北京郵電大學

Beijing University of Posts and Telecommunications

计算机网络实验报告

[网络层数据报分析]

学院：计算机学院
马天成
2022 年 6 月 13 日

北京邮电大学《计算机网络》课程实验报告

<table>
<tr><td>实验名称</td><td colspan="2">网络层数据报分析</td><td>学院</td><td>计算机</td><td>指导教师</td><td>蒋砚军</td></tr>
<tr><td>班 级</td><td>班内序号</td><td colspan="2">学 号</td><td>学生姓名</td><td colspan="2">成绩</td></tr>
<tr><td>2020211305</td><td></td><td colspan="2">2020211376</td><td>马天成</td><td colspan="2"></td></tr>
<tr><td>实验内容</td><td colspan="6">本次实验主要包含下列内容:
1)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。</td></tr>
<tr><td>学生实验报告</td><td colspan="6">(详见“实验报告和源程序”册)</td></tr>
<tr><td>课程设计成绩评定</td><td colspan="6">评语:

成绩:

指导教师签名:

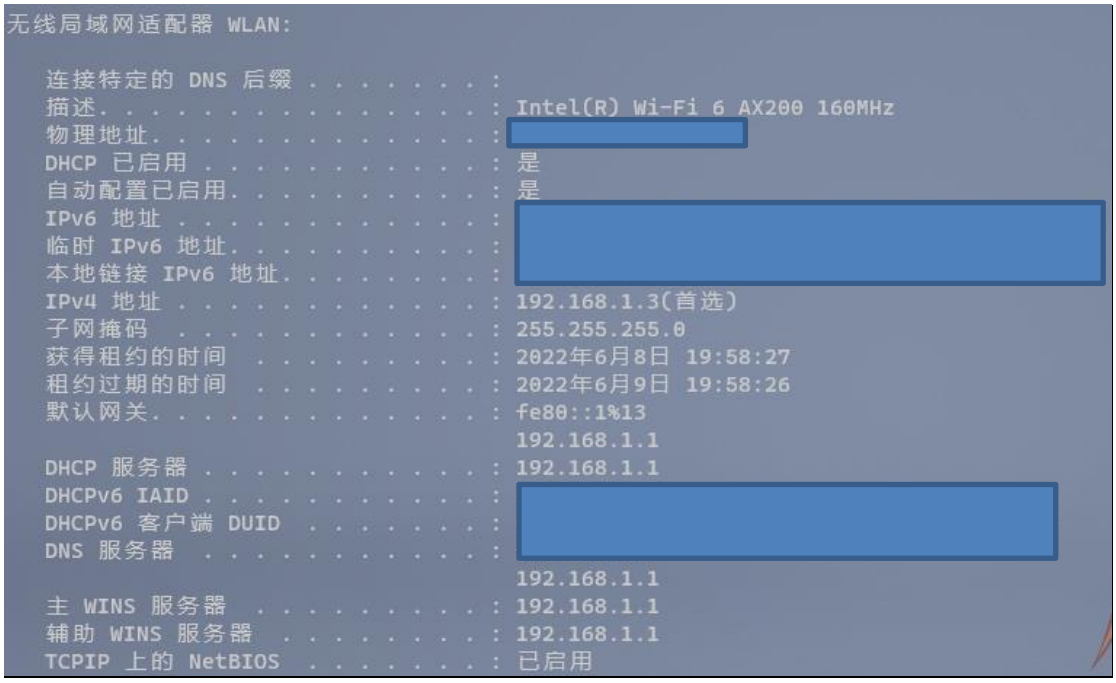
年 月 日</td></tr>
</table>

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

本人 PC 网络适配器参数



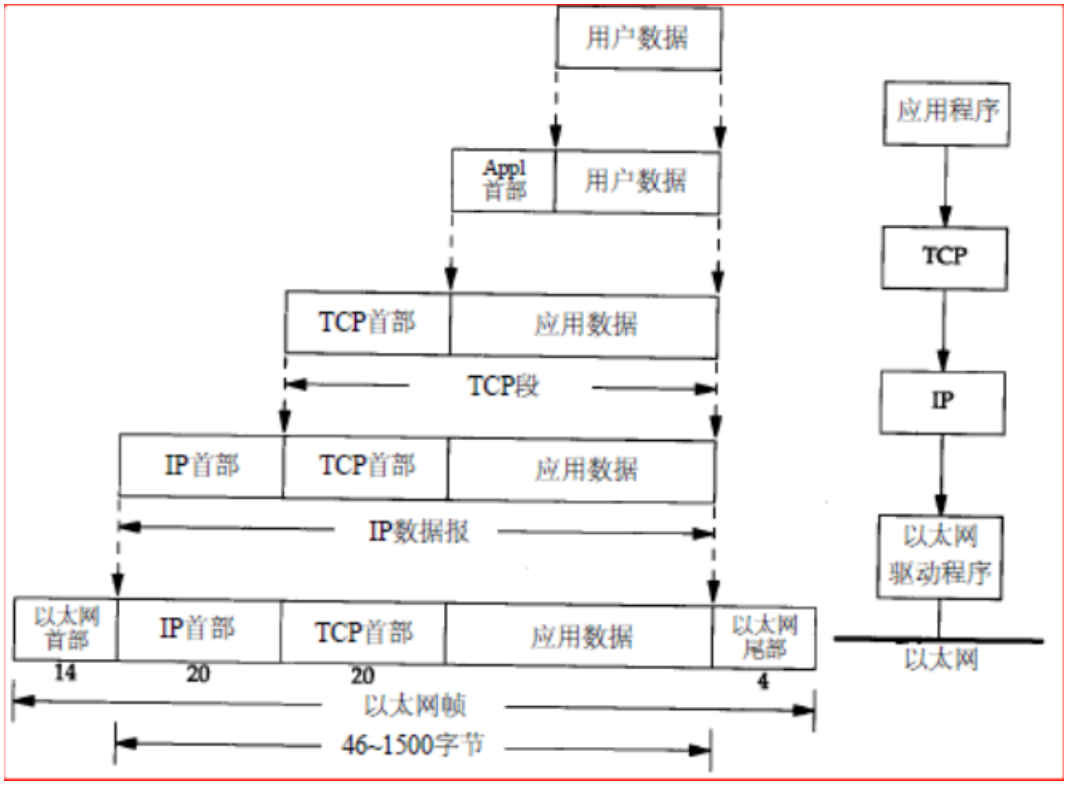
本人服务器公网地址

124.220.8.25（没买域名）

已提前确定网络正常连接。

理论准备

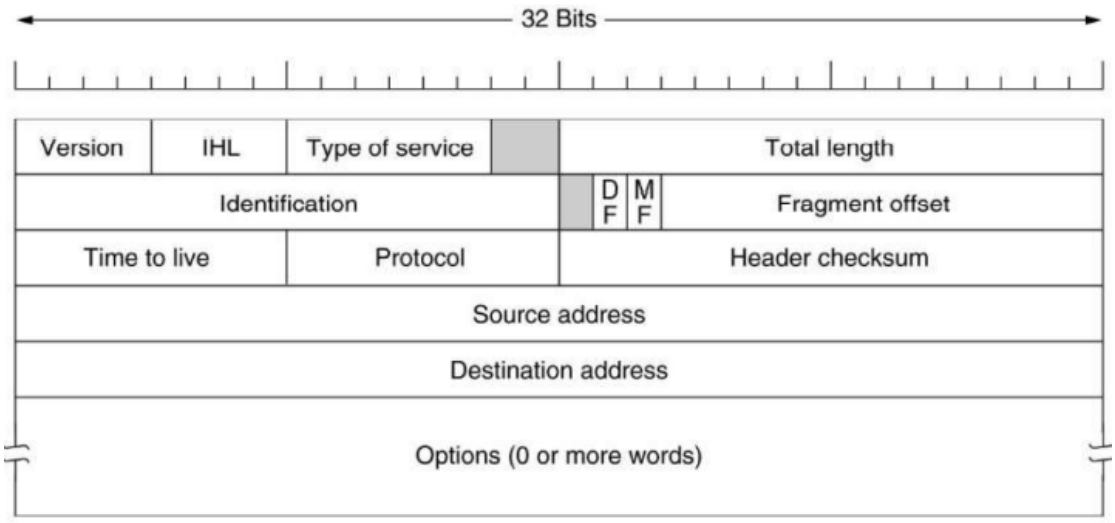
TCP/IP 架构和帧格式



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。

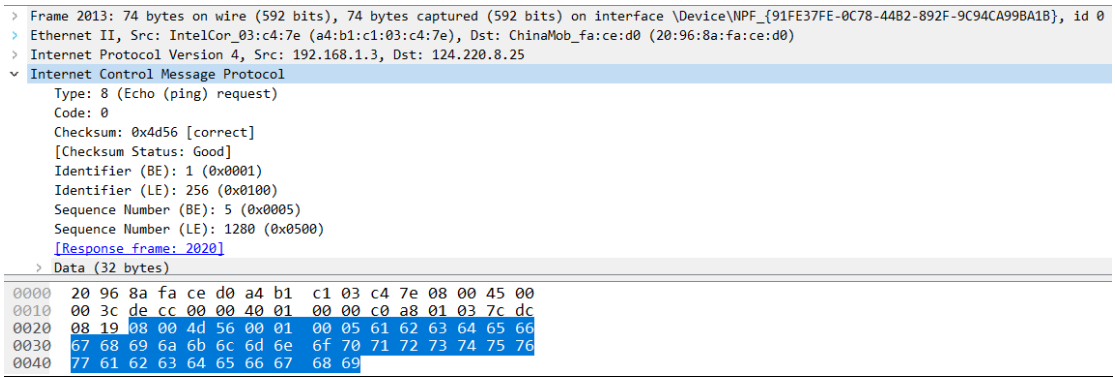
```
C:\Users\Spike>ping 124.220.8.25

正在 Ping 124.220.8.25 具有 32 字节的数据:
来自 124.220.8.25 的回复: 字节=32 时间=26ms TTL=51
来自 124.220.8.25 的回复: 字节=32 时间=25ms TTL=51
来自 124.220.8.25 的回复: 字节=32 时间=26ms TTL=51
来自 124.220.8.25 的回复: 字节=32 时间=26ms TTL=51
```

