

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：	夏豪诚
学 院：	计算机学院
系：	信息安全
专 业：	信息安全
学 号：	3170102492
指导教师：	邱劲松

2019 年 9 月 21 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

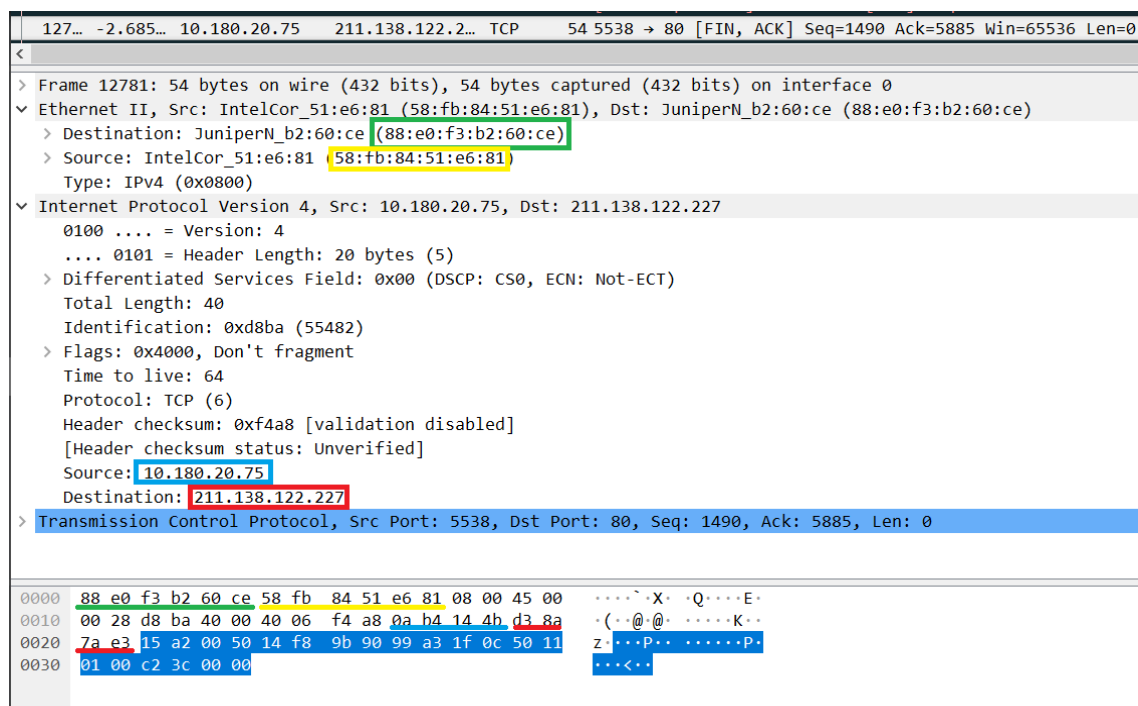
1. 运行 Wireshark 软件，开始捕获数据包，列出你看到的协议名字（至少 5 个）。

