



哈尔滨工业大学  
Harbin Institute of Technology

# 计算机网络 课程实验报告

实验名称	利用 Wireshark 进行协议分析					
姓名	冯开来		院系	计算学部		
班级	1903602		学号	1190201215		
任课教师	李全龙		指导教师	李全龙		
实验地点	格物 207		实验时间	2021.11.24		
实验课表现	出勤、表现得分(10)		实验报告 得分(40)		实验总分	
	操作结果得分(50)					
教师评语						

## 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。

## 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

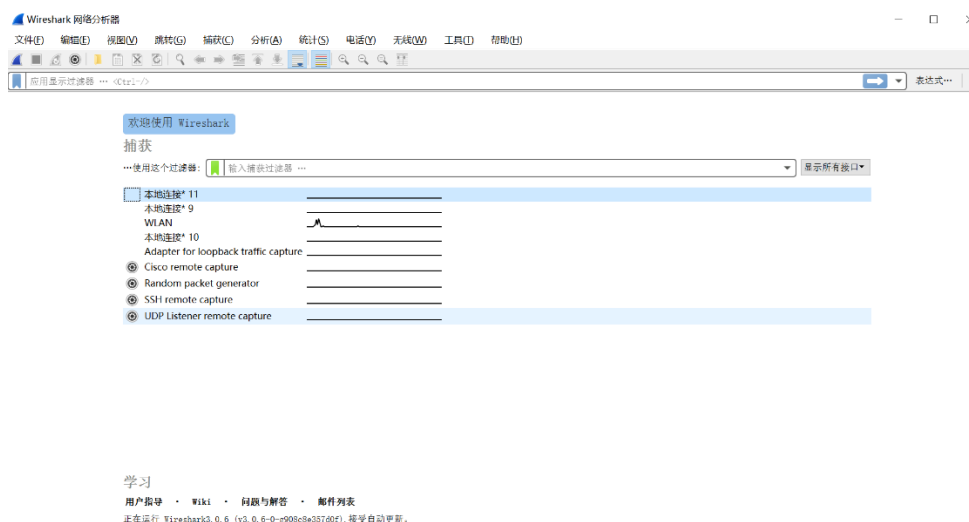
## 选做内容：

- 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

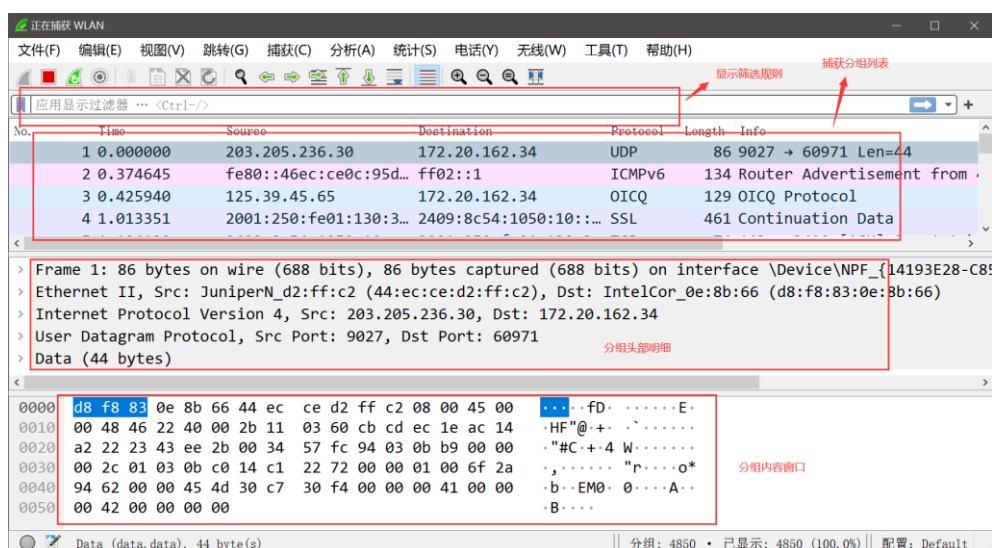
## 实验过程：

### (一) Wireshark的使用

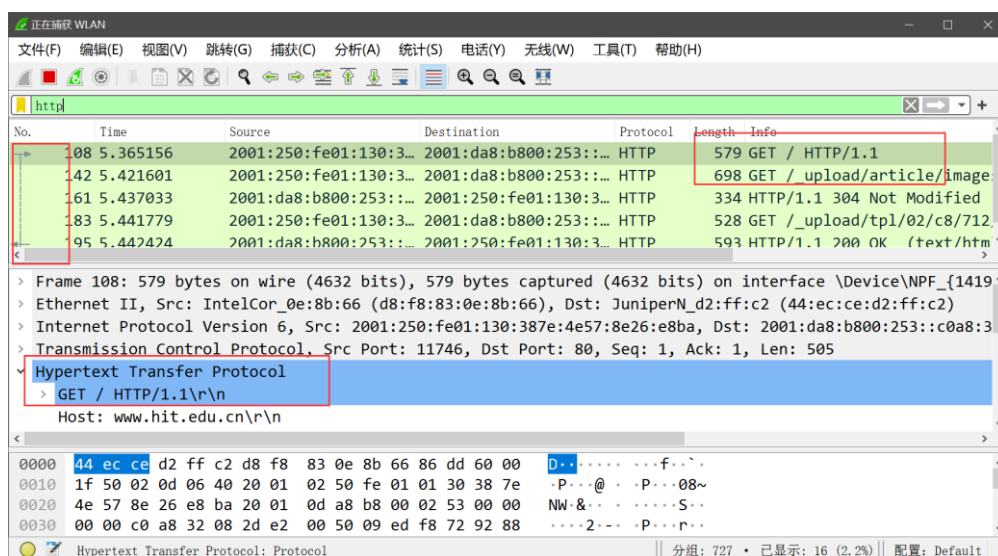
启动Web浏览器，启动Wireshark



开始分组捕获，出现分组捕获窗口：



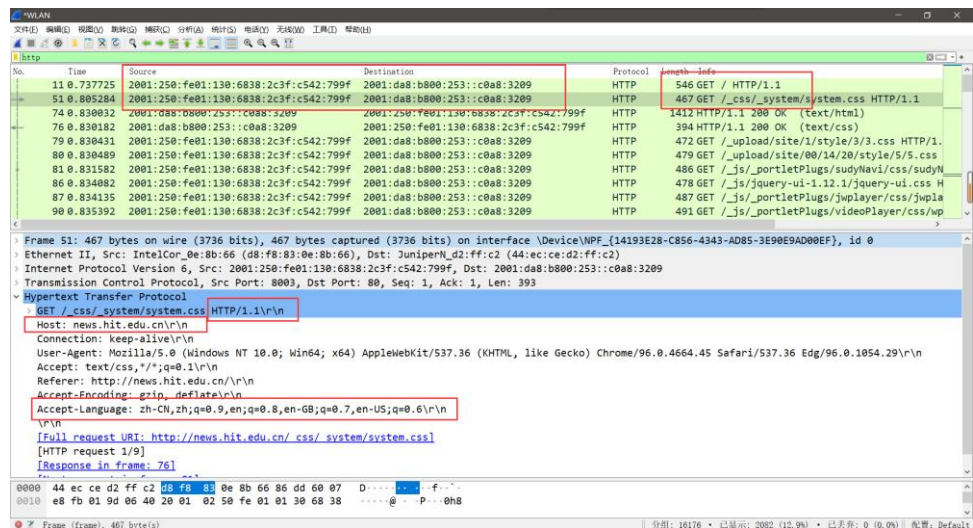
输入URL (<http://www.hit.edu.cn>), 捕获HTTP GET报文:



