

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:	李云帆
学 院:	计算机学院
系:	计算机科学与技术
专 业:	计算机科学与技术
学 号:	3200102555
指导教师:	邱劲松

2022 年 10 月 7 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

提醒：为了避免捕获到大量无关数据包，影响实验观察，建议关闭所有无关软件。实验之前可以提前了解下第六部分有哪些问题。

五、实验数据记录和处理

以下实验记录均需结合屏幕截图，进行文字标注和描述，图片应大小合适、关键部分清晰可见，可直接在图片上进行标注，也可以单独用文本进行描述。

✧ 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, HTTP, ICMP, TLSv1.2, DHCP, SSDP

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： Network layer protocol: IPv4, Transport layer protocol: 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。

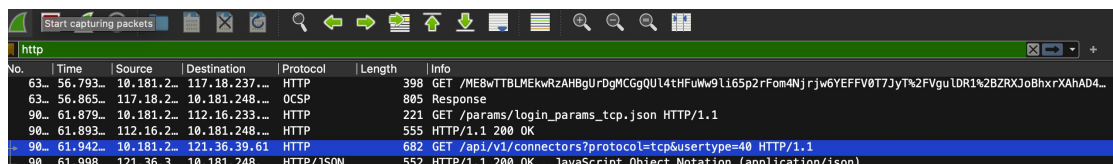
```
> Frame 9: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on int
✓ Ethernet II, Src: JuniperN_67:28:52 (88:e0:f3:67:28:52), Dst: Apple_db:fe
  > Destination: Apple_db:fe:01 (1c:91:80:db:fe:01)
  > Source: JuniperN_67:28:52 (88:e0:f3:67:28:52)
    Type: IPv4 (0x0800)
  ✓ Internet Protocol Version 4, Src: 103.74.50.106, Dst: 10.181.248.236
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ✓ Differentiated Services Field: 0x04 (DSCP: LE, ECN: Not-ECT)
    0000 01.. = Differentiated Services Codepoint: Lower Effort (1)
    .... ..00 = Explicit Congestion Notification: Not ECN-Capable Transp
    Total Length: 52
    Identification: 0x0a74 (2676)
  ✓ 010. .... = Flags: 0x2, Don't fragment
    0... .... = Reserved bit: Not set
    .1.. .... = Don't fragment: Set
    ..0. .... = More fragments: Not set
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 51
    Protocol: TCP (6)
    Header Checksum: 0x9ff6 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 103.74.50.106
    Destination Address: 10.181.248.236
  > Transmission Control Protocol, Src Port: 80, Dst Port: 64964, Seq: 1, Ack
    .....
```

0000	1c 91 80 db fe 01 88 e0 f3 67 28 52 08 00 45 04g(R..E.
0010	00 34 0a 74 40 00 33 06 9f f6 67 4a 32 6a 0a b5	4.t@.3. .gJ2j..
0020	f8 ec 00 50 fd c4 83 60 66 1a 46 76 aa ef 80 10	..P...`f.Fv....
0030	00 3b 69 26 00 00 01 01 08 0a 17 bb 60 3a ce a8	;i&....`.:..
0040	50 72	Pr

