《计算机网络》实验报告 5

利用 Wireshark 进行协议

童超宇

院 （系）：计算机科学与技术学院　　专　　业：计算机科学与技术

学　　号 ：1152130106　　指导教师：刘亚维

2018 年 5 月

目录

1 实验目的

熟悉并掌握 Wireshark 的基本操作，了解网络协议实体间进行交互以及报文交换的情况

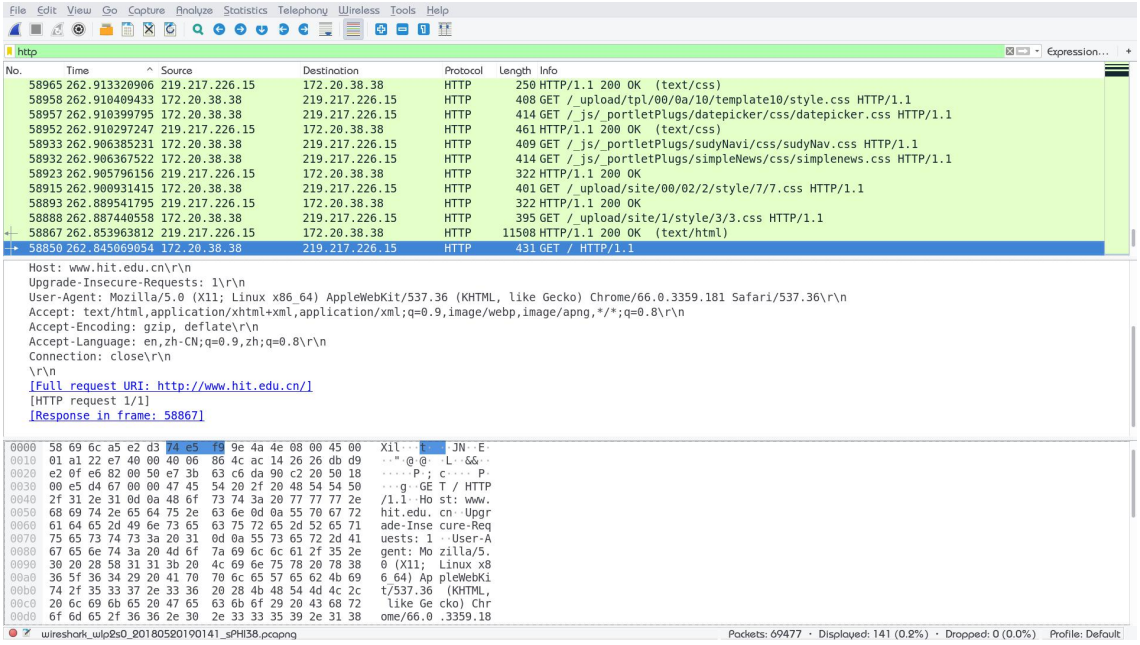
2 实验内容

1)学习 Wireshark 的使用
2)利用 Wireshark 分析 HTTP 协议
3)利用 Wireshark 分析 TCP 协议
4)利用 Wireshark 分析 IP 协议
5)利用 Wireshark 分析 Ethernet 数据帧
选做内容:
a)利用 Wireshark 分析 DNS 协议
b)利用 Wireshark 分析 UDP 协议
c)利用 Wireshark 分析 ARP 协议

3 实验过程

3.1 HTTP 分析(www.hit.edu.cn)

3.1.1 HTTP GET/response 交互



1. 你的浏览器运行的是 HTTP1.0，还是 HTTP1.1?你所访问的服务器所运行 HTTP 协议的版本号是多少?

```
219.217.226.15      172.20.38.38        HTTP    11508 HTTP/1.1 200 OK  (text/html)
172.20.38.38        219.217.226.15      HTTP      431 GET / HTTP/1.1
```

都是 HTTP1.1

2. 你的浏览器向服务器指出它能接收何种语言版本的对象?

```
Accept-Language: en,zh-CN;q=0.9,zh;q=0.8\r\n
```

en,zh-CN

3. 你的计算机的 IP 地址是什么?服务器 http://www.hit.edu.cn 的 IP 地址是多少?

```
219.217.226.15      172.20.38.38
172.20.38.38        219.217.226.15
```

本机 172.20.38.38

服务器 219.217.226.15

4. 从服务器向你的浏览器返回的状态代码是多少?

