哈尔滨工业大学

<<计算机网络>>

实验报告

(2017 年度春季学期)

姓名:	张茗帅
学号:	**1140310606**
学院:	计算机科学与技术学院
教师:	聂兰顺

实验四 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

实验环境：

- 我的个人 PC 机
- Windows10 操作系统
- Wireshark

二、实验内容

必做内容：

(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：

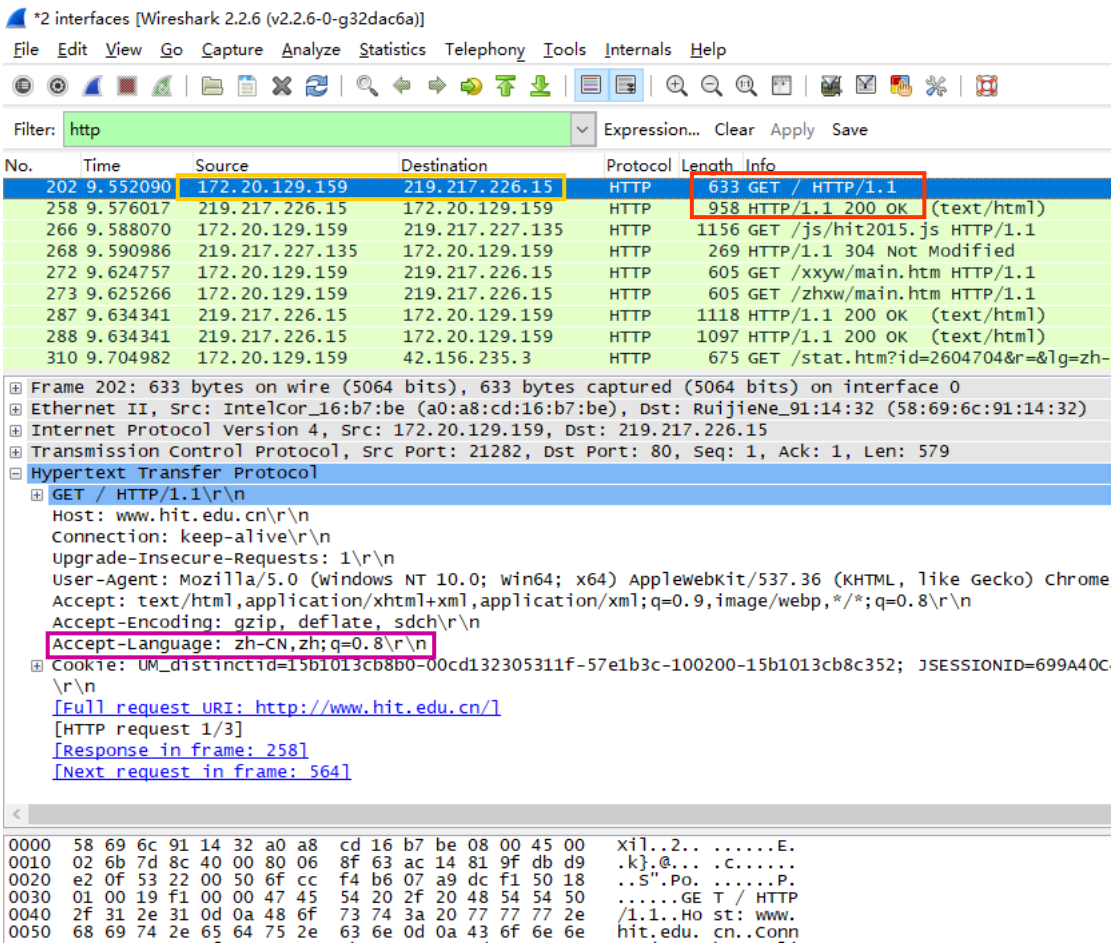
(a) 利用 Wireshark 分析 DNS 协议
(b) 利用 Wireshark 分析 UDP 协议
(c) 利用 Wireshark 分析 ARP 协议

三、实验过程及结果

1 HTTP 分析

（1） HTTP GET/response 交互

打开 Wireshark 进行 capture 配置，准备进行分组俘获，然后打开浏览器，输入地址http://www.hit.edu.cn/（指导书中给的地址已经不存在了），页面加载完毕后，停止分组俘获，得到的俘获窗口内容截图如下：



