1． ARP 的工作原理

我们都知道以太网设备比如网卡都有自己全球唯一的 MAC 地址，它们是以 MAC 地址来传输以太网数据包的，但是它们却识别不了我们 IP 包中的 IP 地址，所以我们在以太网中进行 IP 通信的时候就需要一个协议来建立 IP 地址与 MAC 地址的对应关系，以使 IP 数据包能发到一个确定的地方去。这就是 ARP(Address Resolution Protocol，地址解析协议)。

讲到此处，我们可以在命令行窗口中，输入 arp –a，来看一下效果，类似于这样的条目：

210.118.45.100　　00-0b-5f-e6-c5-d7　　dynamic

就是我们电脑里存储的关于 IP 地址与 MAC 地址的对应关系，dynamic 表示是临时存储在 ARP 缓存中的条目，过一段时间就会超时被删除(xp/2003 系统是 2 分钟)。

这样一来，比如我们的电脑要和一台机器比如 210.118.45.1 通信的时候，它会首先去检查 arp 缓存，查找是否有对应的 arp 条目，如果没有，它就会给这个以太网络发 ARP 请求包广播询问 210.118.45.1 的对应 MAC 地址，当然，网络中每台电脑都会收到这个请求包，但是它们发现 210.118.45.1 并非自己，就不会做出相应，而 210.118.45.1 就会给我们的电脑回复一个 ARP 应答包，告诉我们它的 MAC 地址是 xx-xx-xx-xx-xx-xx，于是我们电脑的 ARP 缓存就会相应刷新，多了这么一条：

210.118.45.1　　xx-xx-xx-xx-xx-xx　　dynamic

为什么要有这么一个 ARP 缓存呢，试想一下如果没有缓存，我们每发一个 IP 包都要发个广播查询地址，岂不是又浪费带宽又浪费资源？

而且我们的网络设备是无法识别 ARP 包的真伪的，如果我们按照 ARP 的格式来发送数据包，只要信息有效计算机就会根据包中的内容做相应的反应。

2． ARP 包的格式

从网络底层看来，一个 ARP 包是分为两个部分的，前面一个是物理帧头，后面一个才是 ARP 帧。

首先，物理帧头，它将存在于任何一个协议数据包的前面，我们称之为 DLC Header，因为这个帧头是在数据链路层构造的，并且其主要内容为收发双方的物理地址，以便硬件设备识别。

DLC Header			
字段	长度(Byte)	默认值	备注
接收方 MAC	6		广播时，为 ff-ff-ff-ff-ff-ff
发送方 MAC	6		
Ethertype	2	0x0806	0x0806 是 ARP 帧的类型值

图 1 物理帧头格式

图 1 是需要我们填充的物理帧头的格式，我们可以看到需要我们填充的仅仅是发送端和接收端的物理地址罢了，是不是很简单呢？

接下来我们看一下 ARP 帧的格式：

ARP Frame			
字段	长度 (Byte)	默认值	备注
硬件类型	2	0x1	以太网类型值
上层协议类型	2	0x0800	上层协议为 IP 协议
MAC 地址长度	1	0x6	以太网 MAC 地址长度为 6
IP 地址长度	1	0x4	IP 地址长度为 4
操作码	2		0x1 表示 ARP 请求包,0x2 表示应答包
发送方 MAC	6		
发送方 IP	4		
接收方 MAC	6		
接收方 IP	4		
填充数据	18		因为物理帧最小长度为 64 字节,前面的 42 字节再加上 4 个 CRC 校验字节,还差 18 个字节

图 2 ARP 帧格式

我们可以看到需要我们填充的同样也只是 MAC，IP，再加上一个 1 或 2 的操作码而已。

3． ARP 包的填充

1) 请求包的填充：

比如我们的电脑 MAC 地址为 aa-aa-aa-aa-aa-aa，IP 为 192.168.0.1

我们想要查询 192.168.0.99 的 MAC 地址，应该怎么来做呢？

首先填充 DLC Header，通过前面的学习我们知道，想要知道某个计算机对应的 MAC 地址是要给全网发送广播的，所以接收方 MAC 肯定是 ffffffffffff，发送方 MAC 当然是自己啦，于是我们的 DLC Header 就填充完成了，如图，加粗的是我们要手动输入的值(当然我编的程序比较智能，会根据你选择的 ARP 包类型帮你自动填入一些字段，你一用便知^_^)。

DLC Header		
字段	长度 (Byte)	填充值

接收方 MAC	6	ffffffffffff
发送方 MAC	6	aaaaaaaaaaaa
Ethertype	2	0x0806

图 3 ARP 请求包中 DLC Header 内容

接下来是 ARP 帧，请求包的操作码当然是 1，发送方的 MAC 以及 IP 当然填入我们自己的，然后要注意一下，这里的接收方 IP 填入我们要查询的那个 IP 地址，就是 192.168.0.99 了，而接收方 MAC 填入任意值就行，不起作用，于是，如图，

ARP Frame		
字段	长度 (Byte)	填充值
硬件类型	2	1
上层协议类型	2	0800
MAC 地址长度	1	6
IP 地址长度	1	4
操作码	2	1
发送方 MAC	6	aaaaaaaaaaaa
发送方 IP	4	192.168.0.1
接收方 MAC	6	任意值 xxxxxxxxxxxx
接收方 IP	4	192.168.0.99
填充数据	18	0

图 4 ARP 请求包中 ARP 帧的内容

如果我们构造一个这样的包发送出去，如果 192.168.0.99 存在且是活动的，我们马上就会收到一个 192.168.0.99 发来的一个响应包，我们可以查看一下我们的 ARP 缓存列表，是不是多了一项类似这样的条目：

192.168.0.99 bb-bb-bb-bb-bb-bb

我们再来看一下 ARP 响应包的构造

2) 响应包的填充

有了前面详细的解说，你肯定就能自己说出响应包的填充方法来了吧，所以我就不细说了，列两个表就好了

比如说给 192.168.0.99（MAC 为 bb-bb-bb-bb-bb-bb）发一个 ARP 响应包，告诉它我们的 MAC 地址为 aa-aa-aa-aa-aa-aa，就是如此来填充各个字段

DLC Header		
字段	长度 (Byte)	填充值
接收方 MAC	6	bbbbbbbbbbbb
发送方 MAC	6	aaaaaaaaaaaa
Ethertype	2	0x0806

图 5 ARP 响应包中 DLC Header 内容

ARP Frame		
字段	长度 (Byte)	填充值
硬件类型	2	1
上层协议类型	2	0800
MAC 地址长度	1	6
IP 地址长度	1	4
操作码	2	2
发送方 MAC	6	aaaaaaaaaaaa
发送方 IP	4	192.168.0.1
接收方 MAC	6	bbbbbbbbbbbb
接收方 IP	4	192.168.0.99
填充数据	18	0

图 6 ARP 响应包中 ARP 帧的内容

这样 192.168.0.99 的 ARP 缓存中就会多了一条关于我们 192.168.0.1 的地址映射。