哈爾濱工業大學
Harbin Institute of Technology

计算机网络

课程实验报告

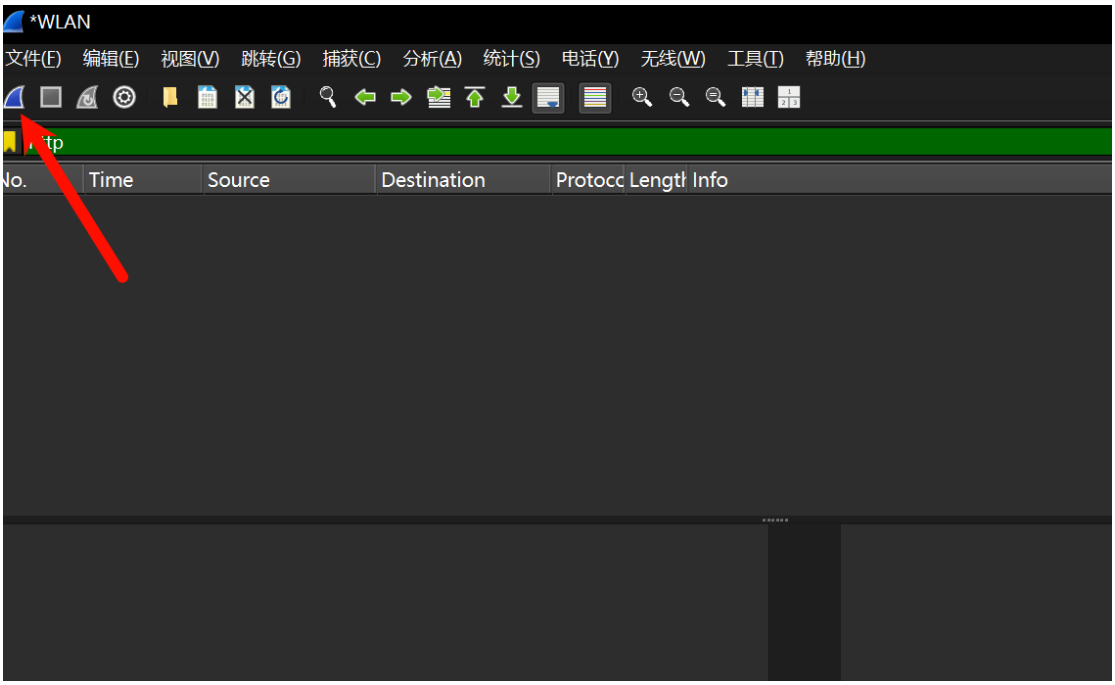
实验名称	利用 Wireshark 进行协议分析					
姓名			院系	计算学部		
班级			学号			
任课教师			指导教师			
实验地点	格物 207		实验时间			
实验课表现	出勤、表现得分(10)		实验报告得分(40)		实验总分	
	操作结果得分(50)					
教师评语						

哈尔滨工业大学计算学部
FACULTY OF COMPUTING, HIT

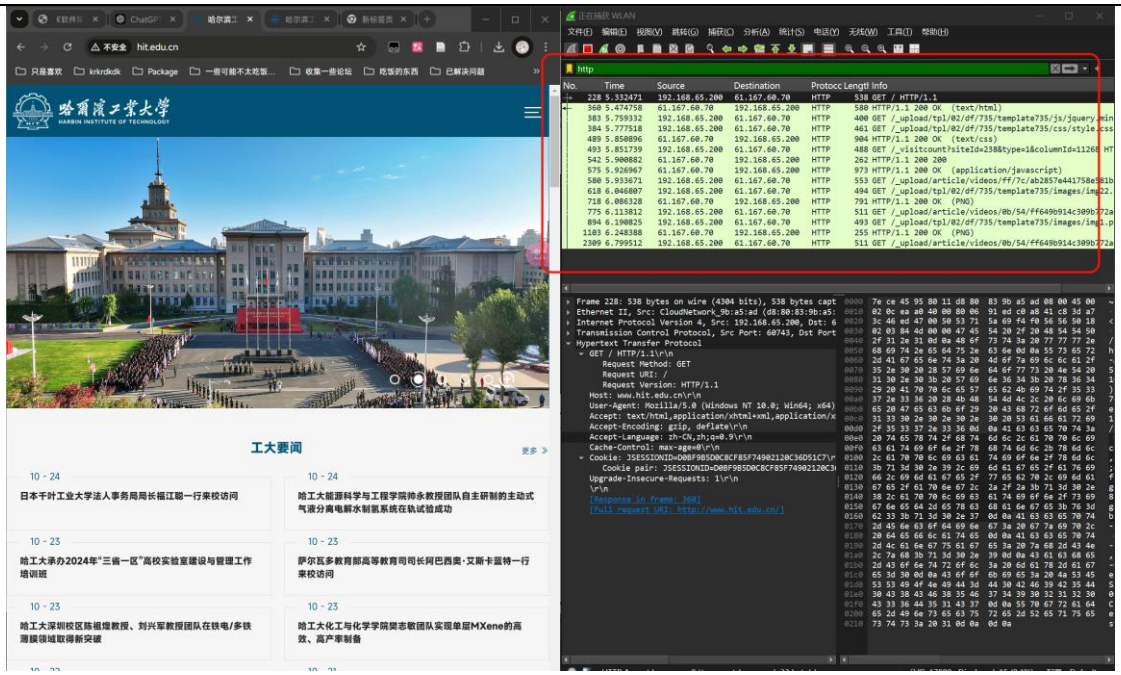
实验目的：
熟悉并掌握 Wireshark 的基本操作，了解网络协议实体间进行交互以 及报文交换的情况。
实验内容：
1) 学习 Wireshark 的使用 2) 利用 Wireshark 分析 HTTP 协议 3) 利用 Wireshark 分析 TCP 协议 4) 利用 Wireshark 分析 IP 协议 5) 利用 Wireshark 分析 Ethernet 数据帧 选做内容： a) 利用 Wireshark 分析 DNS 协议 b) 利用 Wireshark 分析 UDP 协议 c) 利用 Wireshark 分析 ARP 协议
实验过程及实验结果：

1. WireShark的使用

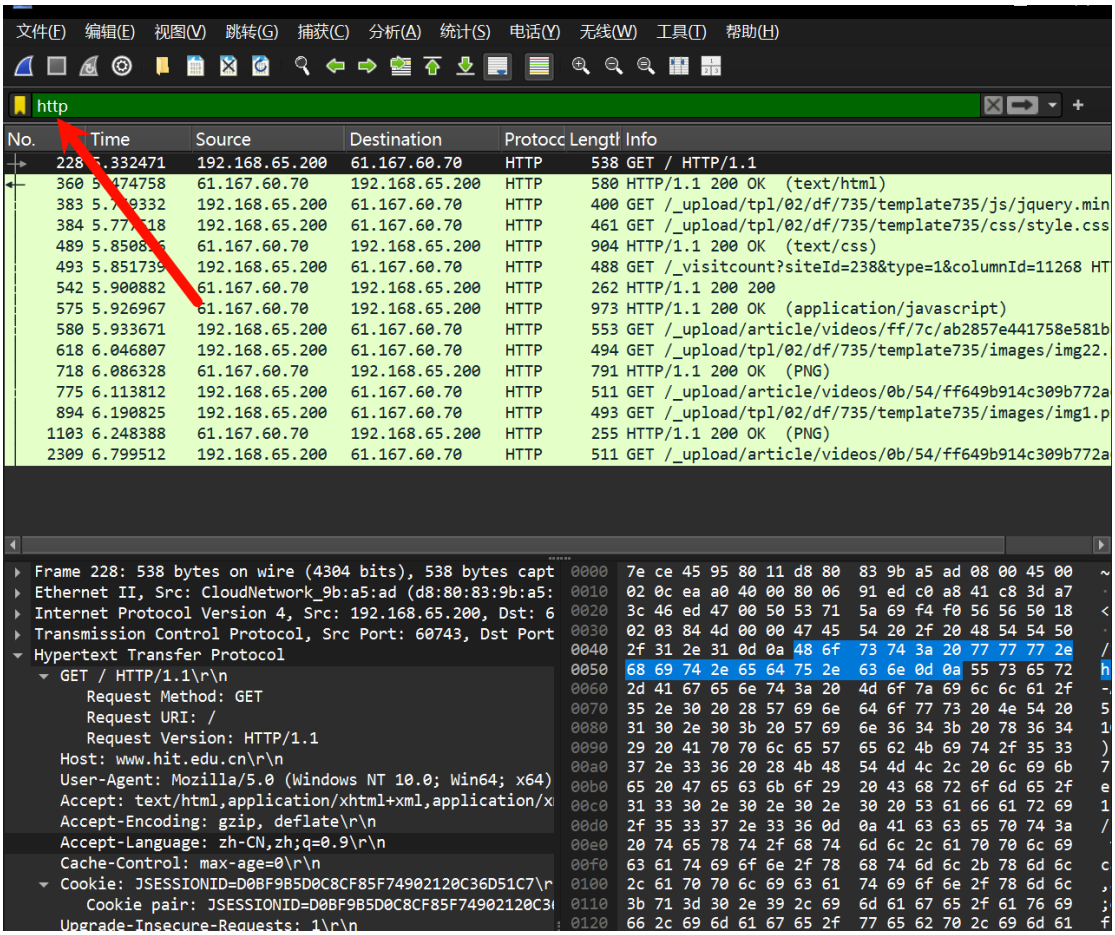
首先打开WireShark并且开始抓包，



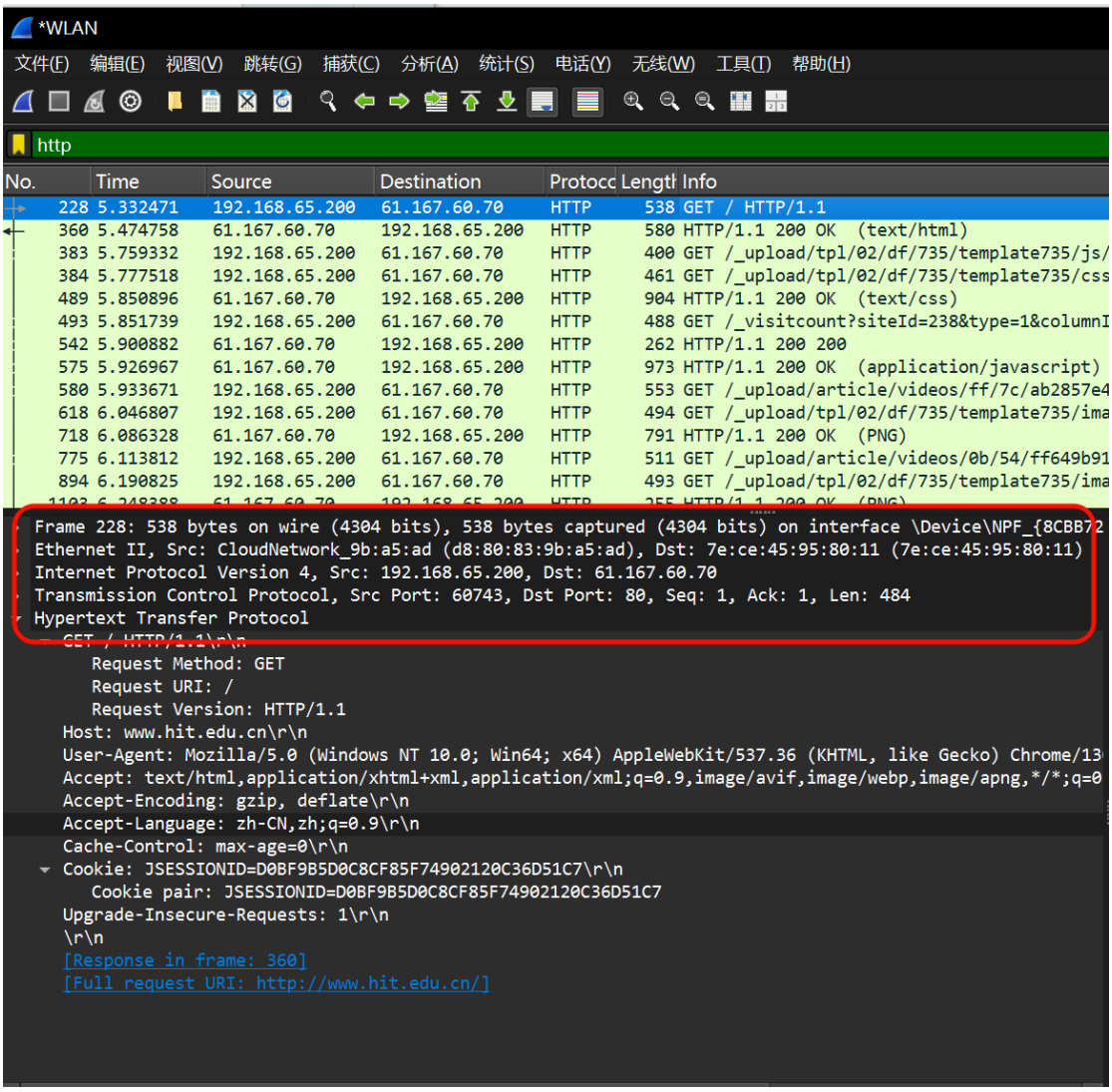
浏览器访问www.hit.edu.cn，在WireShark中可以查看到俘获到的所有分组。



在搜索框中输入http从而筛选http报文，



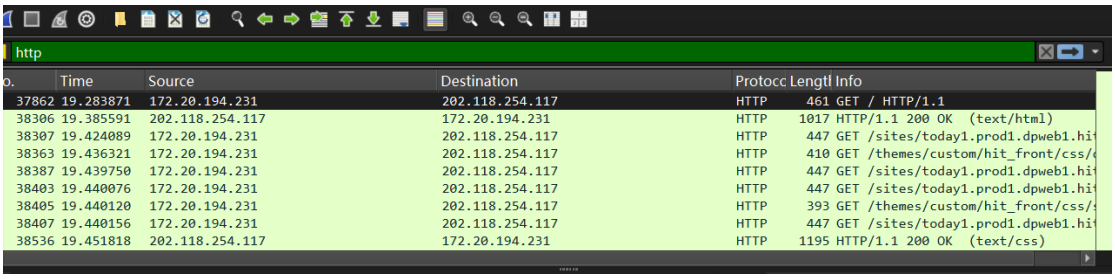
选择第一条 http 报文，以太网帧、IP 数据报、TCP 报文段、以及 HTTP 报文首部信息都将显示在分组首部子窗口中。并且点击三角箭头可以查看每个具体报文的详细信息。



