+1$，那么当且仅当错误多项式等于 $G(x)$ 时，错误多项式除以 $G(x)$ 的余数才为 0。根据突发性错误的定义，第一位和最后一位必须为 1，所以它是否与 $G(x)$ 匹配取决于其他 $r-1$ 个中间位。如果所有的组合被认为是等概率的，则这样一个不正确的帧被当作有效帧而接收的概率是 $\left(\frac{1}{2}\right)^{r-1}$。

同样也可以证明，当一个长度大于 $r+1$ 位的突发性错误发生时，或者几个短一点突发性错误发生时，一个坏帧被当作有效帧而通过检测的概率为 $\left(\frac{1}{2}\right)^{r-1}$。这里假设所有的位模式都是等概率的。

有一些特殊的多项式已经成为国际标准了，其中在 IEEE：802 中使用的多项式为

$$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^6+x^4+x^2+x^1+1$$

以上的多项式有一些很好的特性，其中一个是：它能够检测所有长度小于等于 32 的突发性错误，以及所有只影响奇数个位的突发性错误。

3.3 数据链路层的基本协议

本章我们从三个逐渐复杂的协议入手，在介绍这些协议之前，先明确一些有关通信模型的基本假设。首先，假设物理层、数据链路层和网络层都是独立的进程，它们通过来回传递报文进行通信。在许多情况下，物理层和数据链路层进程会在一个特殊的网络 I/O 电路中的一个处理器上运行；而网络层代码则在主 CPU 上运行。然而，其他的实现方案也是有可能的（比如，三个进程都在同一个 I/O 电路中运行，或者物理层和数据链路层作为过程，被