

**学院：计算机学院**

**马天成**

**2022年6月13日**

**计算机网络实验报告**

**[网络层数据报分析]**

**北京邮电大学《计算机网络》课程实验报告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **实验**  **名称** | 网络层数据报分析 | | **学 院** | 计算机 | **指导教师** | 蒋砚军 |
| **班 级** | **班内序号** | **学 号** | | **学生姓名** | **成绩** | |
| **2020211305** |  | **2020211376** | | **马天成** |  | |
| **实**  **验**  **内**  **容** | 本次实验主要包含下列内容：  l)使用Wireshark软件捕获在使用ping命令时产生的ICMP消息：  2)分析网络层P包头格式，理解各字段的作用，对于分段和校验和进行验证：  3)使用Vireshark软件捕获在使用ARP消息，分析其消息格式，理解其工作原理：  4)使用Wireshark捕获DHCP消息，分析其消息序列，理解DHCP的功能和操作原理：  5)使用Wireshark捕获TCP消息，分析TCP报文段头格式，理解连接建立和释放的原理，差错控制原理、序号和窗口管理的原理。通过本实验学生可以深入理解分层体系结构，理解和掌握TCP/IP协议栈的代表协议-P、TCP、UDP、ICMP、ARP和DHCP协议的要点。 | | | | | |
| **学生实验报告** | （详见“实验报告和源程序”册） | | | | | |
| **课**  **程**  **设**  **计**  **成**  **绩**  **评**  **定** | **评语**:  **成绩**:  指导教师签名：  年 月 日 | | | | | |

注：评语要体现每个学生的工作情况，可以加页。

目录

[本机条件 5](#_Toc105976532)

[本人PC网络适配器参数 5](#_Toc105976533)

[本人服务器公网地址 5](#_Toc105976534)

[理论准备 6](#_Toc105976535)

[TCP/IP架构和帧格式 6](#_Toc105976536)

[IP – Internet网络层（IPv4，IPv6） 7](#_Toc105976537)

[IP包报文格式详解 7](#_Toc105976538)

[捕获IP分组 7](#_Toc105976539)

[解析单个IP包 7](#_Toc105976540)

[捕获长IP分组 8](#_Toc105976541)

[分析一次请求中的第一个IPv4包 9](#_Toc105976542)

[分析一次请求中的最后一个IP v4包 10](#_Toc105976543)

[分析一次请求中最后一个ICMP包 11](#_Toc105976544)

[整体分析 11](#_Toc105976545)

[IP 包头校验和 12](#_Toc105976546)

[IP分段原理 12](#_Toc105976547)

[ICMP-Internet控制消息协议 12](#_Toc105976548)

[ICMP帧格式 13](#_Toc105976549)

[ICMP帧分析 13](#_Toc105976550)

[DHCP-动态主机配置协议 14](#_Toc105976551)

[DHCP报文格式详解 15](#_Toc105976552)

[DHCP释放过程 16](#_Toc105976553)

[以DHCP Release 包来讲解分组 17](#_Toc105976554)

[DHCP建立过程 19](#_Toc105976555)

[第一个包：DHCP Discover包 19](#_Toc105976556)

[第二个包：DHCP Offer包 21](#_Toc105976557)

[第三个包：DHCP REQUEST包 22](#_Toc105976558)

[第四个包：DHCP ACK包 24](#_Toc105976559)

[四次握手结果 26](#_Toc105976560)

[DHCP包理解 26](#_Toc105976561)

[思考：虚拟机IP地址的配置过程 26](#_Toc105976562)

[ARP–地址解析协议 26](#_Toc105976563)

[ARP报文格式详解 27](#_Toc105976564)

[捕获ARP包 27](#_Toc105976565)

[分析ARP包（以本机发送为例） 28](#_Toc105976566)

[ARP工作原理和作用 29](#_Toc105976567)

[TCP连接过程 30](#_Toc105976568)

[TCP的目的 30](#_Toc105976569)

[TCP的做法 30](#_Toc105976570)

[TCP报文格式 31](#_Toc105976571)

[追踪TCP包 31](#_Toc105976572)

[第一次握手 32](#_Toc105976573)

[第二次握手 34](#_Toc105976574)

[第三次握手 35](#_Toc105976575)

[TCP释放连接 35](#_Toc105976576)

[第一次挥手 36](#_Toc105976577)

[第二次挥手 36](#_Toc105976578)

[第三次挥手 37](#_Toc105976579)

[第四次挥手 38](#_Toc105976580)

[四次挥手整体 38](#_Toc105976581)

[Nginx反向代理的四次挥手？ 39](#_Toc105976582)

[架构理解 39](#_Toc105976583)

[以太网帧结构 39](#_Toc105976584)

[IP包帧结构 40](#_Toc105976585)

[TCP帧结构 40](#_Toc105976586)

[UDP帧格式 41](#_Toc105976587)

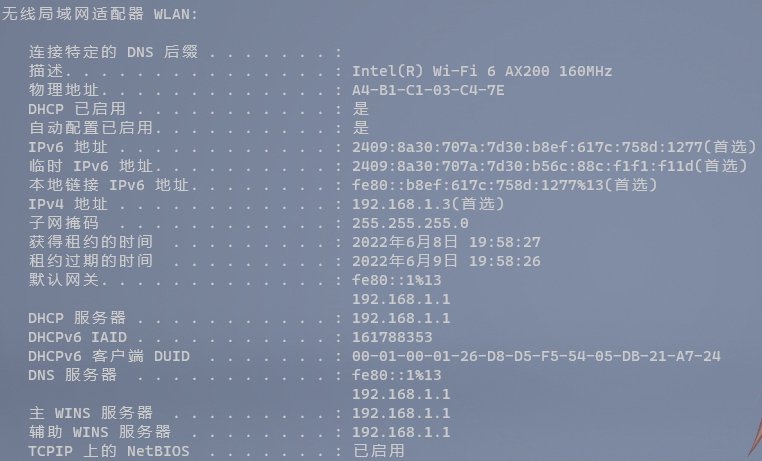
[实验总结 41](#_Toc105976588)

[实验问题 41](#_Toc105976589)

[实验心得 41](#_Toc105976590)

# 本机条件

## 本人PC网络适配器参数





## 本人服务器公网地址

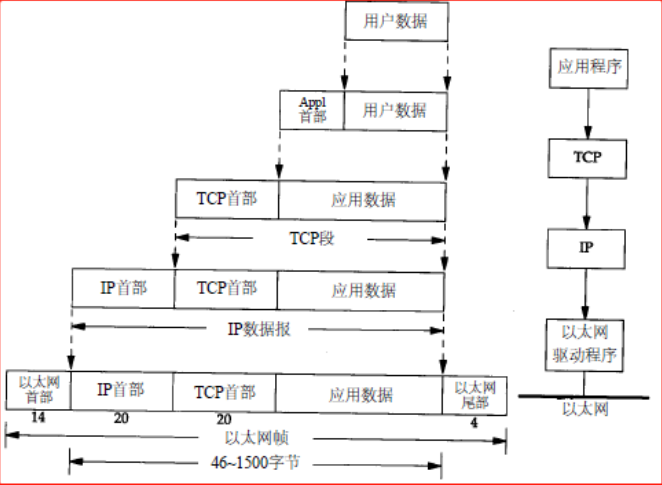
124.220.8.25（没买域名）



已提前确定网络正常连接。

# 理论准备

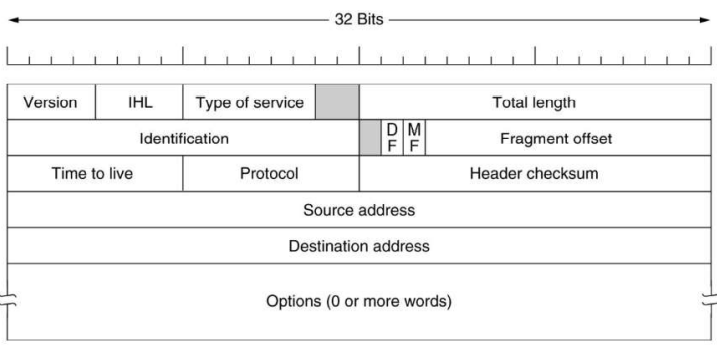
## TCP/IP架构和帧格式



在LAN内，package基本都会以以太网帧头开始，然后在进行后续的拆包。

# IP – Internet网络层（IPv4，IPv6）

## IP包报文格式详解

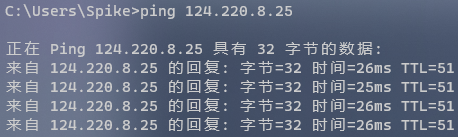


这个在书上已经详细学过，这里不多再赘述。

（若是IPv6，那么头部将会是40Bytes）

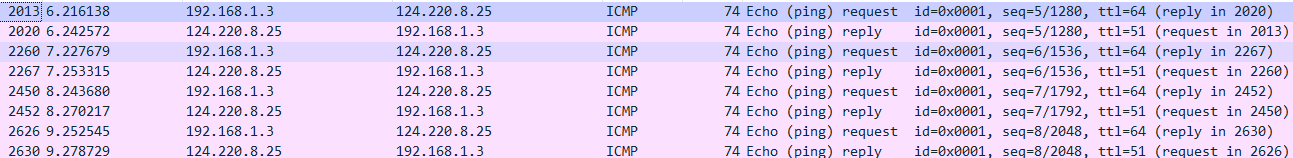
## 捕获IP分组

在过滤器输入ip.dst eq 124.220.8.25 or ip.src eq 124.220.8.25（本人服务器地址），即向我的服务武器进行ping指令。



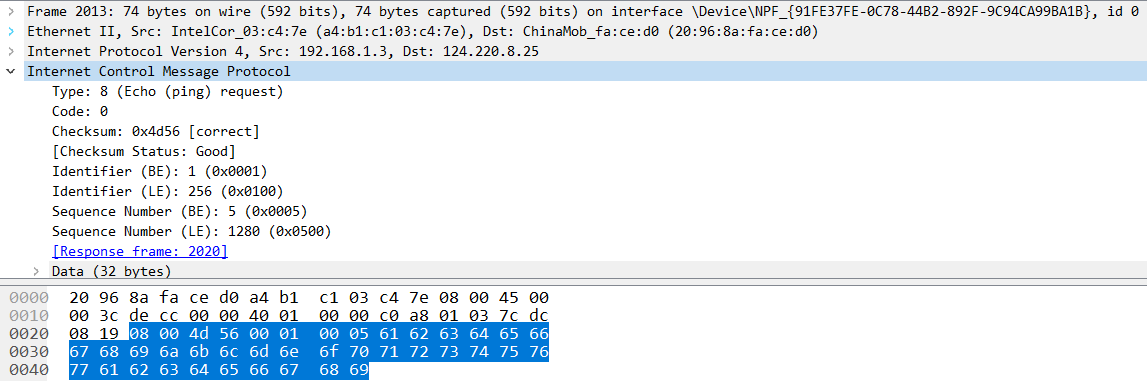
可以看到一共进行了四次包的申请

所以在wireshark中可以获得4组（8个）请求/回应的包：



## 解析单个IP包

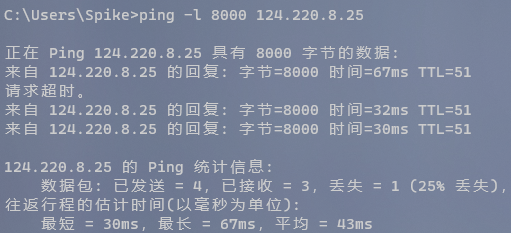
以第一个IP包为例：



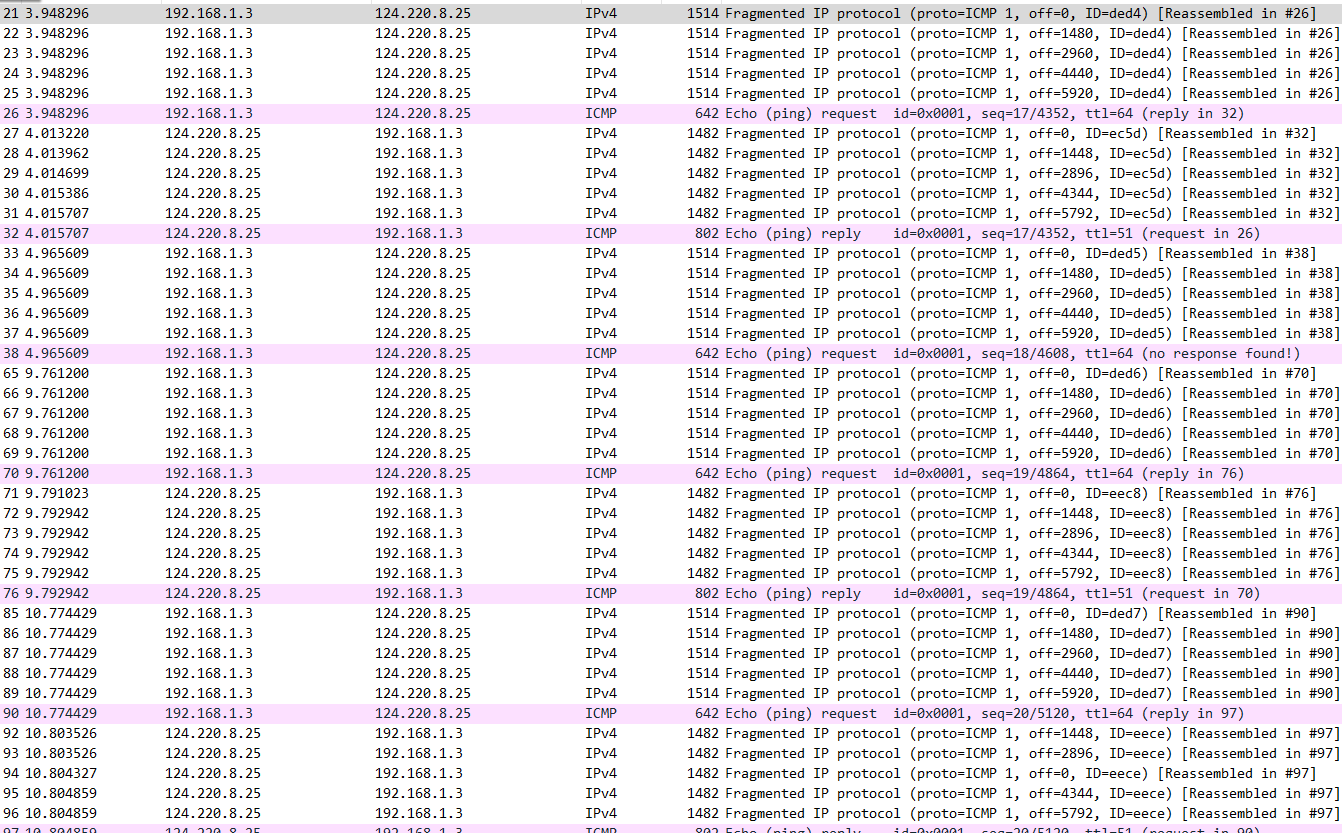
|  |  |  |
| --- | --- | --- |
| 字段 | 数值 | 属性 |
| 以太网帧头 | 前14字节 | 以太网传输必备。因为不需要包装TCP，直接跟着IP包 |
| 版本(4 bit) | 4 | 0100表示IP版本4 |
| IHL(4 bit) | 5 | IP头部长度，这里为20字节（一行四字节，5行） |
| 服务类型 | 00 | 正常时延，正常吞吐量，正常可靠性 |
| 总长度 | 00 3c | 数据分组长60字节 |
| 标识 | de cc | 独特的包标识 |
| DF(1 bit) | 0 | 允许分段（路由分段，透明，非透明） |
| MF(1 bit) | 0 | 更多的段。这里是0，代表最后一包 |
| 片位移(13 bit) | 00 00 | 这里是最后一包且没有位移，所以不分段 |
| 生存周期 | 40 | 64跳，一跳一秒 |
| 协议 | 01 | 用的是ICMP协议 |
| 头部校验和 | 00 00 | 校验正确 |
| 源地址 | c0 a8 01 03 | 192.168.1.3 |
| 目的地址 | 7c dc 08 19 | 124.220.8.25 |
| 包内容 | 40字节 | 跟了40字节的数据。 |

