浙江大学

本科实验报告

课程名称：　计算机网络

实验名称：　网络协议分析

姓　　名：

学　　院：　计算机学院

系：

专　　业：

学　　号：

指导教师：

2020 年 11 月 19 日

浙江大学实验报告

一、实验目的

- 学习使用 Wireshark 抓包工具
- 观察和理解常见网络协议的交互过程
- 理解数据包分层结构和格式

二、实验内容

- Wireshark 是 PC 上使用最广泛的免费抓包工具，可以分析大多数常见的协议数据包,有 Windows 版本和 Mac 版本，可以免费从网上下载
- 掌握网络协议分析软件 Wireshark 的使用，学会配置过滤器
- 观察所在网络出现的各类网络协议，了解其种类和分层结构
- 观察捕获到的数据包格式，理解各字段含义
- 根据要求配置 Wireshark，捕获某一类协议的数据包，并分析解读

三、主要仪器设备

- 联网的 PC 机、Windows、Linux 或 Mac 操作系统、浏览器软件
- WireShark 协议分析软件

四、操作方法与实验步骤

- 安装网络包捕获软件 Wireshark
- 配置网络包捕获软件，捕获所有机器的数据包
- 观察捕获到的数据包，并对照解析结果和原始数据包
- 配置网络包捕获软件，只捕获特定 IP 或特定类型的包
- 抓取以下通信协议数据包，观察通信过程和数据包格式
  - PING：测试一个目标地址是否可达
  - TRACE ROUTE：跟踪一个目标地址的途经路由
  - NSLOOKUP：查询一个域名
  - HTTP：访问一个网页

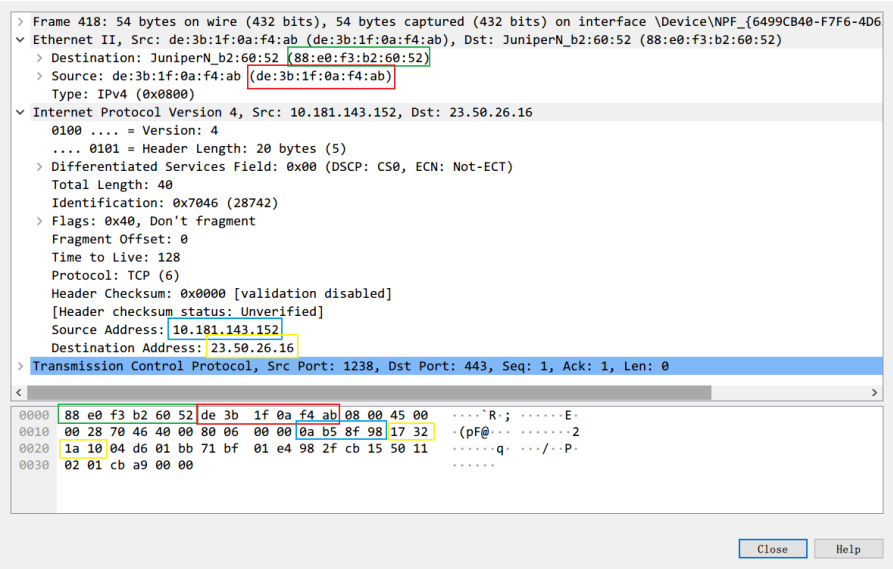
五、实验数据记录和处理

✧ Part One

1. 运行 Wireshark 软件，开始捕获数据包，列出你看到的协议名字（至少 5 个）。

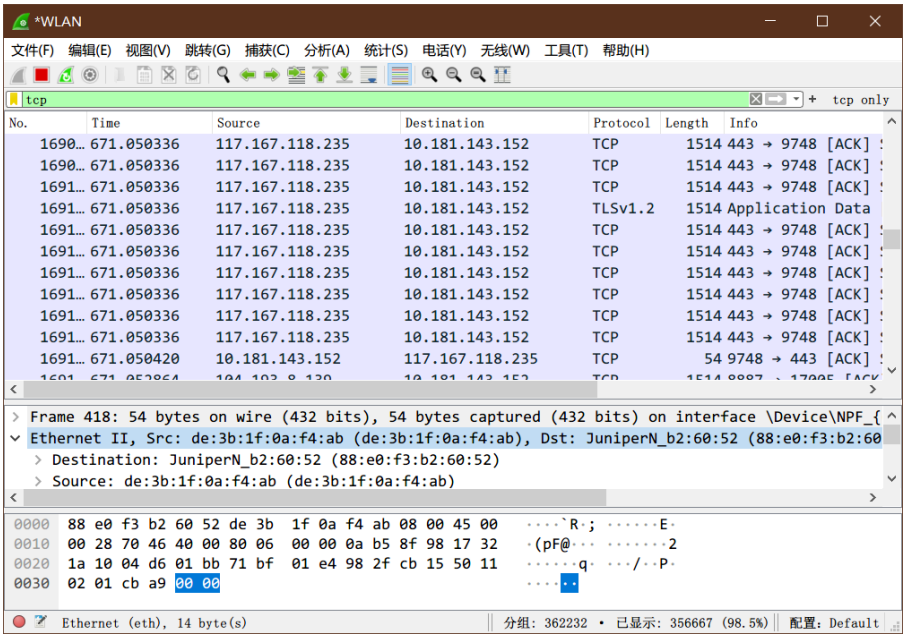
   协议名：TCP, DHCP, OICQ, ARP, HTTP

2. 找一个包含 IP 的数据包，这个数据包有 4 层？最高层协议是 TCP，从 Ethernet 开始往上，各层协议的名字分别为：Ethernet II, IPv4, TCP。展开 IP 层协议，标出源 IP 地址、目标 IP 地址及其在数据包中的具体位置，展开 Ethernet 层，标出源 MAC 地址和目标 MAC 地址及其在数据包中的具体位置。



