

# 哈尔滨工业大学

## <<计算机网络>>

### 实验报告

(2018 年度春季学期)

姓名:	
学号:	
学院:	计算机科学与技术学院
教师:	

## 实验五 利用 Wireshark 进行协议分析

### 一、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。

### 二、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

## 三、实验过程及结果

### 1. HTTP 分析

#### 1.1 HTTP GET/response 交互

No.	Time	Source	Destination	Protocol	Length	Info
1789	6.454194	172.20.26.104	45.32.56.226	HTTP	587	GET / HTTP/1.1
1802	6.4607025	45.32.56.226	172.20.26.104	HTTP	1527	HTTP/1.1 200 OK (text/html)
1820	6.633551	172.20.26.104	45.32.56.226	HTTP	548	GET /css/bootstrap.min.css HTTP/1.1
1821	6.637104	172.20.26.104	45.32.56.226	HTTP	547	GET /css/hux-blog.min.css HTTP/1.1
1864	6.793447	172.20.26.104	45.32.56.226	HTTP	563	GET /js/google-code-prettify/prettify.css HTTP
1885	6.816259	172.20.26.104	45.32.56.226	HTTP	582	GET /js/google-code-prettify/tomorrow-night-ei
1886	6.816408	172.20.26.104	45.32.56.226	HTTP	541	GET /css/search.css HTTP/1.1
1887	6.816558	172.20.26.104	45.32.56.226	HTTP	528	GET /js/jquery.min.js HTTP/1.1
1913	6.938993	45.32.56.226	172.20.26.104	HTTP	1026	HTTP/1.1 200 OK (text/css)
1915	6.944011	172.20.26.104	45.32.56.226	HTTP	524	GET /js/search.js HTTP/1.1
1935	6.967412	45.32.56.226	172.20.26.104	HTTP	1002	HTTP/1.1 200 OK (text/css)
1943	6.973134	45.32.56.226	172.20.26.104	HTTP	1016	HTTP/1.1 200 OK (text/css)
1968	6.998640	45.32.56.226	172.20.26.104	HTTP	354	HTTP/1.1 200 OK (text/css)
1986	7.096513	45.32.56.226	172.20.26.104	HTTP	1053	HTTP/1.1 200 OK (application/javascript)
2240	7.593187	172.20.26.104	45.32.56.226	HTTP	531	GET /js/bootstrap.min.js HTTP/1.1
2294	7.683438	45.32.56.226	172.20.26.104	HTTP	152	HTTP/1.1 200 OK (text/css)
2295	7.692983	172.20.26.104	45.32.56.226	HTTP	530	GET /js/hux-blog.min.js HTTP/1.1

Frame 1789: 587 bytes on wire (4696 bits), 587 bytes captured (4696 bits) on interface 0  
Ethernet II, Src: HonHaiPr\_2e:4f:e9 (ac:d1:b8:2e:4f:e9), Dst: FujianRu\_a5:e2:d3 (58:69:6c:a5:e2:d3)  
Internet Protocol Version 4, Src: 172.20.26.104, Dst: 45.32.56.226  
Transmission Control Protocol, Src Port: 6177 (6177), Dst Port: 80 (80), Seq: 1, Ack: 1, Len: 533  
Hypertext Transfer Protocol  
GET / HTTP/1.1\r\nHost: www.softmargin.com\r\nConnection: keep-alive\r\nUpgrade-Insecure-Requests: 1\r\nUser-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/66.0.3359.132 Safari/537.36\r\nAccept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,\*/\*;q=0.8\r\nAccept-Encoding: gzip, deflate\r\nAccept-Language: zh-CN,zh;q=0.9,en;q=0.8,zh-TW;q=0.7\r\nCookie: Hm\_lvt\_5580aadc4c87d87d220209609e278ef3=1514715188; bdshare\_firsttime=1514724030754; \_ga=GA1.3.175783471.1514715188\r\n[Full request URI: http://www.softmargin.com/]\r\n[HTTP request 1/3]

问题:

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?  
都为 HTTP 1.1
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?  
zh-CN,zh;,en;,zh-TW;
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?  
本地计算机 IP: 172.20.26.104  
服务器 IP: 45.32.56.226
4. 从服务器向你的浏览器返回的状态代码是多少?  
200

## 1.2 HTTP 条件 GET/response 交互

No.	Time	Source	Destination	Protoc	Length	Info
1789	6.454194	172.20.26.104	45.32.56.226	HTTP	587	GET / HTTP/1.1
1819	6.607613	45.32.56.226	172.20.26.104	HTTP	1527	HTTP/1.1 200 OK (text/html)
1820	6.633551	172.20.26.104	45.32.56.226	HTTP	548	GET /css/bootstrap.min.css HTTP/1.1
1821	6.637104	172.20.26.104	45.32.56.226	HTTP	547	GET /css/hux-blog.min.css HTTP/1.1
1864	6.793447	172.20.26.104	45.32.56.226	HTTP	563	GET /js/google-code-prettify/prettify.css HTTP/1.1
1885	6.816259	172.20.26.104	45.32.56.226	HTTP	582	GET /js/google-code-prettify/tomorrow-night.css HTTP/1.1
1886	6.816408	172.20.26.104	45.32.56.226	HTTP	541	GET /css/search.css HTTP/1.1
1887	6.816558	172.20.26.104	45.32.56.226	HTTP	528	GET /js/jquery.min.js HTTP/1.1
1913	6.938993	45.32.56.226	172.20.26.104	HTTP	1026	HTTP/1.1 200 OK (text/css)
1915	6.944011	172.20.26.104	45.32.56.226	HTTP	524	GET /js/search.js HTTP/1.1
1935	6.967412	45.32.56.226	172.20.26.104	HTTP	1002	HTTP/1.1 200 OK (text/css)
1943	6.973134	45.32.56.226	172.20.26.104	HTTP	1016	HTTP/1.1 200 OK (text/css)
1968	6.998640	45.32.56.226	172.20.26.104	HTTP	354	HTTP/1.1 200 OK (text/css)
1986	7.096513	45.32.56.226	172.20.26.104	HTTP	1053	HTTP/1.1 200 OK (application/javascript)
2240	7.593187	172.20.26.104	45.32.56.226	HTTP	531	GET /js/bootstrap.min.js HTTP/1.1
2294	7.683438	45.32.56.226	172.20.26.104	HTTP	152	HTTP/1.1 200 OK (text/css)
2295	7.692983	172.20.26.104	45.32.56.226	HTTP	530	GET /js/hux-blog.min.js HTTP/1.1

Frame 1789: 587 bytes on wire (4696 bits), 587 bytes captured (4696 bits) on interface 0
Ethernet II, Src: HonHaiPr_2e:4f:e9 (ac:d1:b8:2e:4f:e9), Dst: FujianRu_a5:e2:d3 (58:69:6c:a5:e2:d3)
Internet Protocol Version 4, Src: 172.20.26.104, Dst: 45.32.56.226
Transmission Control Protocol, Src Port: 6177 (6177), Dst Port: 80 (80), Seq: 1, Ack: 1, Len: 533
Hypertext Transfer Protocol
GET / HTTP/1.1\r\n
Host: www.softmargin.com\r\n
Connection: keep-alive\r\n
Upgrade-Insecure-Requests: 1\r\n
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/66.0.3359.131 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: zh-CN,zh;q=0.9,en;q=0.8,zh-TW;q=0.7\r\n
Cookie: Hm_lvt_5580aadc4c87d87d2202096d9e278ef3=1514715188; bdshare_firsttime=1514724030754; _ga=GA1.3.175783471.1514715188\r\n
[Full request URI: http://www.softmargin.com/]

- 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？  
并没有
- 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？  
确实返回了文件内容  
根据服务器返回报文的响应码可判断是否返回了文件内容，也可直接查看返回报文是否含文件内容  
HTTP Status Code 为 304 时不返回文件  
HTTP Status Code 为 200 时返回文件
- 分析你的浏览器向服务器发出的较晚的“HTTP GET”请求，在该请求报文中是否有一行是：IF-MODIFIED-SINCE？如果有，在该首部后面跟着的信息是什么？

```

GET / HTTP/1.1\r\n
Host: www.softmargin.com\r\n
Connection: keep-alive\r\n
Cache-Control: max-age=0\r\n
Upgrade-Insecure-Requests: 1\r\n
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/66.0.3359.131 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: zh-CN,zh;q=0.9,en;q=0.8,zh-TW;q=0.7\r\n
Cookie: Hm_lvt_5580aadc4c87d87d2202096d9e278ef3=1514715188; bdshare_firsttime=1514724030754; _ga=GA1.3.175783471.1514715188\r\n
If-None-Match: W/"5a6f2948-794c"\r\n
If-Modified-Since: Mon, 29 Jan 2018 14:01:44 GMT\r\n
\r\n
[Full request URI: http://www.softmargin.com/]

