物理层主要解决的问题是如何在各种传输媒体上传输比特0和1的问题，但是光能传输0和1还不够，为什么这么说呢？本篇文章走进数据链路层，先来看下封装成帧这件事。

一、引入数据链路层的原因

我们先来考虑最简单的情况，两台计算机要互相通信，如何做呢？

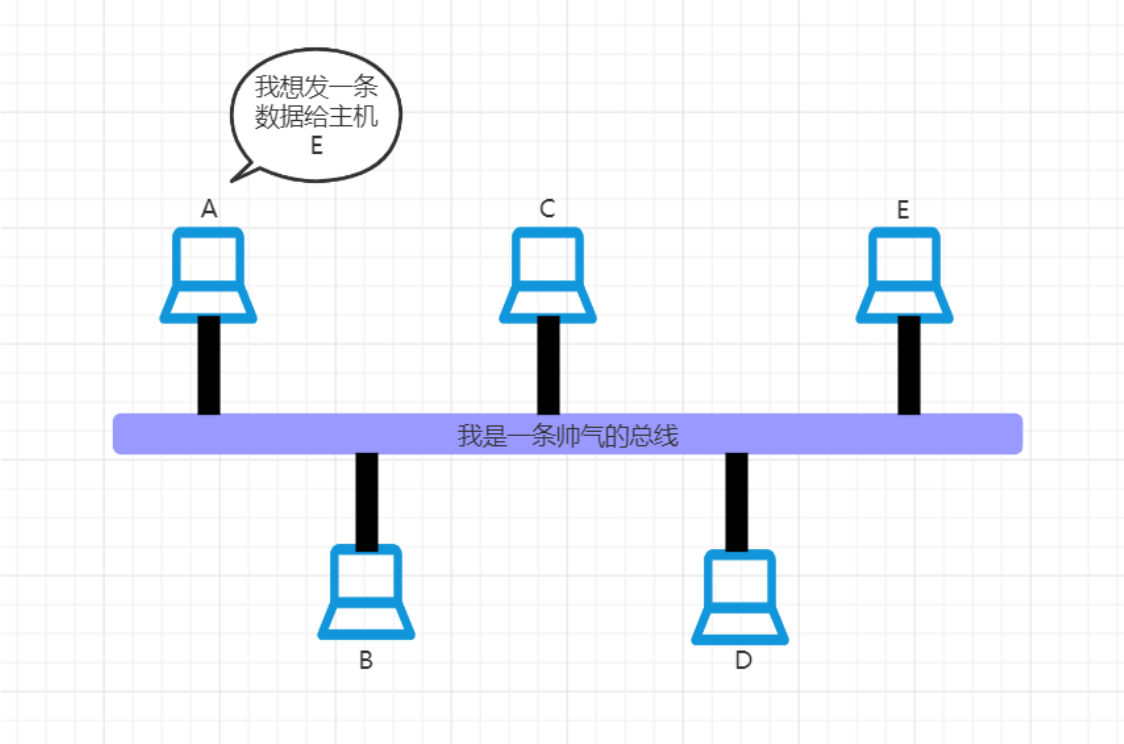
我们需要考虑以下几个问题：

- 采用什么样的传输介质？是用我们平时用的双绞线？还是光纤呢？
- 采用什么样的物理接口呢？比如平时咱们用的RJ45接口？
- 采用怎样的信号来表示比特0和1呢？用方波信号？高电平表示1？低电平表示0？

以上问题，实际上是物理层所要关注的事情。

> 严格来说，传输介质不属于物理层，它并没有被包含在体系结构之中；计算机网络中传输的信号并不是简单的方波信号。

假设我们已经解决了上面的问题，两台主机之间可以互相发送比特0和1了，我们继续思考，实际上计算机网络是由多台计算机构成，比如下面图示这种，利用总线型拓扑结构构建的一个网络：



假设主机A想发送一条消息给主机E，我们知道，在总线型网络拓扑结构中，表示消息的信号都是经过这条总线传输的，所有其他主机都可以收到这个信号，那么自然而然就会有一个问题：**主机E如何知道这条消息是发送给我的呢？其他主机是如何知道这个信息不是发送给我是要丢弃的呢？**

如何解决呢？**这就引出如何标识网络中各个主机的问题了，大家可能听说过网卡的MAC地址，实际上我们就是用这个MAC地址来标识唯一一个网卡。**（切记切记，MAC地址是唯一标识网卡的，MAC地址不等同于主机地址，因为一台主机可能有多个网卡）

