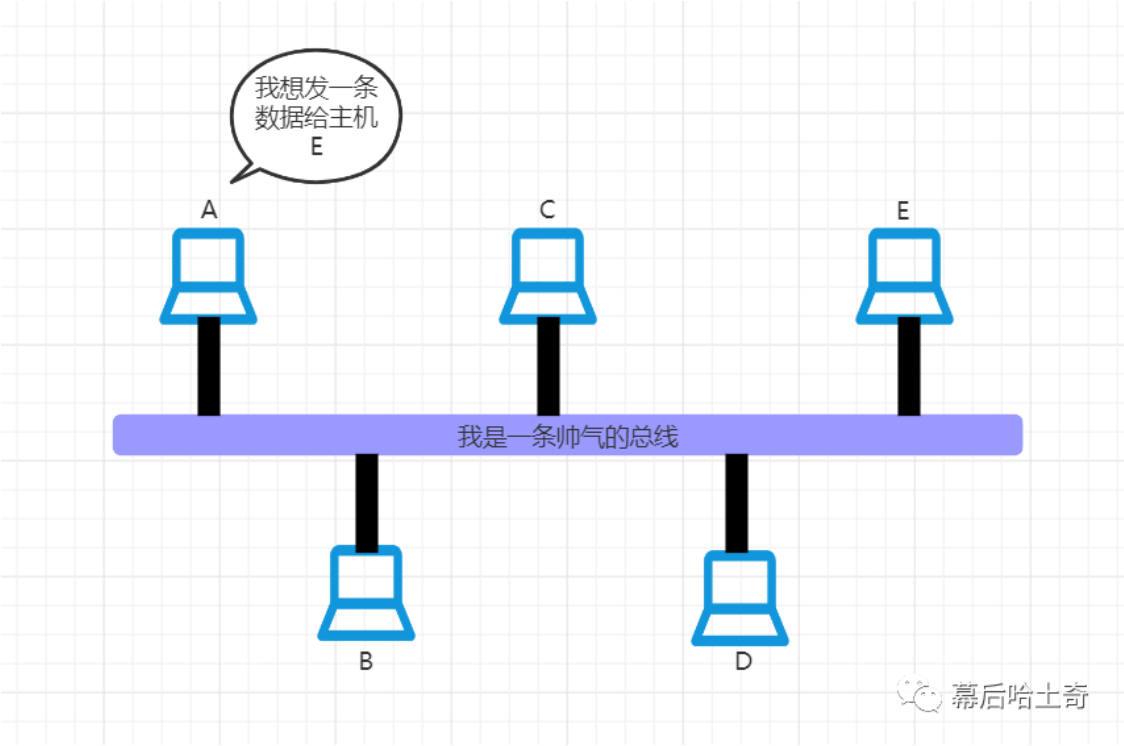
前面我们已经学习过，在数据链路层上传输的数据包称为帧。换句话说就是，数据链路层上以帧为单位传输和处理数据，那么这个帧长什么样子呢？在有线局域网中传输数据，是如何标识对方主机的呢？注意，我们讨论的范围是有线局域网。

一、引言

截至目前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局域网络：



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？

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**，这就需要制定协议，让大家按照一样的规范来发送和接收，这就是本文要讲的以太网协议。

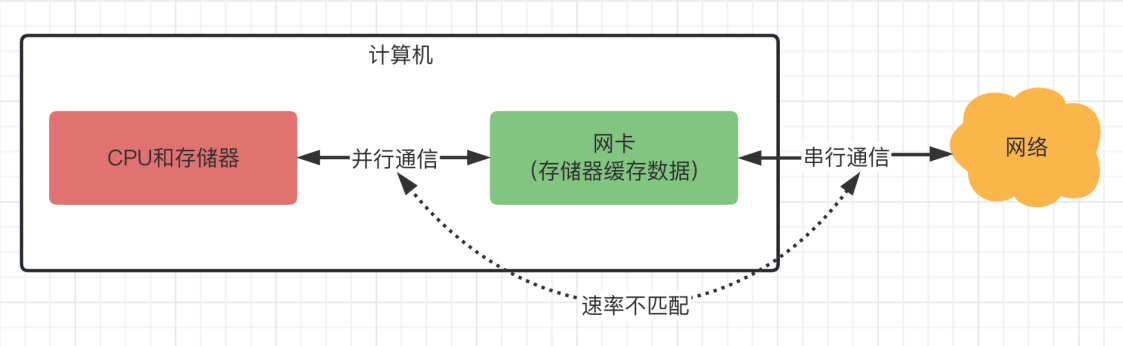
以上，涉及到几个重要的概念：网卡、MAC地址、以太网协议和以太网MAC帧。

二、网络适配器

要将计算机连接到局域网，需要用到对应的网络适配器（Adapter），这些网络适配器一般简称为“网卡”。

一般情况下，一台普通用户的计算机中往往会包含两块网卡，一块是用于接入有线局域网的以太网卡，另一块是用于接入无线局域网的WI-FI网卡。它们各自有一个全球唯一的MAC地址。