协议名： TCP、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。

截图：

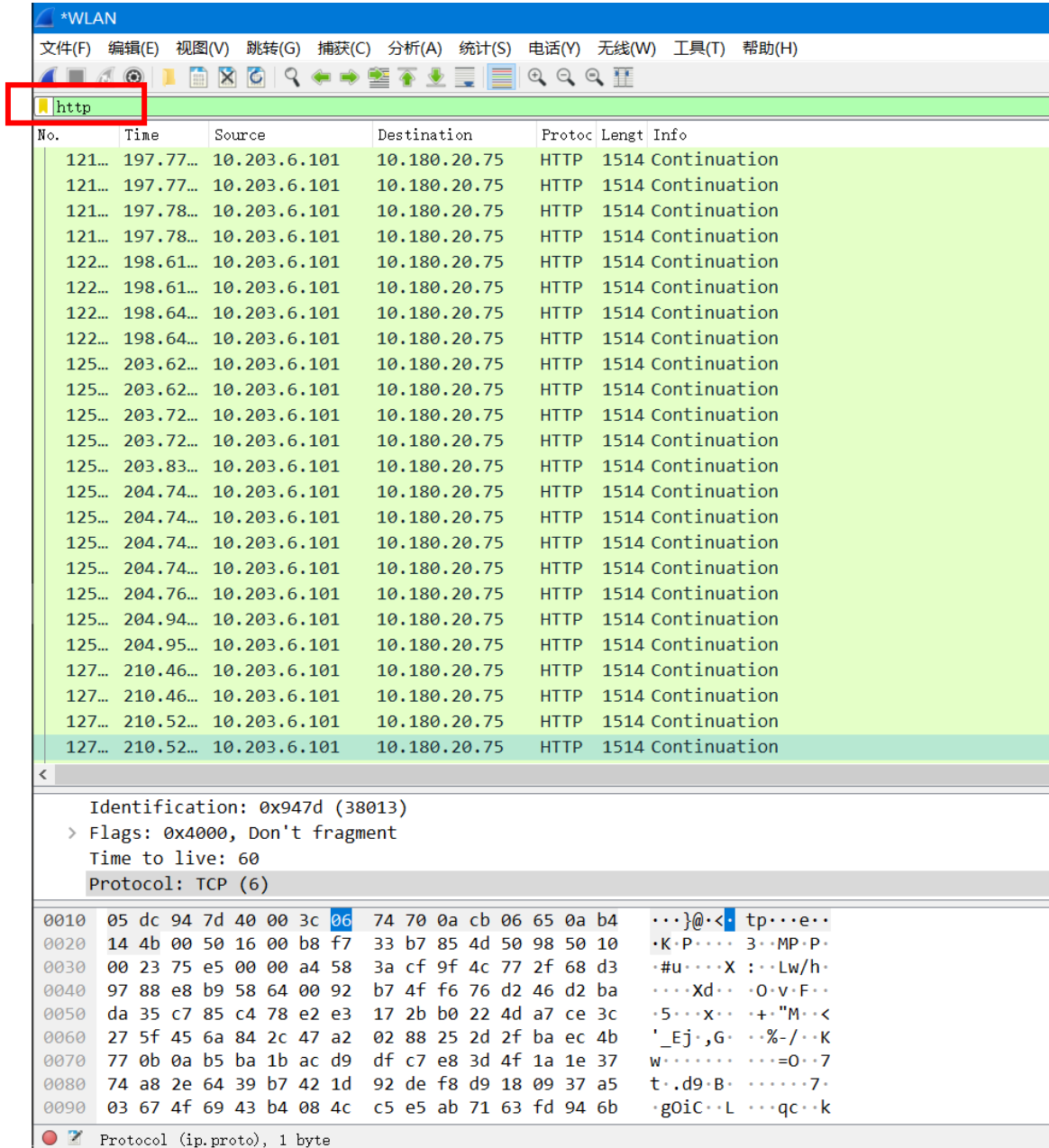


*绿色方框内容对应目标 MAC 地址，绿色横线标出了在数据包中的位置；黄色方框内容对应源 MAC 地址，黄色横线标出了在数据包中的位置；蓝色方框内容对应源 IP 地址，蓝色横线标出了在数据包中的位置；红色方框内容为对应目标 IP 地址，红色横线标出了在数据包中的位置。

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： http ，希望显示的协议类型： HTTP 。

截图：



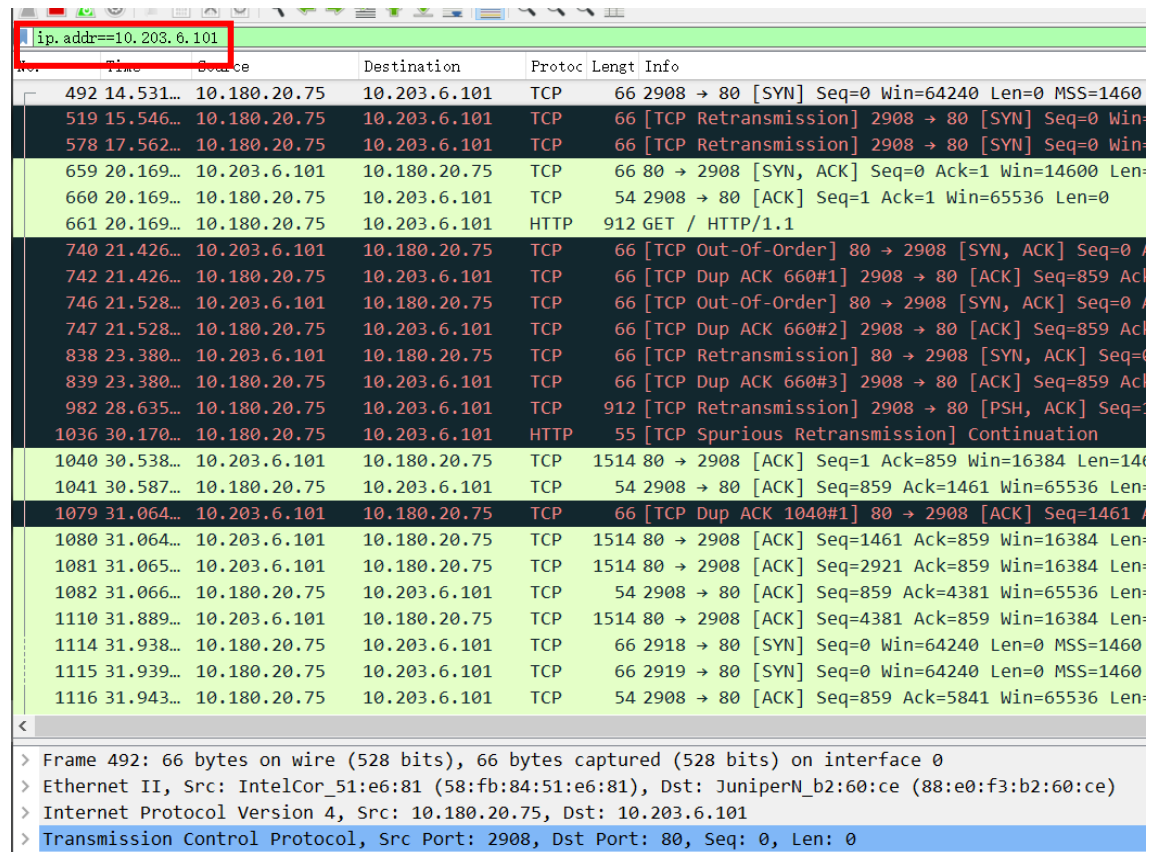
*红色方框标出了使用的过滤器。

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 == 10.203.6.101 ，希望显示的 IP 地址：