2. HTTP分析

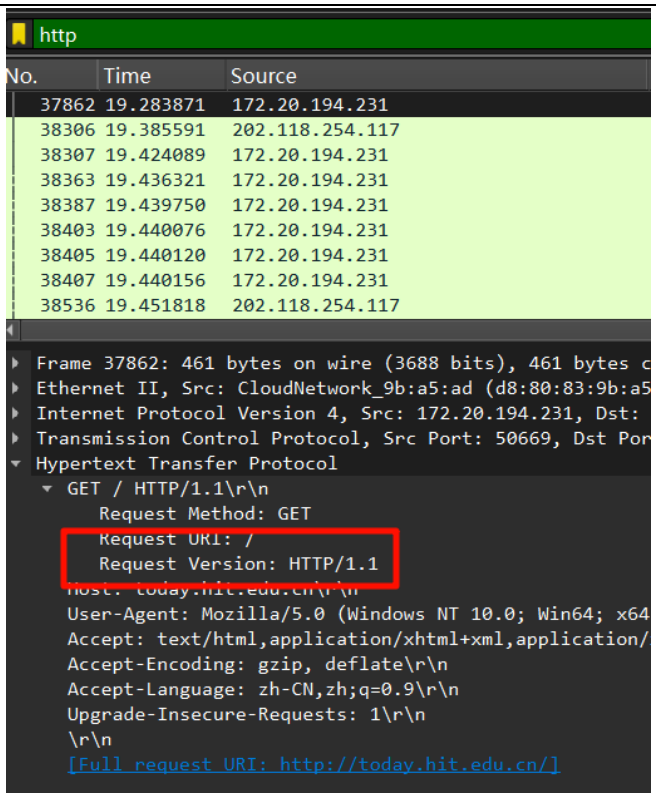
1）HTTP GET/response 交互

WireShark开始抓包，并且在浏览器中访问http://today.hit.edu.cn，今日哈工大网站可以正确返回If-MODIFIED-SINCE的Http header，所以使用此网站能有良好的实验效果，在Wireshark中过滤出http分组，并进行查看。

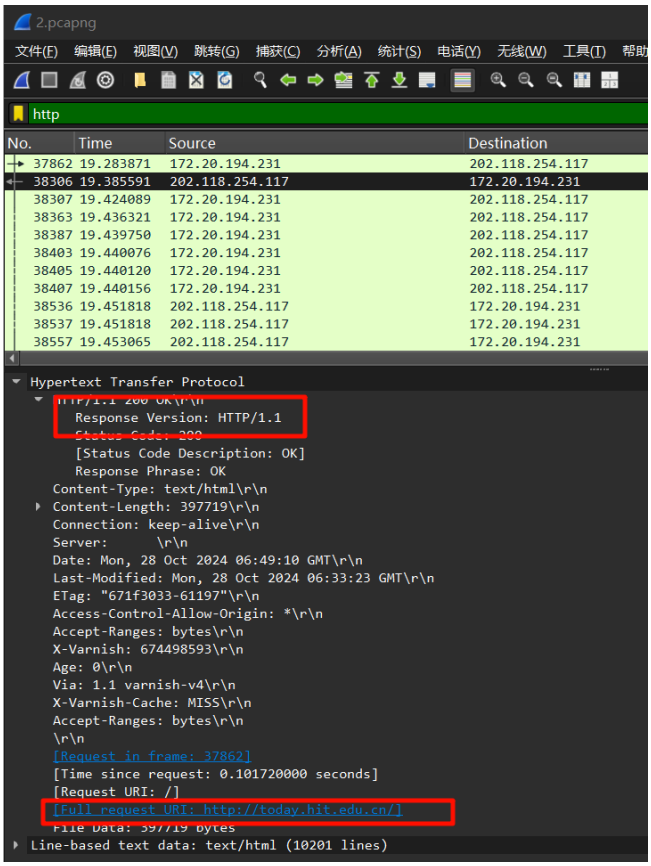


查看其中的第一条http报文，首先是由本机发送给目标服务器的报文，

（1） 可以看见我的浏览器运行的是HTTP1.1协议：



（2） 查看浏览器返回的报文：服务器运行的HTTP协议的版本号也是持久性连接的HTTP1.1，并且返回的报文中的状态码为200。



（3） 从下图可以看出浏览器向服务器指定的接受的语言版本的对象是zh-CN以及zh。

```
        Request Method: GET
        Request URI: /
        Request Version: HTTP/1.1
    Host: today.hit.edu.cn\r\n
    User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KH
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,ima
    Accept-Encoding: gzip, deflate\r\n
    Accept-Language: zh-CN,zh;q=0.9\r\n
    Upgrade-Insecure-Requests: 1\r\n
    \r\n
    [Response in frame: 38306]
    [Full request URI: http://today.hit.edu.cn/]
```

（4） 从下图的IP数据报中可以看见本机的IP地址和服务器的IP地址，其中本机的IP地址是172.20.194.231，服务器的IP地址为202.118.254.117。

```
Internet Protocol Version 4, Src: 172.20.194.231, Dst: 202.118.254.117
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 447
    Identification: 0x8189 (33161)
  ▸ 010. .... = Flags: 0x2, Don't fragment
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 128
    Protocol: TCP (6)
    Header Checksum: 0x3fc7 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.20.194.231
    Destination Address: 202.118.254.117
    [Stream index: 3]
```

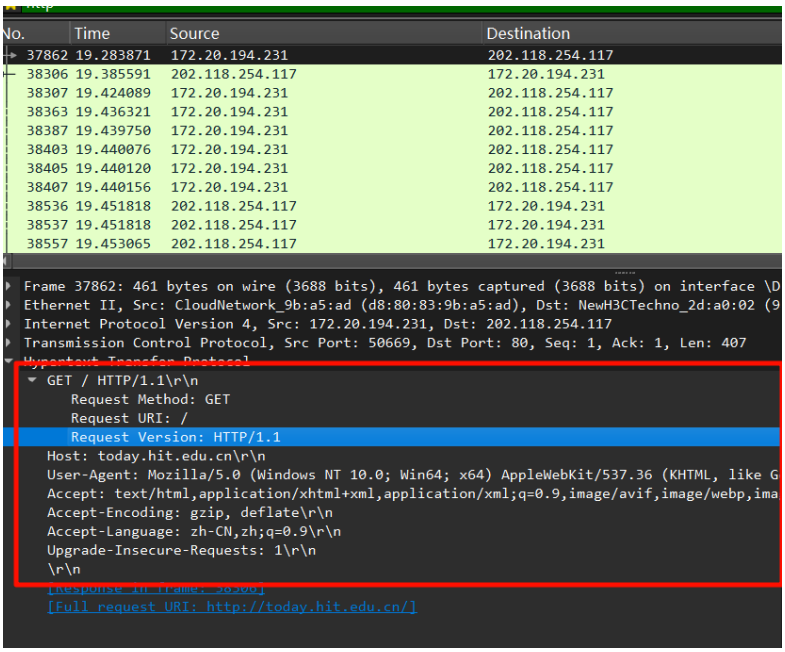
（5） 浏览器返回的状态码为200

```
Hypertext Transfer Protocol
  HTTP/1.1 200 OK\r\n
      Response Version: HTTP/1.1
      Status Code: 200
      [Status Code Description: OK]
      Response Phrase: OK
    Content-Type: text/html; charset=UTF-8\r\
```

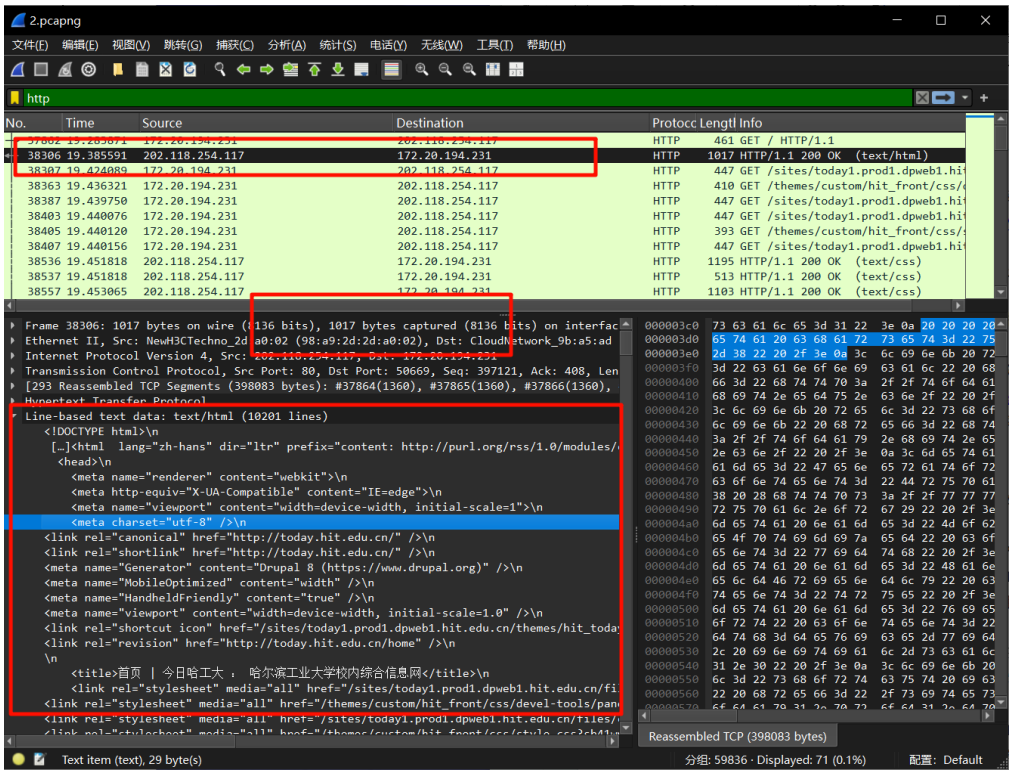
2）HTTP条件GET/response交互

清理浏览器缓存后，开启wireshark，刷新当前页面，查看wireshark俘获的http报文中第一条由客户端发送给服务器的get请求，

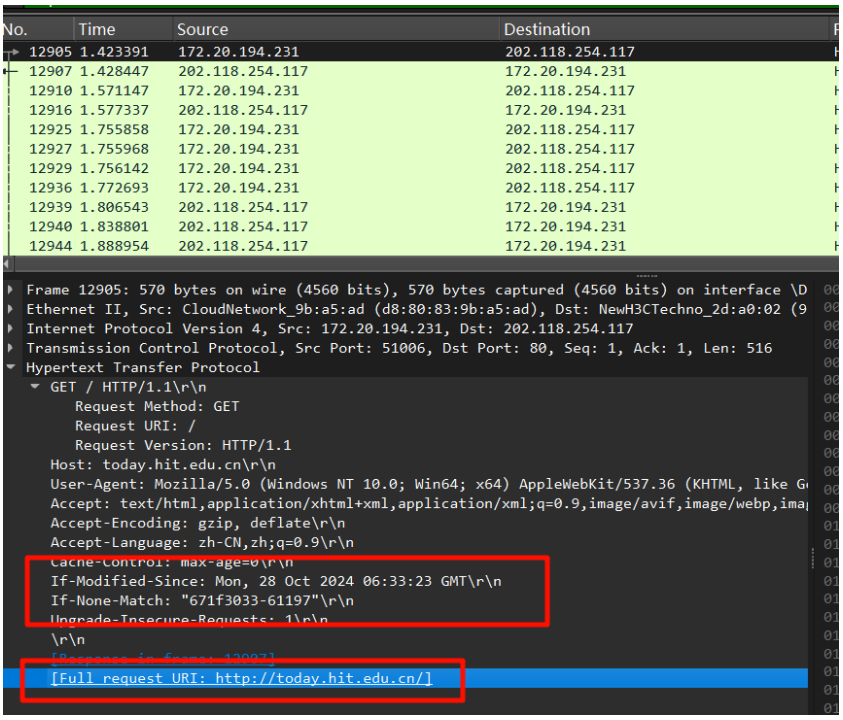
（1） 清理缓存后的首次访问今日哈工大主页是不会有IF-MODIFIED-SINCE请求头：



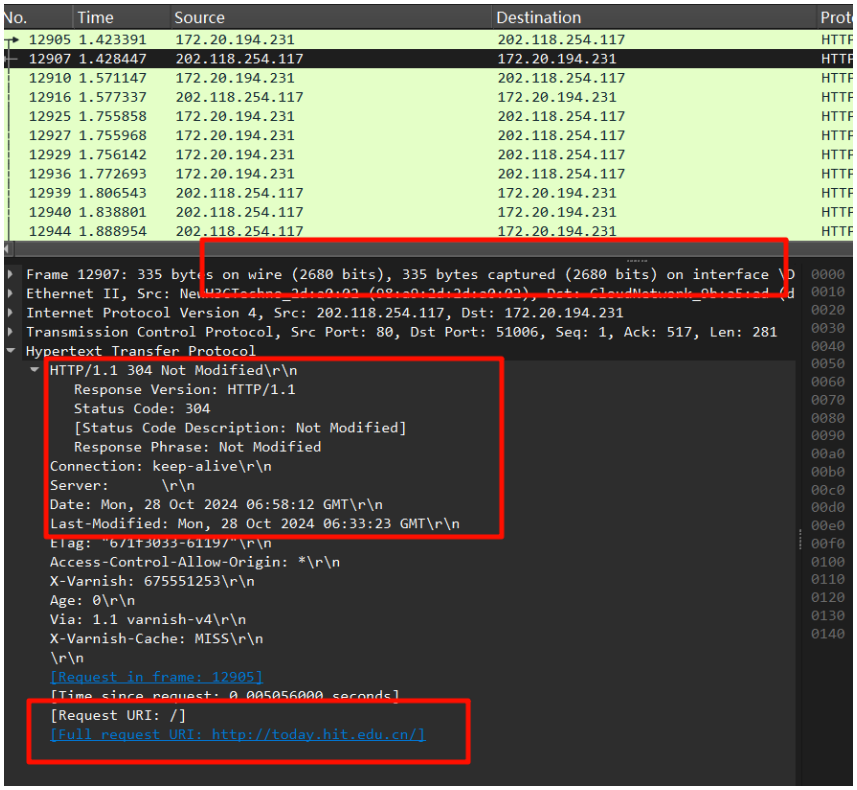
（2） 下图可以看到，服务器是否明确的返回文件可以在wireshark中看见是否有data内容，下图中存在名为Line-based text data: text/html 一栏，并且链路层数据包大小为1017bytes，所以服务器明确返回了文件的内容：