## 捕获长IP分组

因为上述ping的指令并没有发生分组，所以发送一个大的-8k字节的包。

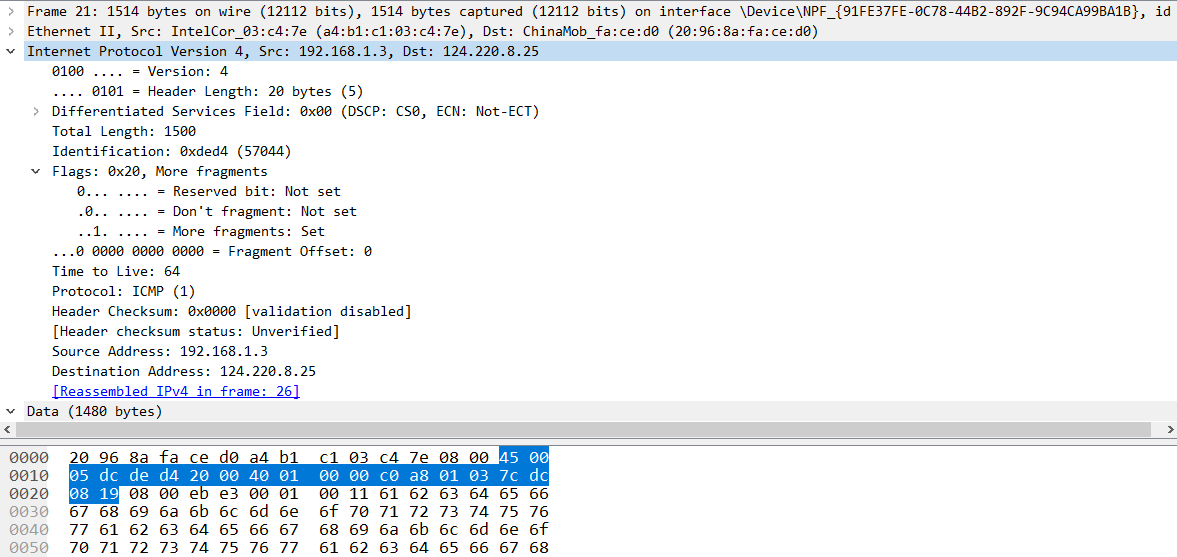


用过滤语句过滤出一下包（小水管2M/s，发的包可能比较多）



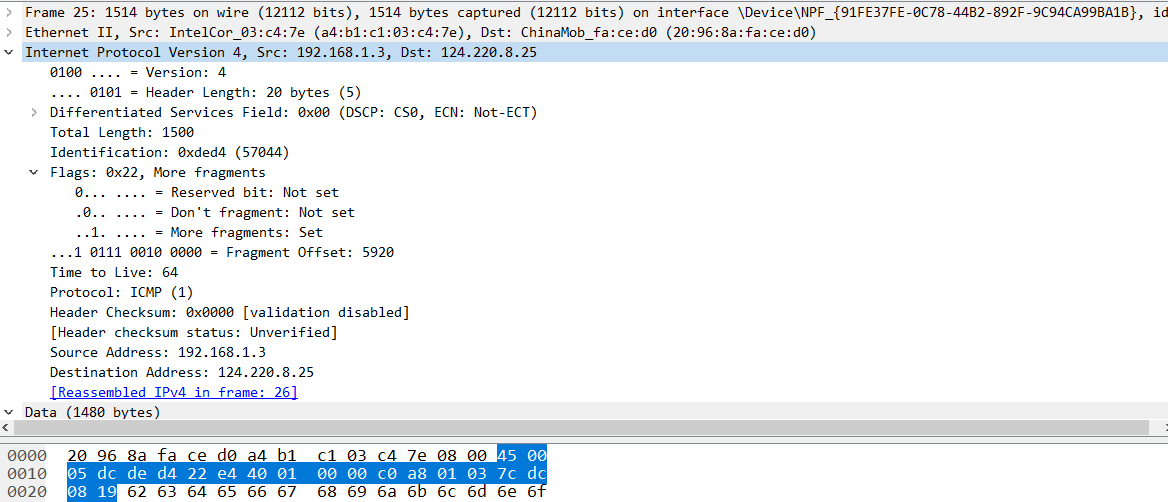
* 可以看到，一共执行了四次请求，发送了4个8000字节分组，产生了四组消息：
* 1，3，4组有发送有回应；
* 第2组没有回应，只有发送。
* 发送过去的是1514字节，其中包括：
  + 最外层的以太网帧头-14Bytes；
  + 还有一层IP帧头-20Bytes。

### 分析一次请求中的第一个IPv4包



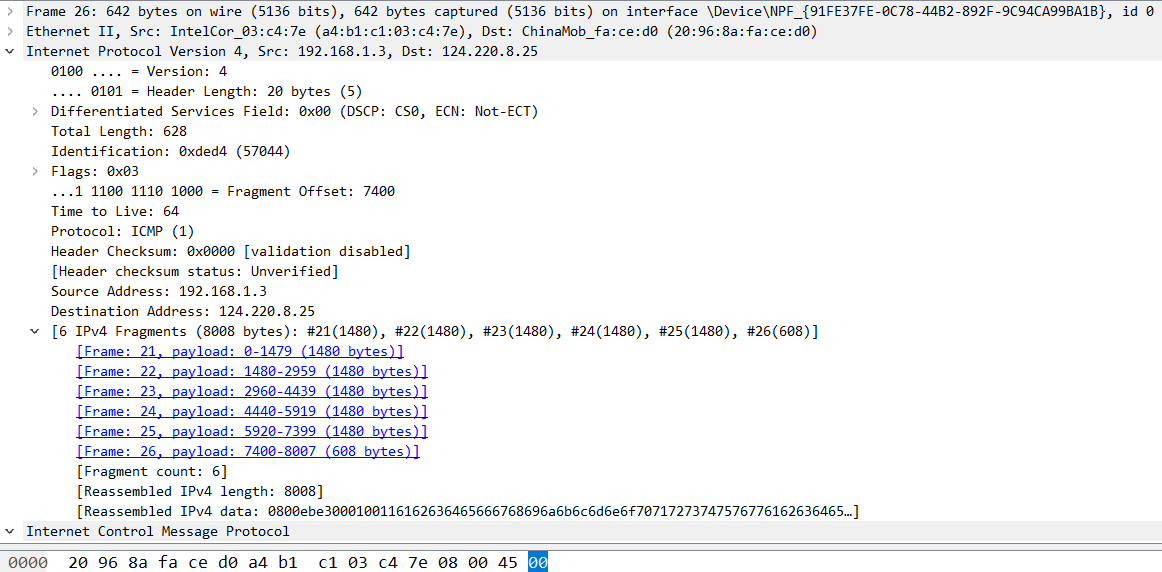
|  |  |  |
| --- | --- | --- |
| 字段 | 数值 | 属性 |
| 以太网帧头 | 前14字节 | 以太网传输必备。因为不需要包装TCP，直接跟着IP包 |
| 版本(4 bit) | 4 | 0100表示IP版本4 |
| IHL(4 bit) | 5 | IP头部长度，这里为20字节（一行四字节，5行） |
| 服务类型 | 00 | 正常时延，正常吞吐量，正常可靠性 |
| 总长度 | **05 dc** | **数据分组长1500字节（IP头占20）** |
| 标识 | de d4 | 独特的包标识 |
| DF(1 bit) | 0 | 允许分段（路由分段，透明，非透明） |
| MF(1 bit) | **1** | **更多的段。这里是1，代表分段且不是最后一段** |
| 片位移(13 bit) | **00 00** | **这里是第一个包且没有位移** |
| 生存周期 | 40 | 64跳，一跳一秒 |
| 协议 | 01 | 用的是ICMP协议 |
| 头部校验和 | 00 00 | 校验正确 |
| 源地址 | c0 a8 01 03 | 192.168.1.3 |
| 目的地址 | 7c dc 08 19 | 124.220.8.25 |
| 内容 | 1480字节 | 跟了**1480**字节的数据（1500-20） |

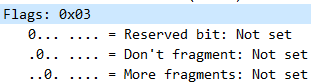
### 分析一次请求中的最后一个IP v4包



|  |  |  |
| --- | --- | --- |
| 字段 | 数值 | 属性 |
| 以太网帧头 | 前14字节 | 以太网传输必备。因为不需要包装TCP，直接跟着IP包 |
| 版本(4 bit) | 4 | 0100表示IP版本4 |
| IHL(4 bit) | 5 | IP头部长度，这里为20字节（一行四字节，5行） |
| 服务类型 | 00 | 正常时延，正常吞吐量，正常可靠性 |
| 总长度 | **05 dc** | **数据分组长1500字节（IP头占20）** |
| 标识 | de d4 | 独特的包标识**（和上面所有的相同，表示同一个数据包）** |
| DF(1 bit) | 0 | 允许分段（路由分段，透明，非透明） |
| MF(1 bit) | **1** | **更多的段。这里是1，代表分段且不是最后一段** |
| 片位移(13 bit) | **22 e4** | **这里是第五包，位移是5920（1480\*4）** |
| 生存周期 | 40 | 64跳，一跳一秒 |
| 协议 | 01 | 用的是ICMP协议 |
| 头部校验和 | 00 00 | 校验正确 |
| 源地址 | c0 a8 01 03 | 192.168.1.3 |
| 目的地址 | 7c dc 08 19 | 124.220.8.25 |
| 内容 | 1480字节 | 跟了1480字节的数据 |

### 分析一次请求中最后一个ICMP包



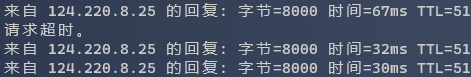


|  |  |  |
| --- | --- | --- |
| 字段 | 数值 | 属性 |
| 以太网帧头 | 前14字节 | 以太网传输必备。因为不需要包装TCP，直接跟着IP包 |
| 版本(4 bit) | 4 | 0100表示IP版本4 |
| IHL(4 bit) | 5 | IP头部长度，这里为20字节（一行四字节，5行） |
| 服务类型 | 00 | 正常时延，正常吞吐量，正常可靠性 |
| 总长度 | **02 47** | **数据分组长628字节（IP头占20）**  **628 = 8000 -5\*1480 + 20 + 8（ICMP多出来的两行）** |
| 标识 | de d4 | 独特的包标识**（和上面所有的相同，表示同一个数据包）** |
| DF(1 bit) | 0 | 允许分段（路由分段，透明，非透明） |
| MF(1 bit) | **0** | **更多的段。这里是0，代表分段是最后一段** |
| 片位移(13 bit) | **22 e4** | **这里是第五包，位移是5920（1480\*4）** |
| 生存周期 | 40 | 64跳，一跳一秒 |
| 协议 | 01 | 用的是ICMP协议 |
| 头部校验和 | 00 00 | 校验正确 |
| 源地址 | c0 a8 01 03 | 192.168.1.3 |
| 目的地址 | 7c dc 08 19 | 124.220.8.25 |
| 内容 | 1480字节 | 跟了1480字节的数据 |

### 整体分析

一共35个包，三组回应。（因为上面写了25%丢包率）

所以第一个，第三个和第四个8000kb正常发送过去且收到回应。



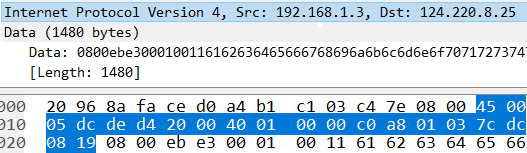
所以在一次发送过程中，**8000字节被分成了6个包进行发送：**

* 前五个是最大的IP包，每个包1500Bytes(头部占20Bytes)
* 最后一个是ICMP包，携带了剩下的600Bytes，且头部为28Bytes（IP20+额外8）
* 标识符相同，属于同一个包

## IP 包头校验和

如果没有在‘选项’里面添加内容，那么前面20字节为IP数据包的首部，IP校验和就是对这10个字（也就是20个字节）求的校验和，校验和说白了就是各个二进制数按位取反再求和（或者求和再按位取反），要注意的是这里的数没有正负之分。求得的校验和放在16bit的校验和字段里面，由于该字段在计算校验和的时候也参与了，为保证校验和计算不受影响，计算之前必须将该字段设置成16bit的0。

当接收到整个IP数据报的时候，再对实际接收到的IP数据报的首部的20个字进行校验和计算，如果数据传输无误，则校验和所有位应该为1（**因为除开校验和字段的16bit的校验和结果应该与校验和字段互为一对反码**）。