10.203.6.101。

截图：



No.	Time	Source	Destination	Protoc	Length	Info
492	14.531...	10.180.20.75	10.203.6.101	TCP	66	2908 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
519	15.546...	10.180.20.75	10.203.6.101	TCP	66	[TCP Retransmission] 2908 → 80 [SYN] Seq=0 Win=
578	17.562...	10.180.20.75	10.203.6.101	TCP	66	[TCP Retransmission] 2908 → 80 [SYN] Seq=0 Win=
659	20.169...	10.203.6.101	10.180.20.75	TCP	66	80 → 2908 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=
660	20.169...	10.180.20.75	10.203.6.101	TCP	54	2908 → 80 [ACK] Seq=1 Ack=1 Win=65536 Len=0
661	20.169...	10.180.20.75	10.203.6.101	HTTP	912	GET / HTTP/1.1
740	21.426...	10.203.6.101	10.180.20.75	TCP	66	[TCP Out-Of-Order] 80 → 2908 [SYN, ACK] Seq=0 A
742	21.426...	10.180.20.75	10.203.6.101	TCP	66	[TCP Dup ACK 660#1] 2908 → 80 [ACK] Seq=859 Ack
746	21.528...	10.203.6.101	10.180.20.75	TCP	66	[TCP Out-Of-Order] 80 → 2908 [SYN, ACK] Seq=0 A
747	21.528...	10.180.20.75	10.203.6.101	TCP	66	[TCP Dup ACK 660#2] 2908 → 80 [ACK] Seq=859 Ack
838	23.380...	10.203.6.101	10.180.20.75	TCP	66	[TCP Retransmission] 80 → 2908 [SYN, ACK] Seq=0
839	23.380...	10.180.20.75	10.203.6.101	TCP	66	[TCP Dup ACK 660#3] 2908 → 80 [ACK] Seq=859 Ack
982	28.635...	10.180.20.75	10.203.6.101	TCP	912	[TCP Retransmission] 2908 → 80 [PSH, ACK] Seq=
1036	30.170...	10.180.20.75	10.203.6.101	HTTP	55	[TCP Spurious Retransmission] Continuation
1040	30.538...	10.203.6.101	10.180.20.75	TCP	1514	80 → 2908 [ACK] Seq=1 Ack=859 Win=16384 Len=140
1041	30.587...	10.180.20.75	10.203.6.101	TCP	54	2908 → 80 [ACK] Seq=859 Ack=1461 Win=65536 Len=
1079	31.064...	10.203.6.101	10.180.20.75	TCP	66	[TCP Dup ACK 1040#1] 80 → 2908 [ACK] Seq=1461 A
1080	31.064...	10.203.6.101	10.180.20.75	TCP	1514	80 → 2908 [ACK] Seq=1461 Ack=859 Win=16384 Len=
1081	31.065...	10.203.6.101	10.180.20.75	TCP	1514	80 → 2908 [ACK] Seq=2921 Ack=859 Win=16384 Len=
1082	31.066...	10.180.20.75	10.203.6.101	TCP	54	2908 → 80 [ACK] Seq=859 Ack=4381 Win=65536 Len=
1110	31.889...	10.203.6.101	10.180.20.75	TCP	1514	80 → 2908 [ACK] Seq=4381 Ack=859 Win=16384 Len=
1114	31.938...	10.180.20.75	10.203.6.101	TCP	66	2918 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
1115	31.939...	10.180.20.75	10.203.6.101	TCP	66	2919 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
1116	31.943...	10.180.20.75	10.203.6.101	TCP	54	2908 → 80 [ACK] Seq=859 Ack=5841 Win=65536 Len=

<

> Frame 492: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
> Ethernet II, Src: IntelCor_51:e6:81 (58:fb:84:51:e6:81), Dst: JuniperN_b2:60:ce (88:e0:f3:b2:60:ce)
> Internet Protocol Version 4, Src: 10.180.20.75, Dst: 10.203.6.101
> Transmission Control Protocol, Src Port: 2908, Dst Port: 80, Seq: 0, Len: 0