## (二) HTTP分析

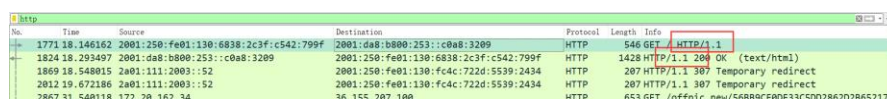
### 1) HTTP GET/response交互

- 启动Web browser, 然后启动Wireshark分组嗅探器。在窗口的显示过滤说明出输入http, 分组列表窗口中将只显示所俘获到的HTTP报文。
- 开始Wireshark分组俘获。
- 在打开的Web browser窗口中输入一下地址: <http://news.hit.edu.cn>
- 停止分组俘获。



根据俘获窗口内容, 思考以下问题:

- 你的浏览器运行的HTTP1.0还是HTTP1.1? 访问的服务器HTTP协议版本号?



第一条是HTTP GET, 浏览器运行HTTP1.1

第二条是服务器端返回, 服务器运行HTTP1.1

b) 你的浏览器向服务器指出接受何种语言版本

```

Transmission Control Protocol, Src Port: 11014, Dst Port: 80, Seq: 1, ACK: 1, Len: 472
Hypertext Transfer Protocol
GET / HTTP/1.1\r\n
Host: news.hit.edu.cn\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 Ge
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;
Accept-Encoding: gzip, deflate\r\n
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6\r\n
\r\n

```

查看GET报文，Accept-Language包含简体繁体中文，英语（美）

c) 你的计算机IP地址？服务器<http://news.hit.edu.cn/news>的IP地址？

No.	Time	Source	Destination	Protocol	Length	Info
1771	18.146162	2001:250:fe01:130:6838:2c3f:c542:799f	2001:da8:b800:253::c0a8:3209	HTTP	546	GET / HTTP/1.1
1824	18.293497	2001:da8:b800:253::c0a8:3209	2001:250:fe01:130:6838:2c3f:c542:799f	HTTP	1428	HTTP/1.1 200 OK
1869	18.548015	2001:111:2003::52	2001:250:fe01:130:fc4c:722d:5539:2434	HTTP	207	HTTP/1.1 307 Tem
2012	19.672186	2001:111:2003::52	2001:250:fe01:130:fc4c:722d:5539:2434	HTTP	207	HTTP/1.1 307 Tem

```

Frame 1771: 546 bytes on wire (4368 bits), 546 bytes captured (4368 bits) on interface \Device\NPF_{14193E28-C856-4343-AD85-3E96}
Ethernet II, Src: IntelCor_0e:8b:66 (d8:f8:83:0e:8b:66), Dst: JuniperN_d2:ff:c2 (44:ec:ce:d2:ff:c2)
Destination: JuniperN_d2:ff:c2 (44:ec:ce:d2:ff:c2)
Source: IntelCor_0e:8b:66 (d8:f8:83:0e:8b:66)
Type: IPv6 (0x86dd)
Internet Protocol Version 6, Src: 2001:250:fe01:130:6838:2c3f:c542:799f, Dst: 2001:da8:b800:253::c0a8:3209
0110 .... = Version: 6
.... 0000 0000 .... = Traffic Class: 0x00 (DSCP: CS0, ECN: Not-ECT)
.... 1111 0010 1010 1101 0010 = Flow Label: 0xf2ad2
Payload Length: 492
Next Header: TCP (6)
Hop Limit: 64
Source Address: 2001:250:fe01:130:6838:2c3f:c542:799f
Destination Address: 2001:da8:b800:253::c0a8:3209

```

根据GET请求，我计算机IP地址即为Source，服务器为Destination

计算机IP: 2001:250:fe01:130:6838:2c3f:c542:799f

服务器IP: 2001:da8:b800:253::c0a8:3209

d) 服务器返回你的浏览器的状态码？

No.	Time	Source	Destination	Protocol	Length	Info
207	18.2...	2001:...	2001:da8...	HTTP	626	GET / HTTP/1.1
219	18.2...	2001:...	2001:250...	HTTP	1472	HTTP/1.1 200 OK (text/html)
223	18.3...	2001:...	2001:da8...	HTTP	576	GET /_visitcount?siteId=49&type=1&colu
231	18.3...	2001:...	2001:250...	HTTP	353	HTTP/1.1 200 200

```

[6 Reassembled TCP Segments (8598 bytes): #212(1440), #213(1440), #214(1440), #215(1440)]
Hypertext Transfer Protocol
HTTP/1.1 200 OK\r\n
Date: Wed, 24 Nov 2021 13:58:30 GMT\r\n

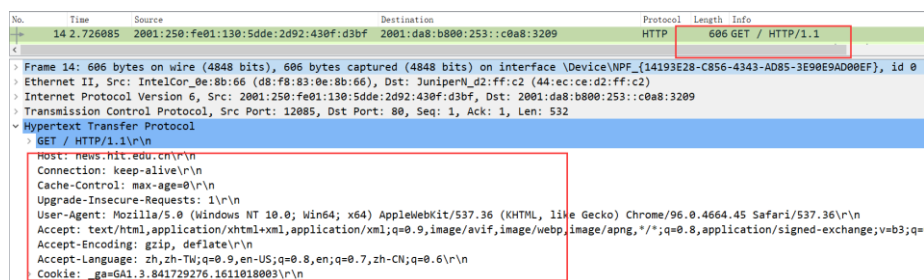
```

状态码为200。

2) HTTP 条件GET/response交互

- 启动浏览器，清空浏览器的缓存（在浏览器中，选择“工具”菜单中的“Internet 选项”命令，在出现的对话框中，选择“删除文件”）。
- 启动Wireshark 分组俘获器。开始Wireshark 分组俘获。
- 在浏览器的地址栏中输入以下URL: <http://hitgs.hit.edu.cn/news>,在你的浏览器中重新输入相同的URL 或单击浏览器中的“刷新”按钮。
- 停止Wireshark 分组俘获，在显示过滤筛选说明处输入“http”,分组列表子窗口中将只显示所俘获到的HTTP 报文。

