

# 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:	信电学院
专 业:	电子科学与技术
学 号:	
指导教师:	陆系群

2020 年 9 月 24 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：ARP、UDP、TCP、NBNS、TLSv1.2、DTLS、DHCP、ICMPv6、IGMPv3

BROWSER、HTTP

29 0.410303	10.192.109.226	10.192.255.255	DTLS	406 Continuation Data
30 0.412291	Shenzhen_f4:ff:41	Broadcast	ARP	56 Who has 10.192.114.164? Tell 10.192.87.173
31 0.412301	10.192.176.79	10.192.255.255	NBNS	92 Name query NB AGA2018<00>
32 0.412305	10.192.237.93	10.192.255.255	BROWSER	243 Host Announcement WIN-PQ7SVMRM0TI, Workstation, Server, NT Workstation, Potential Browser

协议名：DTLS、NBNS、BROWSER

55 0.720500	04:ed:55:7e:00:et	Broadcast	ARP	56 Who has 10.192.43.234? Tell 10.192.87.173
56 0.726515	0.0.0.0	255.255.255.255	DHCP	358 DHCP Discover - Transaction ID 0x5eb0069e
57 0.729516	Shenzhen_f4:ff:41	Broadcast	ARP	56 Who has 10.192.43.234? Tell 10.192.87.173
58 0.731947	10.192.142.51	10.192.255.255	UDP	62 2008 → 2008 Len=20
59 0.731958	10.192.142.51	10.192.255.255	UDP	62 2007 → 2007 Len=20
60 0.731967	Shenzhen_f4:ff:41	Broadcast	ARP	56 Who has 10.192.116.164? Tell 10.192.87.173
61 0.731972	Shenzhen_f4:ff:41	Broadcast	ARP	56 Who has 10.192.43.235? Tell 10.192.87.173

协议名：DHCP

449 4.417063	10.192.182.30	10.192.255.255	NBNS	92 Name query NB WPAD<00>
450 4.422600	140.143.49.61	10.192.19.65	TLSv1.2	100 Application Data
451 4.422607	140.143.49.61	10.192.19.65	TLSv1.2	85 Encrypted Alert

协议名：TLSv1.2

24502 111.022179	10.192.19.65	10.10.98.98	TCP	54 52823 → 443 [ACK] Seq=2067 Ack=23305 Win=65536 Len=0
24503 111.022673	10.192.19.65	10.10.98.77	HTTP	516 GET /v2-upload/a55nhhzp.png HTTP/1.1
24504 111.022777	10.192.19.65	10.10.98.77	HTTP	516 GET /v2-upload/o1b0uogh.png HTTP/1.1
24505 111.025672	10.10.98.77	10.192.19.65	HTTP	307 HTTP/1.1 304 Not Modified
24506 111.025675	10.10.98.77	10.192.19.65	HTTP	307 HTTP/1.1 304 Not Modified

协议名：TCP、HTTP

30492 179.921417	Shenzhen_f4:ff:41	Broadcast	ARP	56 Who has 10.192.94.171? Tell 10.192.87.173
30493 180.001071	fe80::d55f:ecc5:b38...	ff02::16	ICMPv6	90 Multicast Listener Report Message v2
30494 180.001198	10.192.19.65	224.0.0.22	IGMPv3	54 Membership Report / Leave group 224.0.0.252
30495 180.008889	180.97.104.146	10.192.19.65	TLSv1.2	85 Encrypted Alert

协议名：ICMPv6、IGMPv3

42517 313.348604	88:18:72:8c:3b:4b	Broadcast	ARP	56 Who has 10.192.104.97? Tell 10.192.95.123
42518 313.350984	54:ba:d6:67:12:4b	Broadcast	ARP	56 Who has 10.192.0.1? Tell 10.192.247.237
42519 313.351000	10.192.142.51	10.192.255.255	UDP	62 2008 → 2008 Len=20
42520 313.351005	10.192.142.51	10.192.255.255	UDP	62 2007 → 2007 Len=20
42521 313.554870	7c:a1:ae:e6:dc:68	Broadcast	ARP	56 Who has 10.192.74.240? Tell 10.192.141.254
42522 313.554886	10.192.222.66	10.192.255.255	NBNS	92 Name query NB ISATAP<00>
42523 313.556927	Shenzhen_f4:ff:41	Broadcast	ARP	56 Who has 10.192.117.176? Tell 10.192.87.173
42524 313.556937	Shenzhen_f4:ff:41	Broadcast	ARP	56 Who has 10.192.118.176? Tell 10.192.87.173

协议名：UDP、ARP、NBNS

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。

截图:

The screenshot shows a packet capture in Wireshark. The top summary bar highlights the following fields: 19561, 361.469285, 10.192.19.65, 36.152.44.95, TCP, 54 [TCP Retransmission], 60. The packet list shows 'Ethernet II, Src: b8:9a:2a:0f:21:66 (b8:9a:2a:0f:21:66), Dst: 94:29:2f:38:d8:02 (94:29:2f:38:d8:02)'. The packet details pane shows the following structure: Ethernet II (Type: IPv4), Internet Protocol Version 4 (Src: 10.192.19.65, Dst: 36.152.44.95), and Transmission Control Protocol (Src Port: 60798, Dst Port: 443, Seq: 1, Ack: 1, Len: 0). The packet bytes pane shows the raw data with hex and ASCII representations. The source IP address 10.192.19.65 is highlighted in red in the original image, and the destination IP address 36.152.44.95 is highlighted in green. The source MAC address b8:9a:2a:0f:21:66 is highlighted in blue, and the destination MAC address 94:29:2f:38:d8:02 is highlighted in brown.

源 IP 地址 (红色): 10.192.19.65

目标 IP 地址 (绿色): 36.152.44.95

源 MAC 地址 (蓝色): b8:9a:2a:0f:21:66

目标 MAC 地址 (褐色): 94:29:2f:38:d8:02

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:

No.	Time	Source	Destination	Protocol	Length	Info
514	23.164775	10.10.98.77	10.181.233.173	HTTP	307	HTTP/1.1 304 Not Modified
515	23.164777	10.10.98.77	10.181.233.173	HTTP	307	HTTP/1.1 304 Not Modified
516	23.178607	10.10.98.77	10.181.233.173	HTTP	307	[TCP Spurious Retransmission] HTTP/1.1
581	23.463809	10.10.98.77	10.181.233.173	HTTP	307	HTTP/1.1 304 Not Modified
776	43.096148	10.181.233.173	10.10.98.77	HTTP	517	GET /v2-upload/qwtzjfl2.jpg HTTP/1.1
779	43.101784	10.10.98.77	10.181.233.173	HTTP	307	HTTP/1.1 304 Not Modified
967	63.095580	10.181.233.173	10.10.98.77	HTTP	517	GET /v2-upload/ac4xuku2.jpg HTTP/1.1
968	63.098275	10.10.98.77	10.181.233.173	HTTP	307	HTTP/1.1 304 Not Modified
1172	83.096865	10.181.233.173	10.10.98.77	HTTP	517	GET /v2-upload/4dym2aza.jpg HTTP/1.1
1173	83.099621	10.10.98.77	10.181.233.173	HTTP	307	HTTP/1.1 304 Not Modified
1327	103.095768	10.181.233.173	10.10.98.77	HTTP	516	GET /v2-upload/l4irnvwv.jpg HTTP/1.1
1328	103.098313	10.10.98.77	10.181.233.173	HTTP	307	HTTP/1.1 304 Not Modified
1380	112.103504	10.181.233.173	59.111.181.40	HTTP	614	POST /receiver HTTP/1.1 (application/
1383	112.111125	59.111.181.40	10.181.233.173	HTTP	232	HTTP/1.1 200 OK (application/json)

