哈爾濱工業大學
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	袁野		院系	计算学部计算机科学与技术专业	
班级	1903102		学号	1190200122	
任课教师	刘亚维		指导教师	刘亚维	
实验地点	格物 207		实验时间	2021.11.21	
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					

计算机科学与技术学院 SINCE 1956...
School of Computer Science and Technology

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：

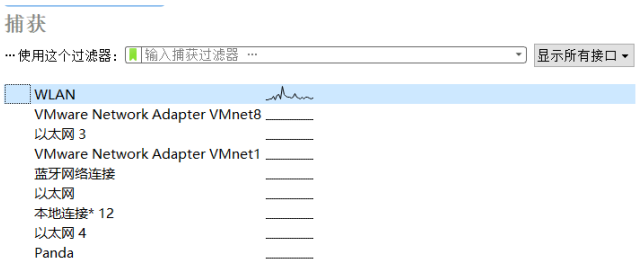
概述本次实验的主要内容，包含的实验项等。

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
6) 利用 Wireshark 分析 DNS 协议
7) 利用 Wireshark 分析 UDP 协议
8) 利用 Wireshark 分析 ARP 协议

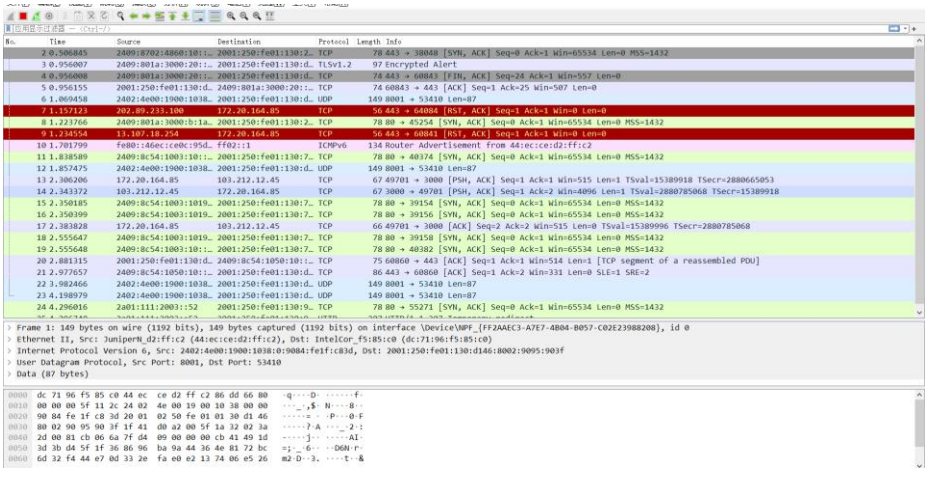
实验过程与结果：

1. Wireshark 的使用

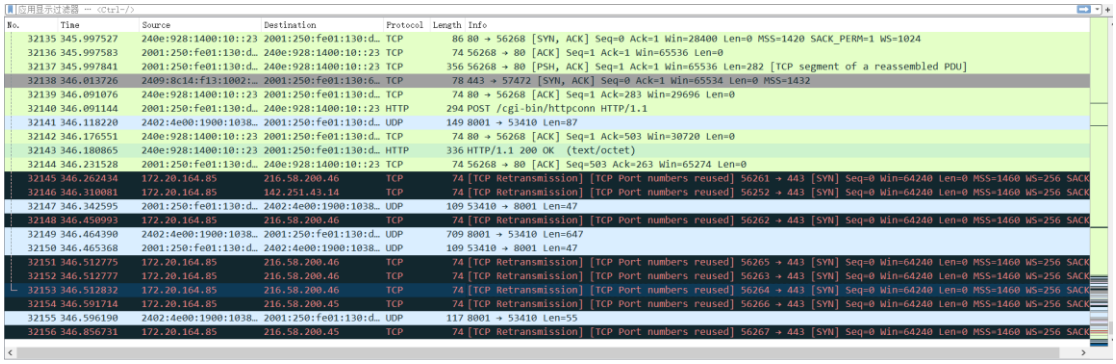
首先在Wireshark官网：https://www.wireshark.org/download.html下载Wireshark，之后捕获器选择接口进行捕获。