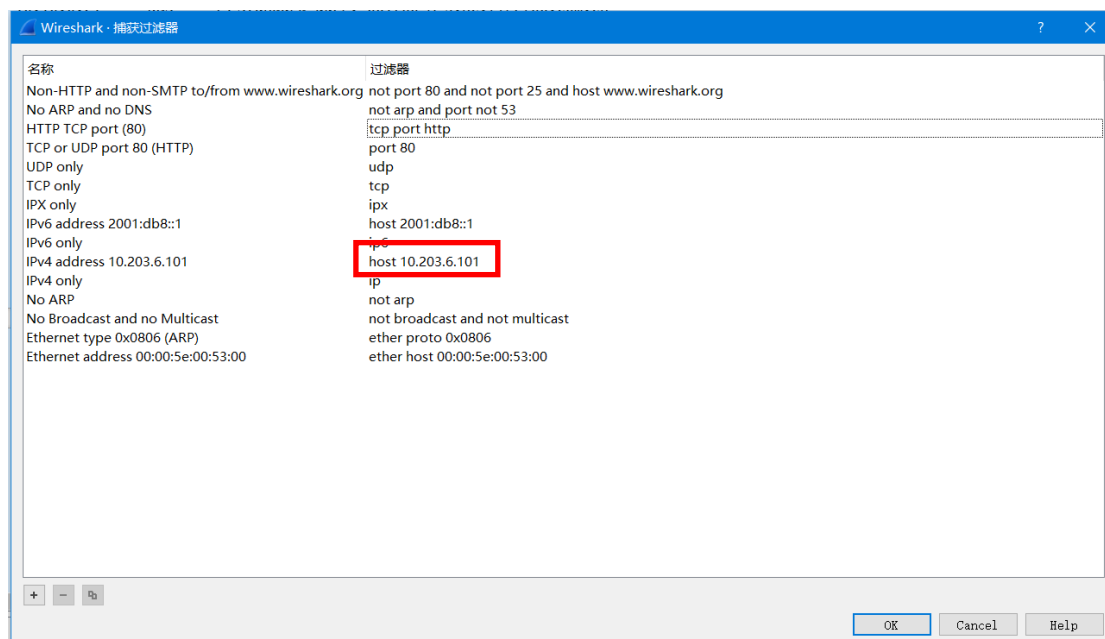
*红色方框标出了使用的过滤器。

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 10.203.6.101 ，希望捕获的 IP 地址：

10.203.6.101。

截图：

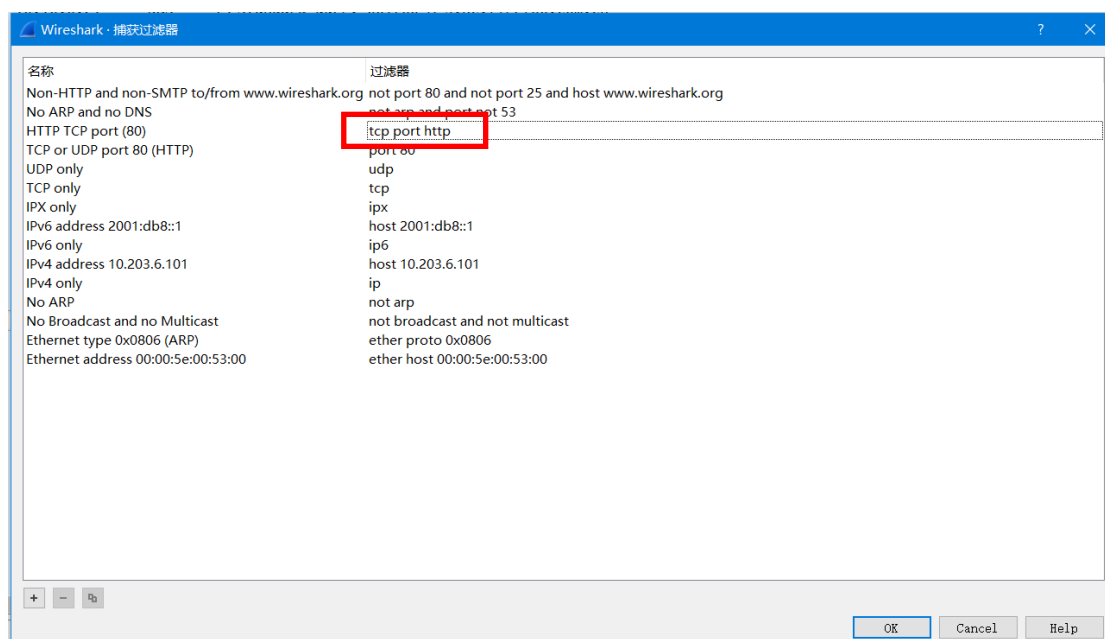


*红色方框标出了使用的过滤器。

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：



*红色方框标出了使用的过滤器。

✧ 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, Network layer: IPv4, Transport layer: UDP, Application layer: 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:

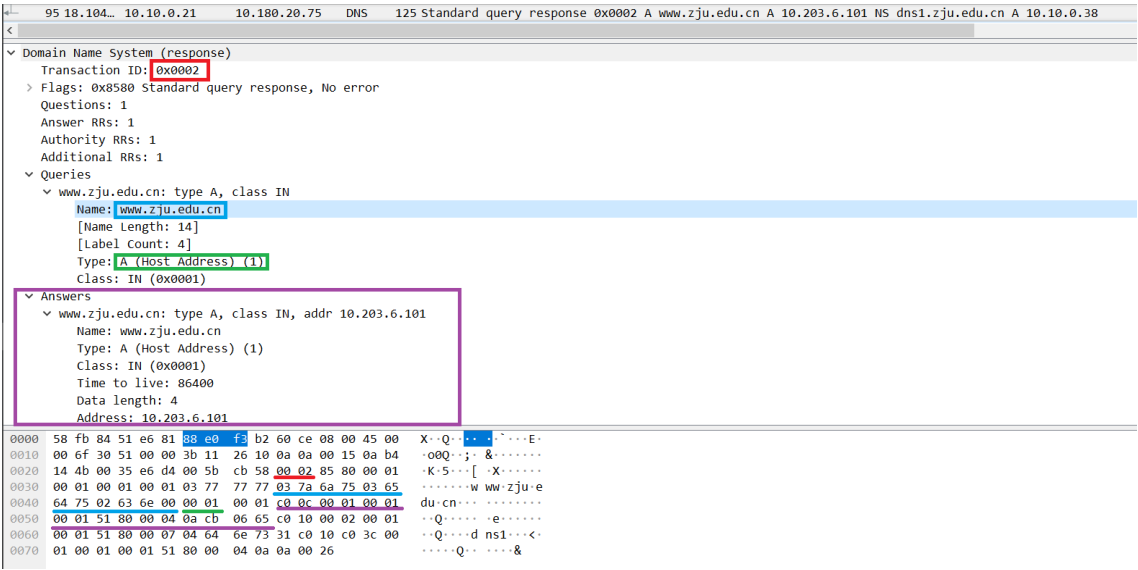
94 18.102... 10.180.20.75 10.10.0.21 DNS 74 Standard query 0x0002 A www.zju.edu.cn

> Frame 94: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
> Ethernet II, Src: IntelCor_51:e6:81 (58:fb:84:51:e6:81), Dst: JuniperN_b2:60:ce (88:e0:f3:b2:60:ce)
> Internet Protocol Version 4, Src: 10.180.20.75, Dst: 10.10.0.21
> User Datagram Protocol, Src Port: 59092, Dst Port: 53
▼ Domain Name System (query)
Transaction ID: 0x0002
> Flags: 0x0100 Standard query
Questions: 1
Answer RRs: 0
Authority RRs: 0
Additional RRs: 0
▼ Queries
▼ www.zju.edu.cn: type A, class IN
Name: www.zju.edu.cn
[Name Length: 14]
[Label Count: 4]
Type: A (Host Address) (1)
Class: IN (0x0001)
[\[Response In: 95\]](#)

0000	88 e0 f3 b2 60 ce 58 fb 84 51 e6 81 08 00 45 00X. Q...E.
0010	00 3c 99 97 00 00 40 11 b7 fc 0a b4 14 4b 0a 0a	<...@.K..
0020	00 15 e6 d4 00 35 00 28 2d 56 00 02 01 00 00 01	...5.(-V.....
0030	00 00 00 00 00 00 03 77 77 77 03 7a 6a 75 03 65w ww.zju.e
0040	64 75 02 63 6e 00 00 01 00 01	du.cn. . .