回答问题：

a) **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？**

从上面的红框内可以看到，我的浏览器和服务器所运行的 HTTP 协议的版本号都是 HTTP 1.1

b) **你的浏览器向服务器指出它能接收何种语言版本的对象？**

在紫框中可以看到，他接受的语言为 zh-CN，即简体中文（中国）

c) **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？**

在橙颜色的框中可以看到源 IP 地址和目的 IP 地址

我计算机的 IP 地址为：: 172.20.129.159

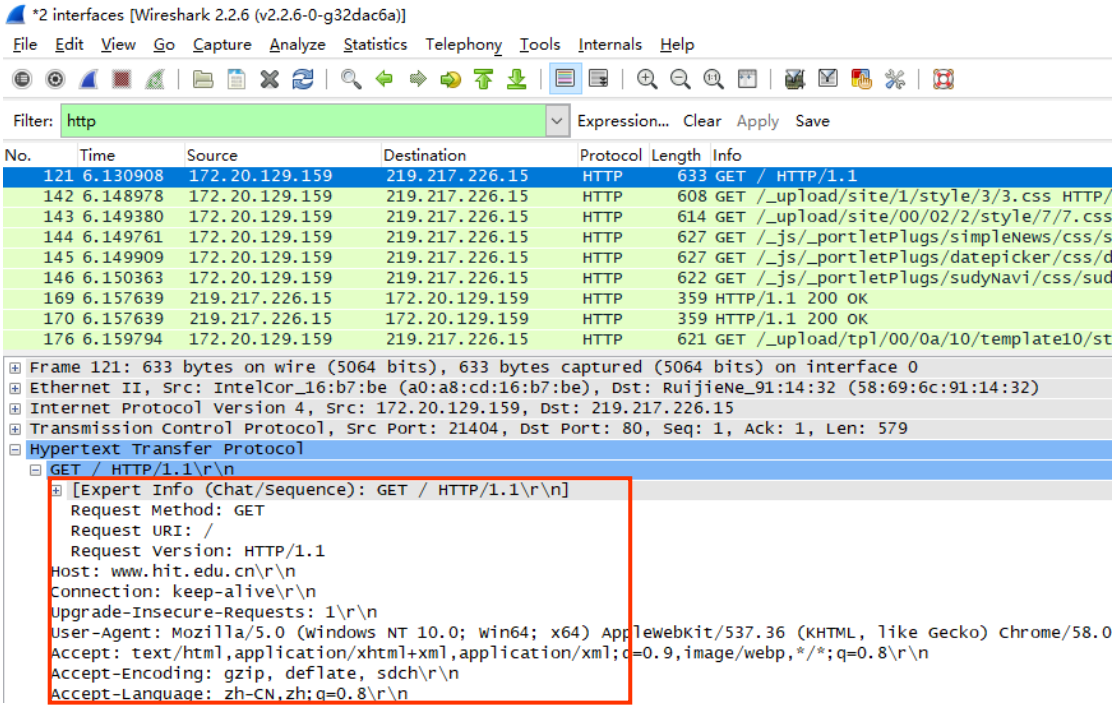
服务器 IP：219.217.226.15

d) **从服务器向你的浏览器返回的状态代码是多少？**

200

（2） HTTP 条件 GET/response 交互

截图如下：



回答问题：

a) **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？**

没有 （在红框内没有发现）

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？**

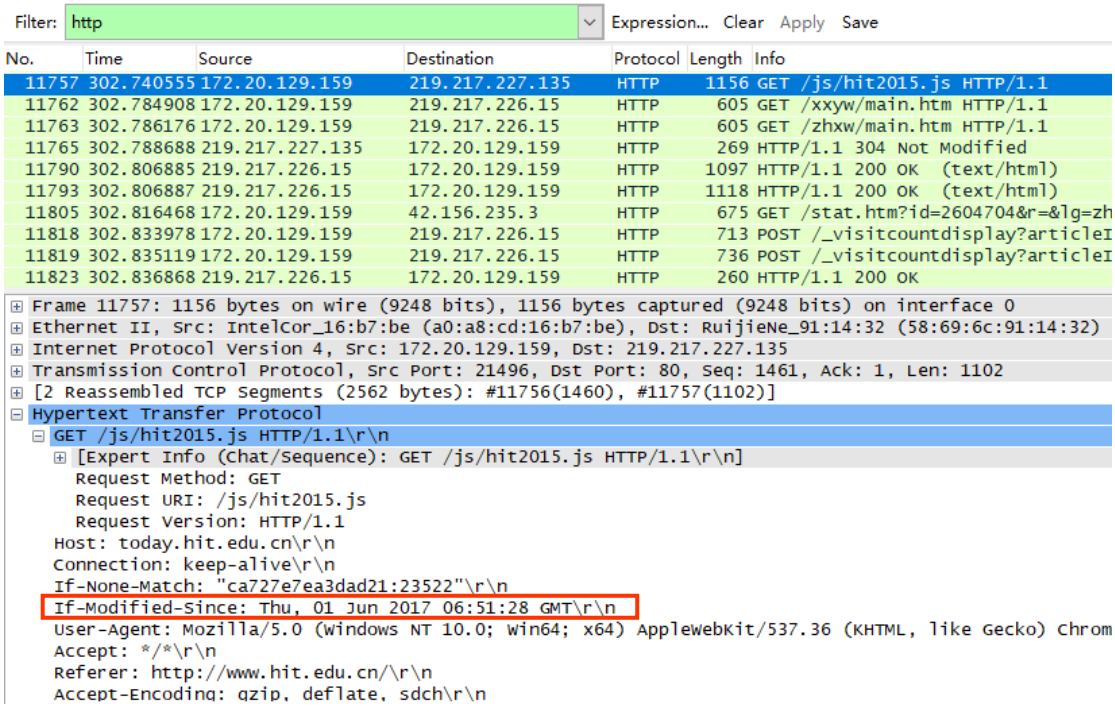
服务器明确返回了内容

HTTP Status Code（状态代码） 为 304 时不明确返回文件

HTTP Status Code（状态代码）为 200 时明确返回文件

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？**

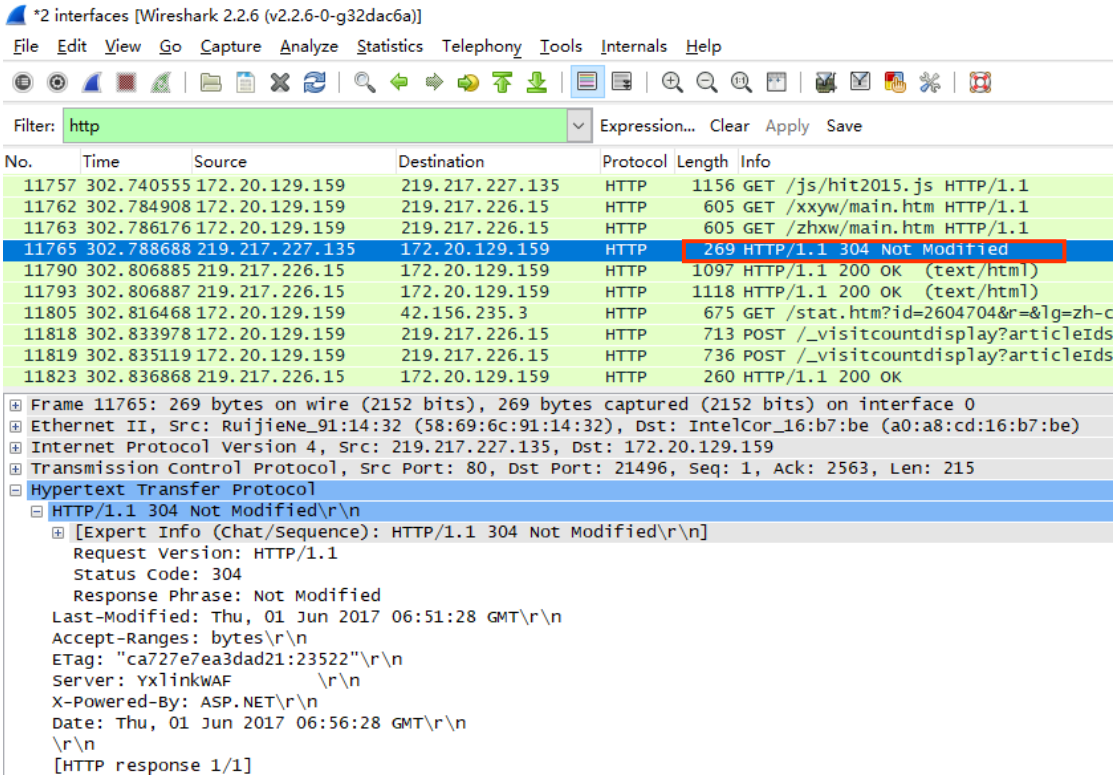
截图如下：



如图在红框内，发现了这个字段。

这个字段后面代表的是时间，即咨询服务器在这个时候之后是否有更新

d) 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。

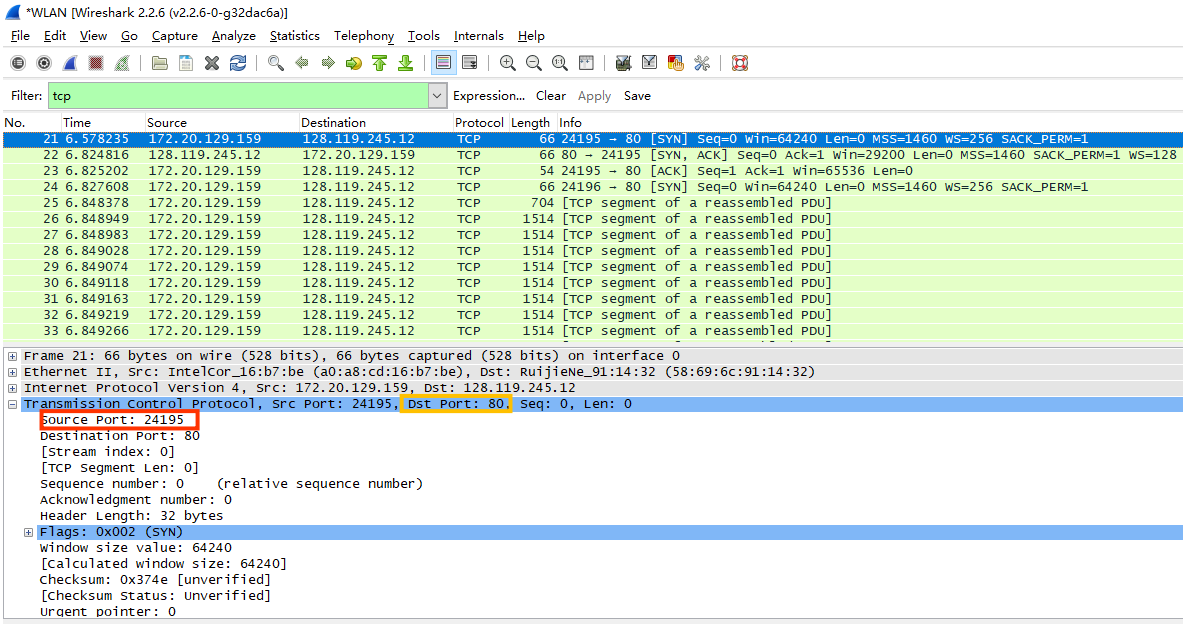


请求响应中的 HTTP 状态代码为 304。（见红色框内）

不会明确返回文件，因为根据之前 HTTP 的 GET 请求中 IF-MODIFIED-SINCE 字段内的时间服务器判断结果为 Not Modified，于是客户端可以使用本地这个没有过期的缓存文件。

2 TCP 分析

获取的 TCP 报文整体情况截图如下



回答问题：

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地址：172.20.129.159

端口号：24195（见红色框内）

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 地址：128.119.245.12

端口号：80 （见橙色框）

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？**

```
[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
  [TCP Flags: ·····S·]
```

如图，初始化TCP连接的TCP SYN报文段的序号是0；

通过Flags标志位，将其中的SYN位置为1，表示该报文段是SYN报文段

4) **服务器向客户端发送的SYNACK报文段序号是多少？该报文段中，Acknowledgement字段的值是多少？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？**

```
[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
  [TCP Flags: ·····A··S·]
Window size value: 29200
```