200

```
11508 HTTP/1.1 200 OK  (text/html)
```

3.1.2 HTTP 条件 GET/response 交互

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

http

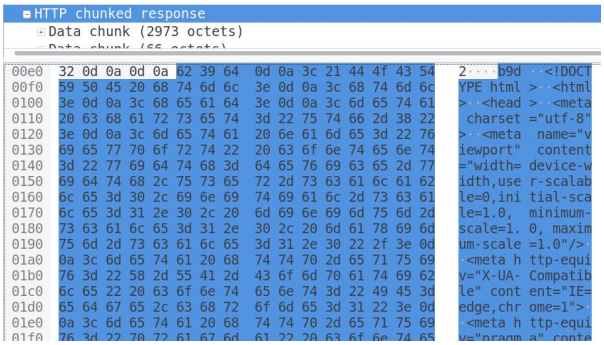
No.	Time	Source	Destination	Protocol	Length	Info
8	0.005645685	192.168.199.100	61.167.60.70	HTTP	510	GET / HTTP/1.1
88	0.021698503	61.167.60.70	192.168.199.100	HTTP	1245	HTTP/1.1 200 OK (text/html)
99	0.055405374	192.168.199.100	61.167.60.70	HTTP	479	GET /_visitcount?siteId=2&type=1&columnId=2 HTTP/1.1
100	0.061757987	61.167.60.70	192.168.199.100	HTTP	253	HTTP/1.1 200 OK
124	0.283005029	192.168.199.100	219.217.226.15	HTTP	531	GET /xxyw/main.htm HTTP/1.1
127	0.283022694	192.168.199.100	219.217.226.15	HTTP	531	GET /zhxw/main.htm HTTP/1.1
138	0.286447297	219.217.226.15	192.168.199.100	HTTP	1291	HTTP/1.1 200 OK (text/html)
144	0.286694065	219.217.226.15	192.168.199.100	HTTP	158	HTTP/1.1 200 OK (text/html)
150	0.328680497	192.168.199.100	61.135.186.152	HTTP	827	GET /v.gif?pid=307&type=3075&l=5333&t=0&s=5333&v=197&f=100
154	0.385398727	61.135.186.152	192.168.199.100	HTTP	362	HTTP/1.1 200 OK
166	0.818237482	192.168.199.100	219.217.226.15	HTTP	498	GET /_visitcount?siteId=20&type=2&columnId=1510 HTTP/1.1
168	0.821844607	219.217.226.15	192.168.199.100	HTTP	224	HTTP/1.1 200 OK

1. 分析你的浏览器向服务器发出的第一个 HTTP GET 请求的内容，在该请求报文中，是否有一行是:IF-MODIFIED-SINCE?

```
Hypertext Transfer Protocol
GET / HTTP/1.1\r\n
Host: www.hit.edu.cn\r\n
Cache-Control: max-age=0\r\n
Upgrade-Insecure-Requests: 1\r\n
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/66.0.3359.181 Safari/537.36\r\n
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8\r\n
Accept-Encoding: gzip, deflate\r\n
Accept-Language: en,zh-CN;q=0.9,zh;q=0.8\r\n
Cookie: JSESSIONID=E7045DC65419DCAC5A74EBDA4F59D243\r\n
Connection: close\r\n
\r\n
[Full request URI: http://www.hit.edu.cn/]
[HTTP request 1/1]
[Response in frame: 88]
```

没有 IF-MODIFIED-SINCE

2. 分析服务器响应报文的内容，服务器是否明确返回了文件的内容?如何获知?



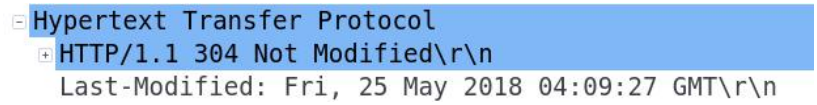
服务器返回了文件内容

通过状态码判断，如 304 表示不返回内容，如 200 表示返回内容

3. 分析你的浏览器向服务器发出的较晚的“HTTPGET”请求，在该请求报文中是否有一行是:IF-MODIFIED-SINCE?如果有，在该首部行后面跟着的信息是什么?

请求/js/hit2015.js 文件时有 IF-MODIFIED-SINCE，后面跟着的一般为上次响应中的 Last-Modified

4. 服务器对较晚的 HTTPGET 请求的响应中的 HTTP 状态代码是多少?服务器是否明确返回了文件的内容?请解释。

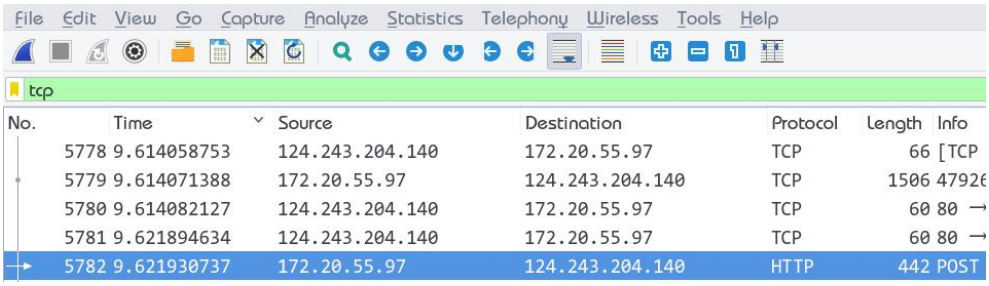


304，表示自从上次请求后，请求的网页未修改过

服务器返回此响应时，不会返回网页内容

3.2 TCP 分析

在 http://en.miui.com/home.php?mod=spacecp&ac=avatar 上传图片



1. 向服务器传送文件的客户端主机的 IP 地址和 TCP 端口号是多少?

```
Source: 172.20.55.97          Source Port: 47926
```