*红色方框内容为交易 ID, 红色横线标出了在数据包中的位置; 蓝色方框内容为查询的域名, 蓝色横线标出了在数据包中的位置; 绿色方框内容为查询类型, 绿色横线标出了在数据包中的位置。

响应包：



*红色方框内容为交易 ID，红色横线标出了在数据包中的位置；蓝色方框内容为查询的域名，蓝色横线标出了在数据包中的位置；绿色方框内容为查询类型，绿色横线标出了在数据包中的位置；紫色方框标出了查询得到的内容，紫色横线标出了其在数据包中的内容。

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？ Date link layer: Ethernet II, Network layer: 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：

arp						
No.	Time	Source	Destination	Protocol	Length	Info
10	12.006...	JuniperN_67:2...	Broadcast	ARP	56	Who has 10.180.124.110? Tell 10.0.2.2
11	12.006...	JuniperN_67:2...	Broadcast	ARP	56	Who has 10.180.124.110? Tell 10.0.2.2
25	24.960...	IntelCor_51:e...	JuniperN_b2:6...	ARP	42	Who has 10.180.16.1? Tell 10.180.20.75
<						
> Frame 25: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0						
> Ethernet II, Src: IntelCor_51:e6:81 (58:fb:84:51:e6:81), Dst: JuniperN_b2:60:ce (88:e0:f3:b2:60:ce)						
v Address Resolution Protocol (request)						
Hardware type: Ethernet (1)						
Protocol type: IPv4 (0x0800)						
Hardware size: 6						
Protocol size: 4						
Opcode: request (1)						
Sender MAC address: IntelCor_51:e6:81 (58:fb:84:51:e6:81)						
Sender IP address: 10.180.20.75						
Target MAC address: JuniperN_b2:60:ce (88:e0:f3:b2:60:ce)						
Target IP address: 10.180.16.1						
0000 88 e0 f3 b2 60 ce 58 fb 84 51 e6 81 08 06 00 01X. .Q.....						
0010 08 00 06 04 00 01 58 fb 84 51 e6 81 0a b4 14 4bX. .Q.....K						
0020 88 e0 f3 b2 60 ce 0a b4 10 01						

*棕色方框中的内容为 Ethernet 层的目标 MAC 地址，棕色横线标出了在数据包中的位置；粉色方框中的内容为操作码，粉色横线标出了在数据包中的位置；蓝色方框中的内容为发送者 MAC 地址，蓝色横线标出了在数据包中的位置；红色方框标出了查询目标的 IP 地址，红色横线标出了在数据包中的位置。

响应包：

26 24.966... JuniperN_b2:6... IntelCor_51:e... ARP 56 10.180.16.1 is at 88:e0:f3:b2:60:ce

<

> Frame 26: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface 0
> Ethernet II, Src: JuniperN_b2:60:ce (88:e0:f3:b2:60:ce), Dst: IntelCor_51:e6:81 (58:fb:84:51:e6:81)
v Address Resolution Protocol (reply)
Hardware type: Ethernet (1)
Protocol type: IPv4 (0x0800)
Hardware size: 6
Protocol size: 4
Opcode: reply (2)
Sender MAC address: JuniperN_b2:60:ce (88:e0:f3:b2:60:ce)
Sender IP address: 10.180.16.1
Target MAC address: IntelCor_51:e6:81 (58:fb:84:51:e6:81)
Target IP address: 10.180.20.75

0000	58 fb 84 51 e6 81 88 e0 f3 b2 60 ce 08 06 00 01	X..Q.....
0010	08 00 06 04 00 02 88 e0 f3 b2 60 ce 0a b4 10 01@...K..
0020	58 fb 84 51 e6 81 0a b4 14 4b 00 00 00 00 00	X..Q....K.....
0030	00 00 00 00 00 00 00 00

*紫色方框标出了查询结果。

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：

18 21.437... 10.180.20.75 10.203.6.101 ICMP 74 Echo (ping) request id=0x0001, seq=11/2816, ttl=64 (reply in 19)

<

> Frame 18: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
> Ethernet II, Src: IntelCor_51:e6:81 (58:fb:84:51:e6:81), Dst: JuniperN_b2:60:ce (88:e0:f3:b2:60:ce)
> Internet Protocol Version 4, Src: 10.180.20.75, Dst: 10.203.6.101
v Internet Control Message Protocol
Type: 8 (Echo (ping) request)
Code: 0
Checksum: 0x4d50 [correct]
[Checksum Status: Good]
Identifier (BE): 1 (0x0001)
Identifier (LE): 256 (0x0100)
Sequence number (BE): 11 (0x000b)
Sequence number (LE): 2816 (0x0b00)
[Response frame: 19]
v Data (32 bytes)
Data: 6162636465666768696a6b6c6d6e6f707172737475767761...
[Length: 32]

0000	88 e0 f3 b2 60 ce 58 fb 84 51 e6 81 08 00 45 00X..Q....E..
0010	00 3c 8a 72 00 00 40 01 c0 20 0a b4 14 4b 0a cb	...r..@...K..
0020	06 65 08 00 4d 50 00 01 00 0b 61 62 63 64 65 66	..e..MP...abcdef
0030	67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76	ghijklmn opqrstuv
0040	77 61 62 63 64 65 66 67 68 69	wabdefg hi

*粉色方框内容为类型，粉色横线标出了在数据包中的位置；蓝色方框内容为序号，蓝色横线标出了在数据包中的位置。

响应包：

19.21.439...	10.203.6.101	10.180.20.75	ICMP	74 Echo (ping) reply	id=0x0001, seq=11/2816, ttl=60 (request in 18)
<					
> Frame 19: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0					
> Ethernet II, Src: JuniperN_b2:60:ce (88:e0:f3:b2:60:ce), Dst: IntelCor_51:e6:81 (58:fb:84:51:e6:81)					
> Internet Protocol Version 4, Src: 10.203.6.101, Dst: 10.180.20.75					
v Internet Control Message Protocol					
Type: 0 (Echo (ping) reply)					
Code: 0					
Checksum: 0x5550 [correct]					
[Checksum Status: Good]					
Identifier (BE): 1 (0x0001)					
Identifier (LE): 256 (0x0100)					
Sequence number (BE): 11 (0x000b)					
Sequence number (LE): 2816 (0x0b00)					
[Request frame: 18]					
[Response time: 1.271 ms]					
v Data (32 bytes)					
Data: 6162636465666768696a6b6c6d6e6f707172737475767761...					
[Length: 32]					
0000	58 fb 84 51 e6 81 88 e0	f3 b2 60 ce 08 00 45 00	X..Q....	..E..	
0010	00 3c d2 10 00 00 3c 01	7c 82 0a cb 06 65 0a b4	.<....<e..	
0020	14 4b 00 00 55 50 00 01	00 0b 61 62 63 64 65 66	.K..UP..	..abcdef	
0030	67 68 69 6a 6b 6c 6d 6e	6f 70 71 72 73 74 75 76	ghijklmn	opqrstuv	
0040	77 61 62 63 64 65 66 67	68 69	wabdefgh	hi	