此外，既然有了地址信息，那么就得区分地址信息和实际数据，如何从信号中区分出地址和数据也是个问题。

此外，这种总线型网络拓扑结构存在消息碰撞问题，这个该如何解决呢？另外需要注意的是，总线型网络拓扑结构早已淘汰，目前主流是用交换机将多台主机互连形成交换式以太网，**那么以太网交换机又是如何实现的呢**？

我们可将这些问题归结到数据链路层。让我们带着这些问题走进数据链路层的世界，希望在学习完之后这些问题都已得到完美回答。

二、数据链路层概述

链路：即Link，就是从一个结点到相邻结点的一段物理链路，而中间没有任何其他的交换结点。

数据链路：即data link，是指把实现通信协议的硬件和软件加到链路上，就构成了数据链路。

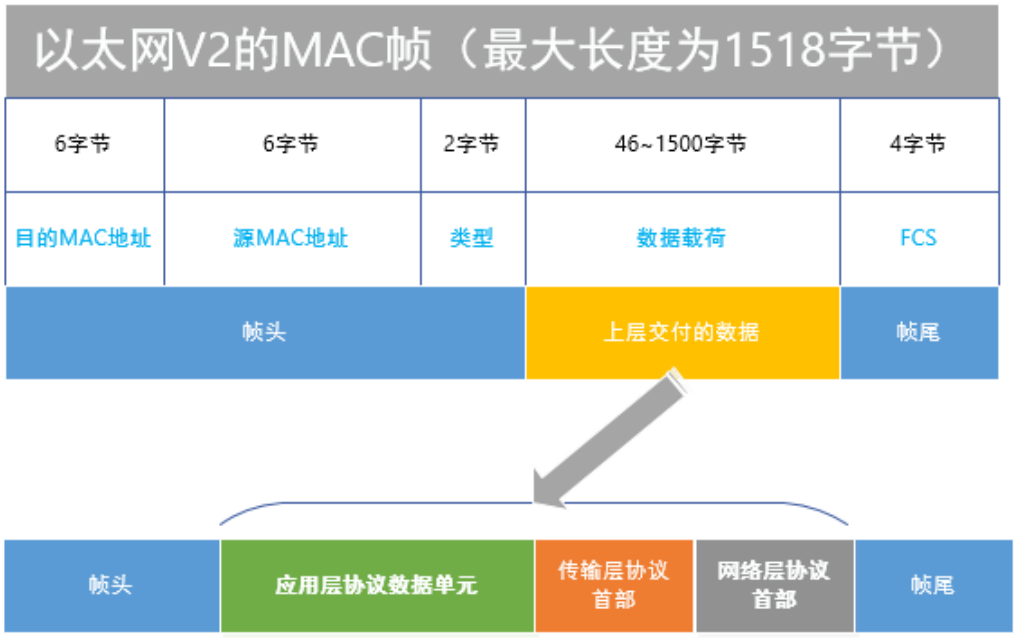
在数据链路层上传输的数据包称为帧。换句话说就是，数据链路层上以帧为单位传输和处理数据。