在计算机内部，网卡与CPU之间的通信，是通过计算机主板上的I/O总线以并行传输方式进行的，网卡与外部以太网之间的通信，一般是通过传输媒体（双绞线、光纤）以串行的方式进行的，显然，网卡除了要实现物理层和数据链路层的功能（需要实现以太网协议），还要进行并行传输和串行传输的转换，由于网络上的数据传输速率和计算机总线上的数据传输速率并不相同，因此在网卡中必须装有对数据进行缓存的存储芯片。



在确保网卡硬件正确的情况下，为了使网卡正常工作，还必须在计算机的操作系统中为网卡安装相应的设备驱动程序，驱动程序负责驱动网卡发送和接收帧。

网卡并不是独立的自治单元，因为网卡本身不带电源而是必须使用所插入的计算机的电源，并受该计算机的控制。因此网卡可看成为一个半自治的单元。

当网卡收到正确帧时，就以中断的方式通知CPU取走数据并将其交付给协议栈中的网络层；当网卡收到误码帧时，则直接抛弃此帧不必通知CPU。

一般情况下，网卡从网络上每接收到一个帧，就检查帧首部的目的MAC地址，按如下情况进行处理：

- 如果目的MAC地址是广播地址，则接受该帧；
- 如果目的MAC地址就是与网卡固化的全球单播MAC地址相同，则接受该帧；
- 如果目的MAC地址是网卡支持的多播地址，则接受该帧；

简单来说，网卡会判断经过它的每个帧，是否是发给它自己的帧，如果是则处理，如果不是则丢弃，不过也有特殊情况，因为网卡也可以被设置为一种特殊的工作模式：混杂模式。工作在混杂模式的网卡，只要收到帧就会收下，而不管帧中目的MAC地址是什么。大家想一想，在这种模式下，其实就是实现了对网络流量的监听，典型的应

用是嗅探器（sniffer），配合相应的工具软件，就可以作为一种非常有用的网络工具来学习和分析网络。

三、MAC地址

我们要想实现局域网内两台机器之间的通信，就需要给每一台机器设定一个唯一标识，类似于我们每个人的身份证，这就是MAC地址。上面强调过，MAC地址是跟网卡有关的，MAC 地址其实是网卡的地址，而不是某台机器的地址。一台电脑会可能有一个或多个网卡，每个网卡都需要有一个唯一的 MAC 地址。

MAC 地址以十六进制来编码，类似这样：

```
00:0c:29:10:5a:55
```

我们知道一个字节是8个bit，一个十六进制位需要4个bit来标识，那么显然MAC地址是由 6 个字节来编码，占了48个bit位。那么网卡的范围有2的48次方个，即281474976710656个，目前MAC地址的数量是够用的了，那如何保证唯一的呢？会不会重复呢？

一般来说不会。因为一个网卡制造商会购买 MAC 地址，更确切地说是 MAC 地址的区块。

MAC 地址最前面的三个字节是 IEEE（Institute of Electrical and Electronics Engineers，电气电子工程师学会）分配的，后面的三个字节由制造商自行分配。

当制造商要生产网卡之前，他们会买前三个字节（由 IEEE 分配），后三个字节自己分配。举个例子，某个网卡生产商买了00:01:02 这三个字节，那么其生产的所有网卡就由这三个字节起首，例如 00:01:02:00:00:01 ，00:01:02:00:00:02 ，等等。

这样，IEEE 分配的前三个字节被各个网卡生产商买走，因此不担心会有重复的情况，而生产商只要保证自己生产的每块网卡使用不同的后三个自定义的字节，就不会有重复的 MAC 地址啦。