```

存在 IF-Modifide-Since,后面跟着请求文件在本地缓存中的更改时间。

- 服务器对较晚的 HTTP GET 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。



Time	Source	Destination	Protocol	Length	Info
428.2.893421	172.20.26.104	45.32.56.226	HTTP	836	GET / HTTP/1.1
471.3.036613	45.32.56.226	172.20.26.104	HTTP	244	HTTP/1.1 304 Not Modified
473.3.046794	172.20.26.104	202.108.23.152	HTTP	874	GET /hm.gif?cc=0&ck=1&cl=24-bit&ds=1536x864&v1=
490.3.075645	202.108.23.152	172.20.26.104	HTTP	310	HTTP/1.1 200 OK (GIF89a) (GIF89a) (image/gif)
491.3.086191	172.20.26.104	45.32.56.226	HTTP	551	GET /img/icon_wechat.png HTTP/1.1
515.3.185871	172.20.26.104	202.108.23.152	HTTP	774	GET /hm.js?3747566a81b32444a5bb7052b11946c3 HT
533.3.227647	172.20.26.104	111.206.37.189	HTTP	788	GET /s.gif?l=http://www.softmargin.com/ HTTP/1.
535.3.228630	45.32.56.226	172.20.26.104	HTTP	458	HTTP/1.1 404 Not Found (text/html)
538.3.244368	202.108.23.152	172.20.26.104	HTTP	219	HTTP/1.1 304 Not Modified
551.3.291105	172.20.26.104	202.108.23.152	HTTP	921	GET /hm.gif?cc=0&ck=1&cl=24-bit&ds=1536x864&v1=
553.3.292501	172.20.26.104	202.108.23.152	HTTP	939	GET /hm.gif?cc=0&ck=1&cl=24-bit&ds=1536x864&v1=
554.3.292771	172.20.26.104	111.206.37.189	HTTP	788	GET /s.gif?l=http://www.softmargin.com/ HTTP/1.
562.3.328040	202.108.23.152	172.20.26.104	HTTP	310	HTTP/1.1 200 OK (GIF89a) (GIF89a) (image/gif)
563.3.328042	202.108.23.152	172.20.26.104	HTTP	310	HTTP/1.1 200 OK (GIF89a) (GIF89a) (image/gif)
569.3.359840	172.20.26.104	111.206.37.189	HTTP	788	GET /s.gif?l=http://www.softmargin.com/ HTTP/1.
597.3.475817	172.20.26.104	111.206.37.189	HTTP	788	GET /s.gif?l=http://www.softmargin.com/ HTTP/1.
708.4.316870	172.20.26.104	202.108.23.152	HTTP	1001	GET /hm.gif?cc=0&ck=1&cl=24-bit&ds=1536x864&v1=

```

GET / HTTP/1.1\r\n
Host: www.softmargin.com\r\n
Connection: keep-alive\r\n
Cache-Control: max-age=0\r\n
Upgrade-Insecure-Requests: 1\r\n
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/66.0.3359.131 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: zh-CN,zh;q=0.9,en;q=0.8,zh-TW;q=0.7\r\n
[truncated]Cookie: Hm_lvt_5580aad4c87d87d202096d9e278ef3=1514715188; bdshare_firsttime=1514724030754; _ga=GA133145951.1514724030754.1514724030754.1514724030754.1514724030754\r\n
If-None-Match: w/"5a6f2948-794c"\r\n
If-Modified-Since: Mon, 29 Jan 2018 14:01:44 GMT\r\n
\r\n
Full request URL: http://www.softmargin.com/

```

304, 没有返回明确文件, 由于本地缓存存有该文件, 并且该文件与服务器的一致, 服务器此文件并没由更新。

## 2 TCP 分析

俘获大量的由本地主机到远程服务器的 TCP 分组:

<p>← → ↺ ↻ ⓘ gaia.cs.umass.edu/wireshark-labs/lab3-1-reply.htm</p> <p>应用 百度一下, 你就知道 HIT MOOC ML Tool Linux DataBase English Web Algorithms Kit Sever 考研 » 知</p> <p>Congratulations!</p> <p>You've now transferred a copy of alice.txt from your computer to gaia.cs.umass.edu. You should now stop Wireshark packet capture. It's time to start analyzing the captured Wireshark packets!</p>					
No.	Time	Source	Destination	Protocol	Length Info
1...	1.7368...	172.20.26.104	128.119.245.12	TCP	54 7133 → 80 [FIN, ACK] Seq=1 Ack=1 Wi...
1...	1.7369...	172.20.26.104	202.108.23.152	TCP	54 7130 → 80 [FIN, ACK] Seq=1 Ack=1 Wi...
1...	1.7381...	172.20.26.104	128.119.245.12	TCP	66 7136 → 80 [SYN] Seq=0 Win=8192 Len=...
1...	1.9823...	128.119.245.12	172.20.26.104	TCP	66 80 → 7136 [SYN, ACK] Seq=0 Ack=1 Wi...
1...	1.9826...	172.20.26.104	128.119.245.12	TCP	54 7136 → 80 [ACK] Seq=1 Ack=1 Win=163...
1...	1.9838...	172.20.26.104	128.119.245.12	TCP	736 [TCP segment of a reassembled PDU]
1...	1.9844...	172.20.26.104	128.119.245.12	TCP	1514 [TCP segment of a reassembled PDU]
1...	1.9845...	172.20.26.104	128.119.245.12	TCP	1514 [TCP segment of a reassembled PDU]
1...	1.9845...	172.20.26.104	128.119.245.12	TCP	1514 [TCP segment of a reassembled PDU]
1...	1.9845...	172.20.26.104	128.119.245.12	TCP	1514 [TCP segment of a reassembled PDU]
1...	1.9846...	172.20.26.104	128.119.245.12	TCP	1514 [TCP segment of a reassembled PDU]
<p>&gt; Frame 147: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0</p> <p>&gt; Ethernet II, Src: HonHaiPr_2e:4f:e9 (ac:d1:b8:2e:4f:e9), Dst: FujianRu_a5:e2:d3 (58:69:6...</p> <p>&gt; Internet Protocol Version 4, Src: 172.20.26.104, Dst: 128.119.245.12</p> <p>&gt; Transmission Control Protocol, Src Port: 7136 (7136), Dst Port: 80 (80), Seq: 0, Len: 0</p> <p>Source Port: 7136</p> <p>Destination Port: 80</p> <p>[Stream index: 2]</p> <p>[TCP Segment Len: 0]</p> <p>Sequence number: 0 (relative sequence number)</p> <p>Acknowledgment number: 0</p> <p>Header Length: 32 bytes</p>					

1. 向 gaia.cs.umass.edu 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？

IP : 172.20.26.104

port : 7136

2. Gaia.cs.umass.edu 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？

IP : 128.119.245.12

port : 80

3. 客户服务器之间用于初始化 TCP 连接的 TCP SYN 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？

1...	1.7381...	172.20.26.104	128.119.245.12	TCP	66	7136 → 80	[SYN]	Seq=0	Win=8192	Len=...
1...	1.9823...	128.119.245.12	172.20.26.104	TCP	66	80 → 7136	[SYN, ACK]	Seq=0	Ack=1	Wi...
1...	1.9826...	172.20.26.104	128.119.245.12	TCP	54	7136 → 80	[ACK]	Seq=1	Ack=1	Win=163...
1...	1.9838...	172.20.26.104	128.119.245.12	TCP	736			[TCP segment of a reassembled PDU]		
1...	1.9844...	172.20.26.104	128.119.245.12	TCP	1514			[TCP segment of a reassembled PDU]		
1...	1.9845...	172.20.26.104	128.119.245.12	TCP	1514			[TCP segment of a reassembled PDU]		
1...	1.9845...	172.20.26.104	128.119.245.12	TCP	1514			[TCP segment of a reassembled PDU]		
1...	1.9845...	172.20.26.104	128.119.245.12	TCP	1514			[TCP segment of a reassembled PDU]		
1...	1.9846...	172.20.26.104	128.119.245.12	TCP	1514			[TCP segment of a reassembled PDU]		

[Stream index: 2]

[TCP Segment Len: 0]

Sequence number: 0 (relative sequence number)

Acknowledgment number: 0

Header Length: 32 bytes

✓ 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

序号为 0，Flags 字段第 11 位表示 SYN 段

4. 服务器向客户端发送的 SYN ACK 报文段序号是多少？该报文段中，Acknowledgement 字段的值是多少？Gaia.cs.umass.edu 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？