（3） 而刷新后重新访问的请求头中可以查看到当前存在IF-MODIFIED-SINCE请求。该首部行后面跟着的信息应当是上一次服务器返回该资源的时间，也就是上一次的返回的reponse中的Last-Modified的http header中的时间数据，主机询问在该时间后服务器中的资源是否被修改。

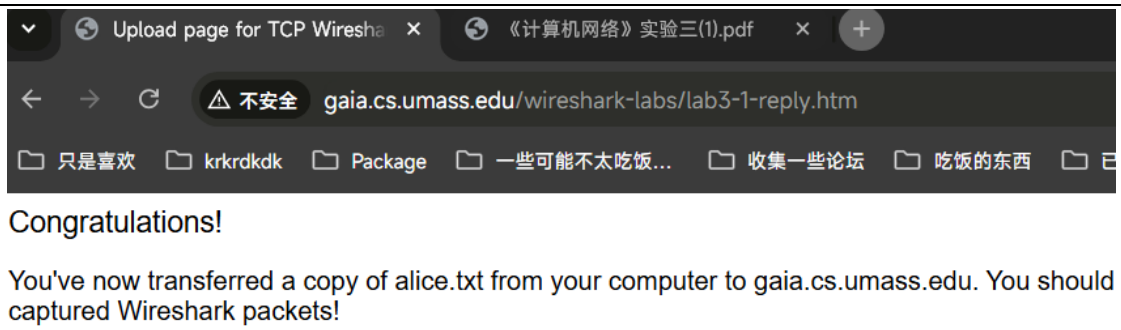


（4） 第二次http get请求的时候，服务器返回的http状态码为304，返回的数据中不存在Line-based text data: text/html栏，并且链路层的数据包大小也仅有334bytes，服务器没有明确返回文件内容：

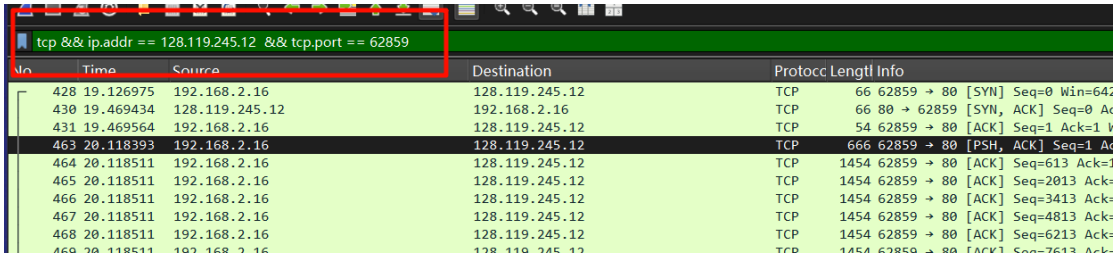


3. TCP分析

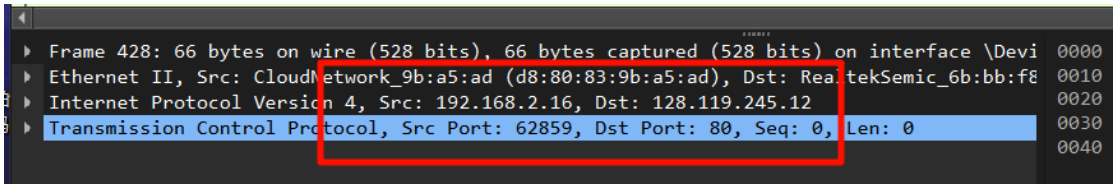
在启动完wireshark并且上传完文件后，当上传页面变为下图时，停止wireshark的俘获：



在wireshark中限定tcp以及ip.addr ==128.119.245.12后，查看到过滤的所有和服务器进行交互的tcp数据包：

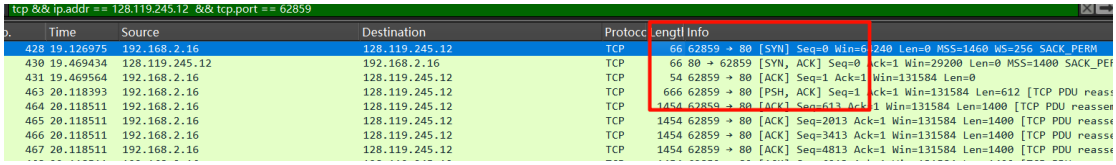


选取第二个数据查看：

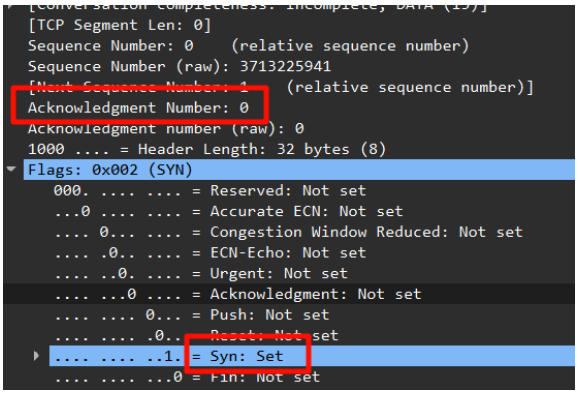


（1） 客户端主机的IP地址为192.168.2.16，因为我在寝室的路由器下，所以网段和校园网有所差异，端口为62859，可以看见图中同时出现了62858和62859两个端口，但62858端口在往后的传输中未被使用，62859在后续的传输中占据了主要地位。

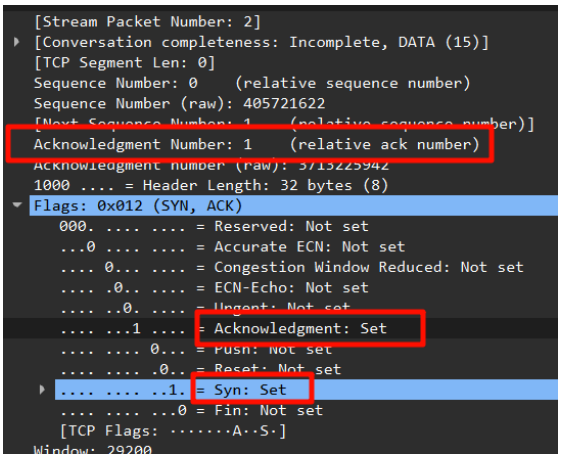
（2） 服务器主机的IP地址是128.119.245.12，并且端口号是80。



（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0，在该报文段中，将SYN标志位置1，ACK标志位置0，表示这是请求建立TCP的第一次握手，标示该报文段是 SYN报文段。

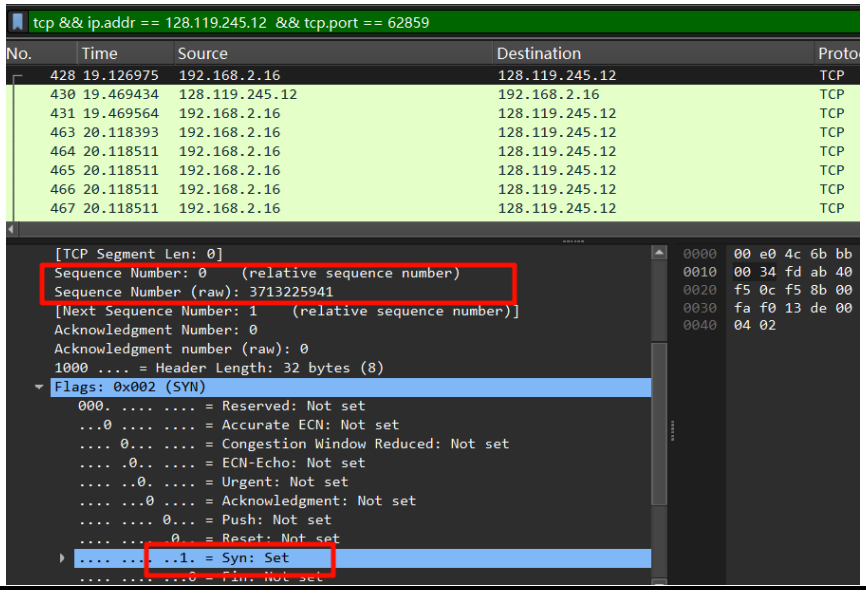


（4） 服务器向客户端发送的 SYNACK 报文段序号是0，该报文段中，Acknowledgement 字段的值是1，客户端发送的第一次握手中并没有承载字节信息，但是仍要占据一个序号，因此客户端下一个要发送的数据字节序号从1开始，将ack_seq置为1，表示seq=0及之前的数据都收到，可以发送后面的数据。将ACK标志位与SYN标志位都置1，表示这是TCP连接建立过程中的第二次握手，标示该报文段是SYNACK报文段。

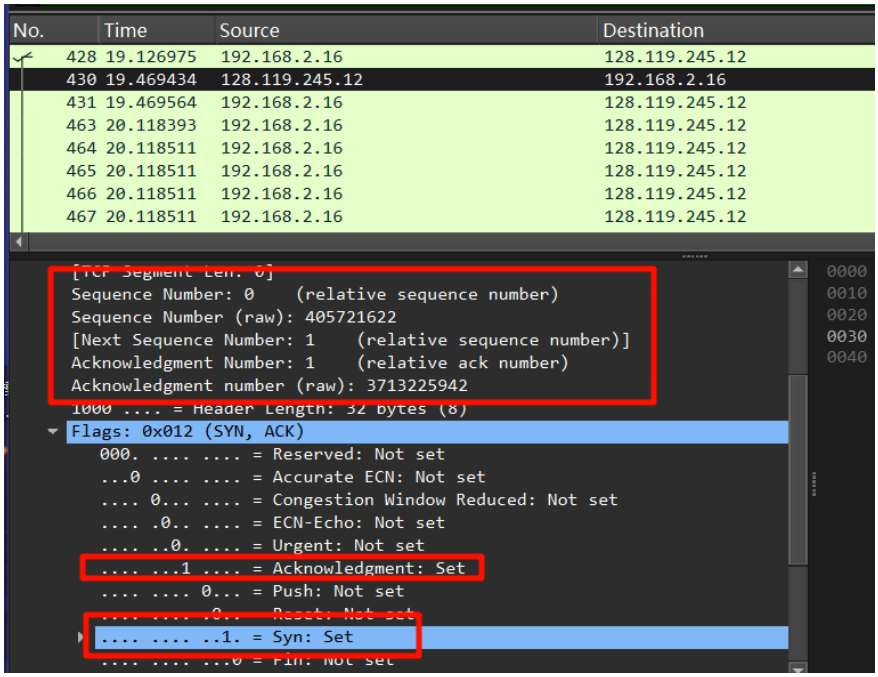


（5） 分析出 TCP 三次握手过程

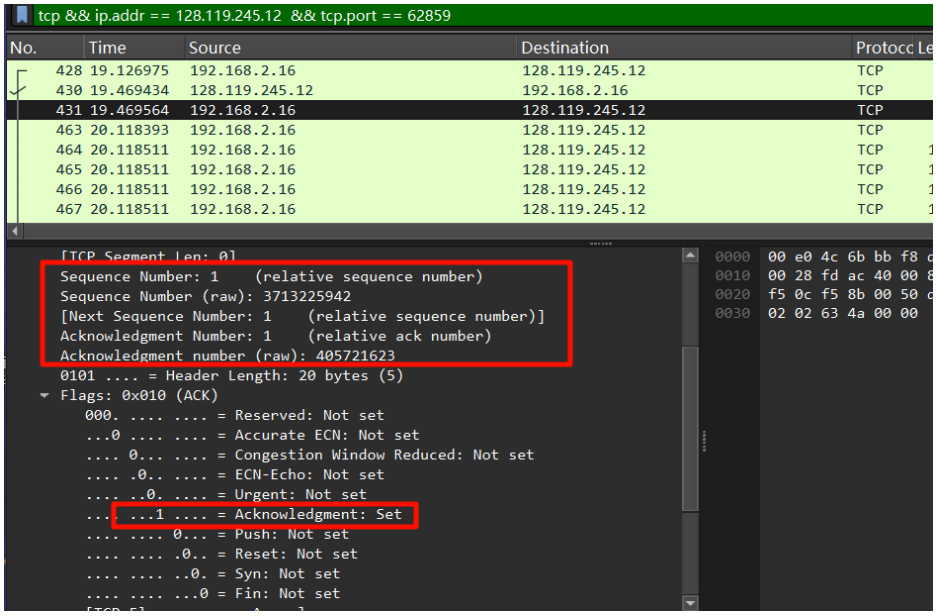
a) TCP的三次握手过程如下图，首先是客户端首先将TCP头部的SYN标志位置为1，主机选取0为发送的第一个字节数据的序号，将相对序号seq位赋为0，实际的seq序号（raw）是下图中的3713225941，将报文发送给服务器端用于第一次握手，等待连接的建立。