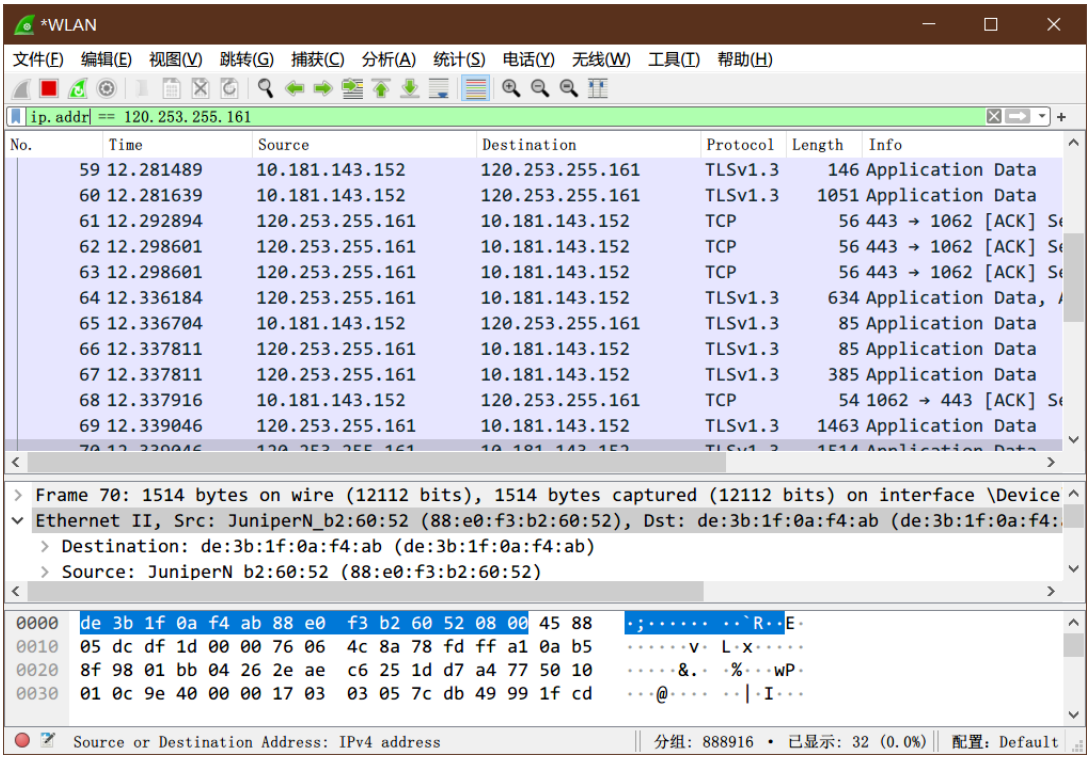
3. 配置应用显示过滤器，让界面只显示某一协议类型的数据包（输入协议名称）。

   使用的过滤器：tcp，希望显示的协议类型：TCP。



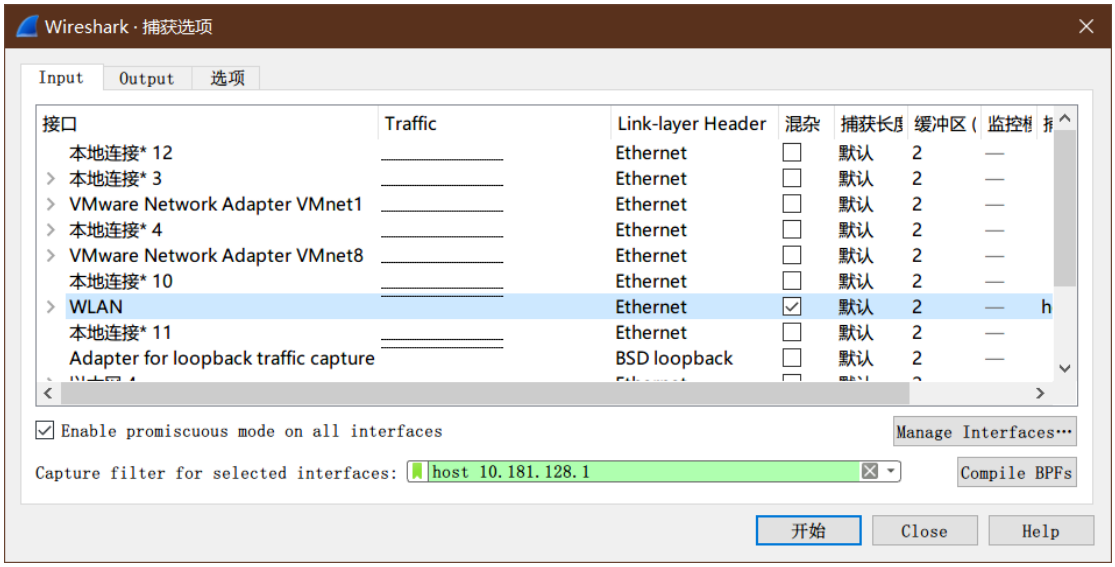
4. 配置应用显示过滤器，让界面只显示某个 IP 地址的数据包（ip.addr==x.x.x.x）。

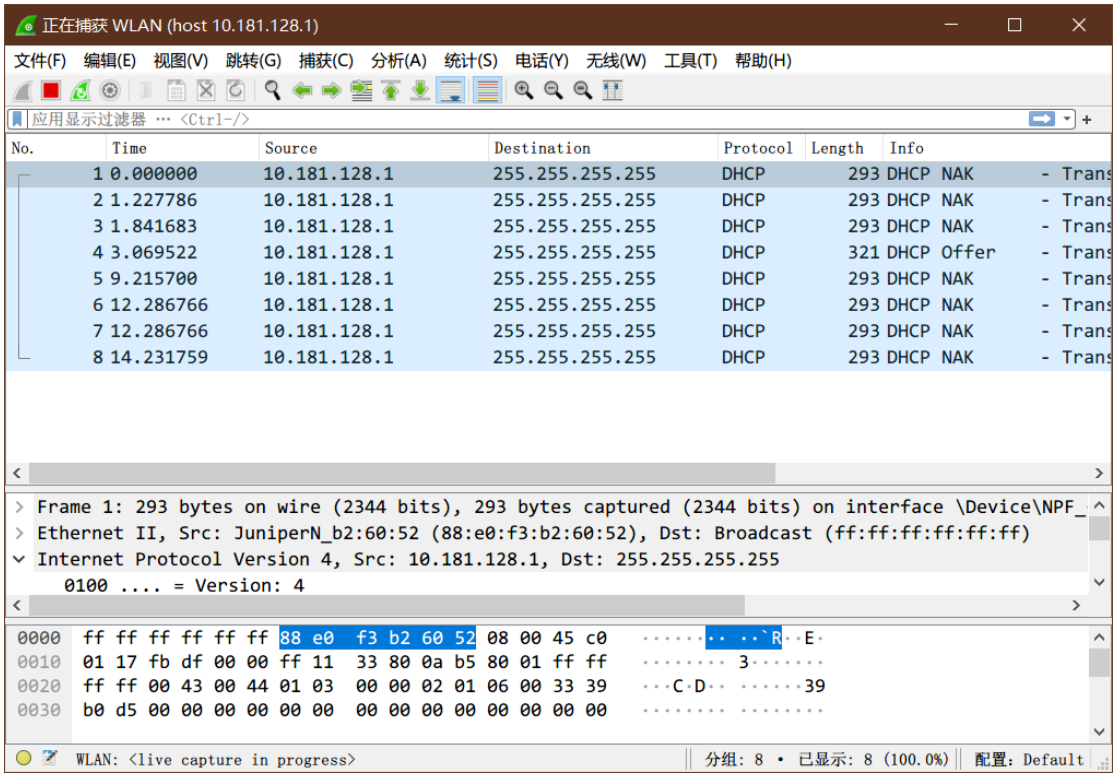
使用的过滤器：ip.addr == 120.253.255.161，希望显示的 IP 地址：120.253.255.161。