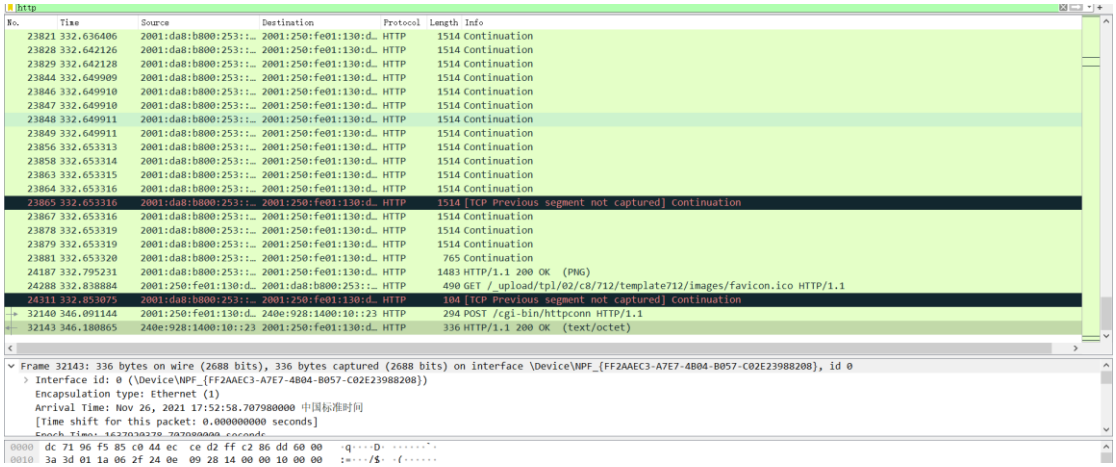
抓包界面如下所示



我们在浏览器中访问http://www.hit.edu.cn，待网页加载完成后，停止分组捕获。



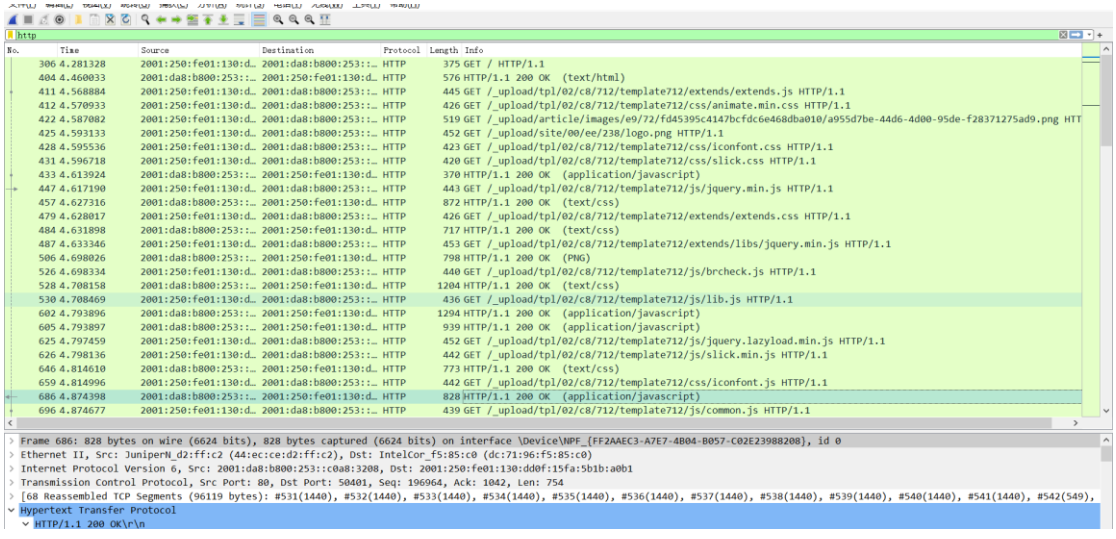
在过滤器中输入http进行过滤：



2. HTTP分析

2.1 HTTP GET/response 交互

首先清除浏览器的缓存，然后过滤出HTTP后重新开始捕获，在浏览器访问 http://jwes.hit.edu.cn/



根据俘获窗口内容，思考以下问题：

- **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？**

```
[Expert Info (Chat/Sequence): GET / HTTP/1.1\r\n]
    [GET / HTTP/1.1\r\n]
    [Severity level: Chat]
    [Group: Sequence]
Request Method: GET
```

由上截图内容可知，我的浏览器运行的是HTTP1.1，访问的服务器运行的是HTTP1.1。

- **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: zh-Hans-CN,zh-Hans;q=0.8,en-US;q=0.5,en;q=0.3\r\n
```

由上截图内容可知，简体中文，美国英语，通用英语

- **你的计算机的 IP 地址是多少？服务器 http://jwes.hit.edu.cn/ 的 IP 地址是多少？**

```
1604 48.185660   219.217.226.139   172.20.113.24     HTTP   176 HTTP/1.1 200 OK  (text/html)
1606 48.199918   172.20.113.24     219.217.226.139   HTTP   479 GET /resources/js/jquery/jquery-1.7.2.min.js HTTP/1.1
1607 48.201031   172.20.113.24     219.217.226.139   HTTP   451 GET /resources/css/common/style1.css HTTP/1.1
```

由以上截图内容可知，本机IP为172.20.113.24，服务器IP为219.217.226.139

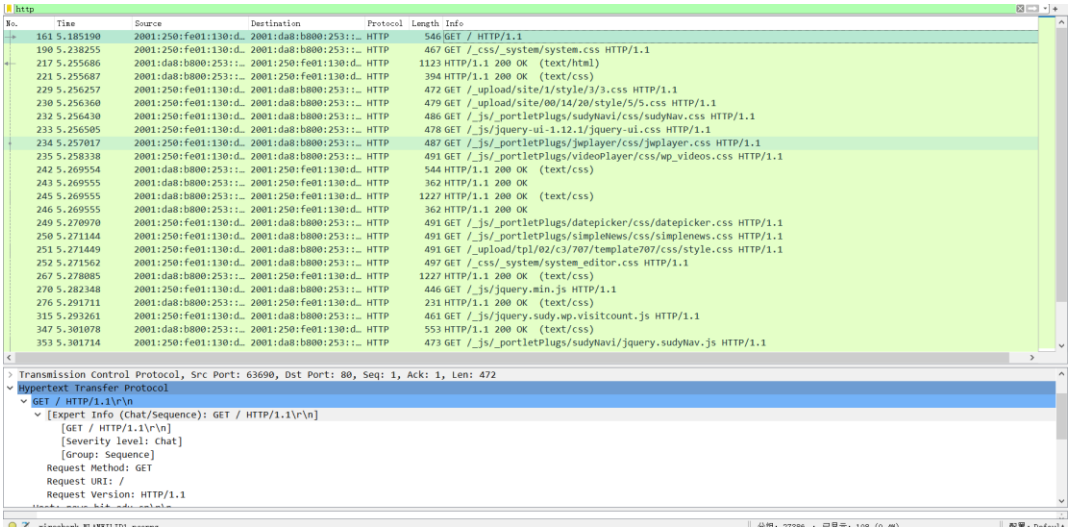
- **从服务器向你的浏览器返回的状态代码是多少？**

