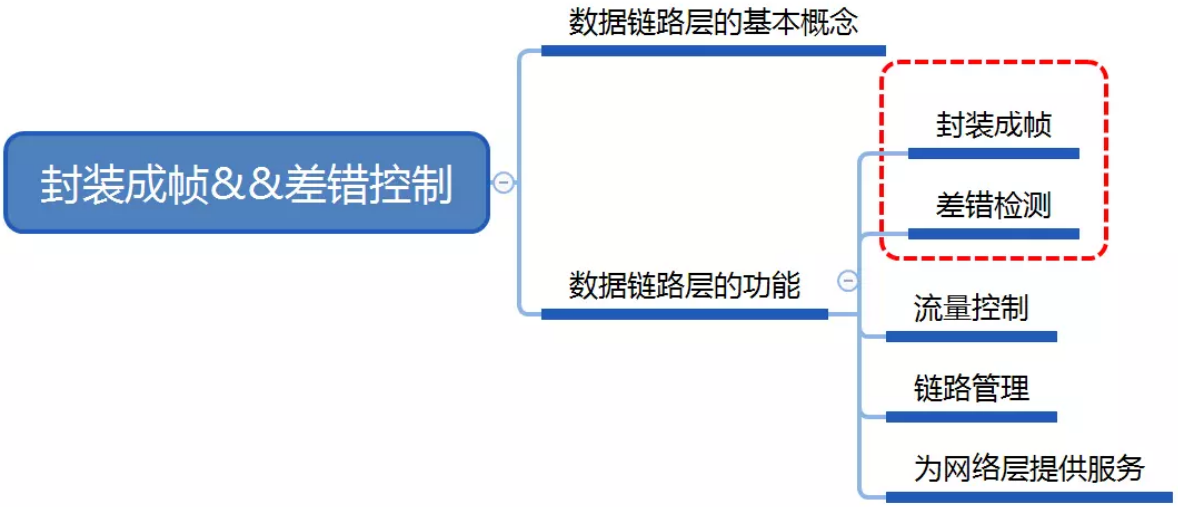
数据链路层功能——封装成帧与差错控制

内容总览



本节开始介绍数据链路层的内容，数据链路层属于计算机网络的低层，需要对其上层网络层提供服务。咱们这里主要介绍数据链路层的一些基本概念和数据链路层的两个功能：封装成帧和差错控制。

1.数据链路层的基本概念

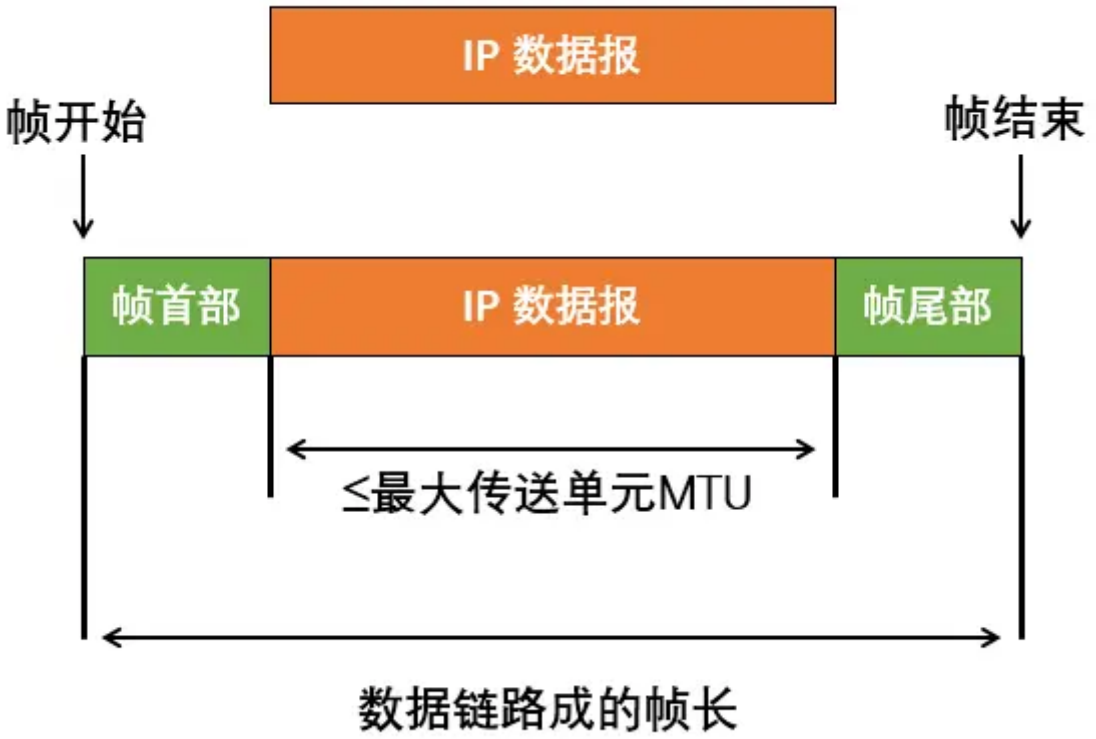
(1) **节点**：主要是指主机、路由器。
(2) **链路**：网络中两个节点之间的**物理通道**。
(3) **数据链路**：网络中两个节点之间的**逻辑通道**，把实现控制数据传输**协议**的硬件和软件加到链路上就构成了数据链路。
(4) **帧**：链路层的协议数据单元，封装网络层的数据报。