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： http ， 希望显示的协议类型： HTTP 。

截图：

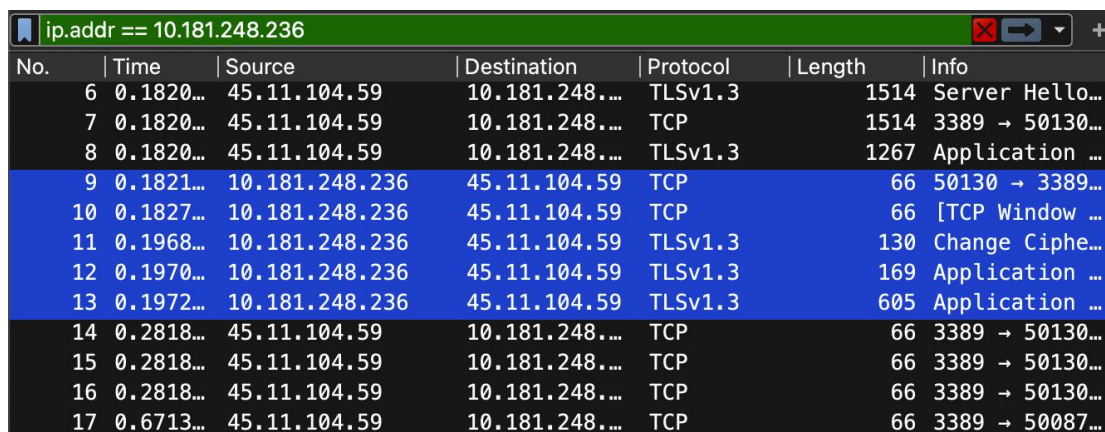


4. 配置应用显示过滤器，让界面只显示某个 IP 地址的数据包（ip.addr==x.x.x.x）。

使用的过滤器： ip.addr == 10.181.248.236 ， 希望显示的 IP 地址：

10.181.248.236 。

截图：

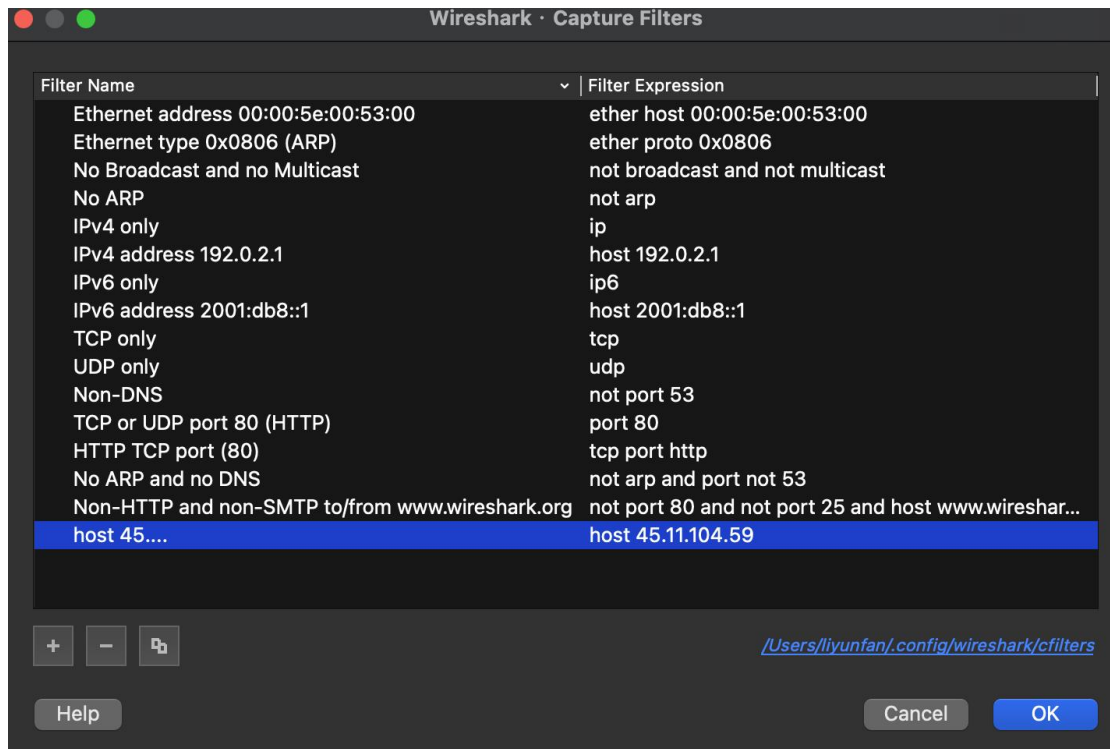


No.	Time	Source	Destination	Protocol	Length	Info
6	0.1820...	45.11.104.59	10.181.248....	TLSv1.3	1514	Server Hello...
7	0.1820...	45.11.104.59	10.181.248....	TCP	1514	3389 → 50130...
8	0.1820...	45.11.104.59	10.181.248....	TLSv1.3	1267	Application ...
9	0.1821...	10.181.248.236	45.11.104.59	TCP	66	50130 → 3389...
10	0.1827...	10.181.248.236	45.11.104.59	TCP	66	[TCP Window ...
11	0.1968...	10.181.248.236	45.11.104.59	TLSv1.3	130	Change Ciphe...
12	0.1970...	10.181.248.236	45.11.104.59	TLSv1.3	169	Application ...
13	0.1972...	10.181.248.236	45.11.104.59	TLSv1.3	605	Application ...
14	0.2818...	45.11.104.59	10.181.248....	TCP	66	3389 → 50130...
15	0.2818...	45.11.104.59	10.181.248....	TCP	66	3389 → 50130...
16	0.2818...	45.11.104.59	10.181.248....	TCP	66	3389 → 50130...
17	0.6713...	45.11.104.59	10.181.248....	TCP	66	3389 → 50087...

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 45.11.104.59 ， 希望捕获的 IP 地址：
45.11.104.59 。