验证如下：

把其他16个字节相加校验取反，为00 00。

## IP分段原理

**注：总长度是剔除掉以太网帧包头的14字节帧头**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 序号 | 21 | 22 | 23 | 24 | 25 | 26 |
| DF | 0 | 0 | 0 | 0 | 0 | 0 |
| MF | 1 | 1 | 1 | 1 | 1 | 0 |
| 偏移量 | 0 | 1480 | 2960 | 4440 | 5920 | 7400 |
| 总长度 | 1500 | 1500 | 1500 | 1500 | 1500 | 1500 |
| 数据长度 | 1480 | 1480 | 1480 | 1480 | 1480 | 1480 |

实际上就是因为一个包太大，一个帧最长为1514字节，所以要分段。

# ICMP-Internet控制消息协议

ICMP 的全称是 Internet Control Message Protocol(互联网控制协议)，它是一种互联网套件，它用于**IP 协议中发送控制消息**。也就是说，ICMP 是依靠 IP 协议来完成信息发送的，它是 IP 的主要部分，但是从体系结构上来讲，它位于 IP 之上，因为 ICMP 报文是承载在 IP 分组中的，就和 TCP 与 UDP 报文段作为 IP 有效载荷被承载那样。这也就是说，当主机收到一个指明上层协议为 ICMP 的 IP 数据报时，它会分解出该数据报的内容给 ICMP，就像分解数据报的内容给 TCP 和 UDP 一样。

## ICMP帧格式



可以看到ICMP在IP包内，又增加了一个包头，其为8字节：

1.   类型：占8位

2.   代码：占8位

3.   检验和：占16位

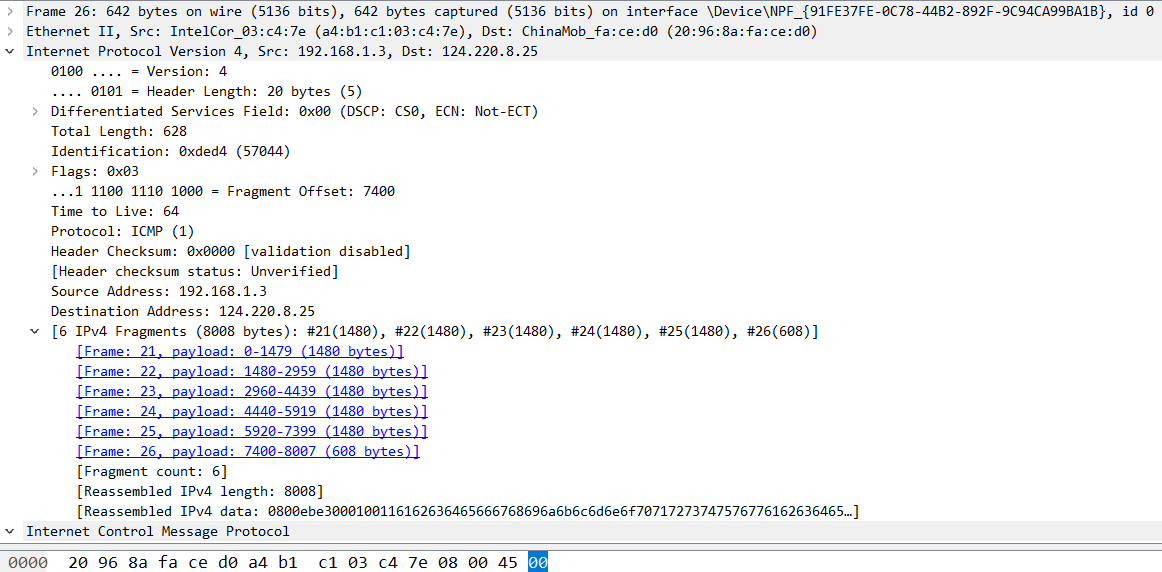
**说明：ICMP所有报文的前4个字节都是一样的，但是剩下的其他字节则互不相同。**

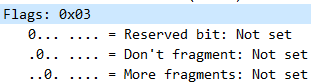
4.   其它字段都ICMP报文类型不同而不同。

* ICMP报文的前4个字节是统一的格式，共有三个字段：即类型，代码和检验和。
* 8位类型和8位代码字段一起决定了ICMP报文的类型。
  + 类型8，代码0：表示回显请求(ping请求)。
  + 类型0，代码0：表示回显应答(ping应答)
  + 类型11，代码0：超时
* 16位的检验和字段：包括数据在内的整个ICMP数据包的检验和；其计算方法和IP头部检验和的计算方法一样的。
* ICMP报文具体分为查询报文和差错报文(对ICMP差错报文有时需要做特殊处理，因此要对其进行区分。如：对ICMP差错报文进行响应时，永远不会生成另一份ICMP差错报文，否则会出现死循环)

## ICMP帧分析

（这里直接照搬IP分段中的ICMP）





|  |  |  |
| --- | --- | --- |
| 字段 | 数值 | 属性 |
| 以太网帧头 | 前14字节 | 以太网传输必备。因为不需要包装TCP，直接跟着IP包 |
| 版本(4 bit) | 4 | 0100表示IP版本4 |
| IHL(4 bit) | 5 | IP头部长度，这里为20字节（一行四字节，5行） |
| 服务类型 | 00 | 正常时延，正常吞吐量，正常可靠性 |
| 总长度 | **02 47** | **数据分组长628字节（IP头占20）**  **628 = 8000 -5\*1480 + 20 + 8（ICMP多出来的两行）** |
| 标识 | de d4 | 独特的包标识**（和上面所有的相同，表示同一个数据包）** |
| DF(1 bit) | 0 | 允许分段（路由分段，透明，非透明） |
| MF(1 bit) | **0** | **更多的段。这里是0，代表分段是最后一段** |
| 片位移(13 bit) | **22 e4** | **这里是第五包，位移是5920（1480\*4）** |
| 生存周期 | 40 | 64跳，一跳一秒 |
| 协议 | 01 | 用的是ICMP协议 |
| 头部校验和 | 00 00 | 校验正确 |
| 源地址 | c0 a8 01 03 | 192.168.1.3 |
| 目的地址 | 7c dc 08 19 | 124.220.8.25 |
| 内容 | 1480字节 | 跟了1480字节的数据 |

# DHCP-动态主机配置协议

**DHCP**（Dynamic Host Configuration Protocol，动态主机配置协议），前身是BOOTP协议，是一个局域网的网络协议，使用UDP协议工作，统一使用两个IANA分配的端口：67（服务器端），68（客户端）。DHCP通常被用于局域网环境，主要作用是集中的管理、分配IP地址，使client动态的获得IP地址、Gateway地址、DNS服务器地址等信息，并能够提升地址的使用率。简单来说，DHCP就是一个不需要账号密码登录的、自动给内网机器分配IP地址等信息的协议。

## DHCP报文格式详解



首先引入报文格式：

* **op**：报文的操作类型。分为请求报文和响应报文。1：请求报文，2：应答报文。即client送给server的封包，设为1，反之为2。
* 请求报文：**DHCP Discover、DHCP Request、DHCP Release、DHCP Inform和DHCP Decline**。
* 应答报文：**DHCP Offer、DHCP ACK和DHCP NAK**。
* **htype**：DHCP客户端的MAC地址类型。MAC地址类型其实是指明网络类型。htype值为1时表示为最常见的以太网MAC地址类型。
* **hlen**：DHCP客户端的MAC地址长度。以太网MAC地址长度为6个字节，即以太网时hlen值为6。
* **hops**：DHCP报文经过的DHCP中继的数目，默认为0。DHCP请求报文每经过一个DHCP中继，该字段就会增加1。没有经过DHCP中继时值为0。(若数据包需经过router传送，每站加1，若在同一网内，为0。)
* **xid**：客户端通过DHCP Discover报文发起一次IP地址请求时选择的随机数，相当于请求标识。用来标识一次IP地址请求过程。在一次请求中所有报文的Xid都是一样的。
* **secs**：DHCP客户端从获取到IP地址或者续约过程开始到现在所消耗的时间，以秒为单位。在没有获得IP地址前该字段始终为0。(DHCP客户端开始DHCP请求后所经过的时间。目前尚未使用，固定为0。)
* **flags**：标志位，只使用第0比特位，是广播应答标识位，用来标识DHCP服务器应答报文是采用单播还是广播发送，0表示采用单播发送方式，1表示采用广播发送方式。其余位尚未使用。(即从0-15bits，最左1bit为1时表示server将以广播方式传送封包给client。)
  + 注意：在客户端正式分配了IP地址之前的第一次IP地址请求过程中，所有DHCP报文都是以广播方式发送的，包括客户端发送的DHCP Discover和DHCP Request报文，以及DHCP服务器发送的DHCP Offer、DHCP ACK和DHCP NAK报文。当然，如果是由DHCP中继器转的报文，则都是以单播方式发送的。另外，IP地址续约、IP地址释放的相关报文都是采用单播方式进行发送的。
* **ciaddr**：DHCP客户端的IP地址。仅在DHCP服务器发送的ACK报文中显示，因为在得到DHCP服务器确认前，DHCP客户端是还没有分配到IP地址的。在其他报文中均显示，只有客户端是Bound、Renew、Rebinding状态，并且能响应ARP请求时，才能被填充。
* **yiaddr**：DHCP服务器分配给客户端的IP地址。仅在DHCP服务器发送的Offer和ACK报文中显示，其他报文中显示为0。
* **siaddr**：下一个为DHCP客户端分配IP地址等信息的DHCP服务器IP地址。仅在DHCP Offer、DHCP ACK报文中显示，其他报文中显示为0。(用于bootstrap过程中的IP地址)
  + 一般来说是服务器的ip地址.但是注意！根据openwrt源码给出的注释，当报文的源地址、siaddr、option­>server\_id字段不一致（有经过跨子网转发）时，通常认为option­>srever\_id字段为真正的服务器ip，siaddr有可能是多次路由跳转中的某一个路由的ip
* **giaddr**：DHCP客户端发出请求报文后经过的第一个DHCP中继的IP地址。如果没有经过DHCP中继，则显示为0。(转发代理（网关）IP地址)
* **chaddr**：DHCP客户端的MAC地址。在每个报文中都会显示对应DHCP客户端的MAC地址。
* **sname**：为DHCP客户端分配IP地址的DHCP服务器名称（DNS域名格式）。在Offer和ACK报文中显示发送报文的DHCP服务器名称，其他报文显示为0。
* **file**：DHCP服务器为DHCP客户端指定的启动配置文件名称及路径信息。仅在DHCP Offer报文中显示，其他报文中显示为空。
* **options**：可选项字段，长度可变，格式为"代码+长度+数据"。