如上图，服务器端向客户端发送的报文段序号为0；

服务器发的acknowledgment number字段是根据上一次客户端发给服务器的seq+1得到的；

通过Flags标志位中的SYN位和ACK位都是1来确定该报文段是一个SYN ACK报文段的。

5） 你能从捕获的数据包中分析出tcp三次握手过程吗？

```
TCP        66 24195 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
TCP        66 80 → 24195 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=128
TCP        54 24195 → 80 [ACK] Seq=1 Ack=1 Win=65536 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？

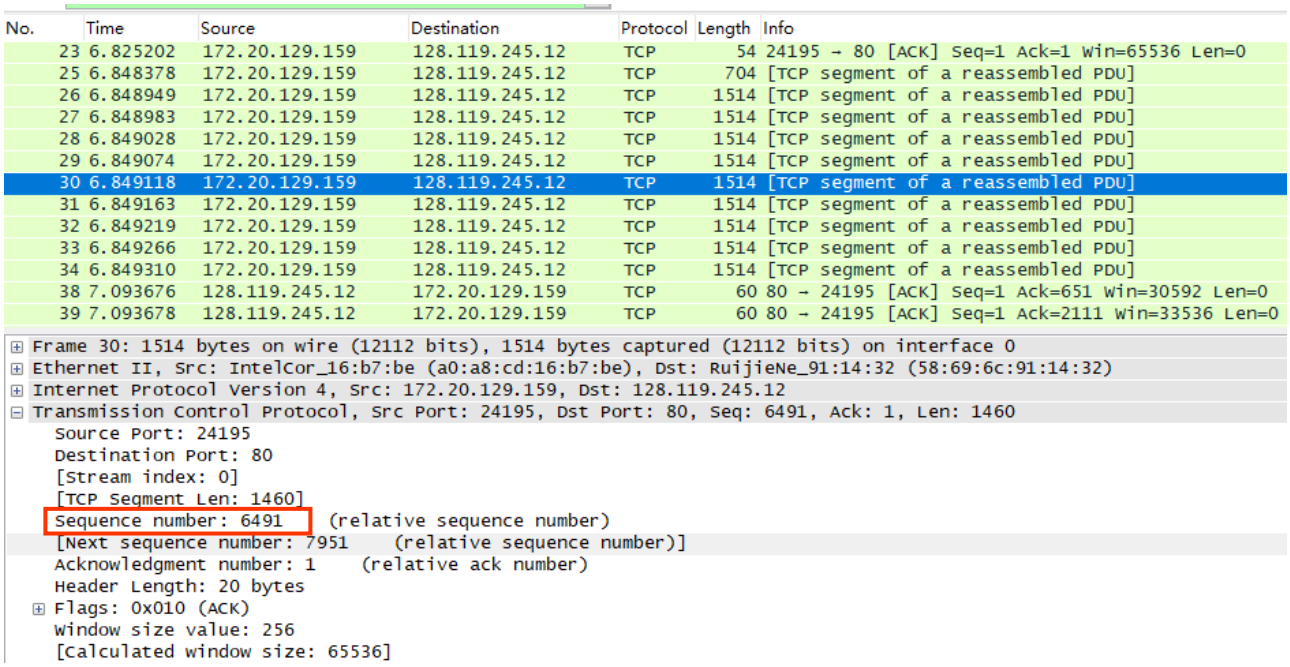
```
No.   Time      Source          Destination     Protocol Length Info
  187 7.602480  172.20.129.159  128.119.245.12  TCP      1514 [TCP segment of a reassembled PDU]
  188 7.602505  172.20.129.159  128.119.245.12  TCP      1514 [TCP segment of a reassembled PDU]
  189 7.602534  172.20.129.159  128.119.245.12  HTTP      539 POST /wireshark-labs/lab3-1-reply.htm HTTP/1.1  (text/plain)
  190 7.681965  SamsungE_2e:3a:13 Broadcast      ARP        56 Who has 172.20.128.1? Tell 172.20.153.128
  191 7.852503  128.119.245.12  172.20.129.159  TCP        60 80 → 24195 [ACK] Seq=1 Ack=98471 Win=183296 Len=0
  192 7.852505  128.119.245.12  172.20.129.159  TCP        60 80 → 24195 [ACK] Seq=1 Ack=99931 Win=186112 Len=0
  193 7.852506  128.119.245.12  172.20.129.159  TCP        60 80 → 24195 [ACK] Seq=1 Ack=101391 Win=189056 Len=0
  194 7.852507  128.119.245.12  172.20.129.159  TCP        60 80 → 24195 [ACK] Seq=1 Ack=102851 Win=192000 Len=0
  195 7.852507  128.119.245.12  172.20.129.159  TCP        60 80 → 24195 [ACK] Seq=1 Ack=104311 Win=194944 Len=0
  196 7.852508  128.119.245.12  172.20.129.159  TCP        60 80 → 24195 [ACK] Seq=1 Ack=105771 Win=197888 Len=0
  197 7.852508  128.119.245.12  172.20.129.159  TCP        60 80 → 24195 [ACK] Seq=1 Ack=107231 Win=200704 Len=0
  198 7.852509  128.119.245.12  172.20.129.159  TCP        60 80 → 24195 [ACK] Seq=1 Ack=108691 Win=203648 Len=0
  199 7.852510  128.119.245.12  172.20.129.159  TCP        60 80 → 24195 [ACK] Seq=1 Ack=110151 Win=206592 Len=0
⊞ Frame 189: 539 bytes on wire (4312 bits), 539 bytes captured (4312 bits) on interface 0
⊞ Ethernet II, Src: IntelCor_16:b7:be (a0:a8:cd:16:b7:be), Dst: RuijieNe_91:14:32 (58:69:6c:91:14:32)
⊞ Internet Protocol Version 4, Src: 172.20.129.159, Dst: 128.119.245.12
⊟ Transmission Control Protocol, Src Port: 24195, Dst Port: 80, Seq: 152487, Ack: 1, Len: 485
    Source Port: 24195
    Destination Port: 80
    [Stream index: 0]
    [TCP Segment Len: 485]
    Sequence number: 152487    (relative sequence number)
    [Next sequence number: 152972    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    Header Length: 20 bytes
  ⊞ Flags: 0x018 (PSH, ACK)
    Window size value: 256
    [Calculated window size: 65536]
```

如图，序号为152487（在红色的框内）

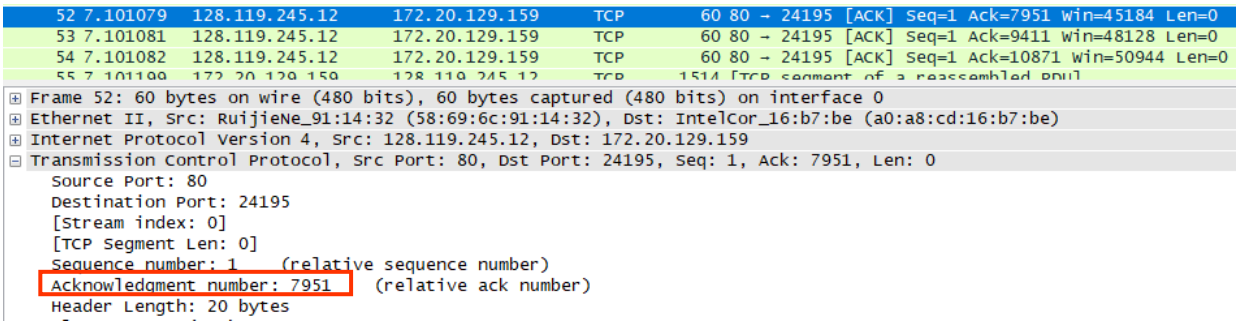
7） 如果将包含HTTP POST命令的TCP报文段看作是TCP连接上的第一个报文段，那么该TCP连接上的第六个报文段的序号是多少？是何时发送的？该报文段所对应的ACK是何时接收的？