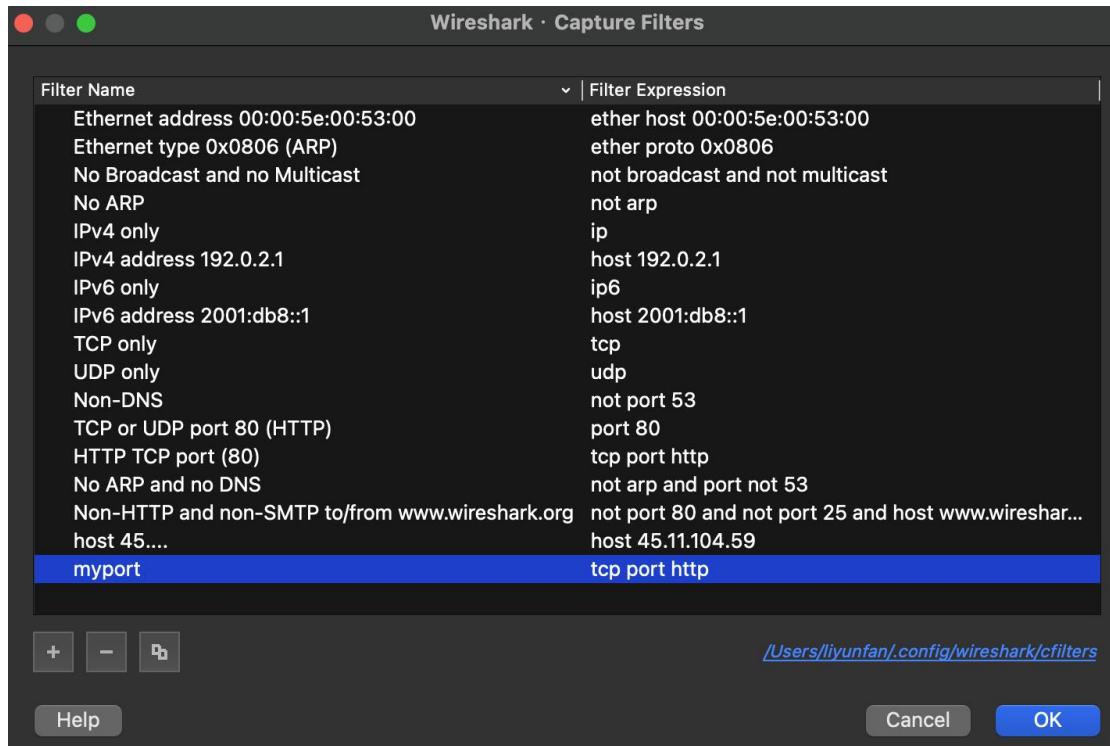
截图：



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 http ， 希望捕获的协议类型： http。

截图：



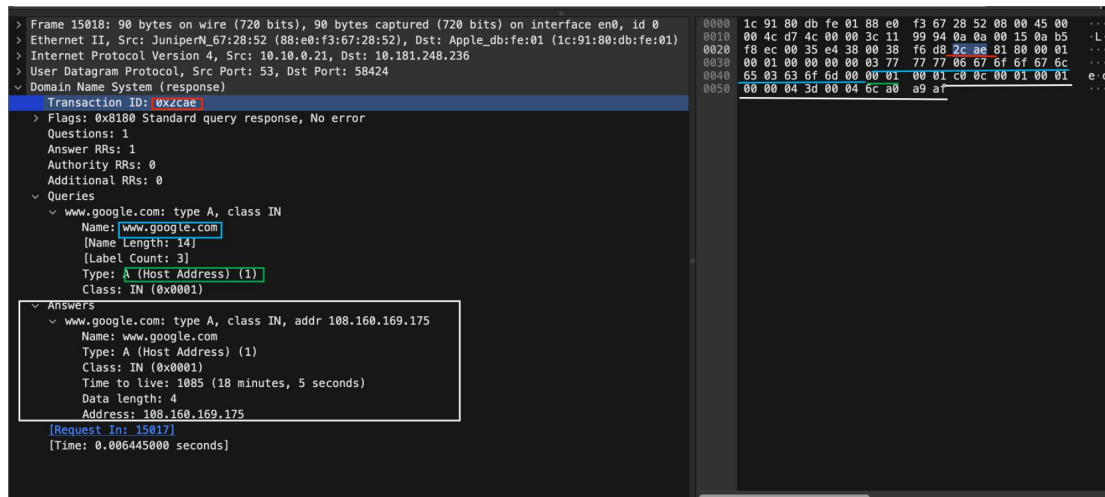
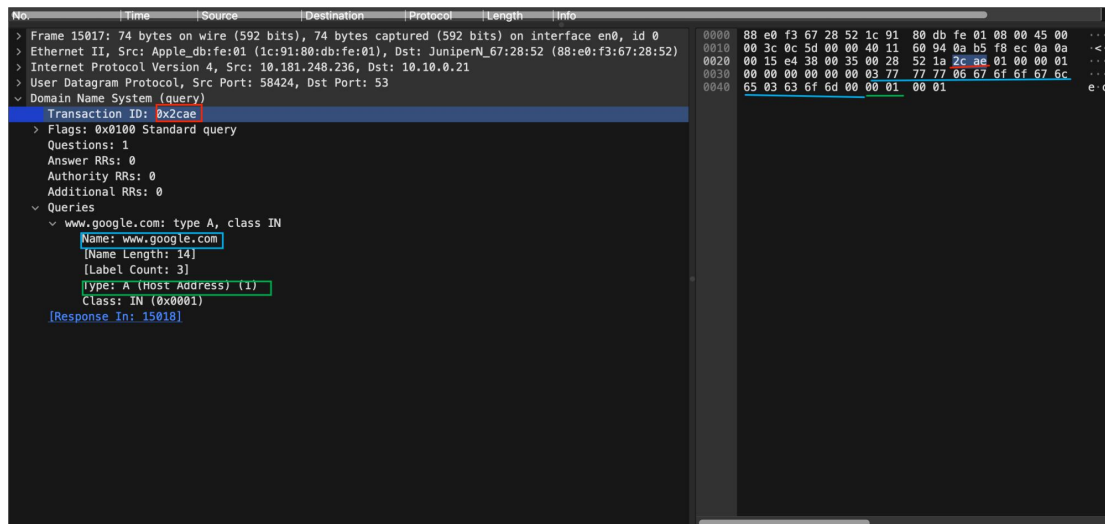
请在下面的每次捕获任务完成后，保存 Wireshark 抓包记录（.pcap 格式），随报告一起提交。每一个任务一个单独文件（如 dns.pcap、ping.pcap、tracert.pcap）。