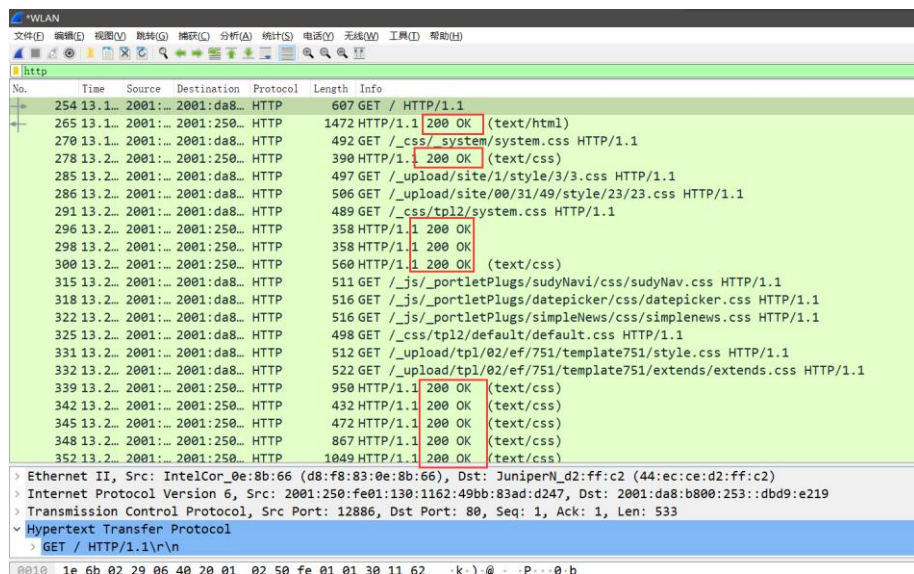
a) 分析你的浏览器向服务器发出的第一个HTTPGET 请求的内容, 在该请求报文中, 是否有一行是: IF-MODIFIED-SINCE?



b) 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？

返回报文的状态码是304 Not Modified 的话，说明缓存的版本是最新的，相应的消息中不包含文件内容；

若服务器返回的报文的状态码是 200 OK 的话，说明缓存的版本不是最新，或者缓存不命中，相应的消息来自服务器端，消息中包含文件内容。若返回的所有报文消息状态码都是 200 OK，所以说明都明确返回了文件的内容。





- c) 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，在该首部行后面跟着的信息是什么？

有IF-MODIFIED-SINCE（本人刷新很多次好不容易刷新出304，太不容易）

在 IF-MODIFIED-SINCE 后面跟着的是一个时间，目的是询问服务器在这个时间之后是否有修改，如果有修改，则向服务器端请求；如果没有修改则由代理从缓存中找到文件数据发送给客户端，并在返回的时候状态码改为 304 Not Modified。

```

7583 118.7055... 2001:da8:b800:253::c0a8:3209 2001:250:fe01:130:5dde:2d92:430f:d3bf HTTP 723 Continuation
7587 118.7559... 2001:250:fe01:130:5dde:2d92:430f:d3bf 2600:140e:6:99d::21cc HTTP 301 GET / HTTP/1.1
7592 118.9581... 2600:140e:6:99d::21cc 2001:250:fe01:130:5dde:2d92:430f:d3bf HTTP 337 HTTP/1.1 304 Not Modified
7603 119.1863... 2001:250:fe01:130:5dde:2d92:430f:d3bf 2600:140e:6::17db:ac71 HTTP 327 GET /DSTROOTCAX3CRL.CFI HTTP/1.1
7606 119.3987... 2600:140e:6::17db:ac71 2001:250:fe01:130:5dde:2d92:430f:d3bf HTTP 342 HTTP/1.1 304 Not Modified
7620 120.0056... 2001:250:fe01:130:5dde:2d92:430f:d3bf 2001:da8:b800:253::c0a8:3209 HTTP 606 GET / HTTP/1.1
7672 120.0693... 2001:da8:b800:253::c0a8:3209 2001:250:fe01:130:5dde:2d92:430f:d3bf HTTP 1491 HTTP/1.1 200 OK (text/html)
7694 120.2673... 172.20.162.34 109.70.240.130 HTTP 447 GET /VA/AUTH-ROOT/MFEWtzBNMeswST
7699 120.9838... 109.70.240.130 172.20.162.34 OSCP 749 Response
7756 137.3463... 172.20.162.34 183.201.240.147 HTTP 229 GET /per-plugin/dl/addons/list/v
7758 137.3862... 183.201.240.147 172.20.162.34 HTTP 1836 HTTP/1.1 200 OK
7767 138.1386... 2001:111:2002::c2 2001:250:fe01:130:5dde:2d92:430f:d3bf HTTP 787 HTTP/1.1 307 Temporary Redirect
    
```

```

Internet Protocol Version 6, Src: 2001:250:fe01:130:5dde:2d92:430f:d3bf, Dst: 2600:140e:6:99d::21cc
Transmission Control Protocol, Src Port: 12180, Dst Port: 80, Seq: 1, Ack: 1, Len: 227
Hypertext Transfer Protocol
  GET / HTTP/1.1\r\n
    Cache-Control: max-age = 3600\r\n
    Connection: Keep-Alive\r\n
    Accept: */*\r\n
    If-Modified-Since: Mon, 26 Jul 2021 16:20:55 GMT\r\n
    If-None-Match: "60fee0e7-2cd"\r\n
    User-Agent: Microsoft-CryptoAPI/10.0\r\n
    Host: x1.c.lencr.org\r\n
    
```