< >

> Frame 457: 520 bytes on wire (4160 bits), 520 bytes captured (4160 bits) on interface 0

> Ethernet II, Src: b8:9a:2a:0f:21:66 (b8:9a:2a:0f:21:66), Dst: JuniperN\_67:28:52 (88:e0:f3:67:28:52)

> Internet Protocol Version 4, Src: 10.181.233.173, Dst: 10.10.98.77

> Transmission Control Protocol, Src Port: 59709, Dst Port: 80, Seq: 1, Ack: 1, Len: 466

> Hypertext Transfer Protocol

0000	88 e0 f3 67 28 52	b8 9a 2a 0f 21 66 08 00 45 00	...g(R...*!f...E...
0010	01 fa 8f 02 40 00 80 06	00 00 0a b5 e9 ad 0a 0a	....@... .....
0020	62 4d e9 3d 00 50 07 46	b8 9e c4 cd 96 a1 50 18	bM...P.F .....P...
0030	01 00 62 a6 00 00 47 45	54 20 2f 76 32 2d 75 70	..b...GE T /v2-up
0040	6c 6f 61 64 2f 34 74 30	6a 61 75 65 76 2e 6a 70	load/4t0 jauvev.jp
0050	65 67 20 48 54 50 2f 31	2e 31 0d 0a 48 6f 73	eg HTTP/ 1.1..Hos
0060	74 3a 20 66 69 6c 65 2e	63 63 39 38 2e 6f 72 67	t: file. cc98.org
0070	0d 0a 43 6f 6e 6e 65 63	74 69 6f 6e 3a 20 6b 65	..Connec tion: ke
0080	65 70 2d 61 6c 69 76 65	0d 0a 55 73 65 72 2d 41	ep-alive ..User-A
0090	67 65 6e 74 3a 20 4d 6f	7a 69 6c 6c 61 2f 35 2e	gent: Mo zilla/5.
00a0	30 20 28 57 69 6e 64 6f	77 73 20 4e 54 20 31 30	0 (Windo ws NT 10
00b0	2e 30 3b 20 57 69 6e 36	34 3b 20 78 36 34 29 20	.0; Win6 4; x64)

#### 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 == 10.10.98.77 ， 希望显示的 IP 地址： 10.10.98.77 。

截图：

No.	Time	Source	Destination	Protocol	Length	Info
1004	103.130568	10.10.98.77	10.181.233.173	TCP	66	[TCP Keep-Alive ACK] 80 → 59923 [ACK]
1037	113.042220	10.181.233.173	10.10.98.77	HTTP	517	GET /v2-upload/3s0umrjz.jpg HTTP/1.1
1038	113.046129	10.10.98.77	10.181.233.173	HTTP	307	HTTP/1.1 304 Not Modified
1039	113.069433	10.10.98.77	10.181.233.173	TCP	307	[TCP Retransmission] 80 → 59921 [PSH,
1040	113.069489	10.181.233.173	10.10.98.77	TCP	66	59921 → 80 [ACK] Seq=2777 Ack=1519 Win
1137	133.040326	10.181.233.173	10.10.98.77	HTTP	517	GET /v2-upload/0jhaexhq.jpg HTTP/1.1
1138	133.046030	10.10.98.77	10.181.233.173	HTTP	307	HTTP/1.1 304 Not Modified
1139	133.086863	10.181.233.173	10.10.98.77	TCP	54	59921 → 80 [ACK] Seq=3240 Ack=1772 Win
1140	133.096475	10.10.98.77	10.181.233.173	HTTP	307	[TCP Spurious Retransmission] HTTP/1.1
1141	133.096520	10.181.233.173	10.10.98.77	TCP	66	[TCP Dup ACK 1139#1] 59921 → 80 [ACK]
1151	136.103437	10.10.98.77	10.181.233.173	TCP	56	80 → 59894 [RST, ACK] Seq=809 Ack=139
1152	136.104014	10.10.98.77	10.181.233.173	TCP	56	80 → 59920 [RST, ACK] Seq=254 Ack=464
1153	136.104018	10.10.98.77	10.181.233.173	TCP	56	80 → 59923 [RST, ACK] Seq=254 Ack=463
1154	136.104019	10.10.98.77	10.181.233.173	TCP	56	80 → 59922 [RST, ACK] Seq=254 Ack=463
1472	178.046348	10.181.233.173	10.10.98.77	TCP	55	[TCP Keep-Alive] 59921 → 80 [ACK] Seq=
1475	178.066396	10.10.98.77	10.181.233.173	TCP	66	[TCP Keep-Alive ACK] 80 → 59921 [ACK]

< >

> Frame 239: 520 bytes on wire (4160 bits), 520 bytes captured (4160 bits) on interface 0

> Ethernet II, Src: b8:9a:2a:0f:21:66 (b8:9a:2a:0f:21:66), Dst: JuniperN\_67:28:52 (88:e0:f3:67:28:52)

> Internet Protocol Version 4, Src: 10.181.233.173, Dst: 10.10.98.77

> Transmission Control Protocol, Src Port: 59894, Dst Port: 80, Seq: 1, Ack: 1, Len: 466

> Hypertext Transfer Protocol

0010	01 fa 8f 4a 40 00 80 06	00 00 0a b5 e9 ad 0a 0a	...J@... .....
0020	62 4d e9 fc 00 50 c7 3f	38 4c 04 cb 66 ee 50 18	bM...P.? 8L...f.P...
0030	00 ff 62 a6 00 00 47 45	54 20 2f 76 32 2d 75 70	..b...GE T /v2-up
0040	6c 6f 61 64 2f 34 74 30	6a 61 75 65 76 2e 6a 70	load/4t0 jauvev.jp
0050	65 67 20 48 54 50 2f 31	2e 31 0d 0a 48 6f 73	eg HTTP/ 1.1..Hos
0060	74 3a 20 66 69 6c 65 2e	63 63 39 38 2e 6f 72 67	t: file. cc98.org
0070	0d 0a 43 6f 6e 6e 65 63	74 69 6f 6e 3a 20 6b 65	..Connec tion: ke