```
Hypertext Transfer Protocol
  HTTP/1.1 200 OK\r\n
    [Expert Info (Chat/Sequence): HTTP/1.1 200 OK\r\n]
      [HTTP/1.1 200 OK\r\n]
      [Severity level: Chat]
      [Group: Sequence]
    Response Version: HTTP/1.1
    Status Code: 200
    [Status Code Description: OK]
    Response Phrase: OK
  Server: Server          \r\n
  Pragma: no-cache\r\n
  Expires: Thu, 01 Jan 1970 00:00:00 GMT\r\n
```

由图中内容可知，返回的状态码为200

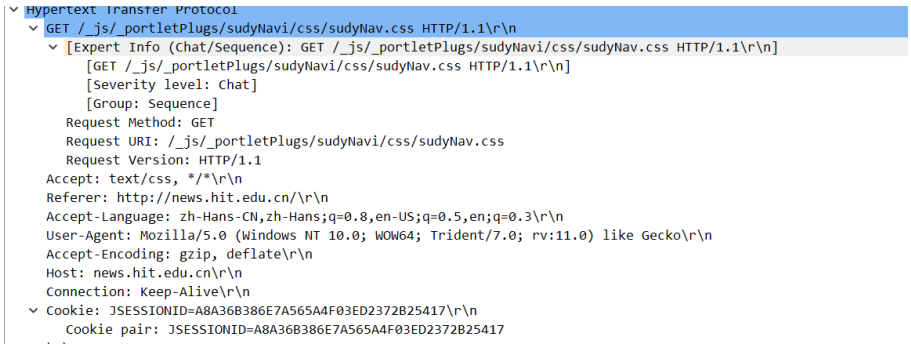
2.2 HTTP 条件 GET/response 交互

清除浏览器缓存后，重新开始捕获，并将过滤器设置为HTTP，通过浏览器http://news.hit.edu.cn/ ，待加载完成后再次访问该网址 ，得到报文。



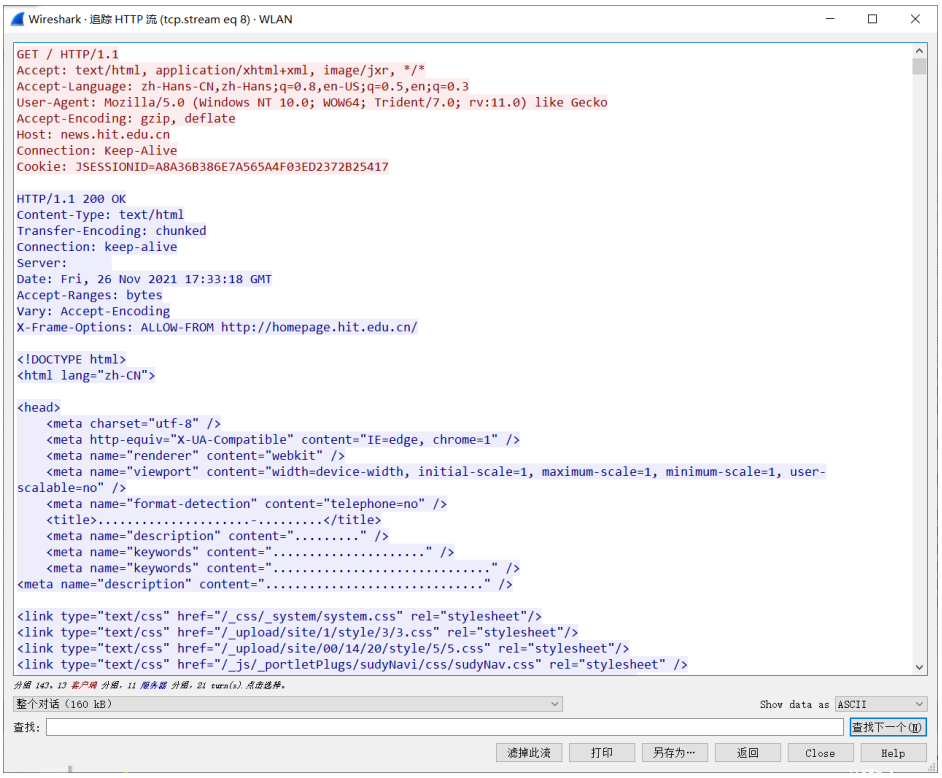
- **分析你的浏览器向服务器发出的第一个 HTTP GET 请求的内容，在该请求报文中，是否有一行**

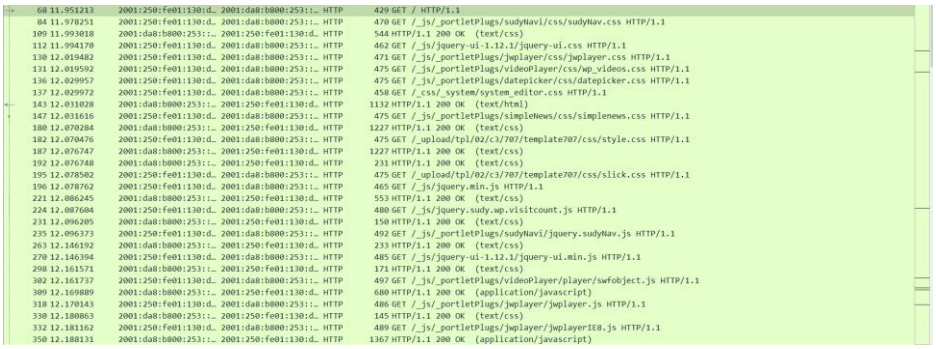
是：IF-MODIFIED-SINCE？



由以上截图中观察报文内容可知，报文中并没有 IF-MODIFIED-SINCE

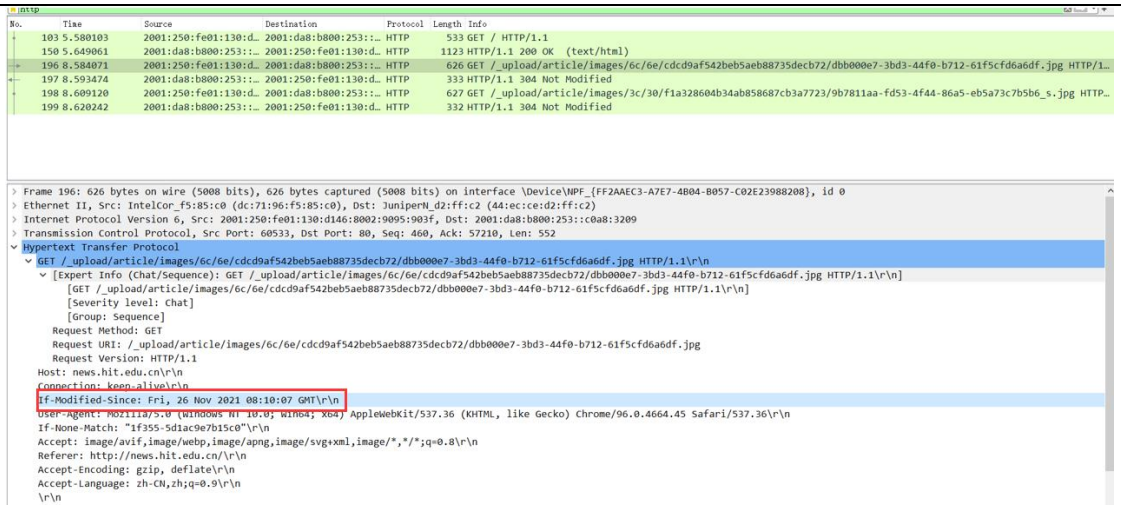
- **分析服务器响应报文的内容，服务器是否明确返回了文件的内容？如何获知？**





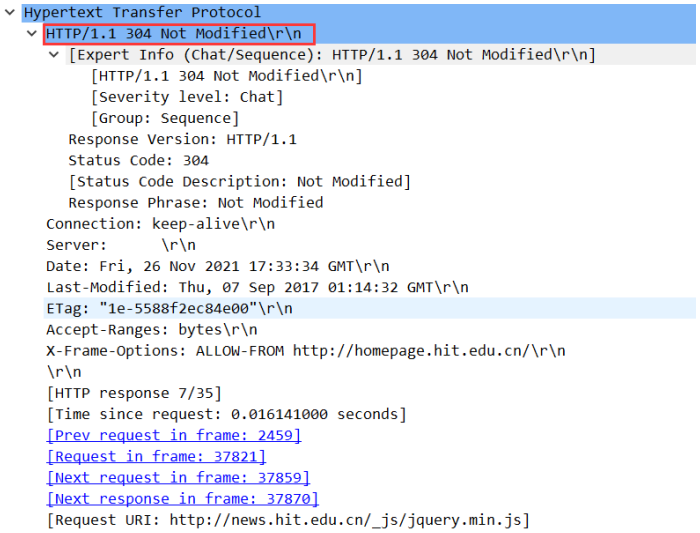
追踪该HTTP流可知服务器返回的所有json文件的状态码均为200，因此服务器明确返回了所有内容。

- **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？**



IF-MODIFIED-SINCE位置如图所示，后面跟着的是当前缓存最后一次更新的时间。

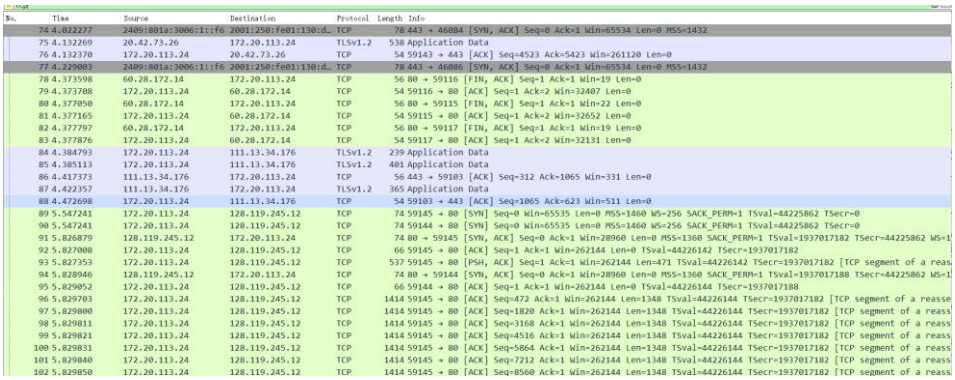
- **服务器对较晚的 HTTP GET 请求的响应中的 HTTP 状态代码是多少？服务器是否明确返回了文件的内容？请解释。**



状态码为304，不会明确返回文件内容，因为服务器经过比对发现本地缓存文件最后更新时间与服务器的文件最后更新时间一致，因此会认为No Modified，表示本地的缓存未过期。

3. TCP分析

通过向 gaia.cs.umass.edu 发送文件可以俘获大量的TCP分组



3.1 浏览追踪信息

- **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？**