- d) 服务器对较晚的HTTP GET 请求的响应中的HTTP 状态码是多少？服务器是否明确返回了文件的内容？请解释。

较晚的 GET 请求的 HTTP 状态码是 304。

服务器不会明确返回文件，因为根据前面HTTP 的GET 请求中 IF-MODIFIED-SINCE 字段内的时间，服务器返回结果为304 Not Modified，这说明客户端会使用本地没有过期的缓存文件。

```

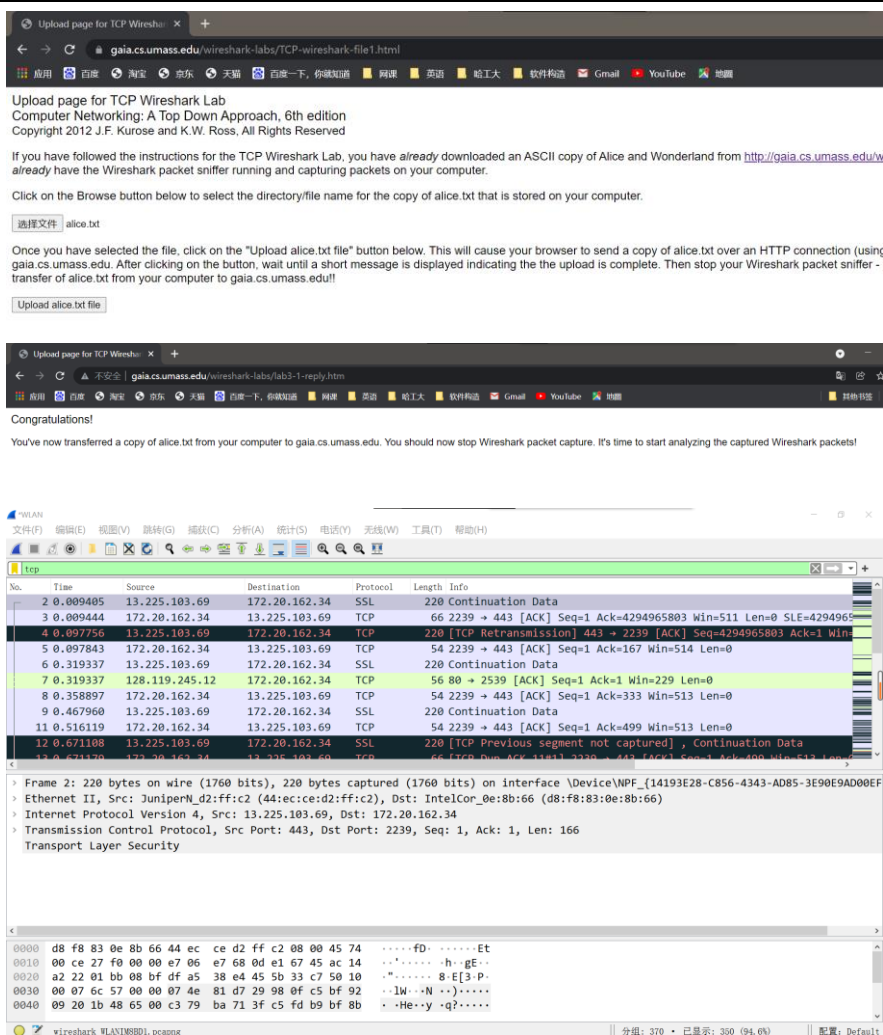
7583 118.7055... 2001:da8:b800:253::c0a8:3209 2001:250:fe01:130:5dde:2d92:430f:d3bf HTTP 723 Continuation
7587 118.7559... 2001:250:fe01:130:5dde:2d92:430f:d3bf 2600:140e:6:99d::21cc HTTP 301 GET / HTTP/1.1
7592 118.9581... 2600:140e:6:99d::21cc 2001:250:fe01:130:5dde:2d92:430f:d3bf HTTP 337 HTTP/1.1 304 Not Modified
7603 119.1863... 2001:250:fe01:130:5dde:2d92:430f:d3bf 2600:140e:6::17db:ac71 HTTP 327 GET /DSTROOTCAX3CRL.CFI HTTP/1.1
7606 119.3987... 2600:140e:6::17db:ac71 2001:250:fe01:130:5dde:2d92:430f:d3bf HTTP 342 HTTP/1.1 304 Not Modified
7620 120.0056... 2001:250:fe01:130:5dde:2d92:430f:d3bf 2001:da8:b800:253::c0a8:3209 HTTP 606 GET / HTTP/1.1
7672 120.0693... 2001:da8:b800:253::c0a8:3209 2001:250:fe01:130:5dde:2d92:430f:d3bf HTTP 1491 HTTP/1.1 200 OK (text/html)
7694 120.2673... 172.20.162.34 109.70.240.130 HTTP 447 GET /VA/AUTH-ROOT/MFEWtzBNMeswST
7699 120.9838... 109.70.240.130 172.20.162.34 OSCP 749 Response
7756 137.3463... 172.20.162.34 183.201.240.147 HTTP 229 GET /per-plugin/dl/addons/list/v
7758 137.3862... 183.201.240.147 172.20.162.34 HTTP 1836 HTTP/1.1 200 OK
7767 138.1386... 2001:111:2002::c2 2001:250:fe01:130:5dde:2d92:430f:d3bf HTTP 787 HTTP/1.1 307 Temporary Redirect
    
```

```

Internet Protocol Version 6, Src: 2001:250:fe01:130:5dde:2d92:430f:d3bf, Dst: 2600:140e:6:99d::21cc
Transmission Control Protocol, Src Port: 12180, Dst Port: 80, Seq: 1, Ack: 1, Len: 227
Hypertext Transfer Protocol
  GET / HTTP/1.1\r\n
    Cache-Control: max-age = 3600\r\n
    Connection: Keep-Alive\r\n
    Accept: */*\r\n
    If-Modified-Since: Mon, 26 Jul 2021 16:20:55 GMT\r\n
    If-None-Match: "60fee0e7-2cd"\r\n
    User-Agent: Microsoft-CryptoAPI/10.0\r\n
    Host: x1.c.lencr.org\r\n
    
```

### (三) TCP分析

- 1) 俘获大量的由本地主机到远程服务器的TCP分组
  - a) 启动浏览器，打开<http://gaia.cs.umass.edu/wireshark-labs/alice.txt>网页，得到ALICE'S ADVENTURES IN WONDERLAND文本，将该文件保存到你的主机上。
  - b) 打开<https://gaia.cs.umass.edu/wireshark-labs/TCP-wireshark-file1.html>网页。
  - c) 启动Wireshark，开始分组俘获。
  - d) 在浏览器中，单击“Upload alice.txt file”按钮，将文件上传到gaia.cs.umass.edu服务器，一旦文件上传完毕，一个简短的贺词信息将显示在你的浏览器窗口中。
  - e) 停止俘获。



## 2) 浏览追踪信息

在显示筛选规则中输入“tcp”，可以看到在本地主机和服务器之间传输的一系列tcp和http报文，你应该能看到包含SYN报文的三次握手。也可以看到有主机向服务器发送的一个HTTP POST报文和一系列的“http continuation”报文。

**根据操作思考以下问题：**

- a) 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？

132	7.898794	172.20.26.96	128.119.245.12	TCP	1514	50468 → 80 [ACK] Seq=613 Ack=1 Win=262144 Len=1460 [TCP segment of a re...
133	7.898794	172.20.26.96	128.119.245.12	TCP	1514	50468 → 80 [ACK] Seq=2073 Ack=1 Win=262144 Len=1460 [TCP segment of a re...
134	7.898794	172.20.26.96	128.119.245.12	TCP	1514	50468 → 80 [ACK] Seq=3533 Ack=1 Win=262144 Len=1460 [TCP segment of a re...
135	7.898796	172.20.26.96	128.119.245.12	TCP	1514	50468 → 80 [ACK] Seq=4993 Ack=1 Win=262144 Len=1460 [TCP segment of a re...
136	7.898796	172.20.26.96	128.119.245.12	TCP	1514	50468 → 80 [ACK] Seq=6453 Ack=1 Win=262144 Len=1460 [TCP segment of a re...
137	7.898797	172.20.26.96	128.119.245.12	TCP	1514	50468 → 80 [ACK] Seq=7913 Ack=1 Win=262144 Len=1460 [TCP segment of a re...
138	7.898797	172.20.26.96	128.119.245.12	TCP	1514	50468 → 80 [ACK] Seq=9373 Ack=1 Win=262144 Len=1460 [TCP segment of a re...

客户端主机IP地址为172.20.26.96，TCP端口号为50468

- b) 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？

132	7.898794	172.20.26.96	128.119.245.12	TCP	1514	50468 → 80 [ACK] Seq=613 Ack=1 Win=262144 Len=1460 [TCP segment of a re...
133	7.898794	172.20.26.96	128.119.245.12	TCP	1514	50468 → 80 [ACK] Seq=2073 Ack=1 Win=262144 Len=1460 [TCP segment of a re...
134	7.898794	172.20.26.96	128.119.245.12	TCP	1514	50468 → 80 [ACK] Seq=3533 Ack=1 Win=262144 Len=1460 [TCP segment of a re...
135	7.898796	172.20.26.96	128.119.245.12	TCP	1514	50468 → 80 [ACK] Seq=4993 Ack=1 Win=262144 Len=1460 [TCP segment of a re...
136	7.898796	172.20.26.96	128.119.245.12	TCP	1514	50468 → 80 [ACK] Seq=6453 Ack=1 Win=262144 Len=1460 [TCP segment of a re...
137	7.898797	172.20.26.96	128.119.245.12	TCP	1514	50468 → 80 [ACK] Seq=7913 Ack=1 Win=262144 Len=1460 [TCP segment of a re...
138	7.898797	172.20.26.96	128.119.245.12	TCP	1514	50468 → 80 [ACK] Seq=9373 Ack=1 Win=262144 Len=1460 [TCP segment of a re...
139	7.898797	172.20.26.96	128.119.245.12	TCP	1514	50468 → 80 [ACK] Seq=10833 Ack=1 Win=262144 Len=1460 [TCP segment of a r...
140	7.898797	172.20.26.96	128.119.245.12	TCP	1514	50468 → 80 [ACK] Seq=12293 Ack=1 Win=262144 Len=1460 [TCP segment of a r...