1...	1.7368...	172.20.26.104	128.119.245.12	TCP	54	7133 → 80	[FIN, ACK]	Seq=1 Ack=1 Wi...
1...	1.7369...	172.20.26.104	202.108.23.152	TCP	54	7130 → 80	[FIN, ACK]	Seq=1 Ack=1 Wi...
✓ 1...	1.7381...	172.20.26.104	128.119.245.12	TCP	66	7136 → 80	[SYN]	Seq=0 Win=8192 Len=...
1...	1.9823...	128.119.245.12	172.20.26.104	TCP	66	80 → 7136	[SYN, ACK]	Seq=0 Ack=1 Wi...
1...	1.9826...	172.20.26.104	128.119.245.12	TCP	54	7136 → 80	[ACK]	Seq=1 Ack=1 Win=163...
1...	1.9838...	172.20.26.104	128.119.245.12	TCP	736	[TCP segment of a reassembled PDU]		
1...	1.9844...	172.20.26.104	128.119.245.12	TCP	1514	[TCP segment of a reassembled PDU]		
1...	1.9845...	172.20.26.104	128.119.245.12	TCP	1514	[TCP segment of a reassembled PDU]		
1...	1.9845...	172.20.26.104	128.119.245.12	TCP	1514	[TCP segment of a reassembled PDU]		
1...	1.9845...	172.20.26.104	128.119.245.12	TCP	1514	[TCP segment of a reassembled PDU]		
1...	1.9846...	172.20.26.104	128.119.245.12	TCP	1514	[TCP segment of a reassembled PDU]		

[Stream index: 2]

[TCP Segment Len: 0]

Sequence number: 0 (relative sequence number)

Acknowledgment number: 1 (relative ack number)

Header Length: 32 bytes

▼ Flags: 0x012 (SYN, ACK)

000. .... = Reserved: Not set

...0 .... = Nonce: Not set

.... 0... = Congestion Window Reduced (CWR) Not set

.... .0.. = ECN-Echo: Not set

.... ..0. = Urgent: Not set

.... ...1 = Acknowledgment: Set

.... .... 0... = Push: Not set

.... .... .0.. = Reset: Not set

> .... .... ..1. = Syn: Set

.... .... ...0 = Fin: Not set

SYNACK 报文段序号为 0, Acknowledgement 字段值为 1, ack 是根据上一次客户端发给服务器的 seq+1 得到的, Flags 字段第 8 位为 1 表示 ACK, 第 11 位为 1 表示 SYN

### 5. 能从捕获的数据包中分析出 tcp 三次握手过程吗?

172.20.26.104	128.119.245.12	TCP	66	7136 → 80	[SYN]	Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SA
128.119.245.12	172.20.26.104	TCP	66	80 → 7136	[SYN, ACK]	Seq=0 Ack=1 Win=29200 Len=0 MSS=14
172.20.26.104	128.119.245.12	TCP	54	7136 → 80	[ACK]	Seq=1 Ack=1 Win=16384 Len=0

首先客户端向服务器发送 seq=0 的建立连接的请求

然后服务器向客户端返回 seq=0,ack=0+1=1 的响应

客户端收到响应, 返回 seq=1,ack=0+1=1 的确认报文, 连接建立

### 6. 包含 HTTP POST 命令的 TCP 报文段的序号是多少?

No.	Time	Source	Destination	Protoc	Lengt	Info
9...	7.2179...	172.20.26.104	128.119.245.12	HTTP	1313	POST /wireshark-labs/lab3-1-reply.h...
1...	8.5324...	128.119.245.12	172.20.26.104	HTTP	831	HTTP/1.1 200 OK (text/html)
10	0.1034...	172.20.5.141	239.255.255.2...	SSDP	143	M-SEARCH * HTTP/1.1
11	0.2050...	172.20.64.192	239.255.255.2...	SSDP	171	M-SEARCH * HTTP/1.1
20	0.2056...	172.20.4.28	239.255.255.2...	SSDP	216	M-SEARCH * HTTP/1.1
30	0.5125...	172.20.101.233	239.255.255.2...	SSDP	216	M-SEARCH * HTTP/1.1
1...	1.5373...	172.20.71.113	239.255.255.2...	SSDP	216	M-SEARCH * HTTP/1.1
1...	1.6386...	172.20.71.113	239.255.255.2...	SSDP	216	M-SEARCH * HTTP/1.1
1...	1.6391...	172.20.71.113	239.255.255.2...	SSDP	216	M-SEARCH * HTTP/1.1
1...	1.7416...	172.20.92.79	239.255.255.2...	SSDP	317	NOTIFY * HTTP/1.1
1...	1.8498...	172.20.8.235	239.255.255.2...	SSDP	179	M-SEARCH * HTTP/1.1

> Frame 938: 1313 bytes on wire (10504 bits), 1313 bytes captured (10504 bits) on interfac...  
 > Ethernet II, Src: HonHaiPr\_2e:4f:e9 (ac:d1:b8:2e:4f:e9), Dst: FujianRu\_a5:e2:d3 (58:69:6...  
 > Internet Protocol Version 4, Src: 172.20.26.104, Dst: 128.119.245.12  
 > Transmission Control Protocol, Src Port: 7136 (7136), Dst Port: 80 (80), Seq: 151745, Ac...  
     Source Port: 7136  
     Destination Port: 80  
     [Stream index: 2]  
     [TCP Segment Len: 1259]  
     Sequence number: 151745 (relative sequence number)  
     [Next sequence number: 153004 (relative sequence number)]  
     Acknowledgment number: 1 (relative ack number)  
     Header Length: 20 bytes  
     Flags: 0x018 (PSH, ACK)  
         000. .... = Reserved: Not set  
         ...0 .... = Nonce: Not set

Seq=151745

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？

No.	Time	Source	Destination	Protoc	Lengt	Info
9...	7.2179...	172.20.26.104	128.119.245.12	HTTP	1313	POST /wireshark-labs/lab3-1-reply.h...
1...	8.5324...	128.119.245.12	172.20.26.104	HTTP	831	HTTP/1.1 200 OK (text/html)
10	0.1034...	172.20.5.141	239.255.255.2...	SSDP	143	M-SEARCH * HTTP/1.1
11	0.2050...	172.20.64.192	239.255.255.2...	SSDP	171	M-SEARCH * HTTP/1.1
20	0.2056...	172.20.4.28	239.255.255.2...	SSDP	216	M-SEARCH * HTTP/1.1
30	0.5125...	172.20.101.233	239.255.255.2...	SSDP	216	M-SEARCH * HTTP/1.1
1...	1.5373...	172.20.71.113	239.255.255.2...	SSDP	216	M-SEARCH * HTTP/1.1
1...	1.6386...	172.20.71.113	239.255.255.2...	SSDP	216	M-SEARCH * HTTP/1.1
1...	1.6391...	172.20.71.113	239.255.255.2...	SSDP	216	M-SEARCH * HTTP/1.1
1...	1.7416...	172.20.92.79	239.255.255.2...	SSDP	317	NOTIFY * HTTP/1.1
1...	1.8498...	172.20.8.235	239.255.255.2...	SSDP	179	M-SEARCH * HTTP/1.1