*粉色方框内容为类型，粉色横线标出了在数据包中的位置；蓝色方框内容为序号，蓝色横线标出了在数据包中的位置。

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 。

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 5 。附上截图：

No.	icmpv6	Source	Destination	Protoc	Length	Info
11	13.189.	10.180.20.75	10.203.6.101	ICMP	106	Echo (ping) request id=0x0001, seq=32/8192, ttl=1 (no response found!)
12	13.189.	10.180.20.75	10.203.6.101	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
13	13.199.	10.180.20.75	10.203.6.101	ICMP	106	Echo (ping) request id=0x0001, seq=33/8448, ttl=1 (no response found!)
14	13.201.	10.180.20.75	10.203.6.101	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
15	13.203.	10.180.20.75	10.203.6.101	ICMP	106	Echo (ping) request id=0x0001, seq=34/8704, ttl=1 (no response found!)
16	13.205.	10.180.20.75	10.203.6.101	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
33	23.238.	10.180.20.75	10.203.6.101	ICMP	106	Echo (ping) request id=0x0001, seq=35/8960, ttl=2 (no response found!)
34	23.242.	10.3.1.18	10.180.20.75	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
35	23.244.	10.180.20.75	10.203.6.101	ICMP	106	Echo (ping) request id=0x0001, seq=36/9216, ttl=2 (no response found!)
36	23.246.	10.3.1.18	10.180.20.75	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
37	23.252.	10.180.20.75	10.203.6.101	ICMP	106	Echo (ping) request id=0x0001, seq=37/9472, ttl=2 (no response found!)
38	23.253.	10.3.1.18	10.180.20.75	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
57	33.270.	10.180.20.75	10.203.6.101	ICMP	106	Echo (ping) request id=0x0001, seq=38/9728, ttl=3 (no response found!)
58	33.290.	10.3.7.78	10.180.20.75	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
59	33.293.	10.180.20.75	10.203.6.101	ICMP	106	Echo (ping) request id=0x0001, seq=39/9984, ttl=3 (no response found!)
60	33.294.	10.3.7.78	10.180.20.75	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
61	33.296.	10.180.20.75	10.203.6.101	ICMP	106	Echo (ping) request id=0x0001, seq=40/10240, ttl=3 (no response found!)
62	33.298.	10.3.7.78	10.180.20.75	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
73	43.336.	10.180.20.75	10.203.6.101	ICMP	106	Echo (ping) request id=0x0001, seq=41/10496, ttl=4 (no response found!)
74	43.353.	10.3.7.226	10.180.20.75	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
75	43.356.	10.180.20.75	10.203.6.101	ICMP	106	Echo (ping) request id=0x0001, seq=42/10752, ttl=4 (no response found!)
76	43.359.	10.3.7.226	10.180.20.75	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
77	43.361.	10.180.20.75	10.203.6.101	ICMP	106	Echo (ping) request id=0x0001, seq=43/11008, ttl=4 (no response found!)
78	43.362.	10.3.7.226	10.180.20.75	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
110	53.300.	10.180.20.75	10.203.6.101	ICMP	106	Echo (ping) request id=0x0001, seq=44/11264, ttl=5 (reply in 111)
111	53.384.	10.203.6.101	10.180.20.75	ICMP	106	Echo (ping) reply id=0x0001, seq=44/11264, ttl=60 (request in 110)
112	53.385.	10.180.20.75	10.203.6.101	ICMP	106	Echo (ping) request id=0x0001, seq=45/11520, ttl=5 (reply in 113)
113	53.389.	10.203.6.101	10.180.20.75	ICMP	106	Echo (ping) reply id=0x0001, seq=45/11520, ttl=60 (request in 112)
114	53.390.	10.180.20.75	10.203.6.101	ICMP	106	Echo (ping) request id=0x0001, seq=46/11776, ttl=5 (reply in 115)
115	53.392.	10.203.6.101	10.180.20.75	ICMP	106	Echo (ping) reply id=0x0001, seq=46/11776, ttl=60 (request in 114)

观察并记录响应包的信息，第一组响应包的发送者 IP 是：10.180.16.1，标记 ICMP 层的类型字段。最后一组响应包的发送者 IP 是：10.203.6.101，标记 ICMP 层的类型字段。附上截图：

第一组：

```
14 13.201... 10.180.16.1 10.180.20.75 ICMP 70 Time-to-live exceeded (Time to live exceeded in transit)
<
> Frame 14: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
> Ethernet II, Src: JuniperN_b2:60:ce (88:e0:f3:b2:60:ce), Dst: IntelCor_51:e6:81 (58:fb:84:51:e6:81)
> Internet Protocol Version 4, Src: 10.180.16.1, Dst: 10.180.20.75
> 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: 10.180.20.75, Dst: 10.203.6.101
> Internet Control Message Protocol
  Type: 8 (Echo (ping) request)
  Code: 0
  Checksum: 0xf7dd [unverified] [in ICMP error packet]
  [Checksum Status: Unverified]
  Identifier (BE): 1 (0x0001)
  Identifier (LE): 256 (0x0100)
  Sequence number (BE): 33 (0x0021)
  Sequence number (LE): 8448 (0x2100)
```