5. 配置捕获过滤器，只捕获某个 IP 地址的数据包（host x.x.x.x）。

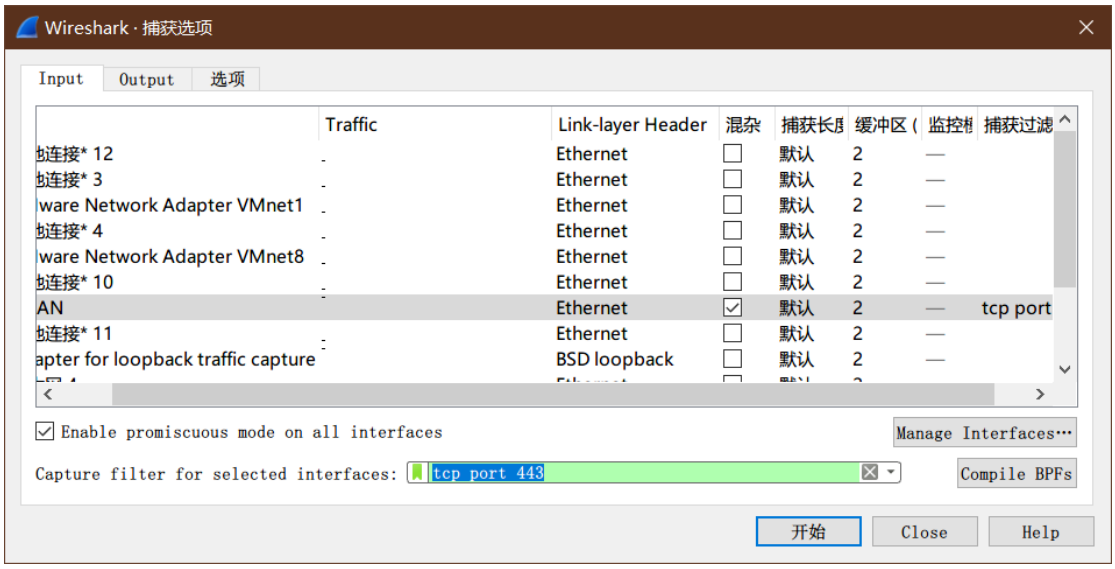
使用的过滤器：host 10.181.128.1，希望捕获的 IP 地址：10.181.128.1。