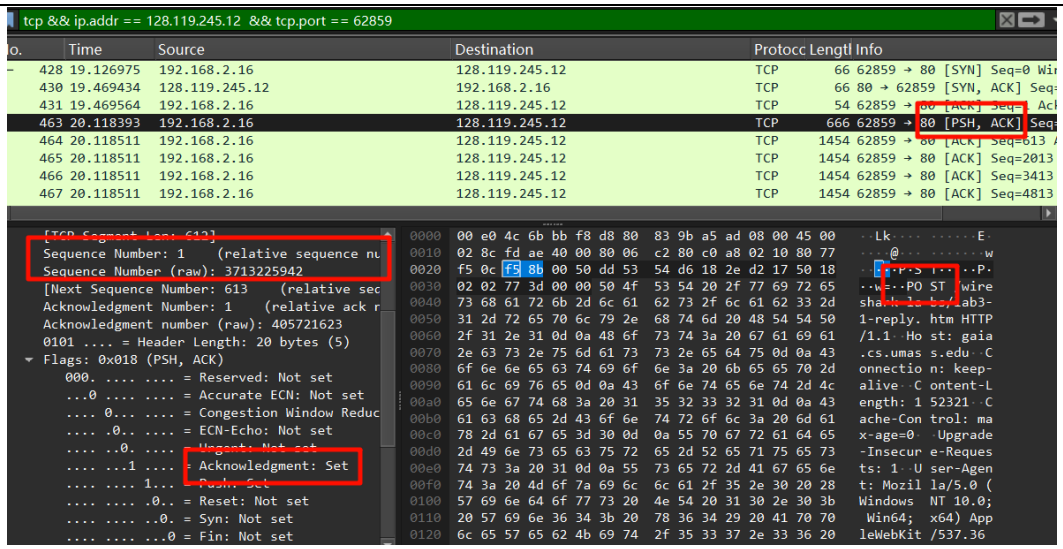
b) 其次过程如下图，服务器端接受到客户端发送的请求连接报文后，准备返回的tcp报文，将TCP头部的SYN标志位和ACK标志位设置为1，设置next_seq_number为1，实际的raw_seq_number为405721622，并且返回ack_num为1，raw_ack_num和上一过程中客户端发送给服务器的raw_seq_num一致，都为3713225942。



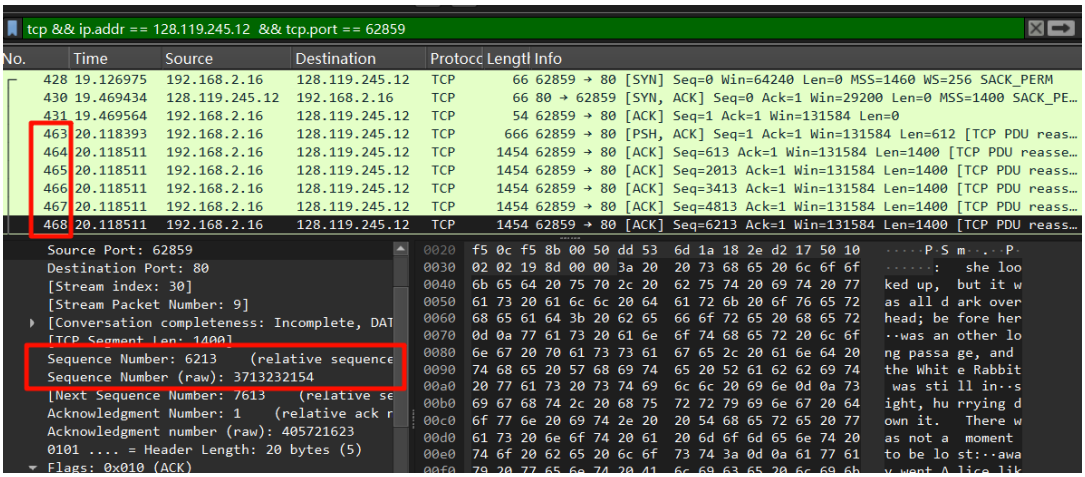
c) 客户端收到确认报文后，建立TCP连接，为了确认收到了服务器端的报文，将ACK标志位置1，SYN标志设置为0，第一个报文没有载荷数据但是也占据了一个序号，因此序号seq为1，和上一个步骤服务器返回的next_seq_num保持一致，同时将相对确认序号ack_seq赋为1，raw_ack_num和上一步骤服务器发送的raw_seq_num保持一致，将报文发送给服务器端用于第三次握手，该报文中可以承载信息。服务器端收到该报文后建立连接。



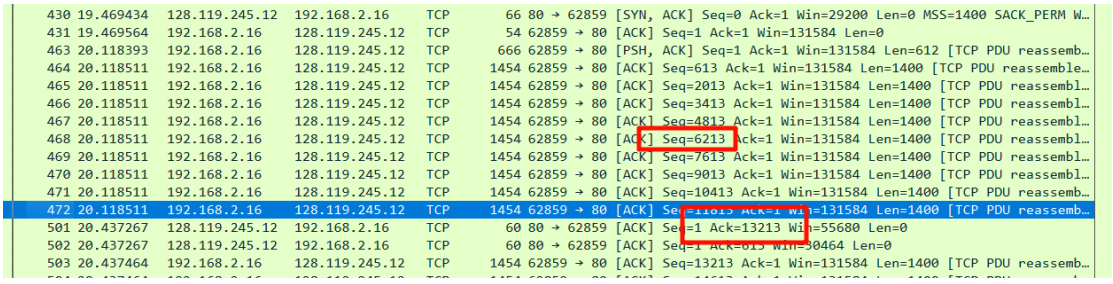
（6） 包含HTTP POST命令的TCP报文段应当是PSH标识被设置为1的TCP段，如下图所示，因为这是在建立TCP连接后，客户端向服务器段发送的第一个TCP数据段，这个段的的相对序号是1，而raw_seq_num被设置为3713225942，似乎是复用了TCP连接的最后一步中的seq序号。



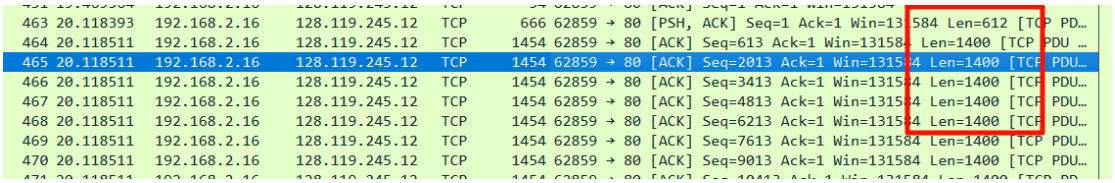
（7） 第六个报文段的相对序号为6213，raw_seq_num为3713232154，发送的时间是TCP建立的连接之后，并且在FIN之前，wireshark中时间为20.118511时刻。



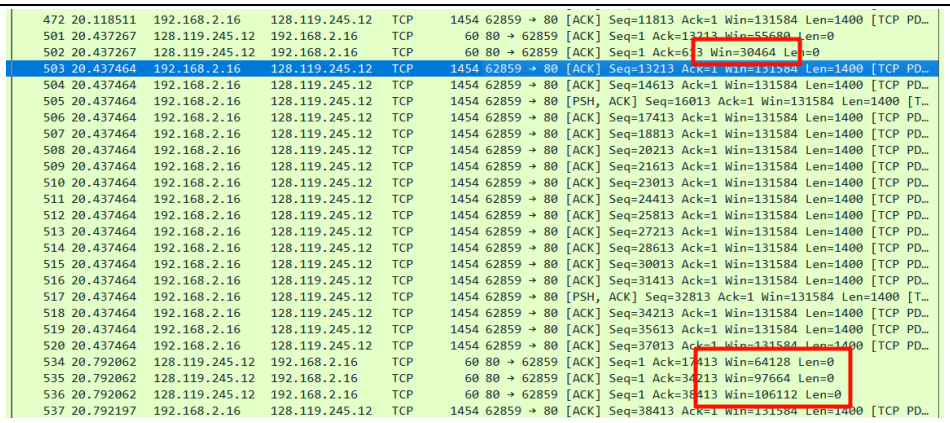
而这个包的确认被包含在 Ack=13213（相对 Ack）中，这是由于 **tcp 的积累确认机制**，所以这个包的接受的时间应当在 Ack=13213 发送之前就已经确认接受。



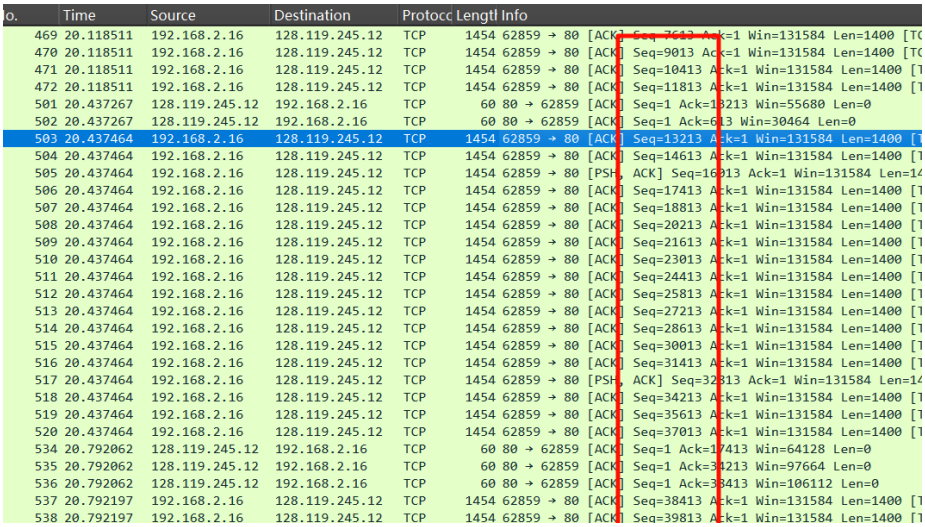
（8） 前六个包的长度为612、1400、1400、1400、1400、1400。



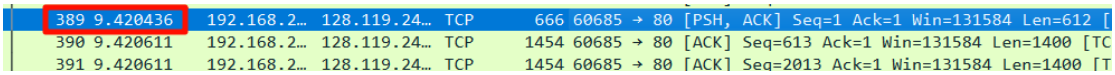
（9） 在整个跟踪过程中，接收端公示的最小的可用缓存空间是win=30646,在限制发送端的传输后，接收端的缓存是足够使用的，可以看见缓存在逐步增大，最终稳定在10k+。



（10） 文件中没有重传的数据段，通过俘获页面的tcp数据段的序号可以看出，序号总体来说是上升的，并且没有重复和减少的情况出现，说明传输过程中没有发生报文的重传

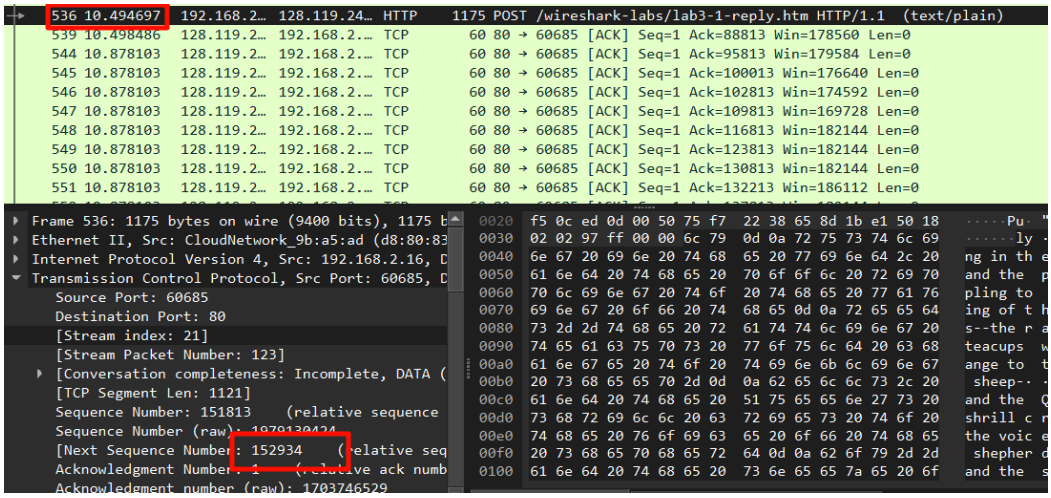


（11） 文件开始传输的部分是



文件结束部分是一个post请求结束，没有显式的FIN标志设置。最后一个字节是152933，因为下一个期待的next_seq_num是152934。所以计算:

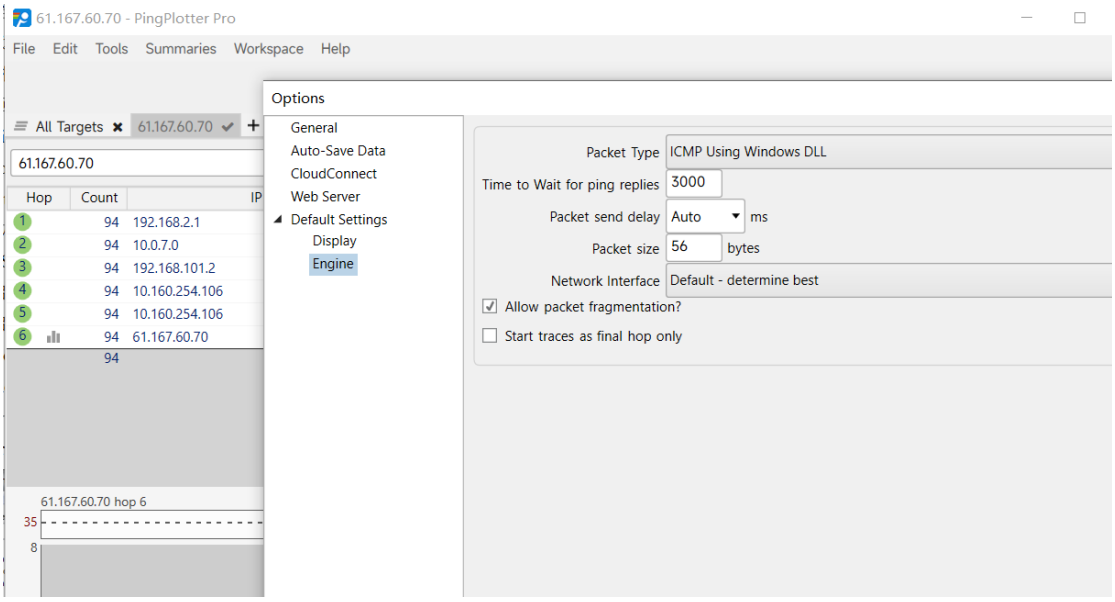
$$throughput = \frac{152933B}{(10.494647 - 9.420426)s} = 142{,}366.42Bps$$