```
  189 7.602534  172.20.129.159  128.119.245.12  HTTP      539 POST /wireshark-labs/lab3-1-reply.htm HTTP/1.1  (text/plain)
  191 7.852503  128.119.245.12  172.20.129.159  TCP        60 80 → 24195 [ACK] Seq=1 Ack=98471 Win=183296 Len=0
  192 7.852505  128.119.245.12  172.20.129.159  TCP        60 80 → 24195 [ACK] Seq=1 Ack=99931 Win=186112 Len=0
  193 7.852506  128.119.245.12  172.20.129.159  TCP        60 80 → 24195 [ACK] Seq=1 Ack=101391 Win=189056 Len=0
  194 7.852507  128.119.245.12  172.20.129.159  TCP        60 80 → 24195 [ACK] Seq=1 Ack=102851 Win=192000 Len=0
  195 7.852507  128.119.245.12  172.20.129.159  TCP        60 80 → 24195 [ACK] Seq=1 Ack=104311 Win=194944 Len=0
  196 7.852508  128.119.245.12  172.20.129.159  TCP        60 80 → 24195 [ACK] Seq=1 Ack=105771 Win=197888 Len=0
  197 7.852508  128.119.245.12  172.20.129.159  TCP        60 80 → 24195 [ACK] Seq=1 Ack=107231 Win=200704 Len=0
⊞ Frame 189: 539 bytes on wire (4312 bits), 539 bytes captured (4312 bits) on interface 0
⊞ Ethernet II, Src: IntelCor_16:b7:be (a0:a8:cd:16:b7:be), Dst: RuijieNe_91:14:32 (58:69:6c:91:14:32)
⊞ Internet Protocol Version 4, Src: 172.20.129.159, Dst: 128.119.245.12
⊞ Transmission Control Protocol, Src Port: 24195, Dst Port: 80, Seq: 152487, Ack: 1, Len: 485
⊟ [108 Reassembled TCP Segments (152971 bytes): #25(650), #26(1460), #27(1460), #28(1460), #29(1460), #30(1460), #31(1460), #32(1460)
    [Frame: 25, payload: 0-649 (650 bytes)]
    [Frame: 26, payload: 650-2109 (1460 bytes)]
    [Frame: 27, payload: 2110-3569 (1460 bytes)]
    [Frame: 28, payload: 3570-5029 (1460 bytes)]
    [Frame: 29, payload: 5030-6489 (1460 bytes)]
    [Frame: 30, payload: 6490-7949 (1460 bytes)]
    [Frame: 31, payload: 7950-9409 (1460 bytes)]
    [Frame: 32, payload: 9410-10869 (1460 bytes)]
    [Frame: 33, payload: 10870-12329 (1460 bytes)]
```

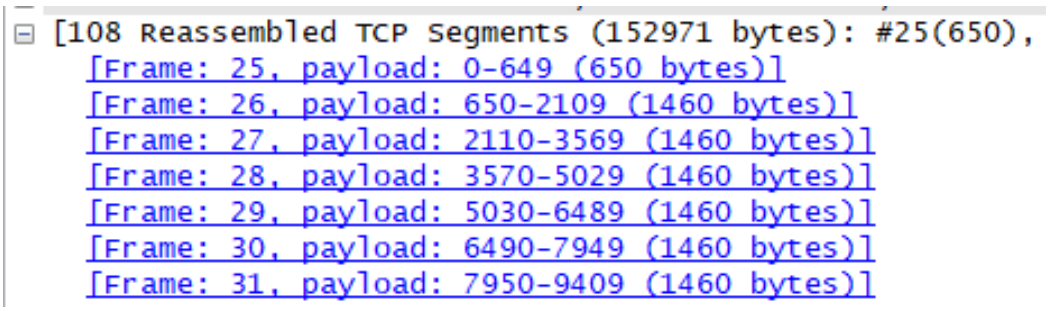
由上图可知，该HTTP协议被108个TCP报文段所封装，双击下面蓝色的第六个（红框中）。得到下图：



即这个TCP段就是我们要找的第6个TCP段，序号为6491，发送时间为在http post发送之前，tcp连接建立之后发送的（发送了前五个TCP段后），该报文所对应的ACK则我们需要找到6491+1460=7951，即我们要找到Acknowledgment number=7951的TCP段（由服务器发回的），仔细寻找后，成功找到如下图：

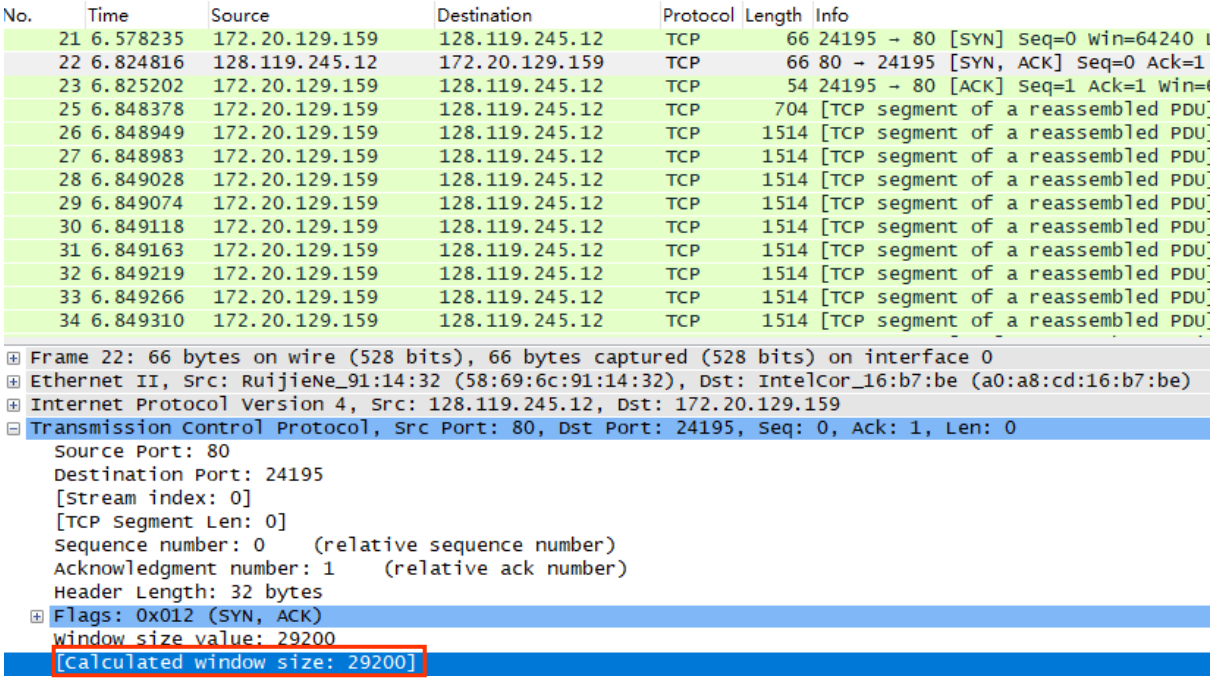


8） **前六个TCP报文段的长度各是多少？**



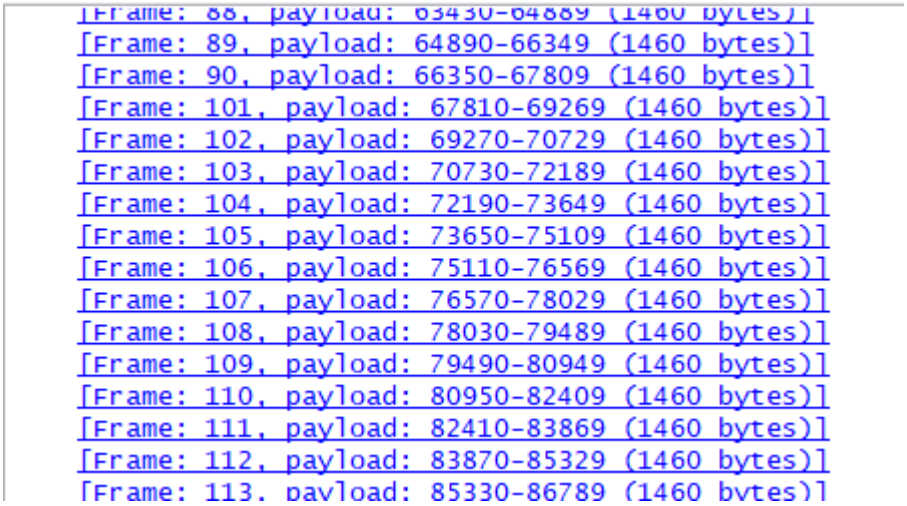
如图，前六个TCP报文段的长度为650bytes、1460bytes、1460bytes、1460bytes、1460bytes、1460bytes。（不包含TCP头部字段的20字节）

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



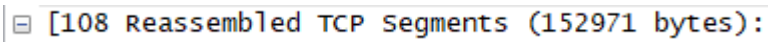
如图，接收端公示的最小的可用缓存空间是29200，由于该窗口大小会一直增加，所以不会出现接收端的缓存是否仍然不够用的情况，而且并没有限制发送端的传输。

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？



没有出现重传的报文段，判断的依据是客户端没有发送序列号相同的报文段，或者也可以在上面的绿色区域观察，一是通过序列号有没有重复的，二是如果重发会被标记为红色，并有提示文字“Resend TCP Segment”，大概是长这个样子，直接某次抓包遇到过重传的，不过这次我抓取的这些包都没有重复的。

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



首先总共发出的有效字节数（不包含头部字段）为152971 bytes 发出这些TCP

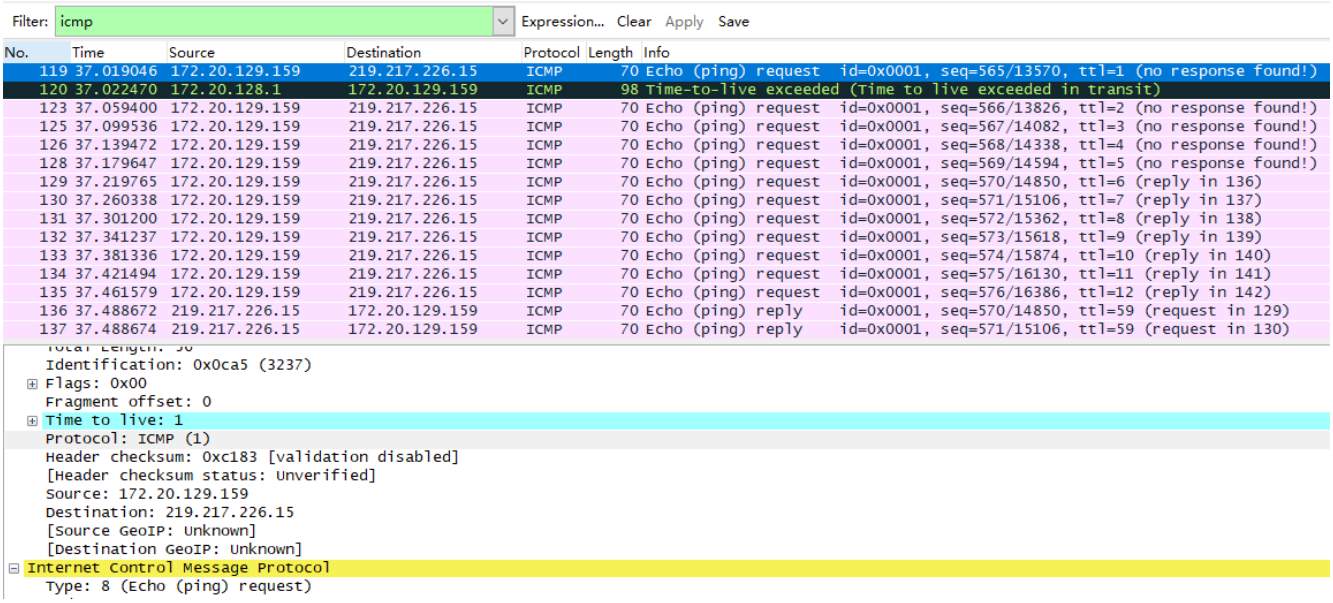
报文段所经历的总时间为（发送第一个TCP报文段的时间到最后一个TCP报文段被ACK确认的时间）7.856015-6.848378=1.007637 s，因此按照下面这个公式进行计算

TCP的**最大吞吐率** = 一个RTT传输的有效数据 / RTT

得到 throughput = 152971 b/1.007637 s = 151811.615 bps

3 IP 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，得到金额图如下：



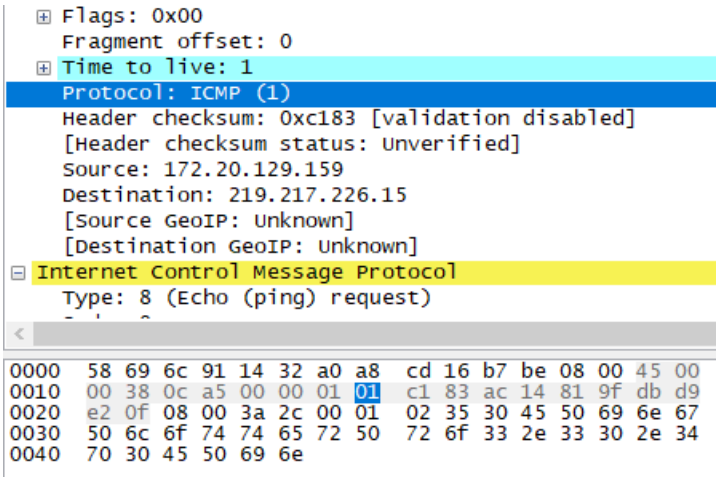
回答问题：

1） 你主机的IP地址是什么？

172.20.129.159

2） 在IP数据包头中，上层协议（upper layer）字段的值是什么？

如下图：上层协议字段的值为01，代表为ICMP报文



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？