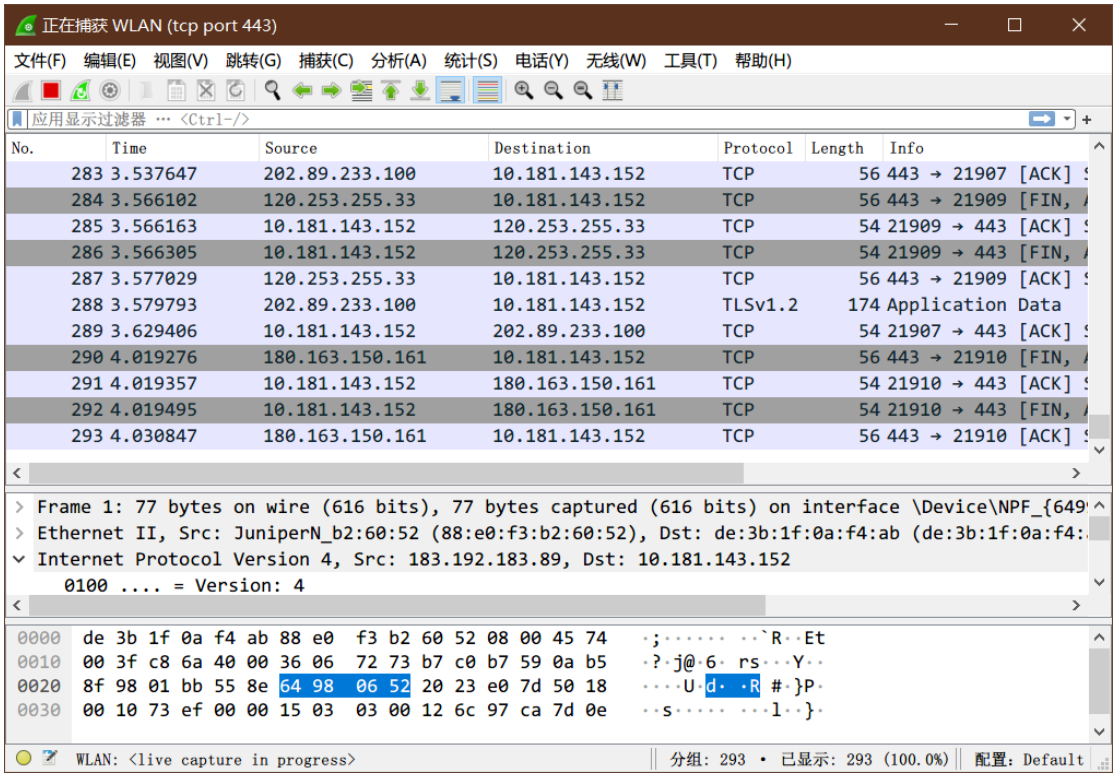




6. 配置捕获过滤器，只捕获某类协议的数据包（tcp port xx 或者 udp port xx）。

使用的过滤器：__tcp port 443__，希望捕获的协议类型：__HTTPS__。



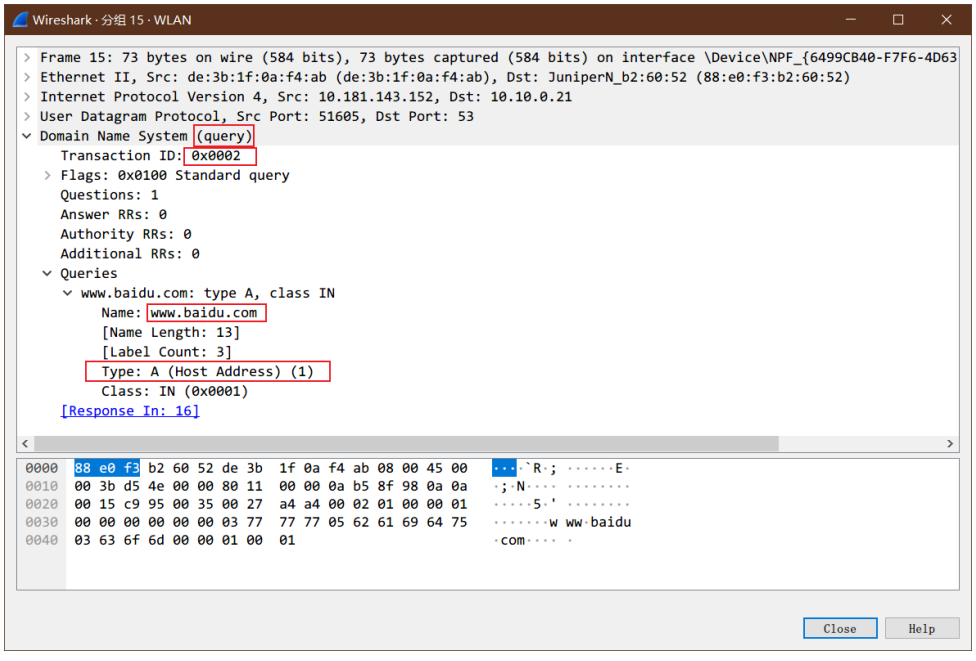


✧ Part Two

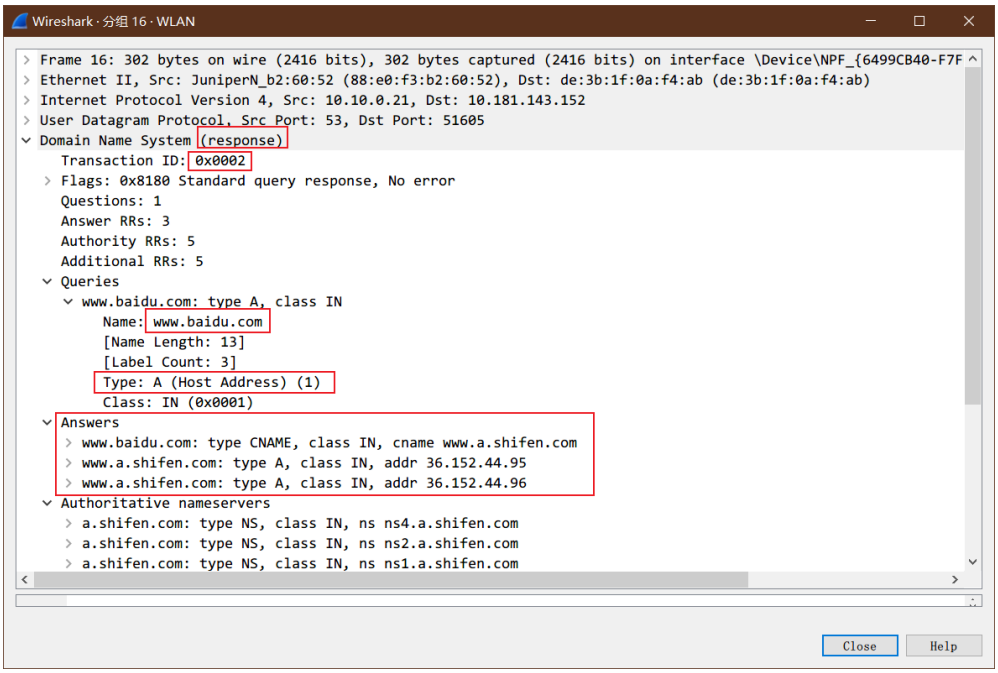
任务 1：使用 nslookup 命令，查询某个域名，并捕获这次的数据包。DNS 数据包由哪几层协议构成？ Frame, Ethernet II, IPv4, UDP 和 DNS **。使用的服务方端口是：** 53 **。**

分别选择一个请求包和一个响应包，展开最高层协议的详细内容，标出交易 ID、查询类型、查询的域名内容以及查询结果。

请求包：



响应包：



任务 2：使用 Ping 命令，分别测试某个 IP 地址和某个域名的连通性，并捕获数据包。捕获到了哪些相关协议数据包？

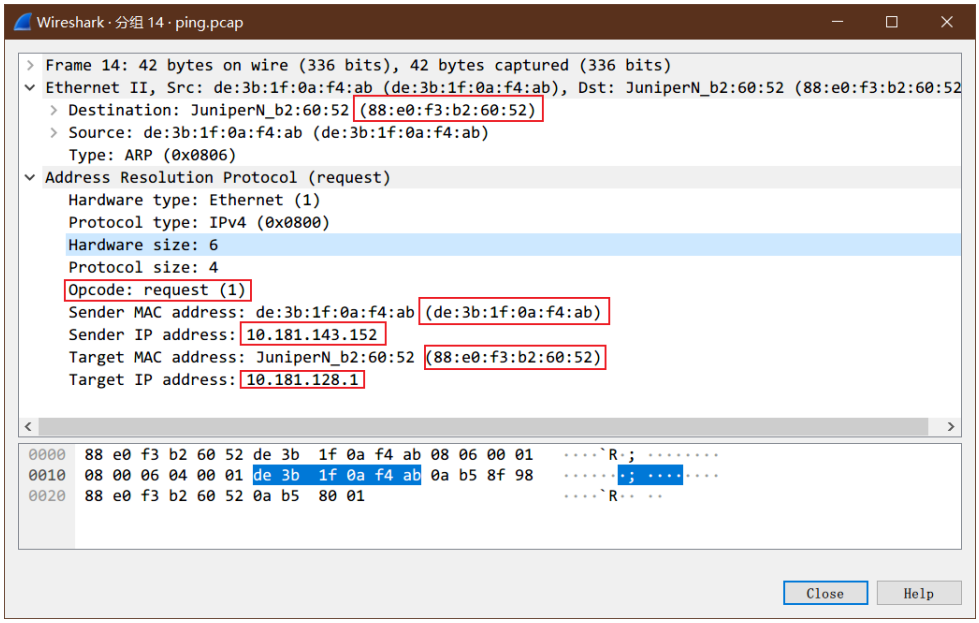
Ping IP 地址时： ICMP

Ping 域名时： ICMP, ARP

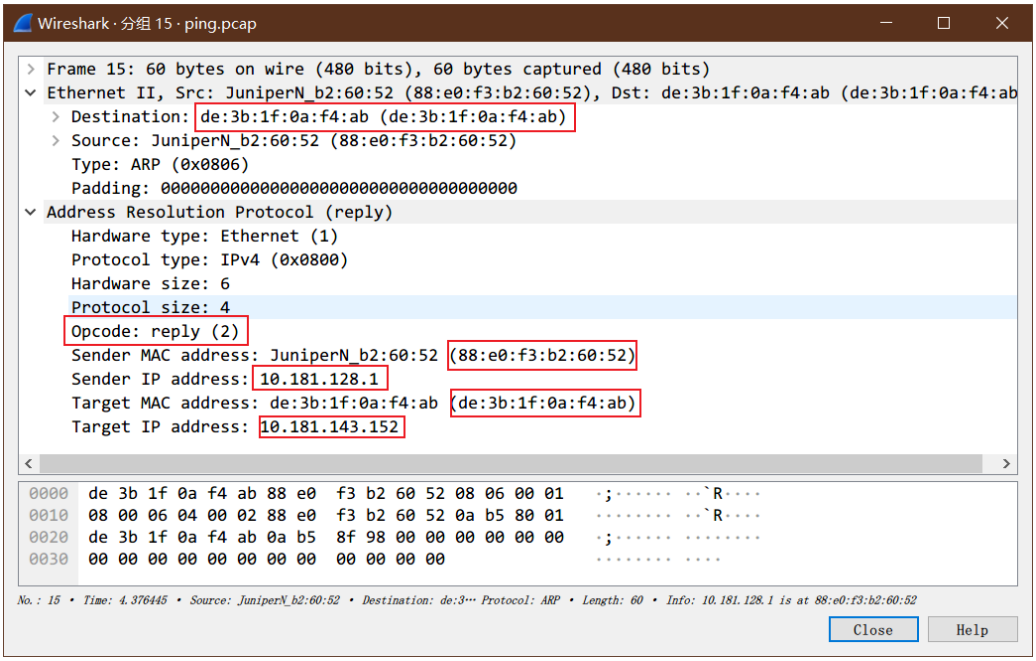
ICMP 数据包分别由哪几层协议构成？ Frame, Ethernet II, IPv4, ICMP