> Frame 938: 1313 bytes on wire (10504 bits), 1313 bytes captured (10504 bits) on interfac...  
 > Ethernet II, Src: HonHaiPr\_2e:4f:e9 (ac:d1:b8:2e:4f:e9), Dst: FujianRu\_a5:e2:d3 (58:69:6...  
 > Internet Protocol Version 4, Src: 172.20.26.104, Dst: 128.119.245.12  
 > Transmission Control Protocol, Src Port: 7136 (7136), Dst Port: 80 (80), Seq: 151745, Ac...  
 > [110 Reassembled TCP Segments (153003 bytes): #190(682), #191(1460), #192(1460), #193(14...  
     [Frame: 190, payload: 0-681 (682 bytes)]  
     [Frame: 191, payload: 682-2141 (1460 bytes)]  
     [Frame: 192, payload: 2142-3601 (1460 bytes)]  
     [Frame: 193, payload: 3602-5061 (1460 bytes)]  
     [Frame: 194, payload: 5062-6521 (1460 bytes)]  
     [Frame: 195, payload: 6522-7981 (1460 bytes)]  
     [Frame: 196, payload: 7982-9441 (1460 bytes)]  
     [Frame: 197, payload: 9442-10901 (1460 bytes)]  
     [Frame: 198, payload: 10902-12361 (1460 bytes)]  
     [Frame: 199, payload: 12362-13821 (1460 bytes)]  
     [Frame: 200, payload: 13822-15281 (1460 bytes)]  
     [Frame: 201, payload: 15282-16741 (1460 bytes)]  
     [Frame: 202, payload: 16742-18201 (1460 bytes)]  
     [Frame: 203, payload: 18202-19661 (1460 bytes)]  
     [Frame: 204, payload: 19662-21121 (1460 bytes)]  
     [Frame: 205, payload: 21122-22581 (1460 bytes)]  
     [Frame: 206, payload: 22582-24041 (1460 bytes)]  
     [Frame: 207, payload: 24042-25501 (1460 bytes)]  
     [Frame: 208, payload: 25502-26961 (1460 bytes)]  
     [Frame: 209, payload: 26962-28421 (1460 bytes)]  
     [Frame: 210, payload: 28422-29881 (1460 bytes)]  
     [Frame: 211, payload: 29882-31341 (1460 bytes)]  
     [Frame: 212, payload: 31342-32801 (1460 bytes)]  
     [Frame: 213, payload: 32802-34261 (1460 bytes)]  
     [Frame: 214, payload: 34262-35721 (1460 bytes)]  
     [Frame: 215, payload: 35722-37181 (1460 bytes)]  
     [Frame: 216, payload: 37182-38641 (1460 bytes)]  
     [Frame: 217, payload: 38642-40101 (1460 bytes)]  
     [Frame: 218, payload: 40102-41561 (1460 bytes)]  
     [Frame: 219, payload: 41562-43021 (1460 bytes)]  
     [Frame: 220, payload: 43022-44481 (1460 bytes)]  
     [Frame: 221, payload: 44482-45941 (1460 bytes)]  
     [Frame: 222, payload: 45942-47401 (1460 bytes)]  
     [Frame: 223, payload: 47402-48861 (1460 bytes)]  
     [Frame: 224, payload: 48862-50321 (1460 bytes)]  
     [Frame: 225, payload: 50322-51781 (1460 bytes)]  
     [Frame: 226, payload: 51782-53241 (1460 bytes)]  
     [Frame: 227, payload: 53242-54701 (1460 bytes)]  
     [Frame: 228, payload: 54702-56161 (1460 bytes)]  
     [Frame: 229, payload: 56162-57621 (1460 bytes)]  
     [Frame: 230, payload: 57622-59081 (1460 bytes)]  
     [Frame: 231, payload: 59082-60541 (1460 bytes)]  
     [Frame: 232, payload: 60542-62001 (1460 bytes)]  
     [Frame: 233, payload: 62002-63461 (1460 bytes)]  
     [Frame: 234, payload: 63462-64921 (1460 bytes)]  
     [Frame: 235, payload: 64922-66381 (1460 bytes)]  
     [Frame: 236, payload: 66382-67841 (1460 bytes)]  
     [Frame: 237, payload: 67842-69301 (1460 bytes)]  
     [Frame: 238, payload: 69302-70761 (1460 bytes)]  
     [Frame: 239, payload: 70762-72221 (1460 bytes)]  
     [Frame: 240, payload: 72222-73681 (1460 bytes)]  
     [Frame: 241, payload: 73682-75141 (1460 bytes)]  
     [Frame: 242, payload: 75142-76601 (1460 bytes)]  
     [Frame: 243, payload: 76602-78061 (1460 bytes)]  
     [Frame: 244, payload: 78062-79521 (1460 bytes)]  
     [Frame: 245, payload: 79522-80981 (1460 bytes)]  
     [Frame: 246, payload: 80982-82441 (1460 bytes)]  
     [Frame: 247, payload: 82442-83901 (1460 bytes)]  
     [Frame: 248, payload: 83902-85361 (1460 bytes)]  
     [Frame: 249, payload: 85362-86821 (1460 bytes)]  
     [Frame: 250, payload: 86822-88281 (1460 bytes)]  
     [Frame: 251, payload: 88282-89741 (1460 bytes)]  
     [Frame: 252, payload: 89742-91201 (1460 bytes)]  
     [Frame: 253, payload: 91202-92661 (1460 bytes)]  
     [Frame: 254, payload: 92662-94121 (1460 bytes)]  
     [Frame: 255, payload: 94122-95581 (1460 bytes)]  
     [Frame: 256, payload: 95582-97041 (1460 bytes)]  
     [Frame: 257, payload: 97042-98501 (1460 bytes)]  
     [Frame: 258, payload: 98502-99961 (1460 bytes)]  
     [Frame: 259, payload: 99962-101421 (1460 bytes)]  
     [Frame: 260, payload: 101422-102881 (1460 bytes)]  
     [Frame: 261, payload: 102882-104341 (1460 bytes)]  
     [Frame: 262, payload: 104342-105801 (1460 bytes)]  
     [Frame: 263, payload: 105802-107261 (1460 bytes)]  
     [Frame: 264, payload: 107262-108721 (1460 bytes)]  
     [Frame: 265, payload: 108722-110181 (1460 bytes)]  
     [Frame: 266, payload: 110182-111641 (1460 bytes)]  
     [Frame: 267, payload: 111642-113101 (1460 bytes)]  
     [Frame: 268, payload: 113102-114561 (1460 bytes)]  
     [Frame: 269, payload: 114562-116021 (1460 bytes)]  
     [Frame: 270, payload: 116022-117481 (1460 bytes)]  
     [Frame: 271, payload: 117482-118941 (1460 bytes)]  
     [Frame: 272, payload: 118942-120401 (1460 bytes)]  
     [Frame: 273, payload: 120402-121861 (1460 bytes)]  
     [Frame: 274, payload: 121862-123321 (1460 bytes)]  
     [Frame: 275, payload: 123322-124781 (1460 bytes)]  
     [Frame: 276, payload: 124782-126241 (1460 bytes)]  
     [Frame: 277, payload: 126242-127701 (1460 bytes)]  
     [Frame: 278, payload: 127702-129161 (1460 bytes)]  
     [Frame: 279, payload: 129162-130621 (1460 bytes)]  
     [Frame: 280, payload: 130622-132081 (1460 bytes)]  
     [Frame: 281, payload: 132082-133541 (1460 bytes)]  
     [Frame: 282, payload: 133542-135001 (1460 bytes)]  
     [Frame: 283, payload: 135002-136461 (1460 bytes)]  
     [Frame: 284, payload: 136462-137921 (1460 bytes)]  
     [Frame: 285, payload: 137922-139381 (1460 bytes)]  
     [Frame: 286, payload: 139382-140841 (1460 bytes)]  
     [Frame: 287, payload: 140842-142301 (1460 bytes)]  
     [Frame: 288, payload: 142302-143761 (1460 bytes)]  
     [Frame: 289, payload: 143762-145221 (1460 bytes)]  
     [Frame: 290, payload: 145222-146681 (1460 bytes)]  
     [Frame: 291, payload: 146682-148141 (1460 bytes)]  
     [Frame: 292, payload: 148142-149601 (1460 bytes)]  
     [Frame: 293, payload: 149602-151061 (1460 bytes)]  
     [Frame: 294, payload: 151062-152521 (1460 bytes)]  
     [Frame: 295, payload: 152522-153981 (1460 bytes)]  
     [Frame: 296, payload: 153982-155441 (1460 bytes)]  
     [Frame: 297, payload: 155442-156901 (1460 bytes)]  
     [Frame: 298, payload: 156902-158361 (1460 bytes)]  
     [Frame: 299, payload: 158362-159821 (1460 bytes)]  
     [Frame: 300, payload: 159822-161281 (1460 bytes)]  
     [Frame: 301, payload: 161282-162741 (1460 bytes)]  
     [Frame: 302, payload: 162742-164201 (1460 bytes)]  
     [Frame: 303, payload: 164202-165661 (1460 bytes)]  
     [Frame: 304, payload: 165662-167121 (1460 bytes)]  
     [Frame: 305, payload: 167122-168581 (1460 bytes)]  
     [Frame: 306, payload: 168582-170041 (1460 bytes)]  
     [Frame: 307, payload: 170042-171501 (1460 bytes)]  
     [Frame: 308, payload: 171502-172961 (1460 bytes)]  
     [Frame: 309, payload: 172962-174421 (1460 bytes)]  
     [Frame: 310, payload: 174422-175881 (1460 bytes)]  
     [Frame: 311, payload: 175882-177341 (1460 bytes)]  
     [Frame: 312, payload: 177342-178801 (1460 bytes)]  
     [Frame: 313, payload: 178802-180261 (1460 bytes)]  
     [Frame: 314, payload: 180262-181721 (1460 bytes)]  
     [Frame: 315, payload: 181722-183181 (1460 bytes)]  
     [Frame: 316, payload: 183182-184641 (1460 bytes)]  
     [Frame: 317, payload: 184642-186101 (1460 bytes)]  
     [Frame: 318, payload: 186102-187561 (1460 bytes)]  
     [Frame: 319, payload: 187562-189021 (1460 bytes)]  
     [Frame: 320, payload: 189022-190481 (1460 bytes)]  
     [Frame: 321, payload: 190482-191941 (1460 bytes)]  
     [Frame: 322, payload: 191942-193401 (1460 bytes)]  
     [Frame: 323, payload: 193402-194861 (1460 bytes)]  
     [Frame: 324, payload: 194862-196321 (1460 bytes)]  
     [Frame: 325, payload: 196322-197781 (1460 bytes)]  
     [Frame: 326, payload: 197782-199241 (1460 bytes)]  
     [Frame: 327, payload: 199242-200701 (1460 bytes)]  
     [Frame: 328, payload: 200702-202161 (1460 bytes)]  
     [Frame: 329, payload: 202162-203621 (1460 bytes)]  
     [Frame: 330, payload: 203622-205081 (1460 bytes)]  
     [Frame: 331, payload: 205082-206541 (1460 bytes)]  
     [Frame: 332, payload: 206542-208001 (1460 bytes)]  
     [Frame: 333, payload: 208002-209461 (1460 bytes)]  
     [Frame: 334, payload: 209462-210921 (1460 bytes)]  
     [Frame: 335, payload: 210922-212381 (1460 bytes)]  
     [Frame: 336, payload: 212382-213841 (1460 bytes)]  
     [Frame: 337, payload: 213842-215301 (1460 bytes)]  
     [Frame: 338, payload: 215302-216761 (1460 bytes)]  
     [Frame: 339, payload: 216762-218221 (1460 bytes)]  
     [Frame: 340, payload: 218222-219681 (1460 bytes)]  
     [Frame: 341, payload: 219682-221141 (1460 bytes)]  
     [Frame: 342, payload: 221142-222601 (1460 bytes)]  
     [Frame: 343, payload: 222602-224061 (1460 bytes)]  
     [Frame: 344, payload: 224062-225521 (1460 bytes)]  
     [Frame: 345, payload: 225522-226981 (1460 bytes)]  
     [Frame: 346, payload: 226982-228441 (1460 bytes)]  
     [Frame: 347, payload: 228442-229901 (1460 bytes)]  
     [Frame: 348, payload: 229902-231361 (1460 bytes)]  
     [Frame: 349, payload: 231362-232821 (1460 bytes)]  
     [Frame: 350, payload: 232822-234281 (1460 bytes)]  
     [Frame: 351, payload: 234282-235741 (1460 bytes)]  
     [Frame: 352, payload: 235742-237201 (1460 bytes)]  
     [Frame: 353, payload: 237202-238661 (1460 bytes)]  
     [Frame: 354, payload: 238662-240121 (1460 bytes)]  
     [Frame: 355, payload: 240122-241581 (1460 bytes)]  
     [Frame: 356, payload: 241582-243041 (1460 bytes)]  
     [Frame: 357, payload: 243042-244501 (1460 bytes)]  
     [Frame: 358, payload: 244502-245961 (1460 bytes)]  
     [Frame: 359, payload: 245962-247421 (1460 bytes)]  
     [Frame: 360, payload: 247422-248881 (1460 bytes)]  
     [Frame: 361, payload: 248882-250341 (1460 bytes)]  
     [Frame: 362, payload: 250342-251801 (1460 bytes)]  
     [Frame: 363, payload: 251802-253261 (1460 bytes)]  
     [Frame: 364, payload: 253262-254721 (1460 bytes)]  
     [Frame: 365, payload: 254722-256181 (1460 bytes)]  
     [Frame: 366, payload: 256182-257641 (1460 bytes)]  
     [Frame: 367, payload: 257642-259101 (1460 bytes)]  
     [Frame: 368, payload: 259102-260561 (1460 bytes)]  
     [Frame: 369, payload: 260562-262021 (1460 bytes)]  
     [Frame: 370, payload: 262022-263481 (1460 bytes)]  
     [Frame: 371, payload: 263482-264941 (1460 bytes)]  
     [Frame: 372, payload: 264942-266401 (1460 bytes)]  
     [Frame: 373, payload: 266402-267861 (1460 bytes)]  
     [Frame: 374, payload: 267862-269321 (1460 bytes)]  
     [Frame: 375, payload: 269322-270781 (1460 bytes)]  
     [Frame: 376, payload: 270782-272241 (1460 bytes)]  
     [Frame: 377, payload: 272242-273701 (1460 bytes)]  
     [Frame: 378, payload: 273702-275161 (1460 bytes)]  
     [Frame: 379, payload: 275162-276621 (1460 bytes)]  
     [Frame: 380, payload: 276622-278081 (1460 bytes)]  
     [Frame: 381, payload: 278082-279541 (1460 bytes)]  
     [Frame: 382, payload: 279542-281001 (1460 bytes)]  
     [Frame: 383, payload: 281002-282461 (1460 bytes)]  
     [Frame: 384, payload: 282462-283921 (1460 bytes)]  
     [Frame: 385, payload: 283922-285381 (1460 bytes)]  
     [Frame: 386, payload: 285382-286841 (1460 bytes)]  
     [Frame: 387, payload: 286842-288301 (1460 bytes)]  
     [Frame: 388, payload: 288302-289761 (1460 bytes)]  
     [Frame: 389, payload: 289762-291221 (1460 bytes)]  
     [Frame: 390, payload: 291222-292681 (1460 bytes)]  
     [Frame: 391, payload: 292682-294141 (1460 bytes)]  
     [Frame: 392, payload: 294142-295601 (1460 bytes)]  
     [Frame: 393, payload: 295602-297061 (1460 bytes)]  
     [Frame: 394, payload: 297062-298521 (1460 bytes)]  
     [Frame: 395, payload: 298522-300000 (1460 bytes)]