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：

2013	6.216138	192.168.1.3	124.220.8.25	ICMP	74 Echo (ping) request id=0x0001, seq=5/1280, ttl=64 (reply in 2020)
2020	6.242572	124.220.8.25	192.168.1.3	ICMP	74 Echo (ping) reply id=0x0001, seq=5/1280, ttl=51 (request in 2013)
2260	7.227679	192.168.1.3	124.220.8.25	ICMP	74 Echo (ping) request id=0x0001, seq=6/1536, ttl=64 (reply in 2267)
2267	7.253315	124.220.8.25	192.168.1.3	ICMP	74 Echo (ping) reply id=0x0001, seq=6/1536, ttl=51 (request in 2260)
2450	8.243680	192.168.1.3	124.220.8.25	ICMP	74 Echo (ping) request id=0x0001, seq=7/1792, ttl=64 (reply in 2452)
2452	8.270217	124.220.8.25	192.168.1.3	ICMP	74 Echo (ping) reply id=0x0001, seq=7/1792, ttl=51 (request in 2450)
2626	9.252545	192.168.1.3	124.220.8.25	ICMP	74 Echo (ping) request id=0x0001, seq=8/2048, ttl=64 (reply in 2630)
2630	9.278729	124.220.8.25	192.168.1.3	ICMP	74 Echo (ping) reply id=0x0001, seq=8/2048, ttl=51 (request in 2626)

解析单个 IP 包

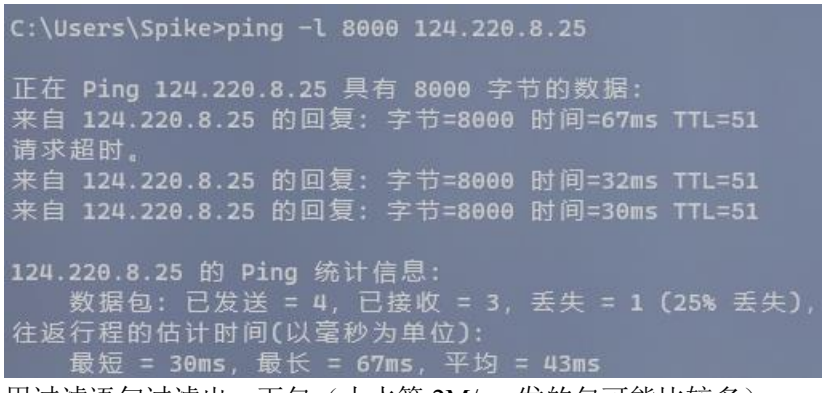
以第一个 IP 包为例：



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）

21	3.948296	192.168.1.3	124.220.8.25	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=ded4) [Reassembled in #26]
22	3.948296	192.168.1.3	124.220.8.25	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=ded4) [Reassembled in #26]
23	3.948296	192.168.1.3	124.220.8.25	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=2960, ID=ded4) [Reassembled in #26]
24	3.948296	192.168.1.3	124.220.8.25	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=4440, ID=ded4) [Reassembled in #26]
25	3.948296	192.168.1.3	124.220.8.25	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=5920, ID=ded4) [Reassembled in #26]
26	3.948296	192.168.1.3	124.220.8.25	ICMP	642 Echo (ping) request id=0x0001, seq=17/4352, ttl=64 (reply in 32)
27	4.013220	124.220.8.25	192.168.1.3	IPv4	1482 Fragmented IP protocol (proto=ICMP 1, off=0, ID=ec5d) [Reassembled in #32]
28	4.013962	124.220.8.25	192.168.1.3	IPv4	1482 Fragmented IP protocol (proto=ICMP 1, off=1448, ID=ec5d) [Reassembled in #32]
29	4.014699	124.220.8.25	192.168.1.3	IPv4	1482 Fragmented IP protocol (proto=ICMP 1, off=2896, ID=ec5d) [Reassembled in #32]
30	4.015386	124.220.8.25	192.168.1.3	IPv4	1482 Fragmented IP protocol (proto=ICMP 1, off=4344, ID=ec5d) [Reassembled in #32]
31	4.015707	124.220.8.25	192.168.1.3	IPv4	1482 Fragmented IP protocol (proto=ICMP 1, off=5792, ID=ec5d) [Reassembled in #32]
32	4.015707	124.220.8.25	192.168.1.3	ICMP	802 Echo (ping) reply id=0x0001, seq=17/4352, ttl=51 (request in 26)
33	4.965609	192.168.1.3	124.220.8.25	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=ded5) [Reassembled in #38]
34	4.965609	192.168.1.3	124.220.8.25	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=ded5) [Reassembled in #38]
35	4.965609	192.168.1.3	124.220.8.25	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=2960, ID=ded5) [Reassembled in #38]
36	4.965609	192.168.1.3	124.220.8.25	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=4440, ID=ded5) [Reassembled in #38]
37	4.965609	192.168.1.3	124.220.8.25	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=5920, ID=ded5) [Reassembled in #38]
38	4.965609	192.168.1.3	124.220.8.25	ICMP	642 Echo (ping) request id=0x0001, seq=18/4608, ttl=64 (no response found!)
65	9.761200	192.168.1.3	124.220.8.25	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=ded6) [Reassembled in #70]
66	9.761200	192.168.1.3	124.220.8.25	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=ded6) [Reassembled in #70]
67	9.761200	192.168.1.3	124.220.8.25	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=2960, ID=ded6) [Reassembled in #70]
68	9.761200	192.168.1.3	124.220.8.25	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=4440, ID=ded6) [Reassembled in #70]
69	9.761200	192.168.1.3	124.220.8.25	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=5920, ID=ded6) [Reassembled in #70]
70	9.761200	192.168.1.3	124.220.8.25	ICMP	642 Echo (ping) request id=0x0001, seq=19/4864, ttl=64 (reply in 76)
71	9.791023	124.220.8.25	192.168.1.3	IPv4	1482 Fragmented IP protocol (proto=ICMP 1, off=0, ID=eec8) [Reassembled in #76]
72	9.792942	124.220.8.25	192.168.1.3	IPv4	1482 Fragmented IP protocol (proto=ICMP 1, off=1448, ID=eec8) [Reassembled in #76]
73	9.792942	124.220.8.25	192.168.1.3	IPv4	1482 Fragmented IP protocol (proto=ICMP 1, off=2896, ID=eec8) [Reassembled in #76]
74	9.792942	124.220.8.25	192.168.1.3	IPv4	1482 Fragmented IP protocol (proto=ICMP 1, off=4344, ID=eec8) [Reassembled in #76]
75	9.792942	124.220.8.25	192.168.1.3	IPv4	1482 Fragmented IP protocol (proto=ICMP 1, off=5792, ID=eec8) [Reassembled in #76]
76	9.792942	124.220.8.25	192.168.1.3	ICMP	802 Echo (ping) reply id=0x0001, seq=19/4864, ttl=51 (request in 70)
85	10.774429	192.168.1.3	124.220.8.25	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=ded7) [Reassembled in #90]
86	10.774429	192.168.1.3	124.220.8.25	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=ded7) [Reassembled in #90]
87	10.774429	192.168.1.3	124.220.8.25	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=2960, ID=ded7) [Reassembled in #90]
88	10.774429	192.168.1.3	124.220.8.25	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=4440, ID=ded7) [Reassembled in #90]
89	10.774429	192.168.1.3	124.220.8.25	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=5920, ID=ded7) [Reassembled in #90]
90	10.774429	192.168.1.3	124.220.8.25	ICMP	642 Echo (ping) request id=0x0001, seq=20/5120, ttl=64 (reply in 97)
92	10.803526	124.220.8.25	192.168.1.3	IPv4	1482 Fragmented IP protocol (proto=ICMP 1, off=1448, ID=eece) [Reassembled in #97]
93	10.803526	124.220.8.25	192.168.1.3	IPv4	1482 Fragmented IP protocol (proto=ICMP 1, off=2896, ID=eece) [Reassembled in #97]
94	10.804327	124.220.8.25	192.168.1.3	IPv4	1482 Fragmented IP protocol (proto=ICMP 1, off=0, ID=eece) [Reassembled in #97]
95	10.804859	124.220.8.25	192.168.1.3	IPv4	1482 Fragmented IP protocol (proto=ICMP 1, off=4344, ID=eece) [Reassembled in #97]
96	10.804859	124.220.8.25	192.168.1.3	IPv4	1482 Fragmented IP protocol (proto=ICMP 1, off=5792, ID=eece) [Reassembled in #97]