分别选择一个 ARP 请求和响应数据包，展开最高层协议的详细内容，标出操作码、发送者 IP 地址、发送者 MAC 地址、查询的目标 IP 地址、Ethernet 层的目标 MAC 地址以及查询结果。

请求包：

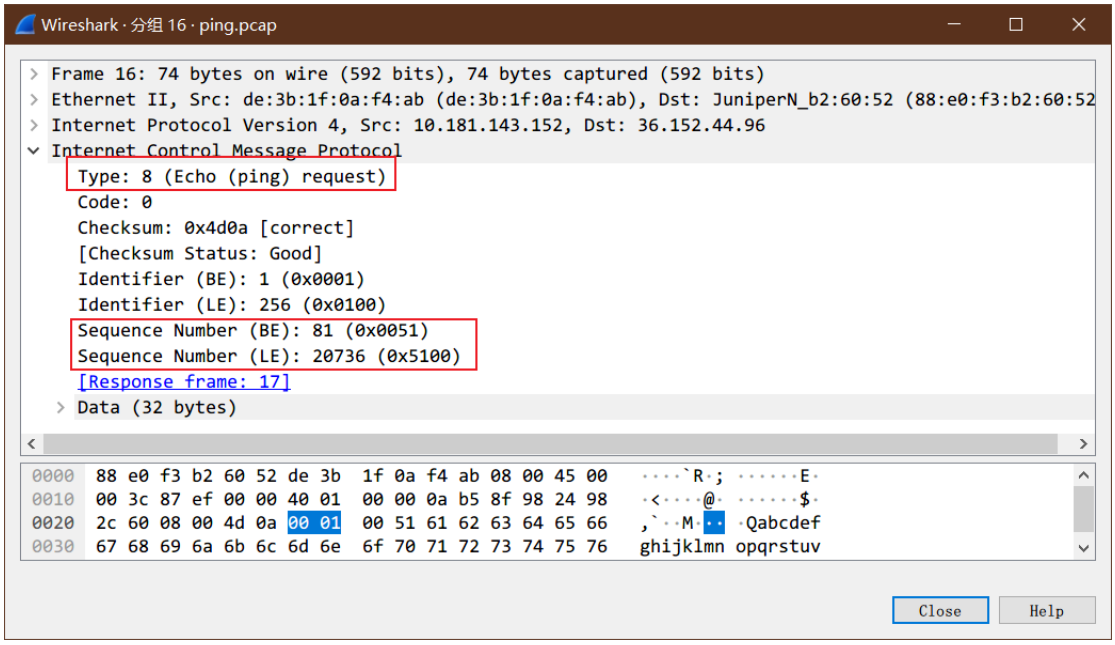


响应包：

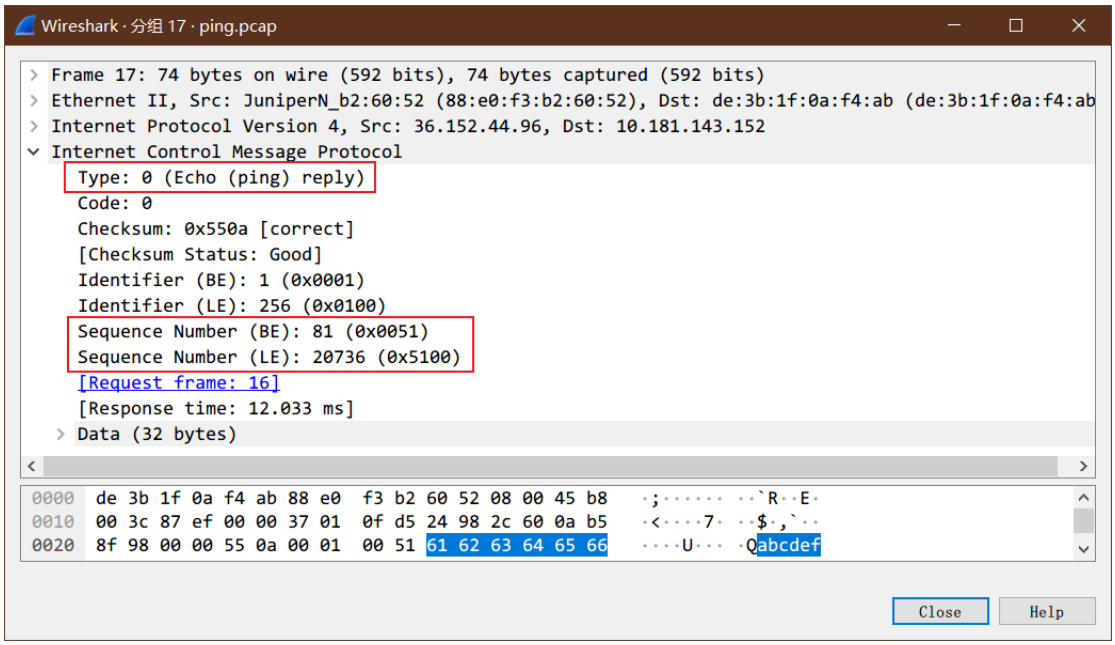


分别选择一个 ICMP 请求和响应数据包，展开最高层协议的详细内容，标出类型、序号。

请求包：

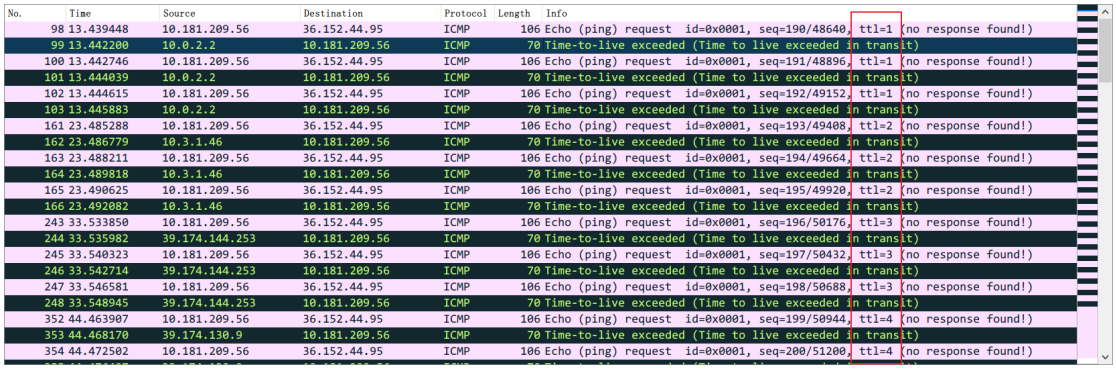


响应包：

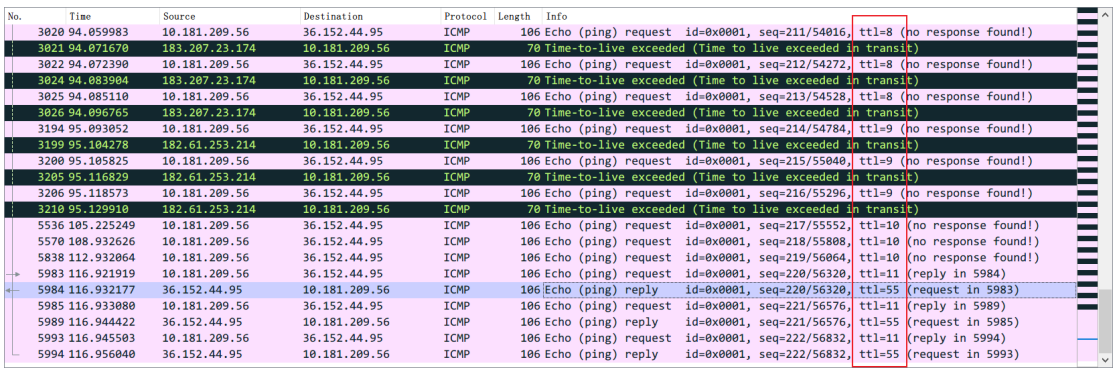


任务 3：使用 Tracert 命令（Mac 下使用 Traceroute 命令），跟踪某个外部 IP 地址的路由，并捕获这次的数据包。跟踪路由使用的数据包协议类型是：__ICMP__，数据包由几层协议构成？_Frame, Ethernet II, IPv4, ICMP_。

观察并记录请求包中 IP 协议层的 TTL 字段变化规律，第一个请求的 TTL 等于_1_，同样 TTL 的请求连续发送了__3__个，然后每次 TTL 增加了_1_，最后一个请求的 TTL 等于__11__。附上截图：



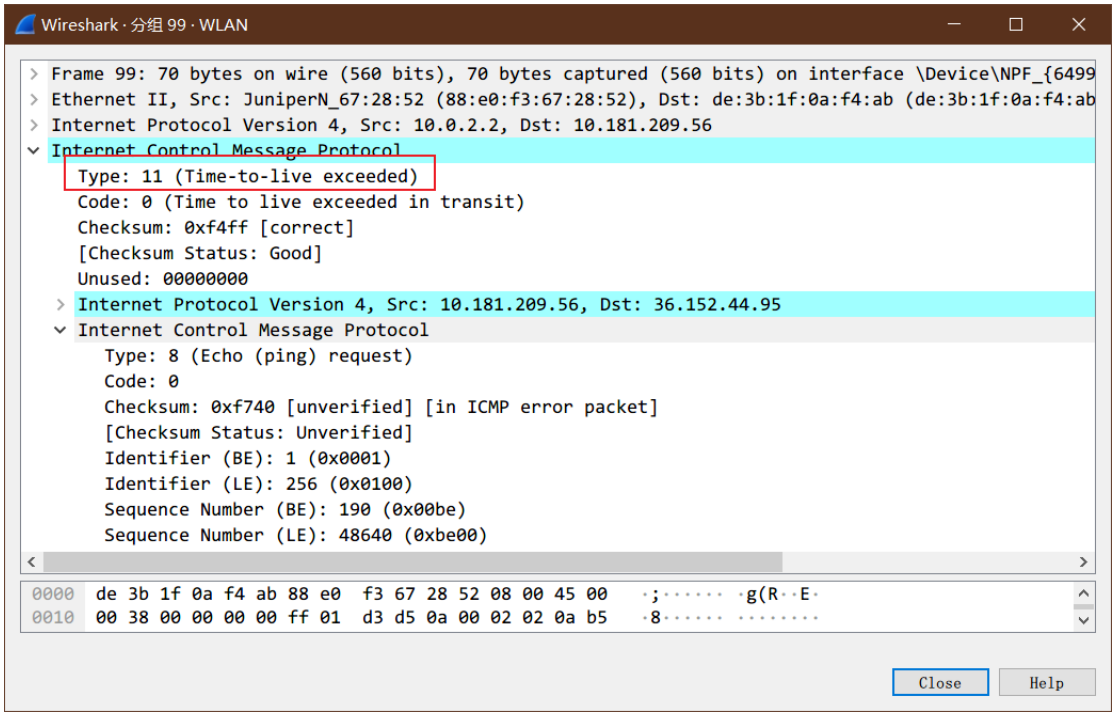
No.	Time	Source	Destination	Protocol	Length	Info
98	13.439448	10.181.209.56	36.152.44.95	ICMP	106	Echo (ping) request id=0x0001, seq=190/48640, ttl=1 (no response found!)
99	13.442200	10.0.2.2	10.181.209.56	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
100	13.442746	10.181.209.56	36.152.44.95	ICMP	106	Echo (ping) request id=0x0001, seq=191/48896, ttl=1 (no response found!)
101	13.444039	10.0.2.2	10.181.209.56	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
102	13.444615	10.181.209.56	36.152.44.95	ICMP	106	Echo (ping) request id=0x0001, seq=192/49152, ttl=1 (no response found!)
103	13.445883	10.0.2.2	10.181.209.56	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
161	23.485288	10.181.209.56	36.152.44.95	ICMP	106	Echo (ping) request id=0x0001, seq=193/49408, ttl=2 (no response found!)
