浙江大學

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：解雲暄

学　　院：计算机学院

专　　业：信息安全

学　　号：3190105871

指导教师：郑扣根

2021 年 10 月 29 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

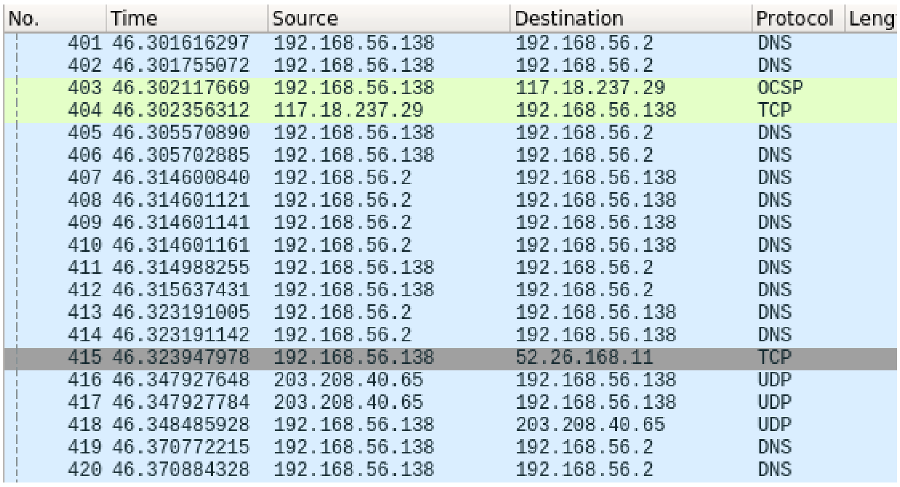
四、 操作方法与实验步骤

- 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

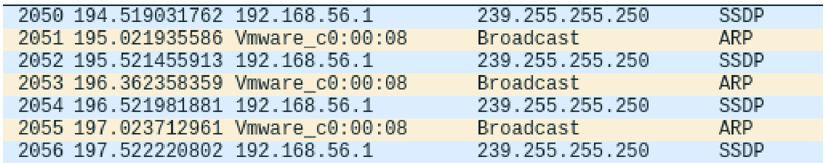
五、 实验数据记录和处理

✧ 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）。



```
No.      Time              Source           Destination      Protocol Leng
  401 46.301616297  192.168.56.138   192.168.56.2      DNS
  402 46.301755072  192.168.56.138   192.168.56.2      DNS
  403 46.302117669  192.168.56.138   117.18.237.29     OCSP
  404 46.302356312  117.18.237.29    192.168.56.138    TCP
  405 46.305570890  192.168.56.138   192.168.56.2      DNS
  406 46.305702885  192.168.56.138   192.168.56.2      DNS
  407 46.314600840  192.168.56.2     192.168.56.138    DNS
  408 46.314601121  192.168.56.2     192.168.56.138    DNS
  409 46.314601141  192.168.56.2     192.168.56.138    DNS
  410 46.314601161  192.168.56.2     192.168.56.138    DNS
  411 46.314988255  192.168.56.138   192.168.56.2      DNS
  412 46.315637431  192.168.56.138   192.168.56.2      DNS
  413 46.323191005  192.168.56.2     192.168.56.138    DNS
  414 46.323191142  192.168.56.2     192.168.56.138    DNS
  415 46.323947978  192.168.56.138   52.26.168.11      TCP
  416 46.347927648  203.208.40.65    192.168.56.138    UDP
  417 46.347927784  203.208.40.65    192.168.56.138    UDP
  418 46.348485928  192.168.56.138   203.208.40.65     UDP
  419 46.370772215  192.168.56.138   192.168.56.2      DNS
  420 46.370884328  192.168.56.138   192.168.56.2      DNS
```

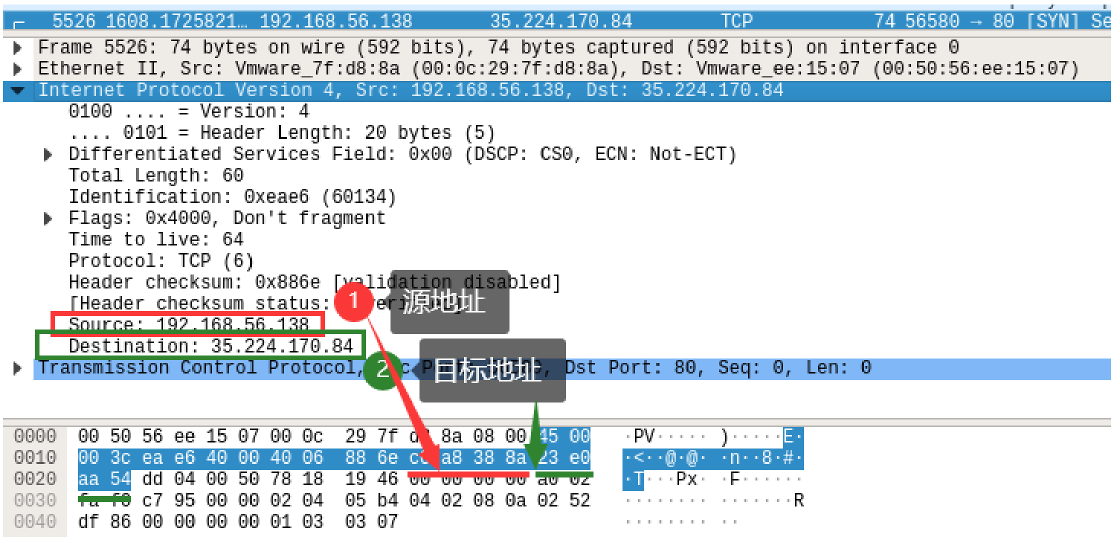


```
2050 194.519031762 192.168.56.1      239.255.255.250   SSDP
2051 195.021935586 Vmware_c0:00:08   Broadcast         ARP
2052 195.521455913 192.168.56.1      239.255.255.250   SSDP
2053 196.362358359 Vmware_c0:00:08   Broadcast         ARP
2054 196.521981881 192.168.56.1      239.255.255.250   SSDP
2055 197.023712961 Vmware_c0:00:08   Broadcast         ARP
2056 197.522220802 192.168.56.1      239.255.255.250   SSDP
```

协议名：DNS，OCSP，TCP，UDP，ARP，SSDP，HTTP 等等。

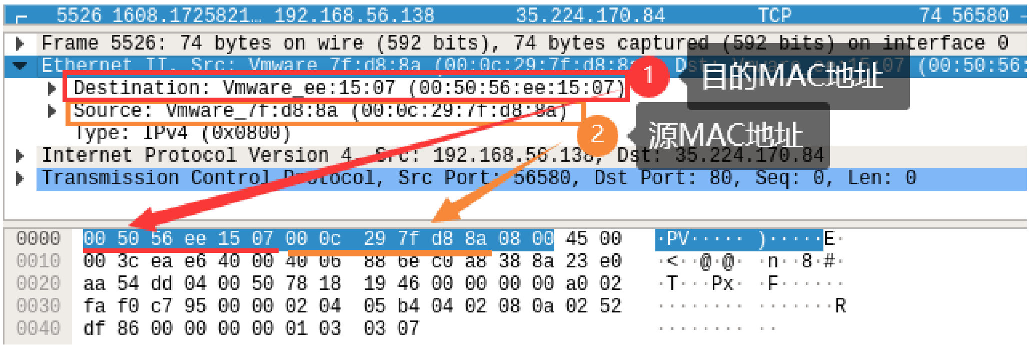
2. 找一个包含 **IP** 的数据包，这个数据包有 <u>**4**</u> 层？最高层协议是 <u>**TCP**</u>，从 **Ethernet** 开始往上，各层协议的名字分别为： <u>**IPv4, TCP**</u> 。

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：



```
 5526 1608.1725821…  192.168.56.138        35.224.170.84        TCP          74 56580 → 80 [SYN] Se
▶ Frame 5526: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
▶ Ethernet II, Src: Vmware_7f:d8:8a (00:0c:29:7f:d8:8a), Dst: Vmware_ee:15:07 (00:50:56:ee:15:07)
▼ Internet Protocol Version 4, Src: 192.168.56.138, Dst: 35.224.170.84
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 60
    Identification: 0xeae6 (60134)
  ▶ Flags: 0x4000, Don't fragment
    Time to live: 64
    Protocol: TCP (6)
    Header checksum: 0x886e [validation disabled]
    [Header checksum status: Unverified]
    Source: 192.168.56.138
    Destination: 35.224.170.84
▶ Transmission Control Protocol, Src Port: 56580, Dst Port: 80, Seq: 0, Len: 0

0000  00 50 56 ee 15 07 00 0c  29 7f d8 8a 08 00 45 00   ·PV·····)·····E·
0010  00 3c ea e6 40 00 40 06  88 6e c0 a8 38 8a 23 e0   ·<··@·@··n··8·#·
0020  aa 54 dd 04 00 50 78 18  19 46 00 00 00 00 a0 02   ·T···Px··F······
0030  fa f0 c7 95 00 00 02 04  05 b4 04 02 08 0a 02 52   ···············R
0040  df 86 00 00 00 00 01 03  03 07                     ··········
```

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：

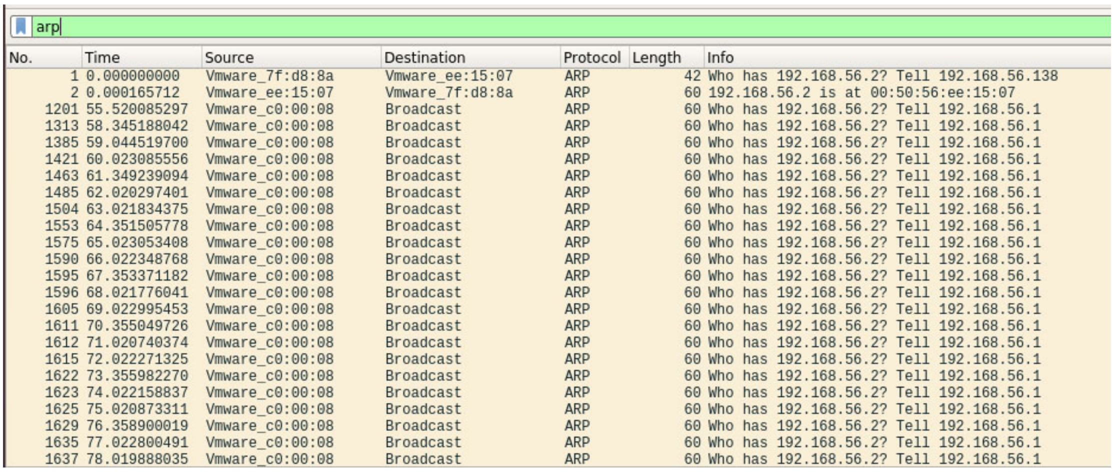


```
┌  5526 1608.1725821…  192.168.56.138        35.224.170.84        TCP           74 56580 -
▶ Frame 5526: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
▼ Ethernet II, Src: Vmware_7f:d8:8a (00:0c:29:7f:d8:8a), Dst: Vmware_ee:15:07 (00:50:56:
   ▶ Destination: Vmware_ee:15:07 (00:50:56:ee:15:07)
   ▶ Source: Vmware_7f:d8:8a (00:0c:29:7f:d8:8a)
     Type: IPv4 (0x0800)
▶ Internet Protocol Version 4, Src: 192.168.56.138, Dst: 35.224.170.84
▶ Transmission Control Protocol, Src Port: 56580, Dst Port: 80, Seq: 0, Len: 0

0000  00 50 56 ee 15 07 00 0c  29 7f d8 8a 08 00 45 00   ·PV·····  )·····E·
0010  00 3c ea e6 40 00 40 06  88 6e c0 a8 38 8a 23 e0   ·<··@·@·  ·n··8·#·
0020  aa 54 dd 04 00 50 78 18  19 46 00 00 00 00 a0 02   ·T···Px·  ·F······
0030  fa f0 c7 95 00 00 02 04  05 b4 04 02 08 0a 02 52   ········  ·······R
0040  df 86 00 00 00 00 01 03  03 07                     ········  ··
```