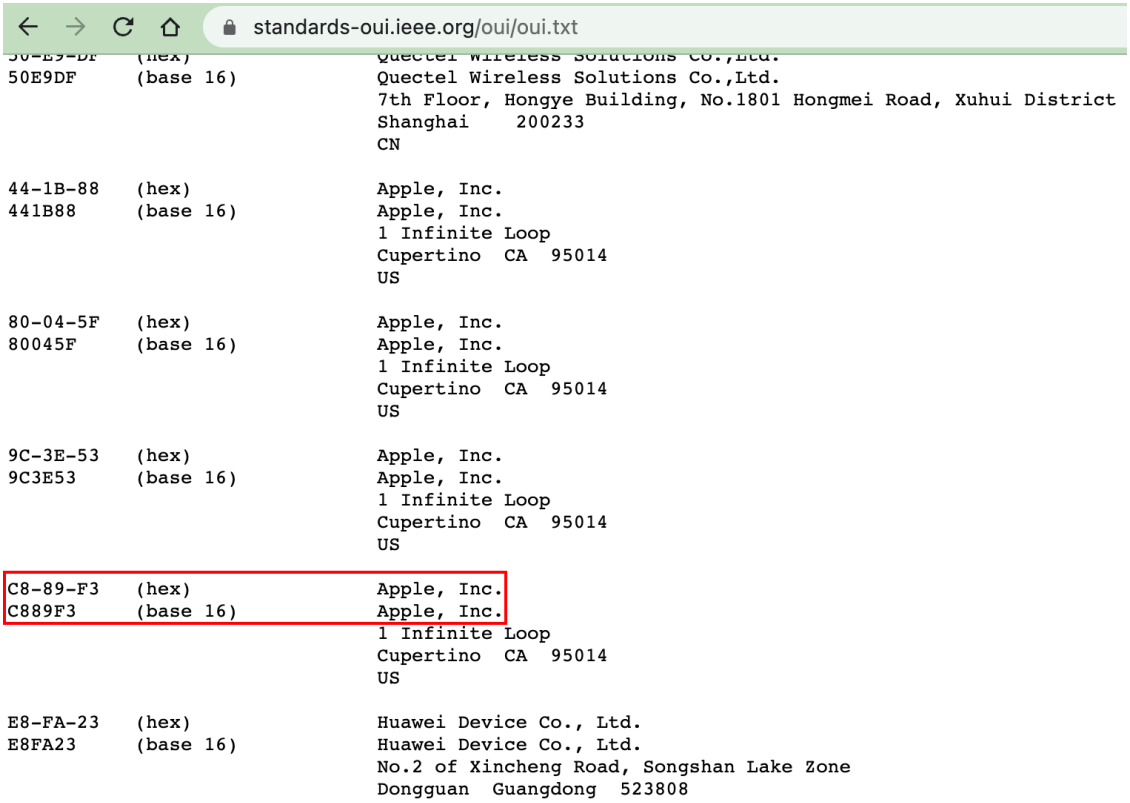
我们可以访问网站看到目前分配的大致情况：http://standards-oui.ieee.org/oui/oui.txt

笔者本地电脑的mac地址为：

```
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        options=6463<RXCSUM,TXCSUM,TSO4,TSO6,CHANNEL_IO,PARTIAL_CSUM,ZEROINVERT_
CSUM>
        ether c8:89:f3:ae:18:7b
        inet6 fe80::1ced:cbd9:a30a:24e0%en0 prefixlen 64 secured scopeid 0xe
        inet 192.168.31.16 netmask 0xffffff00 broadcast 192.168.31.255
        nd6 options=201<PERFORMNUD,DAD>
        media: autoselect
        status: active
bridge0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
```

那上面这个mac地址是如何查询出来的呢？对于很多读者来说，可能是十分简单的问题了，的确，即便没有专业学过计算机的人，只要倒腾过电脑，重装过系统，大多也会知道这个问题的答案：在 Windows 上是 ipconfig，在 Linux 上是 ifconfig。

上面说过，mac地址的前三个字节被各个网卡生产商买走，在上面提到的网址中我们也看到了所属信息：



在众多 MAC 地址中，有一个地址很特殊，其每一个二进制位都是 1，因此是：

> ff:ff:ff:ff:ff:ff

这个 MAC 地址被称为广播地址，广播的意思就是向大家播放，因此顾名思义，广播地址可以代表任意一个网卡，因此发向广播地址的信息就会发送到所在网络的所有网卡上。

四、以太网协议：第二层的语言

我们已经知道第二层用MAC地址来标识每一台机器，就好像我们已经知道对方的身份证号码了，下面我们就得以一个大家认可的语言跟他交流。对于第二层来说，机器之间互相通信的语言或者说协议就是“以太网协议”，只要遵循这个协议，两台机器之间才能互相理解。**以太网协议并不是第二层的唯一协议，但却是最常用的**。就好比英语不是唯一的语言，但却是国际上最常用的。