```
> Internet Protocol Version 4, Src: 172.20.113.24, Dst: 128.119.245.12
> Transmission Control Protocol, Src Port: 59145, Dst Port: 80, Seq: 0, Len: 0
```

客户端主机的IP为172.20.113.24，TCP端口号为59145

- **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？**

服务器IP地址为128.119.245.12，TCP端口号为80

3.2 TCP基础

- **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？**

```
Sequence Number: 0    (relative sequence number)
Sequence Number (raw): 3050453715
[Next Sequence Number: 1    (relative sequence number)]
Acknowledgment Number: 0
Acknowledgment number (raw): 0
1010 .... = Header Length: 40 bytes (10)
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
     [Expert Info (Chat/Sequence): Connection establish request (SYN): server port 80]
        [Connection establish request (SYN): server port 80]
        [Severity level: Chat]
        [Group: Sequence]
   .... .... ...0 = Fin: Not set
```

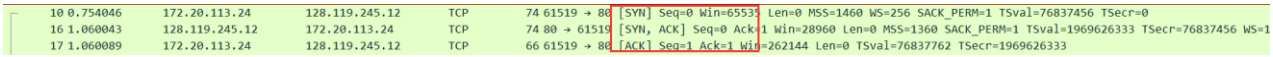
初始化TCP连接的TCP SYN报文段的序号是0；通过Flags标志位来标示该报文段是SYN报文段的

- **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？**

```
Sequence Number: 0    (relative sequence number)
Sequence Number (raw): 1932071241
[Next Sequence Number: 1    (relative sequence number)]
Acknowledgment Number: 1    (relative ack number)
Acknowledgment number (raw): 3329663186
1010 .... = Header Length: 40 bytes (10)
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
   [TCP Flags: ·····A··S·]
```

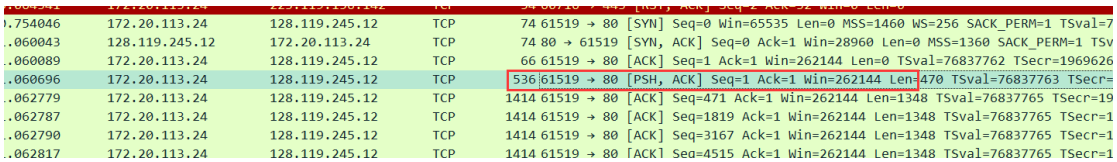
服务器向客户端发送的SYNACK报文段序号是1，Acknowledgment字段的值是1，服务器根据用户上一次发送的报文中的seq+1得到Acknowledgment的值，通过Flags中Syn和Acknowledgment位的值为1来确定该报文段是SYN ACK报文段

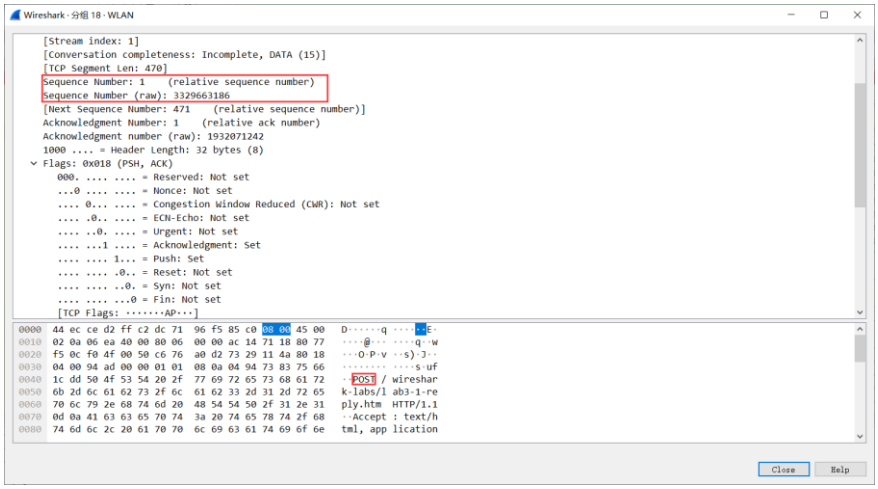
- **你能从捕获的数据包中分析出 tcp 三次握手过程吗？**



客户端先向服务器发送 seq=0 的建立连接的请求；然后服务器向客户端返回 seq=0，ack=1 的响应；最后客户端向服务器返回 seq=1，ack=1的确认报文。

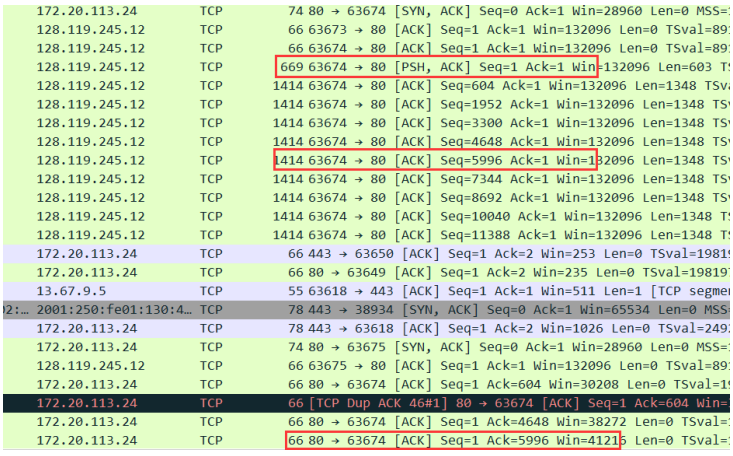
- **包含 HTTP POST 命令的 TCP 报文段的序号是多少？**

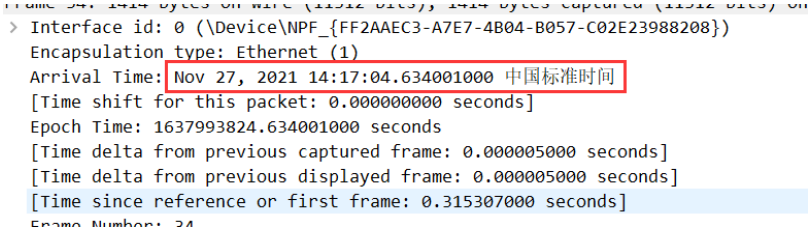




序列号为1，绝对序号为3329663186

- **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？**