**下面分析时会重点分析OP字段。**

## DHCP释放过程

命令行敲下**ipconfig/release**，产生一个包：



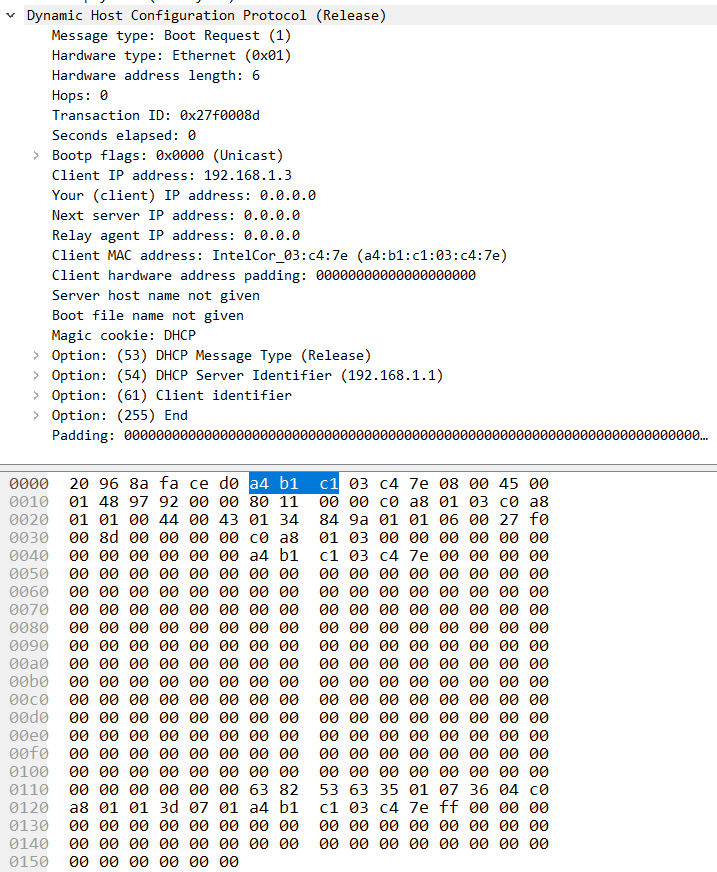
按名字可知，这是一个释放信号。

而且易知，在LAN内，我的局域网IP为192.168.1.3。在命令行验证：



在这之后，我的IP在路由器上被释放掉，即我的以太网物理地址在DHCP表项中不存在了，**只有重新申请才能获得新的动态IP。**

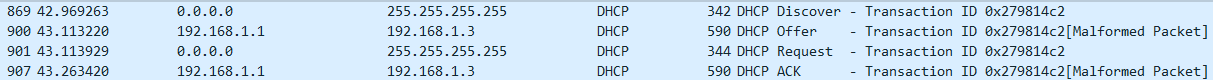
### 以DHCP Release 包来讲解分组



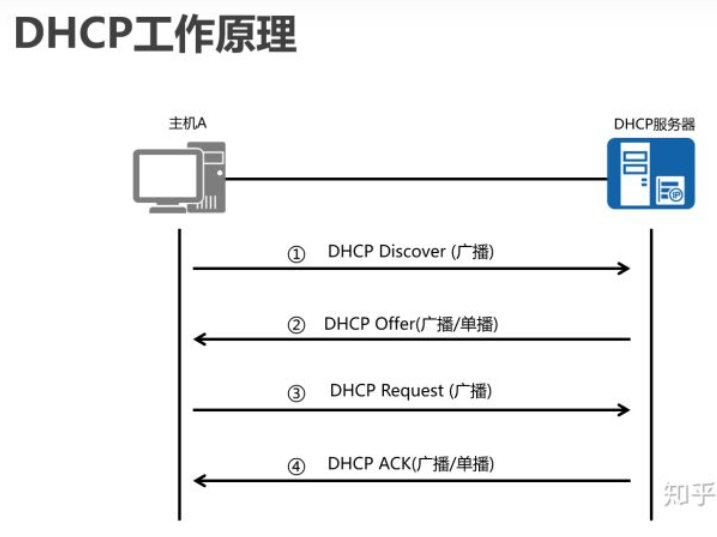
|  |  |  |
| --- | --- | --- |
| 单元的字节数 | 值（十六进制） | 解释 |
| 14 | 略 | 以太网报头 |
| 20 | 略 | I P报头 |
| 2 | 00 44 | UDP报头的开始。  源端口字段：包含发送数据报的应用程序使用的UDP端口。接收端的应用程序将其作为发送响应的目的地址。如果设为全0，则接收端不能发送回复。此处为68 |
| 2 | 00 43 | 目的端口字段：接收瑞计算机上UDP软件使用的端口 |
| 2 | 01 34 | 长度：UDP报头和数据的总长度，最小为8字节 |
| 2 | 84 9a | 校验和：进行检验和计算时，与伪首部一起运算。 |
| 1 | 01 | 报文的操作类型：分为请求报文和响应报文，=1，请求报文：=2，应答报文。此处为1，表示请求报文 |
| 1 | 01 | DHCP客户端的MAC地址类型：通常为1，表示以太网的MAC地址。 |
| 1 | 06 | DHCP客户端的MAC地址长度：此处为6，则DHCP客户端的MAC地址长度为 |
| 1 | 00 | DHCP报文经过的DHCP中继的数目：此处为0,表示没有经过中继 |
| 4 | 27 f0 00 8d | 请求标识：标识一次P地址请求过程。一次请求中所 |
| 2 | 00 00 | DHCP客户端从获取到IP地址或续约过程开始到现在的耗时，秒为单位。此处为0，表示未获得IP地址。DHCP客户端发出请求报文后经过的第一个DHCP中继的IP地址，若没经过DHCP中继，则显示为0。（网关IP地址) |
| 2 | 00 00 | 标志位：广播应答标识位，标识DHCP服务器应答报文是单播还是广播，只使用第1位。=0，单播，=1广播。此处首位为0，表示单播 |
| 4 | C0 a8 01 03 | DHCP客户端的IP地址。 |
| 4 | 00 00 00 00 | DHCP服务器分配给客户端的IP地址，仅在DHCP服务器发送的Offer和ACK报文中显示，其他报文中显示为0。此处全0，因为这不是分配地址的报文。 |
| 4 | 00 00 00 00 | 下个为DHCP客户端分配IP地址等信息的DHCP服务器IP地址，仅在DHCP Offer、DHCP ACK报文中显示，其他报文中显示为0。（用于bootstrap过程中的IP地址)此处为全0，因为DHCP服务器地址暂时未知。 |
| 4 | 00 00 00 00 | DHCP客户端发出请求报文后经过的第一个DHCP中  继的P地址，若没经过DHCP中继，则显示为0。（网  关P地址)。此处为全0，因为没有中继 |
| 16 | A4 b1 c1 03  c4 7e 00 00  00 00 00 00  00 00 00 00 | DHCP客户端的MAC地址，在每个报文中都会显示  对应DHCP客户端的MAC地址。后跟10字节的填充  位，全部为0。此处为0x08d23e524b |
| 64 | 全 00 | 为DHCP客户端分配IP地址的DHCP服务器名称  (DNS域名格式)，在Offer和ACK报文中显示发送  报文的DHCP服务器名称，其他报文显示为0 |
| 128 | 全 00 | DHCP服务器为DHCP客户端指定的启动配置文件名  称及路径信息，仅在DHCP Offer报文中显示，其他  报文中显示为空 |
| 4 | 63 82 53 63 | 后续可选字段的格式。此处为固定值0x63825363,表  示DHCP格式。 |
| … | … | 可选字段 |

## DHCP建立过程

命令行敲下**ipconfig/renew**（命令行会卡在其他interface上）：产生四个包



由此可知，这是四次握手信号。



这四次握手过程，就对应着这四个包。接下来详解这四个包。

### 第一个包：DHCP Discover包



**目的是到达DHCP服务器使得动态获得IP地址（LAN内）**。

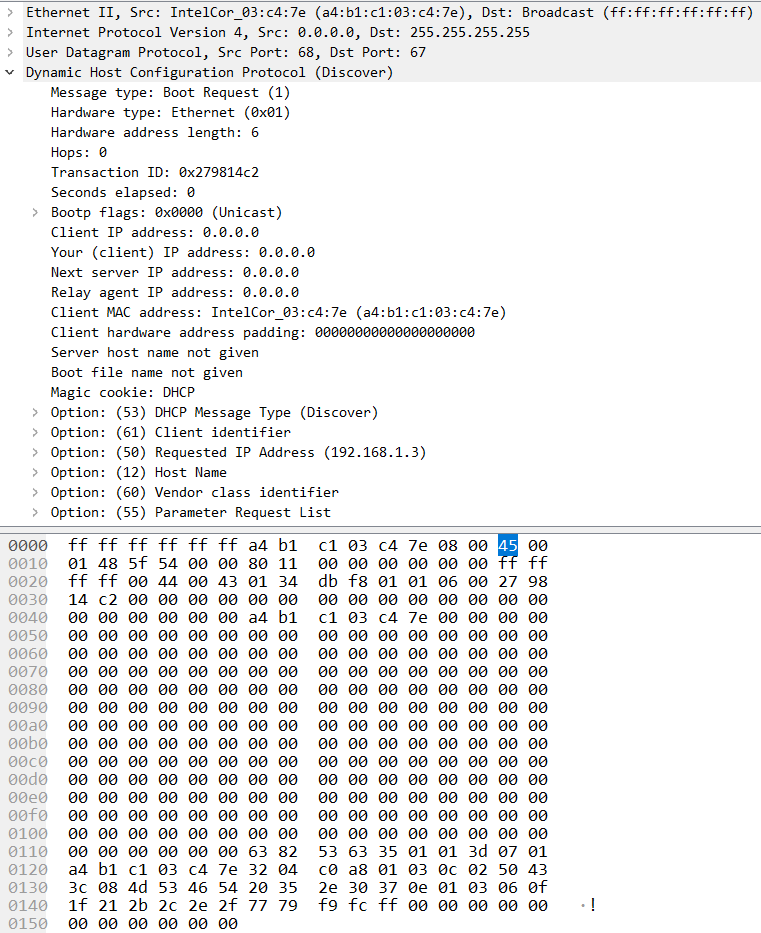
#### 包解析

首先猜测：这是一个类似广播信号的玩意儿，255.255.255.255做网关计算一定是原来的IP地址。那么去查询资料：

[DHCP](https://so.csdn.net/so/search?q=DHCP&spm=1001.2101.3001.7020)客户端通过广播方式发送DHCP DISCOVER请求报文来寻找网络中的DHCP服务器，其中源IP地址为0.0.0.0，目的IP地址为255.255.255.255，因为此时没有IP地址，所以源IP全为0，但是该报文中有用户的MAC地址（就是物理地址）。

可知，这个包将本地的以太网地址发送给DHCP服务器（家里的路由器），**进行动态（内网）IP租赁，使得本地获得一个IP地址**（本机显示首选项是192.168.1.3）。

#### 包内容



打开包内容，确实如猜测，是一个组播地址。**并且source不是IP地址而是MAC地址**（6字节，应该比较智能，因为识别了DHCP Discover包，所以将源显示成MAC地址）。其遵循的协议也是IPv4，可以猜测动态获得的IP地址是4字节。



这里显然也能知道，目的地址是255.255.255（广播）

### 第二个包：DHCP Offer包



**目的是显然从DHCP服务器发过来，给了192.168.1.3一个包，告诉他申请到了。**

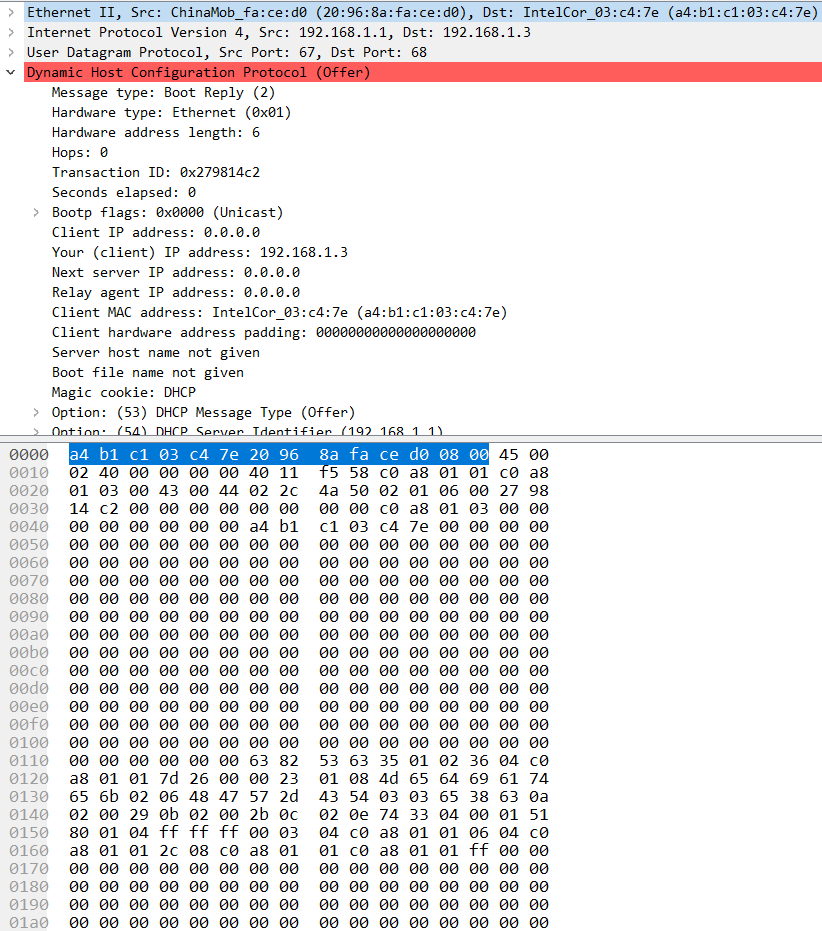
#### 包解析

查阅资料：

DHCP服务器收到客户端发出的**DHCP discover**广播后，通过解析报文，查询dhcpd.conf配置文件。它会从那些还没有租出去的地址中，选择最前面的空置IP，连同其它TCP/IP设定，通过UDP 68端口响应给客户端一个**DHCP offer**数据包（包中包含IP地址、子网掩码、地址租期等信息）。告诉DHCP客户端，该DHCP服务器拥有资源，可以提供DHCP服务。

* 此时还是使用广播进行通讯，源IP地址为DHCP服务器的IP地址，目标地址为255.255.255.255。同时，DHCP服务器为此客户端保留它提供的IP地址，从而不会为其他DHCP客户分配此IP地址。（**这个分析机器比较智能，直接把目标地址换成配给的IP地址了。这个时候显然是不知道有没有被分配到的。我猜测是以太网地址发送，没想到还是广播发送**）
* 由于客户端在开始的时候还没有IP地址，所以在其**DHCP discover**封包内会带有其**MAC地址信息**，并且有一个**XID编号**来辨别该封包，DHCP服务器响应的**DHCP offer**封包则会根据这些资料传递给要求租约的客户。

#### 包内容



这里第一个开头（以太网包头内）就是我的以太网地址（6字节）



在上面的source也可以看出，那个就是DHCP服务器的以太网地址了。

### 第三个包：DHCP REQUEST包



这个包的作用是应该是租赁回应，至于啥作用，来查资料。

#### 包分析

查询资料得知：

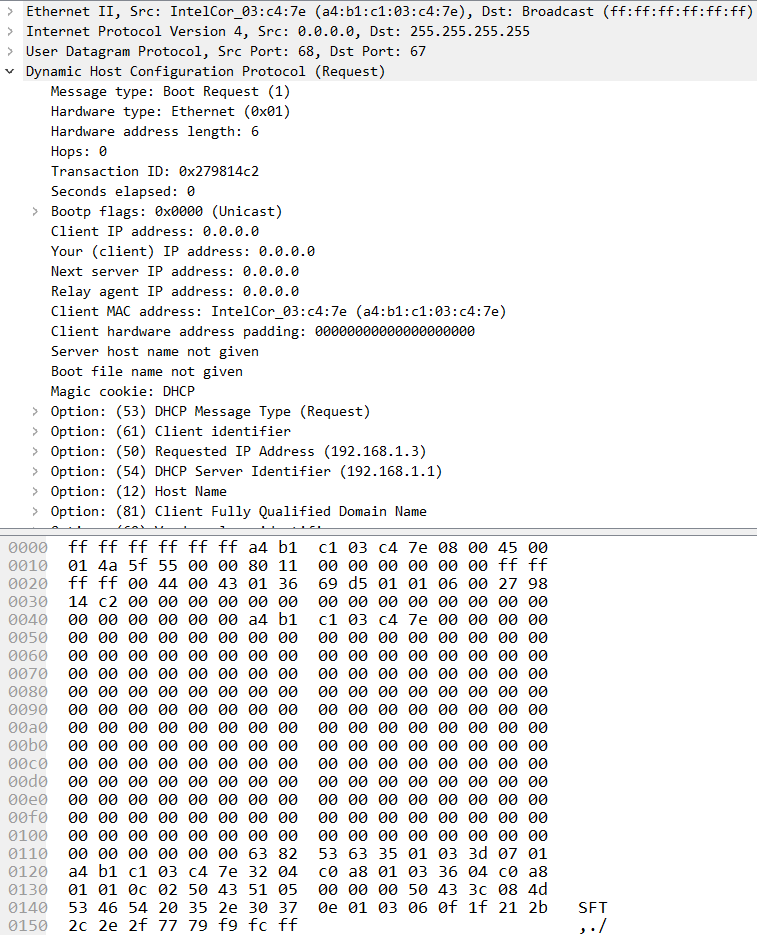
DHCP客户端接受到DHCP offer提供信息之后，如果客户机收到网络上多台DHCP服务器的响应，一般是最先到达的那个，然后以**广播**的方式回答一个**DHCP request**数据包（**包中包含客户端的MAC地址、接受的租约中的IP地址、提供此租约的DHCP服务器地址等**）。告诉所有DHCP服务器它将接受哪一台服务器提供的IP地址，所有其他的DHCP服务器撤销它们的提供以便将IP地址提供给下一次IP租用请求。