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： arp ，希望显示的协议类型： arp 。

截图：



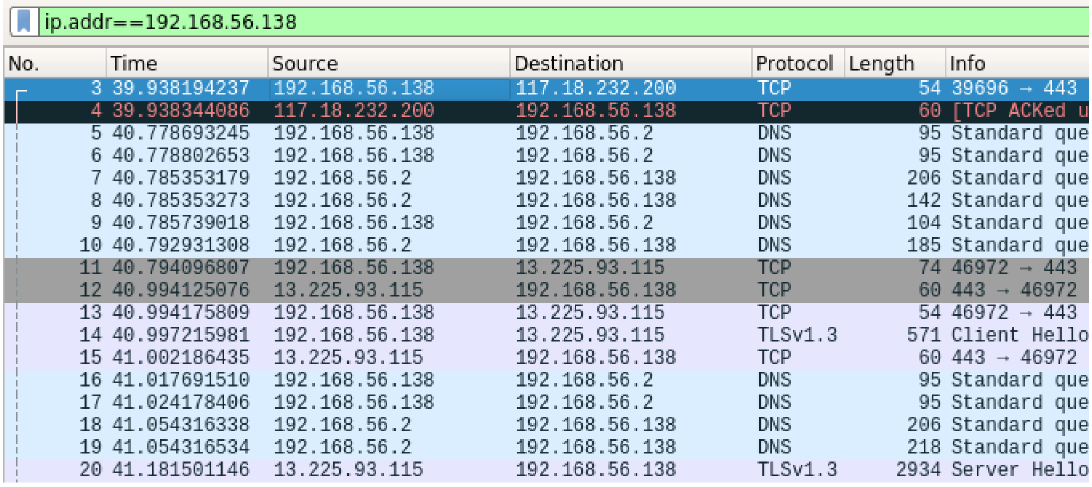
arp

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	Vmware_7f:d8:8a	Vmware_ee:15:07	ARP	42	Who has 192.168.56.2? Tell 192.168.56.138
2	0.000165712	Vmware_ee:15:07	Vmware_7f:d8:8a	ARP	60	192.168.56.2 is at 00:50:56:ee:15:07
1201	55.520085297	Vmware_c0:00:08	Broadcast	ARP	60	Who has 192.168.56.2? Tell 192.168.56.1
1313	58.345188042	Vmware_c0:00:08	Broadcast	ARP	60	Who has 192.168.56.2? Tell 192.168.56.1
1385	59.044519700	Vmware_c0:00:08	Broadcast	ARP	60	Who has 192.168.56.2? Tell 192.168.56.1
1421	60.023085556	Vmware_c0:00:08	Broadcast	ARP	60	Who has 192.168.56.2? Tell 192.168.56.1
1463	61.349239094	Vmware_c0:00:08	Broadcast	ARP	60	Who has 192.168.56.2? Tell 192.168.56.1
1485	62.020297401	Vmware_c0:00:08	Broadcast	ARP	60	Who has 192.168.56.2? Tell 192.168.56.1
1504	63.021834375	Vmware_c0:00:08	Broadcast	ARP	60	Who has 192.168.56.2? Tell 192.168.56.1
1553	64.351505778	Vmware_c0:00:08	Broadcast	ARP	60	Who has 192.168.56.2? Tell 192.168.56.1
1575	65.023053408	Vmware_c0:00:08	Broadcast	ARP	60	Who has 192.168.56.2? Tell 192.168.56.1
1590	66.022348768	Vmware_c0:00:08	Broadcast	ARP	60	Who has 192.168.56.2? Tell 192.168.56.1
1595	67.353371182	Vmware_c0:00:08	Broadcast	ARP	60	Who has 192.168.56.2? Tell 192.168.56.1
1596	68.021776041	Vmware_c0:00:08	Broadcast	ARP	60	Who has 192.168.56.2? Tell 192.168.56.1
1605	69.022995453	Vmware_c0:00:08	Broadcast	ARP	60	Who has 192.168.56.2? Tell 192.168.56.1
1611	70.355049726	Vmware_c0:00:08	Broadcast	ARP	60	Who has 192.168.56.2? Tell 192.168.56.1
1612	71.020740374	Vmware_c0:00:08	Broadcast	ARP	60	Who has 192.168.56.2? Tell 192.168.56.1
1615	72.022271325	Vmware_c0:00:08	Broadcast	ARP	60	Who has 192.168.56.2? Tell 192.168.56.1
1622	73.355982270	Vmware_c0:00:08	Broadcast	ARP	60	Who has 192.168.56.2? Tell 192.168.56.1
1623	74.022158837	Vmware_c0:00:08	Broadcast	ARP	60	Who has 192.168.56.2? Tell 192.168.56.1
1625	75.020873311	Vmware_c0:00:08	Broadcast	ARP	60	Who has 192.168.56.2? Tell 192.168.56.1
1629	76.358900019	Vmware_c0:00:08	Broadcast	ARP	60	Who has 192.168.56.2? Tell 192.168.56.1
1635	77.022800491	Vmware_c0:00:08	Broadcast	ARP	60	Who has 192.168.56.2? Tell 192.168.56.1
1637	78.019888035	Vmware_c0:00:08	Broadcast	ARP	60	Who has 192.168.56.2? Tell 192.168.56.1

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==192.168.56.138 ，希望显示的 IP 地址： 192.168.56.138 。

截图：



ip.addr==192.168.56.138

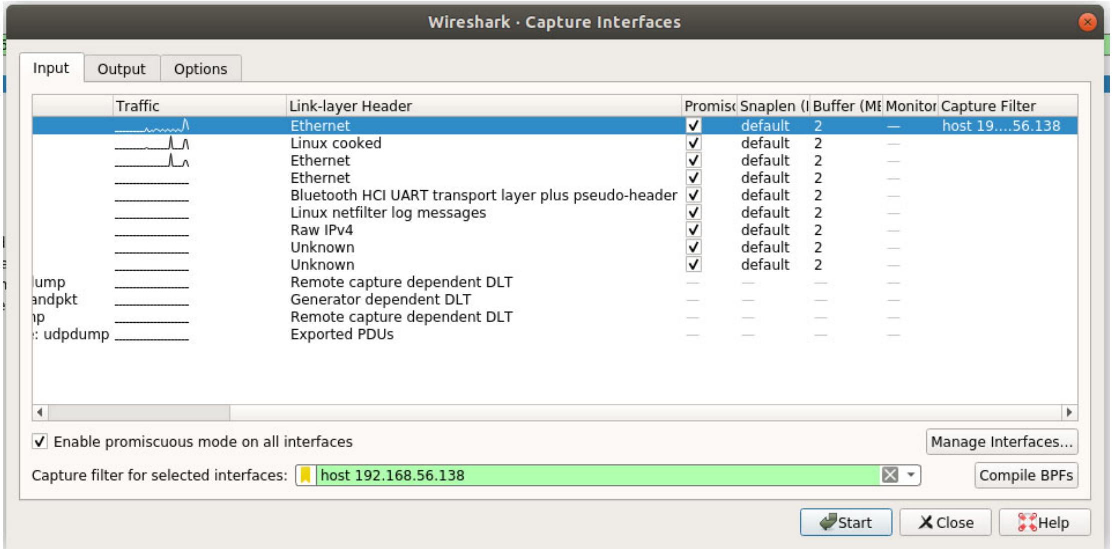
No.	Time	Source	Destination	Protocol	Length	Info
3	39.938194237	192.168.56.138	117.18.232.200	TCP	54	39696 → 443
4	39.938344086	117.18.232.200	192.168.56.138	TCP	60	[TCP ACKed u
5	40.778693245	192.168.56.138	192.168.56.2	DNS	95	Standard que
6	40.778802653	192.168.56.138	192.168.56.2	DNS	95	Standard que
7	40.785353179	192.168.56.2	192.168.56.138	DNS	206	Standard que
8	40.785353273	192.168.56.2	192.168.56.138	DNS	142	Standard que
9	40.785739018	192.168.56.138	192.168.56.2	DNS	104	Standard que
10	40.792931308	192.168.56.2	192.168.56.138	DNS	185	Standard que
11	40.794096807	192.168.56.138	13.225.93.115	TCP	74	46972 → 443
12	40.994125076	13.225.93.115	192.168.56.138	TCP	60	443 → 46972
13	40.994175809	192.168.56.138	13.225.93.115	TCP	54	46972 → 443
14	40.997215981	192.168.56.138	13.225.93.115	TLSv1.3	571	Client Hello
15	41.002186435	13.225.93.115	192.168.56.138	TCP	60	443 → 46972
16	41.017691510	192.168.56.138	192.168.56.2	DNS	95	Standard que
17	41.024178406	192.168.56.138	192.168.56.2	DNS	95	Standard que
18	41.054316338	192.168.56.2	192.168.56.138	DNS	206	Standard que
19	41.054316534	192.168.56.2	192.168.56.138	DNS	218	Standard que
20	41.181501146	13.225.93.115	192.168.56.138	TLSv1.3	2934	Server Hello

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 192.168.56.138，希望捕获的 IP 地址：192.168.56.138。

截图：

在 capture – capture options 中设置 capture filter：



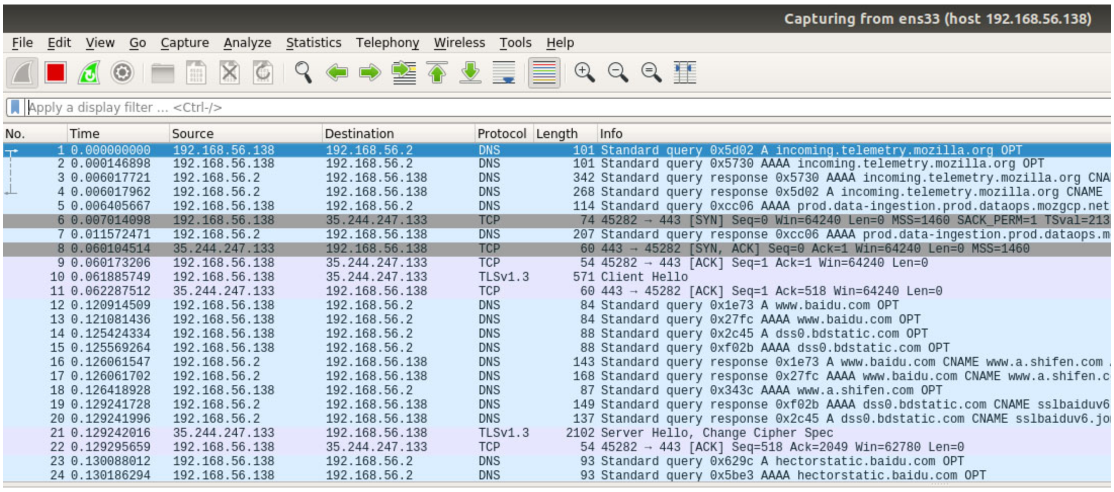
Wireshark · Capture Interfaces

	Traffic	Link-layer Header	Promisc	Snaplen (I	Buffer (ME	Monitor	Capture Filter
		Ethernet	☑	default	2	—	host 19....56.138
		Linux cooked	☑	default	2	—	
		Ethernet	☑	default	2	—	
		Ethernet	☑	default	2	—	
		Bluetooth HCI UART transport layer plus pseudo-header	☑	default	2	—	
		Linux netfilter log messages	☑	default	2	—	
		Raw IPv4	☑	default	2	—	
		Unknown	☑	default	2	—	
		Unknown	☑	default	2	—	
ump		Remote capture dependent DLT	—	—	—	—	
andpkt		Generator dependent DLT	—	—	—	—	
ıp		Remote capture dependent DLT	—	—	—	—	
: udpdump		Exported PDUs	—	—	—	—	

☑ Enable promiscuous mode on all interfaces

Capture filter for selected interfaces: host 192.168.56.138

捕获出的结果：



Capturing from ens33 (host 192.168.56.138)