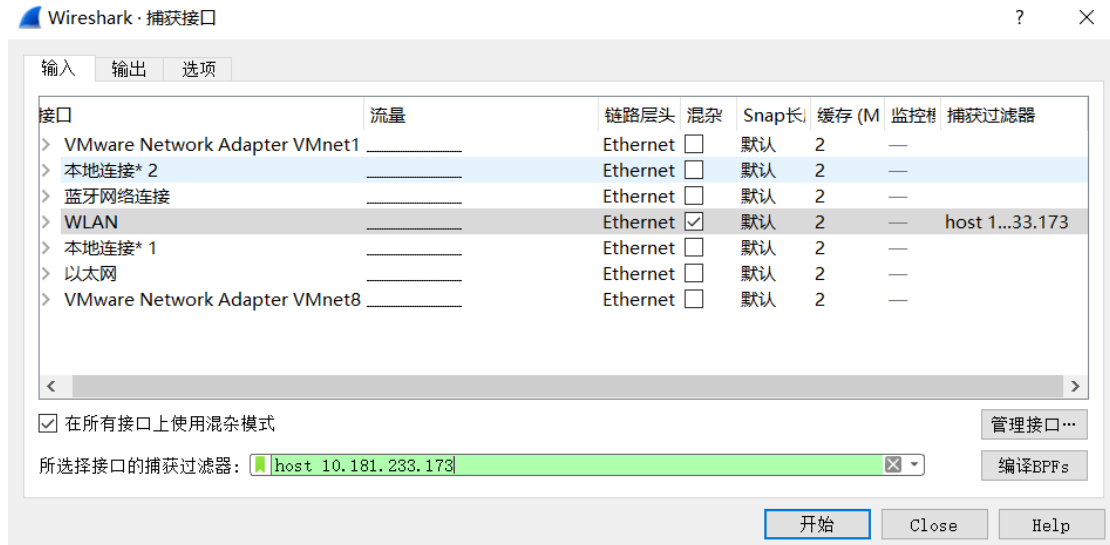
Source (ip.src), 4 bytes      分组: 1615 · 已显示: 83 (5.1%)      Profile: Default

## 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）。

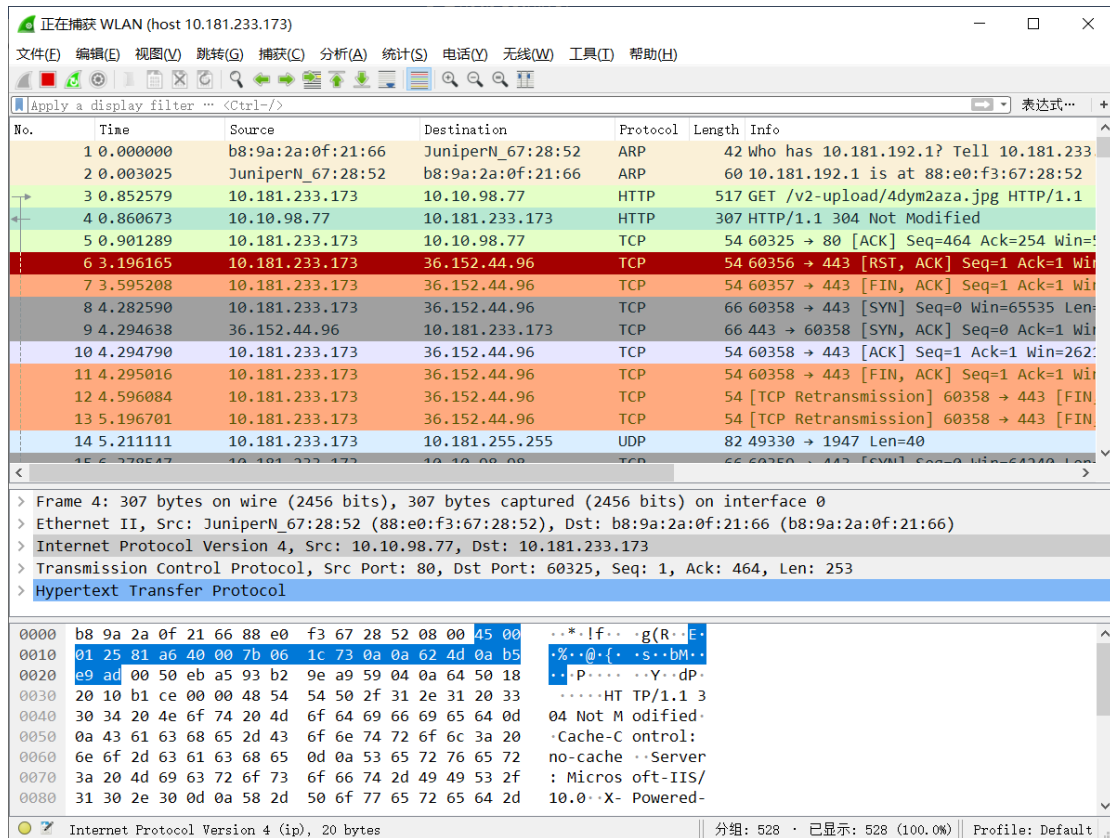
使用的过滤器： host 10.181.233.173 ，希望捕获的 IP 地址： 10.181.233.173 。