```
Frame 49: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on in
> Interface id: 0 (\Device\NPF_{FF2AAEC3-A7E7-4B04-B057-C02E23988208})
  Encapsulation type: Ethernet (1)
  Arrival Time: Nov 27, 2021 14:17:04.948230000 中国标准时间
  [Time shift for this packet: 0.000000000 seconds]
  Epoch Time: 1637993824.948230000 seconds
  [Time delta from previous captured frame: 0.000000000 seconds]
  [Time delta from previous displayed frame: 0.000000000 seconds]
  [Time since reference or first frame: 0.629536000 seconds]
```

报文序号是5996，发送时间是 Nov 27, 2021 14:17:04.634001000 中国标准时间，对应ACK的接收时间为 Nov 27, 2021 14:17:04.948230000 中国标准时间。

- **前六个 TCP 报文段的长度各是多少？**

```
TCP      669 63674 → 80 [PSH, ACK] Seq=1 Ack=1 Win=132096 Len=603 TSval=8918288
TCP     1414 63674 → 80 [ACK] Seq=604 Ack=1 Win=132096 Len=1348 TSval=89182883
TCP     1414 63674 → 80 [ACK] Seq=1952 Ack=1 Win=132096 Len=1348 TSval=8918288
TCP     1414 63674 → 80 [ACK] Seq=3300 Ack=1 Win=132096 Len=1348 TSval=8918288
TCP     1414 63674 → 80 [ACK] Seq=4648 Ack=1 Win=132096 Len=1348 TSval=8918288
TCP     1414 63674 → 80 [ACK] Seq=5996 Ack=1 Win=132096 Len=1348 TSval=8918288
TCP     1414 63674 → 80 [ACK] Seq=7344 Ack=1 Win=132096 Len=1348 TSval=8918288
TCP     1414 63674 → 80 [ACK] Seq=8692 Ack=1 Win=132096 Len=1348 TSval=8918288
```

前六个TCP报文段长度分别为603，1348，1348，1348，1348，1348

- **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？**

```
TCP       74 80 → 63673 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=13
TCP       74 80 → 63674 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=13
TCP       66 63673 → 80 [ACK] Seq=1 Ack=1 Win=132096 Len=0 TSval=8918
TCP       66 63674 → 80 [ACK] Seq=1 Ack=1 Win=132096 Len=0 TSval=8918
TCP      669 63674 → 80 [PSH, ACK] Seq=1 Ack=1 Win=132096 Len=603 TSv
TCP     1414 63674 → 80 [ACK] Seq=604 Ack=1 Win=132096 Len=1348 TSval
TCP     1414 63674 → 80 [ACK] Seq=1952 Ack=1 Win=132096 Len=1348 TSva
TCP     1414 63674 → 80 [ACK] Seq=3300 Ack=1 Win=132096 Len=1348 TSva
TCP     1414 63674 → 80 [ACK] Seq=4648 Ack=1 Win=132096 Len=1348 TSva
TCP     1414 63674 → 80 [ACK] Seq=5996 Ack=1 Win=132096 Len=1348 TSva
TCP     1414 63674 → 80 [ACK] Seq=7344 Ack=1 Win=132096 Len=1348 TSva
TCP     1414 63674 → 80 [ACK] Seq=8692 Ack=1 Win=132096 Len=1348 TSva
```

最小可用缓存空间为132096，发送端的传输以后接收端的缓存够用。

- **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，客户端发送的序列号一直递增。

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

```
    [Coloring Rule Name: HTTP]
    [Coloring Rule String: http || tcp.port == 80 || http2]
> Ethernet II, Src: IntelCor_f5:85:c0 (dc:71:96:f5:85:c0), Dst: JuniperN_d2:ff:c2 (44:ec:ce:d2:f
> Internet Protocol Version 4, Src: 172.20.113.24, Dst: 128.119.245.12
> Transmission Control Protocol, Src Port: 63674, Dst Port: 80, Seq: 151896, Ack: 1, Len: 1029
v [115 Reassembled TCP Segments (152924 bytes) #29(603), #30(1348), #31(1348), #32(1348), #33(1
    [Frame: 29, payload: 0-602 (603 bytes)]
    [Frame: 30, payload: 603-1950 (1348 bytes)]
    [Frame: 31, payload: 1951-3298 (1348 bytes)]
    [Frame: 32, payload: 3299-4646 (1348 bytes)]
    [Frame: 33, payload: 4647-5994 (1348 bytes)]
    [Frame: 34, payload: 5995-7342 (1348 bytes)]
    [Frame: 35, payload: 7343-8690 (1348 bytes)]
```

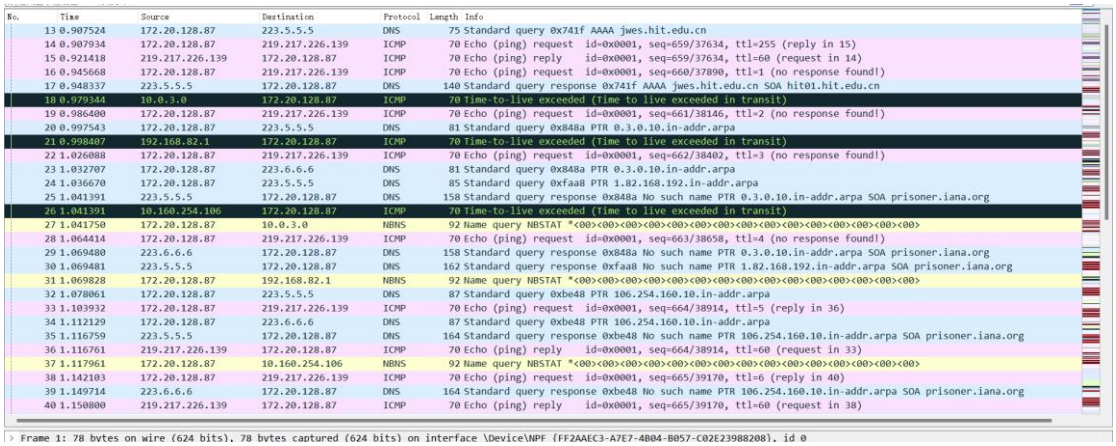
```
v [Timestamps]
    [Time since first frame in this TCP stream: 1.299152000 seconds]
    [Time since previous frame in this TCP stream: 0.000002000 seconds]
```

发送数据的大小一共为152924bytes，时间为1.229152000-0.000002000=1.229150000s，

152924bytes/1.229150000s = 0.949Mbps

4. IP分组

用PingPlotter向 http://jwes.hit.edu.cn/ 发送数据包后用 wireshark 捕获即可。



4.1 对捕获的数据包进行分析

- **你主机的IP地址是什么？**