服务器的IP地址为128.119.245.12，端口号为80。

## 3) TCP基础

### 根据操作思考以下问题：

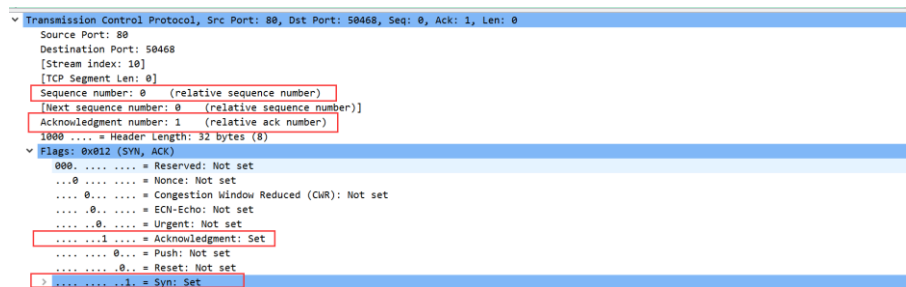
- a) 客户服务器之间用于初始化TCP连接的TCP SYN报文段的序号（sequence number）是多少？在该报文段中，是用什么来标示该报文段是SYN报文段的？



客户服务器之间用于初始化TCP连接的TCP SYN报文段序号是0

报文段中利用一个SYN标志位，该标志位置1时，标识该报文段是SYN报文段

- b) 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？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？



服务器发送的SYNACK报文段序号为0

报文段中，Acknowledgement字段的值为1

通过将客户端发送过来的报文段的 seq+1 得到的ACK的值

通过将 ACK 标志位和 SYN 标志位同时置 1 来标识该报文段是SYNACK 报文段

- c) 你能从捕获的数据包中分析出tcp三次握手过程吗？

172.20.26.96	128.119.245.12	TCP	66 50468 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM=1
128.119.245.12	172.20.26.96	TCP	66 80 → 50468 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 W
172.20.26.96	128.119.245.12	TCP	54 50468 → 80 [ACK] Seq=1 Ack=1 Win=262144 Len=0

第一次握手：客户端会向服务器发送一个TCP SYN报文段，其中SYN标志位置1，初始序列号Seq为0，不携带其他任何数据，请求与服务器建立TCP连接

第二次握手：服务器收到SYN报文段，同意建立连接，则回复一个SYNACK报文段，其中SYN和ACK标志位都置1，初始序列号Seq为0，Ack为1，以此响应客户端请求

第三次握手：客户端收到服务器的SYNACK报文段，回复ACK报文段，其中SYN标志位置0，ACK标志位置1，Seq=1，Ack=1，可以携带数据

- d) 包含HTTP POST命令的TCP报文段的序号是多少？

序列号为152453



284	8.638799	172.20.26.96	128.119.245.12	HTTP	542	POST /wireshark-labs/lab3-1-reply.htm HTTP/1.1 (text/plain)
316	8.898291	128.119.245.12	172.20.26.96	HTTP	831	HTTP/1.1 200 OK (text/html)

```

> Frame 284: 542 bytes on wire (4336 bits), 542 bytes captured (4336 bits) on interface 0
> Ethernet II, Src: IntelCor_ba:dd:30 (f4:8c:50:ba:dd:30), Dst: Ruijiele_a5:e2:d3 (58:69:6c:a5:e2:d3)
> Internet Protocol Version 4, Src: 172.20.26.96, Dst: 128.119.245.12
> Transmission Control Protocol, Src Port: 50468, Dst Port: 80, Seq: 152453, Ack: 1, Len: 488
  Source Port: 50468
  Destination Port: 80
  [Stream index: 10]
  [TCP Segment Len: 488]
  Sequence number: 152453 (relative sequence number)
  [Next sequence number: 152941 (relative sequence number)]
  Acknowledgment number: 1 (relative ack number)
  0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x018 (PSH, ACK)
    
```

- e) 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？

284	8.638799	172.20.26.96	128.119.245.12	HTTP	542	POST /wireshark-labs/lab3-1-reply.htm HTTP/1.1 (text/plain)
317	8.898344	172.20.26.96	128.119.245.12	TCP	54	50468 → 80 [ACK] Seq=152941 Ack=778 Win=261120 Len=0

```

> [Timestamps]
TCP payload (488 bytes)
TCP segment data (488 bytes)
> [106 Reassembled TCP Segments (152940 bytes): #131(612), #132(1460), #133(1460), #134(1460), #135(1460), #136(1460), #137(1460), #138(1460), #139(1460)]
  [Frame: 131, payload: 0-611 (612 bytes)]
  [Frame: 132, payload: 612-2071 (1460 bytes)]
  [Frame: 133, payload: 2072-3531 (1460 bytes)]
  [Frame: 134, payload: 3532-4991 (1460 bytes)]
  [Frame: 135, payload: 4992-6451 (1460 bytes)]
  [Frame: 136, payload: 6452-7911 (1460 bytes)]
  [Frame: 137, payload: 7912-9371 (1460 bytes)]
    
```

134	8.898794	172.20.26.96	128.119.245.12	TCP	1514	50468 → 80 [ACK] Seq=6453 Ack=1 Win=262144 Len=1460 [TCP segment of a re..]
135	7.898796	172.20.26.96	128.119.245.12	TCP	1514	50468 → 80 [ACK] Seq=4993 Ack=1 Win=262144 Len=1460 [TCP segment of a re..]
136	7.898796	172.20.26.96	128.119.245.12	TCP	1514	50468 → 80 [ACK] Seq=6453 Ack=1 Win=262144 Len=1460 [TCP segment of a re..]
137	7.898797	172.20.26.96	128.119.245.12	TCP	1514	50468 → 80 [ACK] Seq=7913 Ack=1 Win=262144 Len=1460 [TCP segment of a re..]
138	7.898797	172.20.26.96	128.119.245.12	TCP	1514	50468 → 80 [ACK] Seq=9373 Ack=1 Win=262144 Len=1460 [TCP segment of a re..]
139	7.898797	172.20.26.96	128.119.245.12	TCP	1514	50468 → 80 [ACK] Seq=10833 Ack=1 Win=262144 Len=1460 [TCP segment of a re..]

```

> Internet Protocol Version 4, Src: 172.20.26.96, Dst: 128.119.245.12
> Transmission Control Protocol, Src Port: 50468, Dst Port: 80, Seq: 6453, Ack: 1, Len: 1460
  Source Port: 50468
  Destination Port: 80
  [Stream index: 10]
  [TCP Segment Len: 1460]
  Sequence number: 6453 (relative sequence number)
  [Next sequence number: 7913 (relative sequence number)]
  Acknowledgment number: 1 (relative ack number)
  0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x010 (ACK)
  Window size value: 1024
    
```