客户端 ip 172.20.55.97，tcp 端口 47926

2. 服务器的 IP 地址是多少?对这一连接，它用来发送和接收 TCP 报文的端口号是多少?

```
Destination: 124.243.204.140 Destination Port: 80
```

服务器 ip 124.243.204.140，tcp 端口 80

3. 客户服务器之间用于初始化 TCP 连接的 TCPSYN 报文段的序号（sequence number）是多少?在该报文段中，是用什么来标示该报文段是 SYN 报文段的?

```
Sequence number: 0    (relative sequenc Sequence number: 2585848389
[Next sequence number: 0    (relative s [Next sequence number: 2585848389]
Acknowledgment number: 0                Acknowledgment number: 0
1000 .... = Header Length: 32 bytes (8) 1000 .... = Header Length: 32 bytes (8)
Flags: 0x002 (SYN)                      Flags: 0x002 (SYN)
```

相对 sequence number 为 0（左图）

实际 sequence number 为 2585848389（右图）

```
Flags: 0x002 (SYN)
    000. .... .... = Reserved: Not set
    ...0 .... .... = Nonce: Not set
    .... 0... .... = Congestion Window Reduced (CWR): Not set
    .... .0.. .... = ECN-Echo: Not set
    .... ..0. .... = Urgent: Not set
    .... ...0 .... = Acknowledgment: Not set
    .... .... 0... = Push: Not set
    .... .... .0.. = Reset: Not set
    .... .... ..1. = Syn: Set
    .... .... ...0 = Fin: Not set
```

通过标志位中的 syn 置 1 来表示 SYN 报文段

4. 服务器向客户端发送的 SYNACK 报文段序号是多少?该报文段中，Acknowledgement 字段的值是多少?服务器是如何决定此值的?在该报文段中，是用什么来标示该报文段是 SYNACK 报文段的?

```
Sequence number: 1310483898
[Next sequence number: 1310483898]
Acknowledgment number: 2585848390
1000 .... = Header Length: 32 bytes
Flags: 0x012 (SYN, ACK)
```

seq number 为 1310483898

ack number 为 2585848390，即发送端 syn 包的 seq number+1

```
Flags: 0x012 (SYN, ACK)
  000. .... .... = Reserved: Not set
  ...0 .... .... = Nonce: Not set
  .... 0... .... = Congestion Window Re
  .... .0.. .... = ECN-Echo: Not set
  .... ..0. .... = Urgent: Not set
  .... ...1 .... = Acknowledgment: Set
```

通过标志位中的 ack 字段置 1 来表示 SYNACK 报文段

5. 你能从捕获的数据包中分析出 tcp 三次握手过程吗?

```
172.20.55.97        124.243.204.140     TCP     66 47926 → 80 [SYN] Seq=2585848389 Win=29200 Len=0 MSS=
124.243.204.140     172.20.55.97        TCP     66 80 → 47926 [SYN, ACK] Seq=1310483898 Ack=2585848390
172.20.55.97        124.243.204.140     TCP     54 47926 → 80 [ACK] Seq=2585848390 Ack=1310483899 Win=2
```

客户端向服务器发送 seq=2585848389 的 SYN 报文来请求建立连接

服务器向客户端返回 seq=1310483898,ack=2585848390 的 SYNACK 报文响应

客户端向服务器发送 seq=2585848390,ack=1310483899 的 ACK 报文完成建立

6. 包含 HTTP POST 命令的 TCP 报文段的序号是多少?

```
5782 9.621930737    172.20.55.97        124.243.204.140     HTTP    442 POST

me 5782: 442 bytes on wire (3536 bits), 442 bytes captured (3536 bits) on interface 0
ernet II, Src: IntelCor_9e:4a:4e (74:e5:f9:9e:4a:4e), Dst: RuijieNe_a5:e2:d3 (58:69:6c
ernet Protocol Version 4, Src: 172.20.55.97, Dst: 124.243.204.140
nsmission Control Protocol, Src Port: 47926, Dst Port: 80, Seq: 2587623330, Ack: 131048
```