截图：

过滤器配置：



捕获结果：

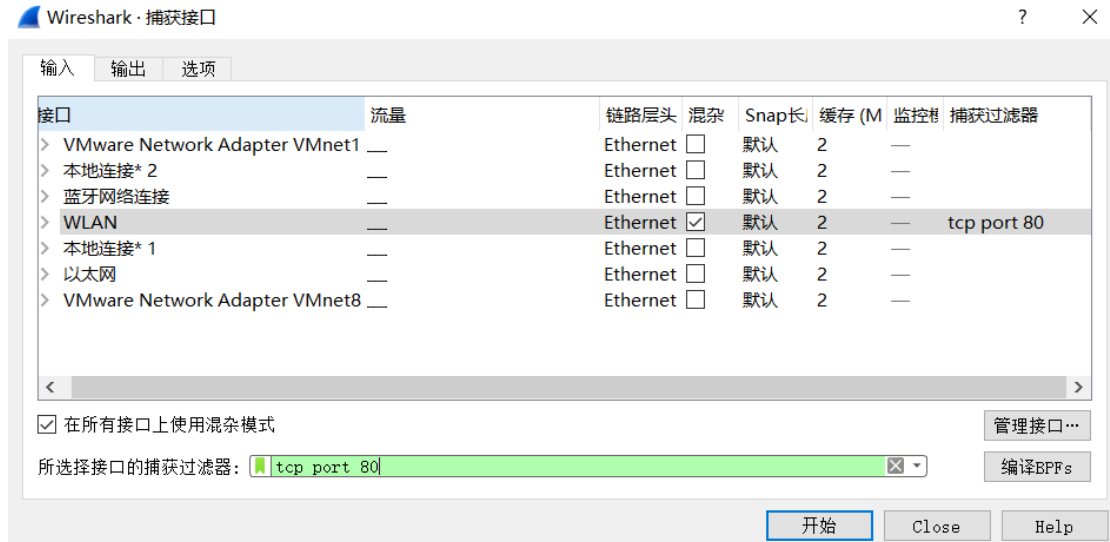


## 6. 配置捕获过滤器，只捕获某类协议的数据包（tcp port xx 或者 udp port xx）。

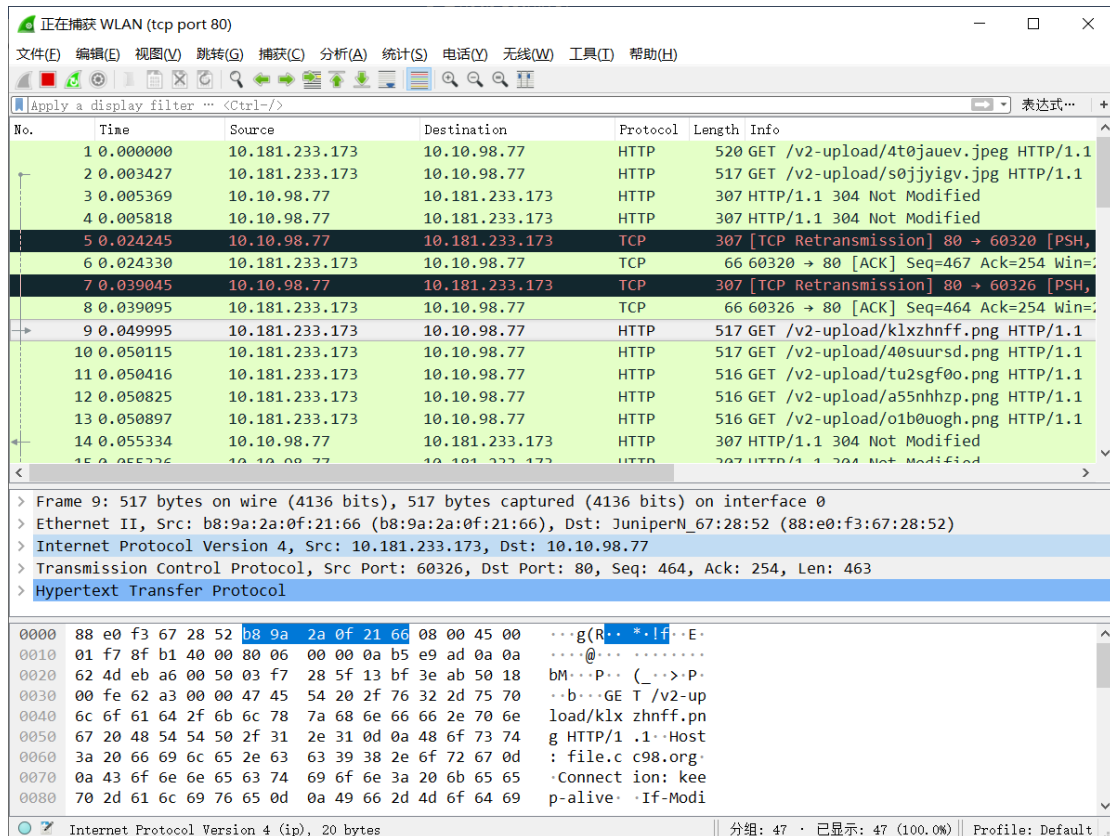
使用的过滤器： tcp port 80 ， 希望捕获的协议类型： HTTP 。

截图：

过滤器配置：



捕获结果：



## ✧ 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? Ethernet II、IPv4、UDP、DNS。使用的服务方端口是: 10.10.0.21。

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。

nslookup:

```
C:\WINDOWS\system32\cmd.exe

C:\Users\24547>nslookup www.baidu.com
服务器:  dns1.zju.edu.cn
Address:  10.10.0.21

非权威应答:
名称:     www.a.shifen.com
Addresses: 36.152.44.95
          36.152.44.96
Aliases:  www.baidu.com
```

请求包:

Wireshark packet capture details for frame 4306 (DNS Standard query):

- Transaction ID: 0x0002
- Flags: 0x0100 Standard query
- Questions: 1
- Answer RRs: 0
- Authority RRs: 0
- Additional RRs: 0
- Queries
  - www.baidu.com: type A, class IN
    - Name: www.baidu.com
    - [Name Length: 13]
    - [Label Count: 3]
    - Type: A (Host Address) (1)
    - Class: IN (0x0001)

Packet bytes (hex and ASCII):

```
0000  94 29 2f 38 d8 02 b8 9a 2a 0f 21 66 08 00 45 00  .)/8...*.!f..E.
0010  00 3b 02 3f 00 00 80 11 00 00 0a c0 32 26 0a 0a  .;?...2&..
0020  00 15 f2 01 00 35 00 27 47 3d 00 02 01 00 00 01  .....5.'G=.....
0030  00 00 00 00 00 00 03 77 77 77 05 62 61 69 64 75  .....w ww.baidu
0040  03 63 6f 6d 00 00 01 00 01                      .com.....
```

交易 ID (红色): 0x0002

查询类型 (绿色): A

查询的域名内容 (蓝色): [www.baidu.com](http://www.baidu.com)



响应包：

4307	10.580388	10.10.0.21	10.192.50.38	DNS	302 Standard query response 0x0002 A www.baidu.com
4308	10.580387	10.192.50.38	10.10.0.21	DNS	73 Standard query 0x0002 AAAA www.baidu.com
< >					
> Frame 4307: 302 bytes on wire (2416 bits), 302 bytes captured (2416 bits) on interface 0					
> Ethernet II, Src: 94:29:2f:38:d8:02 (94:29:2f:38:d8:02), Dst: b8:9a:2a:0f:21:66 (b8:9a:2a:0f:21:66)					
> Internet Protocol Version 4, Src: 10.10.0.21, Dst: 10.192.50.38					
> User Datagram Protocol, Src Port: 53, Dst Port: 61953					
v Domain Name System (response)					
Transaction ID: 0x0002					
> Flags: 0x8180 Standard query response, No error					
Questions: 1					
Answer RRs: 3					
Authority RRs: 5					
Additional RRs: 5					
v Queries					
v www.baidu.com: type A, class IN					
Name: www.baidu.com					
[Name Length: 13]					
[Label Count: 3]					
Type: A (Host Address) (1)					
Class: IN (0x0001)					
v Answers					
> www.baidu.com: type CNAME, class IN, cname www.a.shifen.com					
> www.a.shifen.com: type A, class IN, addr 36.152.44.95					
> www.a.shifen.com: type A, class IN, addr 36.152.44.96					
> Authoritative nameservers					
> Additional records					
[Request In: 4306]					
[Time: 0.002422000 seconds]					
0020	32 26 00 35 f2 01 01 0c 60 de 00 02 81 80 00 01	2&.5... ..			
0030	00 03 00 05 00 05 03 77 77 77 05 62 61 69 64 75	.....w ww-baidu			
0040	03 63 6f 6d 00 00 01 00 01 c0 0c 00 05 00 01 00	.com.....			
0050	00 01 d7 00 0f 03 77 77 77 01 61 06 73 68 69 66	.....ww w-a-shif			
0060	65 6e c0 16 c0 2b 00 01 00 01 00 00 01 39 00 04	en.....9..			
0070	24 98 2c 5f c0 2b 00 01 00 01 00 00 01 39 00 04	\$. , +. ....9..			
0080	24 98 2c 60 c0 2f 00 02 00 01 00 00 01 d6 00 06	\$. , ./... ..			