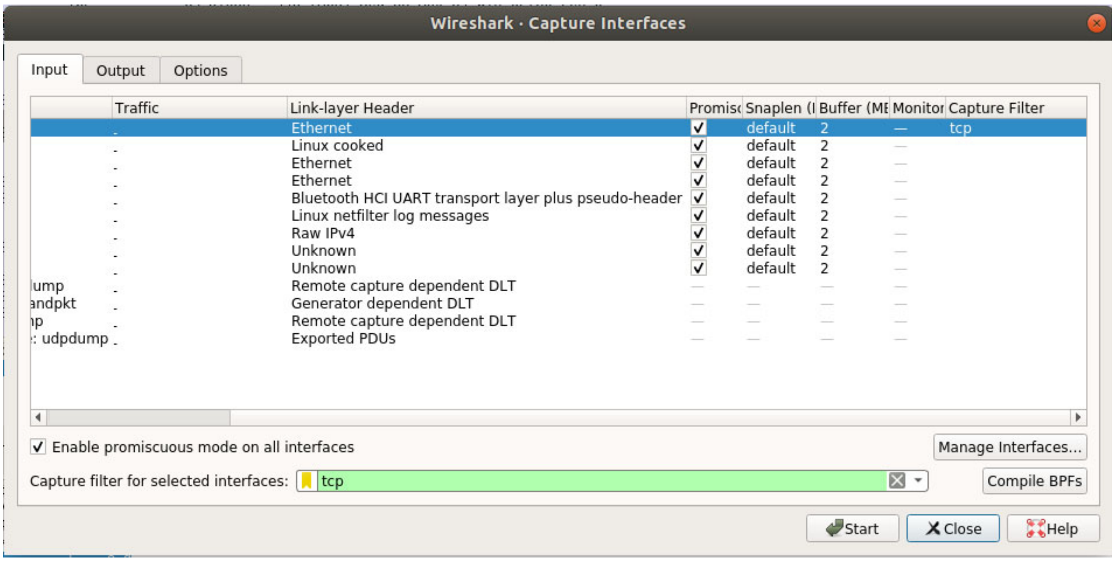
```
No.     Time          Source            Destination       Protocol Length Info
      1 0.000000000   192.168.56.138    192.168.56.2      DNS        101 Standard query 0x5d02 A incoming.telemetry.mozilla.org OPT
      2 0.000146898   192.168.56.138    192.168.56.2      DNS        101 Standard query 0x5730 AAAA incoming.telemetry.mozilla.org OPT
      3 0.006017721   192.168.56.2      192.168.56.138    DNS        342 Standard query response 0x5730 AAAA incoming.telemetry.mozilla.org CNA
      4 0.006017962   192.168.56.2      192.168.56.138    DNS        268 Standard query response 0x5d02 A incoming.telemetry.mozilla.org CNAME
      5 0.006405667   192.168.56.138    192.168.56.2      DNS        114 Standard query 0xcc06 AAAA prod.data-ingestion.prod.dataops.mozgcp.net
      6 0.007014098   192.168.56.138    35.244.247.133    TCP         74 45282 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=213
      7 0.011572471   192.168.56.2      192.168.56.138    DNS        207 Standard query response 0xcc06 AAAA prod.data-ingestion.prod.dataops.m
      8 0.060104514   35.244.247.133    192.168.56.138    TCP         60 443 → 45282 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
      9 0.060173206   192.168.56.138    35.244.247.133    TCP         54 45282 → 443 [ACK] Seq=1 Ack=1 Win=64240 Len=0
     10 0.061885749   192.168.56.138    35.244.247.133    TLSv1.3    571 Client Hello
     11 0.062287512   35.244.247.133    192.168.56.138    TCP         60 443 → 45282 [ACK] Seq=1 Ack=518 Win=64240 Len=0
     12 0.120914509   192.168.56.138    192.168.56.2      DNS         84 Standard query 0x1e73 A www.baidu.com OPT
     13 0.121081436   192.168.56.138    192.168.56.2      DNS         84 Standard query 0x27fc AAAA www.baidu.com OPT
     14 0.125424334   192.168.56.138    192.168.56.2      DNS         88 Standard query 0x2c45 A dss0.bdstatic.com OPT
     15 0.125569264   192.168.56.138    192.168.56.2      DNS         88 Standard query 0xf02b AAAA dss0.bdstatic.com OPT
     16 0.126061547   192.168.56.2      192.168.56.138    DNS        143 Standard query response 0x1e73 A www.baidu.com CNAME www.a.shifen.com .
     17 0.126061702   192.168.56.2      192.168.56.138    DNS        168 Standard query response 0x27fc AAAA www.baidu.com CNAME www.a.shifen.c
     18 0.126418928   192.168.56.138    192.168.56.2      DNS         87 Standard query 0x343c AAAA www.a.shifen.com OPT
     19 0.129241728   192.168.56.2      192.168.56.138    DNS        149 Standard query response 0xf02b AAAA dss0.bdstatic.com CNAME sslbaiduv6
     20 0.129241996   192.168.56.2      192.168.56.138    DNS        137 Standard query response 0x2c45 A dss0.bdstatic.com CNAME sslbaiduv6.jo
     21 0.129242016   35.244.247.133    192.168.56.138    TLSv1.3   2102 Server Hello, Change Cipher Spec
     22 0.129295659   192.168.56.138    35.244.247.133    TCP         54 45282 → 443 [ACK] Seq=518 Ack=2049 Win=62780 Len=0
     23 0.130088012   192.168.56.138    192.168.56.2      DNS         93 Standard query 0x629c A hectorstatic.baidu.com OPT
     24 0.130186294   192.168.56.138    192.168.56.2      DNS         93 Standard query 0x5be3 AAAA hectorstatic.baidu.com OPT
```

可见，这样捕获的结果都是以 192.168.56.138 为 source 或 destination 的。

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：<u> tcp </u>，希望捕获的协议类型：<u> tcp </u>。

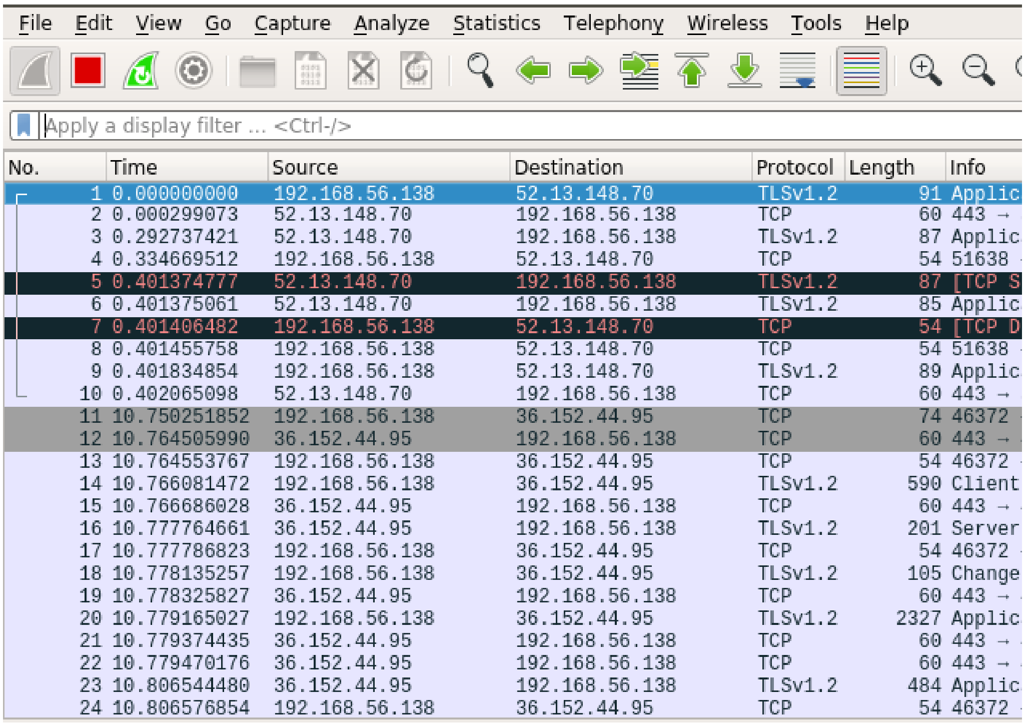
截图：



	Traffic	Link-layer Header	Promisc	Snaplen (I	Buffer (ME	Monitor	Capture Filter
		Ethernet	☑	default	2	—	tcp
		Linux cooked	☑	default	2	—	
		Ethernet	☑	default	2	—	
		Ethernet	☑	default	2	—	
		Bluetooth HCI UART transport layer plus pseudo-header	☑	default	2	—	
		Linux netfilter log messages	☑	default	2	—	
		Raw IPv4	☑	default	2	—	
		Unknown	☑	default	2	—	
		Unknown	☑	default	2	—	
ump		Remote capture dependent DLT	—	—	—	—	
andpkt		Generator dependent DLT	—	—	—	—	
ıp		Remote capture dependent DLT	—	—	—	—	
: udpdump		Exported PDUs	—	—	—	—	

☑ Enable promiscuous mode on all interfaces

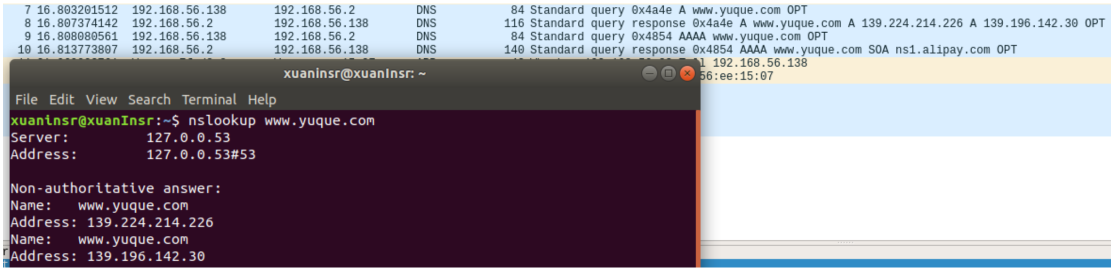
Capture filter for selected interfaces: tcp



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.56.138	52.13.148.70	TLSv1.2	91	Applic
2	0.000299073	52.13.148.70	192.168.56.138	TCP	60	443 →
3	0.292737421	52.13.148.70	192.168.56.138	TLSv1.2	87	Applic
4	0.334669512	192.168.56.138	52.13.148.70	TCP	54	51638
5	0.401374777	52.13.148.70	192.168.56.138	TLSv1.2	87	[TCP S
6	0.401375061	52.13.148.70	192.168.56.138	TLSv1.2	85	Applic
7	0.401406482	192.168.56.138	52.13.148.70	TCP	54	[TCP D
8	0.401455758	192.168.56.138	52.13.148.70	TCP	54	51638
9	0.401834854	192.168.56.138	52.13.148.70	TLSv1.2	89	Applic
10	0.402065098	52.13.148.70	192.168.56.138	TCP	60	443 →
11	10.750251852	192.168.56.138	36.152.44.95	TCP	74	46372
12	10.764505990	36.152.44.95	192.168.56.138	TCP	60	443 →
13	10.764553767	192.168.56.138	36.152.44.95	TCP	54	46372
14	10.766081472	192.168.56.138	36.152.44.95	TLSv1.2	590	Client
15	10.766686028	36.152.44.95	192.168.56.138	TCP	60	443 →
16	10.777764661	36.152.44.95	192.168.56.138	TLSv1.2	201	Server
17	10.777786823	192.168.56.138	36.152.44.95	TCP	54	46372
18	10.778135257	192.168.56.138	36.152.44.95	TLSv1.2	105	Change
19	10.778325827	36.152.44.95	192.168.56.138	TCP	60	443 →
20	10.779165027	192.168.56.138	36.152.44.95	TLSv1.2	2327	Applic
21	10.779374435	36.152.44.95	192.168.56.138	TCP	60	443 →
22	10.779470176	36.152.44.95	192.168.56.138	TCP	60	443 →
23	10.806544480	36.152.44.95	192.168.56.138	TLSv1.2	484	Applic
24	10.806576854	192.168.56.138	36.152.44.95	TCP	54	46372

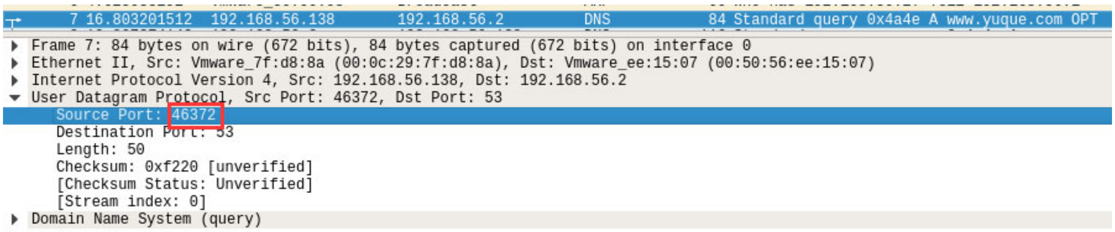
✧ Part Two

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, IPv4, UDP, DNS 五层**。 使用的服务方端口是：**46372**。



```
 7 16.803201512  192.168.56.138    192.168.56.2      DNS      84 Standard query 0x4a4e A www.yuque.com OPT
 8 16.807374142  192.168.56.2      192.168.56.138    DNS     116 Standard query response 0x4a4e A www.yuque.com A 139.224.214.226 A 139.196.142.30 OPT
 9 16.808080561  192.168.56.138    192.168.56.2      DNS      84 Standard query 0x4854 AAAA www.yuque.com OPT
10 16.813773807  192.168.56.2      192.168.56.138    DNS     140 Standard query response 0x4854 AAAA www.yuque.com SOA ns1.alipay.com OPT
```