数据链路层的功能：在物理层提供服务的基础上**向网络层提供服务**，其最基本的服务是将源自网络层来的数据可靠的传输到相邻节点的目标机网络层。其主要作用是**加强物理层传输原始比特流的功能**，将物理层提供的可能出错的物理连接改造成**逻辑上无差错的数据链路**，使之对网络层表现为一条无差错的链路。

> **功能一**：为网络层提供服务。无确认无连接服务，有确认无连接服务、有确认有连接服务。 **功能二**：链路管理，即连接的建立，维持。释放（用于面向连接的服务）。
>
> **功能三**：组帧。
>
> **功能四**：流量控制。
>
> **功能五**：差错控制。

2.封装成帧

组装成帧就是在一段数据的前后部分添加首部和尾部，这样就构成了一个帧。帧是数据链路层的数据传送单元。接收端再收到物理层上交的比特流后，就能根据首部和尾部的标记，从收到的比特流中识别帧的开始和结束。

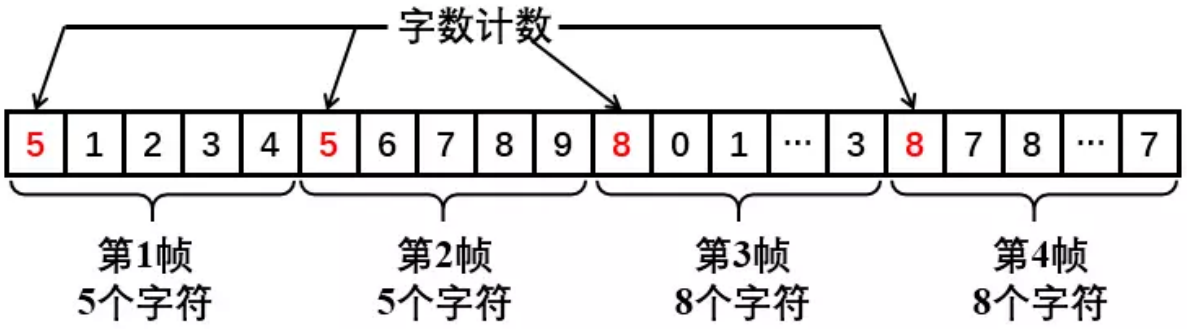


首部和尾部包含很多控制信息，其中一个重要作用就是：**帧定界（确定帧的界限）。** **帧同步**：接收方应当能够从接收到的二进制比特流中区分出帧的起始和终止。 **帧长**：帧的数据部分长度加上帧首部和帧尾部的长度。 **最大传送单元**（MUT，Maximum Transfer Unit）：每一种链路层都规定了所能传送帧的数据部分长度上限。 组帧的四种方式：**字符计数法、字符填充法、零比特填充法、违规编码法。** **透明传输**：指不管所传数据是什么样的比特组合，都应当能够在链路上传送。因此，链路层就看不见有什么妨碍数据传输的东西。 当所传的数据中的比特组恰巧与某一个控制信息完全一样时，就必须采取适当的措施，使接收方不会将这样的错误认为是某种控制信息。这样才能保证数据链路层的传输是透明的。

> **透明传输理解举例：**
>
> 例如如果传送的比特流中数据部分的数据恰好和帧尾部相同，那么在传输过程碰到该部分数据时，接收端可能误认为到这里传输结束了，那直接丢弃之后的数据了，这就会导致数据传输不完整，所以就要采取适当的措施，即使有这些问题也要保证数据能正确的传输，保证链路层对任何数据都能传送，在数据看来，链路层没有东西妨碍自己传送或者说链路层对数据是透明的。