交易 ID（红色）：0x0002

查询类型（褐色）：A

查询的域名内容（蓝色）：[www.baidu.com](http://www.baidu.com)

查询结果（黄、紫、粉）：共 3 条。

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： DNS、ICMP、ARP、TCP

Ping 域名时： ICMP、ARP、TCP

ICMP 数据包分别由哪几层协议构成？ 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。

ping:

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [版本 10.0.18363.1082]
(c) 2019 Microsoft Corporation。保留所有权利。

C:\Users\24547>ping www.baidu.com

正在 Ping www.a.shifen.com [36.152.44.96] 具有 32 字节的数据:
来自 36.152.44.96 的回复: 字节=32 时间=19ms TTL=55
来自 36.152.44.96 的回复: 字节=32 时间=20ms TTL=55
来自 36.152.44.96 的回复: 字节=32 时间=15ms TTL=55
来自 36.152.44.96 的回复: 字节=32 时间=19ms TTL=55

36.152.44.96 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
往返行程的估计时间(以毫秒为单位):
    最短 = 15ms, 最长 = 20ms, 平均 = 18ms

C:\Users\24547>ping 36.152.44.96

正在 Ping 36.152.44.96 具有 32 字节的数据:
来自 36.152.44.96 的回复: 字节=32 时间=12ms TTL=55
来自 36.152.44.96 的回复: 字节=32 时间=21ms TTL=55
来自 36.152.44.96 的回复: 字节=32 时间=19ms TTL=55
来自 36.152.44.96 的回复: 字节=32 时间=17ms TTL=55

36.152.44.96 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
往返行程的估计时间(以毫秒为单位):
    最短 = 12ms, 最长 = 21ms, 平均 = 17ms

C:\Users\24547>
```

请求包:

35	16.689246	b8:9a:2a:0f:21:66	84:46:fe:26:2f:2e	ARP	42 Who has 10.185.255.254? Tell 10.185.134.207
36	17.027133	84:46:fe:26:2f:2e	b8:9a:2a:0f:21:66	ARP	60 10.185.255.254 is at 84:46:fe:26:2f:2e
37	17.929213	10.185.134.207	120.131.14.109	UDP	151 59061 → 9406 Len=109

> Frame 35: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0

> Ethernet II, Src: b8:9a:2a:0f:21:66 (b8:9a:2a:0f:21:66), Dst: 84:46:fe:26:2f:2e (84:46:fe:26:2f:2e)

> Address Resolution Protocol (request)

Hardware type: Ethernet (1)

Protocol type: IPv4 (0x0800)

Hardware size: 6

Protocol size: 4

Opcode: request (1)

Sender MAC address: b8:9a:2a:0f:21:66 (b8:9a:2a:0f:21:66)

Sender IP address: 10.185.134.207

Target MAC address: 84:46:fe:26:2f:2e (84:46:fe:26:2f:2e)

Target IP address: 10.185.255.254

0000	84 46 fe 26 2f 2e b8 9a 2a 0f 21 66 08 06 00 01	.F.&/... *!f...
0010	08 00 06 04 00 01 b8 9a 2a 0f 21 66 0a b9 86 cf	..... *!f...
0020	84 46 fe 26 2f 2e 0a b9 ff fe	.F.&/... ..

操作码（紫色）、发送者 IP 地址（绿色）、发送者 MAC 地址（红色）、查询的目标 IP 地址（褐色）、Ethernet 层的目标 MAC 地址（黄色）

响应包：

35	16.689246	b8:9a:2a:0f:21:66	84:46:fe:26:2f:2e	ARP	42 who has 10.185.255.254? Tell 10.185.134.207
36	17.027133	84:46:fe:26:2f:2e	b8:9a:2a:0f:21:66	ARP	60 10.185.255.254 is at 84:46:fe:26:2f:2e
37	17.929213	10.185.134.207	120.131.14.109	UDP	151 59061 → 9406 Len=109

<

> Frame 36: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0

> Ethernet II, Src: 84:46:fe:26:2f:2e (84:46:fe:26:2f:2e), Dst: b8:9a:2a:0f:21:66 (b8:9a:2a:0f:21:66)

▼ Address Resolution Protocol (reply)

Hardware type: Ethernet (1)

Protocol type: IPv4 (0x0800)

Hardware size: 6

Protocol size: 4

Opcode: reply (2)

Sender MAC address: 84:46:fe:26:2f:2e (84:46:fe:26:2f:2e)

Sender IP address: 10.185.255.254

Target MAC address: b8:9a:2a:0f:21:66 (b8:9a:2a:0f:21:66)

Target IP address: 10.185.134.207

0000 b8 9a 2a 0f 21 66 84 46 fe 26 2f 2e 08 06 00 01 ..\*.!f.F.&/.....

0010 08 00 06 04 00 02 84 46 fe 26 2f 2e 0a b9 ff fe .....F.&/.....

0020 b8 9a 2a 0f 21 66 0a b9 86 c1 00 00 00 00 00 00 ..\*.!f.....

0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....</div>

操作码（黄色）、发送者 IP 地址（绿色）、发送者 MAC 地址（红色）、查询的目标 IP 地址（紫色）、Ethernet 层的目标 MAC 地址（蓝色）

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：

80	37.026419	10.185.134.207	36.152.44.96	ICMP	74 Echo (ping) request id=0x0001, seq=1064/10244, ttl=128 (reply in 81)
81	37.046244	36.152.44.96	10.185.134.207	ICMP	74 Echo (ping) reply id=0x0001, seq=1064/10244, ttl=55 (request in 80)

<

>

> Frame 80: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0

> Ethernet II, Src: b8:9a:2a:0f:21:66 (b8:9a:2a:0f:21:66), Dst: 84:46:fe:26:2f:2e (84:46:fe:26:2f:2e)

> Internet Protocol Version 4, Src: 10.185.134.207, Dst: 36.152.44.96

▼ Internet Control Message Protocol

Type: 8 (Echo (ping) request)

Code: 0

Checksum: 0x4933 [correct]

[Checksum Status: Good]

Identifier (BE): 1 (0x0001)

Identifier (LE): 256 (0x0100)

Sequence number (BE): 1064 (0x0428)

Sequence number (LE): 10244 (0x2804)

[Response frame: 81]

▼ Data (32 bytes)

Data: 6162636465666768696a6b6c6d6e6f707172737475767761...

[Length: 32]

0000 84 46 fe 26 2f 2e b8 9a 2a 0f 21 66 08 00 45 00 .F.&/...\*.!f..E:

0010 00 3c d8 46 00 00 00 01 00 00 0a b9 86 cf 24 98 .<.F.....\$.:

0020 2c 60 08 00 49 33 00 01 04 28 61 62 63 64 65 66 ,...I3... (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 wabcdefg hi

红色为类型，绿色为序号

响应包：

80	37.026419	10.185.134.207	36.152.44.96	ICMP	74 Echo (ping) request	id=0x0001, seq=1064/10244, ttl=128 (reply in 81)
81	37.046244	36.152.44.96	10.185.134.207	ICMP	74 Echo (ping) reply	id=0x0001, seq=1064/10244, ttl=55 (request in 80)

< >

> Frame 81: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0

> Ethernet II, Src: 84:46:fe:26:2f:2e (84:46:fe:26:2f:2e), Dst: b8:9a:2a:0f:21:66 (b8:9a:2a:0f:21:66)

> Internet Protocol Version 4, Src: 36.152.44.96, Dst: 10.185.134.207

> Internet Control Message Protocol

Type: 0 (Echo (ping) reply)

code: 0

Checksum: 0x5133 [correct]

[Checksum Status: Good]

Identifier (BE): 1 (0x0001)

Identifier (LE): 256 (0x0100)

Sequence number (BE): 1064 (0x0428)

Sequence number (LE): 10244 (0x2804)

[Request frame: 80]

[Response time: 19.825 ms]

> Data (32 bytes)

Data: 6162636465666768696a6b6c6d6e6f707172737475767761...

[Length: 32]

0000 b8 9a 2a 0f 21 66 84 46 fe 26 2f 2e 08 00 45 04 ..\*.If.F.&/...E.

0010 00 3c d8 46 00 00 37 01 c8 f6 24 98 2c 60 0a b9 <.F..7. .\$.,'..

0020 86 cf 00 51 33 00 01 04 28 61 62 63 64 65 66 ..Q3... (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 wabcdefg hi

红色为类型，绿色为序号

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？ 3 。

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：

tracert:

```
C:\WINDOWS\system32\cmd.exe
C:\Users\24547>tracert www.baidu.com
通过最多 30 个跃点跟踪
到 www.a.shifen.com [36.152.44.95] 的路由:

 1  29 ms    16 ms    14 ms    10.185.255.254
 2  19 ms    *        *        10.3.7.166
 3  *        *        *        请求超时。
 4  23 ms    18 ms    17 ms    39.174.130.17
 5   8 ms    17 ms    16 ms    221.183.64.53
 6  69 ms    14 ms    32 ms    221.183.42.129
 7  22 ms    26 ms    15 ms    221.183.59.54
 8  30 ms    14 ms    32 ms    166.23.207.183.static.js.chinamobile.com [183.207.23.166]
 9  34 ms    17 ms    15 ms    182.61.253.214
10  *        *        *        请求超时。
11  18 ms    15 ms    14 ms    36.152.44.95

跟踪完成。
C:\Users\24547>
```

## Wireshark 捕获结果:

9 12.294128	10.185.134.207	36.152.44.95	ICMP	106 Echo (ping) request	id=0x0001, seq=38/9728, ttl=1 (no response found!)
10 12.323993	10.185.255.254	10.185.134.207	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)	
11 12.325453	10.185.134.207	36.152.44.95	ICMP	106 Echo (ping) request	id=0x0001, seq=39/9984, ttl=1 (no response found!)
12 12.342359	10.185.255.254	10.185.134.207	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)	
13 12.343955	10.185.134.207	36.152.44.95	ICMP	106 Echo (ping) request	id=0x0001, seq=40/10240, ttl=1 (no response found!)
14 12.358543	10.185.255.254	10.185.134.207	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)	
20 12.417291	10.185.255.254	10.185.134.207	ICMP	70 Destination unreachable (Port unreachable)	
24 13.908898	10.185.255.254	10.185.134.207	ICMP	70 Destination unreachable (Port unreachable)	
26 15.415141	10.185.255.254	10.185.134.207	ICMP	70 Destination unreachable (Port unreachable)	
36 17.896157	10.185.134.207	36.152.44.95	ICMP	106 Echo (ping) request	id=0x0001, seq=41/10496, ttl=2 (no response found!)
37 17.914997	10.3.7.166	10.185.134.207	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)	
38 17.918075	10.185.134.207	36.152.44.95	ICMP	106 Echo (ping) request	id=0x0001, seq=42/10752, ttl=2 (no response found!)
41 21.899246	10.185.134.207	36.152.44.95	ICMP	106 Echo (ping) request	id=0x0001, seq=43/11008, ttl=2 (no response found!)
47 30.420665	10.185.134.207	36.152.44.95	ICMP	106 Echo (ping) request	id=0x0001, seq=44/11264, ttl=3 (no response found!)
49 34.399016	10.185.134.207	36.152.44.95	ICMP	106 Echo (ping) request	id=0x0001, seq=45/11520, ttl=3 (no response found!)
50 38.400151	10.185.134.207	36.152.44.95	ICMP	106 Echo (ping) request	id=0x0001, seq=46/11776, ttl=3 (no response found!)
51 42.397785	10.185.134.207	36.152.44.95	ICMP	106 Echo (ping) request	id=0x0001, seq=47/12032, ttl=4 (no response found!)
52 42.421080	39.174.130.17	10.185.134.207	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)	
53 42.422809	10.185.134.207	36.152.44.95	ICMP	106 Echo (ping) request	id=0x0001, seq=48/12288, ttl=4 (no response found!)
54 42.440996	39.174.130.17	10.185.134.207	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)	
55 42.443236	10.185.134.207	36.152.44.95	ICMP	106 Echo (ping) request	id=0x0001, seq=49/12544, ttl=4 (no response found!)
56 42.460733	39.174.130.17	10.185.134.207	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)	
92 52.511975	39.174.130.17	10.185.134.207	ICMP	70 Destination unreachable (Port unreachable)	
94 53.991375	39.174.130.17	10.185.134.207	ICMP	70 Destination unreachable (Port unreachable)	
98 55.487850	39.174.130.17	10.185.134.207	ICMP	70 Destination unreachable (Port unreachable)	
99 57.967428	10.185.134.207	36.152.44.95	ICMP	106 Echo (ping) request	id=0x0001, seq=50/12800, ttl=5 (no response found!)
100 57.976236	221.183.64.53	10.185.134.207	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)	
101 57.979035	10.185.134.207	36.152.44.95	ICMP	106 Echo (ping) request	id=0x0001, seq=51/13056, ttl=5 (no response found!)
102 57.996069	221.183.64.53	10.185.134.207	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)	
103 57.997543	10.185.134.207	36.152.44.95	ICMP	106 Echo (ping) request	id=0x0001, seq=52/13312, ttl=5 (no response found!)
104 58.013541	221.183.64.53	10.185.134.207	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)	
108 58.040884	221.183.64.53	10.185.134.207	ICMP	70 Destination unreachable (Port unreachable)	
113 59.533139	221.183.64.53	10.185.134.207	ICMP	70 Destination unreachable (Port unreachable)	
115 61.059839	221.183.64.53	10.185.134.207	ICMP	70 Destination unreachable (Port unreachable)	
125 63.517720	10.185.134.207	36.152.44.95	ICMP	106 Echo (ping) request	id=0x0001, seq=53/13568, ttl=6 (no response found!)
126 63.506606	221.183.42.129	10.185.134.207	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)	
127 63.509979	10.185.134.207	36.152.44.95	ICMP	106 Echo (ping) request	id=0x0001, seq=54/13824, ttl=6 (no response found!)
128 63.604631	221.183.42.129	10.185.134.207	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)	
129 63.607898	10.185.134.207	36.152.44.95	ICMP	106 Echo (ping) request	id=0x0001, seq=55/14080, ttl=6 (no response found!)
130 63.639794	221.183.42.129	10.185.134.207	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)	
134 63.852085	221.183.42.129	10.185.134.207	ICMP	70 Destination unreachable (Port unreachable)	
136 65.241517	221.183.42.129	10.185.134.207	ICMP	70 Destination unreachable (Port unreachable)	
138 66.722641	221.183.42.129	10.185.134.207	ICMP	70 Destination unreachable (Port unreachable)	
141 69.176730	10.185.134.207	36.152.44.95	ICMP	106 Echo (ping) request	id=0x0001, seq=56/14336, ttl=7 (no response found!)
142 69.108085	221.183.59.54	10.185.134.207	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)	
143 69.201045	10.185.134.207	36.152.44.95	ICMP	106 Echo (ping) request	id=0x0001, seq=57/14592, ttl=7 (no response found!)
144 69.227945	221.183.59.54	10.185.134.207	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)	