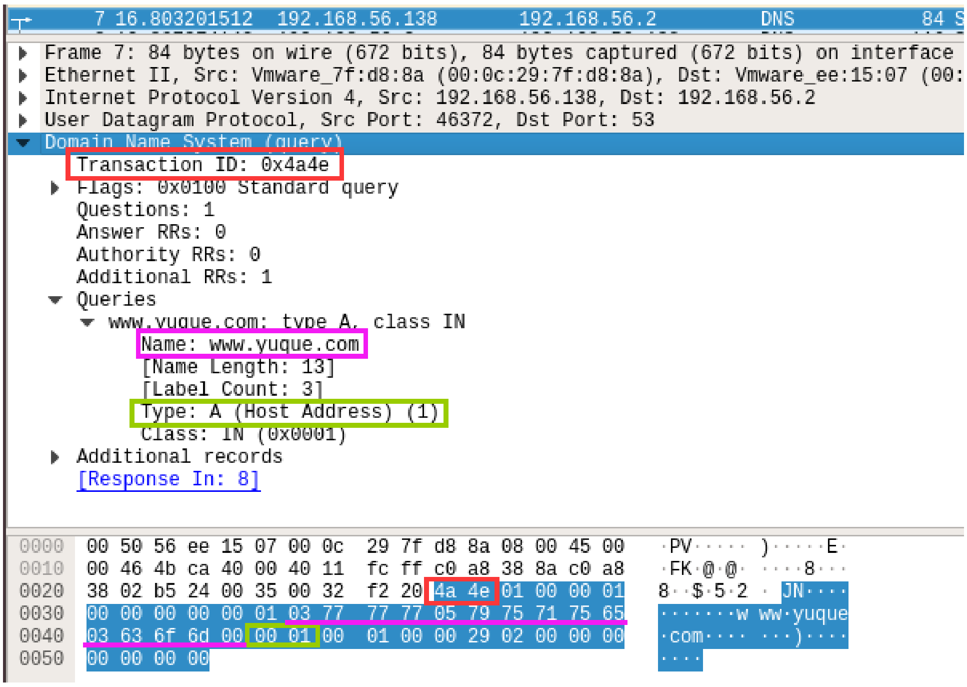
```
xuaninsr@xuanInsr:~$ nslookup www.yuque.com
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
Name:   www.yuque.com
Address: 139.224.214.226
Name:   www.yuque.com
Address: 139.196.142.30
```



```
  7 16.803201512  192.168.56.138     192.168.56.2       DNS       84 Standard query 0x4a4e A www.yuque.com OPT
▸ Frame 7: 84 bytes on wire (672 bits), 84 bytes captured (672 bits) on interface 0
▸ Ethernet II, Src: Vmware_7f:d8:8a (00:0c:29:7f:d8:8a), Dst: Vmware_ee:15:07 (00:50:56:ee:15:07)
▸ Internet Protocol Version 4, Src: 192.168.56.138, Dst: 192.168.56.2
▾ User Datagram Protocol, Src Port: 46372, Dst Port: 53
    Source Port: 46372
    Destination Port: 53
    Length: 50
    Checksum: 0xf220 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 0]
▸ Domain Name System (query)
```

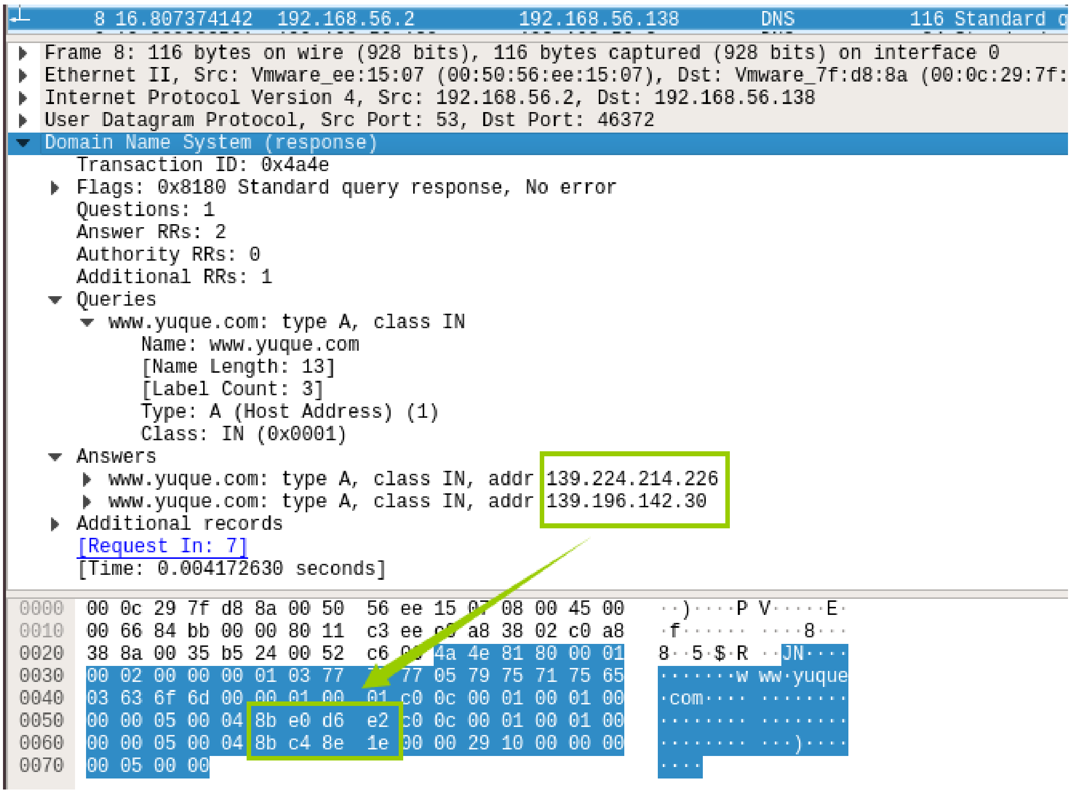
分别选择一个请求包和一个响应包，展开最高层协议的详细内容，标出交易 **ID**、查询类型、查询的域名内容以及查询结果。

请求包：（红色标出的是 transaction ID，草绿色标出的是查询类型，品红色标出的是查询的域名内容）



```
  7 16.803201512  192.168.56.138       192.168.56.2        DNS       84 S
▸ Frame 7: 84 bytes on wire (672 bits), 84 bytes captured (672 bits) on interface
▸ Ethernet II, Src: Vmware_7f:d8:8a (00:0c:29:7f:d8:8a), Dst: Vmware_ee:15:07 (00:
▸ Internet Protocol Version 4, Src: 192.168.56.138, Dst: 192.168.56.2
▸ User Datagram Protocol, Src Port: 46372, Dst Port: 53
▾ Domain Name System (query)
    Transaction ID: 0x4a4e
  ▸ Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 1
  ▾ Queries
    ▾ www.yuque.com: type A, class IN
        Name: www.yuque.com
        [Name Length: 13]
        [Label Count: 3]
        Type: A (Host Address) (1)
        Class: IN (0x0001)
  ▸ Additional records
    [Response In: 8]

0000  00 50 56 ee 15 07 00 0c  29 7f d8 8a 08 00 45 00   ·PV····· )·····E·
0010  00 46 4b ca 40 00 40 11  fc ff c0 a8 38 8a c0 a8   ·FK·@·@· ····8···
0020  38 02 b5 24 00 35 00 32  f2 20 4a 4e 01 00 00 01   8··$·5·2 · JN····
0030  00 00 00 00 00 01 03 77  77 77 05 79 75 71 75 65   ·······w ww·yuque
0040  03 63 6f 6d 00 00 01 00  01 00 00 29 02 00 00 00   ·com···· ···)····
0050  00 00 00 00                                        ····
```

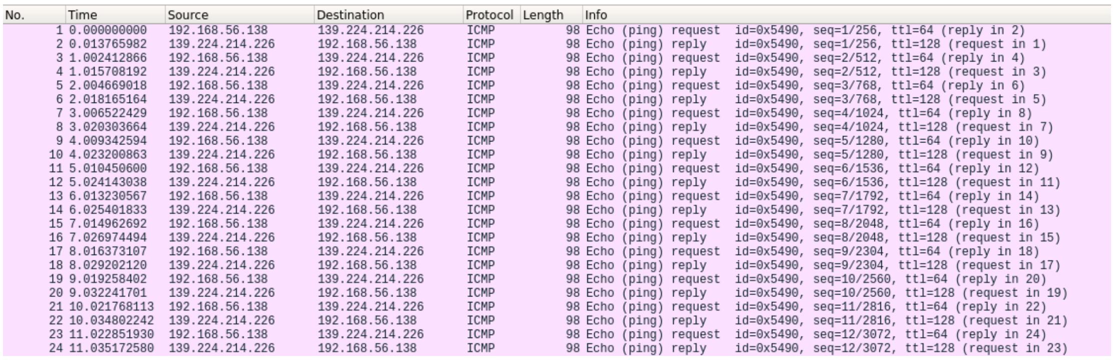
响应包（草绿色标出的是查询结果）：



```
     8 16.807374142   192.168.56.2         192.168.56.138        DNS          116 Standard q
▶ Frame 8: 116 bytes on wire (928 bits), 116 bytes captured (928 bits) on interface 0
▶ Ethernet II, Src: Vmware_ee:15:07 (00:50:56:ee:15:07), Dst: Vmware_7f:d8:8a (00:0c:29:7f:
▶ Internet Protocol Version 4, Src: 192.168.56.2, Dst: 192.168.56.138
▶ User Datagram Protocol, Src Port: 53, Dst Port: 46372
▼ Domain Name System (response)
     Transaction ID: 0x4a4e
   ▶ Flags: 0x8180 Standard query response, No error
     Questions: 1
     Answer RRs: 2
     Authority RRs: 0
     Additional RRs: 1
   ▼ Queries
     ▼ www.yuque.com: type A, class IN
          Name: www.yuque.com
          [Name Length: 13]
          [Label Count: 3]
          Type: A (Host Address) (1)
          Class: IN (0x0001)
   ▼ Answers
     ▶ www.yuque.com: type A, class IN, addr 139.224.214.226
     ▶ www.yuque.com: type A, class IN, addr 139.196.142.30
   ▶ Additional records
     [Request In: 7]
     [Time: 0.004172630 seconds]

0000  00 0c 29 7f d8 8a 00 50  56 ee 15 07 08 00 45 00   ··)····P V·····E·
0010  00 66 84 bb 00 00 80 11  c3 ee c0 a8 38 02 c0 a8   ·f······ ····8···
0020  38 8a 00 35 b5 24 00 52  c6 00 4a 4e 81 80 00 01   8··5·$·R ··JN····
0030  00 02 00 00 00 01 03 77  77 77 05 79 75 71 75 65   ·······w ww·yuque
0040  03 63 6f 6d 00 00 01 00  01 c0 0c 00 01 00 01 00   ·com···· ········
0050  00 00 05 00 04 8b e0 d6  e2 c0 0c 00 01 00 01 00   ········ ········
0060  00 00 05 00 04 8b c4 8e  1e 00 00 29 10 00 00 00   ········ ···)····
0070  00 05 00 00                                        ····
```

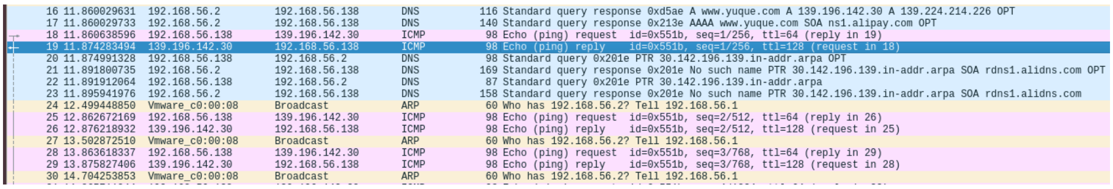
任务 2：使用 **Ping** 命令，分别测试某个 **IP** 地址和某个域名的连通性，并捕获数据包。捕获到了哪些相关协议数据包？