第六个报文段的序号为6453

在HTTP POST命令之前，HTTP连接建立之后发送的

该报文段对应的ACK是在该报文段发送之后，HTTP POST命令之后接收的

- f) 前六个TCP报文段的长度各是多少？

284	8.638799	172.20.26.96	128.119.245.12	HTTP	542	POST /wireshark-labs/lab3-1-reply.htm HTTP/1.1 (text/plain)
317	8.898344	172.20.26.96	128.119.245.12	TCP	54	50468 → 80 [ACK] Seq=152941 Ack=778 Win=261120 Len=0

```

> [Timestamps]
TCP payload (488 bytes)
TCP segment data (488 bytes)
> [106 Reassembled TCP Segments (152940 bytes): #131(612), #132(1460), #133(1460), #134(1460), #135(1460), #136(1460), #137(1460), #138(1460), #139(1460)]
  [Frame: 131, payload: 0-611 (612 bytes)]
  [Frame: 132, payload: 612-2071 (1460 bytes)]
  [Frame: 133, payload: 2072-3531 (1460 bytes)]
  [Frame: 134, payload: 3532-4991 (1460 bytes)]
  [Frame: 135, payload: 4992-6451 (1460 bytes)]
  [Frame: 136, payload: 6452-7911 (1460 bytes)]
  [Frame: 137, payload: 7912-9371 (1460 bytes)]
    
```

第一个TCP报文段长度为612字节，剩下的五个均为1460字节

- g) 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？

244	8.638277	128.119.245.12	172.20.26.96	TCP	60	80 → 50468 [ACK] Seq=1 Ack=95513 Win=182528 Len=0
245	8.638278	128.119.245.12	172.20.26.96	TCP	60	80 → 50468 [ACK] Seq=1 Ack=96973 Win=181632 Len=0
246	8.638744	172.20.26.96	128.119.245.12	TCP	1514	50468 → 80 [ACK] Seq=96973 Ack=1 Win=262144 Len=1460 [TCP segment of a re..]
247	8.638755	172.20.26.96	128.119.245.12	TCP	1514	50468 → 80 [ACK] Seq=98433 Ack=1 Win=262144 Len=1460 [TCP segment of a re..]
248	8.638756	172.20.26.96	128.119.245.12	TCP	1514	50468 → 80 [ACK] Seq=99893 Ack=1 Win=262144 Len=1460 [TCP segment of a re..]
249	8.638757	172.20.26.96	128.119.245.12	TCP	1514	50468 → 80 [ACK] Seq=101353 Ack=1 Win=262144 Len=1460 [TCP segment of a re..]
250	8.638759	172.20.26.96	128.119.245.12	TCP	1514	50468 → 80 [ACK] Seq=102813 Ack=1 Win=262144 Len=1460 [TCP segment of a re..]
251	8.638760	172.20.26.96	128.119.245.12	TCP	1514	50468 → 80 [ACK] Seq=104273 Ack=1 Win=262144 Len=1460 [TCP segment of a re..]
252	8.638761	172.20.26.96	128.119.245.12	TCP	1514	50468 → 80 [ACK] Seq=105733 Ack=1 Win=262144 Len=1460 [TCP segment of a re..]
253	8.638762	172.20.26.96	128.119.245.12	TCP	1514	50468 → 80 [ACK] Seq=107193 Ack=1 Win=262144 Len=1460 [TCP segment of a re..]

```

0101 .... = Header Length: 20 bytes (5)
> Flags: 0x010 (ACK)
Window size value: 1419
[Calculated window size: 181632]
[Window size scaling factor: 128]
Checksum: 0x2516 [unverified]
[Checksum Status: Unverified]
Urgent pointer: 0
> [SEQ/ACK analysis]
    
```

接收端公示的最小的可用缓存空间是181632字节

```

Window size value: 1477
[Calculated window size: 189056]

Window size value: 1523
[Calculated window size: 194944]

Window size value: 1591
[Calculated window size: 203648]

Window size value: 1865
[Calculated window size: 238720]

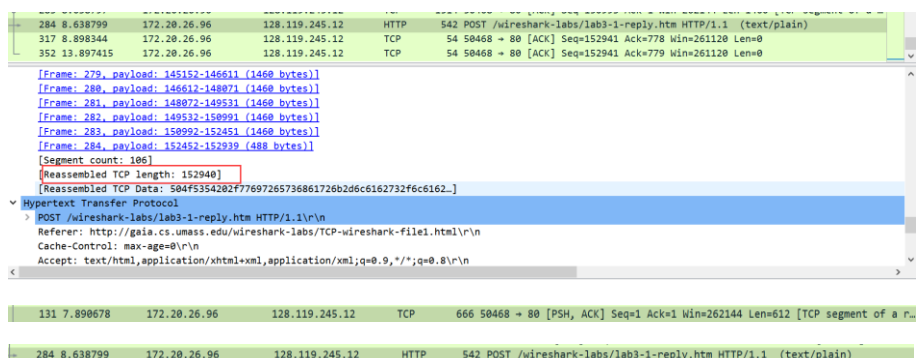
Window size value: 2139
[Calculated window size: 273792]
    
```