162	23.486779	10.3.1.46	10.181.209.56	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
163	23.488211	10.181.209.56	36.152.44.95	ICMP	106	Echo (ping) request id=0x0001, seq=194/49664, ttl=2 (no response found!)
164	23.489818	10.3.1.46	10.181.209.56	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
165	23.490625	10.181.209.56	36.152.44.95	ICMP	106	Echo (ping) request id=0x0001, seq=195/49920, ttl=2 (no response found!)
166	23.492082	10.3.1.46	10.181.209.56	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
243	33.533850	10.181.209.56	36.152.44.95	ICMP	106	Echo (ping) request id=0x0001, seq=196/50176, ttl=3 (no response found!)
244	33.535982	39.174.144.253	10.181.209.56	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
245	33.540323	10.181.209.56	36.152.44.95	ICMP	106	Echo (ping) request id=0x0001, seq=197/50432, ttl=3 (no response found!)
246	33.542714	39.174.144.253	10.181.209.56	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
247	33.546581	10.181.209.56	36.152.44.95	ICMP	106	Echo (ping) request id=0x0001, seq=198/50688, ttl=3 (no response found!)
248	33.548945	39.174.144.253	10.181.209.56	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
352	44.463907	10.181.209.56	36.152.44.95	ICMP	106	Echo (ping) request id=0x0001, seq=199/50944, ttl=4 (no response found!)
353	44.468170	39.174.130.9	10.181.209.56	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
354	44.472502	10.181.209.56	36.152.44.95	ICMP	106	Echo (ping) request id=0x0001, seq=200/51200, ttl=4 (no response found!)