Ping IP 地址时： <u>ICMP (也可能有 ARP)</u>



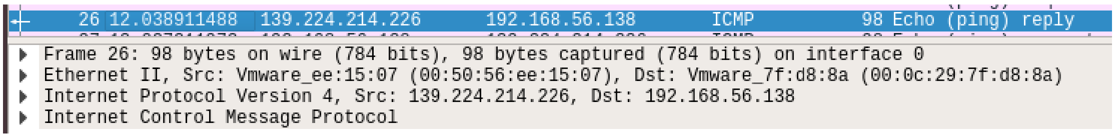
```
No.     Time           Source            Destination       Protocol Length Info
      1 0.000000000    192.168.56.138    139.224.214.226   ICMP       98 Echo (ping) request  id=0x5490, seq=1/256, ttl=64 (reply in 2)
      2 0.013765982    139.224.214.226   192.168.56.138    ICMP       98 Echo (ping) reply    id=0x5490, seq=1/256, ttl=128 (request in 1)
      3 1.002412866    192.168.56.138    139.224.214.226   ICMP       98 Echo (ping) request  id=0x5490, seq=2/512, ttl=64 (reply in 4)
      4 1.015708192    139.224.214.226   192.168.56.138    ICMP       98 Echo (ping) reply    id=0x5490, seq=2/512, ttl=128 (request in 3)
      5 2.004669018    192.168.56.138    139.224.214.226   ICMP       98 Echo (ping) request  id=0x5490, seq=3/768, ttl=64 (reply in 6)
      6 2.018165164    139.224.214.226   192.168.56.138    ICMP       98 Echo (ping) reply    id=0x5490, seq=3/768, ttl=128 (request in 5)
      7 3.006522429    192.168.56.138    139.224.214.226   ICMP       98 Echo (ping) request  id=0x5490, seq=4/1024, ttl=64 (reply in 8)
      8 3.020303664    139.224.214.226   192.168.56.138    ICMP       98 Echo (ping) reply    id=0x5490, seq=4/1024, ttl=128 (request in 7)
      9 4.009342594    192.168.56.138    139.224.214.226   ICMP       98 Echo (ping) request  id=0x5490, seq=5/1280, ttl=64 (reply in 10)
     10 4.023200863    139.224.214.226   192.168.56.138    ICMP       98 Echo (ping) reply    id=0x5490, seq=5/1280, ttl=128 (request in 9)
     11 5.010450600    192.168.56.138    139.224.214.226   ICMP       98 Echo (ping) request  id=0x5490, seq=6/1536, ttl=64 (reply in 12)
     12 5.024143038    139.224.214.226   192.168.56.138    ICMP       98 Echo (ping) reply    id=0x5490, seq=6/1536, ttl=128 (request in 11)
     13 6.013230567    192.168.56.138    139.224.214.226   ICMP       98 Echo (ping) request  id=0x5490, seq=7/1792, ttl=64 (reply in 14)
     14 6.025401833    139.224.214.226   192.168.56.138    ICMP       98 Echo (ping) reply    id=0x5490, seq=7/1792, ttl=128 (request in 13)
     15 7.014962692    192.168.56.138    139.224.214.226   ICMP       98 Echo (ping) request  id=0x5490, seq=8/2048, ttl=64 (reply in 16)
     16 7.026974494    139.224.214.226   192.168.56.138    ICMP       98 Echo (ping) reply    id=0x5490, seq=8/2048, ttl=128 (request in 15)
     17 8.016373107    192.168.56.138    139.224.214.226   ICMP       98 Echo (ping) request  id=0x5490, seq=9/2304, ttl=64 (reply in 18)
     18 8.029202120    139.224.214.226   192.168.56.138    ICMP       98 Echo (ping) reply    id=0x5490, seq=9/2304, ttl=128 (request in 17)
     19 9.019258402    192.168.56.138    139.224.214.226   ICMP       98 Echo (ping) request  id=0x5490, seq=10/2560, ttl=64 (reply in 20)
     20 9.032241701    139.224.214.226   192.168.56.138    ICMP       98 Echo (ping) reply    id=0x5490, seq=10/2560, ttl=128 (request in 19)
     21 10.021768113   192.168.56.138    139.224.214.226   ICMP       98 Echo (ping) request  id=0x5490, seq=11/2816, ttl=64 (reply in 22)
     22 10.034802242   139.224.214.226   192.168.56.138    ICMP       98 Echo (ping) reply    id=0x5490, seq=11/2816, ttl=128 (request in 21)
     23 11.022851930   192.168.56.138    139.224.214.226   ICMP       98 Echo (ping) request  id=0x5490, seq=12/3072, ttl=64 (reply in 24)
     24 11.035172580   139.224.214.226   192.168.56.138    ICMP       98 Echo (ping) reply    id=0x5490, seq=12/3072, ttl=128 (request in 23)
```

Ping 域名时： <u>DNS, ICMP, ARP</u>



```
     16 11.860029631   192.168.56.2      192.168.56.138    DNS       116 Standard query response 0xd5ae A www.yuque.com A 139.196.142.30 A 139.224.214.226 OPT
     17 11.860029733   192.168.56.2      192.168.56.138    DNS       140 Standard query response 0x213e AAAA www.yuque.com SOA ns1.alipay.com OPT
     18 11.860638596   192.168.56.138    139.196.142.30    ICMP       98 Echo (ping) request  id=0x551b, seq=1/256, ttl=64 (reply in 19)
     19 11.874283494   139.196.142.30    192.168.56.138    ICMP       98 Echo (ping) reply    id=0x551b, seq=1/256, ttl=128 (request in 18)
     20 11.874991328   192.168.56.138    192.168.56.2      DNS        98 Standard query 0x201e PTR 30.142.196.139.in-addr.arpa OPT
     21 11.891800735   192.168.56.2      192.168.56.138    DNS       169 Standard query response 0x201e No such name PTR 30.142.196.139.in-addr.arpa SOA rdns1.alidns.com OPT
     22 11.891912064   192.168.56.138    192.168.56.2      DNS        87 Standard query 0x201e PTR 30.142.196.139.in-addr.arpa
     23 11.895941976   192.168.56.2      192.168.56.138    DNS       158 Standard query response 0x201e No such name PTR 30.142.196.139.in-addr.arpa SOA rdns1.alidns.com
     24 12.499448850   Vmware_c0:00:08   Broadcast         ARP        60 Who has 192.168.56.2? Tell 192.168.56.1
     25 12.862672169   192.168.56.138    139.196.142.30    ICMP       98 Echo (ping) request  id=0x551b, seq=2/512, ttl=64 (reply in 26)
     26 12.876218932   139.196.142.30    192.168.56.138    ICMP       98 Echo (ping) reply    id=0x551b, seq=2/512, ttl=128 (request in 25)
     27 13.502872510   Vmware_c0:00:08   Broadcast         ARP        60 Who has 192.168.56.2? Tell 192.168.56.1
     28 13.863618337   192.168.56.138    139.196.142.30    ICMP       98 Echo (ping) request  id=0x551b, seq=3/768, ttl=64 (reply in 29)
     29 13.875827406   139.196.142.30    192.168.56.138    ICMP       98 Echo (ping) reply    id=0x551b, seq=3/768, ttl=128 (request in 28)
     30 14.704253853   Vmware_c0:00:08   Broadcast         ARP        60 Who has 192.168.56.2? Tell 192.168.56.1
```

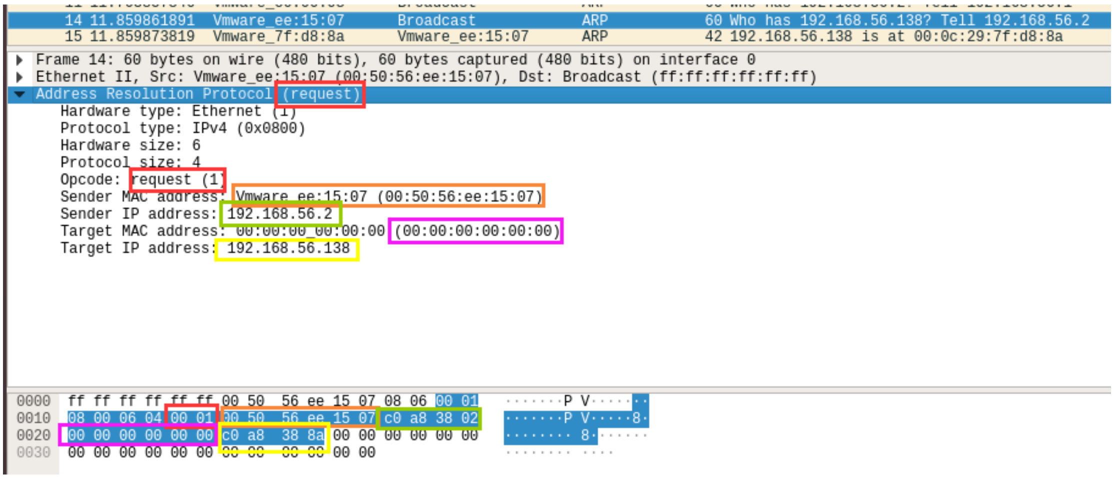
ICMP 数据包分别由哪几层协议构成？ Frame, Ethernet, IPv4, ICMP



```
26 12.038911488  139.224.214.226    192.168.56.138    ICMP    98 Echo (ping) reply
▶ Frame 26: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
▶ Ethernet II, Src: Vmware_ee:15:07 (00:50:56:ee:15:07), Dst: Vmware_7f:d8:8a (00:0c:29:7f:d8:8a)
▶ Internet Protocol Version 4, Src: 139.224.214.226, Dst: 192.168.56.138
▶ Internet Control Message Protocol
```

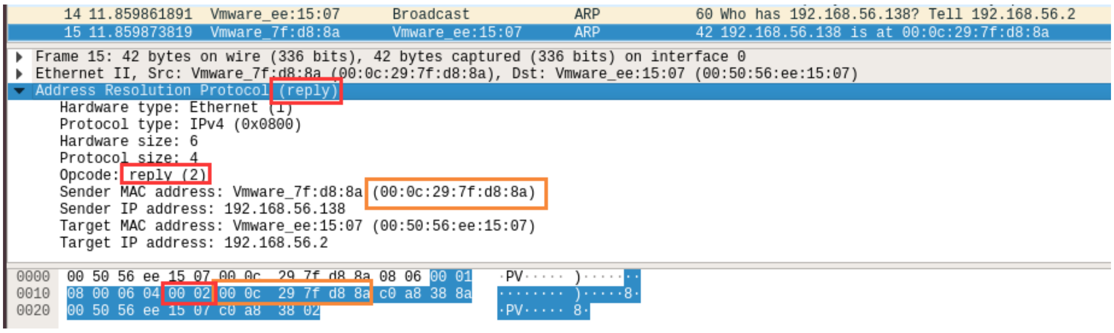
分别选择一个 **ARP** 请求和响应数据包，展开最高层协议的详细内容，标出操作码、发送者 **IP** 地址、发送者 **MAC** 地址、查询的目标 **IP** 地址、**Ethernet** 层的目标 **MAC** 地址以及查询结果。

请求包：



```
14 11.859861891  Vmware_ee:15:07  Broadcast        ARP   60 Who has 192.168.56.138? Tell 192.168.56.2
15 11.859873819  Vmware_7f:d8:8a  Vmware_ee:15:07  ARP   42 192.168.56.138 is at 00:0c:29:7f:d8:8a
▶ Frame 14: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
▶ Ethernet II, Src: Vmware_ee:15:07 (00:50:56:ee:15:07), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
▼ Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: Vmware_ee:15:07 (00:50:56:ee:15:07)
    Sender IP address: 192.168.56.2
    Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Target IP address: 192.168.56.138

0000  ff ff ff ff ff ff 00 50  56 ee 15 07 08 06 00 01   ·······P V·······
0010  08 00 06 04 00 01 00 50  56 ee 15 07 c0 a8 38 02   ·······P V·····8·
0020  00 00 00 00 00 00 c0 a8  38 8a 00 00 00 00 00 00   ········ 8·······
0030  00 00 00 00 00 00 00 00  00 00 00 00               ········ ····
```

响应包：（橙色标出的即为查询结果）

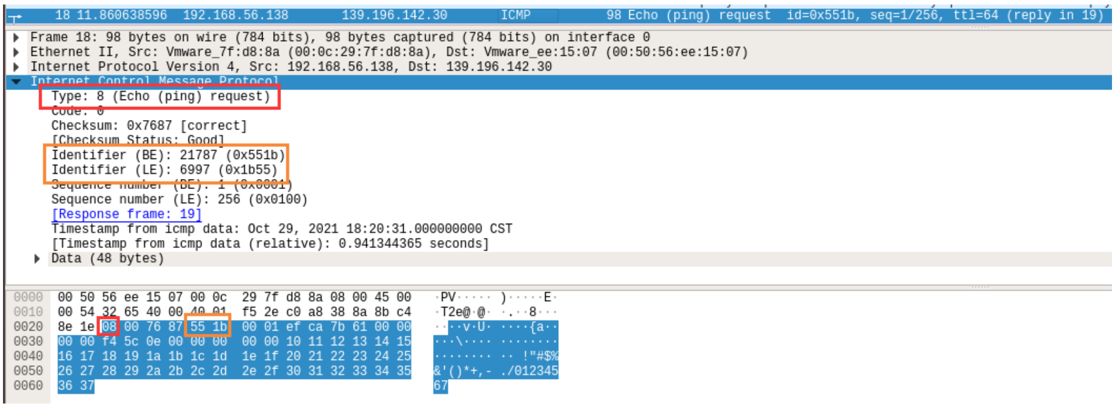


```
14 11.859861891  Vmware_ee:15:07  Broadcast        ARP   60 Who has 192.168.56.138? Tell 192.168.56.2
15 11.859873819  Vmware_7f:d8:8a  Vmware_ee:15:07  ARP   42 192.168.56.138 is at 00:0c:29:7f:d8:8a
▶ Frame 15: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
▶ Ethernet II, Src: Vmware_7f:d8:8a (00:0c:29:7f:d8:8a), Dst: Vmware_ee:15:07 (00:50:56:ee:15:07)
▼ Address Resolution Protocol (reply)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: reply (2)
    Sender MAC address: Vmware_7f:d8:8a (00:0c:29:7f:d8:8a)
    Sender IP address: 192.168.56.138
    Target MAC address: Vmware_ee:15:07 (00:50:56:ee:15:07)
    Target IP address: 192.168.56.2

0000  00 50 56 ee 15 07 00 0c  29 7f d8 8a 08 06 00 01   ·PV····· )·······
0010  08 00 06 04 00 02 00 0c  29 7f d8 8a c0 a8 38 8a   ········ )·····8·
0020  00 50 56 ee 15 07 c0 a8  38 02                     ·PV····· 8·
```