```
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
⊞ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 56
```

如上图，头部为20字节，IP报文总长度为56字节，所以该IP数据包的净载为56-20=36字节。

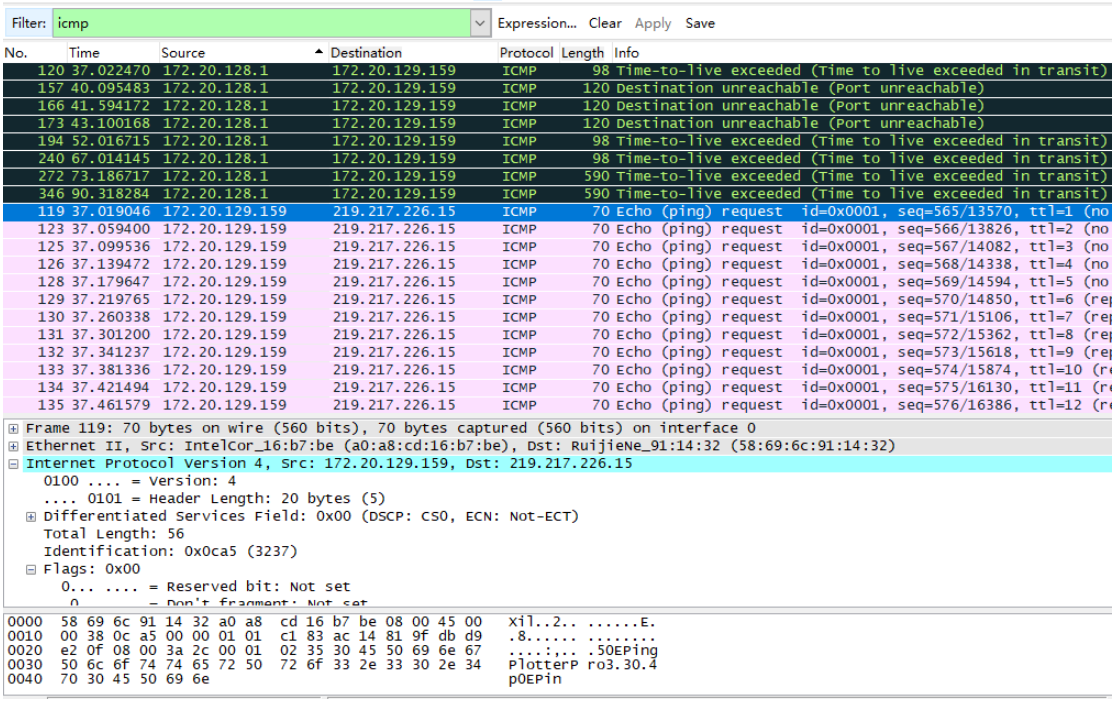
4） 该IP数据包分片了吗？解释你是如何确定该P数据包是否进行了分片