- 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

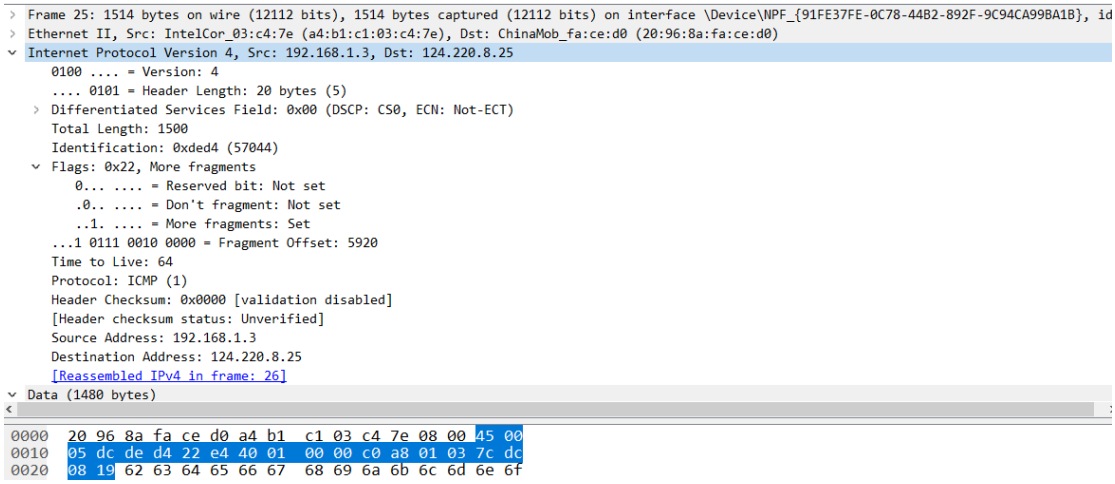
```
> Frame 21: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface \Device\NPF_{91FE37FE-0C78-44B2-892F-9C94CA99BA1B}, id
> Ethernet II, Src: IntelCor_03:c4:7e (a4:b1:c1:03:c4:7e), Dst: ChinaMob_fa:ce:d0 (20:96:8a:fa:ce:d0)
v Internet Protocol Version 4, Src: 192.168.1.3, Dst: 124.220.8.25
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 1500
    Identification: 0xded4 (57044)
  v Flags: 0x20, More fragments
      0... .... = Reserved bit: Not set
      .0.. .... = Don't fragment: Not set
      ..1. .... = More fragments: Set
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 64
    Protocol: ICMP (1)
    Header Checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 192.168.1.3
    Destination Address: 124.220.8.25
    [Reassembled IPv4 in frame: 26]
v Data (1480 bytes)
```

```
0000  20 96 8a fa ce d0 a4 b1  c1 03 c4 7e 08 00 45 00
0010  05 dc de d4 20 00 40 01  00 00 c0 a8 01 03 7c dc
0020  08 19 08 00 eb e3 00 01  00 11 61 62 63 64 65 66
0030  67 68 69 6a 6b 6c 6d 6e  6f 70 71 72 73 74 75 76
0040  77 61 62 63 64 65 66 67  68 69 6a 6b 6c 6d 6e 6f
0050  70 71 72 73 74 75 76 77  61 62 63 64 65 66 67 68
```

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）

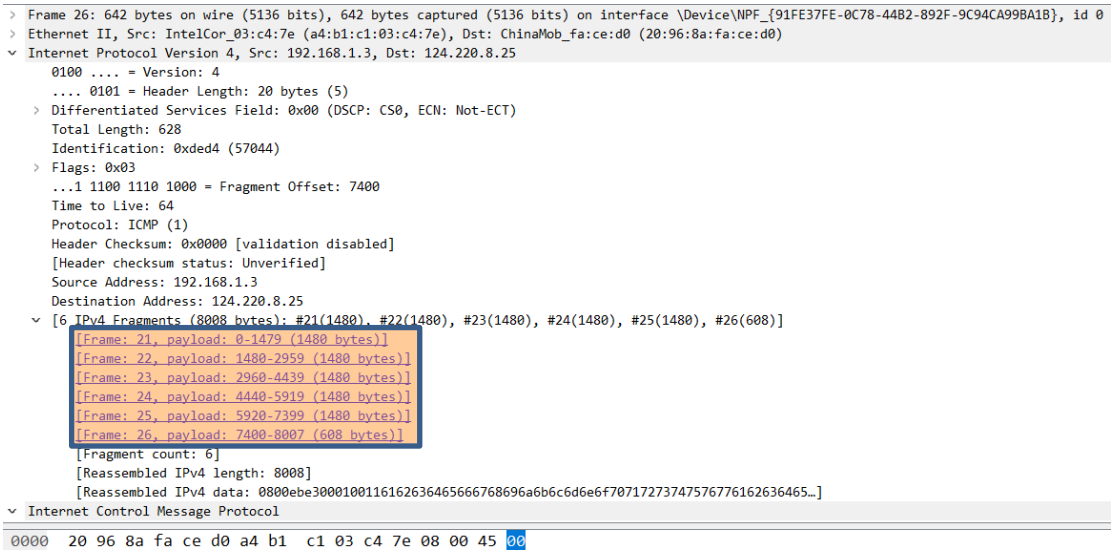
服务类型	00	正常时延，正常吞吐量，正常可靠性
总长度	**05 dc**	**数据分组长 1500 字节（IP 头占 20）**
标识	de d4	独特的包标识
DF(1 bit)	0	允许分段（路由分段，透明，非透明）
MF(1 bit)	**1**	**更多的段。这里是 1，代表分段且不是最后一段**
片位移(13 bit)	**00 00**	**这里是第一个包且没有位移**
生存周期	40	64 跳，一跳一秒
协议	01	用的是 ICMP 协议
头部校验和	00 00	校验正确
源地址	c0 a8 01 03	192.168.1.3
目的地址	7c dc 08 19	124.220.8.25
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




```
Flags: 0x03
    0... .... = Reserved bit: Not set
    .0.. .... = Don't fragment: Not set
    ..0. .... = More fragments: Not set
```

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。

```
来自 124.220.8.25 的回复: 字节=8000 时间=67ms TTL=51
请求超时。
来自 124.220.8.25 的回复: 字节=8000 时间=32ms TTL=51
来自 124.220.8.25 的回复: 字节=8000 时间=30ms TTL=51
```

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**）。

```
Internet Protocol Version 4, Src: 192.168.1.3, Dst: 124.220.8.25
Data (1480 bytes)
    Data: 0800ebe3000100116162636465666768696a6b6c6d6e6f7071727374
    [Length: 1480]
```

```
000   20 96 8a fa ce d0 a4 b1  c1 03 c4 7e 08 00 45 00
010   05 dc de d4 20 00 40 01  00 00 c0 a8 01 03 7c dc
020   08 19 08 00 eb e3 00 01  00 11 61 62 63 64 65 66
```

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

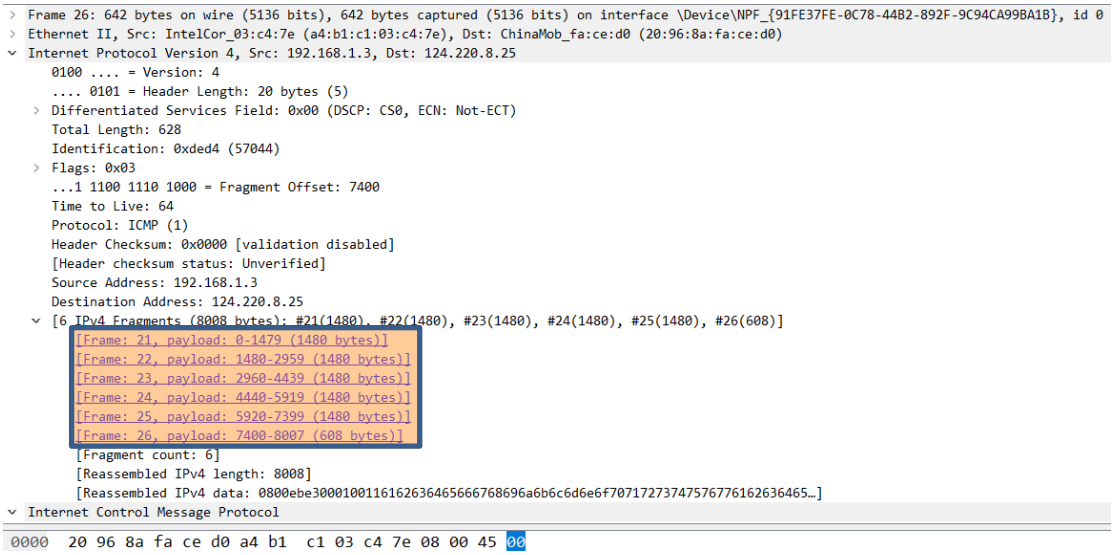
说明：ICMP 所有报文的前 4 个字节都是一样的，但是剩下的其他字节则互不相同。

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）




```
Flags: 0x03
    0... .... = Reserved bit: Not set
    .0.. .... = Don't fragment: Not set
    ..0. .... = More fragments: Not set
```

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

图1 DHCP报文格式

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

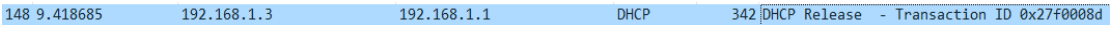
报文都是以广播方式发送的，包括客户端发送的 DHCP Discover 和 DHCP Request 报文，以及 DHCP 服务器发送的 DHCP Offer、DHCP ACK 和 DHCP NAK 报文。当然，如果是由 DHCP 中继器转的报文，则都是以单播方式发送的。另外，IP 地址续约、IP 地址释放的相关报文都是采用单播方式进行发送的。