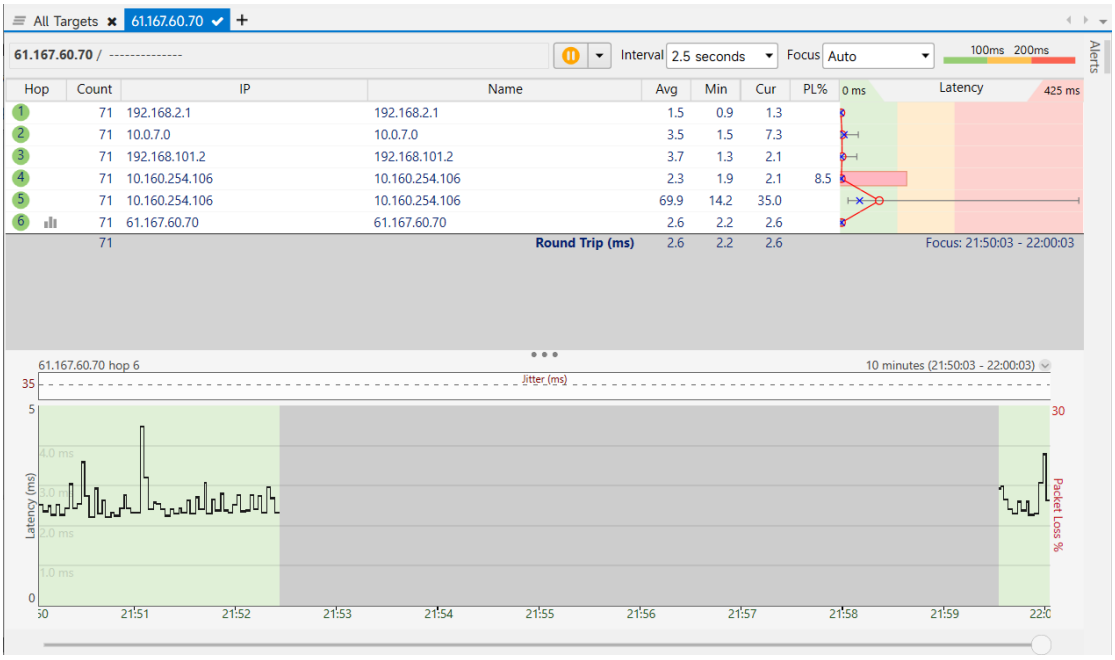
4. IP分析

首先安装Ping Plotter，由于默认的向www.hit.edu.cn发送包的时候使用的是ipv6，遂通过ping www.hit.edu.cn -4命令获得源ipv4地址为61.167.60.70，于是改向此ipv4地址发送包。

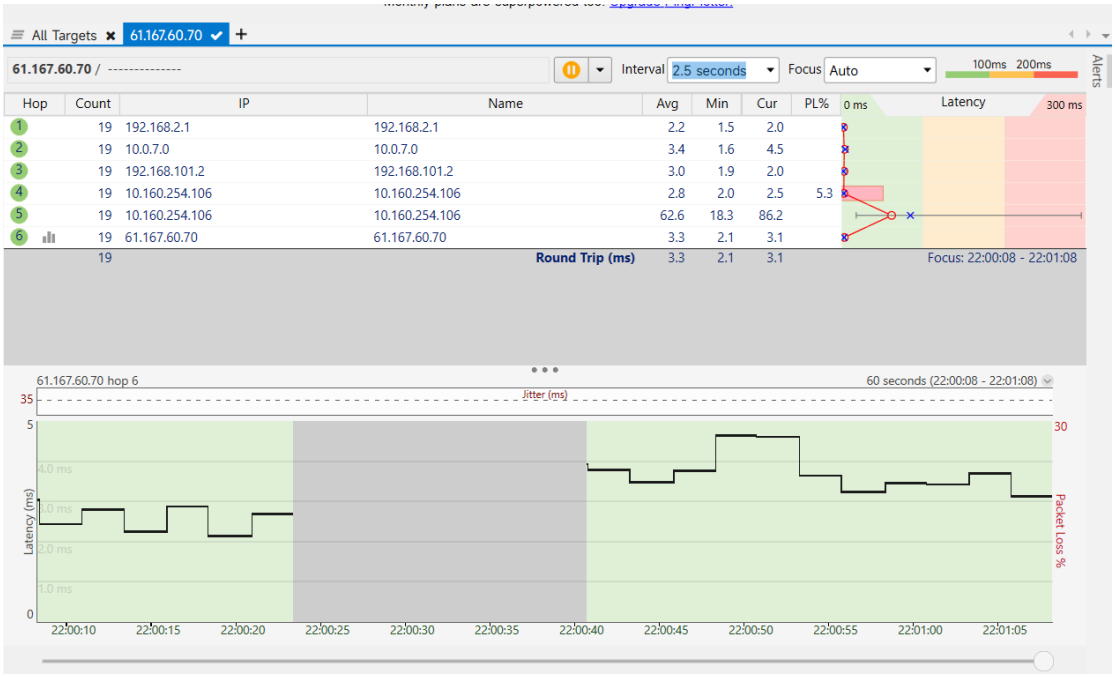
在edit -> options -> default settings -> engine中可以修改包的大小，但是并没有# of times to Trace选项。



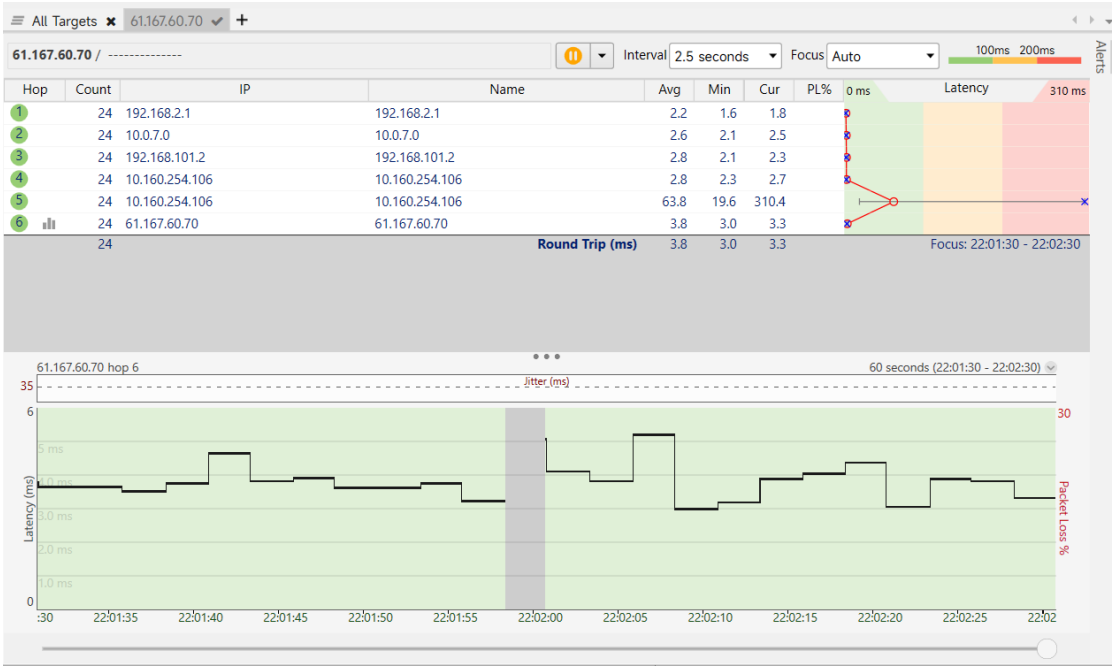
首先设置包的大小为56B。点击start按钮开始向服务器发送ICMP包。



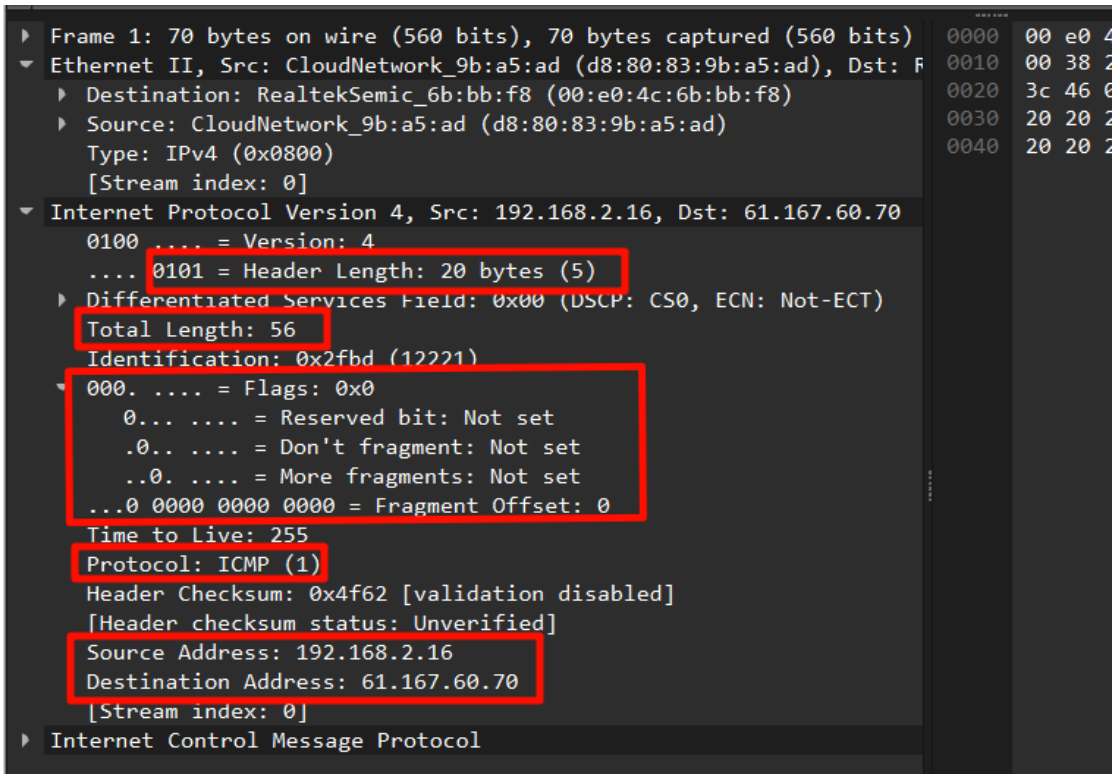
重新设置包的大小为2000B，再次发送。



重新设置包的大小为3500B，再次发送。

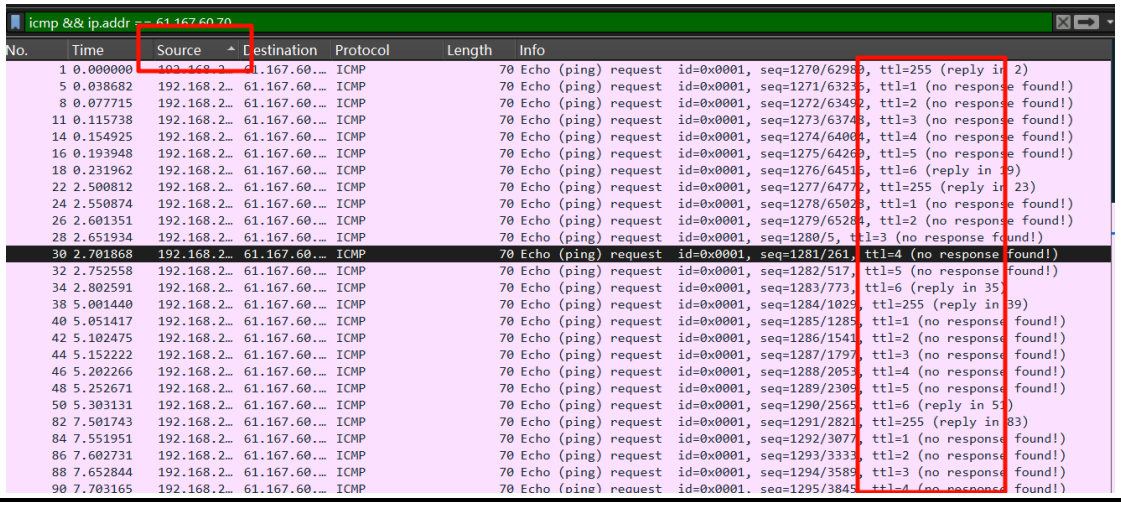


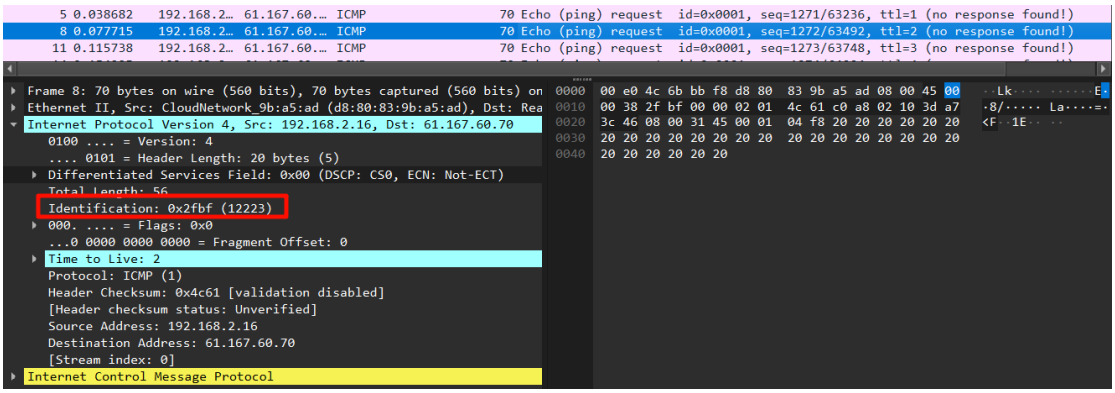
选择第一个我的主机发出的ICMP Echo Request消息，在packet details窗口展开数据包的Internet Protocol部分。