接收端的缓存不会不够用，如图可见，接收端的缓存空间一直在增大

h) 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？

没有重传的报文段，因为发送的报文的序列号没有重复的序列号出现；

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



TCP 段的总长度为 152940 字节

发送第一个段的时间为 7.890678

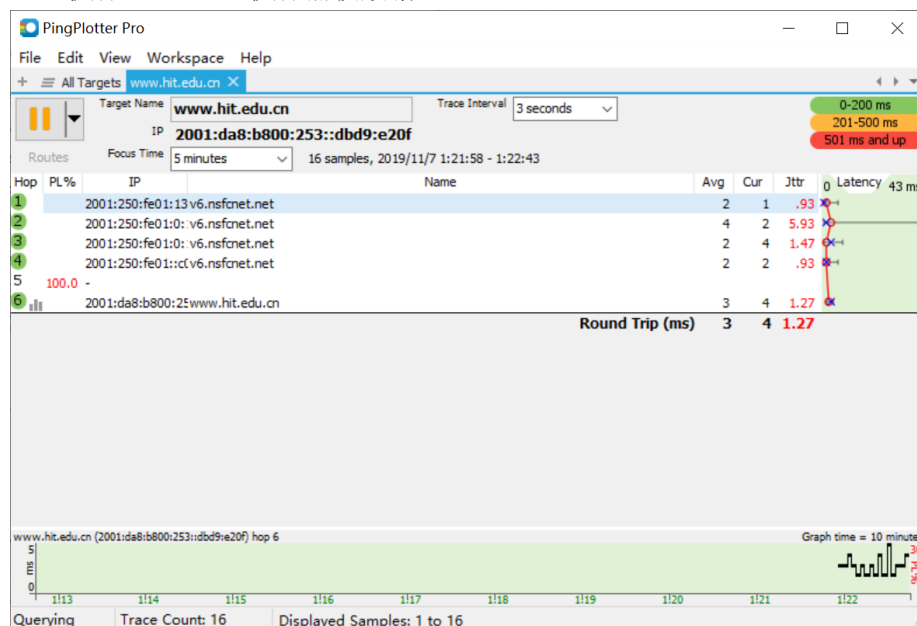
发送最后一个段的时间为 8.638799

时间间隔为  $8.638799 - 7.890678 = 0.748121$  ms

因此吞吐率为  $152940 * 8 / (0.748121 * 10^{-3}) \approx 1.635$  Gbps

#### (四) IP分析

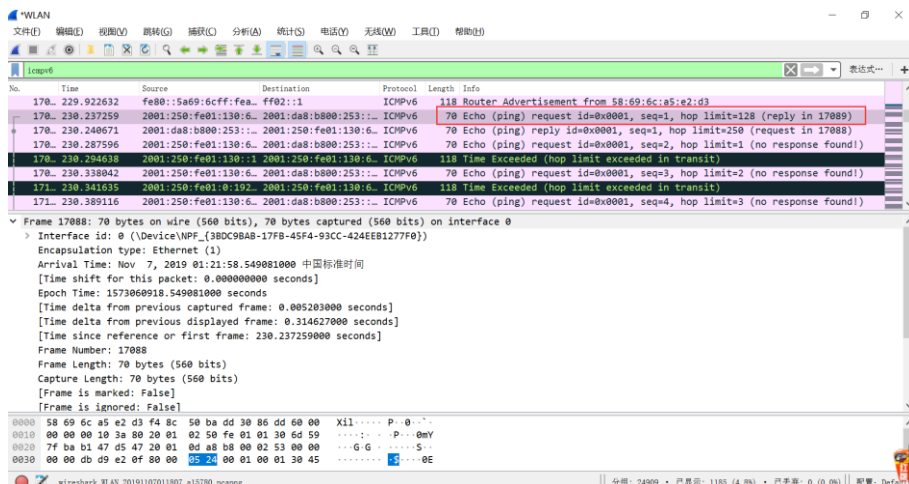
1) 通过执行tracert 执行捕获数据包



2) 对捕获的数据包进行分析

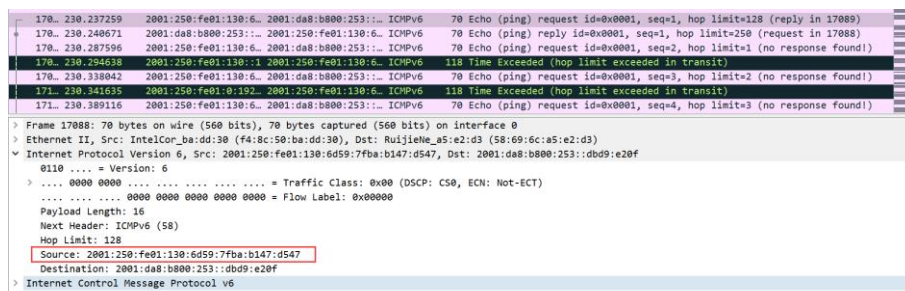
在你的捕获窗口中，应该能看到由你的主机发出的一系列ICMP Echo Request包

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。



思考下列问题：

a) 你主机的IP地址是什么？

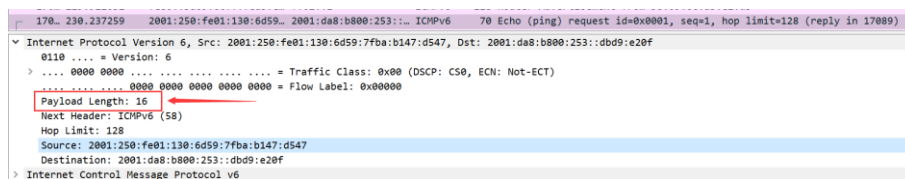


主机的IP地址为2001:250:fe01:130:6d59:7fba:b147:d547

b) 在IP数据包头中，上层协议（upper layer）字段的值是什么？

IPv6数据包没有上层协议upper layer

c) 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？



IPv6头部有40字节；

该IP数据包净载为16字节；

如图。

d) 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

没有分片，因为IPv6数据包不允许分片，没有标识位，也没有片偏移。

3) 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）



```

[2 IPv6 Fragments (1440 bytes): #903(1448), #902(32)]
[Frame: 903, payload: 0-1447 (1448 bytes)]
[Frame: 902, payload: 1448-1479 (32 bytes)]
[Fragment count: 2]

```

More Fragments 位置 1, 说明被分片, 且不是最后一片

Offset 为 0 说明是第一片分片

分片的长度为 1448 字节

- 6) 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:

- a) 原始数据包被分成了多少片?

```

[3 IPv6 Fragments (3460 bytes): #886(1448), #887(1448), #888(564)]
[Frame: 886, payload: 0-1447 (1448 bytes)]
[Frame: 887, payload: 1448-2895 (1448 bytes)]
[Frame: 888, payload: 2896-3459 (564 bytes)]
[Fragment count: 3]

```

分成3片

- b) 这些分片中IP数据报头部哪些字段发生了变化?

```

Fragment Header for IPv6
Next header: ICMPv6 (58)
Reserved octet: 0x00
0000 1011 0101 0... = Offset: 362 (2896 bytes)
.... .... .... .00. = Reserved bits: 0
.... .... .... ...0 = More Fragments: No
Identification: 0x401e3640

```

偏移量字段、More Fragment字段发生了变化。

## (五) 抓取ARP数据包

- 1) 利用MS-DOS命令查看主机APR缓存的内容

```

C:\Users\ASUS>arp -a
接口: 172.20.26.96 --- 0x8
Internet 地址      物理地址          类型
172.20.0.1          58-69-6c-a5-e2-d3 动态
172.20.37.163       58-69-6c-a5-e2-d3 动态
172.20.73.91        58-69-6c-a5-e2-d3 动态
172.20.119.15       58-69-6c-a5-e2-d3 动态
172.20.127.255      ff-ff-ff-ff-ff-ff 静态
224.0.0.22          01-00-5e-00-00-16 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
239.11.20.1         01-00-5e-0b-14-01 静态
239.255.255.250     01-00-5e-7f-ff-fa 静态
255.255.255.255     ff-ff-ff-ff-ff-ff 静态

```

- 2) 在命令行模式下输入: ping 192.168.1.82 (或其他IP地址)

```

C:\Users\ASUS>ping 192.34.56.23

正在 Ping 192.34.56.23 具有 32 字节的数据:
来自 192.34.56.23 的回复: 字节=32 时间=240ms TTL=46
来自 192.34.56.23 的回复: 字节=32 时间=240ms TTL=46
来自 192.34.56.23 的回复: 字节=32 时间=240ms TTL=46
来自 192.34.56.23 的回复: 字节=32 时间=239ms TTL=46

192.34.56.23 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 = 239ms, 最长 = 240ms, 平均 = 239ms