```
⊟ Flags: 0x00
    0... .... = Reserved bit: Not set
    .0.. .... = Don't fragment: Not set
    ..0. .... = More fragments: Not set
  Fragment offset: 0
```

如上图，可以知道该 IP 数据包未进行分片。

首先看到 DF=0,则证明是允许分片的,那到底分片了没有,就接着看 MF 和 Fragment offset，MF 标志位是 0，代表这或者未分片，或者这是众多分片的最后一片，再看段的偏移 Fragment offset，由于其为 0，可以知道这是该 IP 数据包未分片。

（2）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。得到截图如下：



回答问题：

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？

我通过查看多个 ICMP 消息，发现 ID（标识符字段）、TTL、Header checksum 这三个字段总在变化。

2) **哪些字段必须保持常量？哪些字段必须改变？为什么？**

必须改变：

ID 鉴别码，用于区分不同的数据包；

TTL 来自于 traceroute 的要求，用来测试路径上的路由信息；

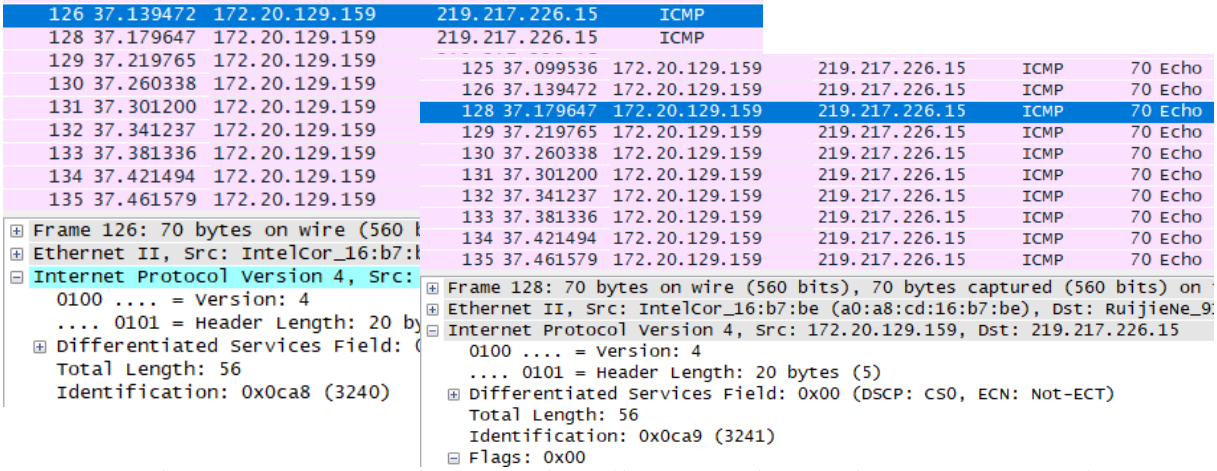
Header Checksum 首部校验和，前面的字段改变，该值也必须跟着改变；

必须保持常量：

除以上(ID,TTL,Header Checksum)外的字段保持常量。

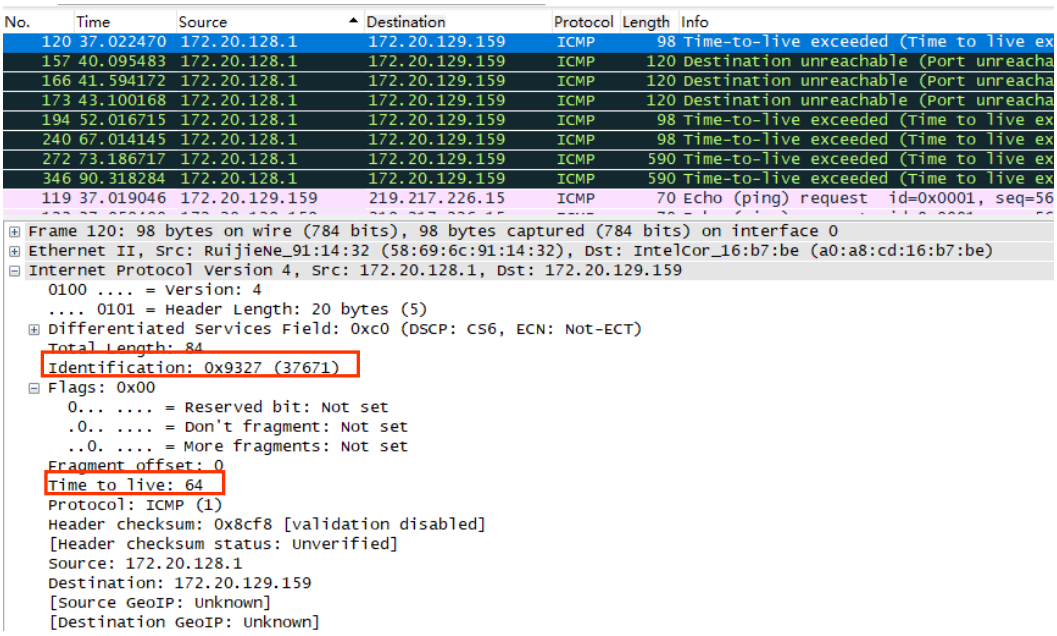
3) **描述你看到的IP数据包Identification字段值的形式。**

标识符是 16 位的，在某一范围内是+1 递增的，如下两幅图：



通过这两幅图可以看出，相邻的两个 IP 数据包第一个的 ID 为 0x0ca8，而第二个 IP 数据包的 ID 为 0xca9。

(3) 找到由最近的路由器（第一跳）返回给你主机的 ICMP Time-to-live exceeded 消息。得到的截图如下



图中标蓝的那一行便是我们要找的 ICMP 报文

回答问题：

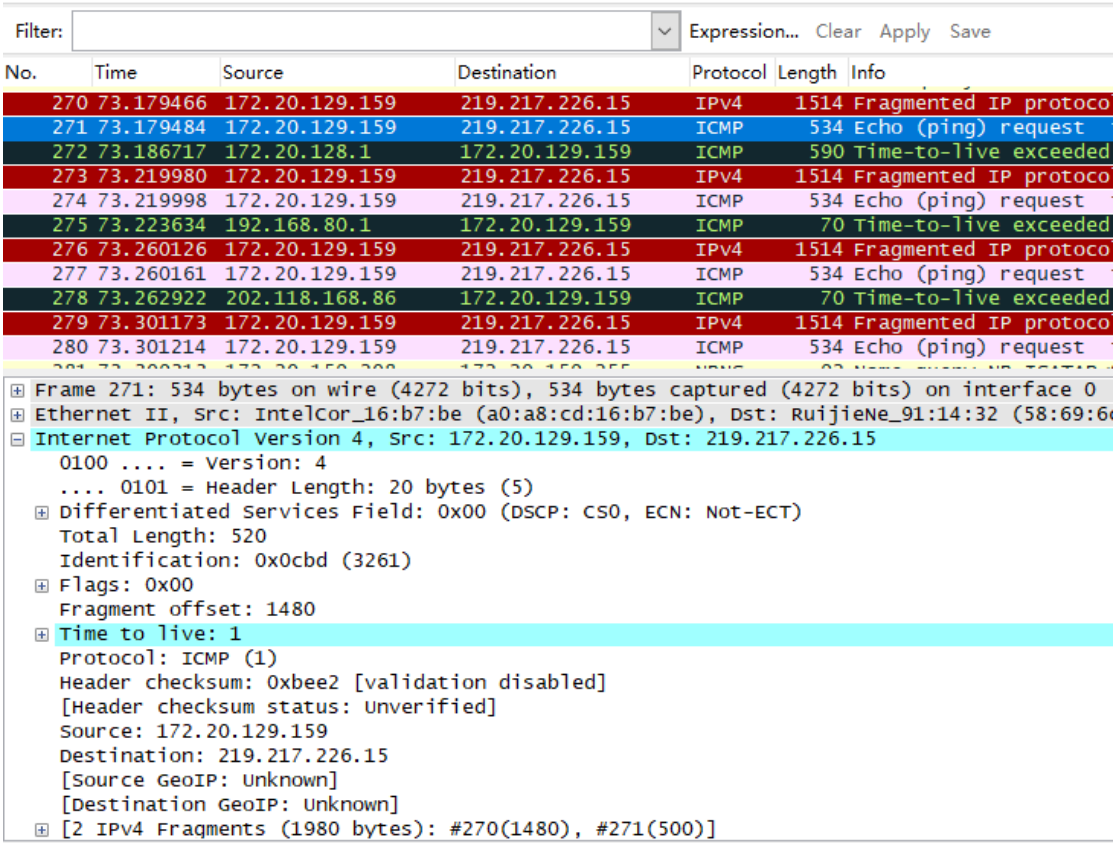
1) **Identification字段和TTL字段的值是什么？**

如图ID为0x9327，TTL为64（见图中红框所圈中的）

2） **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？**

都保持不变， IP 是无连接服务，相同的标识是为了分段后组装成同一段，给同一个主机返回的 ICMP，标识不代表序号，因此 Identification 不变，TTL 消息是相同的，都是 64。

(4) 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。 得到下图



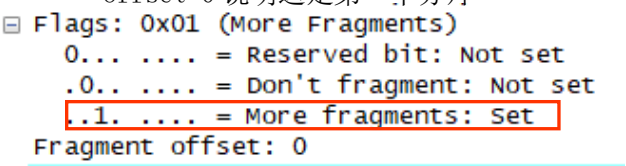
回答问题：

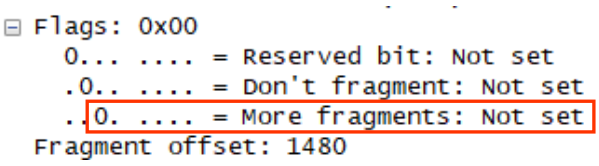
1） **该消息是否被分解成不止一个IP数据报？**

是的，该消息被分解成了2片

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

观察下面左面的图，先看第一个分片，MF 为 1 代表后面还有更多分片，同时 Fragment offset=0 说明这是第一个分片



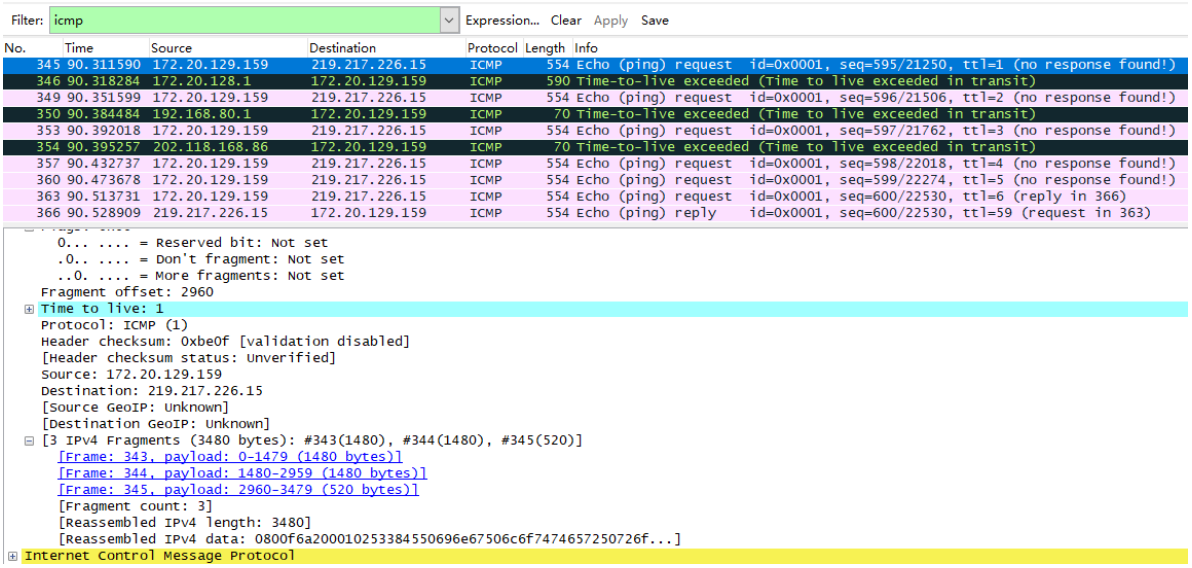


再看第二个分片的 ICMP 报文，如上面右面的图，可以看到此时 MF=0，则证明这是最后一个分片了，即共有两个 IP 分片

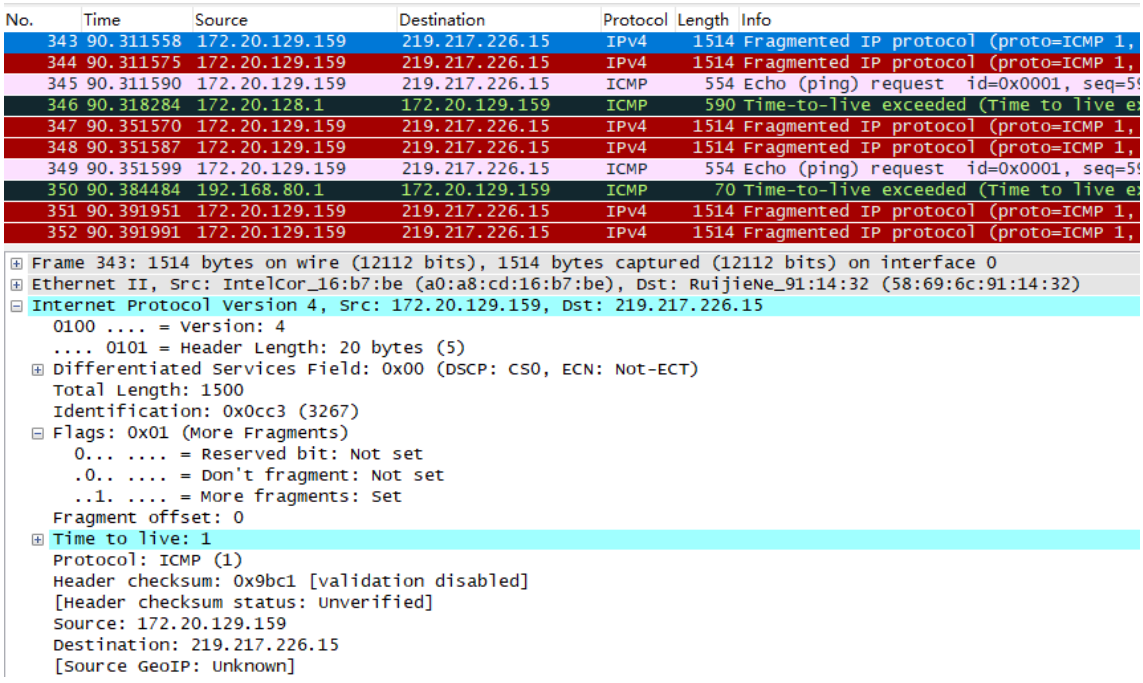
```
[2 IPv4 Fragments (1980 bytes): #270(1480), #271(500)]
  [Frame: 270, payload: 0-1479 (1480 bytes)]
  [Frame: 271, payload: 1480-1979 (500 bytes)]
  [Fragment count: 2]
```

通过上面的图就可以知道每个分片的长度分别为 1480 bytes 和 500 bytes（当然通过之前的每一个分片的偏移量也可以得到该结果）

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。 得到截图如下：



如上，这里面直接显示了包大小为3500字节拆分后的最后一个分片，前两个分片见下图：



如上图其中，No。一列下343，344，345组成了这三个分片

回答问题：

1） **原始数据包被分成了多少片？**