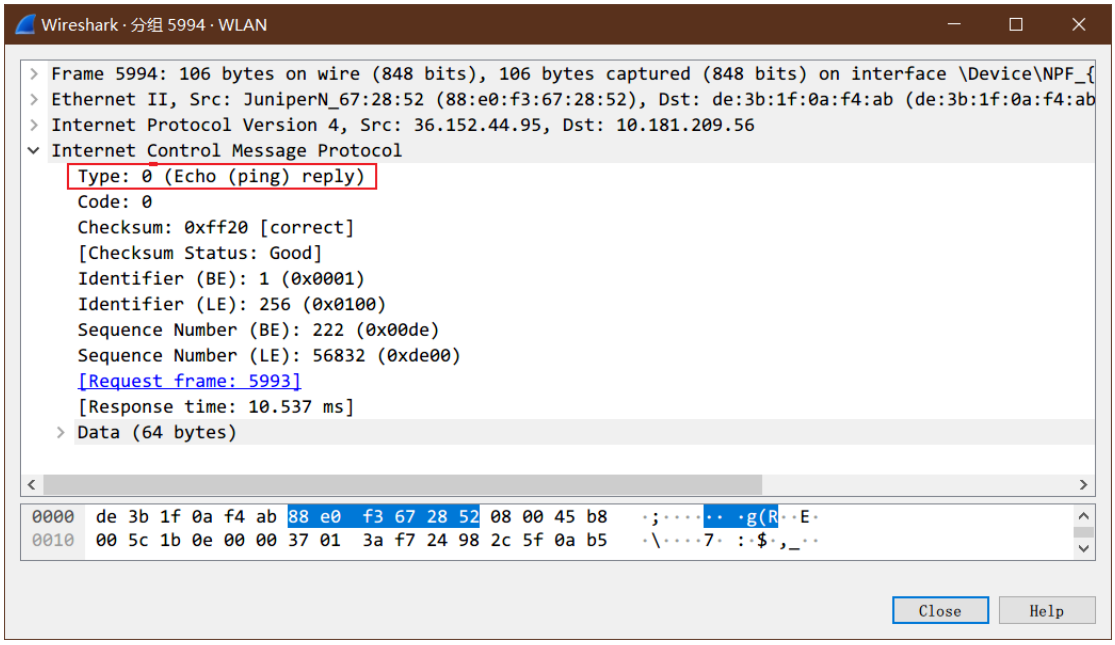
No.	Time	Source	Destination	Protocol	Length	Info
3020	94.059983	10.181.209.56	36.152.44.95	ICMP	106	Echo (ping) request id=0x0001, seq=211/54016, ttl=8 (no response found!)
3021	94.071670	183.207.23.174	10.181.209.56	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
3022	94.072390	10.181.209.56	36.152.44.95	ICMP	106	Echo (ping) request id=0x0001, seq=212/54272, ttl=8 (no response found!)
3024	94.083904	183.207.23.174	10.181.209.56	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
3025	94.085110	10.181.209.56	36.152.44.95	ICMP	106	Echo (ping) request id=0x0001, seq=213/54528, ttl=8 (no response found!)
3026	94.096765	183.207.23.174	10.181.209.56	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
3194	95.093052	10.181.209.56	36.152.44.95	ICMP	106	Echo (ping) request id=0x0001, seq=214/54784, ttl=9 (no response found!)
3199	95.104278	182.61.253.214	10.181.209.56	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
3200	95.105825	10.181.209.56	36.152.44.95	ICMP	106	Echo (ping) request id=0x0001, seq=215/55040, ttl=9 (no response found!)
3205	95.116829	182.61.253.214	10.181.209.56	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
3206	95.118573	10.181.209.56	36.152.44.95	ICMP	106	Echo (ping) request id=0x0001, seq=216/55296, ttl=9 (no response found!)
3210	95.129910	182.61.253.214	10.181.209.56	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
5536	105.225249	10.181.209.56	36.152.44.95	ICMP	106	Echo (ping) request id=0x0001, seq=217/55552, ttl=10 (no response found!)
5570	108.932626	10.181.209.56	36.152.44.95	ICMP	106	Echo (ping) request id=0x0001, seq=218/55808, ttl=10 (no response found!)
5838	112.932064	10.181.209.56	36.152.44.95	ICMP	106	Echo (ping) request id=0x0001, seq=219/56064, ttl=10 (no response found!)
5983	116.921919	10.181.209.56	36.152.44.95	ICMP	106	Echo (ping) request id=0x0001, seq=220/56320, ttl=11 (reply in 5984)
5984	116.932177	36.152.44.95	10.181.209.56	ICMP	106	Echo (ping) reply id=0x0001, seq=220/56320, ttl=55 (request in 5983)
5985	116.933080	10.181.209.56	36.152.44.95	ICMP	106	Echo (ping) request id=0x0001, seq=221/56576, ttl=11 (reply in 5989)
5989	116.944422	36.152.44.95	10.181.209.56	ICMP	106	Echo (ping) reply id=0x0001, seq=221/56576, ttl=55 (request in 5985)
5993	116.945503	10.181.209.56	36.152.44.95	ICMP	106	Echo (ping) request id=0x0001, seq=222/56832, ttl=11 (reply in 5994)
5994	116.956040	36.152.44.95	10.181.209.56	ICMP	106	Echo (ping) reply id=0x0001, seq=222/56832, ttl=55 (request in 5993)

观察并记录响应包的信息，第一组响应包的发送者 IP 是：__10.0.2.2__，标记 ICMP 层的类型字段。最后一组响应包的发送者 IP 是：__36.152.44.95__，标记 ICMP 层的类型字段。附上截图：

第一组：



最后一组：



✧ Part Three

1. 运行 **ipconfig /flushdns** 命令清空 DNS 缓存，然后打开浏览器，访问 **www.zju.edu.cn**，并使用捕获过滤器只捕获访问该网站的数据（过滤器设置：**tcp port 80 or udp port 53**），网页完全打开后，停止捕获。

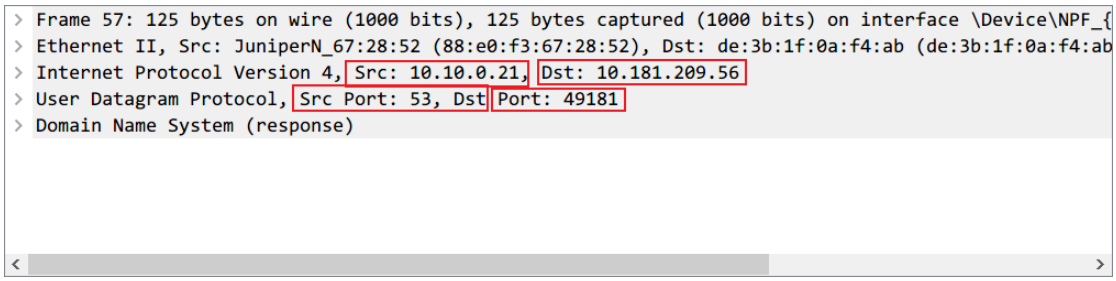
   捕获到的这些最高层的协议数据包分别由哪几层协议构成？

   DNS: Frame, Ethernet II, IPv4, UDP, DNS

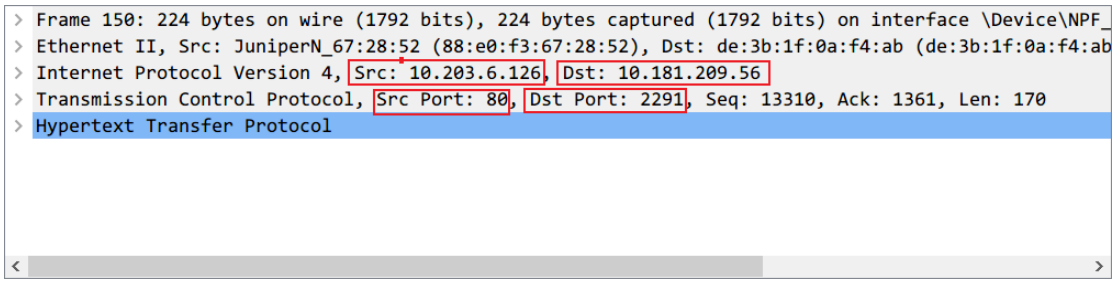
   HTTP: Frame, Ethernet II, IPv4, TCP, HTTP