Seq=2587623330，相对 1774941

7. 如果将包含 HTTP POST 命令的 TCP 报文段看作是 TCP 连接上的第一个报文段，那么该 TCP 连接上的第六个报文段的序号是多少?是何时发送的?该报文段所对应的 ACK 是何时接收的?

```
172.20.55.97        124.243.204.140     TCP     1506 47926 → 80 [ACK] Seq=1 Ack
172.20.55.97        124.243.204.140     TCP      650 47926 → 80 [PSH, ACK] Seq=
172.20.55.97        124.243.204.140     TCP     1506 47926 → 80 [ACK] Seq=2049
172.20.55.97        124.243.204.140     TCP     1506 47926 → 80 [ACK] Seq=3501
172.20.55.97        124.243.204.140     TCP     1506 47926 → 80 [ACK] Seq=4953
172.20.55.97        124.243.204.140     TCP     1506 47926 → 80 [ACK] Seq=6405
```

第六个报文段相对 Seq=6405

它在 http post 结束前，tcp 连接建立后发送，时间戳 429 1.246744452

对应 ack 如下，ack=6405

```
124.243.204.140     172.20.55.97        TCP     60 80 → 47926 [ACK] Seq=1 Ack=6405
```

时间戳 442 1.278457542

8. 前六个 TCP 报文段的长度各是多少?

```
[Frame: 424, payload: 0-1451 (1452 bytes)]
[Frame: 425, payload: 1452-2047 (596 bytes)]
[Frame: 426, payload: 2048-3499 (1452 bytes)]
[Frame: 427, payload: 3500-4951 (1452 bytes)]
[Frame: 428, payload: 4952-6403 (1452 bytes)]
[Frame: 429, payload: 6404-7855 (1452 bytes)]
```

长度如上图所示，如第一个为 1452 字节

9. 在整个跟踪过程中，接收端公示的最小的可用缓存空间是多少?限制发送端的传输以后，接收端的缓存是否仍然不够用?

```
Window size value: 29200
[Calculated window size: 29200]
```

通过 window size value 可知接收端公示的最小可用缓存空间是 29200 字节
该窗口大小会一直增加，不会出现接收端的缓存不够用问题

10. 在跟踪文件中是否有重传的报文段?进行判断的依据是什么?

有重传

```
1506 [TCP Fast Retransmission] 47926 → 80 [ACK] Seq=25281 A
1506 [TCP Retransmission] 47926 → 80 [ACK] Seq=29637 Ack=1
1506 [TCP Fast Retransmission] 47926 → 80 [ACK] Seq=153057
1506 [TCP Retransmission] 47926 → 80 [ACK] Seq=193713 Ack=1
1506 [TCP Retransmission] 47926 → 80 [ACK] Seq=190809 Ack=1
```

利用 tcp.analysis.retransmission 过滤得到如上的重传报文
通过序列号是否与已发送序列号重复来判断是否是重传

11. TCP 连接的 throughput(bytes transferred per unit time)是多少?请写出你的计算过程

```
[1323 Reassembled TCP Segments (1775328 bytes)
```

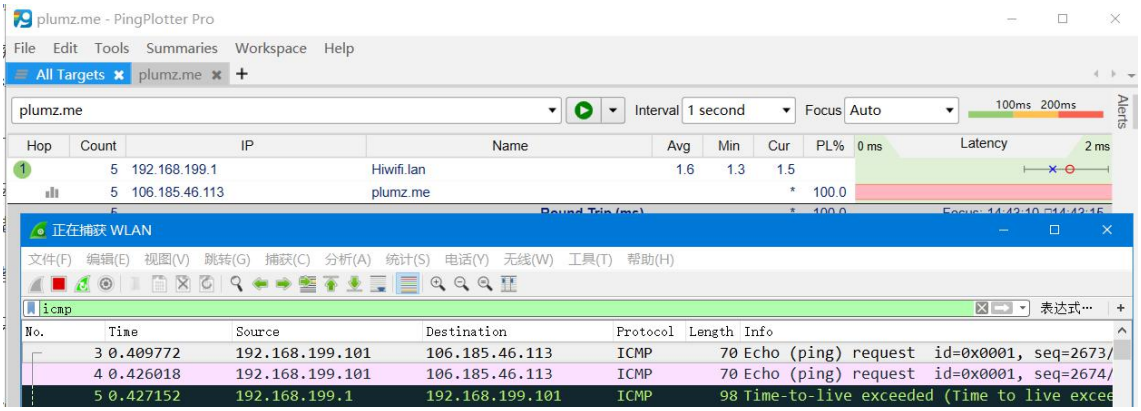
共发送 1775328 字节（1.69M）

```
346 0.000182529 172.20.55.97    124.243.204.140    TCP    66 47926 → 80
```