- **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

```
Dynamic Host Configuration Protocol (Release)
    Message type: Boot Request (1)
    Hardware type: Ethernet (0x01)
    Hardware address length: 6
    Hops: 0
    Transaction ID: 0x27f0008d
    Seconds elapsed: 0
  > Bootp flags: 0x0000 (Unicast)
    Client IP address: 192.168.1.3
    Your (client) IP address: 0.0.0.0
    Next server IP address: 0.0.0.0
    Relay agent IP address: 0.0.0.0
    Client MAC address: IntelCor_03:c4:7e (a4:b1:c1:03:c4:7e)
    Client hardware address padding: 00000000000000000000
    Server host name not given
    Boot file name not given
    Magic cookie: DHCP
  > Option: (53) DHCP Message Type (Release)
  > Option: (54) DHCP Server Identifier (192.168.1.1)
  > Option: (61) Client identifier
  > Option: (255) End
    Padding: 0000000000000000000000000000000000000000000000000000000000000000000000000000…
```

```
0000  20 96 8a fa ce d0 a4 b1  c1 03 c4 7e 08 00 45 00
0010  01 48 97 92 00 00 80 11  00 00 c0 a8 01 03 c0 a8
0020  01 01 00 44 00 43 01 34  84 9a 01 01 06 00 27 f0
0030  00 8d 00 00 00 00 c0 a8  01 03 00 00 00 00 00 00
0040  00 00 00 00 00 00 a4 b1  c1 03 c4 7e 00 00 00 00
0050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0060  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0090  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00c0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00d0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0100  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0110  00 00 00 00 00 00 63 82  53 63 35 01 07 36 04 c0
0120  a8 01 01 3d 07 01 a4 b1  c1 03 c4 7e ff 00 00 00
0130  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0140  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0150  00 00 00 00 00 00
```

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

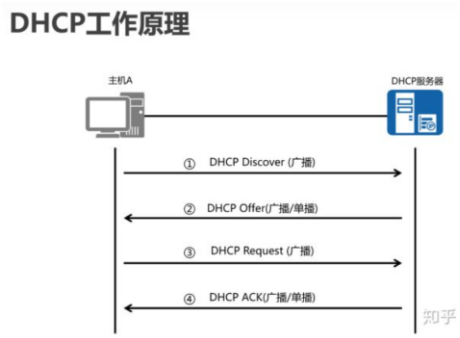
2	00 43	目的端口字段：接收瑞计算机上 UDP 软件使用的端口
2	01 34	长度：UDP 报头和数据的总长度，最小为 8 字节
2	84 9a	校验和：进行检验和计算时，与伪首部一起运算。
1	01	报文的操作类型：分为请求报文和响应报文，=1，请求报文：=2，应答报文。此处为 1，表示请求报文
1	01	DHCP 客户端的 MAC 地址类型：通常为 1，表示以太网的 MAC 地址。
1	06	DHCP 客户端的 MAC 地址长度：此处为 6，则 DHCP 客户端的 MAC 地址长度为
1	00	DHCP 报文经过的 DHCP 中继的数目：此处为 0,表示没有经过中继
4	27 f0 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

```
869 42.969263    0.0.0.0          255.255.255.255    DHCP    342 DHCP Discover - Transaction ID 0x279814c2
900 43.113220    192.168.1.1      192.168.1.3        DHCP    590 DHCP Offer    - Transaction ID 0x279814c2[Malformed Packet]
901 43.113929    0.0.0.0          255.255.255.255    DHCP    344 DHCP Request  - Transaction ID 0x279814c2
907 43.263420    192.168.1.1      192.168.1.3        DHCP    590 DHCP ACK      - Transaction ID 0x279814c2[Malformed Packet]
```

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

```
869 42.969263    0.0.0.0          255.255.255.255    DHCP    342 DHCP Discover - Transaction ID 0x279814c2
```

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 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

```
> Ethernet II, Src: IntelCor_03:c4:7e (a4:b1:c1:03:c4:7e), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
> Internet Protocol Version 4, Src: 0.0.0.0, Dst: 255.255.255.255
> User Datagram Protocol, Src Port: 68, Dst Port: 67
v Dynamic Host Configuration Protocol (Discover)
    Message type: Boot Request (1)
    Hardware type: Ethernet (0x01)
    Hardware address length: 6
    Hops: 0
    Transaction ID: 0x279814c2
    Seconds elapsed: 0
  > Bootp flags: 0x0000 (Unicast)
    Client IP address: 0.0.0.0
    Your (client) IP address: 0.0.0.0
    Next server IP address: 0.0.0.0
    Relay agent IP address: 0.0.0.0
    Client MAC address: IntelCor_03:c4:7e (a4:b1:c1:03:c4:7e)
    Client hardware address padding: 00000000000000000000
    Server host name not given
    Boot file name not given
    Magic cookie: DHCP
  > Option: (53) DHCP Message Type (Discover)
  > Option: (61) Client identifier
  > Option: (50) Requested IP Address (192.168.1.3)
  > Option: (12) Host Name
  > Option: (60) Vendor class identifier
  > Option: (55) Parameter Request List
```

```
0000  ff ff ff ff ff ff a4 b1  c1 03 c4 7e 08 00 45 00
0010  01 48 5f 54 00 00 80 11  00 00 00 00 00 00 ff ff
0020  ff ff 00 44 00 43 01 34  db f8 01 01 06 00 27 98
0030  14 c2 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0040  00 00 00 00 00 00 a4 b1  c1 03 c4 7e 00 00 00 00
0050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0060  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0090  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00c0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00d0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0100  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0110  00 00 00 00 00 00 63 82  53 63 35 01 01 3d 07 01
0120  a4 b1 c1 03 c4 7e 32 04  c0 a8 01 03 0c 02 50 43
0130  3c 08 4d 53 46 54 20 35  2e 30 37 0e 01 03 06 0f
0140  1f 21 2b 2c 2e 2f 77 79  f9 fc ff 00 00 00 00 00   ·!
0150  00 00 00 00 00 00
```

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。

```
0000  ff ff ff ff ff ff a4 b1  c1 03 c4 7e 08 00 45 00
```

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

```
> Ethernet II, Src: ChinaMob_fa:ce:d0 (20:96:8a:fa:ce:d0), Dst: IntelCor_03:c4:7e (a4:b1:c1:03:c4:7e)
> Internet Protocol Version 4, Src: 192.168.1.1, Dst: 192.168.1.3
> User Datagram Protocol, Src Port: 67, Dst Port: 68
v Dynamic Host Configuration Protocol (Offer)
    Message type: Boot Reply (2)
    Hardware type: Ethernet (0x01)
    Hardware address length: 6
    Hops: 0
    Transaction ID: 0x279814c2
    Seconds elapsed: 0
  > Bootp flags: 0x0000 (Unicast)
    Client IP address: 0.0.0.0
    Your (client) IP address: 192.168.1.3
    Next server IP address: 0.0.0.0
    Relay agent IP address: 0.0.0.0
    Client MAC address: IntelCor_03:c4:7e (a4:b1:c1:03:c4:7e)
    Client hardware address padding: 00000000000000000000
    Server host name not given
    Boot file name not given
    Magic cookie: DHCP
  > Option: (53) DHCP Message Type (Offer)
  > Option: (54) DHCP Server Identifier (192.168.1.1)
```

```
0000  a4 b1 c1 03 c4 7e 20 96  8a fa ce d0 08 00 45 00
0010  02 40 00 00 00 00 40 11  f5 58 c0 a8 01 01 c0 a8
0020  01 03 00 43 00 44 02 2c  4a 50 02 01 06 00 27 98
0030  14 c2 00 00 00 00 00 00  00 00 c0 a8 01 03 00 00
0040  00 00 00 00 00 00 a4 b1  c1 03 c4 7e 00 00 00 00
0050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0060  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0090  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00c0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00d0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0100  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0110  00 00 00 00 00 00 63 82  53 63 35 01 02 36 04 c0
0120  a8 01 01 7d 26 00 00 23  01 08 4d 65 64 69 61 74
0130  65 6b 02 06 48 47 57 2d  43 54 03 03 65 38 63 0a
0140  02 00 29 0b 02 00 2b 0c  02 0e 74 33 04 00 01 51
0150  80 01 04 ff ff ff 00 03  04 c0 a8 01 01 06 04 c0
0160  a8 01 01 2c 08 c0 a8 01  01 c0 a8 01 01 ff 00 00
0170  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0180  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0190  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
01a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

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）

```
0000  a4 b1 c1 03 c4 7e 20 96  8a fa ce d0 08 00 45 00
```

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

```
901 43.113929      0.0.0.0          255.255.255.255      DHCP      344 DHCP Request  - Transaction ID 0x279814c2
```

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