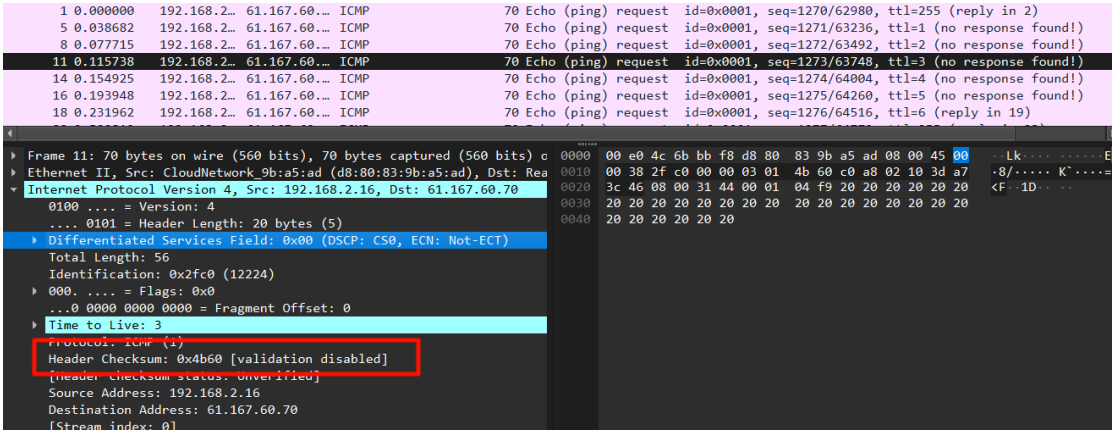


(一) **思考一**，从上图可以看出

1) 本地主机的IP地址是192.168.2.16，服务器的IP地址是61.167.60.70。
2) IP数据包头中，上层协议的字段值是 1，此处表示的是ICMP协议。
3) IP头的总长度（Total Length字段）是56B，其中IP数据包的头部(Header Length字段)是20B，所以真正的净载字节的大小为56B-20B=36B。
4) IP数据包的净载大小为46B。
5) 该数据包没有分片，可以查看Flags标志中的DF（don't fragment）标志以及MF（more fragments）标志为0，标志该数据包可以被分片，并且没有被分片，或者该数据包为最后一篇，且Fragment Offset片位移为0，所以分组是没有被分片的



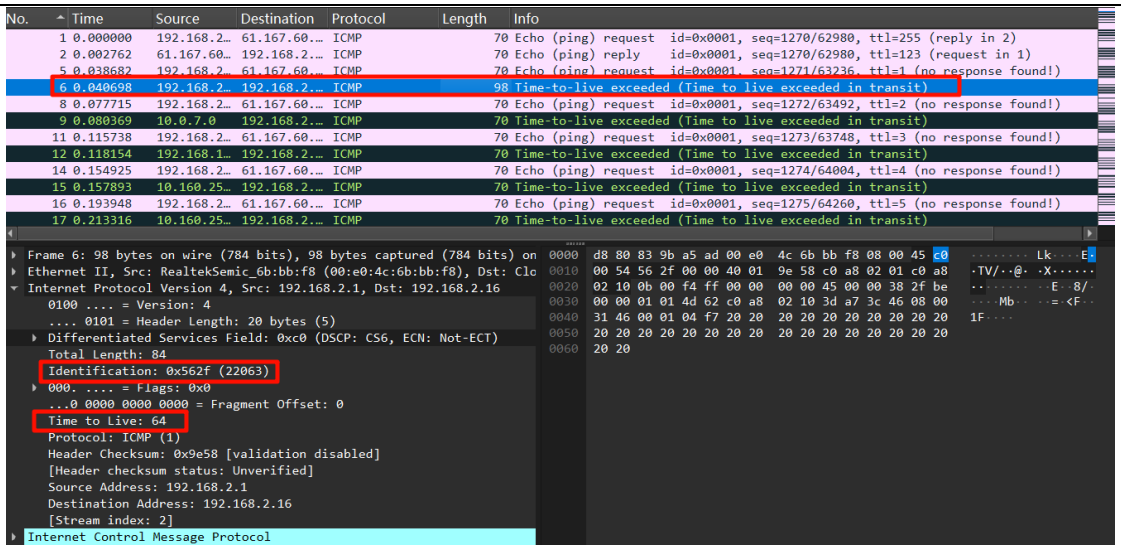




(二) 思考二

对俘获的数据包按照Source IP进行排序，根据对应的数据包进行分析

1) 主机发出的ICMP中的IP数据包的identification、TTL、Header Checksum字段总是在改变。
2) IP数据报中的**版本、首部长度、标志位、协议、源IP地址和目的IP地址**等字段总保持不变，是为了确保数据包能被正确解析、路由，并确保通信的完整性和可靠性。如果这些关键字段在传输过程中发生变化，可能会导致数据包无法正常送达或被错误处理。因此这些字段都应当保持一致。TTL变化的原因是：每经过一个路由器递减 1，防止数据包无限循环。Identification变化的原因是：每次发送新的数据包时递增，用于标识和重组分片。Header Checksum变化的原因是：头部字段（如TTL）变化后，需要重新计算校验和。图中可以看出TTL从1-5都没有正确回复，当TTL为6时才有正确回复以及帧序号。
3) IP数据包中的Identification字段是以16进制数的形式出现的，并且在相邻的两个两个IP数据包中，这个字段是逐步的加一的递增的。



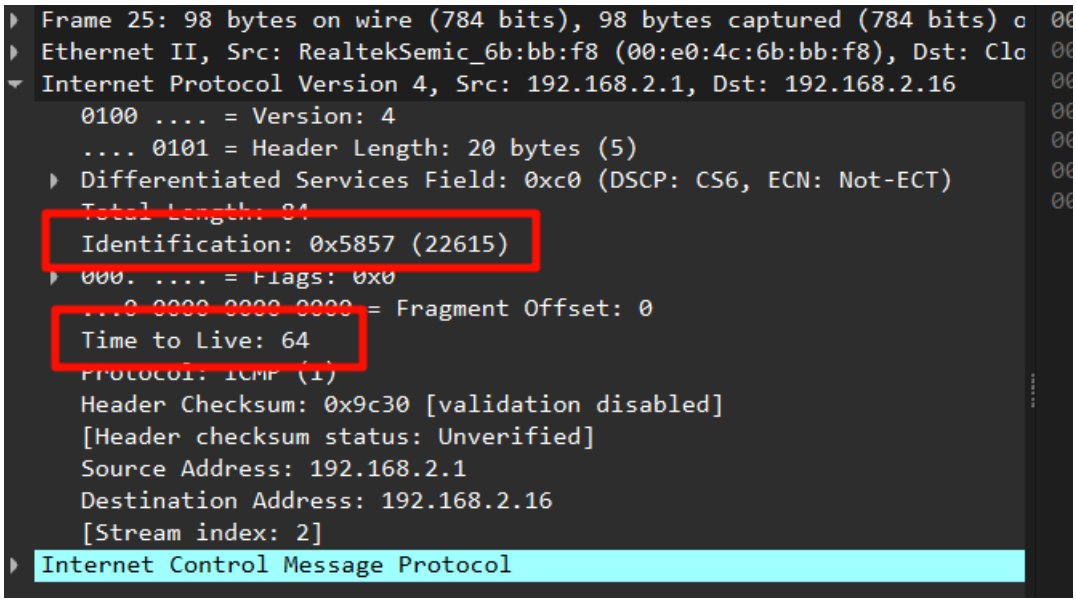
(三) 思考三

上图中是最近的第一条路由器返回的ICMP的TTL exceeded消息。

1） Identification字段是 0x562f（22063），TTL字段是64。

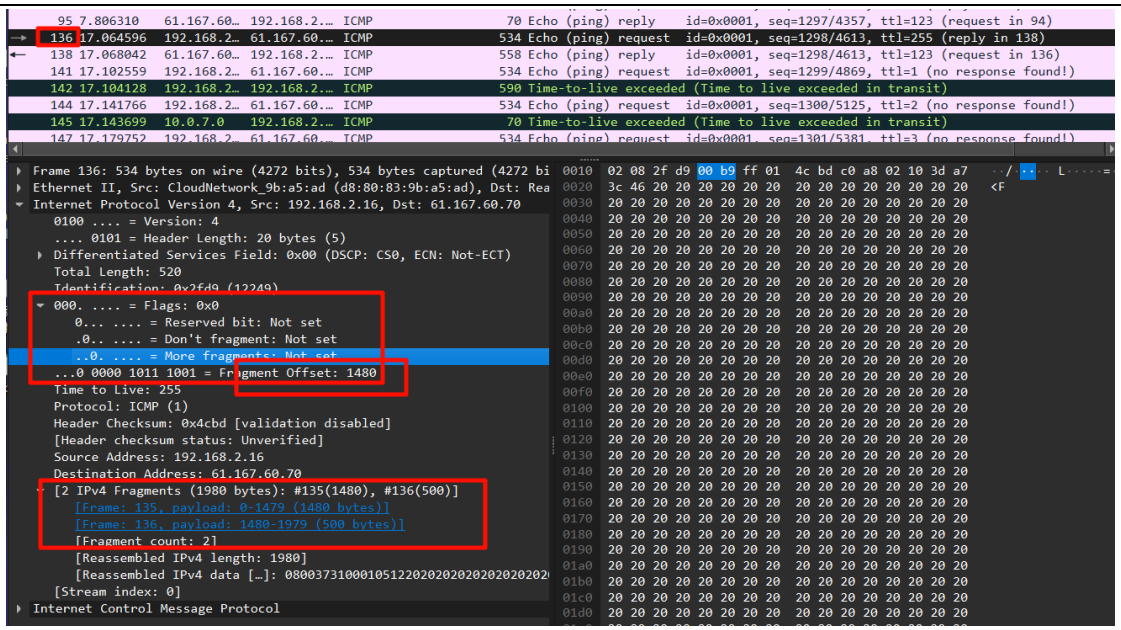
2） 查看下面的图片，

Identification字段是会改变的，因为Identification（标识符）用于在 IP 分片时标识属于同一数据包的所有片段，确保接收方可以正确重组它们。如果数据报没有分片要求，Identification仍会分配一个值，作为该数据包的唯一标识。但是生成策略与具体的系统有关。部分嵌入式系统以及windows可能会采用随机数生成，而linux系统可能会采用顺序Identification值。而TTL字段的初始值的选择也与最近的路由的操作系统有关，Linux下此值通常为64,也就是说当前的网络下的路由器可能是Linux系统或者其他嵌入式类型的系统。



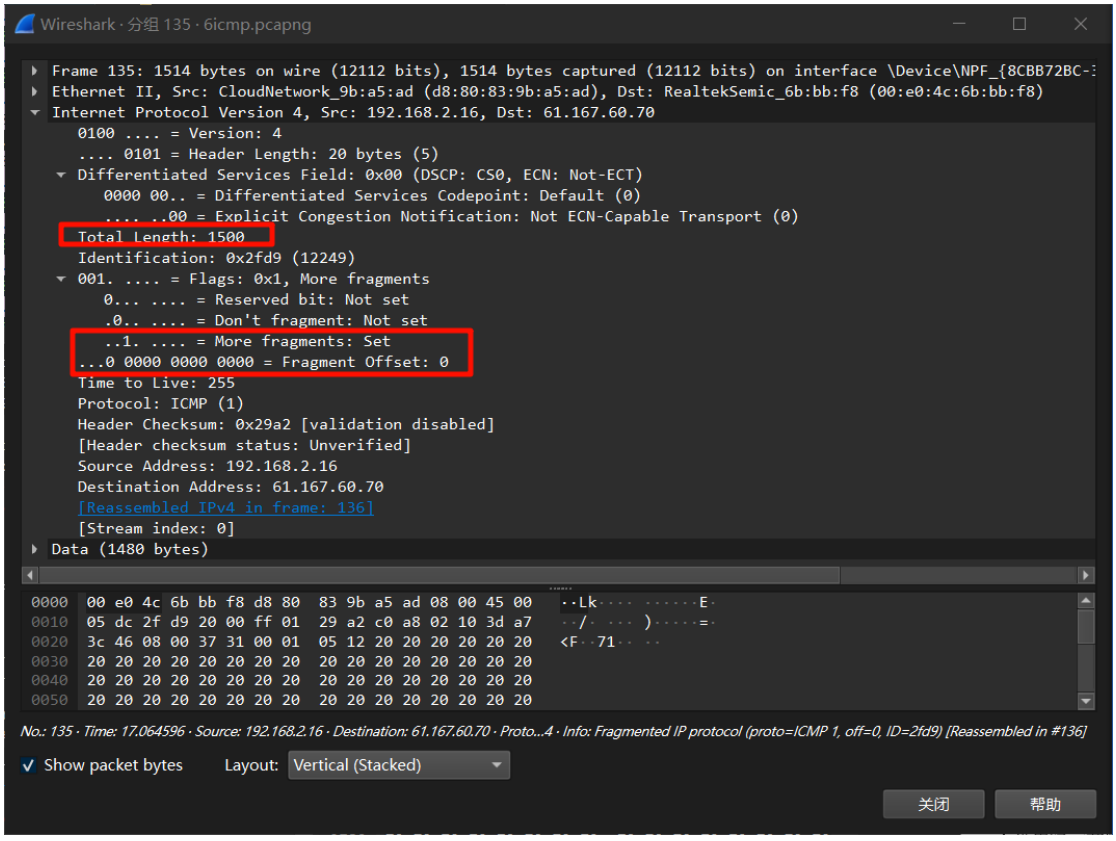
(四) 思考四

包的大小被更改为2000字节后的第一个ICMP Echo Request消息如图所示：



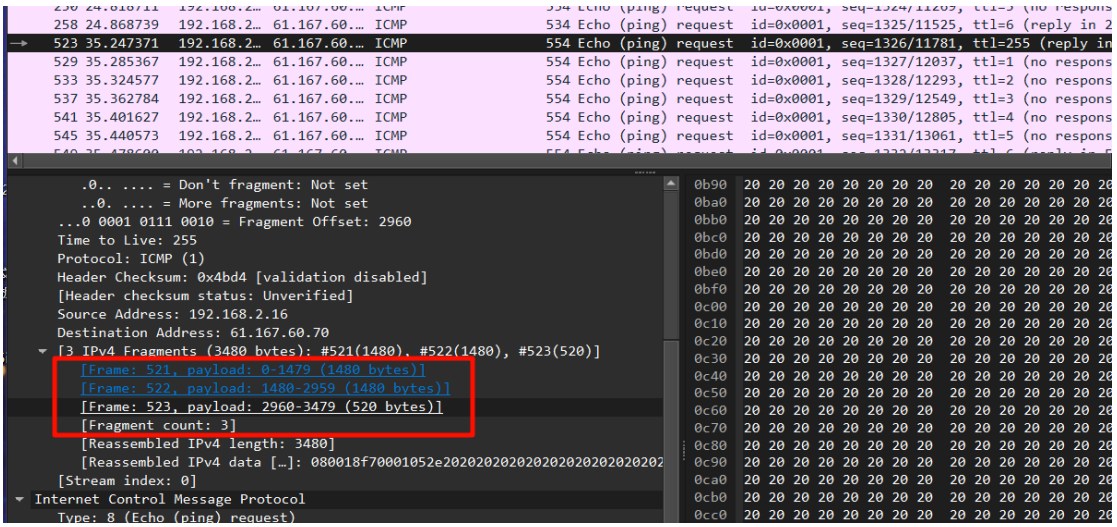
1） 此消息被分解为了多个IP数据包

2） 可以看见IP头部中的don't fragment和More fragment的值为0（此处为0是因为查看的是合并后的IP数据包，下图中含有分片后单独的数据包。），但是Fragment Offset值为1480，更多的可以在下面的内容中看见两片IPv4的分片，长度分别为1480B和500B，所以该消息被分片了。分别查看分片，对于分片1来说，offset为0，**并且more fragments 字段为1。并且分片1的总长度为1500B，除去头部的携带数据的长度为1480B。**