*红色方框标记了发送者 IP 和 ICMP 层类型字段

最后一组：

```
115 53.392... 10.203.6.101 10.180.20.75 ICMP 106 Echo (ping) reply id=0x0001, seq=46/11776, ttl=60 (request in 114)
<
> Frame 115: 106 bytes on wire (848 bits), 106 bytes captured (848 bits) on interface 0
> Ethernet II, Src: JuniperN_b2:60:ce (88:e0:f3:b2:60:ce), Dst: IntelCor_51:e6:81 (58:fb:84:51:e6:81)
> Internet Protocol Version 4, Src: 10.203.6.101, Dst: 10.180.20.75
> Internet Control Message Protocol
  Type: 0 (Echo (ping) reply)
  Code: 0
  Checksum: 0xffd0 [correct]
  [Checksum Status: Good]
  Identifier (BE): 1 (0x0001)
  Identifier (LE): 256 (0x0100)
  Sequence number (BE): 46 (0x002e)
  Sequence number (LE): 11776 (0x2e00)
  [Request frame: 114]
  [Response time: 1.853 ms]
> Data (64 bytes)
```

*红色方框标记了发送者 IP 和 ICMP 层类型字段

◇ 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, Network layer: IPv4, Transport layer:

UDP, Application layer: DNS

HTTP: Data link layer: Ethernet II, Network layer: 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）

DNS:

No.	Time	Source	Destination	Protocol	Length	Info
247	18.564...	10.180.20.75	10.10.0.21	DNS	74	Standard query 0xeeb5 A www.zju.edu.cn
264	18.566	10.10.0.21	10.180.20.75	DNS	175	Standard query response 0xeeb5 A www.zju.edu.cn A 10.203.6.101 NS dns1.zju.edu.cn A 10.10.0.38

> Frame 247: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
> Ethernet II, Src: IntelCor_51:e6:81 (58:fb:84:51:e6:81), Dst: JuniperN_b2:60:ce (88:e0:f3:b2:60:ce)
> Internet Protocol Version 4, Src: 10.180.20.75, Dst: 10.10.0.21
> User Datagram Protocol, Src Port: 53715, Dst Port: 53
> Domain Name System (query)

*红色方框标记了发送者 IP 和 ICMP 层类型字段

HTTP:

49	3.9333...	10.180.20.75	10.203.6.101	HTTP	923	GET / HTTP/1.1
62	3.9607	10.203.6.101	10.180.20.75	HTTP	327	HTTP/1.1 200 OK (text/html)

> Frame 49: 923 bytes on wire (7384 bits), 923 bytes captured (7384 bits) on interface 0
> Ethernet II, Src: IntelCor_51:e6:81 (58:fb:84:51:e6:81), Dst: JuniperN_b2:60:ce (88:e0:f3:b2:60:ce)
> Internet Protocol Version 4, Src: 10.180.20.75, Dst: 10.203.6.101
> Transmission Control Protocol, Src Port: 3080, Dst Port: 80, Seq: 1, Ack: 1, Len: 869
> Hypertext Transfer Protocol

*红色方框标记了发送者 IP 和 ICMP 层类型字段

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; map.zju.edu.cn; my.zju.edu.cn; bksy.zju.edu.cn;
www.news.zju.edu.cn; www.acv.zju.edu.cn; person.zju.edu.cn; ugrs.zju.edu.cn;
zdzsc.zju.edu.cn; iczu.zju.edu.cn; grs.zju.edu.cn; www.ce.zju.edu.cn;
admission.zju.edu.cn; ocw.zju.edu.cn; rwsz.zju.edu.cn; www.ce1.zju.edu.cn;
rd.zju.edu.cn; www.itri.zju.edu.cn; xncfzyjy.zju.edu.cn; www.journals.zju.edu.cn;
www.career.zju.edu.cn; www.zdxqn.edu.cn; culture.zju.edu.cn; www.youth.zju.edu.cn;
zdxz.cuepa.cn; www.tyys.zju.edu.cn; www.qsc.zju.edu.cn; piclb.zju.edu.cn;
zuit.zju.edu.cn; zuaa.zju.edu.cn; libwe.zju.edu.cn; ac.zju.edu.cn; helps.zju.edu.cn;
wefw.zju.edu.cn; word.zju.edu.cn; xxgk.zju.edu.cn; courses.zju.edu.cn;
zuef.zju.edu.cn; www.beian.gov.cn; weibo.com; www.innovation2030.zju.edu.cn;
www.toutiao.com; mp.weixin.qq.com

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：47

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）：

```
GET /_visitcount?siteid=3&type=1&columnid=5 HTTP/1.1
Host: www.zju.edu.cn
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:69.0) Gecko/20100101 Firefox/69.0
Accept: image/webp,*/*
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://www.zju.edu.cn/
Cookie: gr_user_id=06a7f3bb-7026-4104-bfc2-6dcabaddf8aa; grung_uid=b558a5ca-35c2-4b95-bba0-2197968d1ca8; fp_ver=4.7.5; BSF_IT_EXPIRATION=1560211872600; BSF_IT_DEVICEID=TP0027y8rZlwESK9f1s4Kw_3062uYzhu3f1u7Y9YK; kgkFCm5y9f8mkyu81QTP1s4gJpxv1jwz3u9rd1Vhdy8EPT_K5cgvQJj1anzak50JMS68d1Cku55JNP6r5pm11CVvC8_11x0KPH13py4Q1u5; hm_lvt_fe30b0c1ee45421ec1679d18d8f8453-1569161234,1569162565,1569200902

HTTP/1.1 200 OK
Date: Mon, 23 Sep 2019 04:03:57 GMT
Server: Apache
X-Frame-Options: SAMEORIGIN
Frame-Options: SAMEORIGIN
Content-Length: 0
Set-Cookie: 3545510NID=FCE597EE9EDF2691384308C737380; Path=/

GET /_upload/tp1/00/14/20/template20/images/news_top_bg.png HTTP/1.1
Host: www.zju.edu.cn
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:69.0) Gecko/20100101 Firefox/69.0
Accept: image/webp,*/*
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://www.zju.edu.cn/_upload/tp1/00/14/20/template20/style/default.css
Cookie: gr_user_id=06a7f3bb-7026-4104-bfc2-6dcabaddf8aa; grung_uid=b558a5ca-35c2-4b95-bba0-2197968d1ca8; fp_ver=4.7.5; BSF_IT_EXPIRATION=1560211872600; BSF_IT_DEVICEID=TP0027y8rZlwESK9f1s4Kw_3062uYzhu3f1u7Y9YK; kgkFCm5y9f8mkyu81QTP1s4gJpxv1jwz3u9rd1Vhdy8EPT_K5cgvQJj1anzak50JMS68d1Cku55JNP6r5pm11CVvC8_11x0KPH13py4Q1u5; hm_lvt_fe30b0c1ee45421ec1679d18d8f8453-1569161234,1569162565,1569200902; 3545510NID=FCE597EE9EDF2691384308C737380

HTTP/1.1 200 OK
Date: Mon, 23 Sep 2019 04:03:57 GMT
Server: Apache
X-Frame-Options: SAMEORIGIN
Frame-Options: SAMEORIGIN
Last-Modified: Fri, 22 Apr 2019 10:31:56 GMT
ETag: "563-58652cfd77f00"
Accept-Ranges: bytes
Content-Length: 1379
Content-Type: image/png

.PNG
...
```