```
> Ethernet II, Src: IntelCor_03:c4:7e (a4:b1:c1:03:c4:7e), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
> Internet Protocol Version 4, Src: 0.0.0.0, Dst: 255.255.255.255
> User Datagram Protocol, Src Port: 68, Dst Port: 67
v Dynamic Host Configuration Protocol (Request)
    Message type: Boot Request (1)
    Hardware type: Ethernet (0x01)
    Hardware address length: 6
    Hops: 0
    Transaction ID: 0x279814c2
    Seconds elapsed: 0
  > Bootp flags: 0x0000 (Unicast)
    Client IP address: 0.0.0.0
    Your (client) IP address: 0.0.0.0
    Next server IP address: 0.0.0.0
    Relay agent IP address: 0.0.0.0
    Client MAC address: IntelCor_03:c4:7e (a4:b1:c1:03:c4:7e)
    Client hardware address padding: 00000000000000000000
    Server host name not given
    Boot file name not given
    Magic cookie: DHCP
  > Option: (53) DHCP Message Type (Request)
  > Option: (61) Client identifier
  > Option: (50) Requested IP Address (192.168.1.3)
  > Option: (54) DHCP Server Identifier (192.168.1.1)
  > Option: (12) Host Name
  > Option: (81) Client Fully Qualified Domain Name
```

```
0000  ff ff ff ff ff ff a4 b1  c1 03 c4 7e 08 00 45 00
0010  01 4a 5f 55 00 00 80 11  00 00 00 00 00 00 ff ff
0020  ff ff 00 44 00 43 01 36  69 d5 01 01 06 00 27 98
0030  14 c2 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0040  00 00 00 00 00 00 a4 b1  c1 03 c4 7e 00 00 00 00
0050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0060  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0090  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00c0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00d0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0100  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0110  00 00 00 00 00 00 63 82  53 63 35 01 03 3d 07 01
0120  a4 b1 c1 03 c4 7e 32 04  c0 a8 01 03 36 04 c0 a8
0130  01 01 0c 02 50 43 51 05  00 00 00 50 43 3c 08 4d
0140  53 46 54 20 35 2e 30 37  0e 01 03 06 0f 1f 21 2b   SFT
0150  2c 2e 2f 77 79 f9 fc ff                           ,./
```

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。

```
0000  ff ff ff ff ff ff a4 b1  c1 03 c4 7e 08 00 45 00
```

第四个包：DHCP ACK 包

```
907 43.263420   192.168.1.1   192.168.1.3   DHCP   590 DHCP ACK   - Transaction ID 0x279814c2[Malformed Packet]
```

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

```
> Ethernet II, Src: ChinaMob_fa:ce:d0 (20:96:8a:fa:ce:d0), Dst: IntelCor_03:c4:7e (a4:b1:c1:03:c4:7e)
> Internet Protocol Version 4, Src: 192.168.1.1, Dst: 192.168.1.3
> User Datagram Protocol, Src Port: 67, Dst Port: 68
v Dynamic Host Configuration Protocol (ACK)
    Message type: Boot Reply (2)
    Hardware type: Ethernet (0x01)
    Hardware address length: 6
    Hops: 0
    Transaction ID: 0x279814c2
    Seconds elapsed: 0
  > Bootp flags: 0x0000 (Unicast)
    Client IP address: 0.0.0.0
    Your (client) IP address: 192.168.1.3
    Next server IP address: 0.0.0.0
    Relay agent IP address: 0.0.0.0
    Client MAC address: IntelCor_03:c4:7e (a4:b1:c1:03:c4:7e)
    Client hardware address padding: 00000000000000000000
    Server host name not given
    Boot file name not given
    Magic cookie: DHCP
  > Option: (53) DHCP Message Type (ACK)
  > Option: (54) DHCP Server Identifier (192.168.1.1)
```

```
0000  a4 b1 c1 03 c4 7e 20 96  8a fa ce d0 08 00 45 00
0010  02 40 00 00 00 00 40 11  f5 58 c0 a8 01 01 c0 a8
0020  01 03 00 43 00 44 02 2c  47 50 02 01 06 00 27 98
0030  14 c2 00 00 00 00 00 00  00 00 c0 a8 01 03 00 00
0040  00 00 00 00 00 00 a4 b1  c1 03 c4 7e 00 00 00 00
0050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0060  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0090  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00c0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00d0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0100  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0110  00 00 00 00 00 00 63 82  53 63 35 01 05 36 04 c0
0120  a8 01 01 7d 26 00 00 23  01 08 4d 65 64 69 61 74
0130  65 6b 02 06 48 47 57 2d  43 54 03 03 65 38 63 0a
0140  02 00 29 0b 02 00 2b 0c  02 0e 74 33 04 00 01 51
0150  80 01 04 ff ff ff 00 03  04 c0 a8 01 01 06 04 c0
0160  a8 01 01 2c 08 c0 a8 01  01 c0 a8 01 01 ff 00 00
0170  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0180  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0190  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
01a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

这个源地址和目的地址依旧用的是以太网地址。

```
0000  a4 b1 c1 03 c4 7e 20 96  8a fa ce d0 08 00 45 00
```

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。

```
[spike@localhost ~]$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.7  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::5861:180b:ac17:84e  prefixlen 64  scopeid 0x20<link>
        inet6 2409:8a30:707a:7d30:6a66:376c:edf4:71a7  prefixlen 64  scopeid 0x0
```

（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 地址获取物理地址的一个 TCP/IP 协议。主机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

地址解析协议是通过报文工作的。报文包括如下字段：

**ARP报文格式
**

硬件类型		协议类型
硬件地址长度	协议长度	操作类型
发送方硬件地址（0-3字节）		
发送方硬件地址（4-5字节）		发送方IP地址（0-1字节）
发送方IP地址（2-3字节）		目标硬件地址（0-1字节）
目标硬件地址（2-5字节）		
目标IP地址（0-3字节）		

- 硬件类型：指明了发送方想知道的硬件接口类型，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。

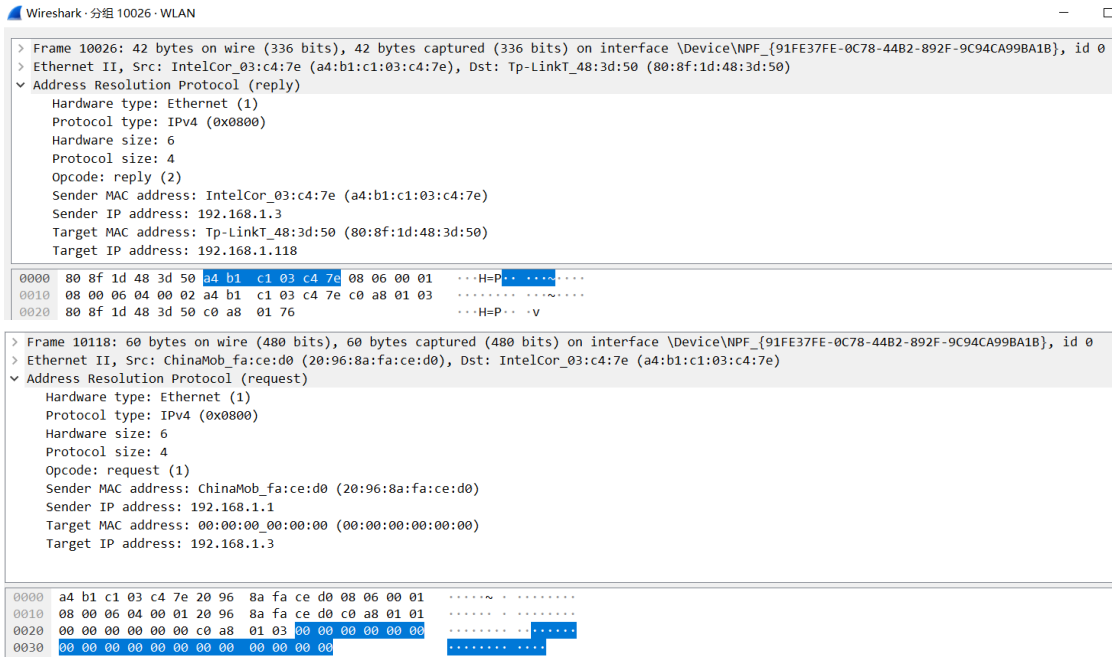
捕获 ARP 包

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：

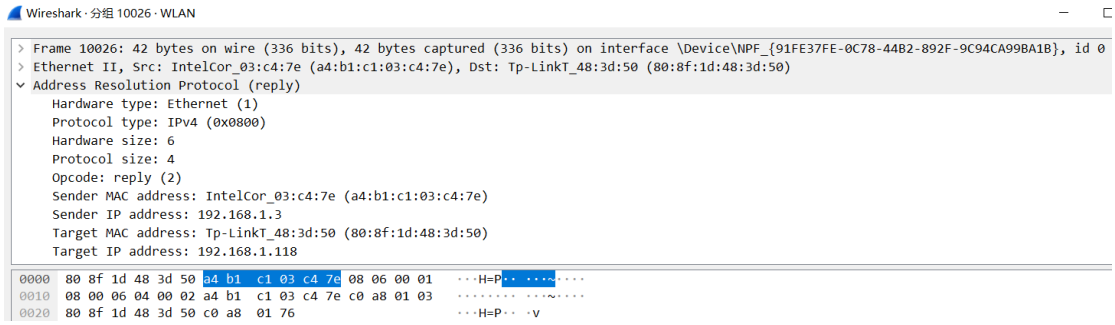
3. 打开命令提示符窗口，输入 ping **www.bing.com**，回车执行：

```
C:\Users\Spike>ping www.bing.com

正在 Ping china.bing123.com [202.89.233.101] 具有 32 字节的数据:
来自 202.89.233.101 的回复: 字节=32 时间=41ms TTL=114
来自 202.89.233.101 的回复: 字节=32 时间=40ms TTL=114
来自 202.89.233.101 的回复: 字节=32 时间=44ms TTL=114
来自 202.89.233.101 的回复: 字节=32 时间=40ms TTL=114
```