2.1.字符计数法（非重点，了解即可）

字符计数法：帧首部使用一个计数字段（第一个字节）来表明帧的长度。在接收时根据帧首部的字数计数就可以知道一个帧的长度。



缺点：如果其中某个帧的计数错误，那么会导致之后传输的数据全部错误。

> 对于上图，如果第1帧的计数在传输过程中发生错误又5变成了4，那么接收端在接收到4个字符就认为该帧数据接收完毕了，开始接收下一帧，认为下一帧的字数计数是4，下一帧的数据也会出错.....最终导致数据错误。

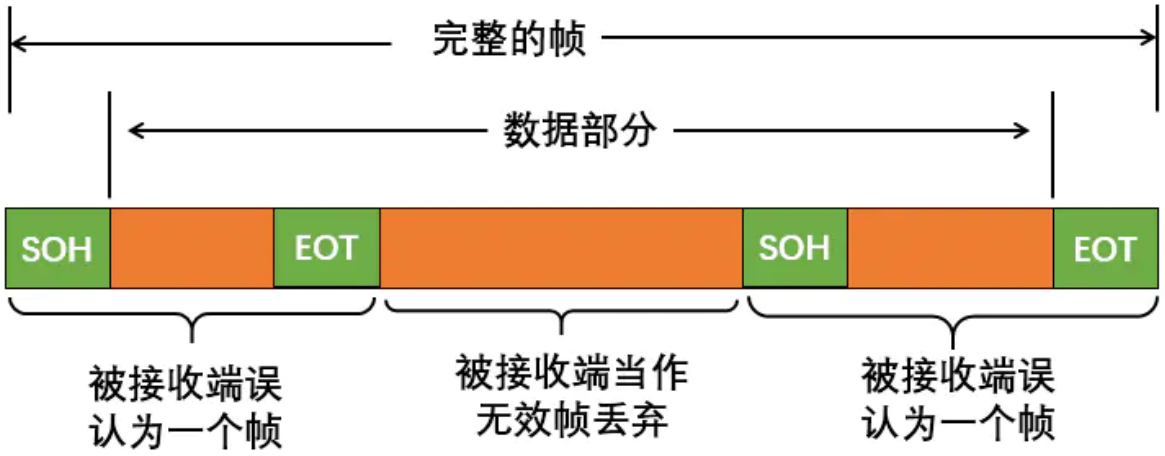
2.2.字符填充法

当数据时由可打印的ASCII码组成的文本文件时，帧定界可以使用特殊的**帧定界符。**控制字符**SOH**（Start Of Header）放在一帧的最前面，表示帧的首部开始。另一个控制符**EOT**（End Of Transmission）表示帧结束。SOH和EOT是控制符的名称，它们的二进制编码分别是0000 0001和0000 0100。

> 当使用ASCII编码时，一共有128个不同的ASCII码，其中可打印的有95个，有33个是不可打印的，，SOH和EOT就是不可打印的编码，所以可以用来作为控制符。

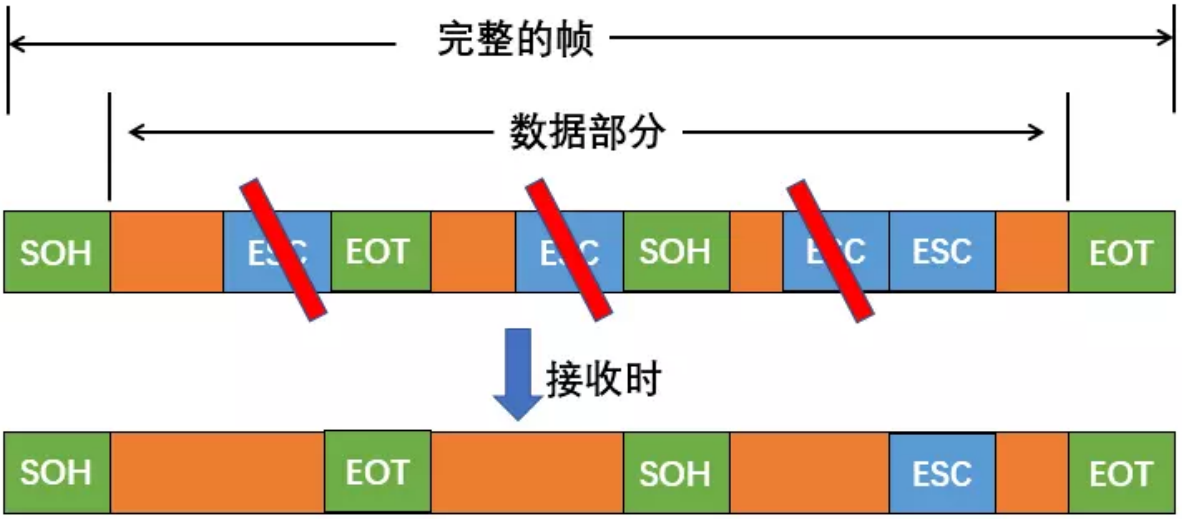
当传送到帧是用文本文件组成的帧时（文本文件中的字符都是从键盘上输入的），其数据部分显然不会出现像SOH或EOT这样的帧定界控制符，可见不管从键盘上输入什么字符都可以放在帧上传输过去，因此这样的传输就是透明传输。