```
ame 14: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface \Device\N
nernet II, Src: IntelCor_f5:85:c0 (dc:71:96:f5:85:c0), Dst: JuniperN_d2:ff:c2 (44:ec:ce:
ternet Protocol Version 4, Src: 172.20.128.87, Dst: 219.217.226.139
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 56
Identification: 0x1574 (5492)
Flags: 0x00
```

172.20.128.87

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

```
Time to Live: 255
Protocol: ICMP (1)
Header Checksum: 0x00
```

上层字段为ICMP，值为1。

- **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？**

```
Internet Protocol Version 4, Src: 172.20.128.87, Dst: 219.217.226.1
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
> Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 56
  Identification: 0x1574 (5492)
> Flags: 0x00
  ...0 0000 0000 0000 = Fragment Offset: 0
  Time to Live: 255
  Protocol: ICMP (1)
```

IP头有20字节。

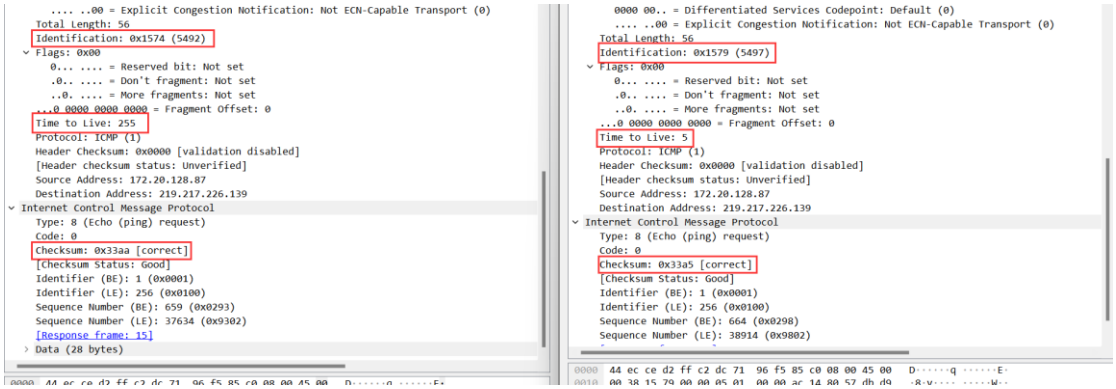
总长为56字节，净载大小为总长减头部大小，即56-20=36字节。

- **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**

```
Flags: 0x00
    0... .... = Reserved bit: Not set
    .0.. .... = Don't fragment: Not set
    ..0. .... = More fragments: Not set
...0 0000 0000 0000 = Fragment Offset: 0
Time to Live: 255
```

查看more fragments位可知其位数为0，且偏移量为0，故当前数据包并未分片。

- **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？**



Identification、TTL和Checksum总在发生变化

- **哪些字段必须保持常量？哪些字段必须改变？为什么？**

必须保持常量的是版本号、首部长度、Differentiated Services Field 以及协议（始终 为ICMP）。必须改变的是 TTL、Checksum 和 Identification，TTL 为生存时间，每次转发 必然改变；由于TTL的改变，Checksum 自然也会改变；Identification 则是用于区分不同的 ICMP 报文。

- **描述你看到的IP数据包Identification字段值的形式。**

四位16进制，每个包的Identification每次加一。

- **Identification字段和TTL字段的值是什么？**

```
Total Length: 56
Identification: 0x0000 (0)
Flags: 0x00
    0... .... = Reserved bit: Not set
    .0.. .... = Don't fragment: Not set
    ..0. .... = More fragments: Not set
...0 0000 0000 0000 = Fragment Offset: 0
Time to Live: 254
Protocol: ICMP (1)
Header Checksum: 0x7daf [validation disabled]
[Header checksum status: Unverified]
```

Identification是0，TTL字段为254.

- **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？**

不变，因为是第一跳路由器发回的报文，TTL不变；IP是无连接服务，标识不是序列号，相同的标识是为了分段后重组，给同一个主机发送的ICMP报文，TTL不变，则Identification字段不变。

- **该消息是否被分解成不止一个IP数据报？**

  被分成了两个IP数据包。

- **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？**

```
    .... ..00 = Explicit Congestion Notification: No
  Total Length: 1500
  Identification: 0x1593 (5523)
v Flags: 0x20, 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)
```

  More fragments位值为1，表明当前信息分片，且当前分片不是最后一片，该分片长度为1500。

- **原始数据包被分成了多少片？**

```
  [Header checksum status: Unverified]
  Source Address: 172.20.128.87
  Destination Address: 219.217.226.139
> [3 IPv4 Fragments (3480 bytes): #323(1480), #324(1480), #325(520)]
Internet Control Message Protocol
  Type: 8 (Echo (ping) request)
  Code: 0
```

  被分成了三片。

- **这些分片中IP数据报头部哪些字段发生了变化？**

```
v Flags: 0x20, 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: 60
  Protocol: ICMP (1)
  Header Checksum: 0x8a33 [validation disabled]
```

```
v Flags: 0x20, More fragments
    0... .... = Reserved bit: Not set
    .0.. .... = Don't fragment: Not set
    ..1. .... = More fragments: Set
  ...0 0101 1100 1000 = Fragment Offset: 1480
  Time to Live: 60
  Protocol: ICMP (1)
  Header Checksum: 0x897a [validation disabled]
```

```
Flags: 0x01
  0... .... = Reserved bit: Not set
  .0.. .... = Don't fragment: Not set
  ..0. .... = More fragments: Not set
...0 1011 1001 0000 = Fragment Offset: 2960
Time to Live: 60
Protocol: ICMP (1)
Header Checksum: 0xac81 [validation disabled]
[Header checksum status: Unverified]
```

  标志位部分、偏移量和 Checksum 部分发生了变化

5. 抓取 ARP 数据包

查看arp缓存内容：