```
[3 IPv4 Fragments (3480 bytes): #343(1480), #344(1480), #345(520)]
  [Frame: 343, payload: 0-1479 (1480 bytes)]
  [Frame: 344, payload: 1480-2959 (1480 bytes)]
  [Frame: 345, payload: 2960-3479 (520 bytes)]
  [Fragment count: 3]
```

如图可知被分成了三片

2） **这些分片中IP数据报头部哪些字段发生了变化？**

总长度字段、MF 标志位、偏移量（Fragment offset）、校验和 这四个字段发生了变化。

第一片与第二片的总长度字段均为 1500 bytes（因为包含 20 字节的头部），第三片的总长度字段为 540 bytes；

第一片与第二片的 MF 标志位均为 1，第三片的 MF 标志位为 0

第一片的偏移量为 0，第二片的偏移量为 1480，第三片的偏移量为 2960；

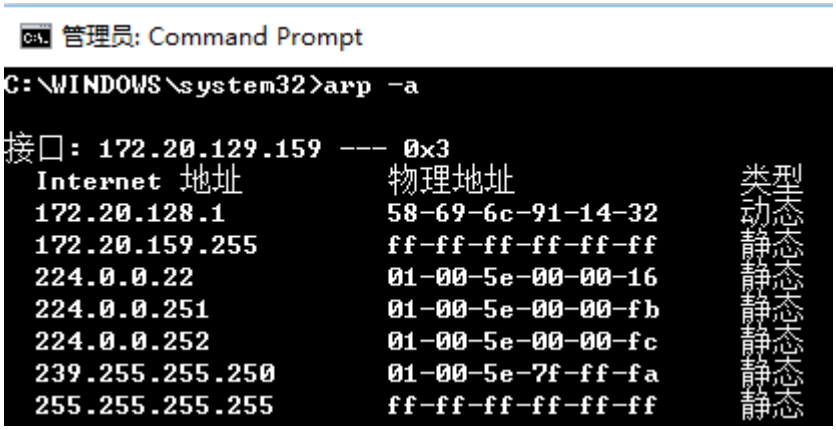
由于每一个分片的报文不是完全相同，则每一个分片的校验和字段一定也不相同。

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？

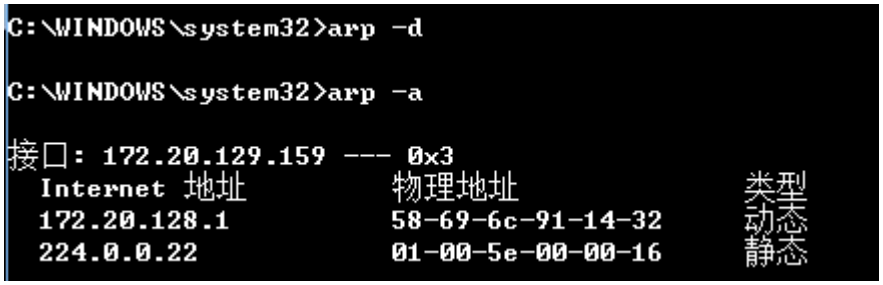
输入apr -a查看主机上ARP缓存的内容，结果如下图所示

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：

通过 arp –d 命令来清楚本机的ARP的缓存内容，如下图

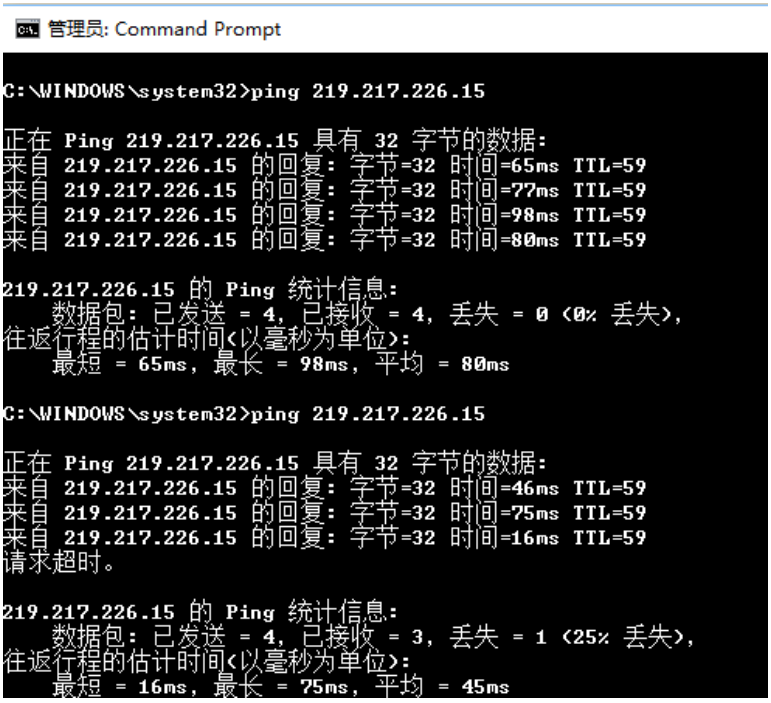


1） **ARP数据包的格式是怎样的？由几部分构成，各个部分所占的字节数是多少？**

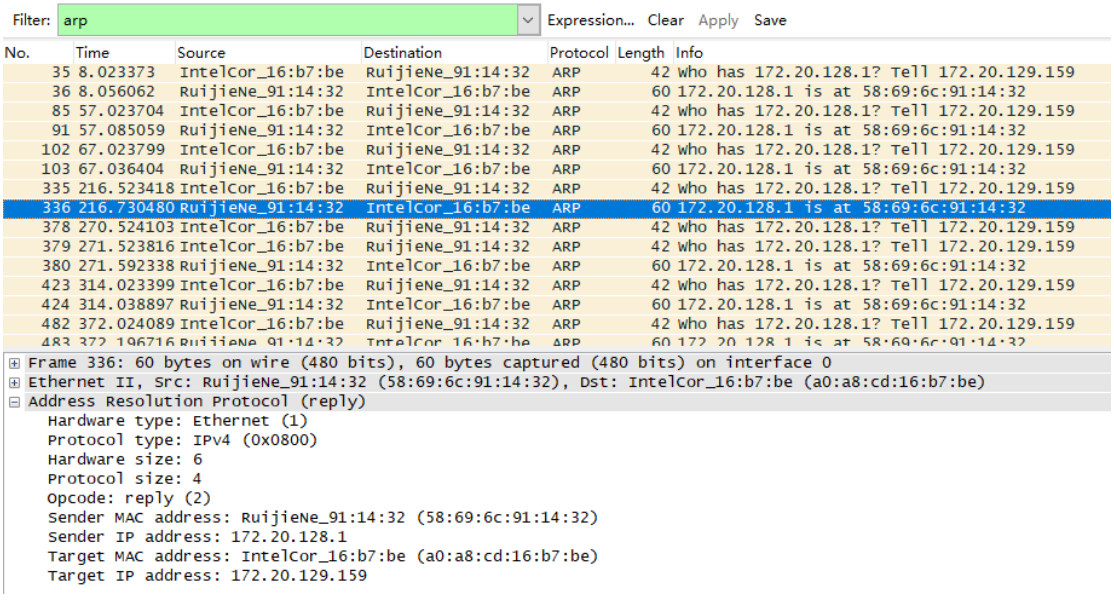
ARP数据包的格式如下图：

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）。

通过 Ping 命令首先给 219.217.226.15 这个 IP 地址发送测试包，如下图



然后打开 Wireshark 开始俘获包，得到如下截图：



2） **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。如下图这是一个请求包：