✧ 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？ Data link layer:Ethernet II, Networklayer:IPv4, Transport layer: UDP, Application layer: DNS。使用的服务方端口是：58424。

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。

截图参考（此处应替换成实际截获的数据，请求和响应各一个）：



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

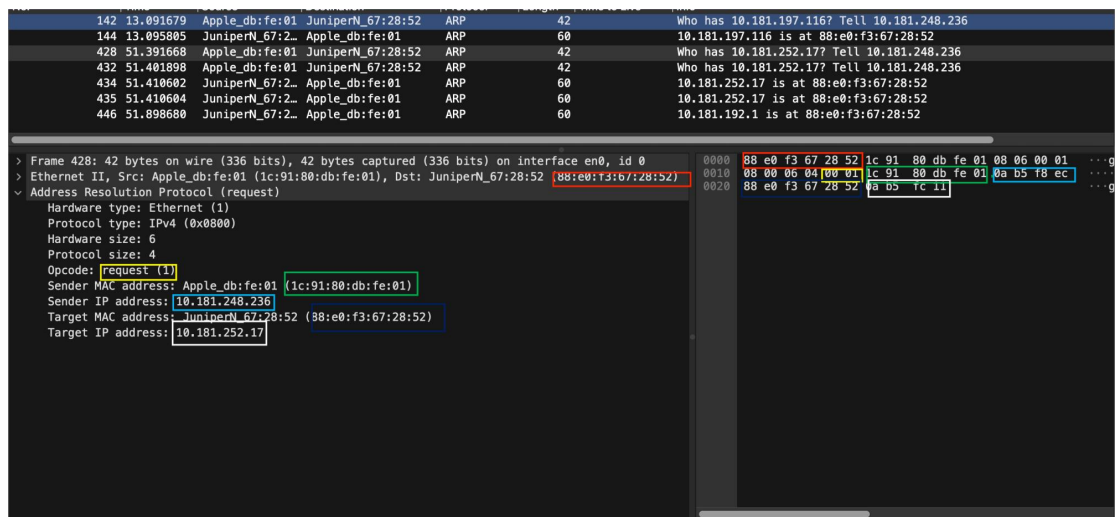
Ping 域名时: _____ DNS, ICMP

ICMP 数据包分别由哪几层协议构成? _____ Data link layer:Ethernet II,
Networklayer:IPv4,ICMP

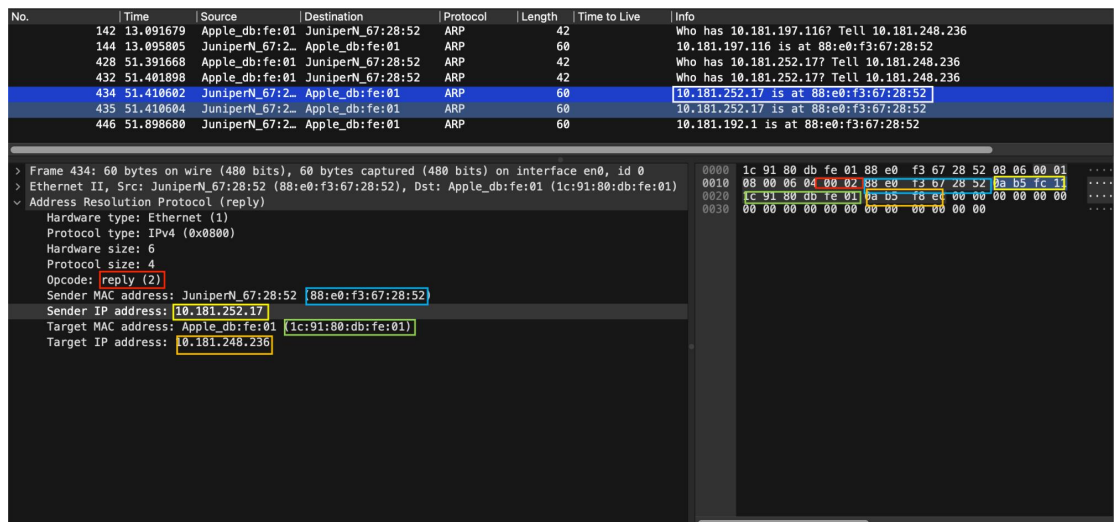
分别选择一个 ARP 请求和响应数据包, 展开最高层协议的详细内容, 标出操作码、发送者 IP 地址、发送者 MAC 地址、查询的目标 IP 地址、Ethernet 层的目标 MAC 地址以及查询结果。