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。

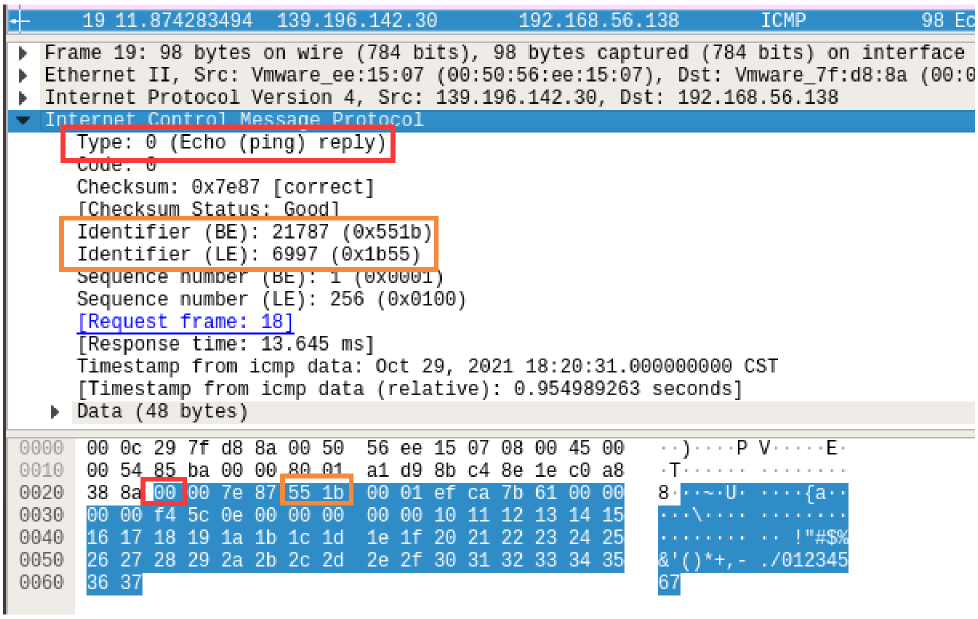
（红色表示类型，橙色表示序号）

请求包：



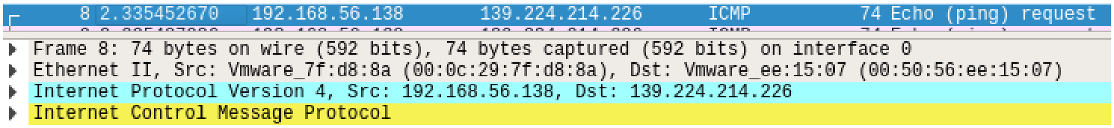
```
  18 11.860638596  192.168.56.138       139.196.142.30       ICMP      98 Echo (ping) request  id=0x551b, seq=1/256, ttl=64 (reply in 19)
▶ Frame 18: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
▶ Ethernet II, Src: Vmware_7f:d8:8a (00:0c:29:7f:d8:8a), Dst: Vmware_ee:15:07 (00:50:56:ee:15:07)
▶ Internet Protocol Version 4, Src: 192.168.56.138, Dst: 139.196.142.30
▼ Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0x7687 [correct]
    [Checksum Status: Good]
    Identifier (BE): 21787 (0x551b)
    Identifier (LE): 6997 (0x1b55)
    Sequence number (BE): 1 (0x0001)
    Sequence number (LE): 256 (0x0100)
    [Response frame: 19]
    Timestamp from icmp data: Oct 29, 2021 18:20:31.000000000 CST
    [Timestamp from icmp data (relative): 0.941344365 seconds]
  ▶ Data (48 bytes)

0000  00 50 56 ee 15 07 00 0c  29 7f d8 8a 08 00 45 00   ·PV·····  )·····E·
0010  00 54 32 65 40 00 40 01  f5 2e c0 a8 38 8a 8b c4   ·T2e@·@·  ·.··8···
0020  8e 1e 08 00 76 87 55 1b  00 01 ef ca 7b 61 00 00   ····v·U·  ····{a··
0030  00 00 f4 5c 0e 00 00 00  00 00 10 11 12 13 14 15   ···\····  ········
0040  16 17 18 19 1a 1b 1c 1d  1e 1f 20 21 22 23 24 25   ········  ·· !"#$%
0050  26 27 28 29 2a 2b 2c 2d  2e 2f 30 31 32 33 34 35   &'()*+,-  ./012345
0060  36 37                                              67
```

响应包：



```
  19 11.874283494  139.196.142.30       192.168.56.138       ICMP      98 Ec
▶ Frame 19: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface
▶ Ethernet II, Src: Vmware_ee:15:07 (00:50:56:ee:15:07), Dst: Vmware_7f:d8:8a (00:0
▶ Internet Protocol Version 4, Src: 139.196.142.30, Dst: 192.168.56.138
▼ Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0
    Checksum: 0x7e87 [correct]
    [Checksum Status: Good]
    Identifier (BE): 21787 (0x551b)
    Identifier (LE): 6997 (0x1b55)
    Sequence number (BE): 1 (0x0001)
    Sequence number (LE): 256 (0x0100)
    [Request frame: 18]
    [Response time: 13.645 ms]
    Timestamp from icmp data: Oct 29, 2021 18:20:31.000000000 CST
    [Timestamp from icmp data (relative): 0.954989263 seconds]
  ▶ Data (48 bytes)

0000  00 0c 29 7f d8 8a 00 50  56 ee 15 07 08 00 45 00   ··)····P V·····E·
0010  00 54 85 ba 00 00 80 01  a1 d9 8b c4 8e 1e c0 a8   ·T······ ········
0020  38 8a 00 00 7e 87 55 1b  00 01 ef ca 7b 61 00 00   8···~·U· ····{a··
0030  00 00 f4 5c 0e 00 00 00  00 00 10 11 12 13 14 15   ···\···· ········
0040  16 17 18 19 1a 1b 1c 1d  1e 1f 20 21 22 23 24 25   ········ ·· !"#$%
0050  26 27 28 29 2a 2b 2c 2d  2e 2f 30 31 32 33 34 35   &'()*+,- ./012345
0060  36 37                                              67
```

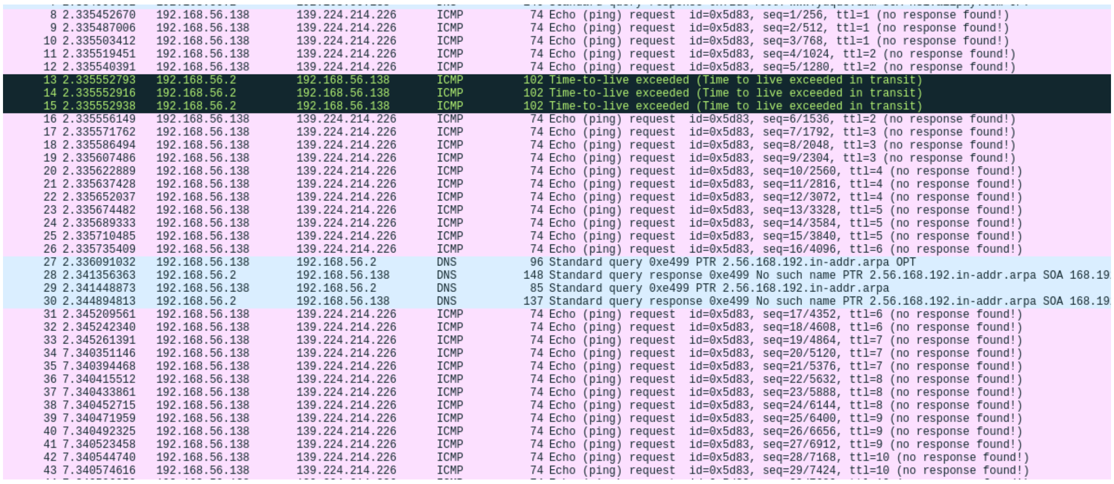
任务 3：使用 **Tracert** 命令（**Mac** 下使用 **Traceroute** 命令），跟踪某个外部 **IP** 地址的路由，并捕获这次的数据包。跟踪路由使用的数据包协议类型是：**ICMP**，数据包由几层协议构成？**4 层：Frame, Ethernet, IPv4, ICMP** 。



```
  8 2.335452670   192.168.56.138       139.224.214.226      ICMP          74 Echo (ping) request
▸ Frame 8: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
▸ Ethernet II, Src: Vmware_7f:d8:8a (00:0c:29:7f:d8:8a), Dst: Vmware_ee:15:07 (00:50:56:ee:15:07)
▸ Internet Protocol Version 4, Src: 192.168.56.138, Dst: 139.224.214.226
▸ Internet Control Message Protocol
```

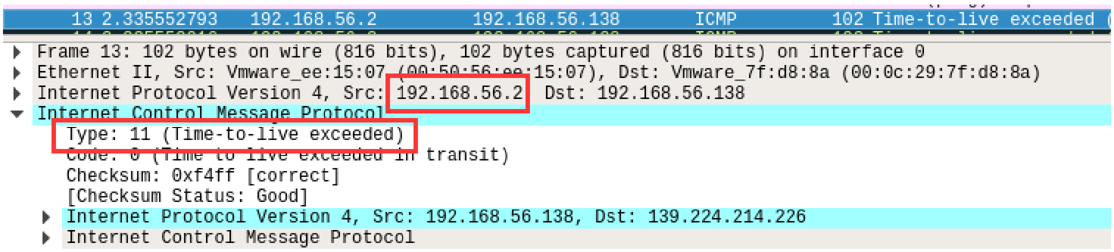
观察并记录请求包中 **IP** 协议层的 **TTL** 字段变化规律，第一个请求的 **TTL** 等于 **1**，同样 **TTL** 的请求连续发送了 **3** 个，然后每次 **TTL** 增加了 **1**，最后一个请求的 **TTL** 等于 **10** 。附上截图：