第六个报文段 Seq=6523, 在 http post 发送之前, tcp 连接建立之后发送。具体发送时间

Arrival Time: May 29, 2018 20:52:06.210985000 0й00000000

No.	Time	Source	Destination	Protocol	Length	Info
2...	2.227446	128.119.245.12	172.20.26.104	TCP	60	80 → 7136 [ACK] Seq=1 Ack=683 Win=30592...
2...	2.227513	172.20.26.104	128.119.245.12	TCP	1514	[TCP segment of a reassembled PDU]
2...	2.228669	128.119.245.12	172.20.26.104	TCP	60	80 → 7136 [ACK] Seq=1 Ack=2143 Win=3353...
2...	2.228670	128.119.245.12	172.20.26.104	TCP	60	80 → 7136 [ACK] Seq=1 Ack=3603 Win=3648...
2...	2.228670	128.119.245.12	172.20.26.104	TCP	60	80 → 7136 [ACK] Seq=1 Ack=6523 Win=4236...
2...	2.228723	172.20.26.104	128.119.245.12	TCP	1514	[TCP segment of a reassembled PDU]
2...	2.228739	172.20.26.104	128.119.245.12	TCP	1514	[TCP segment of a reassembled PDU]
2...	2.228755	172.20.26.104	128.119.245.12	TCP	1514	[TCP segment of a reassembled PDU]
2...	2.228768	172.20.26.104	128.119.245.12	TCP	1514	[TCP segment of a reassembled PDU]
2...	2.228781	172.20.26.104	128.119.245.12	TCP	1514	[TCP segment of a reassembled PDU]
2...	2.228794	172.20.26.104	128.119.245.12	TCP	1514	[TCP segment of a reassembled PDU]

> Internet Protocol Version 4, Src: 128.119.245.12, Dst: 172.20.26.104

✓ Transmission Control Protocol, Src Port: 80 (80), Dst Port: 7136 (7136), Seq: 1, Ack: 6523, Len: 0

Source Port: 80

Destination Port: 7136

[Stream index: 2]

[TCP Segment Len: 0]

Sequence number: 1 (relative sequence number)

Acknowledgment number: 6523 (relative ack number)

Header Length: 20 bytes

> Flags: 0x010 (ACK)

Window size value: 331

[Calculated window size: 42368]

[Window size scaling factor: 128]

> Checksum: 0xf96b [validation disabled]

Urgent pointer: 0

> [SEQ/ACK analysis]

对应的 ack 即为服务器返回的第六个 ack, 返回时间为 :

Arrival Time: May 29, 2018 20:52:06.455073000 0й00000000

## 8. 前六个 TCP 报文段的长度各是多少 ?

如图 :

9...	7.217913	172.20.26.104	128.119.245.12	HTTP	1313 POST /wireshark-labs/lab3-1-reply.htm H...
1...	8.532458	128.119.245.12	172.20.26.104	HTTP	831 HTTP/1.1 200 OK (text/html)
7...	5.561231	111.30.131.187	172.20.26.104	SSL	135 Continuation Data
7...	5.829477	111.30.131.187	172.20.26.104	SSL	135 Continuation Data
1...	1.736825	172.20.26.104	128.119.245.12	TCP	54 7133 → 80 [FIN, ACK] Seq=1 Ack=1 Win=64...
1...	1.736985	172.20.26.104	202.108.23.152	TCP	54 7130 → 80 [FIN, ACK] Seq=1 Ack=1 Win=63...
1...	1.738146	172.20.26.104	128.119.245.12	TCP	66 7136 → 80 [SYN] Seq=0 Win=8192 Len=0 MS...
1...	1.816299	172.20.26.104	45.32.56.226	TCP	66 7143 → 8000 [SYN] Seq=0 Win=8192 Len=0 ...
1...	1.816299	172.20.26.104	45.32.56.226	TCP	66 7144 → 8000 [SYN] Seq=0 Win=8192 Len=0 ...
1...	1.958243	45.32.56.226	172.20.26.104	TCP	66 8000 → 7143 [SYN, ACK] Seq=0 Ack=1 Win=...
1...	1.958245	45.32.56.226	172.20.26.104	TCP	66 8000 → 7144 [SYN, ACK] Seq=0 Ack=1 Win=...