* + 此时，由于还没有得到DHCP服务器的最后确认，客户端仍然使用0.0.0.0为源IP地址，255.255.255.255为目标地址进行广播。
  + 事实上，并不是所有DHCP客户端都会无条件接受DHCP服务器的offer，特别是如果这些主机上安装有其它TCP/IP相关的客户机软件。客户端也可以用**DHCP request**向服务器提出DHCP选择，这些选择会以不同的号码填写在DHCP Option Field里面。客户机可以保留自己的一些TCP/IP设定。

显然，这个包还有个冲突：**局域网内多个DHCP服务器。所以在第一个包发出之后，主机可能会获得多个DHCP OFFER。**需要主机为了不让其他DHCP服务器有服务冲突，所以只回复最早到达的，即回复第二个包，发出相应的request第三个包。

**有一说一，以前还真没考虑过局域网的多路由问题。以为真的是一个简单的局域网。**

#### 包内容



这里可以看出这里依旧是广播地址。用的源地址依旧是我的以太网地址。



### 第四个包：DHCP ACK包

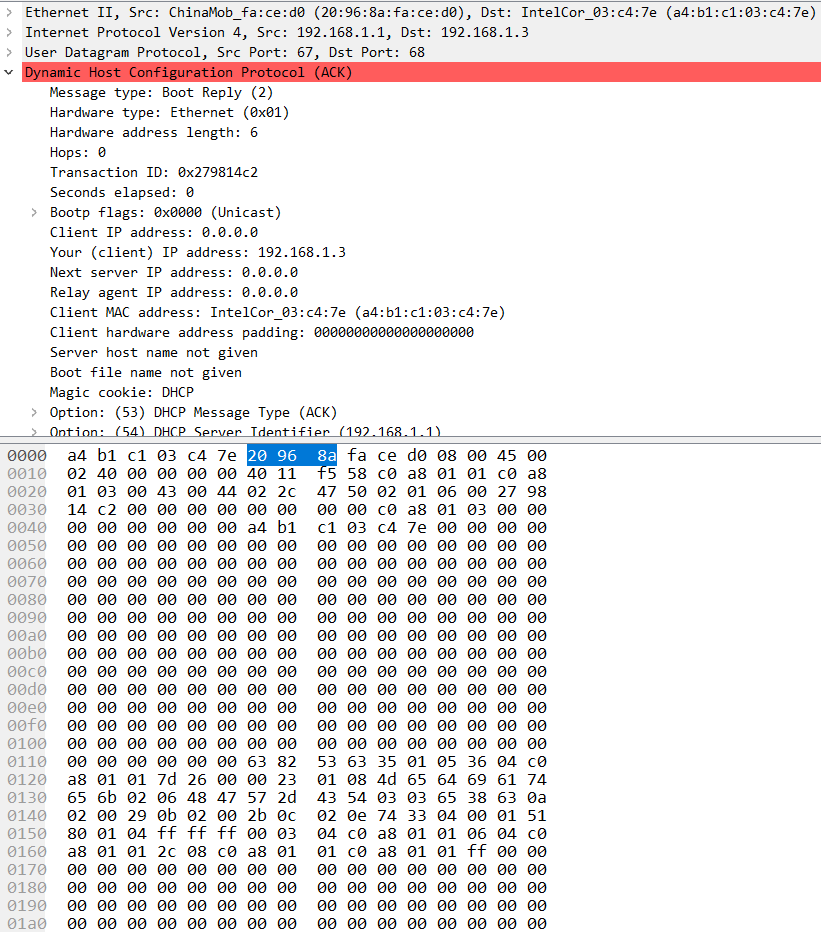


**这个包显然就是回应的ack包了，服务器来说明你已经租赁好相应的IP啦！**

#### 包分析

**IP地址分配确认 & 租约确认（DHCP Ack）**：当DHCP服务器接收到客户机的**DHCP request**之后，会广播返回给客户机一个**DHCP ack**消息包，表明已经接受客户机的选择，告诉DHCP客户端可以使用它提供的IP地址。并将这一IP地址的合法租用以及其他的配置信息都放入该广播包发给客户机。

#### 包内容



这个源地址和目的地址依旧用的是以太网地址。



## 四次握手结果

命令行按下ipconfig/all：



可以看出，这个DHCP控制协议运行成功，**PC成功在DHCP服务器上获得了一个内网（LAN）IP地址，且租赁时间为一天。**我的注册表项将会在我家的路由器上呆一天。如果没有续约，那么他会消失。

## DHCP包理解

地位：在局域网内动态分配IP地址建立连接。

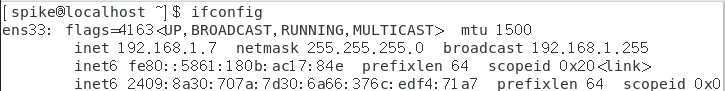
工具：以太网地址。

方式：用的广播地址，就是以太网中讲过的隐藏终端和暴露终端有些关联。

## 思考：虚拟机IP地址的配置过程

这个过程让我想起了在**CentOS上配置网卡的过程**。

当时配置是手动配置网卡（虚拟网卡），当时不懂，就选了个192.168.1.6（往后排），然后默认的GATEWAY就是255.255.255.0。



（这是我其中一个虚拟机网卡，是后端一号机）。

现在看来就是我**在虚拟机的硬件网卡上把这些信息写死了**，所谓的IPv4地址也不是租赁而是我写的192.168.1.6。当时连接也用的桥接模式，而不是共享啥的，现在也算稍微理解一些这个东西了。

# ARP–地址解析协议

地址解析协议，即ARP（Address Resolution Protocol），是根据[IP地址](https://baike.baidu.com/item/IP%E5%9C%B0%E5%9D%80" \t "_blank)获取[物理地址](https://baike.baidu.com/item/%E7%89%A9%E7%90%86%E5%9C%B0%E5%9D%80/2129)的一个[TCP/IP协议](https://baike.baidu.com/item/TCP%2FIP%E5%8D%8F%E8%AE%AE)。[主机](https://baike.baidu.com/item/%E4%B8%BB%E6%9C%BA/455151)发送信息时将包含目标IP地址的ARP请求广播到局域网络上的所有主机，并接收返回消息，以此确定目标的物理地址；收到返回消息后将该IP地址和物理地址存入本机ARP缓存中并保留一定时间，下次请求时直接查询ARP缓存以节约资源。

在TCP/IP网络环境下，每个主机都各自拥有一个32位的IP地址，这种互联网地址是在网际范围标识主机的一种逻辑地址。为了让报文在物理网路上传送，必须知道对方目的主机的物理地址。这样就存在着把IP地址变换成物理地址的地址转换问题。

以WLAN环境为例，为了正确地向目的主机传送报文，必须把目的主机的32位IP地址转换成为48位以太网的地址。这就需要在互连层有一组服务将IP地址转换为相应物理地址，这组协议就是ARP协议。

## ARP报文格式详解



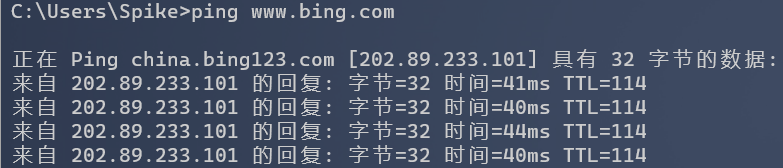
* 硬件类型：指明了发送方想知道的硬件[接口](https://baike.baidu.com/item/%E6%8E%A5%E5%8F%A3)类型，以太网的值为1；
* 协议类型：指明了发送方提供的高层[协议](https://baike.baidu.com/item/%E5%8D%8F%E8%AE%AE/13020269)类型，IP为0800（16进制）；
* 硬件地址长度和协议长度：指明了硬件地址和高层协议地址的长度，这样ARP报文就可以在任意硬件和任意协议的网络中使用；
* 操作类型：用来表示这个报文的类型，ARP请求为1，ARP响应为2，RARP请求为3，RARP响应为4；
* 发送方硬件地址（0-3字节）：源主机硬件地址的前3个字节；
* 发送方硬件地址（4-5字节）：源主机硬件地址的后3个字节；
* 发送方IP地址（0-1字节）：源主机硬件地址的前2个字节；
* 发送方IP地址（2-3字节）：源主机硬件地址的后2个字节；
* 目标硬件地址（0-1字节）：目的主机硬件地址的前2个字节；
* 目标硬件地址（2-5字节）：目的主机硬件地址的后4个字节；
* 目标IP地址（0-3字节）：目的主机的IP地址。

## 捕获ARP包

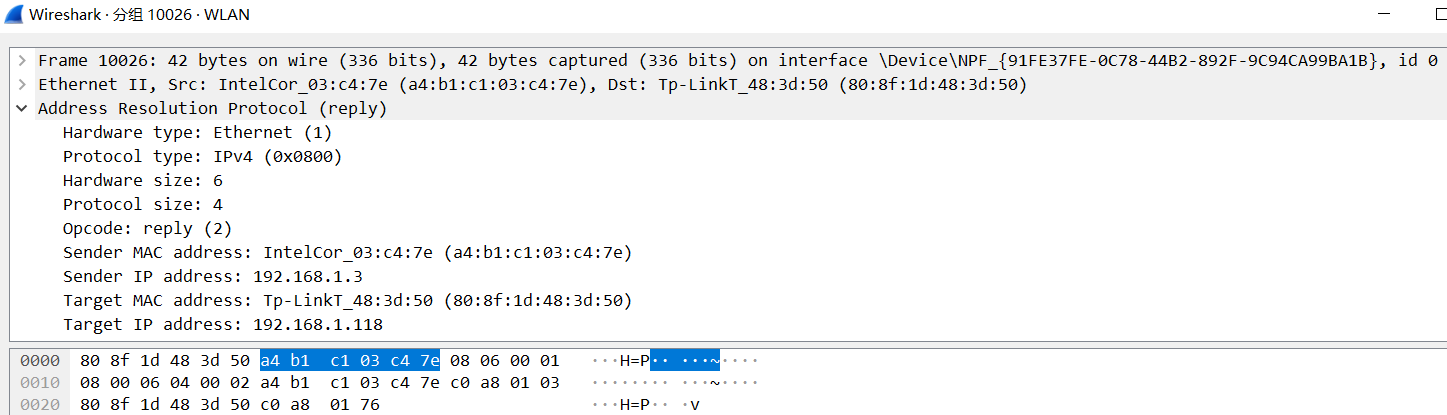
1. 运行Wireshark软件，在首页选择进行连接的网络，这里选择WLAN:

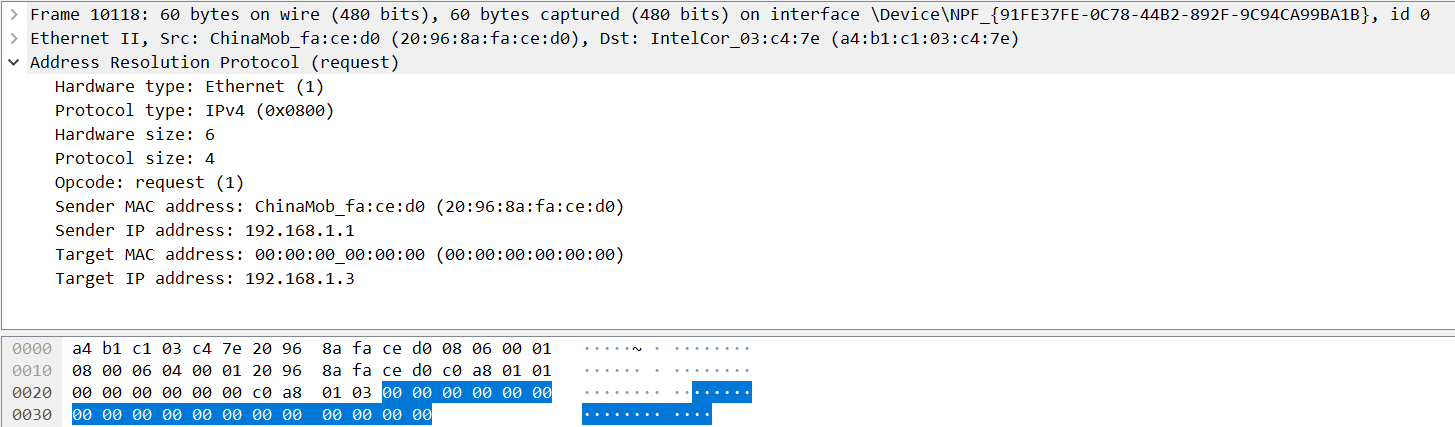
2. 在应用显示过滤器中输入arp,过滤出ARP报文：

3. 打开命令提示符窗口，输入ping **www.bing.com**，回车执行：

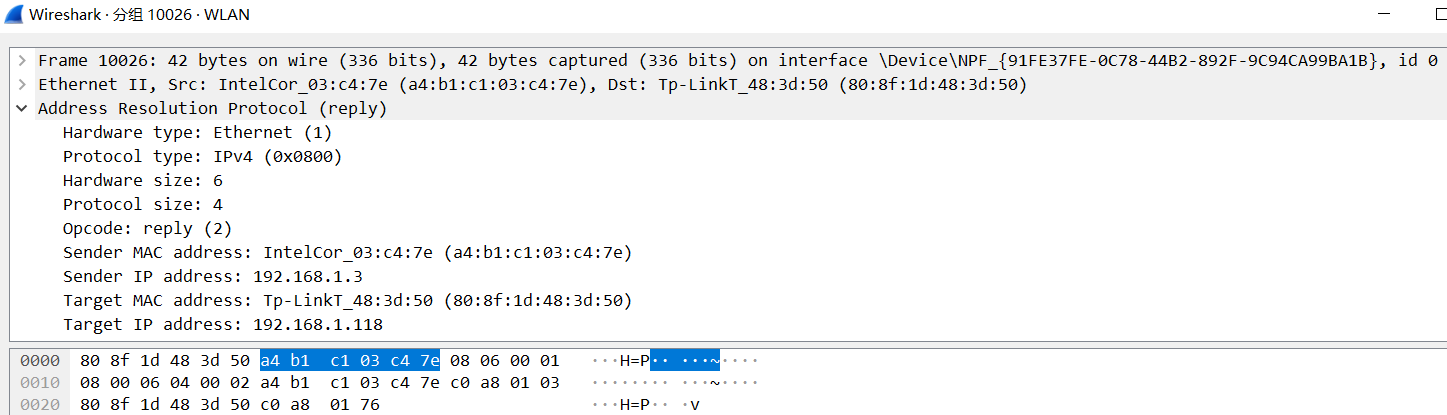


4. 观察Wireshark捕获的数据报：



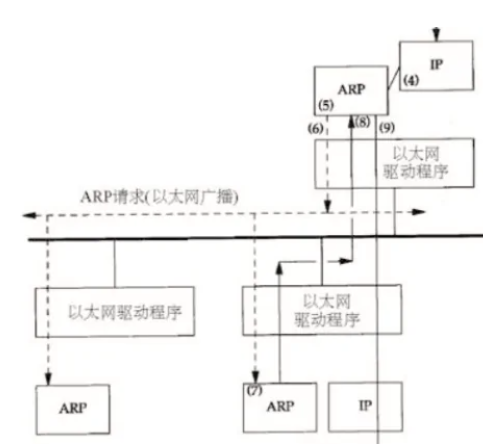


## 分析ARP包（以本机发送为例）



|  |  |  |
| --- | --- | --- |
| 单元的字节数 | 值（十六进制） | 解释 |
| 14 | 略 | 以太网报头 |
| 2 | 00 01 | ARP报头的开始。硬件类型：表示ARP报文可在哪  种类型的网络上传输。  此处为1，表示为以太网地址。 |
| 2 | 08 00 | 上层协议类型：表示硬件地址要映射的协议地址类型。  此处为0x0800,表示Pv4协议。 |
| 1 | 06 | MAC地址长度：标识MAC地址长度，字节为单位。  此处为6，表示MAC地址长度为6字节。 |
| 1 | 04 | IP协议地址长度：标识IP地址长度，字节为单位。  此处为4，表示IP地址长度为4字节。 |
| 2 | 00 02 | 操作类型：本次ARP报文类型；=1，ARP请求报文；  =2,ARP应答报文。这里是应答报文。 |
| 6 | A4 b1 c1 03 c4 7e | 源MAC地址，表示发送方设备的硬件地址。  此处为A4 b1 c1 03 c4 7e |
| 4 | C0 a8 01 03 | 源IP地址，表示发送方的IP地址。  此处为192.168.1.3 |
| 6 | 80 8f 1d 48 3d 50 | 目的MAC地址，表示接收方设备的硬件地址。如果  为全0，则表示任意地址。 |
| 4 | C0 a8 01 76 | 目的P地址，表示接受方的IP地址。  此处为192.168.1.118 |

## ARP工作原理和作用

****

* 其目的是为了使**本地保存相应的MAC地址**，在发送包的时候假如缓冲区有对应的MAC地址，那么直接使用，封装IP包，不用再广播找。
* ARP分组发送在DHCP Request和DHCP ACK之间，此时路由器请求目的地址的MAC地址，但是此时IP地址还没有分配，所以本机没有回复。ARP分组发送在IP地址分配完成之后，本机向局域网发送广播，请求路由器的MAC地址.
* 对第二个ARP的回复，其中OPER字段设为2，表示一个回复分组，分组内容中发送方的部分为路由器的P和MAC地址。ARP探针(ARP Probe)分组，意图是探测P地址在局域网中是否己经被使用，其发送方MAC地址被设为本机，发送方IP地址和接收方MAC地
* 址被设为空，接收方P地址设为想要探测的P地址，即本机的P地址。
* ARP声明(ARP Announcement)分组，意图是在局域网中“声明”这个P地址，除了发送方P地址被设为本机P之外，其他内容和ARP Probe分组相同，其他主机可以用发送方P和MAC地址在ARP缓存中建立映射。

# TCP连接过程

## TCP的目的

传输控制协议（TCP，Transmission Control Protocol）是为了在不可靠的互联网络上提供可靠的端到端字节流而专门设计的一个传输协议。

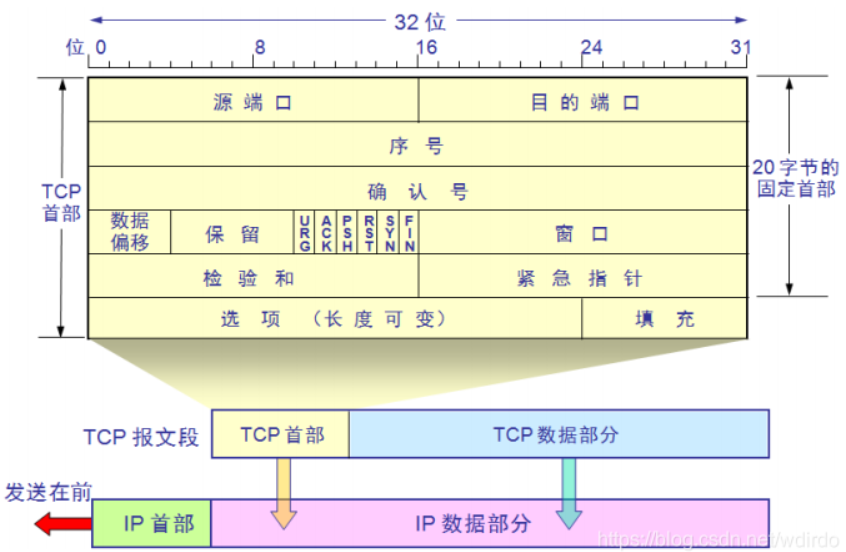
互联网络与单个网络有很大的不同，因为互联网络的不同部分可能有截然不同的拓扑结构、带宽、延迟、数据包大小和其他参数。TCP的设计目标是能够动态地适应互联网络的这些特性，而且具备面对各种故障时的健壮性。

不同主机的应用层之间经常需要可靠的、像管道一样的连接，但是IP层不提供这样的流机制，而是提供不可靠的包交换。

## TCP的做法

应用层向TCP层发送用于网间传输的、用8位字节表示的数据流，然后TCP把数据流分区成适当长度的报文段（通常受该计算机连接的网络的数据链路层的最大传输单元（MTU）的限制）。之后TCP把结果包传给IP层，由它来通过网络将包传送给接收端实体的TCP层。TCP为了保证不发生丢包，就给每个包一个序号，同时序号也保证了传送到接收端实体的包的按序接收。然后接收端实体对已成功收到的包发回一个相应的确认（ACK）；如果发送端实体在合理的往返时延（[RTT](https://baike.baidu.com/item/RTT" \t "_blank)）内未收到确认，那么对应的数据包就被假设为已丢失将会被进行重传。TCP用一个校验和函数来检验数据是否有错误；在发送和接收时都要计算校验和。

## TCP报文格式



## 追踪TCP包

1. 打开WLAN，输入**ip.addr eq 124.220.8.25 && tcp.port eq 80**，追踪访问服务器的包。

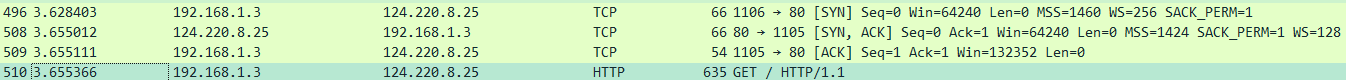
2. 打开浏览器，访问我的服务器主页



（文本中这是一个NAT穿透技术的使用。

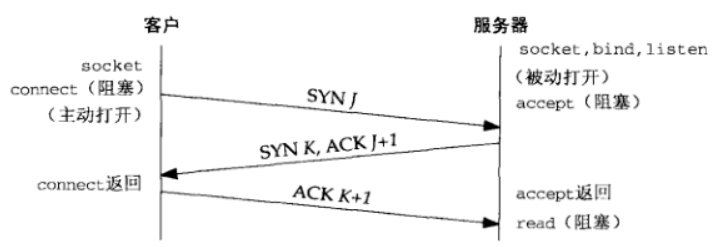
因为项目要外部访问数据库，且服务器没配好就先用这个顶着了）

3. 查看三次握手包：

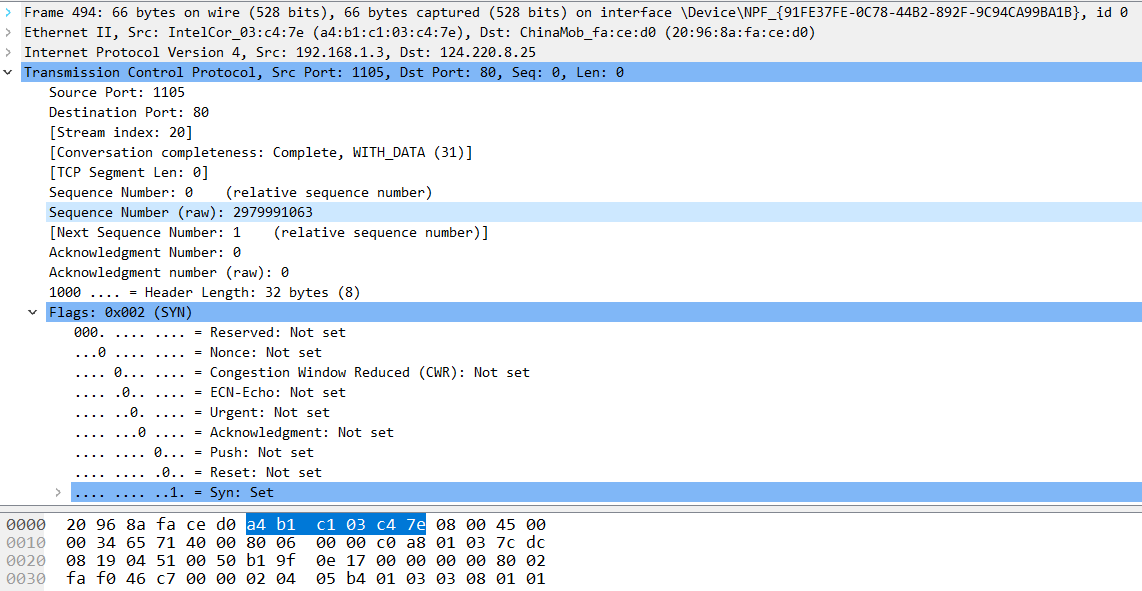


可以看到，确实是**三个包建立了TCP连接**。

原理图如下：（以传输层socket的TCP通信协议为例进行讲解，会更直观）

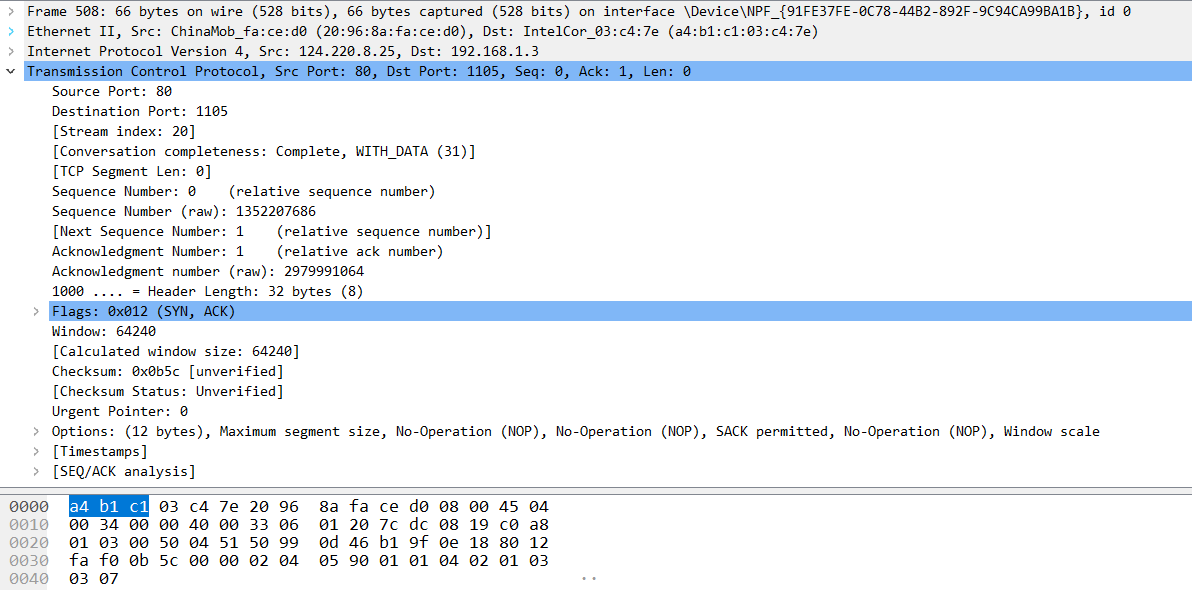


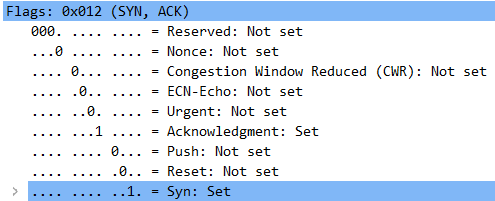
### 第一次握手



|  |  |  |
| --- | --- | --- |
| 字段 | 内容（0x） | 含义 |
| Source Port | 04 51 | 源端口，此时为 1105 |
| Destination Port | 00 50 | 源端口，此时为 80 |
| Sequence Number(raw) | B1 9f 0e 17 | 随机序列号**2939991063，有大用** |
| Ack Number | 00 00 00 00 | ACK序列号0，**也有大用** |
| Header Length | 8 | 当前偏移量，代表包长度 |
| Resv | 0 | 保留项，无用 |
| CWR（Congestion Window Reduce） | 0 | 拥塞窗口减少标志，用来表明它接收到了设置 ECE 标志的 TCP 包。并且，发送方收到消息之后，通过减小发送窗口的大小来降低发送速率。 |
| ECE（ECN Echo） | 0 | 用来在 TCP 三次握手时表明一个 TCP 端是具备 ECN 功能的。在数据传输过程中，它也用来表明接收到的 TCP 包的 IP 头部的 ECN 被设置为 11，即网络线路拥堵。 |
| URG（Urgent） | 0 | 表示本报文段中发送的数据是否包含紧急数据。URG=1 时表示有紧急数据。当 URG=1 时，后面的紧急指针字段才有效。 |
| ACK | 0 | 表示前面的确认号字段是否有效。ACK=1 时表示有效。只有当 ACK=1 时，前面的确认号字段才有效。TCP 规定，连接建立后，ACK 必须为 1。 |
| PSH（Push） | 0 | 告诉对方收到该报文段后是否立即把数据推送给上层。如果值为 1，表示应当立即把数据提交给上层，而不是缓存起来。 |
| RST | 0 | 表示是否重置连接。如果 RST=1，说明 TCP 连接出现了严重错误（如主机崩溃），必须释放连接，然后再重新建立连接。 |
| SYN | 1 | 在建立连接时使用，用来同步序号。当 SYN=1，ACK=0 时，表示这是一个请求建立连接的报文段；当 SYN=1，ACK=1 时，表示对方同意建立连接。SYN=1 时，说明这是一个请求建立连接或同意建立连接的报文。只有在前两次握手中 SYN 才为 1。 |
| FIN | 0 | 标记数据是否发送完毕。如果 FIN=1，表示数据已经发送完成，可以释放连接。 |
| Window Size | fa f0 | 从 Ack Number 开始还可以接收多少字节的数据量，也表示当前接收端的接收窗口还有多少剩余空间。该字段可以用于 TCP 的流量控制。 |
| TCP Checksum | 46 c7 | 确认传输的数据是否有损坏 |
| Urgent Pointer | 00 00 | 仅当前面的 URG 控制位为 1 时才有意义。它指出本数据段中为紧急数据的字节数，占 16 位。当所有紧急数据处理完后，TCP 就会告诉应用程序恢复到正常操作。即使当前窗口大小为 0，也是可以发送紧急数据的，因为紧急数据无须缓存。 |
| Option | 02 04 05 b4 01 03 03 08 01 01 04 02 | 长度必须是 32bits 的整数倍(偏移字节数) |