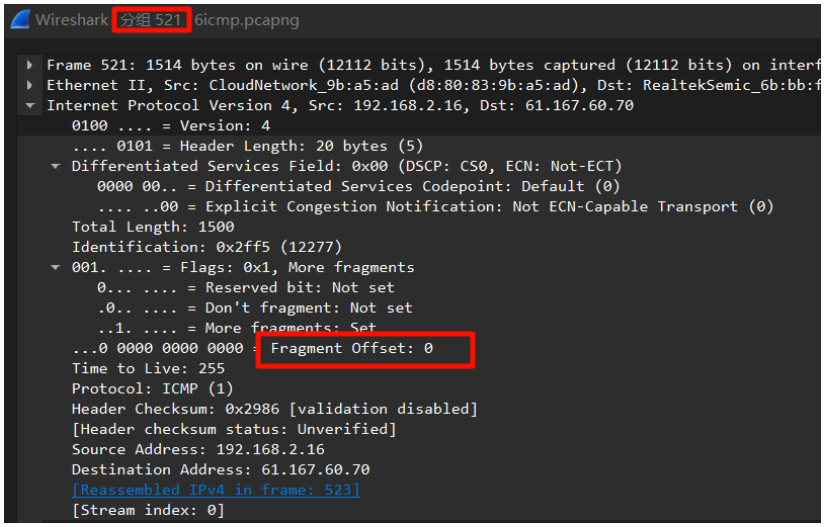
(五) 思考五

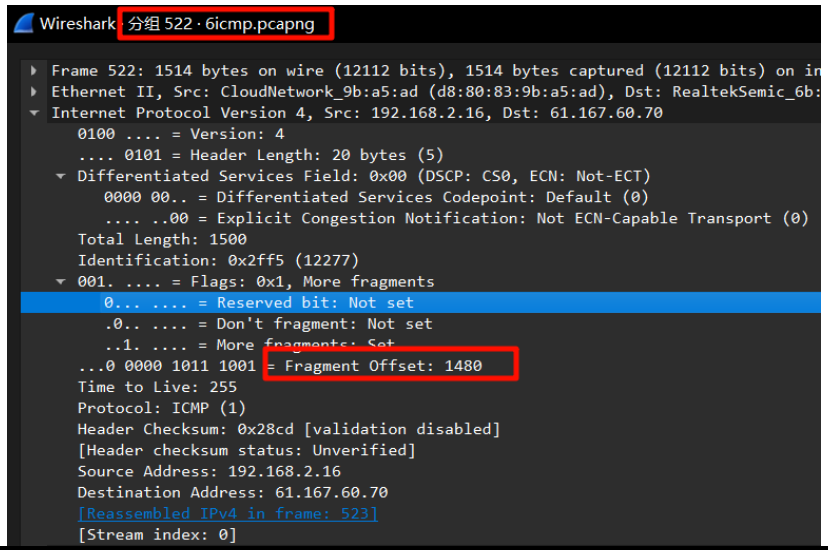
搜索到发送bao包大小为3500B的ICMP数据包：

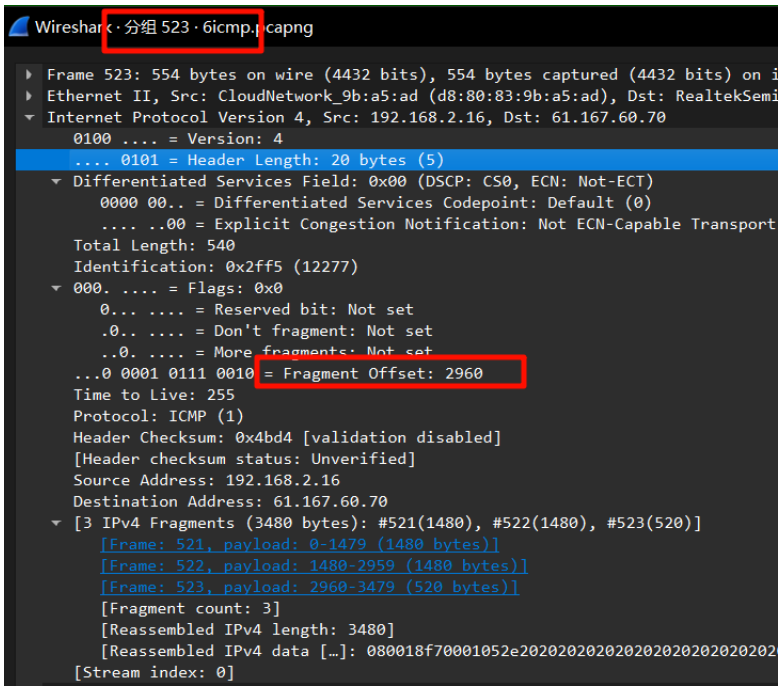


1） 首先从图中可以看出，这些数据包被分成了三片。

2） 分别查看三个分片中的IP数据包头部，







从中可以看出，这三个分片的一般字段与identification字段的值都相同，标志位、总长度和片偏移发生了变化。前两个分片的标志位为001（0x1），总长度为1500，第三个分片的标志位是000，总长度是540，三个分片的片偏移分别为0、1480和2960。

5. 抓取ARP数据包

（1） 输入命令 arp -a 查看当前主机的arp缓存，如下面两张图所示。ARP缓存的第一列是当前局域网设备的IP地址，第二列是上述IP地址对应的MAC地址，以16进制表示。第三列表示ARP缓存的类型，dynamic表示缓存项是通过 ARP 请求动态获取的，在一段时间后会过期，Static表示缓存项是通过静态配置添加的，不会自动过期。

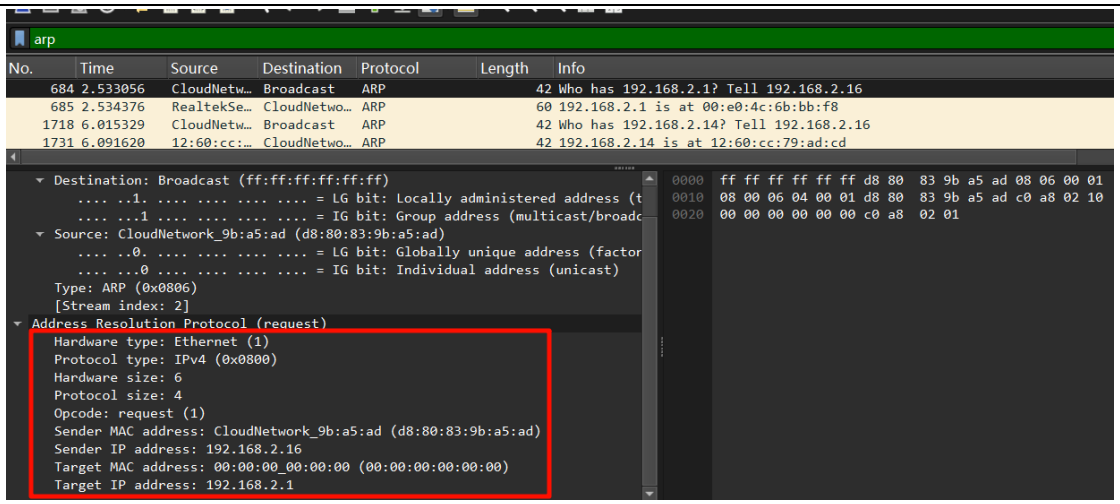
```
接口： 192.168.56.1 --- 0x18
  Internet 地址         物理地址              类型
  192.168.56.255        ff-ff-ff-ff-ff-ff     静态
  224.0.0.2             01-00-5e-00-00-02     静态
  224.0.0.22            01-00-5e-00-00-16     静态
  224.0.0.167           01-00-5e-00-00-a7     静态
  224.0.0.251           01-00-5e-00-00-fb     静态
  224.0.0.252           01-00-5e-00-00-fc     静态
  224.32.32.72          01-00-5e-20-20-48     静态
  239.255.255.250       01-00-5e-7f-ff-fa     静态
  239.255.255.251       01-00-5e-7f-ff-fb     静态

接口： 192.168.224.1 --- 0x29
  Internet 地址         物理地址              类型
  192.168.239.255       ff-ff-ff-ff-ff-ff     静态
  224.0.0.2             01-00-5e-00-00-02     静态
  224.0.0.22            01-00-5e-00-00-16     静态
  224.0.0.167           01-00-5e-00-00-a7     静态
  224.0.0.251           01-00-5e-00-00-fb     静态
  224.32.32.72          01-00-5e-20-20-48     静态
  239.255.255.250       01-00-5e-7f-ff-fa     静态
  239.255.255.251       01-00-5e-7f-ff-fb     静态
  255.255.255.255       ff-ff-ff-ff-ff-ff     静态

接口： 169.254.83.107 --- 0x4a
  Internet 地址         物理地址              类型
  224.0.0.22                                  静态
  224.0.0.251                                 静态
  239.255.255.250                             静态
```

```
接口： 192.168.2.16 --- 0xc
  Internet 地址         物理地址              类型
  192.168.2.1           00-e0-4c-6b-bb-f8     动态
  192.168.2.8           88-a4-c2-c0-c8-9e     动态
  192.168.2.9           24-cf-24-cf-75-91     动态
  192.168.2.20          76-4c-f4-ef-59-ff     动态
  192.168.2.24          70-32-17-cc-f7-31     动态
  192.168.2.25          d4-5d-64-36-bc-5d     动态
  192.168.2.255         ff-ff-ff-ff-ff-ff     静态
  224.0.0.2             01-00-5e-00-00-02     静态
  224.0.0.22            01-00-5e-00-00-16     静态
  224.0.0.167           01-00-5e-00-00-a7     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
  224.32.32.72          01-00-5e-20-20-48     静态
  239.255.255.250       01-00-5e-7f-ff-fa     静态
  239.255.255.251       01-00-5e-7f-ff-fb     静态
  255.255.255.255       ff-ff-ff-ff-ff-ff     静态
```

（2） 通过命令arp -d *删除本地的全部ARP缓存，通过对本地的192.168.2.14端口进行ping请求，并进行wireshark的抓包。



当前版本的Wireshark使用Ethernet II解析ARP包，Ethernet II的封装格式如图：

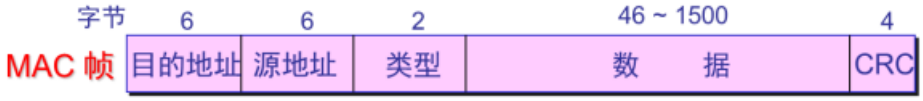
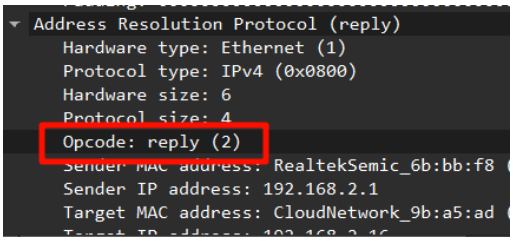


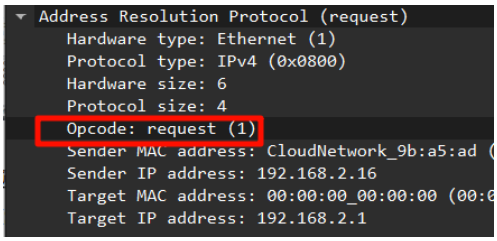
图 6-10 Ethernet II 的封装格式

以太网中的ARP请求和应答的分组格式如图：

1） ARP数据包的格式如上图所示，长度共28字节，由九部分组成，其中硬件类型部分占2字节，协议类型部分占2字节，硬件地址长度和协议地址长度分别占 1字节，OP字段占2字节，发送端MAC地址占6字节，发送端IP地址占4字节，目标MAC地址占6字节，目标IP地址占4字节。**和我们在wireshark抓包获取的长度完全一致。**