发送端握手后第一个包发送时间 346 1.210210431
最后一个包发送时间 662 0.000004476
时间总计 3159ms
吞吐量为 1775328B/3159ms=4.28Mbps

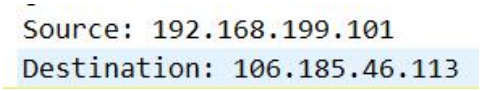
3.3 IP 分析

向 plumz.me 发 ping



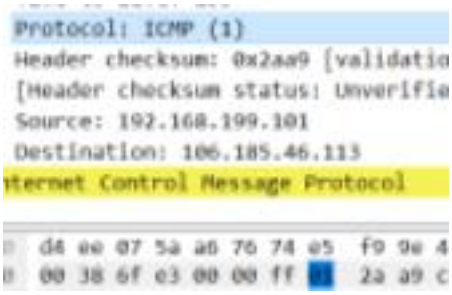
A. 选择第一个你的主机发出的 ICMP Echo Request 消息，在 packet details 窗口展开数据包的 Internet Protocol 部分

1. 你主机的 IP 地址是什么?



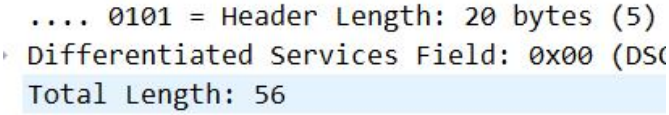
本机 IP 192.168.199.101

2. 在 IP 数据包头中，上层协议（upper layer）字段的值是什么?



上层协议字段的值是 01，表示 ICMP 协议

3. IP 头有多少字节?该 IP 数据包的净载为多少字节?并解释你是怎样确定该 IP 数据包的净载大小的?



IP 头有 20 字节，总长 56 字节

净载为 Total Length-Header Length=56B-20B=36B

4. 该 IP 数据包分片了吗?解释你是如何确定该 IP 数据包是否进行了分片

```
Flags: 0x0000
    0... .... .... .... = Reserved bit: Not set
    .0.. .... .... .... = Don't fragment: Not set
    ..0. .... .... .... = More fragments: Not set
    ...0 0000 0000 0000 = Fragment offset: 0
```

没有，因为 More fragments 为 0，且 Fragment offest 为 0，表示无分片

B. 单击 Source 列按钮，这样将对捕获的数据包按源 IP 地址排序。选择第一个你的主机发出的 ICMPEchoRequest 消息，在 packetdetails 窗口展开数据包的 InternetProtocol 部分。在“listingofcapturedpackets”窗口，你会看到许多后续的 ICMP 消息（或许还有你主机上运行的其他协议的数据包）

No.	Time	Source	Destination	Protocol	Length	Info
266	5.405732	192.168.199.101	106.185.46.113	ICMP	70	Echo (ping) request id=0x0001, seq=2922/
267	5.427361	192.168.199.101	106.185.46.113	ICMP	70	Echo (ping) request id=0x0001, seq=2923/
268	5.448845	192.168.199.101	106.185.46.113	ICMP	70	Echo (ping) request id=0x0001, seq=2924/
269	5.469296	192.168.199.101	106.185.46.113	ICMP	70	Echo (ping) request id=0x0001, seq=2925/
270	5.489479	192.168.199.101	106.185.46.113	ICMP	70	Echo (ping) request id=0x0001, seq=2926/
271	5.509831	192.168.199.101	106.185.46.113	ICMP	70	Echo (ping) request id=0x0001, seq=2927/
272	5.530223	192.168.199.101	106.185.46.113	ICMP	70	Echo (ping) request id=0x0001, seq=2928/

1. 你主机发出的一系列 ICMP 消息中 IP 数据报中哪些字段总是发生改变?

   id，ttl，header checksun

2. 哪些字段必须保持常量?哪些字段必须改变?为什么?

   id 必须改变:identification 是标识，用于区分不同的数据包

   ttl 必须改变:来自于 traceroute 的要求，用来测试路径上的路由信息

   header checksum 必须改变:首部校验和，由于 id 与 ttl 字段变化，需要重新计算 header checksum

   其他字段保持常量

3. 描述你看到的 IP 数据包 Identification 字段值的形式

   16 位，+1 递增，如下图所示

```
Identification: 0x0a97 (2711)
Identification: 0x0a98 (2712)
```

C. 找到由最近的路由器（第一跳）返回给你主机的 ICMPTime-to-liveexceeded 消息

27	4.668007	192.168.199.1	192.168.199.101	ICMP	98 Time-to-live exceeded (Time to live exceeded in transit)

1. Identification 字段和 TTL 字段的值是什么?

```
Identification: 0x3c12 (15378)
Flags: 0x0000
Time to live: 64
```

2. 最近的路由器（第一跳）返回给你主机的 ICMPTime-to-liveexceeded 消息中这些值是否保持不变?为什么?