截图参考 (此处应替换成实际截获的数据, 请求和响应各一):

请求:

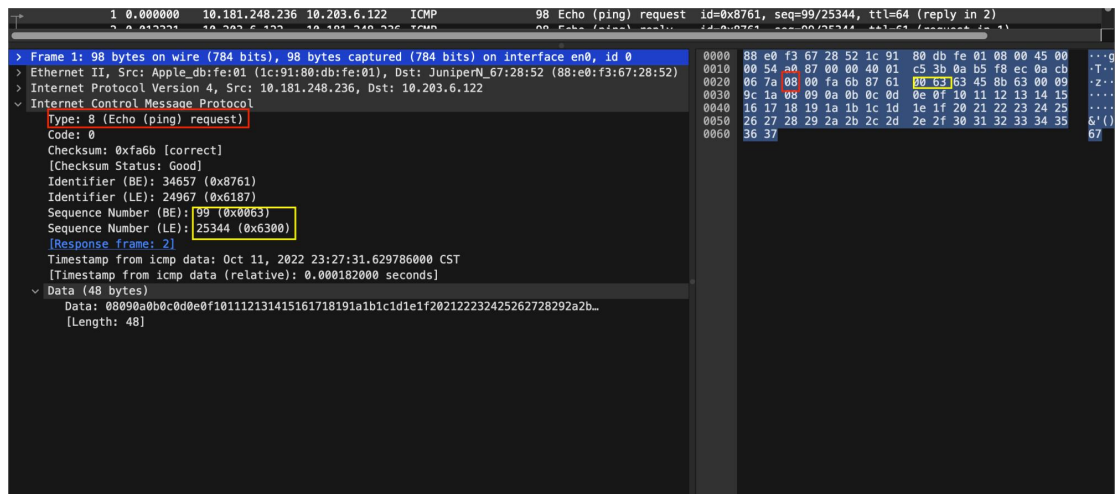


响应:

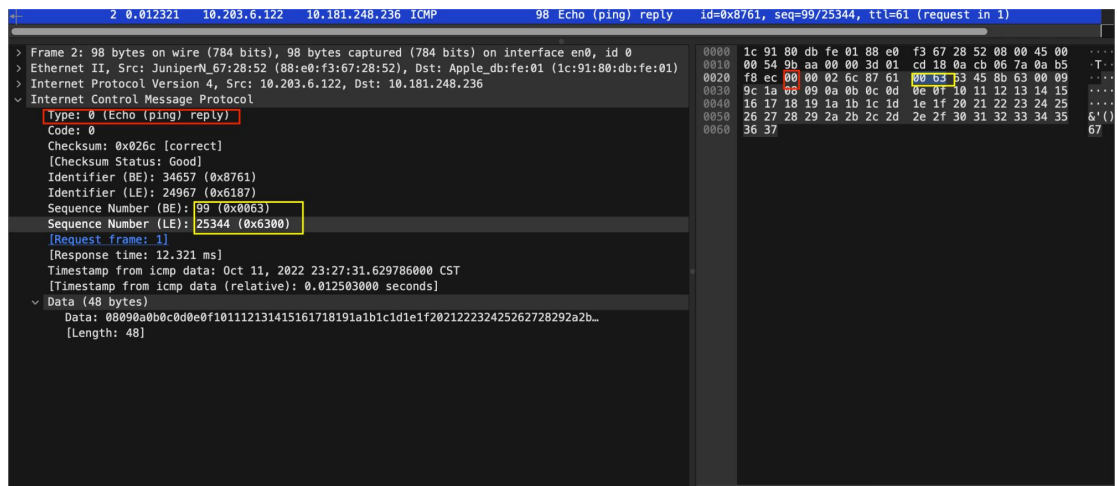


分别选择一个 ICMP 请求和响应数据包，展开最高层协议的详细内容，标出类型、序号。

请求:



响应:



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? 4。

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 4。附上截图:

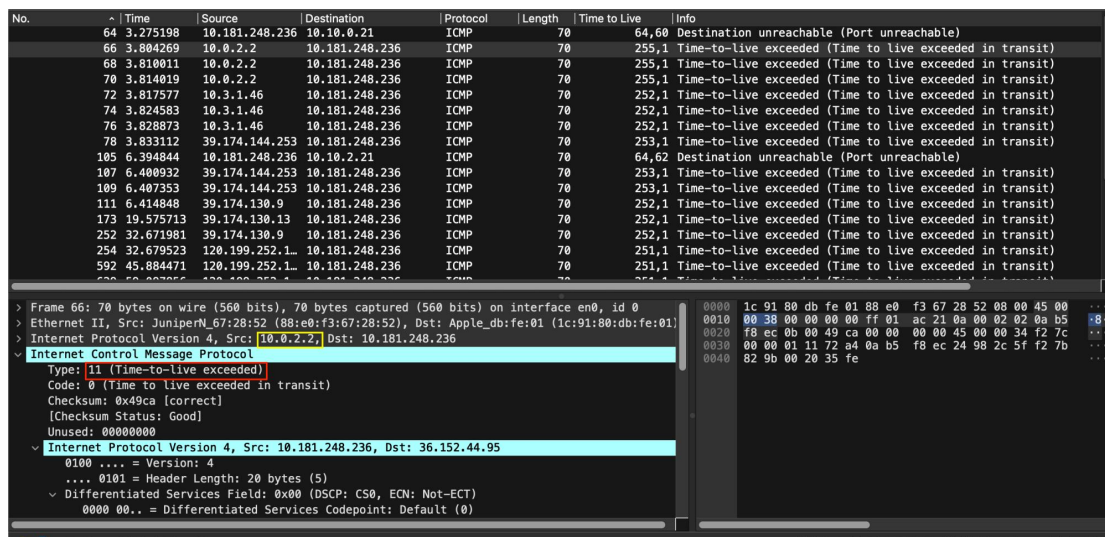
截图参考 (此处应替换成实际截获的数据):

No.	Time	Source	Destination	Protocol	Length	Time to Live	Info
65	3.790301	10.181.248.236	36.152.44.95	UDP	66	1	62075 → 33435 Len=24
67	3.805474	10.181.248.236	36.152.44.95	UDP	66	1	62075 → 33436 Len=24
69	3.810239	10.181.248.236	36.152.44.95	UDP	66	1	62075 → 33437 Len=24
71	3.814225	10.181.248.236	36.152.44.95	UDP	66	2	62075 → 33438 Len=24
73	3.818642	10.181.248.236	36.152.44.95	UDP	66	2	62075 → 33439 Len=24
75	3.824814	10.181.248.236	36.152.44.95	UDP	66	2	62075 → 33440 Len=24
77	3.828978	10.181.248.236	36.152.44.95	UDP	66	3	62075 → 33441 Len=24
106	6.396086	10.181.248.236	36.152.44.95	UDP	66	3	62075 → 33442 Len=24
108	6.401240	10.181.248.236	36.152.44.95	UDP	66	3	62075 → 33443 Len=24
110	6.407608	10.181.248.236	36.152.44.95	UDP	66	4	62075 → 33444 Len=24
172	19.569595	10.181.248.236	36.152.44.95	UDP	66	4	62075 → 33445 Len=24

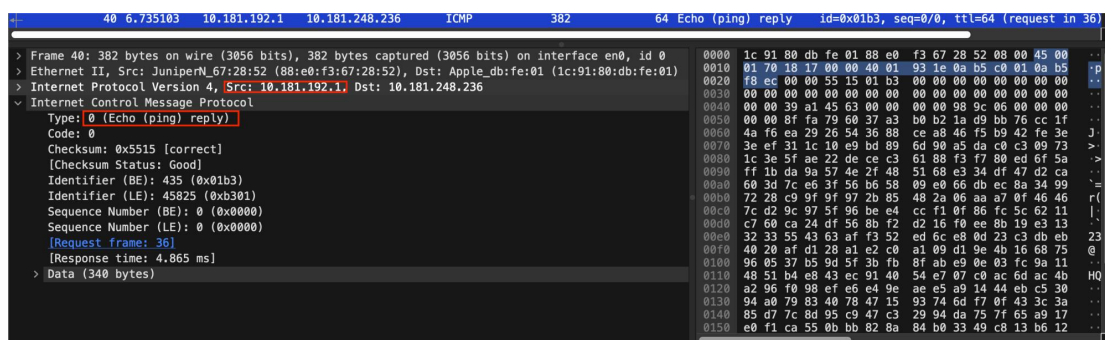
观察并记录响应包的信息，第一组响应包的发送者 IP 是： 10.0.2.2，
标记 ICMP 层的类型字段。最后一组响应包的发送者 IP 是： 10.181.192.1，
标记 ICMP 层的类型字段。附上截图：