但是当数据部分是非ASCII码的文本文件时（如二进制代码的计算机程序或图像等），数据中的某个字节的二进制代码恰好和SOH或EOT这样的控制字符一样。这样接收端就会错误的找到帧的边界，把部分帧收下，丢弃部分数据。



为了解决透明传输问题，**在发送端的数据链路层中出现控制字符SOH或EOT的前面插入一个转义字符“ESC”（其二进制编码为0001 1011）。而在接收端的数据链路层在把数据送往网络层之前删除这个插入的转入字符。**

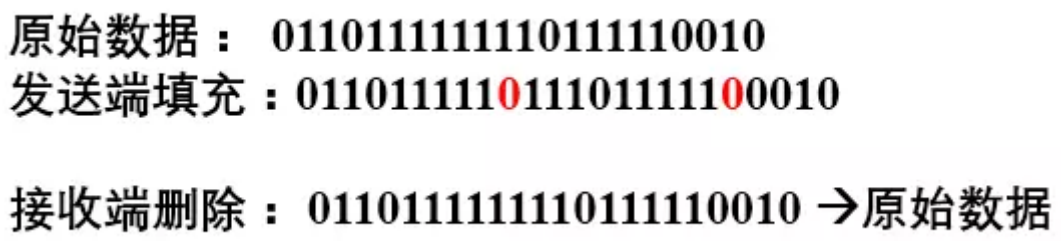
> 如果转义字符也在数据中，同样需要在转义字符前再插入一个转义字符。当接收端接收到连续的两个转义字符时，就删除其中前面的一格个。



2.3.零比特填充法

> **零比特填充法帧的控制信息的首部和尾部是相同的。**

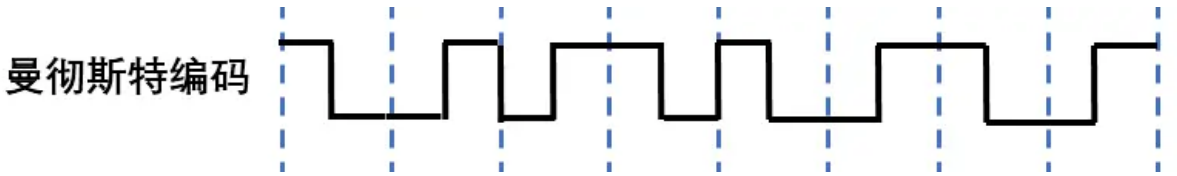
零比特填充法：**在发送端，扫描整个信息字段，只要有连续的5个1，就立即填入1个0。在接收端收到一个帧时，先找到标志字段确定边界，再用硬件对比特流进行扫描。发现连续5个1时，就把后面的0删除。**



零比特填充保证了透明传输，在传送的比特流中可以传送任意比特组合，而不会引起对帧边界的判断错误。

2.4.违规编码法（知道这个名字即可，有兴趣的看内容）

> 在曼彻斯特编码中，一个码元可以用前高后低表示1，前低后高表示0，只有这两种情况，不存在前高后高和前低后低这两种情况。所以可以使用这两种不存在的情况作为定界帧的起始和终止。



由于字节计数法中的count字段的脆弱性（其值若有差错将导致灾难性后果）及字符填充实现上的复杂性和不兼容性，目前普遍使用的是**比特填充**和**违规编码法。**