> Frame 938: 1313 bytes on wire (10504 bits), 1313 bytes captured (10504 bits) on interface 0  
 > Ethernet II, Src: HonHaiPr\_2e:4f:e9 (ac:d1:b8:2e:4f:e9), Dst: FujianRu\_a5:e2:d3 (58:69:6c:a5:e2:d3)  
 > Internet Protocol Version 4, Src: 172.20.26.104, Dst: 128.119.245.12  
 > Transmission Control Protocol, Src Port: 7136 (7136), Dst Port: 80 (80), Seq: 151745, Ack: 1, Len: 1...  
 √ [110 Reassembled TCP Segments (153003 bytes): #190(682), #191(1460), #192(1460), #193(1460), #194(14...  
     [Frame: 190, payload: 0-681 (682 bytes)]  
     [Frame: 191, payload: 682-2141 (1460 bytes)]  
     [Frame: 192, payload: 2142-3601 (1460 bytes)]  
     [Frame: 193, payload: 3602-5061 (1460 bytes)]  
     [Frame: 194, payload: 5062-6521 (1460 bytes)]  
     [Frame: 195, payload: 6522-7981 (1460 bytes)]  
     [Frame: 196, payload: 7982-9441 (1460 bytes)]  
     [Frame: 197, payload: 9442-10901 (1460 bytes)]  
     [Frame: 198, payload: 10902-12361 (1460 bytes)]  
     [Frame: 199, payload: 12362-13821 (1460 bytes)]  
     [Frame: 200, payload: 13822-15281 (1460 bytes)]

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？

No.	Time	Source	Destination	Protocol	Length	Info
1...	1.958714	172.20.26.104	45.32.56.226	TCP	54	7144 → 8000 [ACK] Seq=1 Ack=1 Win=16384 Len=0
1...	1.958970	172.20.26.104	45.32.56.226	TCP	345	7143 → 8000 [PSH, ACK] Seq=1 Ack=1 Win=16384 Len=...
1...	1.959004	172.20.26.104	45.32.56.226	TCP	345	7144 → 8000 [PSH, ACK] Seq=1 Ack=1 Win=16384 Len=...
1...	1.982392	128.119.245.12	172.20.26.104	TCP	66	80 → 7136 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0
1...	1.982622	172.20.26.104	128.119.245.12	TCP	54	7136 → 80 [ACK] Seq=1 Ack=1 Win=16384 Len=0
1...	1.983814	172.20.26.104	128.119.245.12	TCP	736	[TCP segment of a reassembled PDU]
1...	1.984452	172.20.26.104	128.119.245.12	TCP	1514	[TCP segment of a reassembled PDU]
1...	1.984508	172.20.26.104	128.119.245.12	TCP	1514	[TCP segment of a reassembled PDU]
1...	1.984545	172.20.26.104	128.119.245.12	TCP	1514	[TCP segment of a reassembled PDU]
1...	1.984582	172.20.26.104	128.119.245.12	TCP	1514	[TCP segment of a reassembled PDU]
1...	1.984631	172.20.26.104	128.119.245.12	TCP	1514	[TCP segment of a reassembled PDU]

> Frame 188: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0  
 > Ethernet II, Src: FujianRu\_a5:e2:d3 (58:69:6c:a5:e2:d3), Dst: HonHaiPr\_2e:4f:e9 (ac:d1:b8:2e:4f:e9)  
 > Internet Protocol Version 4, Src: 128.119.245.12, Dst: 172.20.26.104  
 > Transmission Control Protocol, Src Port: 80 (80), Dst Port: 7136 (7136), Seq: 0, Ack: 1, Len: 0

如图，接收端公示的最小的可用缓存空间是 29200，该窗口大小会一直增加，不会出现接收端的缓存不够用的情况。

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

没有出现重传，没有重复的序号报文。

总报文大小：

TCP length: 152884]

开始时间：

Arrival Time: May 29, 2018 20:52:06.210985000 0й0000й00

结束时间:

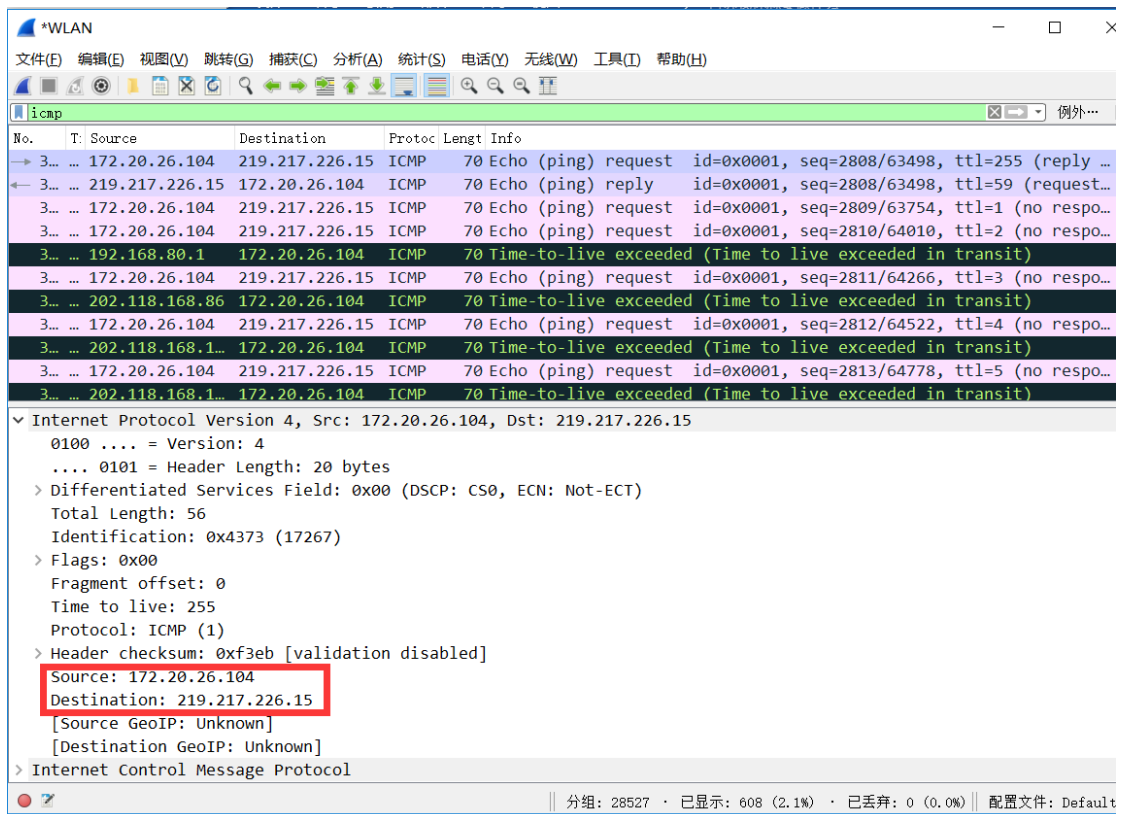
Arrival Time: May 29, 2018 20:52:06.455073000 0й0000й00

throughput = 152884byte / 0.24s = 637016.66KB/S

### 3. IP 分析

#### A. 对捕获的数据包进行分析

1. 在你的捕获窗口中，应该能看到由你的主机发出的一系列 ICMP Echo Request 包和中间路由器返回的一系列 ICMP TTL-exceeded 消息。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？

主机 IP 地址:172.20.26.104

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

```

3... 202.118.168.1... 172.20.26.104 ICMP 70 Time-to-live exceeded (Time to live exceeded in transit)
  Identification: 0x4373 (17267)
  > Flags: 0x00
  Fragment offset: 0
  Time to live: 255
  Protocol: ICMP (1)
  > Header checksum: 0xf3eb [validation disabled]
  Source: 172.20.26.104
  Destination: 219.217.226.15
  [Source GeoIP: Unknown]
  [Destination GeoIP: Unknown]
  > Internet Control Message Protocol

0000  58 69 6c a5 e2 d3 ac d1 b8 2e 4f e9 08 00 45 00  Xil.... ..0...E.
0010  00 38 43 73 00 00 ff 01 f3 eb ac 14 1a 68 db d9  .8Cs.... ..h..
0020  e2 0f 08 00 2b 45 00 01 0a f8 20 20 20 20 20 20  ....+E.. ..
0030  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0040  20 20 20 20 20 20

```

上层协议字段的值是 01。

### 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？

```

> Frame 3497: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
> Ethernet II, Src: HonHaiPr_2e:4f:e9 (ac:d1:b8:2e:4f:e9), Dst: FujianRu_a5:e2:d3 (58:69:6c:a5:e2:d3)
  > Internet Protocol Version 4, Src: 172.20.26.104, Dst: 219.217.226.15
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 56
  Identification: 0x4373 (17267)
  > Flags: 0x00
  Fragment offset: 0
  Time to live: 255

0000  58 69 6c a5 e2 d3 ac d1 b8 2e 4f e9 08 00 45 00  Xil.... ..0...E.
0010  00 38 43 73 00 00 ff 01 f3 eb ac 14 1a 68 db d9  .8Cs.... ..h..
0020  e2 0f 08 00 2b 45 00 01 0a f8 20 20 20 20 20 20  ....+E.. ..
0030  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0040  20 20 20 20 20 20