截图参考（此处应替换成实际截获的数据）：

第一组：



最后一组：



请在下面的捕获任务完成后，保存 Wireshark 抓包记录（.pcap 格式），随报告一起提交。文件名 http.pcap。

✧ 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。

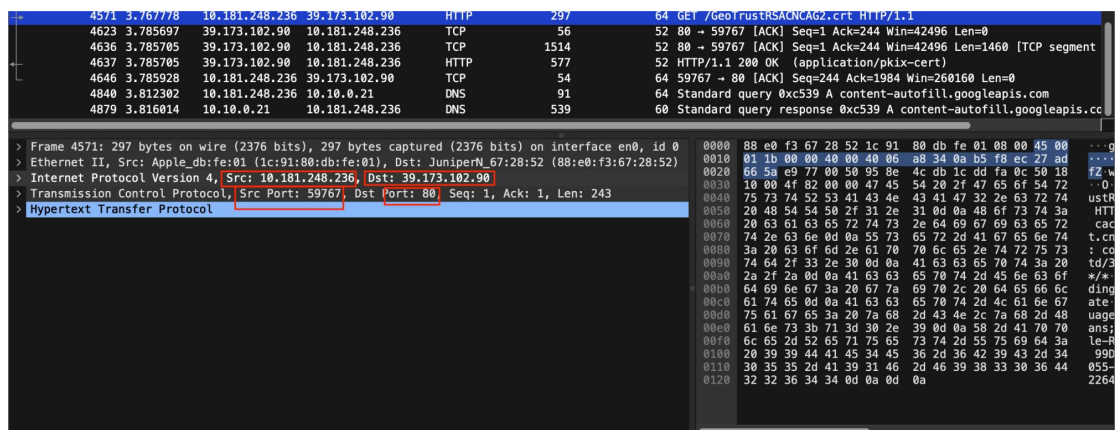
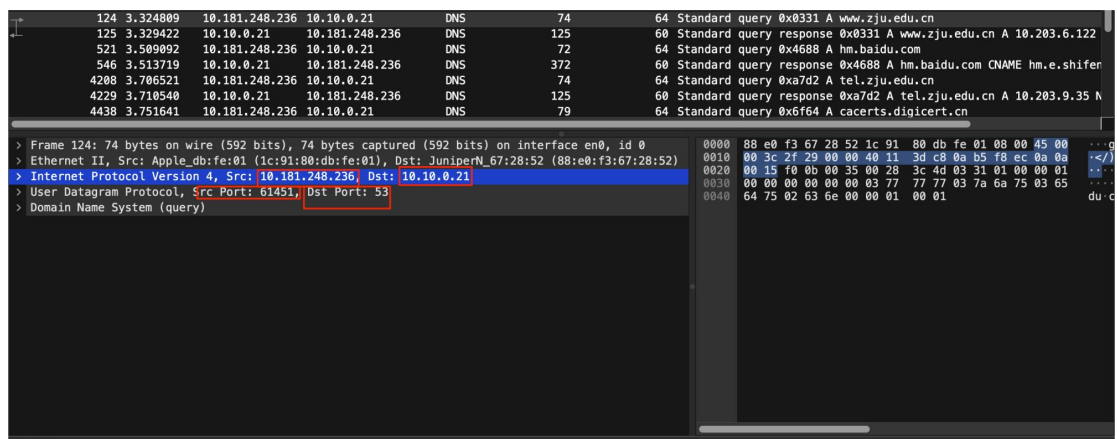
捕获到的这些最高层的协议数据包分别由哪几层协议构成？

DNS : Data link layer:Ethernet II, Networklayer:IPv4,Transport layer:UDP, Application layer: DNS

HTTP: Data link layer:Ethernet II, Networklayer:IPv4,Transport layer:TCP, Application layer: 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)

截图参考 (此处应替换成实际截获的数据) :



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.google.com 2c46232.pd.gladns.com www.zju.edu.cn hm.baidu.com tel.zju.edu.cn cacerts.digicert.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 (最好有多个 HTTP 会话的):

截图示例 (此处应替换成实际截获的数据):

会话 1:

```
GET /gts1c3/ME8wTTBLMEKwRzAHBqUrDgMCGgQUxy55it3%2FYTSzUu1HQr17xsAkB2MEFIp0f6%2BFze6VzT2c00JGFpNxNR0nAhAVz29XD0oD7QoAE6RT86pI HTTP/1.1
Host: ocsdpki.google
User-Agent: com.apple.trustd/3.0
Accept: */*
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh-Hans;q=0.9
X-Apple-Request-Uid: E2999AF8-5F71-466A-B8B9-70A79FBEC23C

HTTP/1.1 200 OK
Server: ocsdp_responder
Content-Length: 471
X-XSS-Protection: 0
X-Frame-Options: SAMEORIGIN
Date: Tue, 11 Oct 2022 18:10:29 GMT
Cache-Control: public, max-age=14400
Content-Type: application/ocsp-response
Age: 1183

0...
.....0.....+.....0.....0.....t.....=...F..q5..'..20221010134026Zs0q0I0 ..+.....y...a4...GB...$.c...t.....=...F..q5...'..oW..J..
$.c...t.....=...F..q5..'..K..L&...T...}.e.....20221010134421Z...20221017124420Z0
*..H...
.....Bb3...B...
.....!m.....k..t.....)0m.....K..n/ ..6:W.a?.....
}.:..7..=1.gs.GE].....u:..L^Q.....W^./...{.'~.....C"\\.....Q..
.....
A;.ZZ.....3;...78<.Z..B...U.S....."G.K...{v...#...D/*..N...L..g.; X..3s.j...;F..
```