### 第二次握手





这里就分析重要的地方了。

**SYN = 1**

服务器向用户端发送的为**同步报文段，代表前两次握手的第二次握手。**

这是第二次握手，为了表示是返回的第一次握手的回应信号，**Acknowledge Number**所以返回的时候是要返回**第一次握手的帧Sequence+1**。

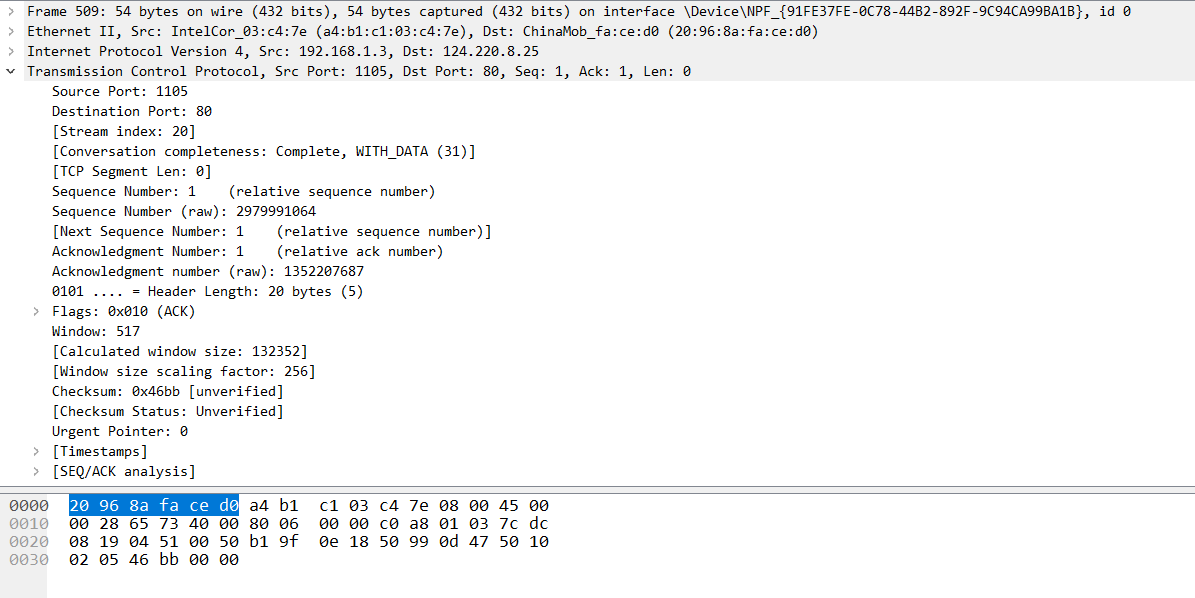
**第一次**：（随机序列）

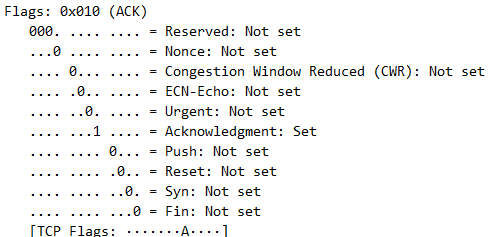
**第二次**：

**ACK = 1**

这里代表是回应功能的包。意味着这是需要回应第一个包。

### 第三次握手





这个包和上两个包也有密切的关系

**SYN = 0**

代表这是握手的第三次，所以此值为0

**第一次**：（随机序列）

**第二次**：（+1）

**第三次**：**（随机序列不随机，和前两个包联系起来）**

**第二次**：（随机序列）

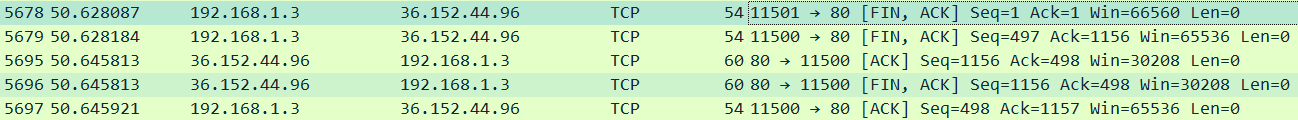
**第三次**：（+1）

**ACK = 1**

这里代表是回应信号。在这个包完成任务后，整个三次握手的过程就结束了。

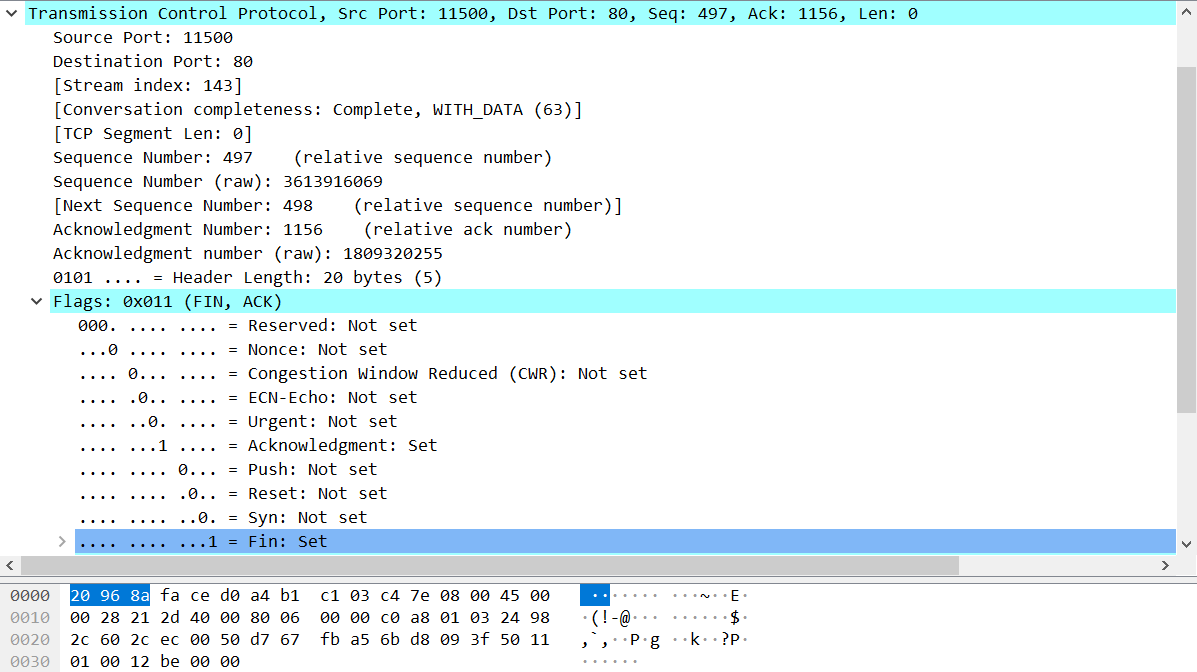
## TCP释放连接

重新捕获，然后关闭网页，得到四个包：



显然这是TCP释放连接的四次挥手。（这里每个包就不详细分析了）

### 第一次挥手

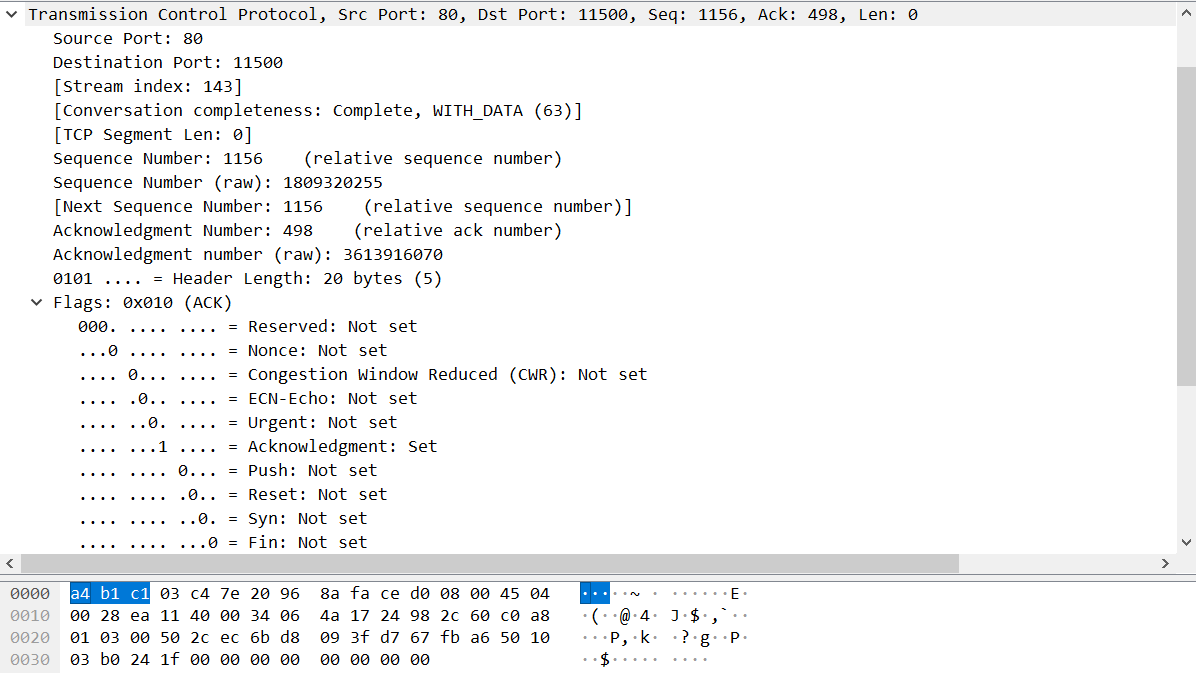


客户端给服务器发送TCP包，用来关闭客户端到服务器的数据

传送。将标志位**FIN和ACK置为1**，序号为X=1,确认序号为Z=1。

Seq = 497，Ack = 1156

### 第二次挥手

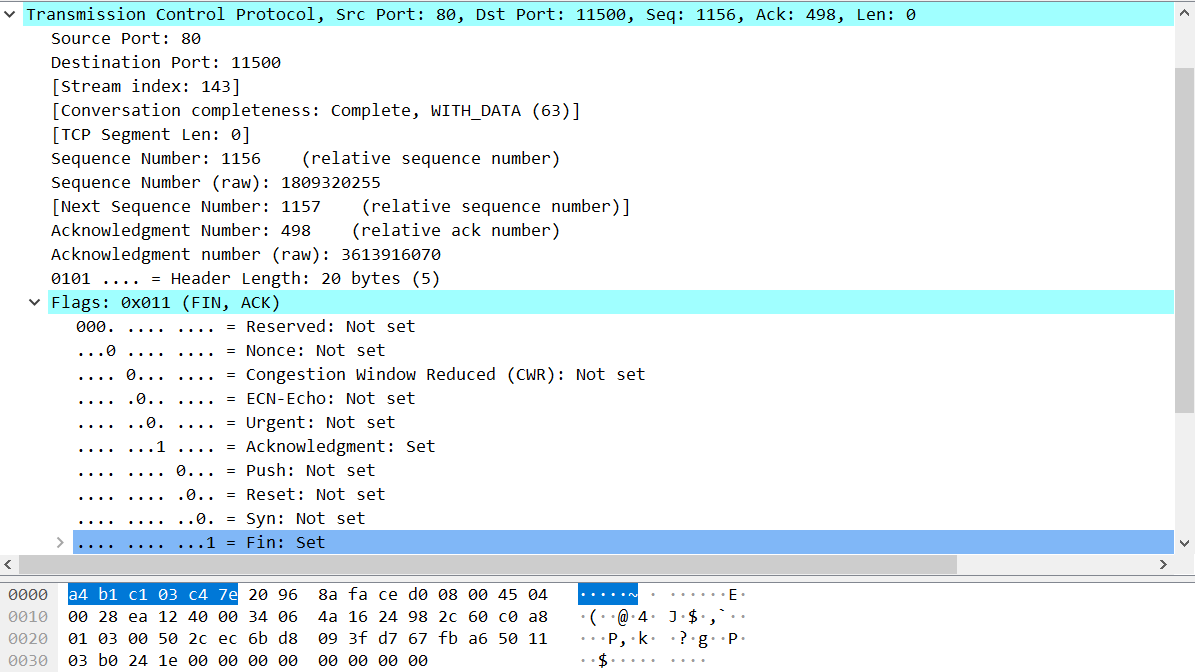


服务器收到FIN后，发回一个ACK(标志位ACK=1),确认序号

为收到的序号加1，即X=X+1=2。序号为收到的确认序号=Z。

Seq = 1156 = 第一次挥手Ack，Ack = 498 = 第一次挥手Seq+1

### 第三次挥手

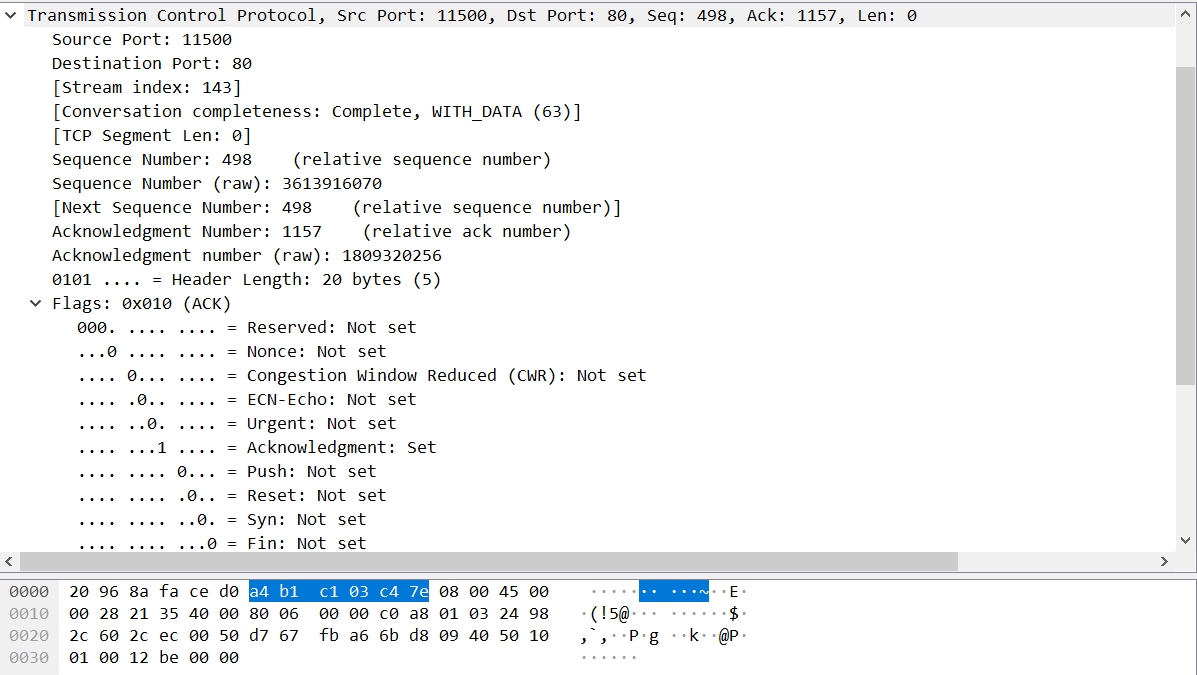


第三次挥手：服务器关闭与客户端的连接，发送一个FIN。标志位FIN和

ACK置为1，序号为Y=1,确认序号为X=2。

Seq = 1157 = 第二次挥手Ack+1，Ack = 498 = 第二次挥手Seq

### 第四次挥手

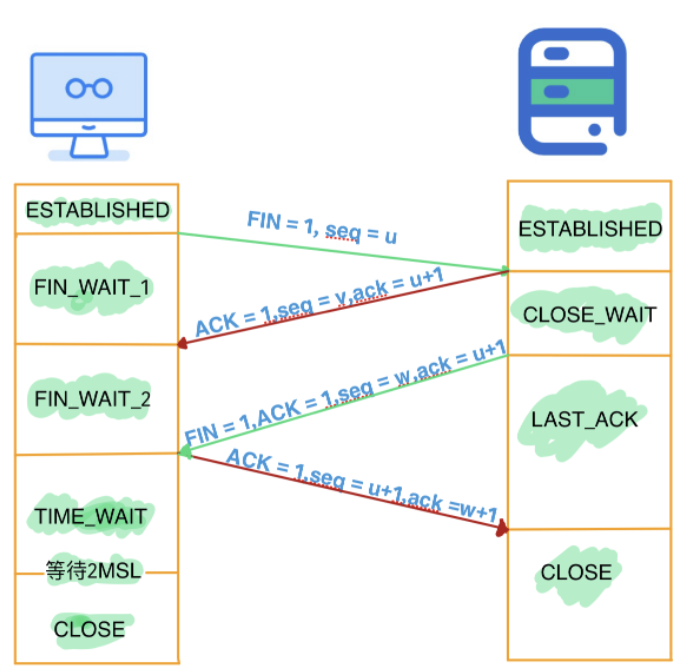


第四次挥手：客户端收到服务器发送的FN之后，发回ACK确认（标志位

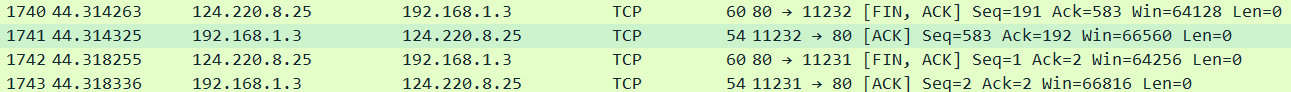
ACK=1),确认序号为收到的序号加1，即Y+1=2。序号为收到的确认序号X=2。