运行 sudo traceroute www.yuque.com -I：



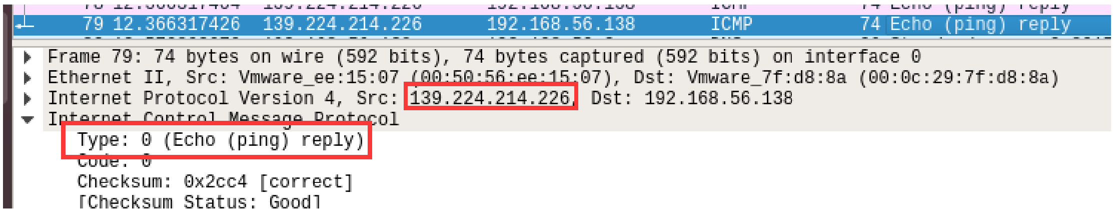
```
  8 2.335452670  192.168.56.138   139.224.214.226  ICMP     74 Echo (ping) request  id=0x5d83, seq=1/256, ttl=1 (no response found!)
  9 2.335487006  192.168.56.138   139.224.214.226  ICMP     74 Echo (ping) request  id=0x5d83, seq=2/512, ttl=1 (no response found!)
 10 2.335503412  192.168.56.138   139.224.214.226  ICMP     74 Echo (ping) request  id=0x5d83, seq=3/768, ttl=1 (no response found!)
 11 2.335519451  192.168.56.138   139.224.214.226  ICMP     74 Echo (ping) request  id=0x5d83, seq=4/1024, ttl=2 (no response found!)
 12 2.335540391  192.168.56.138   139.224.214.226  ICMP     74 Echo (ping) request  id=0x5d83, seq=5/1280, ttl=2 (no response found!)
 13 2.335552793  192.168.56.2     192.168.56.138   ICMP    102 Time-to-live exceeded (Time to live exceeded in transit)
 14 2.335552916  192.168.56.2     192.168.56.138   ICMP    102 Time-to-live exceeded (Time to live exceeded in transit)
 15 2.335552938  192.168.56.2     192.168.56.138   ICMP    102 Time-to-live exceeded (Time to live exceeded in transit)
 16 2.335556149  192.168.56.138   139.224.214.226  ICMP     74 Echo (ping) request  id=0x5d83, seq=6/1536, ttl=2 (no response found!)
 17 2.335571762  192.168.56.138   139.224.214.226  ICMP     74 Echo (ping) request  id=0x5d83, seq=7/1792, ttl=3 (no response found!)
 18 2.335586494  192.168.56.138   139.224.214.226  ICMP     74 Echo (ping) request  id=0x5d83, seq=8/2048, ttl=3 (no response found!)
 19 2.335607486  192.168.56.138   139.224.214.226  ICMP     74 Echo (ping) request  id=0x5d83, seq=9/2304, ttl=3 (no response found!)
 20 2.335622889  192.168.56.138   139.224.214.226  ICMP     74 Echo (ping) request  id=0x5d83, seq=10/2560, ttl=4 (no response found!)
 21 2.335637428  192.168.56.138   139.224.214.226  ICMP     74 Echo (ping) request  id=0x5d83, seq=11/2816, ttl=4 (no response found!)
 22 2.335652037  192.168.56.138   139.224.214.226  ICMP     74 Echo (ping) request  id=0x5d83, seq=12/3072, ttl=4 (no response found!)
 23 2.335674482  192.168.56.138   139.224.214.226  ICMP     74 Echo (ping) request  id=0x5d83, seq=13/3328, ttl=5 (no response found!)
 24 2.335689333  192.168.56.138   139.224.214.226  ICMP     74 Echo (ping) request  id=0x5d83, seq=14/3584, ttl=5 (no response found!)
 25 2.335710485  192.168.56.138   139.224.214.226  ICMP     74 Echo (ping) request  id=0x5d83, seq=15/3840, ttl=5 (no response found!)
 26 2.335735409  192.168.56.138   139.224.214.226  ICMP     74 Echo (ping) request  id=0x5d83, seq=16/4096, ttl=6 (no response found!)
 27 2.336091032  192.168.56.138   192.168.56.2     DNS      96 Standard query 0xe499 PTR 2.56.168.192.in-addr.arpa OPT
 28 2.341356363  192.168.56.2     192.168.56.138   DNS     148 Standard query response 0xe499 No such name PTR 2.56.168.192.in-addr.arpa SOA 168.19
 29 2.341448873  192.168.56.138   192.168.56.2     DNS      85 Standard query 0xe499 PTR 2.56.168.192.in-addr.arpa
 30 2.344894813  192.168.56.2     192.168.56.138   DNS     137 Standard query response 0xe499 No such name PTR 2.56.168.192.in-addr.arpa SOA 168.19
 31 2.345209561  192.168.56.138   139.224.214.226  ICMP     74 Echo (ping) request  id=0x5d83, seq=17/4352, ttl=6 (no response found!)
 32 2.345242340  192.168.56.138   139.224.214.226  ICMP     74 Echo (ping) request  id=0x5d83, seq=18/4608, ttl=6 (no response found!)
 33 2.345261391  192.168.56.138   139.224.214.226  ICMP     74 Echo (ping) request  id=0x5d83, seq=19/4864, ttl=7 (no response found!)
 34 7.340351146  192.168.56.138   139.224.214.226  ICMP     74 Echo (ping) request  id=0x5d83, seq=20/5120, ttl=7 (no response found!)
 35 7.340394468  192.168.56.138   139.224.214.226  ICMP     74 Echo (ping) request  id=0x5d83, seq=21/5376, ttl=7 (no response found!)
 36 7.340415512  192.168.56.138   139.224.214.226  ICMP     74 Echo (ping) request  id=0x5d83, seq=22/5632, ttl=8 (no response found!)
 37 7.340433861  192.168.56.138   139.224.214.226  ICMP     74 Echo (ping) request  id=0x5d83, seq=23/5888, ttl=8 (no response found!)
 38 7.340452715  192.168.56.138   139.224.214.226  ICMP     74 Echo (ping) request  id=0x5d83, seq=24/6144, ttl=8 (no response found!)
 39 7.340471959  192.168.56.138   139.224.214.226  ICMP     74 Echo (ping) request  id=0x5d83, seq=25/6400, ttl=9 (no response found!)
 40 7.340492325  192.168.56.138   139.224.214.226  ICMP     74 Echo (ping) request  id=0x5d83, seq=26/6656, ttl=9 (no response found!)
 41 7.340523458  192.168.56.138   139.224.214.226  ICMP     74 Echo (ping) request  id=0x5d83, seq=27/6912, ttl=9 (no response found!)
 42 7.340544740  192.168.56.138   139.224.214.226  ICMP     74 Echo (ping) request  id=0x5d83, seq=28/7168, ttl=10 (no response found!)
 43 7.340574616  192.168.56.138   139.224.214.226  ICMP     74 Echo (ping) request  id=0x5d83, seq=29/7424, ttl=10 (no response found!)
```

观察并记录响应包的信息，第一组响应包的发送者 **IP** 是：**192.168.56.2**，标记 **ICMP** 层的类型字段。



```
   13 2.335552793   192.168.56.2         192.168.56.138       ICMP         102 Time-to-live exceeded (
▸ Frame 13: 102 bytes on wire (816 bits), 102 bytes captured (816 bits) on interface 0
▸ Ethernet II, Src: Vmware_ee:15:07 (00:50:56:ee:15:07), Dst: Vmware_7f:d8:8a (00:0c:29:7f:d8:8a)
▸ Internet Protocol Version 4, Src: 192.168.56.2, Dst: 192.168.56.138
▾ Internet Control Message Protocol
    Type: 11 (Time-to-live exceeded)
    Code: 0 (Time to live exceeded in transit)
    Checksum: 0xf4ff [correct]
    [Checksum Status: Good]
  ▸ Internet Protocol Version 4, Src: 192.168.56.138, Dst: 139.224.214.226
  ▸ Internet Control Message Protocol
```

最后一组响应包的发送者 **IP** 是：**139.224.214.226**，标记 **ICMP** 层的类型字段。附上截图：



```
   79 12.366317426  139.224.214.226      192.168.56.138       ICMP          74 Echo (ping) reply
▸ Frame 79: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
▸ Ethernet II, Src: Vmware_ee:15:07 (00:50:56:ee:15:07), Dst: Vmware_7f:d8:8a (00:0c:29:7f:d8:8a)
▸ Internet Protocol Version 4, Src: 139.224.214.226, Dst: 192.168.56.138
▾ Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0
    Checksum: 0x2cc4 [correct]
    [Checksum Status: Good]
```

✧ 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。

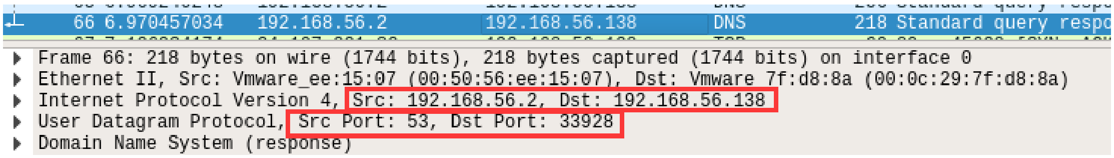
   捕获到的这些最高层的协议数据包分别由哪几层协议构成？

   DNS： 5 层： Frame, Ethernet, IPv4, UDP, DNS

   HTTP: 5 层：Frame, Ethernet, IPv4, TCP, HTTP

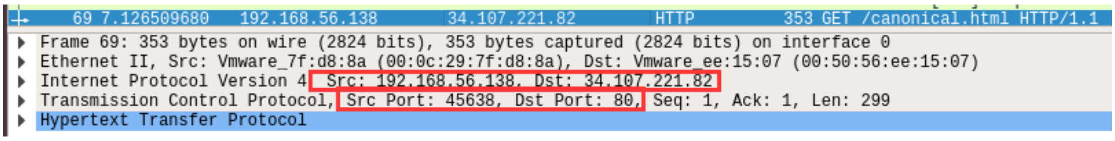
   每种协议选取一个代表展开后截图，并标出源和目标 **IP** 地址、源和目标端口）

DNS:



```
   66 6.970457034   192.168.56.2        192.168.56.138      DNS        218 Standard query respo
▸ Frame 66: 218 bytes on wire (1744 bits), 218 bytes captured (1744 bits) on interface 0
▸ Ethernet II, Src: Vmware_ee:15:07 (00:50:56:ee:15:07), Dst: Vmware_7f:d8:8a (00:0c:29:7f:d8:8a)
▸ Internet Protocol Version 4, Src: 192.168.56.2, Dst: 192.168.56.138
▸ User Datagram Protocol, Src Port: 53, Dst Port: 33928
▸ Domain Name System (response)
```

HTTP:



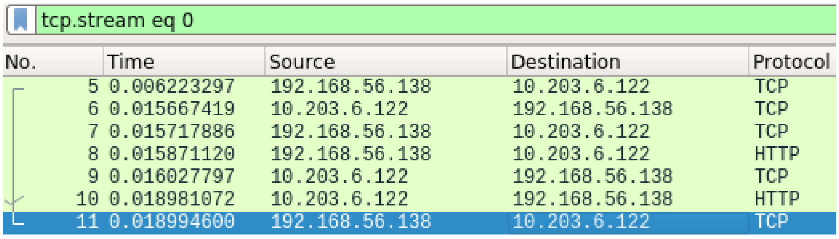
```
   69 7.126509680   192.168.56.138   34.107.221.82     HTTP      353 GET /canonical.html HTTP/1.1
▸ Frame 69: 353 bytes on wire (2824 bits), 353 bytes captured (2824 bits) on interface 0
▸ Ethernet II, Src: Vmware_7f:d8:8a (00:0c:29:7f:d8:8a), Dst: Vmware_ee:15:07 (00:50:56:ee:15:07)
▸ Internet Protocol Version 4, Src: 192.168.56.138, Dst: 34.107.221.82
▸ Transmission Control Protocol, Src Port: 45638, Dst Port: 80, Seq: 1, Ack: 1, Len: 299
▸ Hypertext Transfer Protocol
```

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； ocsp.dcocsp.cn； hm.baidu.com； tel.zju.edu.cn 等

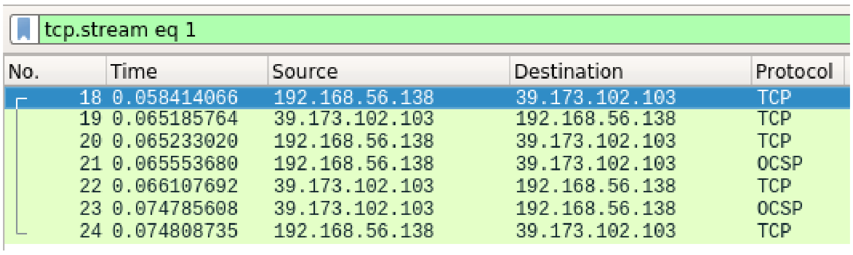
3. 使用显示过滤器 **tcp.stream eq X**，让 **X** 从 **0** 开始变化，直到没有数据。分析浏览器为了获取网页数据，总共建立了几个连接？（一个 **TCP** 流对应一个 **TCP** 连接）

   TCP 连接数： 5



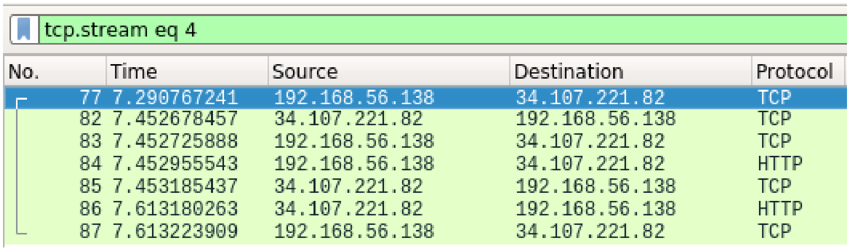
tcp.stream eq 0

No.	Time	Source	Destination	Protocol
5	0.006223297	192.168.56.138	10.203.6.122	TCP
6	0.015667419	10.203.6.122	192.168.56.138	TCP
7	0.015717886	192.168.56.138	10.203.6.122	TCP
8	0.015871120	192.168.56.138	10.203.6.122	HTTP
9	0.016027797	10.203.6.122	192.168.56.138	TCP
10	0.018981072	10.203.6.122	192.168.56.138	HTTP
11	0.018994600	192.168.56.138	10.203.6.122	TCP



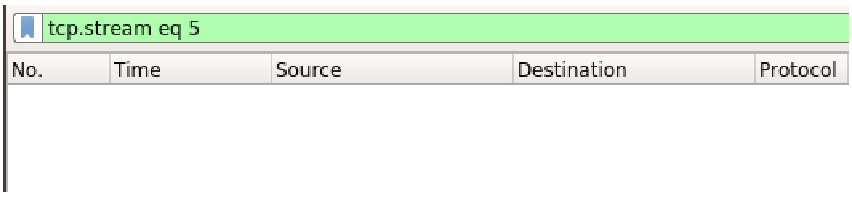
tcp.stream eq 1

No.	Time	Source	Destination	Protocol
18	0.058414066	192.168.56.138	39.173.102.103	TCP
19	0.065185764	39.173.102.103	192.168.56.138	TCP
20	0.065233020	192.168.56.138	39.173.102.103	TCP
21	0.065553680	192.168.56.138	39.173.102.103	OCSP
22	0.066107692	39.173.102.103	192.168.56.138	TCP
23	0.074785608	39.173.102.103	192.168.56.138	OCSP
24	0.074808735	192.168.56.138	39.173.102.103	TCP



tcp.stream eq 4

No.	Time	Source	Destination	Protocol
77	7.290767241	192.168.56.138	34.107.221.82	TCP
82	7.452678457	34.107.221.82	192.168.56.138	TCP
83	7.452725888	192.168.56.138	34.107.221.82	TCP
84	7.452955543	192.168.56.138	34.107.221.82	HTTP
85	7.453185437	34.107.221.82	192.168.56.138	TCP
86	7.613180263	34.107.221.82	192.168.56.138	HTTP
87	7.613223909	192.168.56.138	34.107.221.82	TCP