不变，因为给同一个主机返回的 ICMP 报文的标识不代表序号，所以 TTL 消息是相同的，因此 Identification 也不变

D. 单击 Time 列按钮，这样将对捕获的数据包按时间排序。找到在将包大小改为 2000 字节后你的主机发送的第一个 ICMP Echo Request 消息

1. 该消息是否被分解成不止一个 IP 数据报?

```
Source: 192.168.199.101
Destination: 106.185.46.113
[2 IPv4 Fragments (1980 bytes): #51(1480), #52(500)]
  [Frame: 51, payload: 0-1479 (1480 bytes)]
  [Frame: 52, payload: 1480-1979 (500 bytes)]
```

是，分成了两个

2. 观察第一个 IP 分片，IP 头部的哪些信息表明数据包被进行了分片?IP 头部的哪些信息表明数据包是第一个而不是最后一个分片?该分片的长度是多少

```
Total Length: 1500
Identification: 0x0b98 (2968)
Flags: 0x2000, More fragments
  0... .... .... .... = Reserved bit: Not set
  .0.. .... .... .... = Don't fragment: Not set
  ..1. .... .... .... = More fragments: Set
  ...0 0000 0000 0000 = Fragment offset: 0
```

MF=1 或位移>0 表示进行了分片

MF=1 且位移=0 表示分片是第一个而不是最后一个

该分片的长度是 1500B（total length）

E. 找到在将包大小改为 3500 字节后你的主机发送的第一个 ICMP Echo Request 消息

1. 原始数据包被分成了多少片?

```
[3 IPv4 Fragments (3480 bytes): #800(1480), #801(1480), #802(520)]
  [Frame: 800, payload: 0-1479 (1480 bytes)]
  [Frame: 801, payload: 1480-2959 (1480 bytes)]
  [Frame: 802, payload: 2960-3479 (520 bytes)]
  [Fragment count: 3]
```

分成了三片

2. 这些分片中 IP 数据报头部哪些字段发生了变化?

```
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0,
Total Length: 1500
Identification: 0x4b2d (19245)
Flags: 0x2000, More fragments
  0... .... .... .... = Reserved bit: Not set
  .0.. .... .... .... = Don't fragment: Not set
  ..1. .... .... .... = More fragments: Set
  ...0 0000 0000 0000 = Fragment offset: 0
Time to live: 255
Protocol: ICMP (1)
Header checksum: 0xce53 [validation disabled]
```

```
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0
Total Length: 1500
Identification: 0x4b2d (19245)
Flags: 0x20b9, More fragments
  0... .... .... .... = Reserved bit: Not set
  .0.. .... .... .... = Don't fragment: Not se
  ..1. .... .... .... = More fragments: Set
  ...0 0000 1011 1001 = Fragment offset: 185
Time to live: 255
Protocol: ICMP (1)
Header checksum: 0xcd9a [validation disabled]
```

```
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0,
Total Length: 540
Identification: 0x4b2d (19245)
Flags: 0x0172
  0... .... .... .... = Reserved bit: Not set
  .0.. .... .... .... = Don't fragment: Not set
  ..0. .... .... .... = More fragments: Not set
  ...0 0001 0111 0010 = Fragment offset: 370
Time to live: 255
Protocol: ICMP (1)
Header checksum: 0xf0a1 [validation disabled]
```