4. 观察 Wireshark 捕获的数据报：



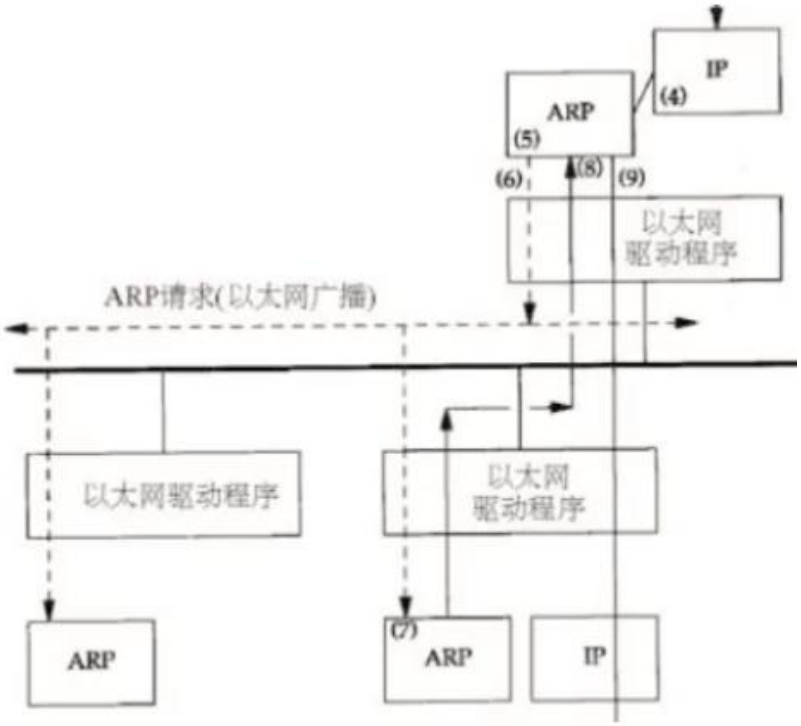
分析 ARP 包（以本机发送为例）



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已经被使用，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。

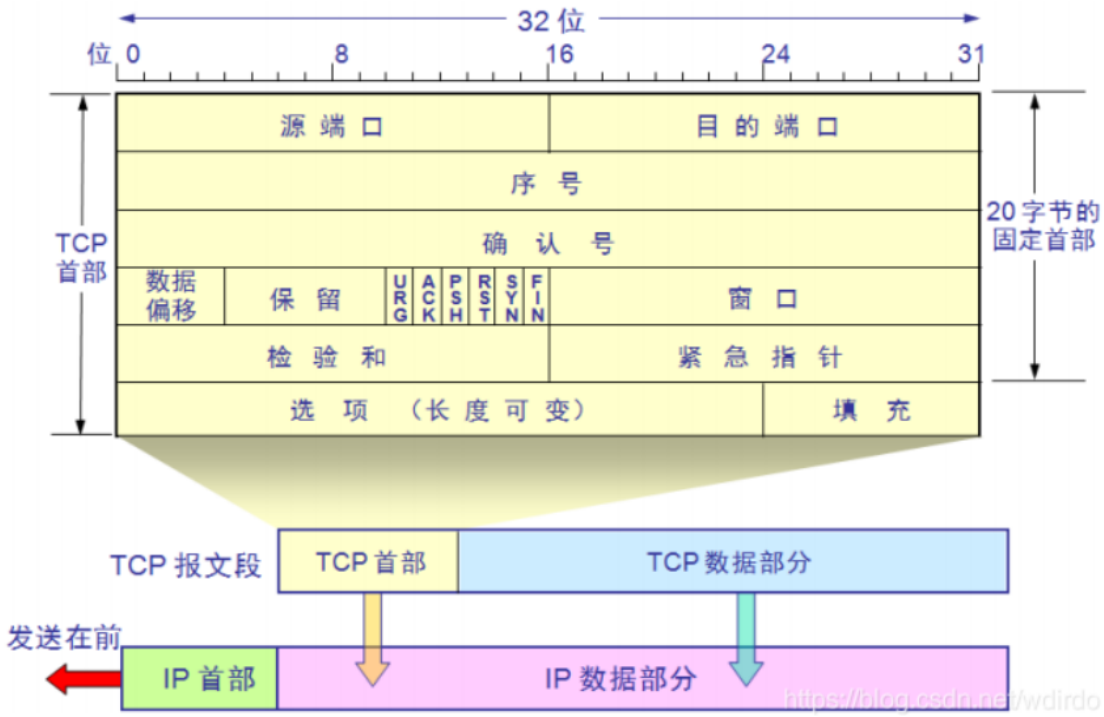
互联网络与单个网络有很大的不同，因为互联网络的不同部分可能有截然不同的拓扑结构、带宽、延迟、数据包大小和其他参数。TCP 的设计目标是能够动态地适应互联网络的这些特性，而且具备面对各种故障时的健壮性。

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）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



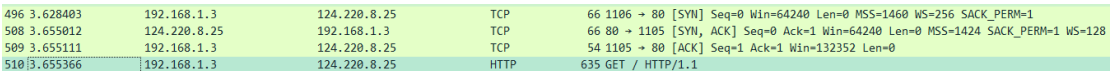
追踪 TCP 包

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。

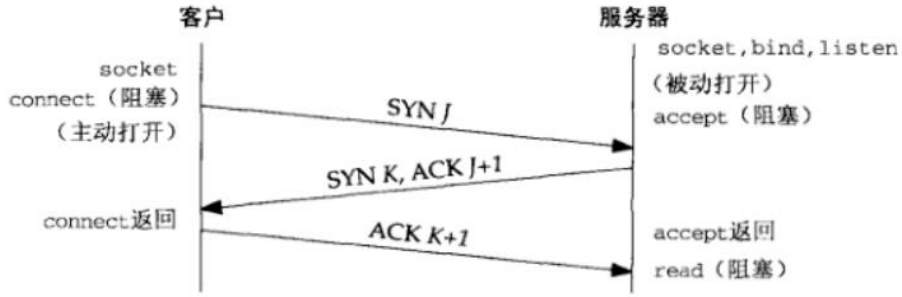
因为项目要外部访问数据库，且服务器没配好就先用这个顶着了）

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）



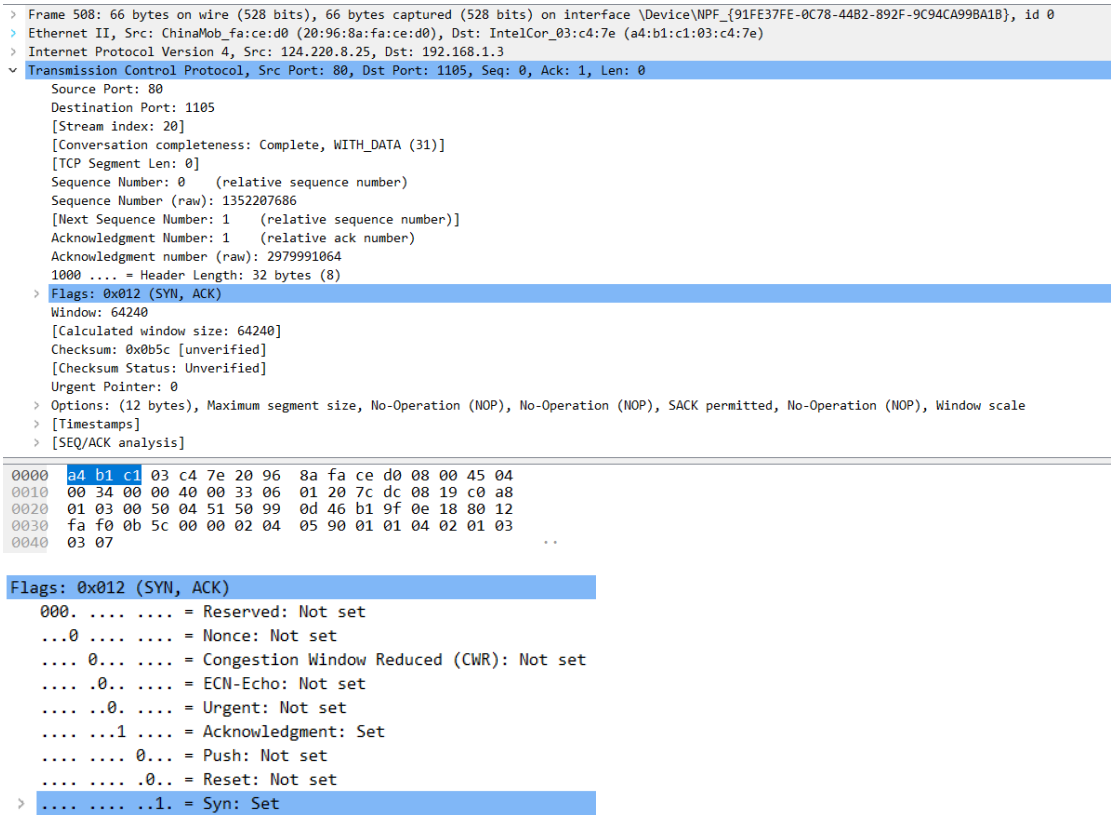
第一次握手

```
> Frame 494: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface \Device\NPF_{91FE37FE-0C78-44B2-892F-9C94CA99BA1B}, id 0
> Ethernet II, Src: IntelCor_03:c4:7e (a4:b1:c1:03:c4:7e), Dst: ChinaMob_fa:ce:d0 (20:96:8a:fa:ce:d0)
> Internet Protocol Version 4, Src: 192.168.1.3, Dst: 124.220.8.25
v Transmission Control Protocol, Src Port: 1105, Dst Port: 80, Seq: 0, Len: 0
    Source Port: 1105
    Destination Port: 80
    [Stream index: 20]
    [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 0]
    Sequence Number: 0    (relative sequence number)
    Sequence Number (raw): 2979991063
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 0
    Acknowledgment number (raw): 0
    1000 .... = Header Length: 32 bytes (8)
  v Flags: 0x002 (SYN)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
      .... 0... .... = Congestion Window Reduced (CWR): Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...0 .... = Acknowledgment: Not set
      .... .... 0... = Push: Not set
      .... .... .0.. = Reset: Not set
    > .... .... ..1. = Syn: Set

0000  20 96 8a fa ce d0 a4 b1  c1 03 c4 7e 08 00 45 00
0010  00 34 65 71 40 00 80 06  00 00 c0 a8 01 03 7c dc
0020  08 19 04 51 00 50 b1 9f  0e 17 00 00 00 00 80 02
0030  fa f0 46 c7 00 00 02 04  05 b4 01 03 03 08 01 01
```