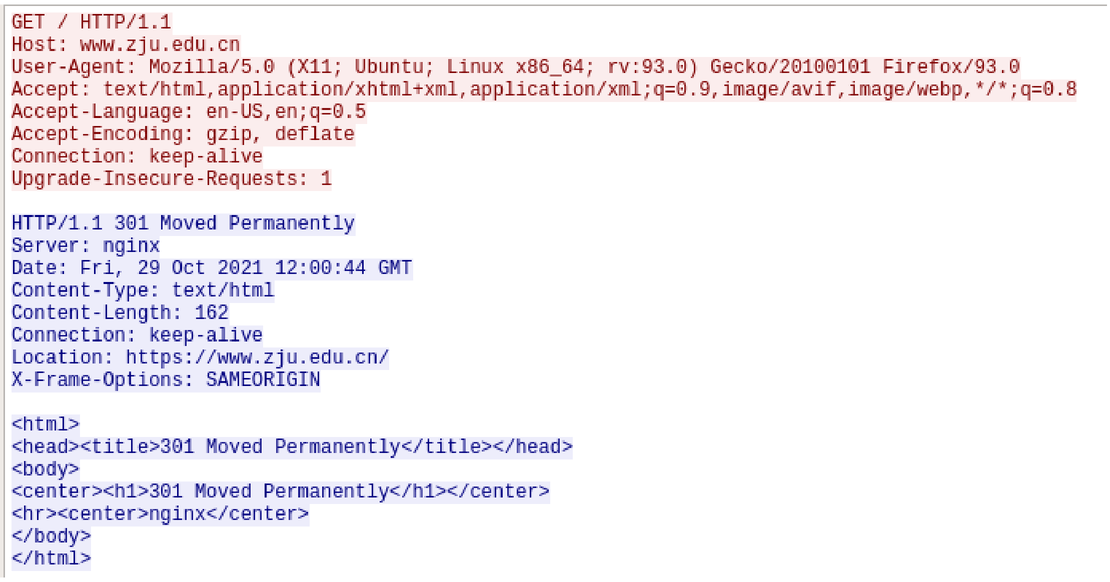


tcp.stream eq 5

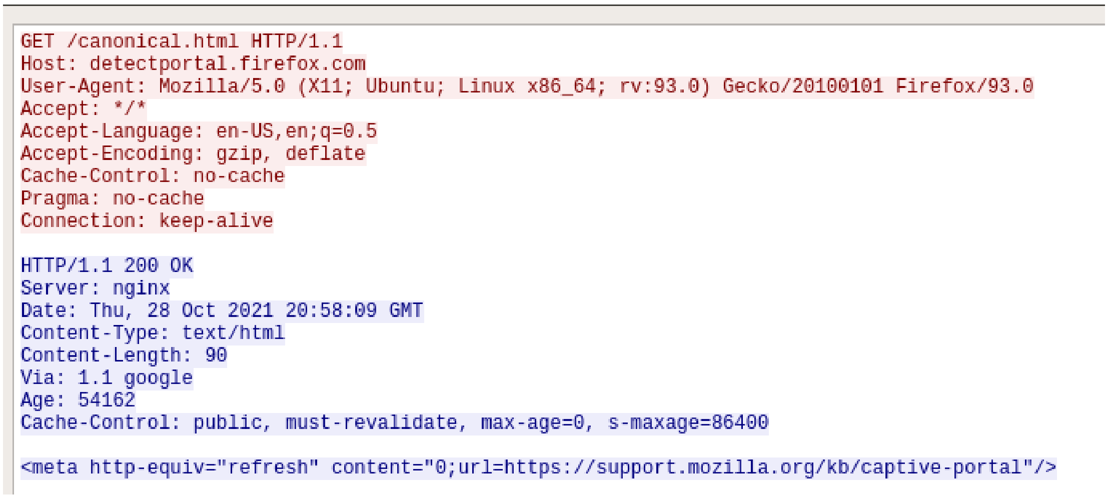
No.	Time	Source	Destination	Protocol

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。

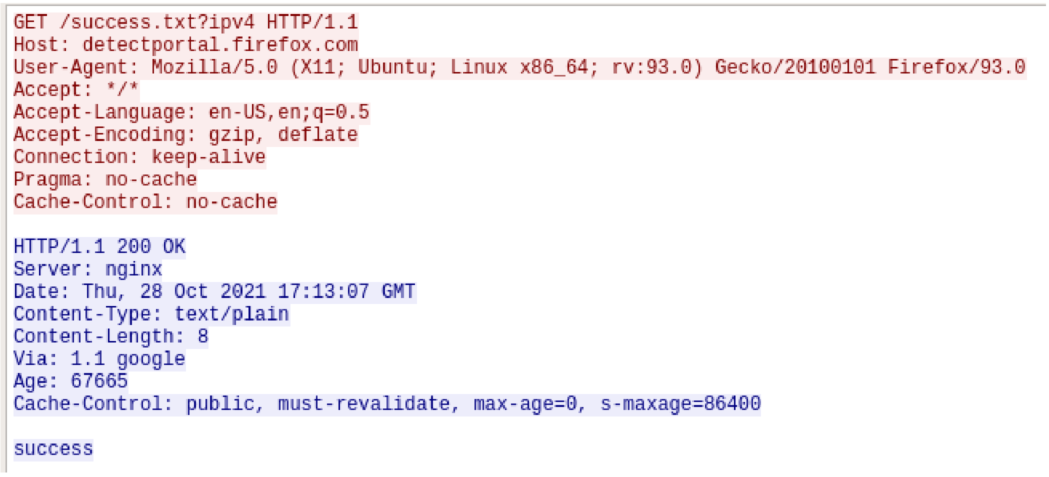
HTTP 会话数： 3



```
GET / HTTP/1.1
Host: www.zju.edu.cn
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:93.0) Gecko/20100101 Firefox/93.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1

HTTP/1.1 301 Moved Permanently
Server: nginx
Date: Fri, 29 Oct 2021 12:00:44 GMT
Content-Type: text/html
Content-Length: 162
Connection: keep-alive
Location: https://www.zju.edu.cn/
X-Frame-Options: SAMEORIGIN

<html>
<head><title>301 Moved Permanently</title></head>
<body>
<center><h1>301 Moved Permanently</h1></center>
<hr><center>nginx</center>
</body>
</html>
```

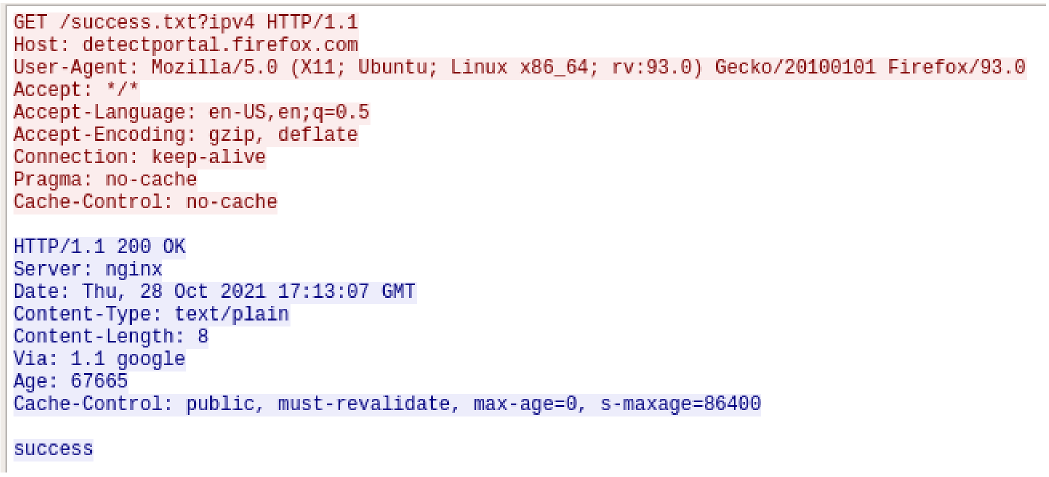


```
GET /canonical.html HTTP/1.1
Host: detectportal.firefox.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:93.0) Gecko/20100101 Firefox/93.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cache-Control: no-cache
Pragma: no-cache
Connection: keep-alive

HTTP/1.1 200 OK
Server: nginx
Date: Thu, 28 Oct 2021 20:58:09 GMT
Content-Type: text/html
Content-Length: 90
Via: 1.1 google
Age: 54162
Cache-Control: public, must-revalidate, max-age=0, s-maxage=86400

<meta http-equiv="refresh" content="0;url=https://support.mozilla.org/kb/captive-portal"/>
```



```
GET /success.txt?ipv4 HTTP/1.1
Host: detectportal.firefox.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:93.0) Gecko/20100101 Firefox/93.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache

HTTP/1.1 200 OK
Server: nginx
Date: Thu, 28 Oct 2021 17:13:07 GMT
Content-Type: text/plain
Content-Length: 8
Via: 1.1 google
Age: 67665
Cache-Control: public, must-revalidate, max-age=0, s-maxage=86400

success
```

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）：**



```
GET /success.txt?ipv4 HTTP/1.1
Host: detectportal.firefox.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:93.0) Gecko/20100101 Firefox/93.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache

HTTP/1.1 200 OK
Server: nginx
Date: Thu, 28 Oct 2021 17:13:07 GMT
Content-Type: text/plain
Content-Length: 8
Via: 1.1 google
Age: 67665
Cache-Control: public, must-revalidate, max-age=0, s-maxage=86400

success
```

红色为请求部分，蓝色为相应部分。

（所有 TCP 流都最多只有 1 个 HTTP 会话）

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？

tcp port 80 && host [IP 地址]

（用需要的 IP 地址替换上面的 [IP 地址]）

- 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。

当计算机没有缓存请求的 IP 地址的物理地址时会请求地址解析，出现 ARP 数据包。

Ping 一个域名时会先用 DNS 解析为 IP 地址，因此会额外出现 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？

使用 -I 后发送的是 ICMP 数据包。首先发送 TTL 为 1 的数据包，重复三次；每个数据包转发 1 次后在下一个路由超时，从而返回一个 ICMP 超时信息，从这个信息中可以跟踪到第 1 个路由器；然后发送 TTL 为 2 的数据包，在第 2 个路由器超时，因此可以跟踪到第 2 个路由；以此类推。

- 如何理解 TCP 连接和 HTTP 会话？他们之间存在什么关系？

TCP 连接负责规定两台设备之间数据的传输方式，即 A 设备如何才能快速、稳定地将需要传输的内容发送给 B 设备。HTTP 规定数据的格式，也就是定义一套对有效数据的封装来使得每一台设备都能理解收到的数据包。也即是计算机遵照 HTTP 协议建立会话，封装数据，再通过 TCP 协议进行连接以传输这些数据。

- 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？

从开销和性能上比较，UDP 协议具有较少的额外开销，性能更好；而 TCP 有三次握手等更多的额外开销。但是如果数据被分成若干个包发送，那么 TCP 则更为可靠。

DNS 的数据包一般较小，不需要分成若干个包，因此使用 UDP 发送可以保证较少的额外开销；出现错误时进行重发即可；而 HTTP 的数据一般较大，对于较大的、被分成多个包的数据，TCP 的额外开销并不会带来很大影响，而且可以提高数据传送的可靠性。因此 DNS 更适合 UDP 协议，而 HTTP 更适合 TCP 协议。

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：

- 捕获过滤器中设置 tcp port xx 的端口是什么意思，有什么筛选作用？

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：

- Ubuntu 下的很多指令（如 traceroute 和清除 DNS 缓存等）都与实验指导上所给的不同；需要多去查阅资料进行了解。

你对本实验安排有哪些更好的建议呢？欢迎献计献策：

- 通过本实验，我对 wireshark 的使用有了多方面的了解；同时也对各种协议的结构和功能有了初步的认识。
- 本实验可以考虑放在第 1 个实验的位置，同时稍微添加一些讲解和描述，可以帮助同学们对计算机网络有一个预备性的认识，也可以掌握 wireshark 的使用。