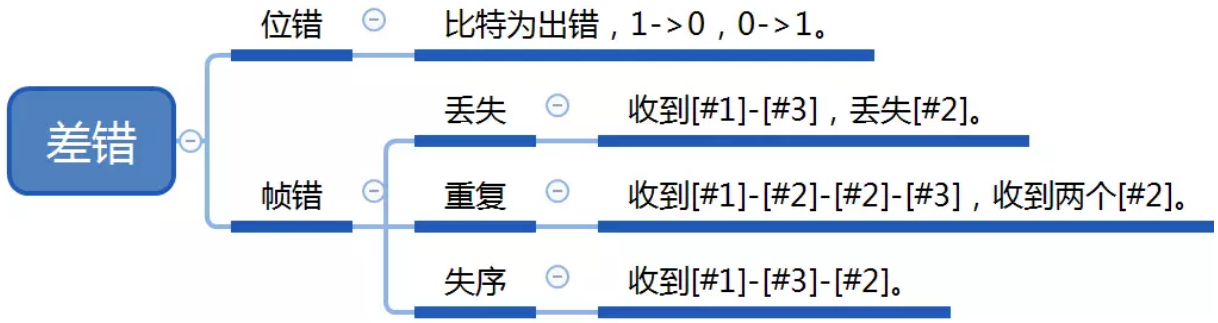
3.差错控制

差错的由来：现实中的通信链路都不会是理想的，由于噪声的存在，比特在传输的过程可能会产生差错。

噪声分为**全局性**和**局部性。**

> (1) 全局性噪声是由于线路电气特性所产生的**随机噪声（热噪声）**，是信道**固有的，随机存在的。**可以通过提高信噪比来减少或避免干扰。
>
> (2) 局部性噪声是外界**特定的**短暂原因所造成的**冲击噪声**，是产生差错的主要原因。可以**利用编码技术来解决。**