从左到右为三个分片

前 2 个分片 MF=1，第 3 个为 0

前 2 个分片 total length=1500，第 3 个为 540

片偏移与 header checksum 也是变化的，其他不变

3.4 ARP 数据包

3.4.1 查看 ARP 缓存

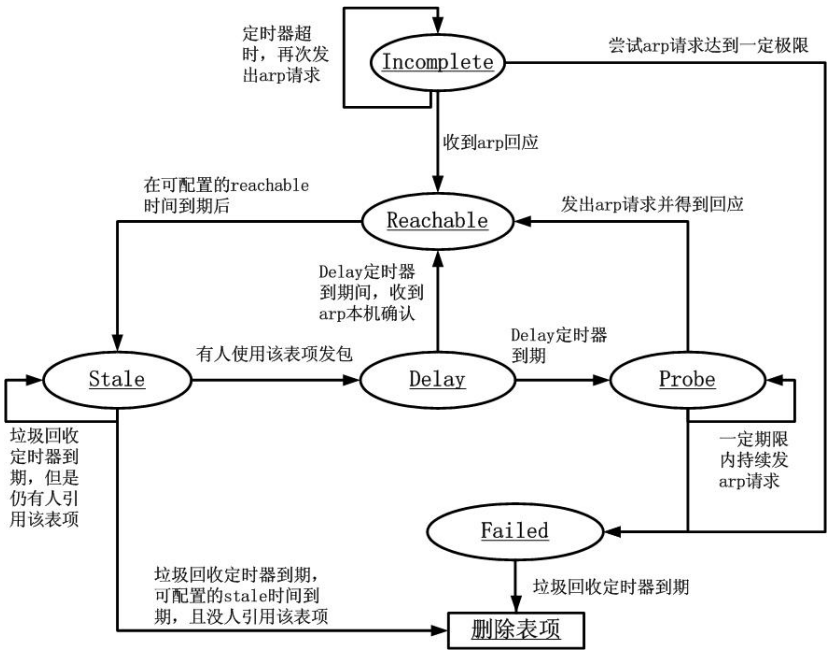
用 ip neigh 或 arp（已弃用）查看 ARP 表

```
~ ip neigh show
192.168.199.215 dev wlp2s0 lladdr e4:e4:ab:5d:7a:6f STALE
192.168.199.206 dev wlp2s0 lladdr a8:1b:5a:15:27:2a STALE
192.168.199.1 dev wlp2s0 lladdr d4:ee:07:5a:a6:76 REACHABLE
fe80::2413:466:4d:e8fe dev wlp2s0 lladdr 9c:e8:2b:2c:53:35 STALE
fe80::5a69:6cff:fea5:e2d3 dev wlp2s0 lladdr 58:69:6c:a5:e2:d3 router STAL
E
fe80::2ae:faff:fed0:8ffc dev wlp2s0 lladdr 00:ae:fa:d0:8f:fc STALE
```

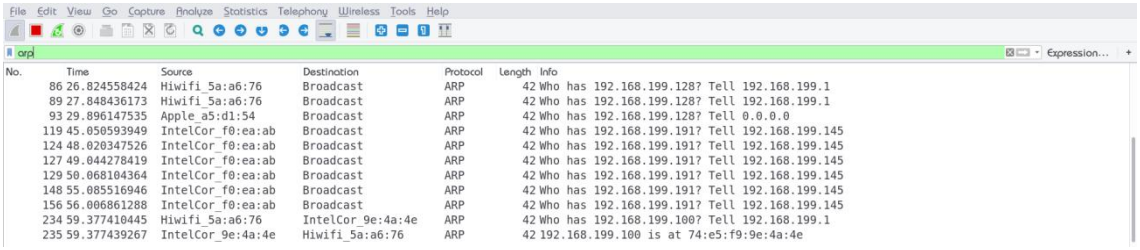
从左至右为 ip（4/6），接口（网卡）名，链路层地址，状态

如第 1 条 stale 表示有效但可疑（已超时），第 3 条 reachable 表示有效且可信任（未超时）

它们的状态转换如下（https://blog.csdn.net/dog250/article/details/7251689）



3.4.2 分析数据包



1. ARP 数据包的格式是怎样的?由几部分构成，各个部分所占的字节数是多少?

ARP 数据包格式如下图:

```
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_f0:ea:ab (f4:96:34:f0:ea:ab)
  Sender IP address: 192.168.199.145
  Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
  Target IP address: 192.168.199.112
```

如上图所示，由 9 部分构成，分别是硬件类型（2 字节），协议类型（2 字节），硬件地址长度（1 字节），协议地址长度（1 字节），OP（2 字节)，发送端 MAC 地址（6 字节），发送端 IP 地址（4 字节），目的 MAC 地址（6 字节），目的 IP 地址（4 字节）

2. 如何判断一个 ARP 数据是请求包还是应答包?

```
Address Resolution Protocol (request)
  Hardware type: Ethernet (1)
  Protocol type: IPv4 (0x0800)
  Hardware size: 6
  Protocol size: 4
  Opcode: request (1)
```

```
Address Resolution Protocol (reply)
  Hardware type: Ethernet (1)
  Protocol type: IPv4 (0x0800)
  Hardware size: 6
  Protocol size: 4
  Opcode: reply (2)
```

OP 字段值为 0x0001 时是请求包，为 0x0002 时是应答包

左图为请求包，右图为应答包

3. 为什么 ARP 查询要在广播帧中传送，而 ARP 响应要在一个有着明确目的局域网地址的帧中传送?