```

20 字节。净载 36 字节。可查看数据报长度字段，为 56，减去首部字段即为净载。

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

```

> Frame 3497: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
> Ethernet II, Src: HonHaiPr_2e:4f:e9 (ac:d1:b8:2e:4f:e9), Dst: FujianRu_a5:e2:d3 (58:69:6c:a5:e2:d3)
  > Internet Protocol Version 4, Src: 172.20.26.104, Dst: 219.217.226.15
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 56
  Identification: 0x4373 (17267)
  > Flags: 0x00
  Fragment offset: 0
  Time to live: 255

0000  58 69 6c a5 e2 d3 ac d1 b8 2e 4f e9 08 00 45 00  Xil.... ..0...E.
0010  00 38 43 73 00 00 ff 01 f3 eb ac 14 1a 68 db d9  .8Cs.... ..h..
0020  e2 0f 08 00 2b 45 00 01 0a f8 20 20 20 20 20 20  ....+E.. ..
0030  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0040  20 20 20 20 20 20

```

Fragment offset (13 bits) (ip\_frag\_offset), 2 字节 | 分组: 28527 · 已显示: 608 (2.1%) · 已丢弃: 0 (0.0%) | 配置文件: Default

没有。查看标志位。 Fragment offset: 0

- 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 ICMP Echo Request 消息，在 packet details 窗口展开数据包的 Internet Protocol 部分。在“listing of captured packets”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）

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？

答：identification,ttl,header checksum,

2. 哪些字段必须保持常量？哪些字段必须改变？为什么？

答：保持常量：Version,header length, upper layer protocol 源 IP、目的 IP 等

必须改变：identification,ttl,header checksum

ID 必须改变：鉴别码，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，前面的字段改变，该值也跟着改变；

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

```
> Frame 3498: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
> Ethernet II, Src: FujianRu_a5:e2:d3 (58:69:6c:a5:e2:d3), Dst: HonHaiPr_2e:4f:e9 (ac:d1:b8:2e:4f:e9)
> Internet Protocol Version 4, Src: 219.217.226.15, Dst: 172.20.26.104
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes
> Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 56
  Identification: 0xd9ac (55724)
> Flags: 0x00
  Fragment offset: 0
  Time to live: 59

0000  ac d1 b8 2e 4f e9 58 69  6c a5 e2 d3 08 00 45 00  ....O.Xi l....E.
0010  00 38 d9 ac 00 00 3b 01  21 b3 db d9 e2 0f ac 14  .8....;. !.....
0020  1a 68 00 00 33 45 00 01  0a f8 20 20 20 20 20 20  .h..3E.. ..
0030  20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20
0040  20 20 20 20 20 20
```

答：16 位。在某范围内递增。

4. 找到由最近的路由器（第一跳）返回给你主机的 ICMPTime-to-live exceeded 消息。

思考下列问题：

Identification 字段和 TTL 字段的值是什么？最近的路由器（第一跳）返回给你主机的 ICMP Time-to-live exceeded 消息中这些值是否保持不变？为什么？



Identification : 0x7769, TTL : 254,

Identification 会变, TTL 保持不变。因为 Identification 标识一个报文, 不同报文

Identification 不同。因为是第一跳路由器发回的数据报, 故 TTL 是最大值减 1,

总是等于 254。

5. 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。

思考下列问题 :

该消息是否被分解成不止一个 IP 数据报 ?

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

```
Internet Protocol Version 4, Src: 172.20.26.104, Dst: 219.217.226.15
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 520
    Identification: 0x4e68 (20072)
  > Flags: 0x00
    Fragment offset: 1480
    Time to live: 5
    Protocol: ICMP (1)
  > Header checksum: 0xe06e [validation disabled]
    Source: 172.20.26.104
    Destination: 219.217.226.15
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
  > [2 IPv4 Fragments (1980 bytes): #6703(1480), #6704(500)]
    [Frame: 6703, payload: 0-1479 (1480 bytes)]
    [Frame: 6704, payload: 1480-1979 (500 bytes)]
    [Fragment count: 2]
    [Reassembled IPv4 length: 1980]
    [Reassembled IPv4 data: 080026560000115ed20202020202020202020202020202020...]
```

分成了两个。

```
.... ..00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
Total Length: 1500
Identification: 0x4e67 (20071)
  > Flags: 0x01 (More Fragments)
    0... .... = Reserved bit: Not set
    .0.. .... = Don't fragment: Not set
    ..1. .... = More fragments: Set
  Fragment offset: 0
  > Time to live: 1
  > [Expert Info (Note/Sequence): "Time To Live" only 1]
    ["Time To Live" only 1]
```

More fragments=1 表示分片了且不是最后一块, 该分片的长度是 1500B

### C. 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。

思考下列问题：

原始数据包被分成了多少片？

这些分片中 IP 数据报头部哪些字段发生了变化？

1...	172.20.0.1	172.20.26.104	ICMP	590	Time-to-live exceeded (Time to live exceeded in transit)
1...	172.20.26.104	219.217.226.15	ICMP	554	Echo (ping) request id=0x0001, seq=6569/43289, ttl=2 (no respo...
1...	192.168.80.1	172.20.26.104	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
1...	172.20.26.104	219.217.226.15	ICMP	554	Echo (ping) request id=0x0001, seq=6570/43545, ttl=3 (no respo...
1...	202.118.168.86	172.20.26.104	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
1...	172.20.26.104	219.217.226.15	ICMP	554	Echo (ping) request id=0x0001, seq=6571/43801, ttl=4 (no respo...
1...	202.118.168.1...	172.20.26.104	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
1...	172.20.26.104	219.217.226.15	ICMP	554	Echo (ping) request id=0x0001, seq=6572/44057, ttl=5 (no respo...
[Source GeoIP: Unknown]					
[Destination GeoIP: Unknown]					
v [3 IPv4 Fragments (3480 bytes): #102(1480), #103(1480), #104(520)]					
[Frame: 102, payload: 0-1479 (1480 bytes)]					
[Frame: 103, payload: 1480-2959 (1480 bytes)]					
[Frame: 104, payload: 2960-3479 (520 bytes)]					
[Fragment count: 3]					
[Reassembled IPv4 length: 3480]					
[Reassembled IPv4 data: 0800047d000119a8202020202020202020202020202020...]					
v Internet Control Message Protocol					
Type: 8 (Echo (ping) request)					

分成三片

前 2 个分片More fragments=1, 后两个分片offset 变为 1480 和 2960

## 4.抓取 ARP 数据包

1. 利用 MS-DOS 命令 `arp` 或 `c:\windows\system32\arp` 查看主机上 ARP 缓存的内容。

说明 ARP 缓存中每一列的含义是什么？

输入 `arp - a` 查看主机上 ARP 缓存的内容，结果如下图所示（截图显示部分）：

管理员: 命令提示符

C:\Windows\system32&gt;arp -a

```

接口: 192.168.244.1 --- 0x2
Internet 地址      物理地址      类型
192.168.244.254    00-50-56-ec-c8-d3    动态
192.168.244.255    ff-ff-ff-ff-ff-ff    静态
224.0.0.2          01-00-5e-00-00-02    静态
224.0.0.5          01-00-5e-00-00-05    静态
224.0.0.22         01-00-5e-00-00-16    静态
224.0.0.100        01-00-5e-00-00-64    静态
224.0.0.251        01-00-5e-00-00-fb    静态
224.0.0.252        01-00-5e-00-00-fc    静态
224.0.0.253        01-00-5e-00-00-fd    静态
224.67.40.111      01-00-5e-43-28-6f    静态
224.117.200.182     01-00-5e-75-c8-b6    静态
224.168.100.17      01-00-5e-28-64-11    静态
224.193.235.115     01-00-5e-41-eb-73    静态
225.4.76.124        01-00-5e-04-4c-7c    静态
225.255.141.183     01-00-5e-7f-8d-b7    静态
227.70.211.58       01-00-5e-46-d3-3a    静态
229.255.255.250     01-00-5e-7f-ff-fa    静态
230.4.109.47        01-00-5e-04-6d-2f    静态

```

ARP 缓存中的每一列分别表示 IP 地址所对应的物理地址和类型（动态配置或静态配置）

2. 清除主机上 ARP 缓存的内容，抓取 ping 命令时的数据包。分析数据包，回答下面的问题：

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？

数据包格式：

以太网的 ARP 请求和应答的分组格式，如图 6-11 所示。



图 6-11 ARP 请求和应答的分组格式

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）。

截取一个 ARP 数据包：

```
▼ Address Resolution Protocol (request)
  Hardware type: Ethernet (1)
  Protocol type: IPv4 (0x0800)
  Hardware size: 6
  Protocol size: 4
  Opcode: request (1)
  Sender MAC address: 3c:b6:b7:c6:5f:f1 (3c:b6:b7:c6:5f:f1)
  Sender IP address: 172.20.0.148
  Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
  Target IP address: 172.20.108.38
```

