一、封装成帧.

封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。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，从收到的比特流中识别帧的开始和结束。

帧定界符

首部和尾部包含许多的控制信息，他们的一个重要作用：**帧定界**（确定帧的界限）。

帧同步：

接收方应当能从接收到的二进制比特流中区分出帧的起始和终止。

帧的数据部分有最大值. 称为最大传送单元（MTU）

组帧的四种方法：

1. 字符计数法.
2. 字符(节)填充法.
3. 零比特填充法.
4. 违规编码法.

二、透明传输.

无论所传数据是什么样的比特组合，都应当能在链路上传送

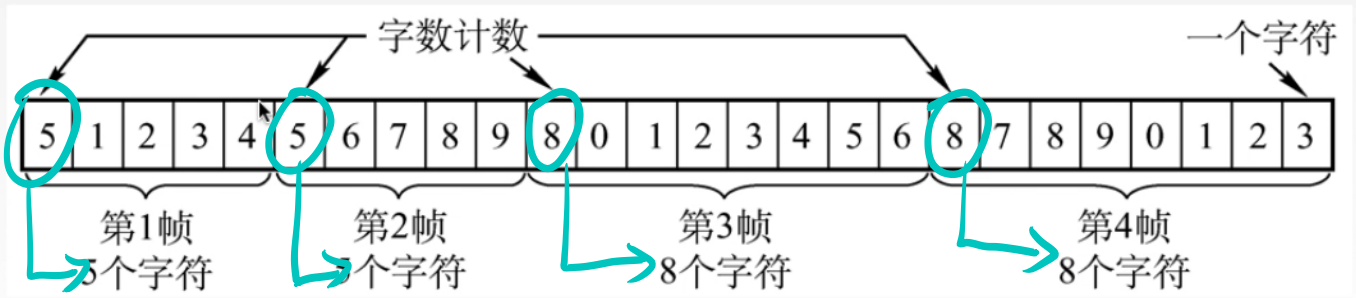
数据链路层看不见有什么妨碍数据传输的东西

当所传数据中的比特组合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，使收方不会将这样的数据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。

三、组帧的四种方法.

1. 字符计数法.

帧首部使用一个计数字段（第一个**字节**，八位）来标明帧内字符数。

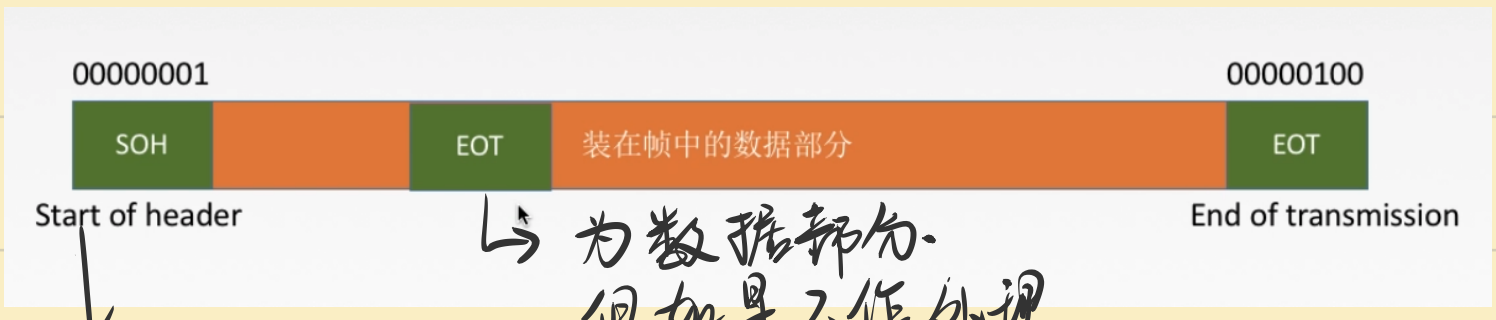


缺点：如果计数字段错误，可能引起后面大规模错误的发生.

2. 字符填充法.

例 设 00000001 为 SOH，00000100 为 EOT.
如果遇到 SOH 那么代表帧开始.
EOT 代表帧结束.

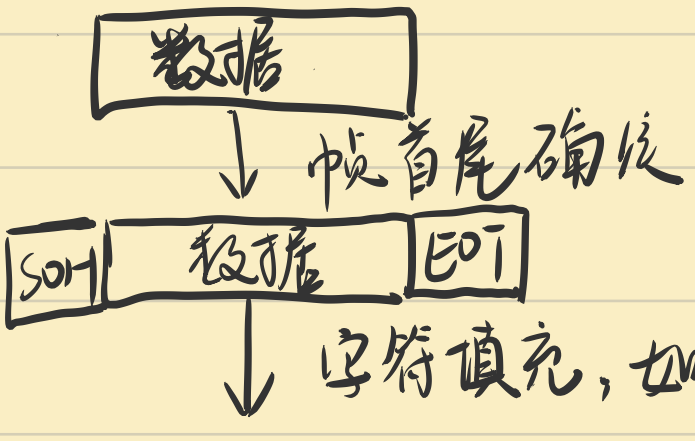
问题：数据部分可能存在与 SOH、EOT 相同的比特流.