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**。

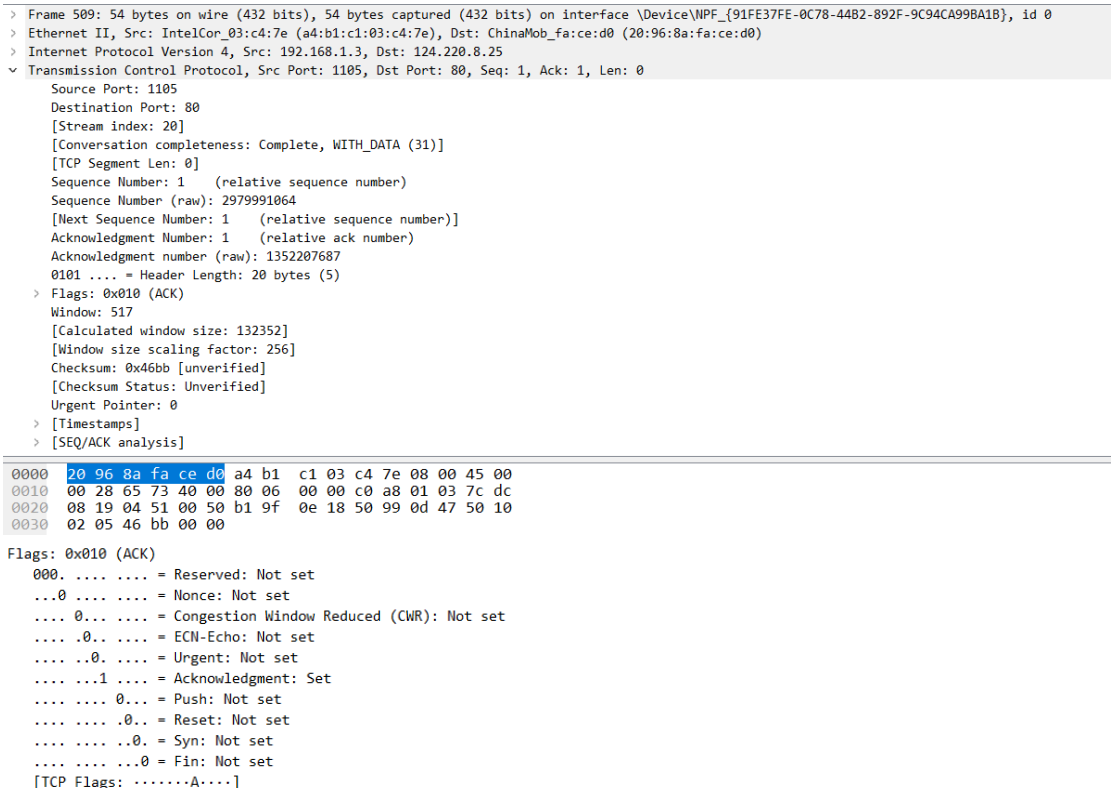
第一次： `Sequence Number (raw): 2979991063`（随机序列）

第二次： `Acknowledgment number (raw): 2979991064`

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： `Sequence Number (raw): 2979991063`（随机序列）

第二次： `Acknowledgment number (raw): 2979991064`（+1）

第三次： `Sequence Number (raw): 2979991064` **（随机序列不随机，和前两个包联系起来）**

第二次： `Sequence Number (raw): 1352207686`（随机序列）

第三次： `Acknowledgment number (raw): 1352207687`（+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：

```
5679 50.628184    192.168.1.3      36.152.44.96     TCP      54 11500 → 80 [FIN, ACK] Seq=497 Ack=1156 Win=65536 Len=0
5695 50.645813    36.152.44.96     192.168.1.3      TCP      60 80 → 11500 [ACK] Seq=1156 Ack=498 Win=30208 Len=0
5696 50.645813    36.152.44.96     192.168.1.3      TCP      60 80 → 11500 [FIN, ACK] Seq=1156 Ack=498 Win=30208 Len=0
5697 50.645921    192.168.1.3      36.152.44.96     TCP      54 11500 → 80 [ACK] Seq=498 Ack=1157 Win=65536 Len=0
```

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

```
Transmission Control Protocol, Src Port: 11500, Dst Port: 80, Seq: 497, Ack: 1156, Len: 0
    Source Port: 11500
    Destination Port: 80
    [Stream index: 143]
    [Conversation completeness: Complete, WITH_DATA (63)]
    [TCP Segment Len: 0]
    Sequence Number: 497    (relative sequence number)
    Sequence Number (raw): 3613916069
    [Next Sequence Number: 498    (relative sequence number)]
    Acknowledgment Number: 1156    (relative ack number)
    Acknowledgment number (raw): 1809320255
    0101 .... = Header Length: 20 bytes (5)
  Flags: 0x011 (FIN, ACK)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
      .... 0... .... = Congestion Window Reduced (CWR): Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...1 .... = Acknowledgment: Set
      .... .... 0... = Push: Not set
      .... .... .0.. = Reset: Not set
      .... .... ..0. = Syn: Not set
    > .... .... ...1 = Fin: Set
```

```
0000  20 96 8a fa ce d0 a4 b1  c1 03 c4 7e 08 00 45 00   ·· ····· ···~··E·
0010  00 28 21 2d 40 00 80 06  00 00 c0 a8 01 03 24 98   ·(!-@··· ·····$·
0020  2c 60 2c ec 00 50 d7 67  fb a5 6b d8 09 3f 50 11   ,`,··P·g ··k··?P·
0030  01 00 12 be 00 00                                  ·····
```

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

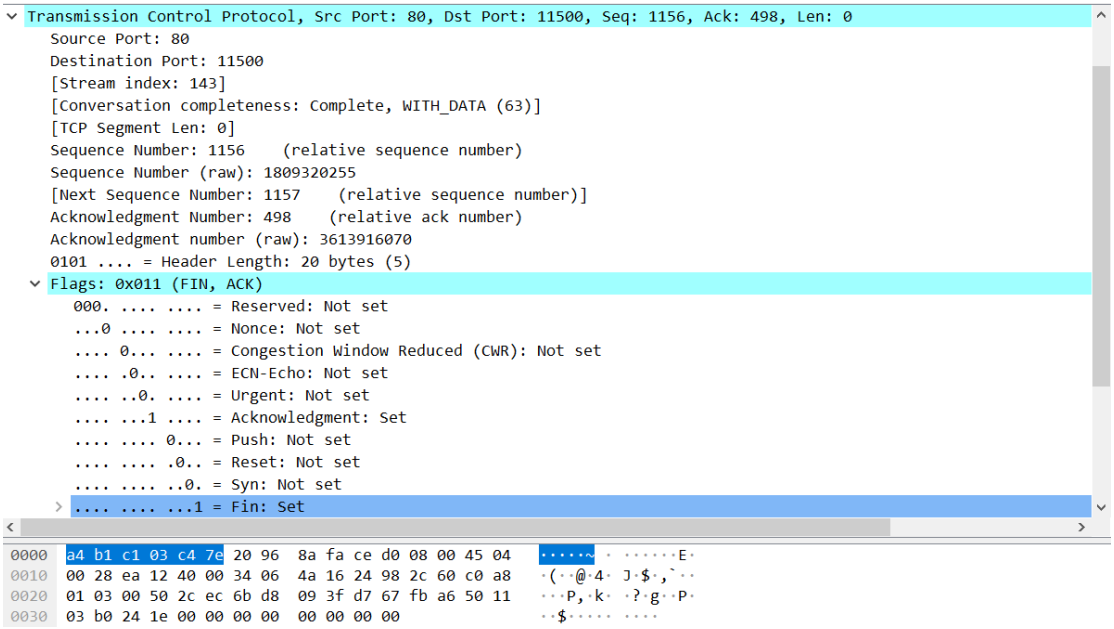
```
Transmission Control Protocol, Src Port: 80, Dst Port: 11500, Seq: 1156, Ack: 498, Len: 0
    Source Port: 80
    Destination Port: 11500
    [Stream index: 143]
    [Conversation completeness: Complete, WITH_DATA (63)]
    [TCP Segment Len: 0]
    Sequence Number: 1156    (relative sequence number)
    Sequence Number (raw): 1809320255
    [Next Sequence Number: 1156    (relative sequence number)]
    Acknowledgment Number: 498    (relative ack number)
    Acknowledgment number (raw): 3613916070
    0101 .... = Header Length: 20 bytes (5)
  Flags: 0x010 (ACK)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
      .... 0... .... = Congestion Window Reduced (CWR): Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...1 .... = Acknowledgment: Set
      .... .... 0... = Push: Not set
      .... .... .0.. = Reset: Not set
      .... .... ..0. = Syn: Not set
      .... .... ...0 = Fin: Not set
```

```
0000  a4 b1 c1 03 c4 7e 20 96  8a fa ce d0 08 00 45 04   ····~ · ·····E·
0010  00 28 ea 11 40 00 34 06  4a 17 24 98 2c 60 c0 a8   ·(··@·4· J·$·,`··
0020  01 03 00 50 2c ec 6b d8  09 3f d7 67 fb a6 50 10   ···P,·k· ·?·g··P·
0030  03 b0 24 1f 00 00 00 00  00 00 00 00              ··$····· ····
```