```
⊞ Frame 379: 42 bytes on wire (336 bits), 42 bytes captured (336
⊞ Ethernet II, Src: IntelCor_16:b7:be (a0:a8:cd:16:b7:be), Dst:
⊟ Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: IntelCor_16:b7:be (a0:a8:cd:16:b7:be)
    Sender IP address: 172.20.129.159
    Target MAC address: RuijieNe_91:14:32 (58:69:6c:91:14:32)
    Target IP address: 172.20.128.1
```

而下面这幅图就是一个响应包：

```
⊞ Frame 91: 60 bytes on wire (480 bits), 60 bytes captured (480
⊞ Ethernet II, Src: RuijieNe_91:14:32 (58:69:6c:91:14:32), Dst:
⊟ Address Resolution Protocol (reply)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: reply (2)
    Sender MAC address: RuijieNe_91:14:32 (58:69:6c:91:14:32)
    Sender IP address: 172.20.128.1
    Target MAC address: IntelCor_16:b7:be (a0:a8:cd:16:b7:be)
    Target IP address: 172.20.129.159
```

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。

5 抓取 UDP 数据包

利用 QQ 发送消息，抓包截图如下：

Filter: udp Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
3	1.879689	172.20.128.1	255.255.255.255	DHCP	370	DHCP ACK - Transa
11	5.467809	123.151.13.169	172.20.129.159	OICQ	121	OICQ Protocol
22	10.336361	172.20.129.159	123.151.13.169	UDP	193	4003 → 8000 Len=151
23	10.463452	172.20.129.159	202.118.224.101	DNS	78	Standard query 0xd150
27	10.527052	123.151.13.169	172.20.129.159	UDP	73	8000 → 4003 Len=31
28	10.550387	172.20.129.159	202.118.224.100	DNS	78	Standard query 0xd150
29	10.559664	172.20.129.159	202.118.224.101	DNS	75	Standard query 0x7643

```
⊞ Frame 22: 193 bytes on wire (1544 bits), 193 bytes captured (1544 bits) on interface 0
⊞ Ethernet II, Src: IntelCor_16:b7:be (a0:a8:cd:16:b7:be), Dst: RuijieNe_91:14:32 (58:69:6c
⊞ Internet Protocol Version 4, Src: 172.20.129.159, Dst: 123.151.13.169
⊟ User Datagram Protocol, Src Port: 4003, Dst Port: 8000
    Source Port: 4003
    Destination Port: 8000
    Length: 159
    Checksum: 0x0bd0 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 1]
⊞ Data (151 bytes)
```

回答问题：

1) **消息是基于UDP的还是TCP的？**

消息是基于UDP的，见上面截图中的红框

2) **你的主机ip地址是什么？目的主机ip地址是什么？**

打开IP报文段，可以得到下图：

```
Source: 172.20.129.159
Destination: 123.151.13.169
```

我主机的IP地址：172.20.129.159

目的IP地址为：123.151.13.169

3) **你的主机发送QQ消息的端口号和QQ服务器的端口号分别是多少？**

```
Source Port: 4003
Destination Port: 8000
```

如图，发送QQ消息的端口号为：4003

QQ服务器的端口号为：8000

4) **数据报的格式是什么样的？都包含哪些字段，分别占多少字节？**

UDP 数据报格式如下图：

UDP数据报格式有首部和数据两个部分。首部很简单，共8字节。包括：
源端口号：2字节
目的端口号：2字节
长度：2字节，UDP用户数据报的总长度，以字节为单位。
校验和：2字节，用于校验UDP数据报的数字段和包含UDP数据报首部的“伪首部”。其校验方法同IP分组首部中的首部校验和。

抓包得到的UDP报文段如下：

```
User Datagram Protocol, Src Port: 4003, Dst Port: 8000
    Source Port: 4003
    Destination Port: 8000
    Length: 159
    Checksum: 0x0bd0 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 1]
```

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？**

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。

可以看出。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，因为每次只发送一个数据报，然后等待服务器响应。

6 DNS 协议分析

打开浏览器，输入http://www.hit.edu.cn/

观察 Wireshark 中抓到的包如下：

Filter: dns Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.20.129.159	202.118.224.101	DNS	74	Standard query 0x61a9 AAAA www.hit.edu.cn
2	0.006937	202.118.224.101	172.20.129.159	DNS	126	Standard query response 0x61a9 AAAA www.hit.edu.cn SOA hit01.hit.edu.cn
236	3.505297	202.118.224.101	172.20.129.159	DNS	73	Standard query response 0xe3af Server failure AAAA www.hitce.net
237	3.513504	202.118.224.100	172.20.129.159	DNS	73	Standard query response 0xe3af Server failure AAAA www.hitce.net
251	7.306036	172.20.129.159	202.118.224.101	DNS	75	Standard query 0x6ace A cdn.onenote.net
252	7.307625	172.20.129.159	202.118.224.101	DNS	75	Standard query 0xefb2 AAAA cdn.onenote.net
253	7.307824	172.20.129.159	202.118.224.101	DNS	88	Standard query 0x5480 A cdn.content.prod.cms.msn.com
254	7.307893	172.20.129.159	202.118.224.101	DNS	88	Standard query 0x551f AAAA cdn.content.prod.cms.msn.com
255	7.332249	172.20.129.159	202.118.224.100	DNS	75	Standard query 0xefb2 AAAA cdn.onenote.net
256	7.332249	172.20.129.159	202.118.224.100	DNS	88	Standard query 0x551f AAAA cdn.content.prod.cms.msn.com

选取一个 dns 的 query 包，查看其 IP 源地址和目的地址如下：

```
Source: 172.20.129.159
Destination: 202.118.224.101
```

即我主机的地址为：172.20.129.159
本地域名服务器的 IP 地址为：202.118.224.101

再看一下 UDP 对应的端口号：

```
User Datagram Protocol, Src Port: 57590, Dst Port: 53
    Source Port: 57590
    Destination Port: 53
```

源端口号为：57590
目的端口号为：53

选取其中一个子查询报文

```
158 1.630749   172.20.129.159    202.118.224.101    DNS      78 Standard query 0xc1b3 AAAA hit.alljournals.cn
```

```
Domain Name System (query)
    [Response In: 162]
    Transaction ID: 0xc1b3
  ⊞ Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  ⊟ Queries
    ⊟ hit.alljournals.cn: type AAAA, class IN
        Name: hit.alljournals.cn
        [Name Length: 18]
        [Label Count: 3]
        Type: AAAA (IPv6 Address) (28)
        Class: IN (0x0001)
```

可以看到，他要查询主机域名为 hit.alljournals.cn 的主机 IP 地址

得到 DNS 对应返回信息为

```
231 1.693860   202.118.224.101   172.20.129.159    DNS      94 Standard query response 0x8207 A hit.alljournals.cn A 118.186.245.88
```

```
    Source Port: 53
    Destination Port: 54657
    Length: 60
    Checksum: 0x0e35 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 17]
⊟ Domain Name System (response)
    [Request In: 157]
    [Time: 0.063631000 seconds]
    Transaction ID: 0x8207
  ⊞ Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 1
    Authority RRs: 0
    Additional RRs: 0
  ⊞ Queries
  ⊟ Answers
    ⊞ hit.alljournals.cn: type A, class IN, addr 118.186.245.88
```

可以看到，主机域名为 hit.alljournals.cn 对应的主机 IP 地址为 118.186.245.88

四、实验心得

通过本次实验，我掌握了使用 Wireshark 工具进行抓包的方法，并且我能够熟练地使用它。刚开始有些生疏，做前面实验速度非常的慢，但做的多了以后也就熟悉了，后面的实验做起来也相对快了许多。在对 IP 进行分析时，需要下载 pingplotter，刚开始在网上下载的版本可能太新了，根本用不了，后来换了低版本才可以使用。与此同时，我还对于之前学习的 HTTP 协议、TCP 协议、UDP 协议、IP 协议、ARP 协议以及 DNS 域名搜索的过程进行了全面的复习，因为若想对 Wireshark 俘获的包进行正确的分析，首先必须要知道每个包的详细内容，例如头部字段都含有哪些部分，每个部分都代表着什么等等。总而言之，经过大量的时间打磨后，我收获颇丰。