```

- 3) 启动Wireshark, 开始分组俘获。



### 思考下面问题

#### a) 说明ARP缓存中每一列的含义是什么？

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）。

#### b) 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:

##### i. 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



由 9 部分构成：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）。

##### ii. 如何判断一个ARP数据是请求包还是应答包

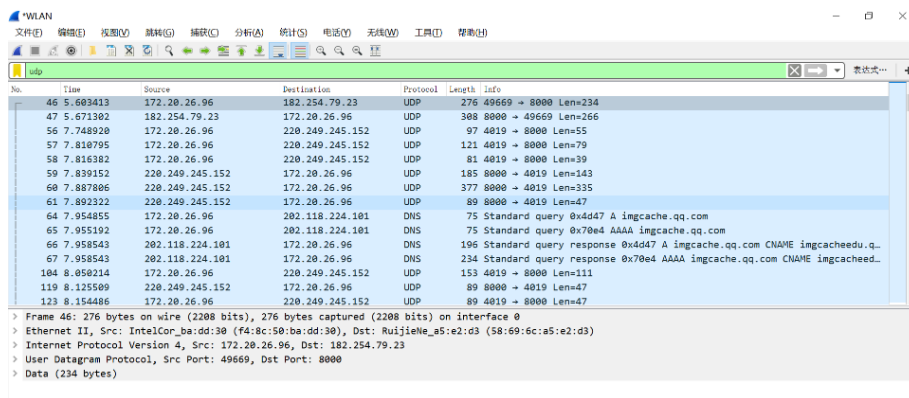
通过 OP 字段。当 OP 字段值为 0x0001 时是请求包，当 OP 字段值为 0x0002时是应答包。

##### iii. 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？

因为进行 ARP 查询时是不知道目的 IP 地址对应的 MAC 地址的，因此需要广播查询；而发送ARP 响应报文时是知道查询主机的 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，因此 ARP 响应要在一个有着明确目的局域网地址的帧中传送。

### (六) 抓取UDP数据包

- 1) 启动 Wireshark，开始分组捕获；
- 2) 发送 QQ 消息给你的好友；
- 3) 停止Wireshark组捕获
- 4) 在显示筛选规则中输入“udp”并展开数据包的细节，如图所示。



### 分析QQ通讯中捕获到的UDP数据包。根据操作思考以下问题：

a) 消息是基于UDP的还是TCP的？

UDP

b) 你的主机ip地址是什么？目的主机ip地址是什么？

56 7.748920	172.20.26.96	220.249.245.152	UDP	97 4019 → 8000 Len=55
57 7.810795	172.20.26.96	220.249.245.152	UDP	121 4019 → 8000 Len=79
58 7.816382	172.20.26.96	220.249.245.152	UDP	81 4019 → 8000 Len=39

主机IP地址为172.20.26.96，目的主机IP地址为220.249.245.152

c) 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？

56 7.748920	172.20.26.96	220.249.245.152	UDP	97 4019 → 8000 Len=55
57 7.810795	172.20.26.96	220.249.245.152	UDP	121 4019 → 8000 Len=79
58 7.816382	172.20.26.96	220.249.245.152	UDP	81 4019 → 8000 Len=39

主机端口号为4019，QQ服务器端口号为8000

d) 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？

来源端口	目的端口	长度域	校验和
------	------	-----	-----

56 7.748920	172.20.26.96	220.249.245.152	UDP	97 4019 → 8000 Len=55
57 7.810795	172.20.26.96	220.249.245.152	UDP	121 4019 → 8000 Len=79
58 7.816382	172.20.26.96	220.249.245.152	UDP	81 4019 → 8000 Len=39
59 7.839152	220.249.245.152	172.20.26.96	UDP	185 8000 → 4019 Len=143
60 7.887806	220.249.245.152	172.20.26.96	UDP	377 8000 → 4019 Len=335

Frame 57: 121 bytes on wire (968 bits), 121 bytes captured (968 bits) on interface 0	
Ethernet II, Src: IntelCor_ba:dd:30 (f4:8c:50:ba:dd:30), Dst: Ruijiele_a5:e2:d3 (58:09:6c:a5:e2:d3)	
Internet Protocol Version 4, Src: 172.20.26.96, Dst: 220.249.245.152	
User Datagram Protocol, Src Port: 4019, Dst Port: 8000	
Source Port: 4019	Destination Port: 8000
Length: 87	Checksum: 0x1b53 [unverified]
[Checksum Status: Unverified]	
[Stream index: 1]	
[Timestamps]	
Data (79 bytes)	

首部8个字节。源端口号2字节；目的端口号2字节；长度2字节；校验和2字节

e) 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？

因为服务器需返回接收的结果给客户端。

因为服务器只提供了一次返回的 ACK，所以不保证数据一定送达，即为不可靠传输；

对比前面的TCP可以看出，因为UDP 数据包没有序列号，不能像TCP 协议那样先三次握手建立连接再发送数据，这样每次只发送一个数据报，然后等待服务器响应。

### (七) 利用WireShark进行DNS协议分析

打开浏览器键入:www.baidu.com，并打开Wireshark进行抓包。

14 2.363902	172.20.53.231	202.118.224.101	UNS	73 Standard query 0x5be4 A www.baidu.com
15 2.364690	172.20.53.231	202.118.224.101	DNS	73 Standard query 0x3cdc AAAA www.baidu.com
16 2.369292	202.118.224.101	172.20.53.231	DNS	132 Standard query response 0x5be4 A www.baidu.c
17 2.369834	202.118.224.101	172.20.53.231	DNS	157 Standard query response 0x3cdc AAAA www.baid
115 2.549304	172.20.53.231	202.118.224.101	DNS	75 Standard query 0xcd5 A b1.bdstatic.com
116 2.549304	172.20.53.231	202.118.224.101	DNS	75 Standard query 0x7f65 A s1.bdstatic.com
117 2.549447	172.20.53.231	202.118.224.101	DNS	72 Standard query 0x31e2 A t1.baidu.com
118 2.549676	172.20.53.231	202.118.224.101	DNS	72 Standard query 0x26be AAAA t1.baidu.com
119 2.549868	172.20.53.231	202.118.224.101	DNS	75 Standard query 0x03be AAAA b1.bdstatic.com

1. 查询的目的地址均为相同的202.118.224.101
2. 可以知道这是哈工大的dns服务器，经过ip地址查询确实这是一个来自哈尔滨市南岗区的教务网ip地址。

### DNS查询报文:

```

v Internet Protocol Version 4, Src: 172.20.162.34, Dst: 202.118.224.100
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 76
    Identification: 0xe2bf (58047)
  > Flags: 0x00
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 128
    Protocol: UDP (17)
    Header Checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.20.162.34
    Destination Address: 202.118.224.100
v User Datagram Protocol, Src Port: 59243, Dst Port: 53
  Source Port: 59243
  Destination Port: 53
  Length: 56
  Checksum: 0xf95b [unverified]
  
```

### DNS回复报文:

```

v Internet Protocol Version 4, Src: 202.118.224.100, Dst: 172.20.162.34
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 394
    Identification: 0x37fb (14331)
  > Flags: 0x00
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 60
    Protocol: UDP (17)
    Header Checksum: 0x4c56 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 202.118.224.100
    Destination Address: 172.20.162.34
v User Datagram Protocol, Src Port: 53, Dst Port: 59243
  Source Port: 53
  Destination Port: 59243
  Length: 374
  Checksum: 0x2e70 [unverified]
  
```

### 心得体会:

1. 虽然本次实验比较复杂琐碎,耗时比较长,学会了如何合理安排时间(虽然上学期计统和软构已经够恶心人了,计网显得非常和善)
2. HTTP 协议、TCP 协议、UDP 协议、IP 协议、DNS 协议等等的报文格式有了更深入的了解,明白了每一个标志位的含义与作用,深入地了解了协议交互的过程以及工作方式;
3. 另外,我更加理解了网络协议实体间进行交互及报文交换的大致流程与情形,也学习了简单的 Wireshark 工具的抓包方法和 PingPlotter 工具的使用。