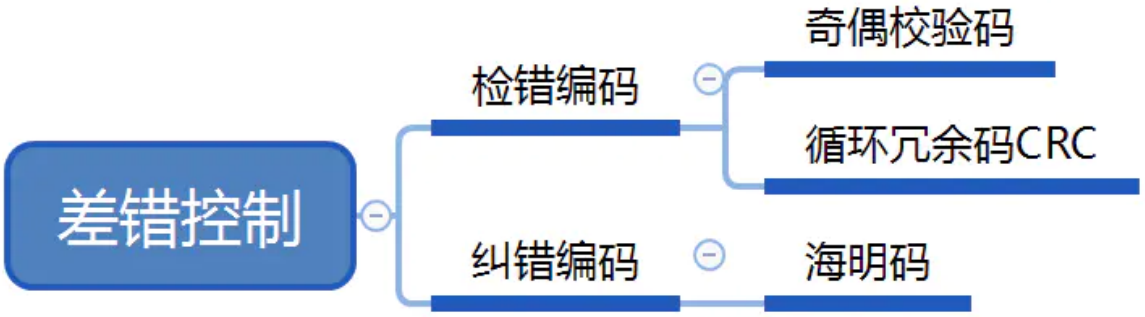
在数据传输过程中，差错可以分为**位错和帧错。**



> 注：发送的帧为[#1]-[#2]-[#3]。

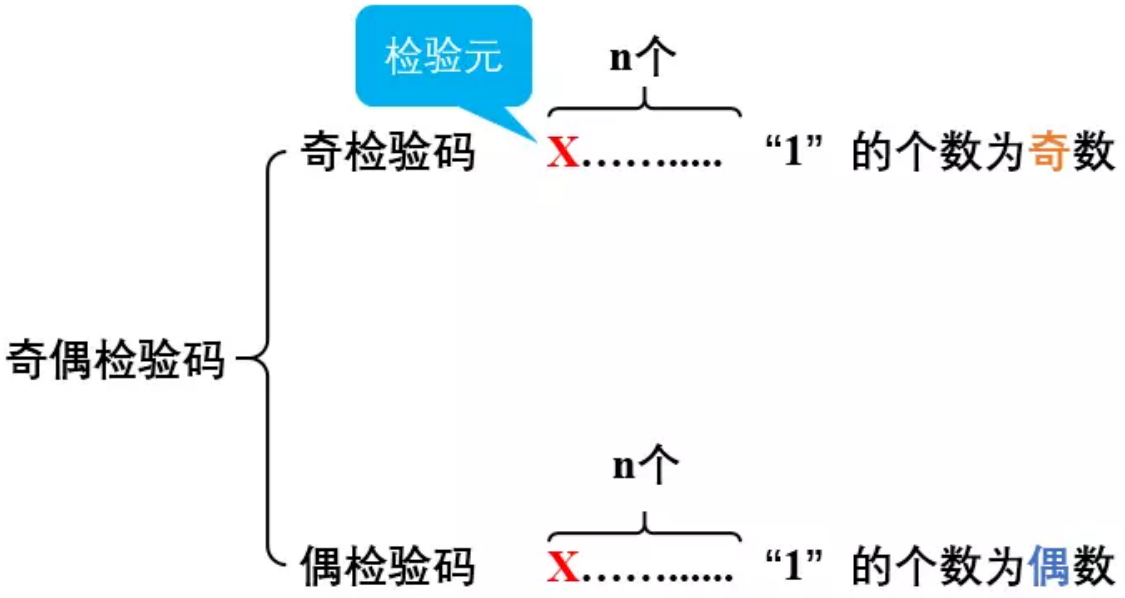
差错控制方法：**检错编码和纠错编码** 。

> 这里的差错控制是针对的是比特错，帧错后面再说。



3.1.奇偶校验码

奇偶检验码分为**奇检验码和偶检验码。**由n-1位信息元和在数据信息前的1位校验元组成。



> (1) 奇检验码：将数据转换为二进制数据，数据中的“1”的个数加上检验元的“1”的个数为奇数，如果接收端的1的个数是偶数，说明一定发生了错误。
>
> (2) 偶检验码：将数据转换为二进制数据，数据中的“1”的个数加上检验元的“1”的个数为偶数。如果接收端的1的个数是奇数，说明一定发生了错误。

奇偶检验码的特点：

(1) 奇偶检验码可以检测比特错，但是不能检测到是哪个比特出现差错。

> 例如，如果一个字符S的ASCII编码从底到高一次为1100101，采用奇校验，那么发送端发送的就是11100101（加上校验元，1的个数是奇数），经过传输后，如果接收端收到的字符，如果是11100111，1的个数变成了6个，说明一定发生了错误，但是不能检测是哪个比特出现了错误。

(2) 奇偶校验码只能检测出奇数个比特错误，检错能力为50%。

> 对于上例，如果接收端接收的数据是11010101，即发生了2位比特错误，但是2个位错正好相抵消，即一个由0变为1，另一个由1变成0，数据中的1个数仍然是5，所以同样不能检测出错误。同理，只有发生奇数个比特错误时，才可以检测出来。

3.2.循环冗余校验码(记住英文缩写是CRC)

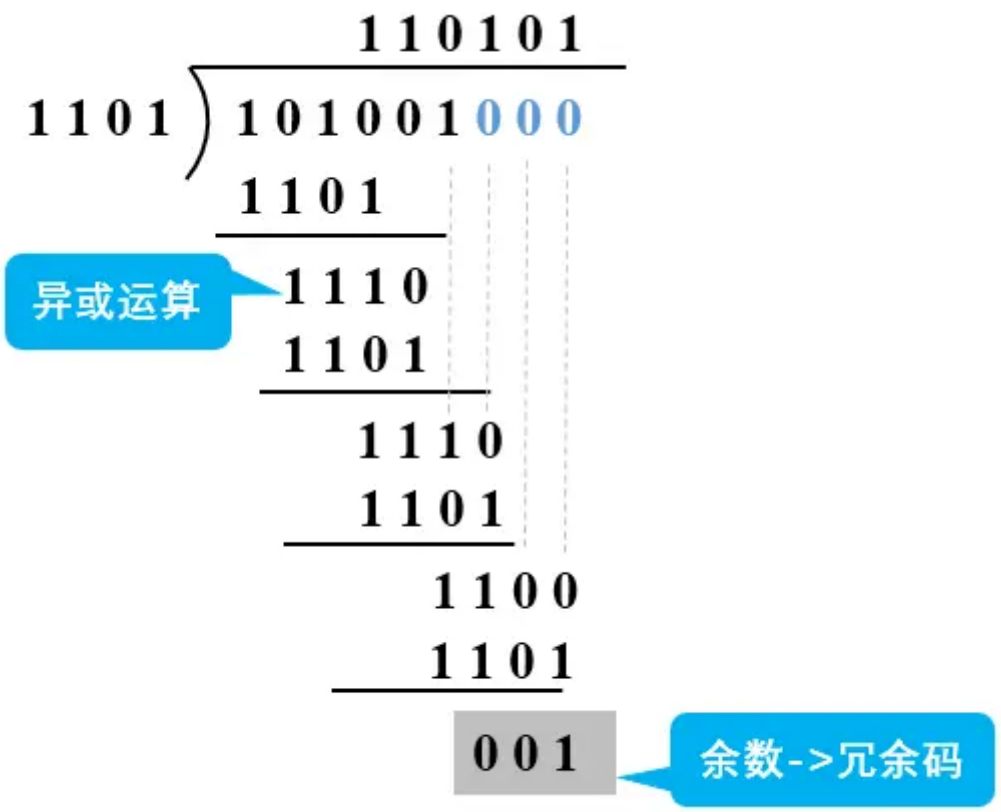
冗余编码：在数据发送前，先按某种关系附加上一定的冗余位，构成一个符合某一规则的码字后再发送。当要发送的有效数据变化时，相应的冗余位也随之变化，使码字遵从不变的规则。接收端根据收到的码字是否仍符合原规则，从而判断是否出错。