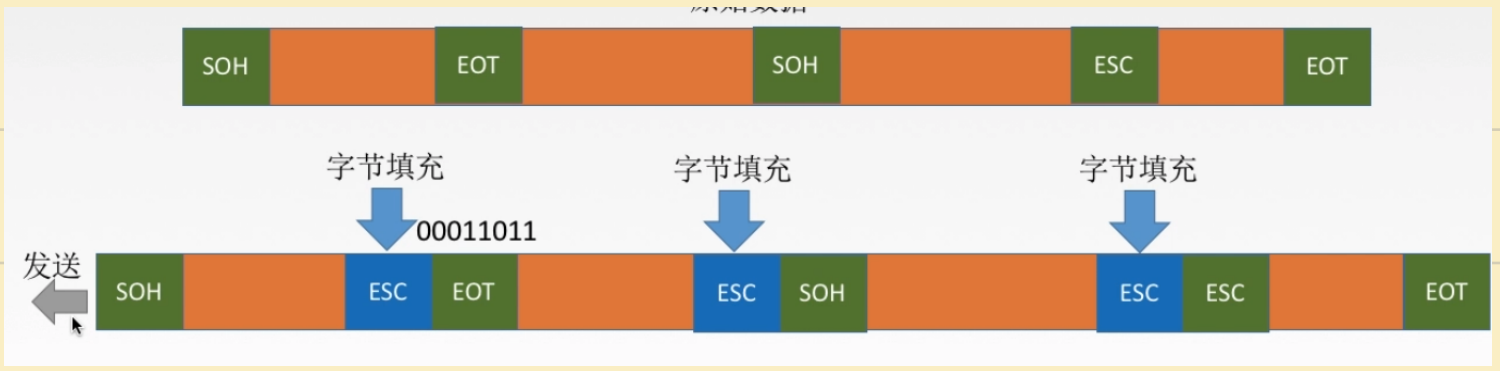


↳ 为数据部分.
但如果不作处理.
会被当作结束符导致帧提前结束

开始.

解决方法：使用转义字符 ESC
遇到 ESC 则不管后面是什么
一律当作数据处理.





接收时查看数据部分，如果EOT、SOH、ESC前存在ESC，则把前面这个ESC去掉。

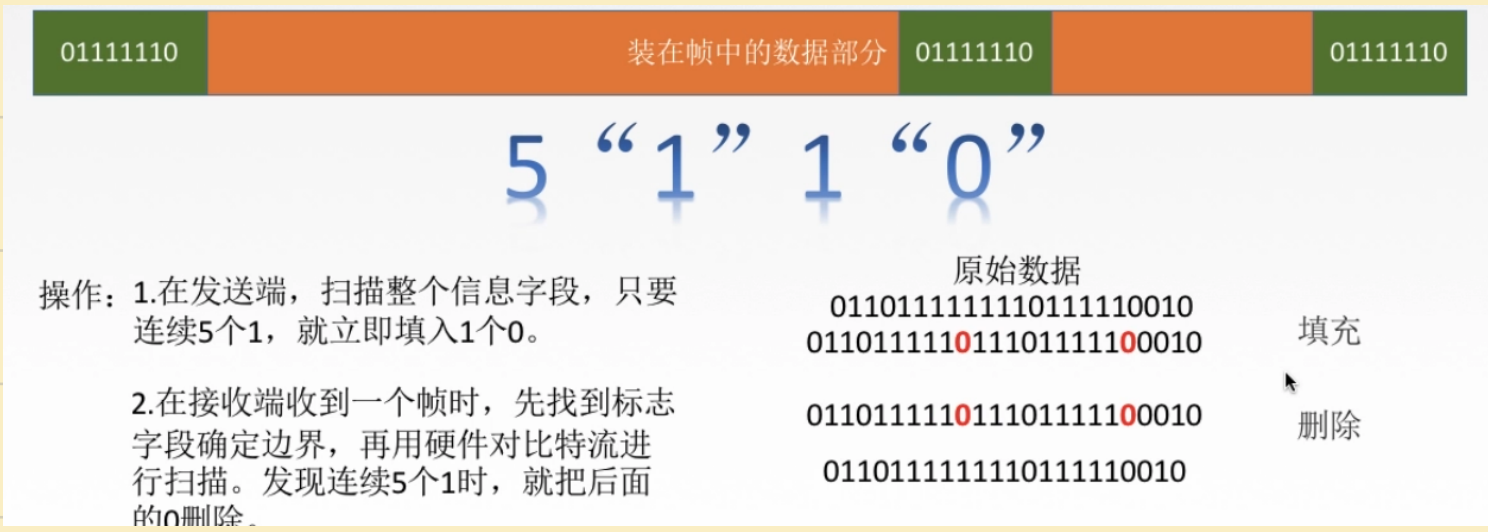
3、零比特填充法

将左右边界均设为01111110。

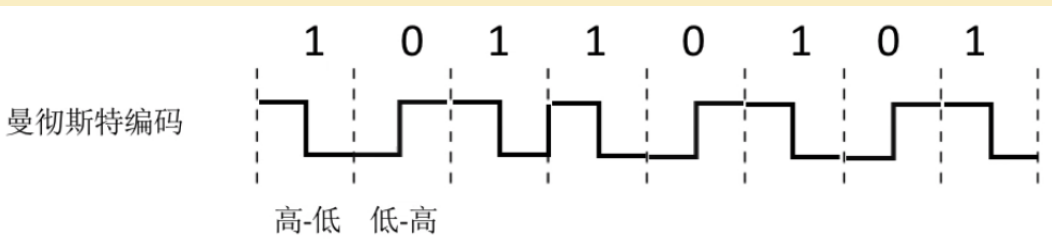
先将数据部分填充：如果有连续5个1，则在第5个1后面填0。

然后加上左右边界。发送。

接收时去掉首尾标志后，逆操作以删除每5个1后的一个0。



4、违规编码法



可以用“**高-高**”，“**低-低**”来定界帧的起始和终止。

由于字节计数法中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，目前较普遍使用的帧同步法是**比特填充**和**违规编码法**。