上面提到了协议，其实协议就是计算机网络世界里面的语言。为什么要有协议呢？我们知道网线中传输的都是0或1，类似于这样：

```
01101011101101010110011001
```

但是我们人类并不知道这样一串信息的意义啊。因此，以太网协议就是规定了什么样的信息会被传输，以及传输的顺序。在传输的信息里，至少要包含：

- 发送方的地址
- 接收方的地址
- 信息的实际内容

这样一个信息的总的单元，术语称为帧（Frame），也是我们之前的文章已经提前认识的重要名词。

4.1 帧

首先上面提到了两个地址，就是接收方和发送方的MAC地址（这里简述为MAC地址，但是要牢记MAC地址是跟网卡对应的）。

这里有个小问题，**到底哪个MAC地址在前面比较好呢？**

网络的先驱者们认为接收方的 MAC 地址对一台机器来说更有价值，因为可以立即得知信息是不是发给我们的。如果信息是发给我们的，那么我们阅读信息；如果不是发给我们的，那么大可不必理会。

还有我们知道，发送方发送消息需要穿过OSI七层模型：应用层-->传输层-->网络层-->数据链路层-->物理层。在穿梭到数据链路层的时候，已经经过了网络层（第3层），**那么网络层（第3层）就会告诉数据链路层（第2层）它所用的协议是什么了**。

接收方处理的时候顺序是：物理层-->数据链路层-->网络层-->传输层-->应用层。**接收方的机器的第 2 层首先检验 MAC 目标地址，如果和自己的 MAC 地址一样，那么接收方的机器的第 2 层需要将信息发送到第 3 层的对应协议**。

对了，我们在《数据链路层篇：差错检测问题》就提到了数据链路层可以使用CRC循环冗余校验算法来检测帧是否发生了误码。

CRC 是循环冗余校验（Cyclic Redundancy Check）的缩写，简单说来，CRC 对于每条发送的信息都是不一样的。发送方使用某公式计算出被传送数据所含信息的一个 CRC 值，并将此值附在被传送数据后，接收方则对接收到的同一数据进行相同的计算，得到另一个 CRC 值。