关于帧，就要说说数据链路层中一个重要问题：封装成帧。

三、封装成帧问题

所谓封装成帧，意思很简单：指数据链路层给上层交付的协议数据单元增加帧头和帧尾使之成帧。

帧头和帧尾中包含了重要的控制信息，我们以【以太网MAC帧】格式为例说明：



我们可以看到在以太网MAC帧中，帧头和帧尾分别包含了一些重要信息，这些信息都是咱们数据链路层中最重要的控制信息。

比如包含了目的MAC地址和源MAC地址，这解决了局域网内主机的地址寻址问题。还有包含了一个FCS是用来校验帧数据在传输过程中是否出错。

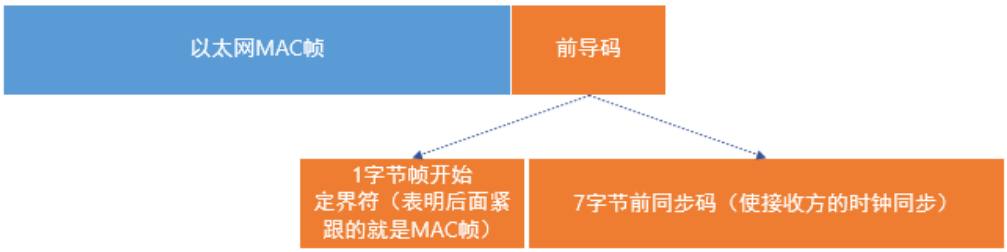
成功封装为帧后，通过物理层，将构成帧的各比特转换成电信号发送到传输媒体。

接收方如何从比特流中提取出一个个帧呢？其实帧头和帧尾的一个作用之一是帧定界，比如PPP点对点协议，PPP的帧格式为：

在PPP协议的帧头和帧尾中各包含一字节的标志字段，其作用就是帧定界。

我们注意到，以太网MAC帧中没有定义帧定界标志，那么接收方又是如何从比特流中提取出以太网各个MAC帧的呢？

实际上，以太网的数据链路层封装好MAC帧后，将其交付给物理层，物理层会在MAC帧前面添加8字节的前导码，然后再将比特流转换为电信号发送：



另外，以太网还规定了帧间发送间隔为96比特发送时间，因为MAC帧不需要帧结束定界符。

想明白了如何对帧进行定界后，我们再来思考一个问题。我们知道PPP帧中用标志字段来进行帧定界，我们抽象为如下：

但是如果除了帧头的标志位置，其他地方比如上层交付的数据单元中恰好也有一样的flag标志字段呢？**接收方接收帧的时候就会出现误判！**

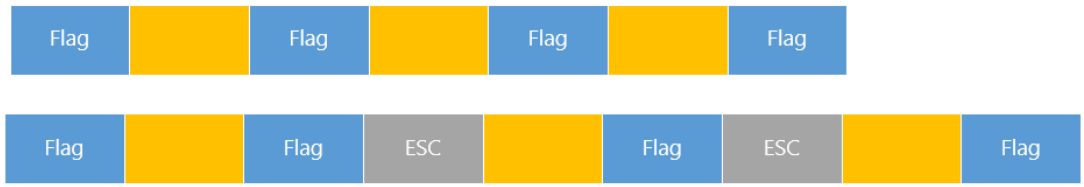
而我们要避免这个问题发生，就是要实现一个重要的目标：透明传输。

四、透明传输问题

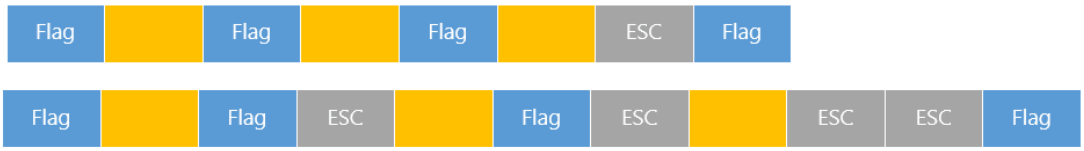
透明传输，是指数据链路层对上层交付的传输数据没有任何限制，就好像数据链路层不存在一样。

如何避免如上情况发生呢？即如何实现透明传输呢？

我们可以在发送帧之前对帧进行扫描，如果发现有flag标志字段，可在其前面加上转义字符，这样接收方接收帧时，遇到flag+ESC的组合就可以认为这个flag只是一个普通的字符而已，当成普通数据处理。



那么问题又来了，如果普通数据中也有转义字符呢？很简单，转义加转义，标识它是一个普通的转义字符，而不是标识其他字符的转义字符。



以上说的是**面向字节**的物理链路，使用转义字符进行填充实现透明传输，但是我们一般是用的**面向比特**的物理链路，该如何实现透明传输呢?

以点对点协议的某个帧为例:

上图中两端的01111110为帧定界标志（中间是6个1），假如在帧的数据部分也出现了两个01111110（标红的两个），但是实际上他们是普通的数据，而不是帧定界符，这种情况如何处理呢?在发送前，我们可以采用零比特填充法，**扫描帧的比特数据，每5个1后面插入一个0，确保帧定界符在整个帧中的唯一性**。接收方在接收帧时，只需要将数据部分中5个1后面的0剔除即可。

当然了，这里介绍的两种方式仅仅是作为一般性原理演示说明，实际的各种数据链路层都有其实现透明传输的具体方法，不一定会使用这里介绍的两种方法。

最后说明下，为了提高帧的传输效率，应当使帧的数据部分长度尽可能大一些。但考虑到差错控制等多种因素，每一种数据链路层协议都规定了帧的数据部分的长度上限，即最大传输单元MTU（Maximum Transfer Unit），这个我们后续会详细说明。