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，捕获过滤器应怎么写？

答：

host 指定 IP 地址 and tcp port http //抓取指定 IP 地址收到和发出的所有数据包,并且只捕获 HTTP 流量

- 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

一个域名和 Ping 一个 IP 地址出现的数据包有什么不同？

答：Ping 发送的是 ICMP 类型的协议数据包。在需要获取对应的 IP 地址的 MAC 地址时，会出现 ARP 数据包。在 ping 一个域名时，会出现 DNS 数据包，请求对应域名的 IP 地址后发送 ICMP 数据包，而在 ping 一个 IP 地址时，直接发送 ICMP 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？

答：Tracert/Traceroute 发送的是 ICMP 类型的 ICMP 协议数据包。整个路由器跟踪过程依靠 TTL 实现，通过向目的地址发送一系列的探测包，设置探测包的 TTL 初始值分别为 1,2,3..., 根据返回的超时通知（ICMP Time Exceeded Message）得到整个路由跟踪所需的信息。

以下详细说明 Tracert 的跟踪过程：

1. 从源地址（即本机 IP 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，每一次转发过程会使得 TTL 减 1，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；
3. 当源地址收到该 ICMP 包时，显示路由器的 IP 信息；
4. 重复 1~3 步骤，并每次设置 TTL 加 1；
5. 直至目标地址收到探测数据包，并返回 ICMP 回应答复（ICMP Echo Reply），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？

答：TCP 连接指的是传输层以 TCP 协议建立的端对端的连接。通过 TCP 连接，可以实现客户服务器和服务器的数据交互。

而应用层通过 HTTP 协议发出一次请求并受到对应的响应为一次 HTTP 会话

TCP 连接和 HTTP 会话之间的关系：TCP 协议对应于传输层，而 HTTP 协议对应于应用层，HTTP 协议是建立在 TCP 协议之上的，当浏览器需要从服务器获取网页数据的时候，会发出一次 HTTP 请求。HTTP 会通过 TCP 建立一个连接通道，在过去，当

HTTP 数据接收完以后会将 TCP 连接终止，而现在从 HTTP/1.1 起，默认都开启了 Keep-Alive，保持连接特性，简单地说，当一个网页打开完成后，客户端和服务端之间用于传输 HTTP 数据的 TCP 连接不会关闭，如果客户端再次访问这个服务器上的网页，会继续使用这一条已经建立的连接 Keep-Alive 不会永久保持连接，它有一个保持时间，可以在不同的服务器软件（如 Apache）中设定这个时间。虽然这里使用 TCP 连接保持了一段时间，但是这个时间是有限范围的，到了时间点依然是会关闭的。而 HTTP 协议中一次 HTTP 请求和一次响应就代表一次会话，在 HTTP 会话中 TCP 连接的作用是，保证数据通信的完整性和可靠性，防止丢包。

● 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？

答：客户端向 DNS 服务器查询域名，一般返回的内容都不超过 512 字节，不需要分包，用 UDP 传输即可。而且 UDP 相比 TCP 经过不用经过三次握手来建立连接，这样 DNS 服务器负载更低，响应更快。

由于 HTTP 用于获取万维网页面，而 TCP 是一个可靠的、面向连接的协议，允许从一台机器发出的字节流正确无误地交付到互联网上的另一台机器，它把输入的字节流分割成离散的报文，并把每个报文传递给互联网层，在目标机器，接受 TCP 进程把收到的报文重新装配到输出流中，这充分符合了万维网页面对于数据准确和完整性的要求。如若使用 UDP 等不保证数据完整性的协议，可能导致页面内容的缺失等问题。

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：

问题：

1. 在查阅资料后我发现 IP 和 ICMP 协议都属于 Network layer，一个 ICMP 包为什么需要同时使用两个网络层的协议而不将控制报文的功能添加在基本的 IP 功能中呢？
2. ARP 协议具体获取 MAC 地址的机制是怎么样的呢？何时会更新 MAC 地址缓存？
3. DNS 域名解析中 SOA、A、AAAA 等各项记录的作用是什么呢？

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：

在实验过程中我遇到了以下问题，在第一次进行实验的 Part3 时，我对配置了应用显示过滤器，使得列表中只显示 DNS 协议数据包，而后发现了大量的数据包，其中有许多域名对应的 IP 地址的请求和回应，这使得当时的我非常困惑，怀疑自己的实验过程出现了问题，但在多次尝试重新清除 DNS 缓存，并通过浏览器打开 www.zju.edu.cn 后得到的结果都是如此后，我查阅了相关的资料，才知道打开一个网页会自动请求网页上所有连接网站对应的域名的 IP 地址，由此解开了我的困惑。通过这次的曲折，让我明白了应该相信实验结果并如实记录，同时应当善于查阅相关资料来佐证自己的实验结果，我认为这是很重要的一点，对今后的实验的顺利进行也是非常重要的。

你对本实验安排有哪些更好的建议呢？欢迎献计献策：

可以添加 TCP 三次握手，四次挥手过程观察实验，并大致了解不同 FLAGS 的值的含义，可以帮助我们更好地理解 TCP 协议下，连接的建立和结束，也可以对包的序号变化有所认识。