ARP 查询时不知道目的 IP 地址对应的 MAC 地址，所以要广播

ARP 响应报文知道查询主机的 MAC 地址（通过查询主机发出的查询报文获得），且局域网中的其他主机不需要此次查询的结果，所以要在一个有着明确目的局域网地址的帧中传送

3.5 UDP 数据包

```
172.20.55.97        182.254.110.91      UDP      489 4027 → 8000 Len=447
172.20.55.97        182.254.110.91      UDP      497 4027 → 8000 Len=455
172.20.55.97        182.254.110.91      UDP      497 4027 → 8000 Len=455
172.20.55.97        182.254.110.91      UDP      513 4027 → 8000 Len=471
172.20.55.97        182.254.110.91      UDP      513 4027 → 8000 Len=471
```

IP地址: 182.254.110.91 广东省广州市 腾讯云

1. 消息是基于 UDP 的还是 TCP 的?

```
Protocol: UDP (17)
```

UDP

2. 你的主机 IP 地址是什么?目的主机 IP 地址是什么?

```
Source: 172.20.55.97
Destination: 182.254.110.91
```

本机 IP:172.20.55.97

目的 IP:182.254.110.91

3. 你的主机发送 QQ 消息的端口号和 QQ 服务器的端口号分别是多少?

```
Source Port: 4027
Destination Port: 8000
```

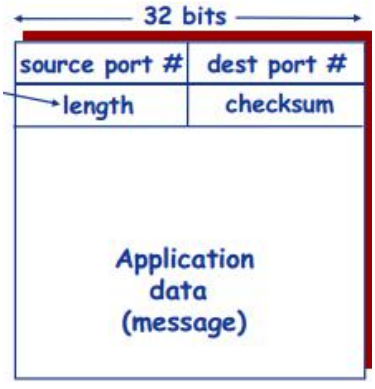
客户端端口号:4027

服务器端口号:8000

4. 数据报的格式是什么样的?都包含哪些字段，分别占多少字节?

UDP 数据报格式如下图:

```
User Datagram Protocol, Src Port
    Source Port: 4027
    Destination Port: 8000
    Length: 455
    Checksum: 0x6760 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 17]
Data (447 bytes)
    Data: 02372503fb06118d20d53e
    [Length: 447]
```



由 5 部分构成，分别是源端口号（4 字节），目的端口号（4 字节），长度（4 字节），校验和（4 字节）和应用层数据

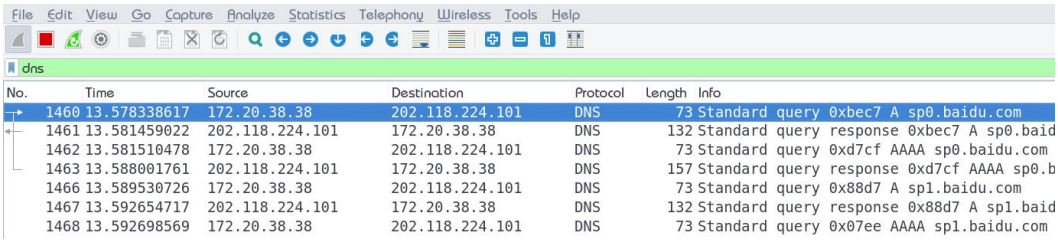
5. 为什么你发送一个 ICQ 数据包后，服务器又返回给你的主机一个 ICQ 数据包?这 UDP 的不可靠数据传输有什么联系?对比前面的 TCP 协议分析，你能看出 UDP 是无连接的吗?

服务器返回一个 ICQ 是起到 ACK 的作用，因为 UDP 是无连接的，所以服务器返回客户端一个 ICO 来确认收到

可以看出 UDP 是无连接的，因为 UDP 数据包没有序列号，每次只发送一个数据报，然后等待服务器响应，没有像 TCP 一样的握手过程

3.6 DNS 协议分析

1. 打开浏览器，输入 www.baidu.com,DNS 查询消息如下图:



本机 IP 172.20.38.38，本地域名服务器 IP 202.118.224.101

2. UDP 报文的源端口号 46896，目的端口号 53

```
User Datagram Protocol,
  Source Port: 46896
  Destination Port: 53
```

3. DNS 查询报文内容如下

```
Domain Name System (query)
  Transaction ID: 0xbec7
  Flags: 0x0100 Standard query
  Questions: 1
  Answer RRs: 0
  Authority RRs: 0
  Additional RRs: 0
  Queries
    sp0.baidu.com: type A, class IN
```

表示查询 sp0.baidu.com 的 IP

4. DNS 回复信息

```
Domain Name System (response)
  Transaction ID: 0xbec7
  Flags: 0x8180 Standard query response, No error
  Questions: 1
  Answer RRs: 3
  Authority RRs: 0
  Additional RRs: 0
  Queries
    sp0.baidu.com: type A, class IN
  Answers
    sp0.baidu.com: type CNAME, class IN, cname www.a.shifen.com
    www.a.shifen.com: type A, class IN, addr 119.75.216.20
    www.a.shifen.com: type A, class IN, addr 119.75.213.61
```

这是一条 cname 记录，表示 sp0.baidu.com 被映射为 www.a.shifen.com