会话 2:

```
GET /gts1c3/MFAwTjBMMEowSDAHBqUrDgMCGgQUxy55it3%2FYTSzUu1HQr17xsAkB2MEFIp0f6%2BFze6VzT2c00JGFpNxNR0nAhEA%2Bmu6kEwmHfISVPZ9zN5L2w%3D%3D HTTP/1.1
Host: ocsdpki.google
User-Agent: com.apple.trustd/3.0
Accept: */*
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh-Hans;q=0.9
X-Apple-Request-Uid: AF3697B0-28C3-4606-914D-77F12278A022

HTTP/1.1 200 OK
Server: ocsdp_responder
Content-Length: 472
X-XSS-Protection: 0
X-Frame-Options: SAMEORIGIN
Date: Tue, 11 Oct 2022 18:05:36 GMT
Cache-Control: public, max-age=14400
Content-Type: application/ocsp-response
Age: 1476

0...
.....0.....+.....0.....0.....t.....=...F..q5..'..20221010134421Z0t0r0J0 ..+.....y...a4...GB...
$.c...t.....=...F..q5..'..K..L&...T...}.e.....20221010134421Z...20221017124420Z0
*..H...
.....Bb3...B...
.....!m.....k..t.....)0m.....K..n/ ..6:W.a?.....
}.:..7..=1.gs.GE].....u:..L^Q.....W^./...{.'~.....C"\\.....Q..
.....
A;.ZZ.....3;...78<.Z..B...U.S....."G.K...{v...#...D/*..N...L..g.; X..3s.j...;F..
```

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 192.168.1.1 //抓取 192.168.1.1 收到和发出的所有数据包

src host 192.168.1.1 //源地址， 192.168.1.1 发出的所有数据包

dst host 192.168.1.1 //目标地址， 192.168.1.1 收到的所有数据包

- 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？

ARP 数据包

同一网段内

ping 域名需要进行一次 DNS 查询，会有 DNS 数据包

- 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？

IP 数据包

首先它发送一份 TTL 字段为 1 的 IP 数据包给目的主机，处理这个数据包的第一个路由器将 TTL 值减 1，然后丢弃该数据报，并给源主机发送一个 ICMP 报文（“超时”信息，这个报文包含了路由器的 IP 地址，这样就得到了第一个路由器的地址），然后 traceroute 发送一个 TTL 为 2 的数据报来得到第二个路由器的 IP 地址，继续这个过程，直至这个数据报到达目的主机。

- 如何理解 TCP 连接和 HTTP 会话？他们之间存在什么关系？

在像 HTTP 这样的 Client-Server（客户端 - 服务器）协议中，会话分为三个阶段：

客户端建立一条 TCP 连接（如果传输层不是 TCP，也可以是其他适合的连接）。

客户端发送请求并等待应答。

服务器处理请求并送回应答，回应包括一个状态码和对应的数据。

TCP 用来给 HTTP 会话

- 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？

当请求体和响应的大小比较小时，通过 TCP 协议进行传输不仅需要传输更多的数据，还会消耗更多的资源，多次通信以及信息传输带来的时间成本在 DNS 查询较小时是无法被忽视的，TCP 连接带来的可靠性在 DNS 的场景中没能发挥太大的作用。

从理论上来说，一个 UDP 数据包的大小最多可以达到 64KB，这对于一个常见的 DNS 查询其实是一个非常大的数值；但是在实际生产中，一旦数据包中的数据超过了传送链路的最大传输单元 (MTU, 也就是单个数据包大小的上限，一般为 1500 字节)，当前数据包就可能会被分片传输、丢弃，部分的网络设备甚至会直接拒绝处理包含 EDNS(0) 选项的请求，这就会导致使用 UDP 协议的 DNS 不稳定。

TCP 作为可靠的传输协议，可以非常好的解决这个问题，通过序列号、重传等机制能够保证消息的不重不漏，消息接受方的 TCP 栈会对分片的数据重新进行拼装，DNS 等应用层协议可以直接使用处理好的完整数据。同时，当数据包足够大的时候，TCP 三次握手带来的额外开销比例就会越来越小

七、 讨论、心得

在完成本实验后，你可能会有很多待解答的问题，你可以把它们记在这里，接下来的学习中，你也许会逐渐得到答案的，同时也可以让老师了解到你有哪些困惑，老师在课堂可以安排针对性地解惑。等到课程结束后，你再回头看看这些问题时你或许会有不同的见解：

我在使用 Traceroute 命令时出现了 ttl 一直在 64 和 62 之间反复变化的情况，同时只有在抓包到 UDP 时才符合 1->2->3->...的变化趋势，但是在查找资料过后并不知道是什么原因

在实验过程中你可能会遇到的困难，并得到了宝贵的经验教训，请把它们记录下来，提供给其他人参考吧：

你对本实验安排有哪些更好的建议呢？欢迎献计献策：

希望能给出使用命令的一些教程和模板