145 69.229484	10.185.134.207	36.152.44.95	ICMP	106 Echo (ping) request	id=0x0001, seq=58/14848, ttl=7 (no response found!)
146 69.245057	221.183.59.54	10.185.134.207	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)	
152 74.813916	10.185.134.207	36.152.44.95	ICMP	106 Echo (ping) request	id=0x0001, seq=59/15104, ttl=8 (no response found!)
153 74.844402	183.207.23.166	10.185.134.207	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)	
154 74.847539	10.185.134.207	36.152.44.95	ICMP	106 Echo (ping) request	id=0x0001, seq=60/15360, ttl=8 (no response found!)
155 74.862211	183.207.23.166	10.185.134.207	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)	
156 74.865353	10.185.134.207	36.152.44.95	ICMP	106 Echo (ping) request	id=0x0001, seq=61/15616, ttl=8 (no response found!)
157 74.897688	183.207.23.166	10.185.134.207	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)	
159 75.871248	10.185.134.207	36.152.44.95	ICMP	106 Echo (ping) request	id=0x0001, seq=62/15872, ttl=9 (no response found!)
160 75.905921	182.61.253.214	10.185.134.207	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)	
161 75.908232	10.185.134.207	36.152.44.95	ICMP	106 Echo (ping) request	id=0x0001, seq=63/16128, ttl=9 (no response found!)
162 75.926093	182.61.253.214	10.185.134.207	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)	
163 75.927682	10.185.134.207	36.152.44.95	ICMP	106 Echo (ping) request	id=0x0001, seq=64/16384, ttl=9 (no response found!)
164 75.943528	182.61.253.214	10.185.134.207	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)	
170 81.456693	10.185.134.207	36.152.44.95	ICMP	106 Echo (ping) request	id=0x0001, seq=65/16640, ttl=10 (no response found!)
171 85.398690	10.185.134.207	36.152.44.95	ICMP	106 Echo (ping) request	id=0x0001, seq=66/16896, ttl=10 (no response found!)
172 89.397572	10.185.134.207	36.152.44.95	ICMP	106 Echo (ping) request	id=0x0001, seq=67/17152, ttl=10 (no response found!)
176 93.397698	10.185.134.207	36.152.44.95	ICMP	106 Echo (ping) request	id=0x0001, seq=68/17408, ttl=11 (reply in 177)
177 93.416080	36.152.44.95	10.185.134.207	ICMP	106 Echo (ping) reply	id=0x0001, seq=68/17408, ttl=55 (request in 176)
178 93.417661	10.185.134.207	36.152.44.95	ICMP	106 Echo (ping) request	id=0x0001, seq=69/17664, ttl=11 (reply in 179)
179 93.433512	36.152.44.95	10.185.134.207	ICMP	106 Echo (ping) reply	id=0x0001, seq=69/17664, ttl=55 (request in 178)
180 93.436771	10.185.134.207	36.152.44.95	ICMP	106 Echo (ping) request	id=0x0001, seq=70/17920, ttl=11 (reply in 181)
181 93.450708	36.152.44.95	10.185.134.207	ICMP	106 Echo (ping) reply	id=0x0001, seq=70/17920, ttl=55 (request in 180)

观察并记录响应包的信息，第一组响应包的发送者 IP 是：10.185.255.254，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:

9	12.294128	10.185.134.207	36.152.44.95	ICMP	106 Echo (ping) request	id=0x0001, seq=38/9728, ttl=1 (no re
10	12.323993	10.185.255.254	10.185.134.207	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)	

<

> Frame 10: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0

> Ethernet II, Src: 84:46:fe:26:2f:2e (84:46:fe:26:2f:2e), Dst: b8:9a:2a:0f:21:66 (b8:9a:2a:0f:21:66)

> Internet Protocol Version 4, Src: 10.185.255.254, Dst: 10.185.134.207

> 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: 10.185.134.207, Dst: 36.152.44.95

> Internet Control Message Protocol

Type: 8 (Echo (ping) request)

Code: 0

Checksum: 0xf7d8 [unverified] [in ICMP error packet]

[Checksum Status: Unverified]

Identifier (BE): 1 (0x0001)

Identifier (LE): 256 (0x0100)

Sequence number (BE): 38 (0x0026)

Sequence number (LE): 9728 (0x2600)

最后一组:

180	93.436771	10.185.134.207	36.152.44.95	ICMP	106 Echo (ping) request	id=0x0001, seq=70/17920, ttl=11 (reply in 181)
181	93.450768	36.152.44.95	10.185.134.207	ICMP	106 Echo (ping) reply	id=0x0001, seq=70/17920, ttl=55 (request in 180)

<

> Frame 181: 106 bytes on wire (848 bits), 106 bytes captured (848 bits) on interface 0

> Ethernet II, Src: 84:46:fe:26:2f:2e (84:46:fe:26:2f:2e), Dst: b8:9a:2a:0f:21:66 (b8:9a:2a:0f:21:66)

> Internet Protocol Version 4, Src: 36.152.44.95, Dst: 10.185.134.207

> Internet Control Message Protocol

Type: 0 (Echo (ping) reply)

Code: 0

Checksum: 0xffb8 [correct]

[Checksum Status: Good]

Identifier (BE): 1 (0x0001)

Identifier (LE): 256 (0x0100)

Sequence number (BE): 70 (0x0046)

Sequence number (LE): 17920 (0x4600)

[Request frame: 180]

[Response time: 13.997 ms]

> Data (64 bytes)

## ✧ 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: Ethernet II、IPv4、UDP、DNS

HTTP: Ethernet II、IPv4、TCP、HTTP

dns:

3	3.884361	10.192.50.38	10.10.0.21	DNS	74 Standard query 0x6db9 A www.zju.edu.cn
4	3.888245	10.10.0.21	10.192.50.38	DNS	125 Standard query response 0x6db9 A www.zju.edu.cn

> Frame 3: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0  
> Ethernet II, Src: b8:9a:2a:0f:21:66 (b8:9a:2a:0f:21:66), Dst: 94:29:2f:38:d8:02 (94:29:2f:38:d8:02)  
> Internet Protocol Version 4, Src: 10.192.50.38, Dst: 10.10.0.21  
> User Datagram Protocol, Src Port: 54912, Dst Port: 53  
> Domain Name System (query)

http:

4	0.004484	10.192.50.38	10.203.6.126	HTTP	438 GET / HTTP/1.1
15	0.012816	10.203.6.126	10.192.50.38	HTTP	221 HTTP/1.1 200 OK (text/html)

> Frame 4: 438 bytes on wire (3504 bits), 438 bytes captured (3504 bits) on interface 0  
> Ethernet II, Src: b8:9a:2a:0f:21:66 (b8:9a:2a:0f:21:66), Dst: 94:29:2f:38:d8:02 (94:29:2f:38:d8:02)  
> Internet Protocol Version 4, Src: 10.192.50.38, Dst: 10.203.6.126  
> Transmission Control Protocol, Src Port: 63909, Dst Port: 80, Seq: 1, Ack: 1, Len: 384  
> 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: dns1.zju.edu.cn、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（最好有多个 HTTP 会话的）:

### 会话 1:

```
GET / HTTP/1.1
Accept: text/html, application/xhtml+xml, image/jxr, */*
Accept-Language: zh-Hans-CN,zh-Hans;q=0.5
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; Trident/7.0; rv:11.0) like Gecko
Accept-Encoding: gzip, deflate
Host: www.zju.edu.cn
Connection: Keep-Alive
Cookie: Hm_lvt_fe30bbc1ee45421ec1679d1b8d8f8453=1601118426,1601118510; _ga=GA1.3.1907101151.1587623429

HTTP/1.1 200 OK
Server: nginx
Date: Sat, 26 Sep 2020 11:24:10 GMT
Content-Type: text/html
Content-Length: 12979
Connection: keep-alive
Set-Cookie: route=66f625e2c030628c4d6133593896946a; Path=/
X-Frame-Options: SAMEORIGIN
Frame-Options: SAMEORIGIN
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Encoding: gzip
```

红色为请求，蓝色为响应

### 会话 2:

```
..7a@c...@....9.,....GET /_visitcount?siteId=590&type=1&columnId=32642 HTTP/1.1
Accept: image/png, image/svg+xml, image/jxr, image/*;q=0.8, */*;q=0.5
Referer: http://www.zju.edu.cn/
Accept-Language: zh-Hans-CN,zh-Hans;q=0.5
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; Trident/7.0; rv:11.0) like Gecko
Accept-Encoding: gzip, deflate
Host: www.zju.edu.cn
Connection: Keep-Alive
Cookie: Hm_lvt_fe30bbc1ee45421ec1679d1b8d8f8453=1601118426,1601118510; _ga=GA1.3.1907101151.1587623429;
route=66f625e2c030628c4d6133593896946a

HTTP/1.1 200 OK
Server: nginx
Date: Sat, 26 Sep 2020 11:24:10 GMT
Content-Length: 0
Connection: keep-alive
X-Frame-Options: SAMEORIGIN
Frame-Options: SAMEORIGIN
Set-Cookie: JSESSIONID=576EEBF98AE2B511451E0512B68BF48B; Path=/
```

红色为请求，蓝色为响应

## 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？

host xx.xx.xx.xx



- 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？

1、Ping 发送的是 ICMP 类型的协议数据包。

2、假设主机 A 的 IP 地址为 IP1，主机 B 的 IP 地址为 IP2，主机 A 向主机 B 发起 ping 命令。当目标 IP 地址（即 IP2）与其 MAC 地址的映射不在发送者（A）的 ARP 缓存表中时，会出现 ARP 数据包，即主机 A 要发送询问信息，询问主机 B 的 MAC 地址。

3、Ping 域名会出现 DNS 数据包，而 Ping IP 不会。因为 ICMP 协议封装 IP 数据包需要目标 IP 地址，Ping 域名需要先获取域名对应的 IP 地址。

- 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？

1、Tracert/Traceroute 发送的是 ICMP 类型的协议数据包。

2、整个路由跟踪的流程：

- 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；
- 到达路由器时，将 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；
- 当源地址收到该 ICMP 包时，显示这一跳路由信息；
- 重复 1~5，并每次设置 TTL 加 1；
- 直至目标地址收到探测数据包，并返回 ICMP 回应答复（ICMPEcho 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？

TCP 协议对应于传输层，HTTP 协议对应于应用层，HTTP 协议建立在 TCP 协议基

础之上。当浏览器需要从服务器获取网页数据时，会发出一次 HTTP 请求，HTTP 通过 TCP 建立连接。

- 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？

UDP 面向无连接，可靠性不高，是一次性的传递，不需要接收方的响应；TCP 面向连接，可靠性较高，资源开销大。

1、DNS 对效率要求较高。采用 TCP 传输，则域名解析时间=TCP 连接时间+DNS 交易时间；采用 UDP 传输，则域名解析时间= DNS 交易时间。采用 UDP 传输，DNS 域名解析时间更小，通信效率更高。

2、HTTP 是无状态的协议，对传输的可靠性要求较高，故采用 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：

- 1、ARP、TCP、UDP、HTTP 等协议的意义。
- 2、子网、IP 地址、MAC 地址等的意义。
- 3、Wireshark 各种颜色代表的意义。

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：

1、显示过滤器和捕获过滤器是不同的，配置时间也是不一样的。显示过滤器应该在抓包之后使用，显示特定的协议包，捕获过滤器是在抓包之前用，只抓特定的数据包。

2、udp port 53 与 DNS 有关，tcp port 80 与 HTTP 有关，在做 part 3 的时候要分开抓包。只配置一个捕获过滤器不能同时抓到 DNS 和 HTTP 有关的数据包。

3、做 part 3 的时候最好使用 ie 浏览器，或者其它一些没有访问过网站的浏览器，以减

少干扰。在用 edge 和 chrome 访问 [www.zju.edu.cn](http://www.zju.edu.cn) 时，我抓到的 DNS 协议有关的数据包就又多又杂，影响数据了。后来在助教的建议下改用了 ie，抓到的包就干净了很多。

4、在抓与 ARP 协议有关的数据包时，尽量不要挑选教室之类主机很多的地方，会出现 ARP 问询得不到响应的情况。

你对本实验安排有哪些更好的建议呢？欢迎献计献策：

1、希望 lab 上交截止时间能够适当延长。整个 lab1 只是跟着教程做实验，很多命令都不懂背后的意义，baidu 和 google 后也只是稍微清楚一些。计网的内容又多又复杂，没有形成完整的知识架构之前，做实验就是糊里糊涂的，一个协议查完又会延伸开去查一些相关的东西，特别耗时间。

2、希望 Wireshark 的教程能够再详细些。感觉做完这个 lab，对 Wireshark 只能算是入门，界面上的很多东西还是不清楚。虽然这可能与我们的理论课的滞后有关系，但还是觉得对 Wireshark 界面的介绍应该详细一些。