如果这两个 CRC 一致，说明发送中没有出错；如果不一致，说明发送中出现了差错，接收方可要求发送方重新发送该数据。

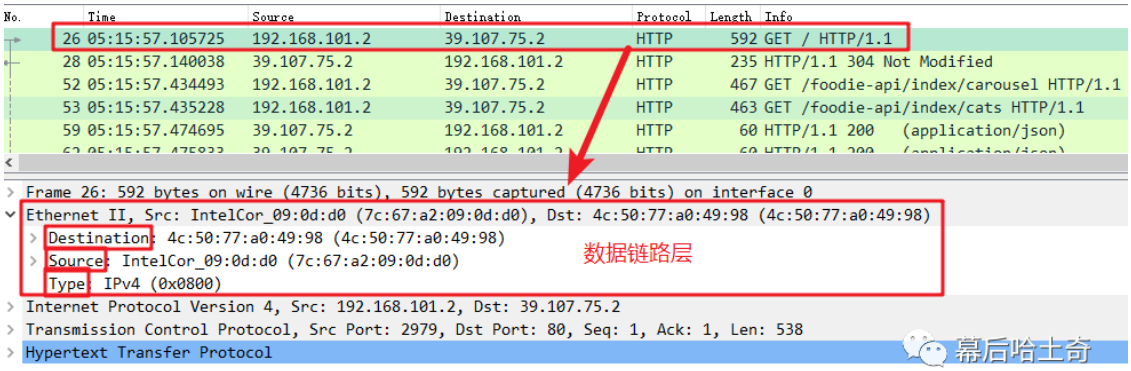
CRC校验原理之前已经详细说明过了，这里就不过多赘述了。

因此，以太网帧目前包含了如下信息：

DST 地址 （接收方 MAC）	SRC 地址 （发送方 MAC）	第 3 层 使用的协议	要发送的信息	CRC

4.2 实际抓个包看下数据链路层

这里用了wireshark尝试对某个http网站抓了一个包，wireshark的使用方法先不说明，我们直接来看某个请求的情况：

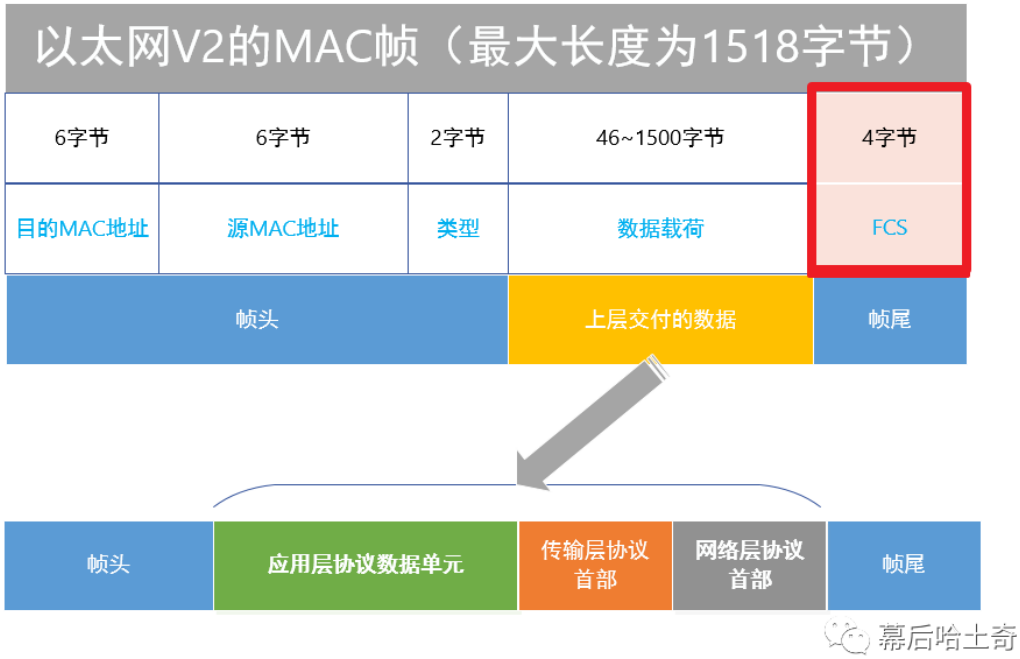


从实际抓包来看，Destination标识接收方MAC地址，Source标识发送方MAC地址，下面的Type标识上一层即网络层使用的是IPV4协议，确实跟我们上面说的一致，不过好像没看到CRC校验码，这是为什么呢?

此外，不知道读者朋友们是否还记得在《数据链路层篇：封装成帧问题》中介绍的前导码：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。抓包中也丝毫看不到此信息，究竟是为什么呢?

原来 Wireshark 抓包前，在物理层网卡已经去掉了一些之前几层加的东西，比如前导同步码，FCS等等，之后利用校验码CRC校验，正确时才会进行下一步操作，这时才开始进行抓包，因此，抓包软件抓到的是去掉前导同步码、FCS之外的数据，没有校验字段。

4.3 一帧的大小是多少呢?



帧分为帧头和实际消息。帧头即固定元素比如MAC地址、第三层协议以及CRC：

- 接收方和发送方的 MAC 地址分别占用 6 个字节；
- 第 3 层的协议用 2 个字节编码；
- CRC 用 4 个字节编码。

6 x 2 + 2 + 4 = 18。因此以太网的帧头一共有 18 个字节。

以太网帧的最小尺寸是 64 字节。那么一帧中最少报文长度为46字节。

以太网帧的最大尺寸是 1518 字节。那么一帧中最大报文长度为1500字节。

4.4 无效帧

在以太网中，无效帧有如下几种情况：

- MAC帧的长度不是整数个字节；
- 通过MAC帧的FCS字段的值检测出帧有误码；
- MAC帧的长度不在64～1518字节范围内。

结合目前所学，我们小结一下在机器 A 和机器 B 之间交换数据的过程如下：

- 机器 A 上的一个应用发送数据到机器 B 的应用上；
- 在机器 A 这个发送方这端，此数据从上到下穿过 OSI 的各层；

- 发送方的第 3 层告知第 2 层所使用的协议是什么；
- 第 2 层就用这些信息，包装成一个帧，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；
- 机器 B 收到了机器 A 发送的这个帧，首先检查帧头部的第一个元素：接收方的 MAC 地址；
- 如果等于机器 B 的 MAC 地址，那么机器 B 读取帧中接下来的信息；否则直接丢弃；
- 依据帧中的协议部分，接收方的第 2 层就把数据正确地发送给第 3 层；
- 数据在接收方就从下到上，直达机器 B 的应用了。

那么就知道在一个有线局域网中机器之间如何通信了。