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

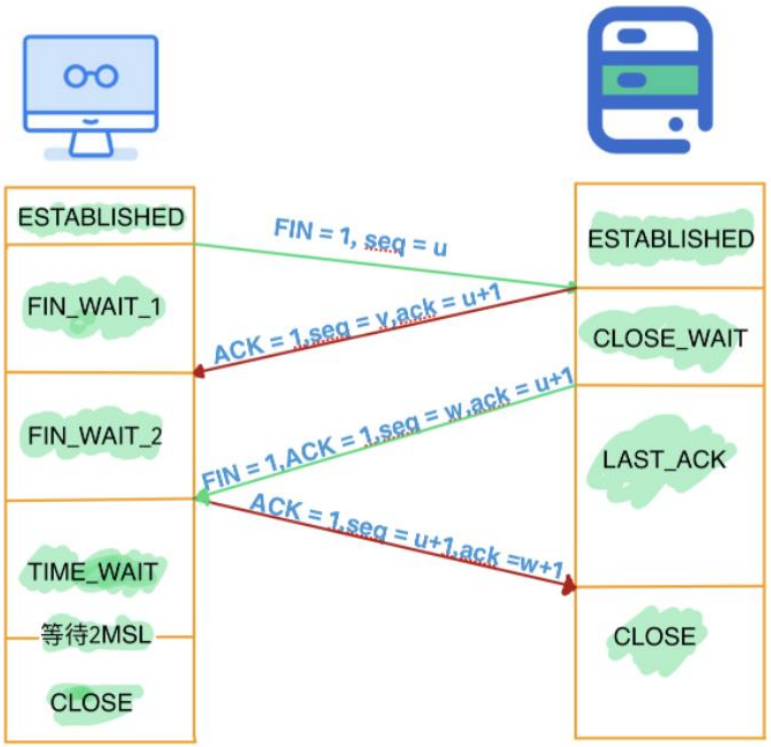
```
Transmission Control Protocol, Src Port: 11500, Dst Port: 80, Seq: 498, Ack: 1157, Len: 0
    Source Port: 11500
    Destination Port: 80
    [Stream index: 143]
    [Conversation completeness: Complete, WITH_DATA (63)]
    [TCP Segment Len: 0]
    Sequence Number: 498    (relative sequence number)
    Sequence Number (raw): 3613916070
    [Next Sequence Number: 498    (relative sequence number)]
    Acknowledgment Number: 1157    (relative ack number)
    Acknowledgment number (raw): 1809320256
    0101 .... = Header Length: 20 bytes (5)
  Flags: 0x010 (ACK)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
      .... 0... .... = Congestion Window Reduced (CWR): Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...1 .... = Acknowledgment: Set
      .... .... 0... = Push: Not set
      .... .... .0.. = Reset: Not set
      .... .... ..0. = Syn: Not set
      .... .... ...0 = Fin: Not set
```

```
0000  20 96 8a fa ce d0 a4 b1  c1 03 c4 7e 08 00 45 00    ·····   ···~··E·
0010  00 28 21 35 40 00 80 06  00 00 c0 a8 01 03 24 98   ·(!5@···  ·····$·
0020  2c 60 2c ec 00 50 d7 67  fb a6 6b d8 09 40 50 10   ,`,··P·g  ··k··@P·
0030  01 00 12 be 00 00                                  ·····
```

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

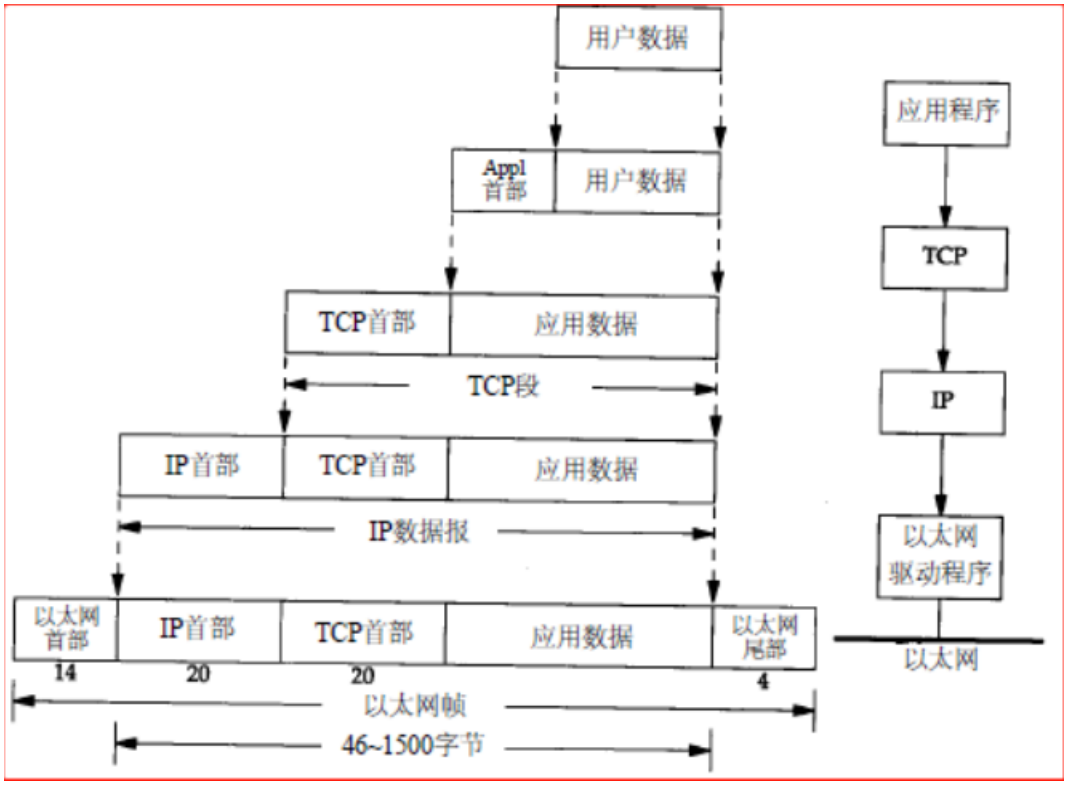


Nginx 反向代理的四次挥手?

1740	44.314263	124.220.8.25	192.168.1.3	TCP	60 80 → 11232 [FIN, ACK] Seq=191 Ack=583 Win=64128 Len=0
1741	44.314325	192.168.1.3	124.220.8.25	TCP	54 11232 → 80 [ACK] Seq=583 Ack=192 Win=66560 Len=0
1742	44.318255	124.220.8.25	192.168.1.3	TCP	60 80 → 11231 [FIN, ACK] Seq=1 Ack=2 Win=64256 Len=0
1743	44.318336	192.168.1.3	124.220.8.25	TCP	54 11231 → 80 [ACK] Seq=2 Ack=2 Win=66816 Len=0

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。

1、MAC帧格式

6	6	2		4
目的地址	源地址	类型	数据（46~1500）	FCS

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。

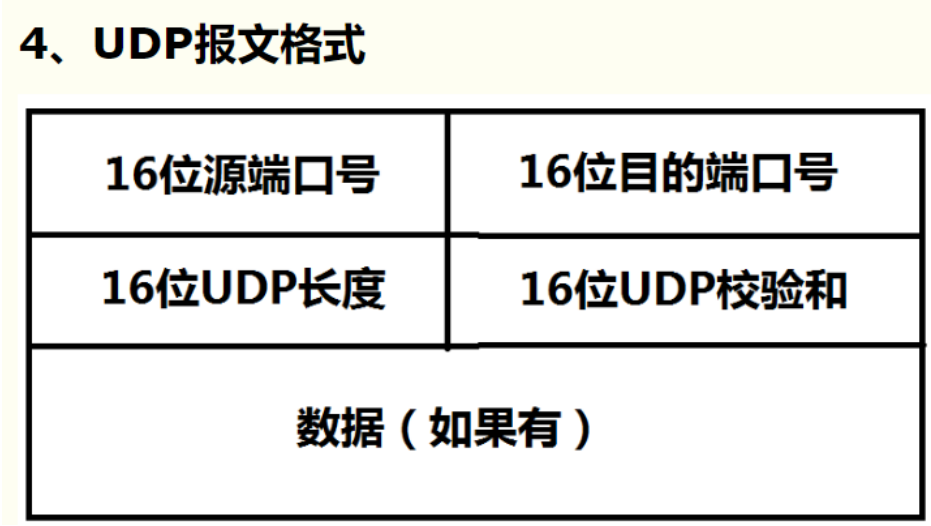
IP 包帧结构

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。