循环冗余检验原理：在发送端，假设要发送k个比特，CRC运算就是在这k个比特后添加供查错检验的n位冗余码，n位冗余码又称为**帧检验序列FCS**（Frame Check Sequence），然后构成一个帧发送出去，一共发送（k + n）个位。这n位冗余码可以通过发送的数据和一个数相除得来，这个数是收发双发事先约定好的数。接收端收到发送端发送的（k + n）位比特后，需要将这些比特位和FCS相除，如果余数是0，表示没有差错就接收，反之，则丢弃。

举例说明：如果客户端要发送的数据是M = 101001，除数多项式P = 1101。

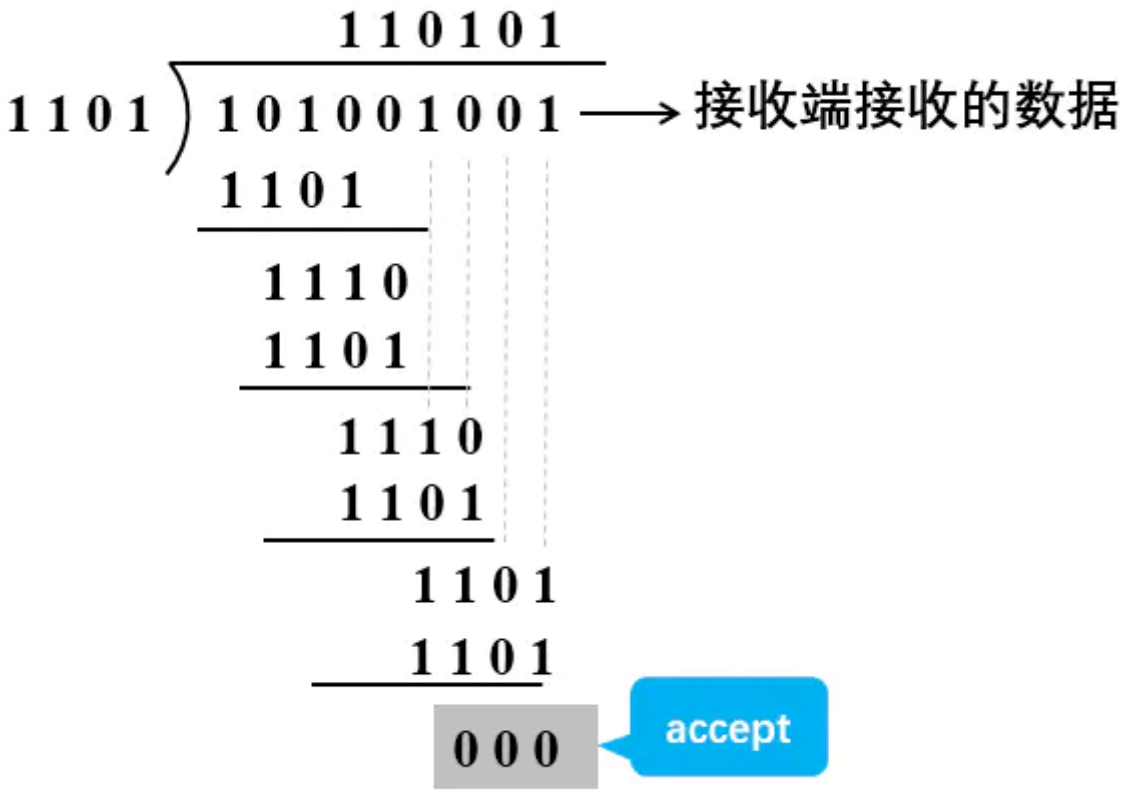
> (1) 先确定n的大小。由于1101 = 1 * 23 + 1 * 22 + 0 * 21 + 1 * 21，所以该多项式是3阶，即（n = 3），也就是如**果多项式是N位，阶就是N-1位**，同样如果除数多项式为11000，那么阶数就是4，n的值就是4。