每种协议选取一个代表展开后截图，并标出源和目标 **IP** 地址、源和目标端口）

DNS:



HTTP:



2. 为了打开网页，浏览器查询了哪些相关的域名？

   域名列表：www.zju.edu.cn

3. 使用显示过滤器 **tcp.stream eq X**，让 **X** 从 **0** 开始变化，直到没有数据。分析浏览器为了获取网页数据，总共建立了几个连接？（一个 **TCP** 流对应一个 **TCP** 连接）

   TCP 连接数：1

4. 右键点击某个 **HTTP** 数据包，选择跟踪 **TCP** 流，可以看到 **HTTP** 会话的数据。分析浏览器与 **WEB** 服务器之间进行了几次 **HTTP** 会话(一对 **HTTP** 请求和响应对应一次 **HTTP** 会话）？注意：一个 **TCP** 流上可能存在多个 **HTTP** 会话。

   HTTP 会话数：2

5. 选择一个 HTTP 的 TCP 流进行截图，标出请求和响应部分：

```
GET / HTTP/1.1
Host: www.zju.edu.cn
Connection: keep-alive
DNT: 1
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/87.0.4280.66 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/
avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9,en-GB;q=0.8,en;q=0.7
Cookie: _ga=GA1.3.1424851968.1583492921;
Hm_lvt_fe30bbc1ee45421ec1679d1b8d8f8453=1605756519,1605756693

HTTP/1.1 200 OK
Server: nginx
Date: Thu, 19 Nov 2020 04:05:37 GMT
Content-Type: text/html
Content-Length: 13041
Connection: keep-alive
Set-Cookie: route=42bbc6a24acf9f019080162ce2732518; Path=/
X-Frame-Options: SAMEORIGIN
Frame-Options: SAMEORIGIN
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Encoding: gzip
```

```
N?.:l.]IX=..Q.(I.1..y.k.n.Q
..;+ GmCY2.g..pZ.gbUYZ..!...".l.
.(.3]@c.jC...RG0.....GET /_visitcount?siteId=590&type=1&columnId=32642
HTTP/1.1
Host: www.zju.edu.cn
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/87.0.4280.66 Safari/537.36
DNT: 1
Accept: image/avif,image/webp,image/apng,image/*,*/*;q=0.8
Referer: http://www.zju.edu.cn/
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9,en-GB;q=0.8,en;q=0.7
Cookie: _ga=GA1.3.1424851968.1583492921;
Hm_lvt_fe30bbc1ee45421ec1679d1b8d8f8453=1605756519,1605756693;
route=42bbc6a24acf9f019080162ce2732518

HTTP/1.1 200 OK
Server: nginx
Date: Thu, 19 Nov 2020 04:05:38 GMT
Content-Length: 0
Connection: keep-alive
X-Frame-Options: SAMEORIGIN
Frame-Options: SAMEORIGIN
Set-Cookie: JSESSIONID=ED0768E9C0DA0C3082BC469332A3D909; Path=/
```

六、实验结果分析与思考

- 如果只想捕获某个特定 WEB 服务器 IP 地址相关的 HTTP 数据包，捕获过滤器应该怎么写？

  ```
  host <ip> && tcp port 80
  ```

- Ping 发送的是什么类型的协议数据包？什么情况下会出现 ARP 数据包？ Ping 一个域名和 Ping 一个 IP 地址出现的数据包有什么不同？

  ping 发送的是 ICMP 类型的协议数据包；

  如果目标主机 IP 与自己不在同一网段，则要发送 ping 请求给默认网关，如果目标网关没有对应的 MAC 地址，就会发送一个 ARP 广播包；

  ping 域名会先进行 DNS 域名解析，将域名转换为 IP；

- Tracert/Traceroute 发送的是什么类型的协议数据包，整个路由跟踪过程是如何进行的？

  tracert/traceroute 发送的是 ICMP 类型的协议数据包；

  路由跟踪过程：

  1) 先从源地址发出一个 ICMP 请求回显（ICMP Echo Request）数据包到目的地址，并将 TTL 设置为 1
  2) 到达路由器时，将 TTL 减 1
  3) 当 TTL 变为 0 时，包被丢弃，路由器向源地址发回一个 ICMP 超时通知（ICMP Time Exceeded Message），内含发送 IP 包的源地址，IP 包的所有内容及路由器的 IP 地址
  4) 当源地址收到该 ICMP 包时，显示这一跳路由信息
  5) 重复 1~4，并每次设置 TTL 加 1
  6) 直至目标地址收到探测数据包，并返回 ICMP 回应答复（ICMP Echo Reply）
  7) 当源地址收到 ICMP Echo Reply 包时，停止 tracert

- 如何理解 TCP 连接和 HTTP 会话？他们之间存在什么关系？

  TCP 是连接协议，是通讯方式，HTTP 是一种客户端与服务端之间的传输协议；

  HTTP 会话是建立在 TCP 连接的基础上的，一个 HTTP 会话要先建立一个 TCP 连接，然后在此基础上进行 HTTP 的请求与应答；

- DNS 为什么选择使用 UDP 协议进行传输？而 HTTP 为什么选择使用 TCP 协议？

  使用 UPD 只需要一次请求一次应答，而使用 TCP 要经过三次握手、发送数据及应答，四次挥手，而且 DNS 一般数据量不大，使用 UDP 可以节省网络资源；

  HTTP 对安全性和可靠性要求较高，而且一般数据量比较大，有上下文，因此使用 TCP；

七、讨论、心得

遇到的困难：

在清空 DNS 缓存然后设置捕获过滤器只捕获有关 www.zju.edu.cn 的数据包并用浏览器访问该域名时，并没有抓到 DNS 相关的包。

解决方案：清除捕获过滤器重新操作，找到 DNS 相关的数据包后用显示过滤器进行过滤完成了该部分实验

本次实验内容比较基础，也比较简单，主要考察对 Wireshark 的基本使用，Wireshark 是一个功能强大的抓包工具，可以对数据包进行细致的分析，熟练掌握 Wireshark 的使用方法，对计算机网络的实践学习有着莫大的帮助。

在实验中也遇到了不少问题，大部分都在查找相关资料后得到了有效的解决，经过这次实验，对理论课上的知识也有了更深的理解，获益匪浅。