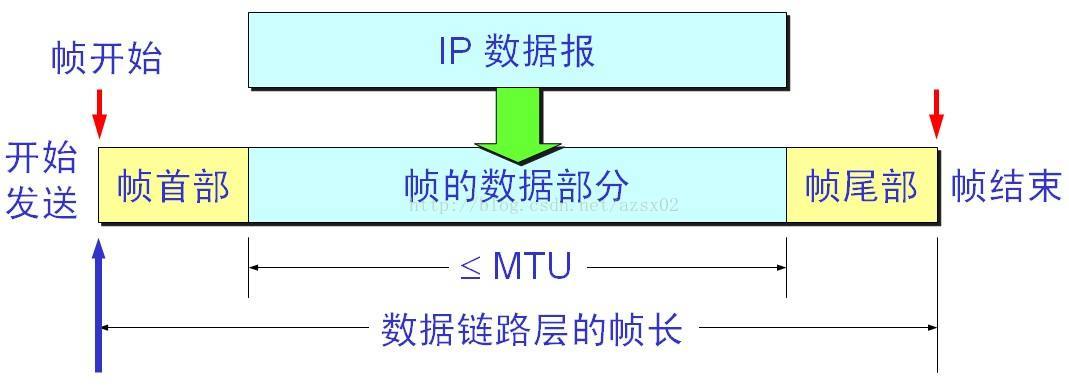
### 一. 封装成帧

      封装成帧（framing）就是在一段数据的前后分别添加首部和尾部，这样就构成了一个帧。接收端在收到物理层上交的比特流后，就能根据首部和尾部的标记，从比特流中识别帧的开始和结束。



**首部和尾部的一个重要作用就是进行帧定界（即确定帧的界限）**。此外，首部和尾部还包括许多必要的控制信息。

为了提高帧的传输效率，应当使帧的数据部分长度尽量大于首部和尾部的长度。最大传输单元MTU（Maximum Transfer Unit）：帧数据部分长度上限。

当数据是由可打印的ASCII码组成的文本文件时，帧定界可以使用特殊的帧定界符。ASCII码是7位编码，一共可以组合成128个不同的ASCII码，其中可打印的有95个，而不可打印的控制字符有33个。（可打印的字符就是可以在键盘上输入的字符，我们使用的标准键盘有47个键可输入94个字符，包括使用shift键，加上空格键，一共可以输入95个可打印字符）

控制字符**SOH（start of header）**放在一帧的最前面，表示帧的首部开始。另一个控制字符**EOT（end of transmission）**表示帧的结束。注意：SOH和EOT都只是控制字符的名称，他们的十六进制编码分别是01（二进制是00000001）和04（二进制是00000100）。SOH，EOT 并不是S O H E O T这几个字符，只是名字而已。

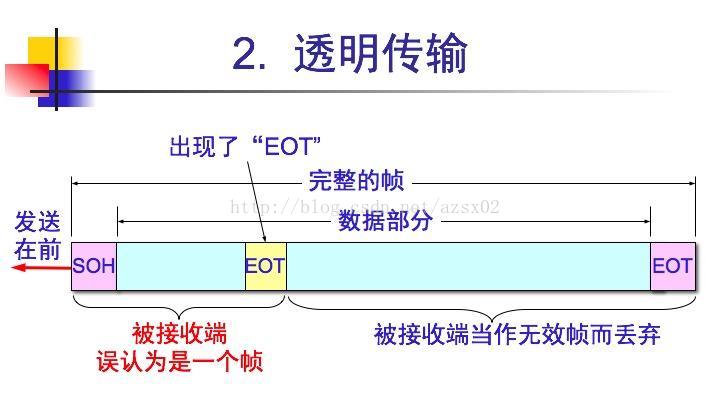
当传输中出现差错，帧定界符的作用更加明显。假设未发送完一个帧而发生故障，接收端就知道收到的数据是不完整的帧（只有首部），必须丢弃。

---------------------

### 二. 透明传输

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

但当数据部分是非ASCII码的文本文件时（如二进制代码的计算机程序或者图像等），情况就不同了。如果数据中的某个字节的二进制代码恰好和SOH或EOT这种控制字符一样，数据链路层就好错误地“找到帧的边界”，把部分帧收下（误以为是个完整的帧），而把剩下的那部分数据丢弃（这部分找不到帧定界控制字符SOH）。这种情况就不是透明传输，因为当遇到书中碰巧出现字符“EOT”时就传不过去了，而其后面的数据因找不到“SOH”被接收端当做无效帧而丢弃。但实际上数据中出现的字符“EOT”并非控制字符而仅仅是二进制数据00000100 。