```
C:\Users\Youngsc>arp -a

接口: 172.20.1.6 --- 0x18
  Internet 地址         物理地址              类型
  172.20.0.1            44-ec-ce-d2-ff-c2     动态
  172.20.23.75          44-ec-ce-d2-ff-c2     动态
  172.20.28.217         44-ec-ce-d2-ff-c2     动态
  172.20.29.137         44-ec-ce-d2-ff-c2     动态
  172.20.41.68          44-ec-ce-d2-ff-c2     动态
  172.20.41.102         44-ec-ce-d2-ff-c2     动态
  172.20.41.165         44-ec-ce-d2-ff-c2     动态
  172.20.51.69          44-ec-ce-d2-ff-c2     动态
  172.20.52.196         44-ec-ce-d2-ff-c2     动态
  172.20.63.206         44-ec-ce-d2-ff-c2     动态
  172.20.66.144         44-ec-ce-d2-ff-c2     动态
  172.20.83.239         44-ec-ce-d2-ff-c2     动态
  172.20.98.111         44-ec-ce-d2-ff-c2     动态
  172.20.118.162        44-ec-ce-d2-ff-c2     动态
  172.20.132.195        44-ec-ce-d2-ff-c2     动态
  172.20.204.188        44-ec-ce-d2-ff-c2     动态
  172.20.229.134        44-ec-ce-d2-ff-c2     动态
  172.20.250.97         44-ec-ce-d2-ff-c2     动态
  172.20.255.255        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
  239.255.255.250       01-00-5e-7f-ff-fa     静态
  255.255.255.255       ff-ff-ff-ff-ff-ff     静态
```

在命令行模式下输入：ping 172.17.42.105

```
C:\Users\Youngsc>ping 172.17.42.105

正在 Ping 172.17.42.105 具有 32 字节的数据:
来自 172.17.42.105 的回复: 字节=32 时间=54ms TTL=61
来自 172.17.42.105 的回复: 字节=32 时间=7ms TTL=61
来自 172.17.42.105 的回复: 字节=32 时间=10ms TTL=61
来自 172.17.42.105 的回复: 字节=32 时间=13ms TTL=61

172.17.42.105 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 = 7ms, 最长 = 54ms, 平均 = 21ms
```

启动wireshark进行捕获

No.	Time	Source	Destination	Protocol	Length	Info
51	5.450546	IntelCor_f5:85:c0	JuniperN_d2:ff:c2	ARP	42	Who has 172.20.83.239? Tell 172.20.1.6
52	5.461857	JuniperN_d2:ff:c2	IntelCor_f5:85:c0	ARP	56	172.20.83.239 is at 44:ec:ce:d2:ff:c2
56	5.954663	IntelCor_f5:85:c0	JuniperN_d2:ff:c2	ARP	42	Who has 172.20.0.1? Tell 172.20.1.6
57	6.071490	JuniperN_d2:ff:c2	IntelCor_f5:85:c0	ARP	60	172.20.0.1 is at 44:ec:ce:d2:ff:c2
148	23.455028	IntelCor_f5:85:c0	JuniperN_d2:ff:c2	ARP	42	Who has 172.20.51.69? Tell 172.20.1.6
149	23.471000	JuniperN_d2:ff:c2	IntelCor_f5:85:c0	ARP	56	172.20.51.69 is at 44:ec:ce:d2:ff:c2
191	36.463616	IntelCor_f5:85:c0	JuniperN_d2:ff:c2	ARP	42	Who has 172.20.132.195? Tell 172.20.1.6
192	36.478412	JuniperN_d2:ff:c2	IntelCor_f5:85:c0	ARP	56	172.20.132.195 is at 44:ec:ce:d2:ff:c2
211	41.950501	IntelCor_f5:85:c0	JuniperN_d2:ff:c2	ARP	42	Who has 172.20.83.239? Tell 172.20.1.6
212	42.050744	JuniperN_d2:ff:c2	IntelCor_f5:85:c0	ARP	56	172.20.83.239 is at 44:ec:ce:d2:ff:c2
283	46.959327	IntelCor_f5:85:c0	JuniperN_d2:ff:c2	ARP	42	Who has 172.20.0.1? Tell 172.20.1.6
285	47.037909	JuniperN_d2:ff:c2	IntelCor_f5:85:c0	ARP	60	172.20.0.1 is at 44:ec:ce:d2:ff:c2
445	63.456519	IntelCor_f5:85:c0	JuniperN_d2:ff:c2	ARP	42	Who has 172.20.229.134? Tell 172.20.1.6
446	63.470034	JuniperN_d2:ff:c2	IntelCor_f5:85:c0	ARP	56	172.20.229.134 is at 44:ec:ce:d2:ff:c2
455	69.453950	IntelCor_f5:85:c0	JuniperN_d2:ff:c2	ARP	42	Who has 172.20.51.69? Tell 172.20.1.6
456	69.469557	JuniperN_d2:ff:c2	IntelCor_f5:85:c0	ARP	56	172.20.51.69 is at 44:ec:ce:d2:ff:c2

5.1

ARP表的格式如下。在ARP表中，每一项表示一个IP地址到物理地址的映射。每一项第一列是IP地址，第二列是物理地址，第三列是类型。