Seq = 498 = 第三次挥手Ack，Ack = 1157 = 第三次挥手Seq

### 四次挥手整体

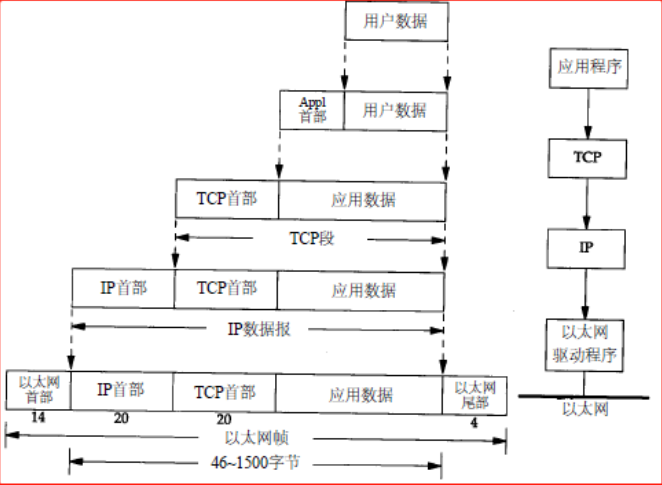


## Nginx反向代理的四次挥手？



非常奇怪，试了无数次，都是服务器发出Fin信号？

## 架构理解



这里是应用程序接到数据链路层协议的过程。

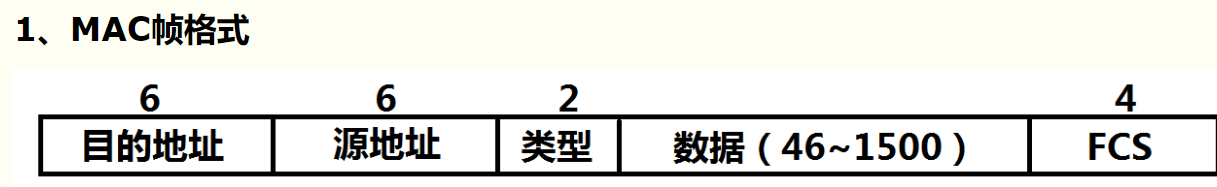
可知，**以太网包头部是14Bytes，IP包头部是20Bytes（IPv4协议）**。

若是**IPv6，那么IP包头部是40Bytes**。

在这个包装下，用到的才是真正的包。

### 以太网帧结构

这是常用的以太网地址帧，其前14Bytes在LAN内都在开头加上，因为要走以太网总线。

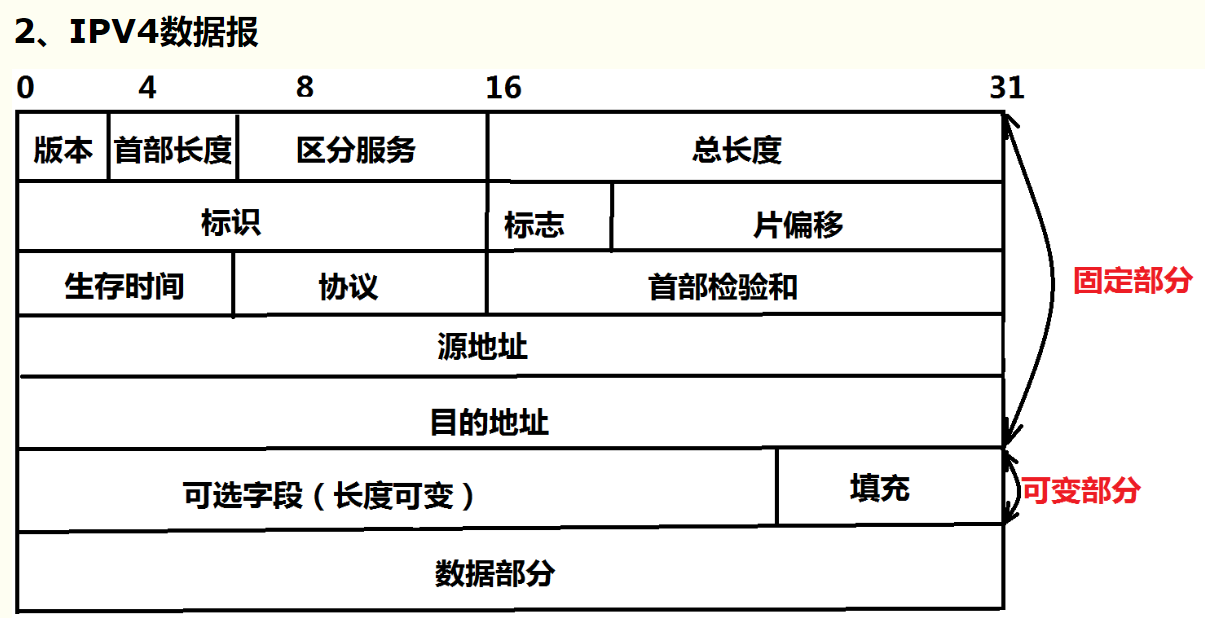


在整个过程中，**只要走以太网物理层，那就一定要加上这个前14Bytes（以太网帧头）**。

而在我们的家用路由器中，他就是一根以太网线连接，所以包头都会有以太网帧头。

所以在我们整个分析包的过程中，开头14字节都是以太网帧头。

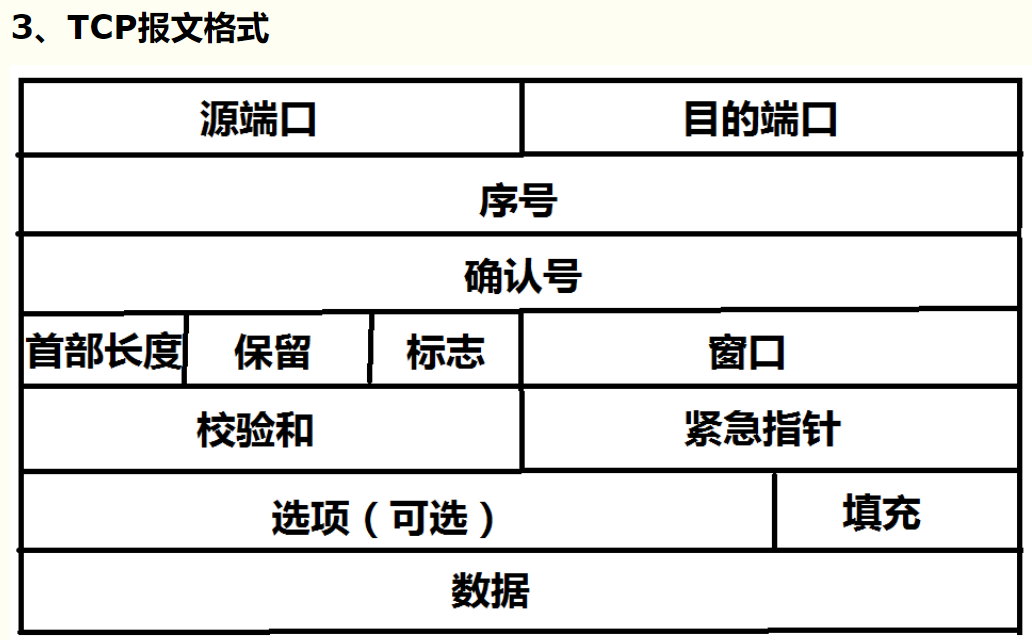
### IP包帧结构



这就是在**DHCP包发挥作用后，包含PC租赁的LAN IP的，以太网帧包裹的，IP帧。**

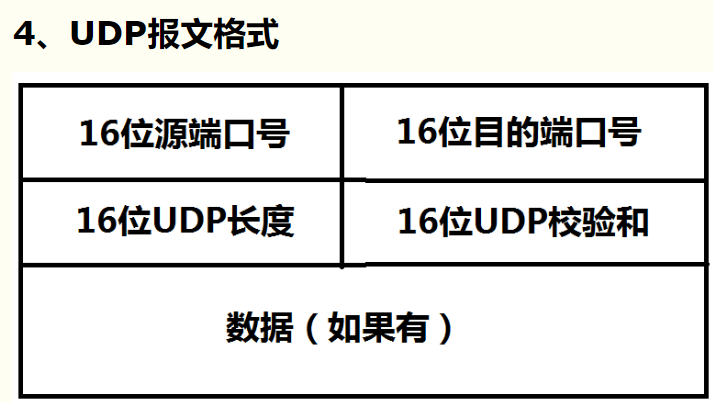
他的头部主要占据20Bytes，所以在一般的IP包中，他也会有**14Bytes（以太网帧头）+20Bytes（IP帧头）**。**这是分析IP数据包的关键。**

### TCP帧结构



在IP包的包装下，才能有这个可靠的数据报服务。

### UDP帧格式



UDP和TCP地位相等，但是不可靠。这种不可靠服务主要用于对精度要求不高但是实时性要求高的场景，如视频通话之类。

# 实验总结

## 实验问题

1. wireshark使用问题：文档描述不全，参考别的班的文档才开始做，

2. DHCP理解问题：广播发送，以太网地址，内网申请动态IP，可能会多个DHCP服务器

3. TCP连接四次挥手时我的Nginx反向代理不遵循基本的四次挥手？

## 实验心得

在计算机网络“网络层数据分组的捕获和解析”实验中，我完成了对网络数据包的抓取与分析。本次实验的进展波折，主要是用自己服务器整活导致一系列问题。

我从实验开始至实验报告大致完成共花费了12个小时，但我从本次实验中得到的收获是巨大的：除了实验中的内容之外，我还借助此次机会，对控制协议数据包进行了捕获与分析，对TCP的三次握手及可靠传输机制，对ICMP中使用到的组播技术有了深刻的理解。

此外，这次实验让我加深了对IP、TCP数据段首部的理解，了解了整体架构，TCP/IP的整体使用方法，以及各个控制协议在整个系统中的用处，收获巨大。