2） 通过OP字段来判断当前ARP数据包是请求包还是应答包，如下图所示，当OP字段为0x0001时表示为请求包，为0x0002时表示为应答包：



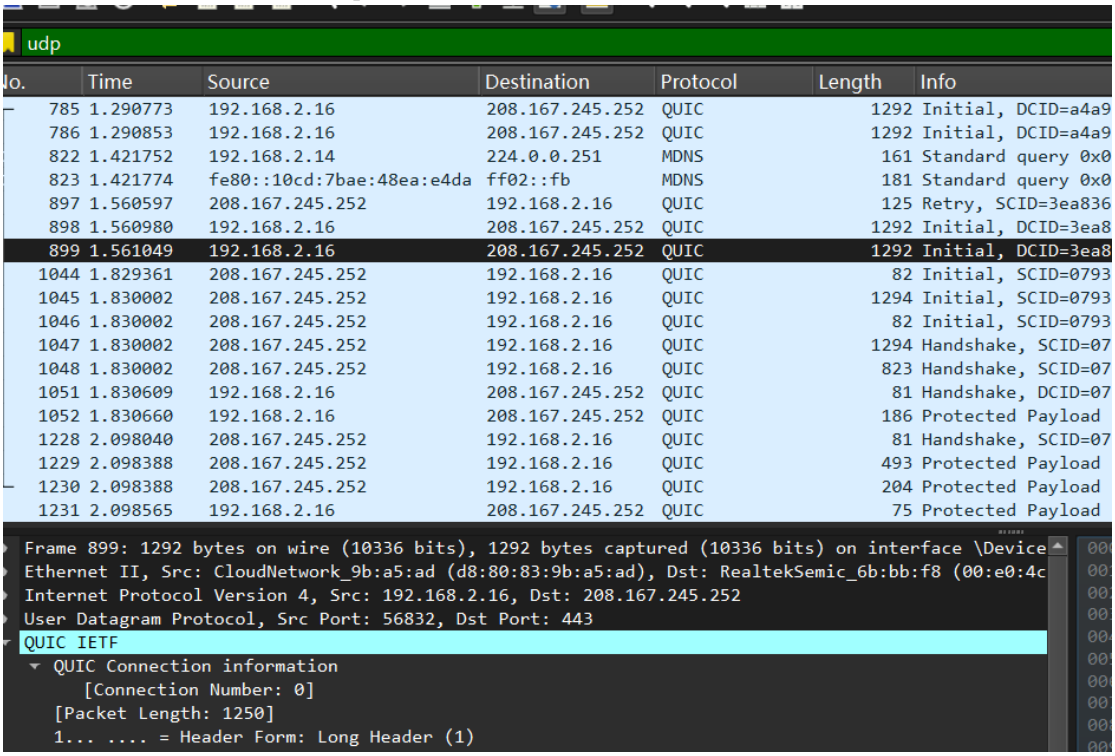


3） 因为ARP查询的时候，主机只知道目标的IP地址，不知道目标的MAC地址，只能通过广播MAC地址向所有的设备发送ARP请求，能够确保所有设备都可以接受到请求，当IP匹配的设备才会发出响应。而ARP响应的时候，由于ARP查询过程中包括了查询主机的IP地址和MAC地址，可以之间将应答确切的发送给主机，不需要进行广播过程。

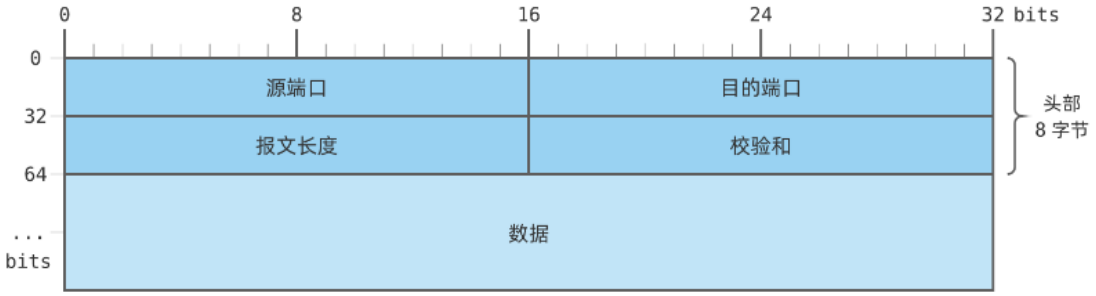
6. 抓取UDP数据包

很不幸!!! QQ9的协议更新为新的名为TQ的协议，这是一个基于TCP的协议，所以实验无法正常进行，我们选择其他方式进行实验。

我们借助QUIC协议进行测试，QUIC协议是一种由谷歌开发的新型网络传输协议，它基于UDP实现，提供了快速连接建立、多路复用、端到端加密、连接迁移等特性，旨在降低延迟、提高吞吐量和可靠性，以取代传统的TCP协议，目前已被选为HTTP/3的底层传输协议。通过访问https://http3check.net/about，可以通过wireshark抓去quic包。



1) Quic是基于udp协议的。
2) 本地主机ip地址为192.168.2.16，目的主机的ip是208.167.245.252.
3) 我的端口号是56832，目的主机端口号是443
4) UDP数据报的格式如下图所示：其中源端口号大小为2B，目的端口号大小为2B，UDP段长度大小为2B，校验和大小为2B。

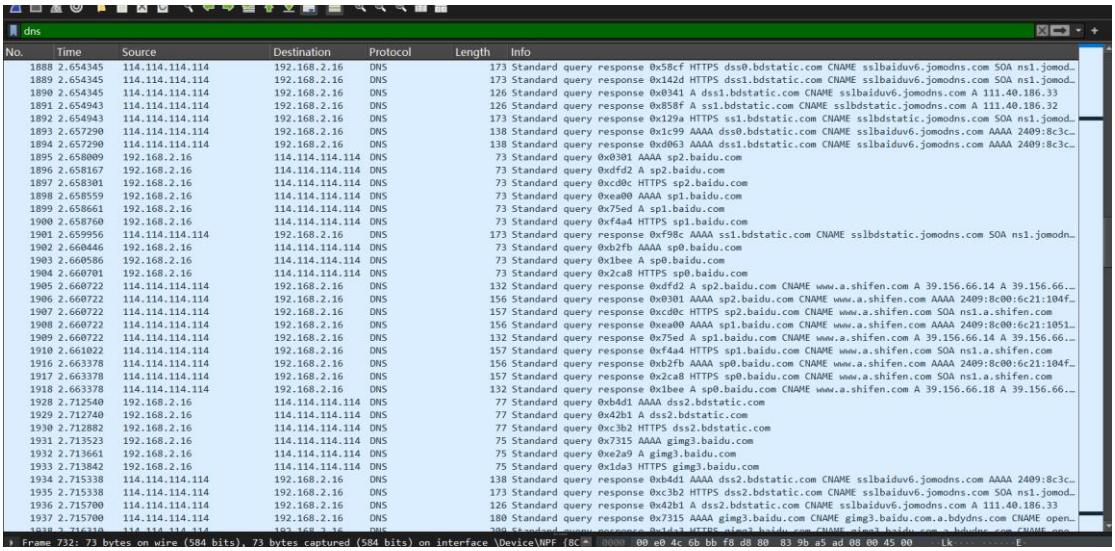


5) 和ICQ协议一致。QUIC协议在“发送一个ICQ数据包后，服务器又返回给主机一个 ICQ数据包”这种情况下也是一致的，这是因为在QUIC中，当客户端向服务器发送数据包时，服务器可以根据接收到的数据包进行响应。这是因为UDP提供的是不可靠的无连接的传输服务，客户端无法确认服务器是否接收到信息，因此需要一个QUIC报文表示收到（ACK机制）。可以看出UDP是物链接的，与TCP需要进行三次握手以建立连接不同，UDP不需要建立连接，数据包可以直接发送。但QUIC提供了连接管理尽管底层使用UDP。在QUIC中，连接是有状态的，客户端和服务器之间通

过特定的连接ID保持状态，能够识别和管理不同的流和数据包。这使得QUIC具备了某种程度上的连接性，同时保留了UDP的高效性和低延迟。

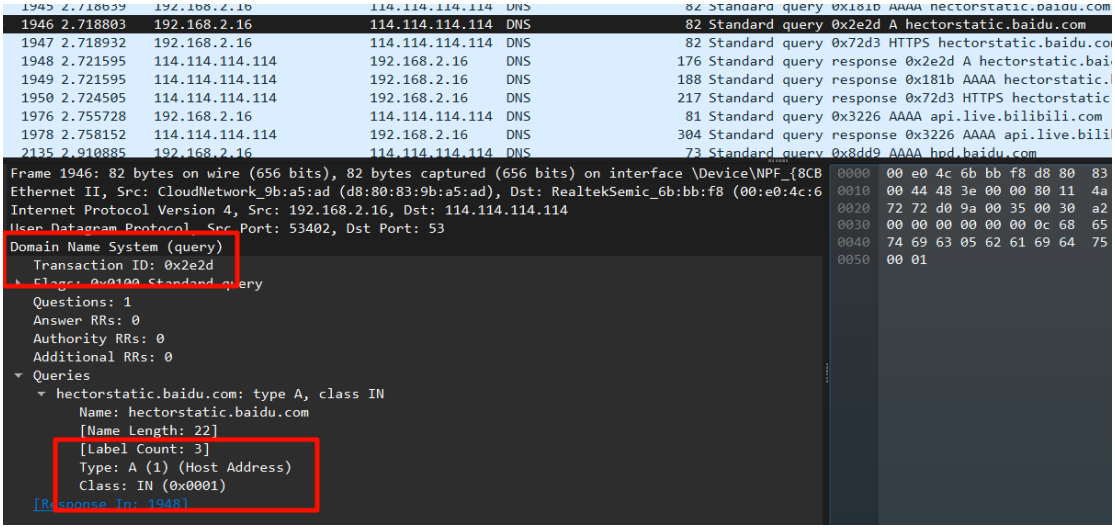
7. 利用WireShark进行DNS协议分析

按照实验指导书的要求，访问baidu.com并利用wireshark进行抓包，与DNS有关的结果如下：



主机想要访问域名时，首先向本地域名服务器查询是否存储该域名对应IP的映射，若本地域名服务器中没有缓存，则由本地域名服务器继续查询，分为递归和迭代两种方式，本地域名服务器首先查询根域名服务器，由根域名服务器返回顶级域名服务器的IP地址，再继续查询。

其中，查询baidu.com的DNS报文详细信息如下：

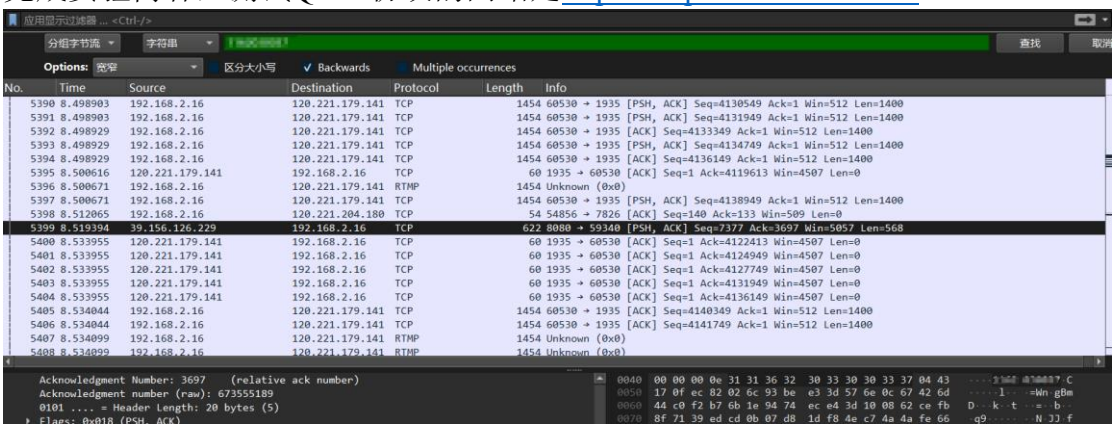


由上可知，本机IP是192.168.2.16，目的IP是114.114.114.114，源端口号是53402，目的端口号是53。并且由信息可知，DNS协议利用的是无连接不可靠的UDP服务，只标识了端口号、校验和等少量信息，不传输更多的控制信息。DNS通过TransactionID来标识查询和响应报文，并且同一查询对应的响应报文ID是相同的序号。对应的响应报文如上图所示，其中还标注了记录类型type为A，class为IN：

问题讨论：

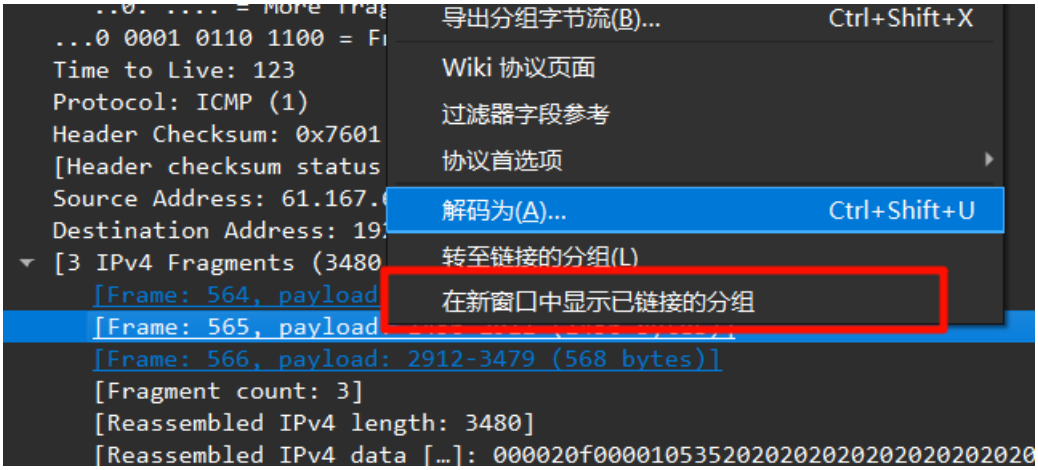
（1） http://hitgs.hit.edu.cn/news网址已经迁移，访问今日哈工大http://today.hit.edu.cn

（2） 新版qq更改了消息传输协议，原有的udp+oicq的过滤方式无法找到，我通过qq号查找字节流获得了数据包，发现为tcp协议，qq协议已经更改为TCP，使用谷歌的QUIC协议完成实验内容，测试QUIC协议的网站是https://http3check.net/about。



（3） 新版pingplotter在实现traceroute程序功能时，具体实现方法与实验指导书上不同，设置packet size的路径是edit>options>engine>packet size，通过修改此处并且重新点击页面中的绿色开始按钮，就可以发送设定长度大小的一系列数据包。

（4） IP分析时无法查看到more fragment被设置为1的情况。需要单独通过对对应的分片在独立窗口进行打开，才能查看到不同的偏移和MF标志为1。



心得体会：

结合实验过程和结果给出实验的体会和收获。