```
C:\Users\Youngsc>arp -a

接口: 172.20.1.6 --- 0x18
  Internet 地址         物理地址              类型
  172.20.0.1            44-ec-ce-d2-ff-c2     动态
  172.20.23.75          44-ec-ce-d2-ff-c2     动态
  172.20.28.217         44-ec-ce-d2-ff-c2     动态
  172.20.29.137         44-ec-ce-d2-ff-c2     动态
  172.20.41.68          44-ec-ce-d2-ff-c2     动态
  172.20.41.102         44-ec-ce-d2-ff-c2     动态
  172.20.41.165         44-ec-ce-d2-ff-c2     动态
  172.20.51.69          44-ec-ce-d2-ff-c2     动态
  172.20.52.196         44-ec-ce-d2-ff-c2     动态
  172.20.63.206         44-ec-ce-d2-ff-c2     动态
  172.20.66.144         44-ec-ce-d2-ff-c2     动态
  172.20.83.239         44-ec-ce-d2-ff-c2     动态
  172.20.98.111         44-ec-ce-d2-ff-c2     动态
  172.20.118.162        44-ec-ce-d2-ff-c2     动态
  172.20.132.195        44-ec-ce-d2-ff-c2     动态
  172.20.204.188        44-ec-ce-d2-ff-c2     动态
  172.20.229.134        44-ec-ce-d2-ff-c2     动态
  172.20.250.97         44-ec-ce-d2-ff-c2     动态
  172.20.255.255        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
  239.255.255.250       01-00-5e-7f-ff-fa     静态
  255.255.255.255       ff-ff-ff-ff-ff-ff     静态
```

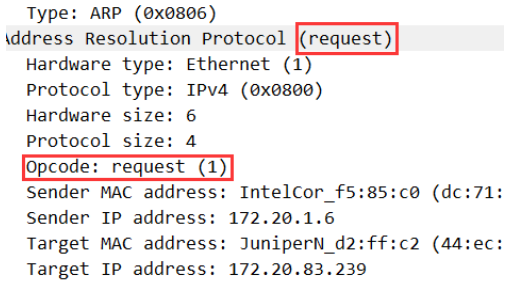
5.2

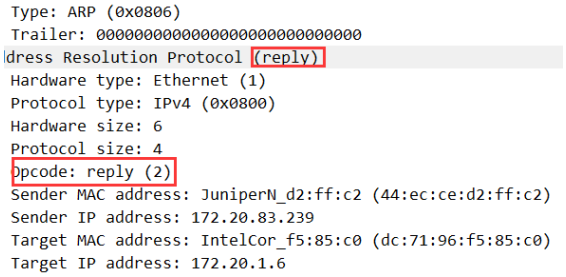
- **ARP数据包的格式是怎样的？由几部分构成，各个部分所占的字节数是多少？**

格式如上图所示，共由九部分组成。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）

- **如何判断一个ARP数据是请求包还是应答包？**

  可以通过 Opcode 字段判断，若为 1 则是请求包；若为 2 则是应答包。





- **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的局域网地址的帧中传送？**

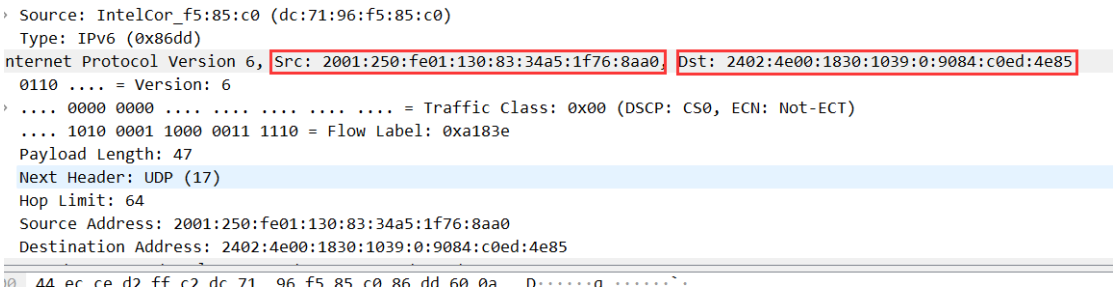
因为进行 ARP 查询时并不知道目的 IP 地址对应的 MAC 地址，所以需要广播查询；而 ARP 响应报文知道查询主机的 MAC 地址（通过查询主机发出的查询报文获得），且局域网中的其他主机不需要此次查询的结果，因此 ARP 响应要在一个有着明确目的的局域网地址的帧中传送。

6. 抓取UDP数据包

- **消息是基于UDP的还是TCP的？**

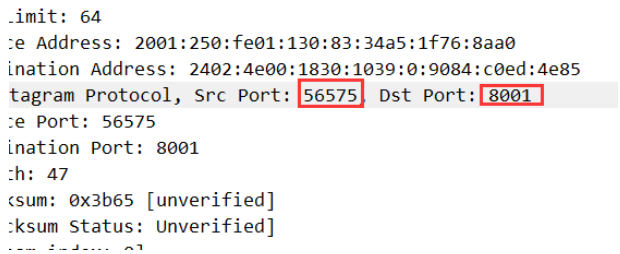
  UDP

- **你的主机ip地址是什么？目的主机ip地址是什么？**



我的主机IP为2001:250:fe01:130:83:34a5:1f76:8aa0，目的主机IP为2402:4e00:1830:1039:0:9084:c0ed:4e85

- **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？**



主机的QQ消息端口号为56575，服务器的端口号为8001

- **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？**

```
Source Port: 56575
Destination Port: 8001
Length: 47
Checksum: 0x3b65 [unverified]
[Checksum Status: Unverified]
[Stream index: 0]
> [Timestamps]
UDP payload (39 bytes)
```

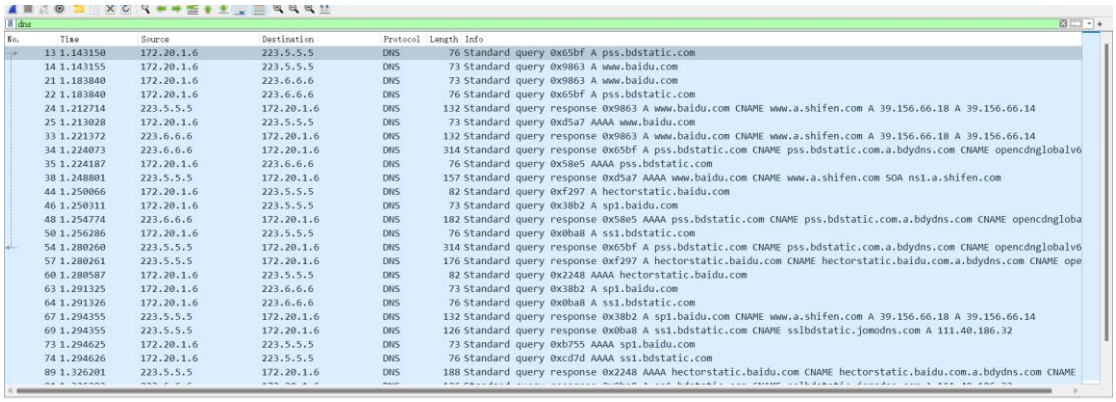
UDP 数据报由五部分构成，分别是源端口号（4 字节），目的端口号（4 字节），长度（4 字节），校验和（4 字节）和应用层数据。

- **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 UDP 是不可靠的数据传输，需要上层协议来实现可靠数据传输，因此每次发送 ICQ 报文后又回复一个 ICQ 数据包来确认。UDP 是无连接的，因为可以看到发送数据之前没有连接的建立过程（如 TCP 的三次握手），没有序列号，因此为无连接数据传输。

7. 利用 WireShark 进行 DNS 协议分析

利用 Wireshark 进行 DNS 协议抓包的结果如下。



问题讨论：

问题1：用Edge、谷歌浏览器多次访问各种网址，但报文中始终没有出现If-Modified-Since字段，最终将浏览器更换为IE之后才成功。

问题2：实验指导书中给出的网址，访问后使用wireshark分析后发现ipv6地址，造成难以按照实验步骤进行分析。解决方案：更改访问网站为IPv4网站（http://jwes.hit.edu.cn/或http://today.hit.edu.cn/等）

心得体会：

1、 本次实验学会了如何使用Wireshark进行抓包。

2、 本次实验对于各个协议之间进行报文格式和报文交换有了进一步的了解。