(2) 计算n为冗余码（FCS）。n位冗余码是M 与 P的余数，先用n位的0代替冗余码添加到M之后即被除数为101001000，然后在于P相除，所得的余数就是冗余码。



> (3) 发送端发送的数据就是101001001。

> (4) 如果接收端接收到的数据是101001001，那么同样除以P，看余数是否为0。如果是0，表示数据没有发生错误，则接受，反之则丢弃。



FCS的生成以及接收端CRC检验都是由硬件实现的，处理很迅速，因此不会延误数据的传输。

在数据链路层仅仅使用循环冗余检验CRC差错检测技术，只能做到对帧的无差错接收，即凡是接收端数据链路层接受的帧，都可以以非非常接近1的概率认为这些帧在传输过程中没有产生差错。接收端丢弃的帧虽然曾经接收到了，但最终还是因为有差错被丢弃。所以通常可以认为：**凡是接收端数据链路层接收的帧均为无差错。**

> 注：链路层使用CRC检验，能够实现无比特差错的传输，只接受没有错误的信息，丢弃错误的数据，但这**并不是可靠传输。可靠传输是发送端发送什么，接收端就收到什么。**

3.3.汉明校验/海明校验

海明码可以发现双比特错，纠正单比特错。

工作流程：

> (1) 确定校验码位数r。
>
> (2) 确定检验码和数据的位置。
>
> (3) 求出校验码的值。
>
> (4) 检错并纠错。

(1) 确定校验码的位数r

> 海明不等式：**2r ≥ k + r +1** r为冗余信息位，k为信息位。

加入要发送的数据D = 101101，数据位数k = 6，带入海明不等式，满足不等式的r最小值为4。即D = 101101的海明码应该有10（6 + 4）位。其中6位是数据位，4位是检验码位。

(2) 确定校验码和数据的位置
假设4为检验码分配为P1、P2、P3、P4；数据从左到右分别为D1、D2....D6
检验码要放在2的n次幂的位置，即1、2、4、8这四个位置。

数据位	1	2	3	4	5	6	7	8	9	10
代码	P1	P2	D1	P3	D2	D3	D4	P4	D5	D6
实际值			1		0	1	1		0	1

(3) 求出检验码的值

二进制	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010
数据位	1	2	3	4	5	6	7	8	9	10
代码	P1	P2	D1	P3	D2	D3	D4	P4	D5	D6
实际值	0	0	1	0	0	1	1	1	0	1

> 这里因为最大的位数就是10，10最少需要4个二进制位表示，所以所有数据都是用4个二进制位表示。

对于每一个校验码都可以检验一个或多个数字。

> 对于第一个检验码P1，它负责检验所有二进制第4位（最后一位）是1的数据位，即数据位1、3、5、7和9。P1的值需要和这些数据位的数异或运算，需要最后的结果是0来确定P1的值。所以P1^ 1 ^ 0 ^ 1 ^ 0 = 0，得P1 = 0。

> 对于第二个检验码P2，它负责校验所有二进制位第3位是1的数据位，即数据位2、3、6、7、10，P2的值计算方法和P1相同，即P2 ^ 1 ^ 1 ^ 1 ^ 1 = 0，得P2 = 0。同理可以计算出其余的检验码的值。

所以求出数据的海明码位0010011101。

(4) 检错并纠错

假设在数据的传输过程中第5位出错，因此接收到的数据位就是0010111101。接收端接收到数据后，会进行检错，分别计算检验码与检验的位的异或的值。

> P1和数据位3、5、7、9的值作异或运算：0 ^ 1 ^ 1 ^ 1 ^ 0 = 1;

> P2和数据位3、6、7、10的值作异或运算：0 ^ 1 ^ 1 ^ 1 ^ 1 = 0;

> P3和数据位5、6、7的值作异或运算：0 ^ 1 ^ 1 ^ 1= 1;

> P4和数据位9、10的值作异或运算：1 ^ 0 ^ 1 = 0;

将上述的值逆序组成一个二进制数：0101，其所对应的十进制数就是5，即第5位发生了错误，所以直接将第5位的数变为0即可实现纠错的功能。

4.总结