为了解决透明传输问题，****字节填充法或字符填充**：在控制字符SOH、EOT的前面插入一个**转义字符ESC（其十六进制编码是1B，二进制是00011011）。**而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义字符。**

### **差错检测**

**现实的通信链路都不会是理想的。传输过程中，1可能变成0, 0 可能变成1 。这就叫比特差错。——**误码率**。 误码率和信噪比有很大的关系，提高信噪比就可以提高**

**因此，在计算机网络传输数据时，必须采用各种差错控制技术。目前在数据链路层广泛使用了循环冗余检验（CRC）的检错技术。**

**在数据链路层的CRC检验都是用硬件完成的，处理很迅速，因此不会延误数据的传输。**

**为什么数据链路层要以帧为单位来传送数据呢？因为如果不以帧为单位，就无法加入冗余码来进行差错检验。**

****传输差错分为两类：一类就是前面所说的最基本的比特差错。第二类：收到的帧出现了帧丢失、帧重复和帧失序。（停止等待协议，ARQ）。****

**数据链路层并不需要给网络层提供“可靠传输”的服务。过去OSI的观点是：必须让数据链路层向上提供可靠传输。因此在CRC的基础上，增加了帧编号、确认和重传机制。**

**而现在的互联网采用了区别对待的方法：**

**对于通信质量良好的有线传输链路，数据链路层协议不使用确认和重传机制，即不要求数据链路层向上提供可靠传输的服务。如果在数据链路层传输数据时出现了差错并且需要进行改正，那么改正差错的任务就由上层协议（例如，运输层的TCP协议）来完成。**

**对于通信质量较差的无线传输链路，数据链路层协议使用确认和重传机制，数据链路层向上提供可靠传输的服务**

#### CCRC（循环冗余检验）

<https://blog.csdn.net/qq_33411687/article/details/82593466>

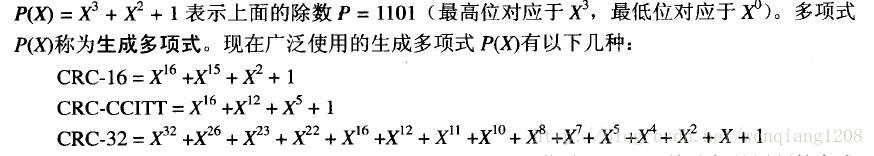
**M=101001，除数P=1101，n=3，M补n位0，除P，得到n位余数，补在M后面**

**首先来介绍几个概念**

**(1)模2运算：实际上是按位异或运算，即相同为0，相异为1，也就是不考虑进位、借位的二进制加减运算。如：1111+1010 = 0101**

**(2)**FCS**：其实就是冗余码，帧检验序列(Frame Check Sequence)**

**(3)**生成多项式**：其实就是除数，比如下面将要用到的除数p = 1101**



**---------------------**

**在接收端把接收到的数据M = 101001001以帧为单位进行CRC检验：把收到的每一个帧都除以相同的除数p(模2运算)，然后检查得到的余数R。**

**如果在传输过程中没有差错，那么经过检验后得到余数R肯定是0。**

**(读者可以自己检验下，被除数现在是M = 101001001,除数P= 1101，看余数是否为0)**

**总之，在接收端对接收到的每一个帧经过CRC检验后，有两种情况：**

**(1)余数R = 0，则判断这个帧没有问题，就接受**

**(2)余数R != 0,则判断这个帧有差错，就丢弃。**

**在数据链路层，发送端帧检验序列FCS的生成和接收端的CRC 检验都是用硬件完成的，处理很迅速，因此并不会延误数据的传输。最后强调，**在数据链路层仅仅使用CRC差错检测技术，则只能做到对帧的无差错接受，但还不是可靠传输**。**

**模2除法：**

**规则:假设被除数X,和除数P,余数R**

**1. 被除数X除以P(对X和P做模2加减法),被除数首位为1时,商1,为0时商0**

**2. 所得余数去除首位(左移一位):**

**- R第一位为0,将其作为新的被除数,除以0,此时其首位为0,商即为0**

**- R第一位为1,将其作为新的被除数,除以P,此时其首位为1,商即为1**

**3. 重复第2步直到R位数少于P位数**

**---------------------**