## ② 如何判断一个 ARP 数据是请求包还是应答包？

从 op 字段判断：1 是请求，2 是应答

## 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？

因为 ARP 进行查询工作时，并不知道目的 MAC 地址，所以要通过广播的形式，向局域网中所有节点进行查询；但 ARP 响应时已经知道源 MAC 地址（就是发送广播帧的节点），所以要在一个有着明确目的局域网地址的帧中传送。

# 5. 抓取 UDP 数据包

## 1. 消息是基于 UDP 的还是 TCP 的？

UDP。

## 2. 你的主机 IP 地址是什么？目的主机 IP 地址是什么？

答：本机 IP:172.20.141.150,目的 IP:123.151.13.167

```
▶ Internet Protocol Version 4, Src: 172.20.141.150, Dst: 123.151.13.167
▲ User Datagram Protocol, Src Port: 4015, Dst Port: 8000
```

## 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？

答：4015, 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？

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。

抓取的一个 UDP 数据报如下所示：

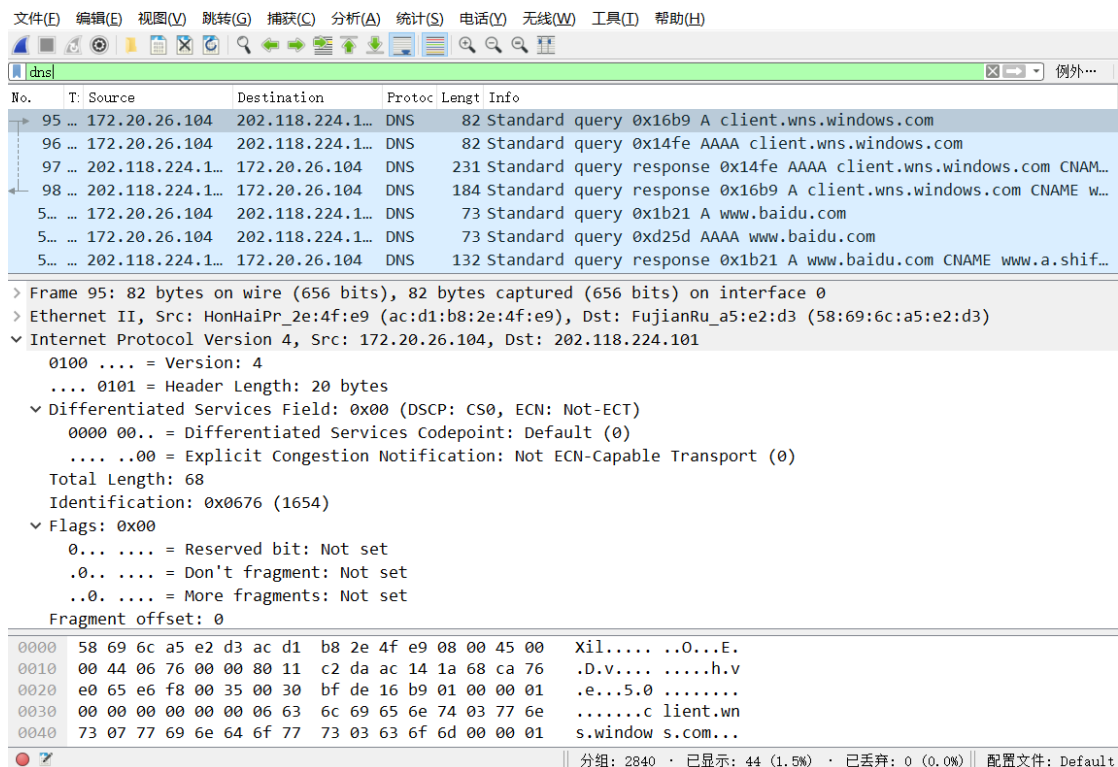
```
Source Port: 4015
Destination Port: 8000
Length: 159
Checksum: 0xa3e5 [unverified]
```

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?

返回的 ICQ 数据包时作确认用的, 因为 UDP 是不可靠的数据传输协议, 所以需要这种确认机制。能看出来, UDP 协议没有握手过程, UDP 数据包没有序列号, 因此不能像 TCP 协议那样先握手再发送数据, 而是直接发送数据。

## 6. 利用 Wireshark 进行 DNS 协议分析

1. 打开浏览器, 输入 [www.baidu.com](http://www.baidu.com), DNS 查询消息如下图:



2. 我的电脑 IP 地址: 172.20.26.104, 本地域名服务器 IP 地址: 202.118.224.101.如图:

Source: 172.20.26.104

Destination: 202.118.224.101

3. UDP 报文的源端口号 59128, 目的端口号 53

User Datagram Protocol, Src Port: 59128 (59128), Dst Port: 53 (53)  
 Source Port: 59128  
 Destination Port: 53  
 Length: 48  
 Checksum: 0xbfde [validation disabled]  
 [Good Checksum: False]

4. DNS 报文格式





DNS协议报文格式

**会话标识 (2 字节):** 是 DNS 报文的 ID 标识, 对于请求报文和其对应的应答报文, 这个字段是相同的, 通过它可以区分 DNS 应答报文是哪个请求的响应

**标志 (2 字节):**

- QR (1bit)      查询/响应标志, 0 为查询, 1 为响应
- opcode (4bit) 0 表示标准查询, 1 表示反向查询, 2 表示服务器状态请求
- AA (1bit)      表示授权回答
- TC (1bit)      表示可截断的
- RD (1bit)      表示期望递归
- RA (1bit)      表示可用递归
- rcode (4bit)   表示返回码, 0 表示没有差错, 3 表示名字差错, 2 表示服务器错误 (Server Failure)

**数量字段 (总共 8 字节):**

Questions、Answer RRs、Authority RRs、Additional RRs 各自表示后面的四个区域的数量。Questions 表示查询问题区域节的数量, Answers 表示回答区域的数量, Authoritative nameservers 表示授权区域的数量, Additional records 表示附加区域的数量

```
> User Datagram Protocol, Src Port: 53 (53), Dst Port: 53912 (53912)
v Domain Name System (response)
  [Request In: 753]
  [Time: 0.002935000 seconds]
  Transaction ID: 0xadd9
  v Flags: 0x8180 Standard query response, No error
    1... .. = Response: Message is a response
    .000 0... .. = Opcode: Standard query (0)
    .... 0... .. = Authoritative: Server is not an authority for domain
    .... ..0. .... = Truncated: Message is not truncated
    .... ..1 .... = Recursion desired: Do query recursively
    .... ..1... .. = Recursion available: Server can do recursive queries
    .... ..0... .. = Z: reserved (0)
    .... ..0. .... = Answer authenticated: Answer/authority portion was not authenticated by the server
    .... ..0 .... = Non-authenticated data: Unacceptable
    .... ..0000 = Reply code: No error (0)
  Questions: 1
  Answer RRs: 2
  Authority RRs: 0
  Additional RRs: 0
  v Queries
    v t12.baidu.com: type A, class IN
      Name: t12.baidu.com
```

## 5. DNS 响应报文

```
Domain Name System (response)
  [Request In: 756]
  [Time: 0.025279000 seconds]
  Transaction ID: 0x3629
  Flags: 0x8180 Standard query response, No error
    1... .. = Response: Message is a response
    .000 0... .. = Opcode: Standard query (0)
    .... 0... .. = Authoritative: Server is not an authority for domain
    .... ..0... .. = Truncated: Message is not truncated
    .... ..1... .. = Recursion desired: Do query recursively
    .... ..1... .. = Recursion available: Server can do recursive queries
    .... ..0... .. = Z: reserved (0)
    .... ..0... .. = Answer authenticated: Answer/authority portion was not authenticated by the server
    .... ..0... .. = Non-authenticated data: Unacceptable
    .... ..0000 = Reply code: No error (0)
  Questions: 1
  Answer RRs: 1
  Authority RRs: 1
  Additional RRs: 0
  Queries
    t2.baidu.com: type AAAA, class IN
      Name: t2.baidu.com
      [Name Length: 12]
      [Label Count: 3]
      Type: AAAA (IPv6 Address) (28)
      Class: IN (0x0001)
  Answers
    t2.baidu.com: type CNAME, class IN, cname simage.jomodns.com
      Name: t2.baidu.com
```

## 四、实验心得

经过此次实验，从应用层的 HTTP、DNS 到传输层的 TCP、UDP，再到网络层的 IP、ICMP 在到链路层的 ARP，利用 Wireshark，对每一协议进行了深入的跟踪与分析，基本了解了每层的功能和不同协议的报文格式，实际的感受了每个报文设计的巧妙，每个字段的必要，以及从事计算机网络工作者所做的工作的重要性以及他们的智慧。

实验过程中由于第一次使用 Wireshark，很多功能都不太了解，抓到的数据包太多无法准确确定。除了通过包的类型的筛选以外，关闭其他的一些后台程序减少网络数据包的交互，能够更好容易的定位包。此时网络数据包相对干净，分析每个包，总能有特别发现，也是挺有趣儿的。