哈尔滨工业大学

**计算机网络**

**实验报告**

**(2019年度秋季学期)**

|  |  |
| --- | --- |
| **姓名：** | **史纪元** |
| **学号：** | **1173300919** |
| **学院：** | **计算机科学与技术学院** |
| **教师：** | **刘亚维** |

## 实验二 可靠数据传输协议-GBN 协议的设计与实现

## 一、实验目的

熟悉并掌握 Wireshark 的基本操作，了解网络协议实体间进行交互以及报文交换的情况。

## 二、实验内容

1) 学习Wireshark的使用

2) 利用Wireshark分析HTTP协议

3) 利用Wireshark分析TCP协议

4) 利用Wireshark分析IP协议

5) 利用Wireshark分析Ethernet数据帧

选做内容：

a) 利用Wireshark分析DNS协议

b) 利用Wireshark分析UDP协议

c) 利用Wireshark分析ARP协议

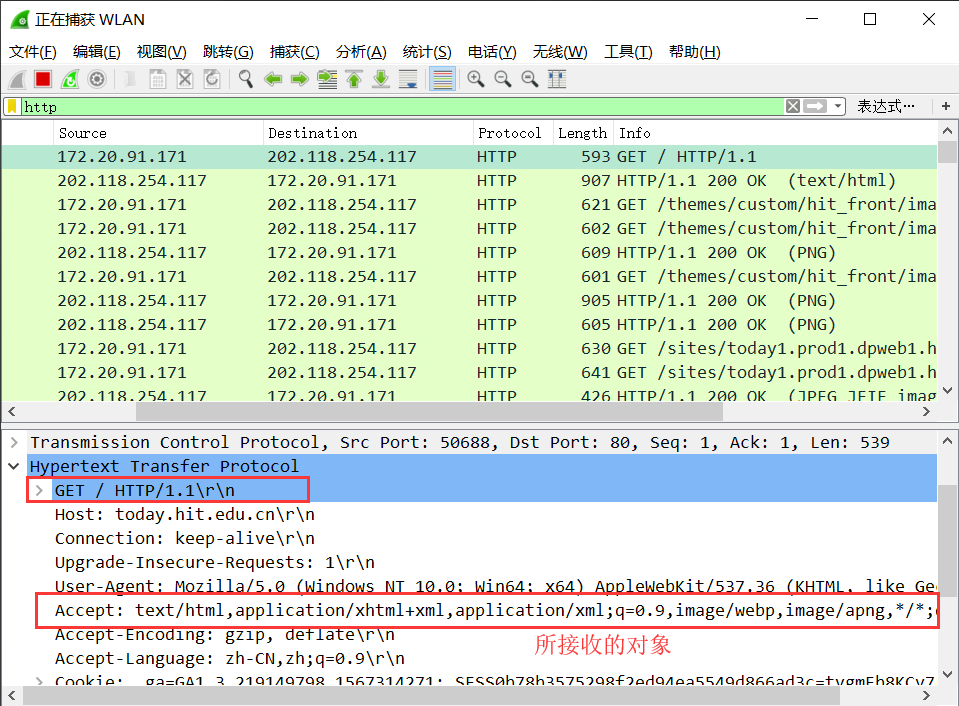
## 三、实验过程及结果

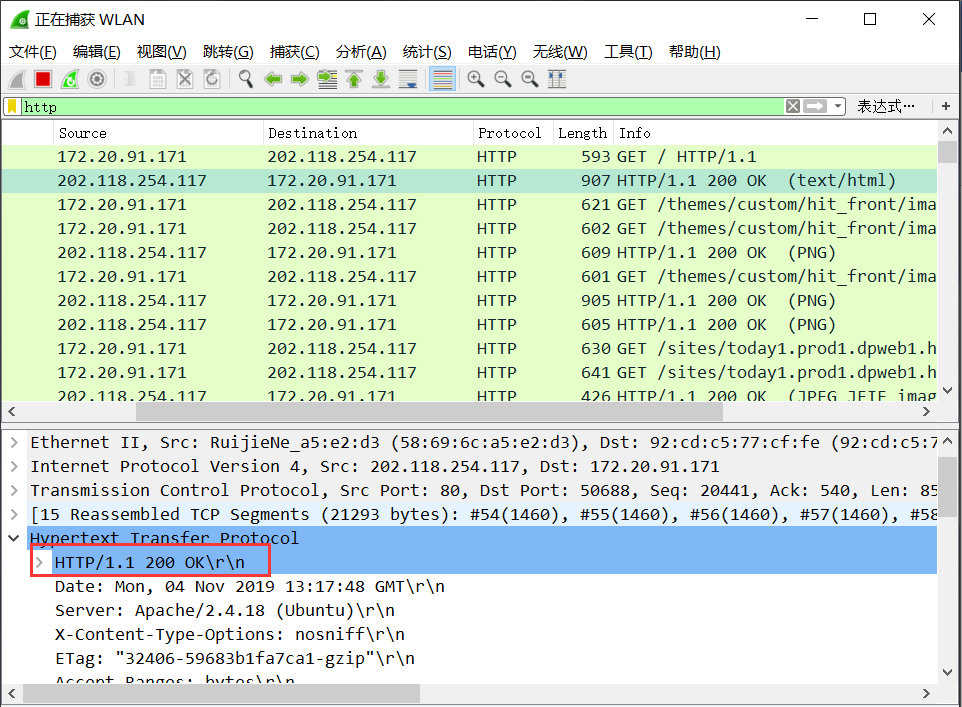
**（一）HTTP分析**

**1）HTTP GET/response 交互**

打开Wireshark分组俘获、并用web浏览器访问today.hit.eud.cn，在显示过滤说明处输入“http”，查看俘获到的HTTP报文

**思考问题**





首先找到俘获的第一个报文，即本机发出的请求报文，可以看到我的浏览器运行的HTTP协议是HTTP1.1，而从响应报文可以看到服务器运行的HTTP协议版本号也是HTTP1.1。

上图中标出了我的浏览器相服务器指出的它能接收的对象的语言版本为：

Accept:text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,\*/\*;q=0.8,application/signed-exchange;v=b3\r\n

在俘获分组列表中可以看到我的计算机的IP地址为172.20.91.171，服务器的IP地址为202.118.254.117。

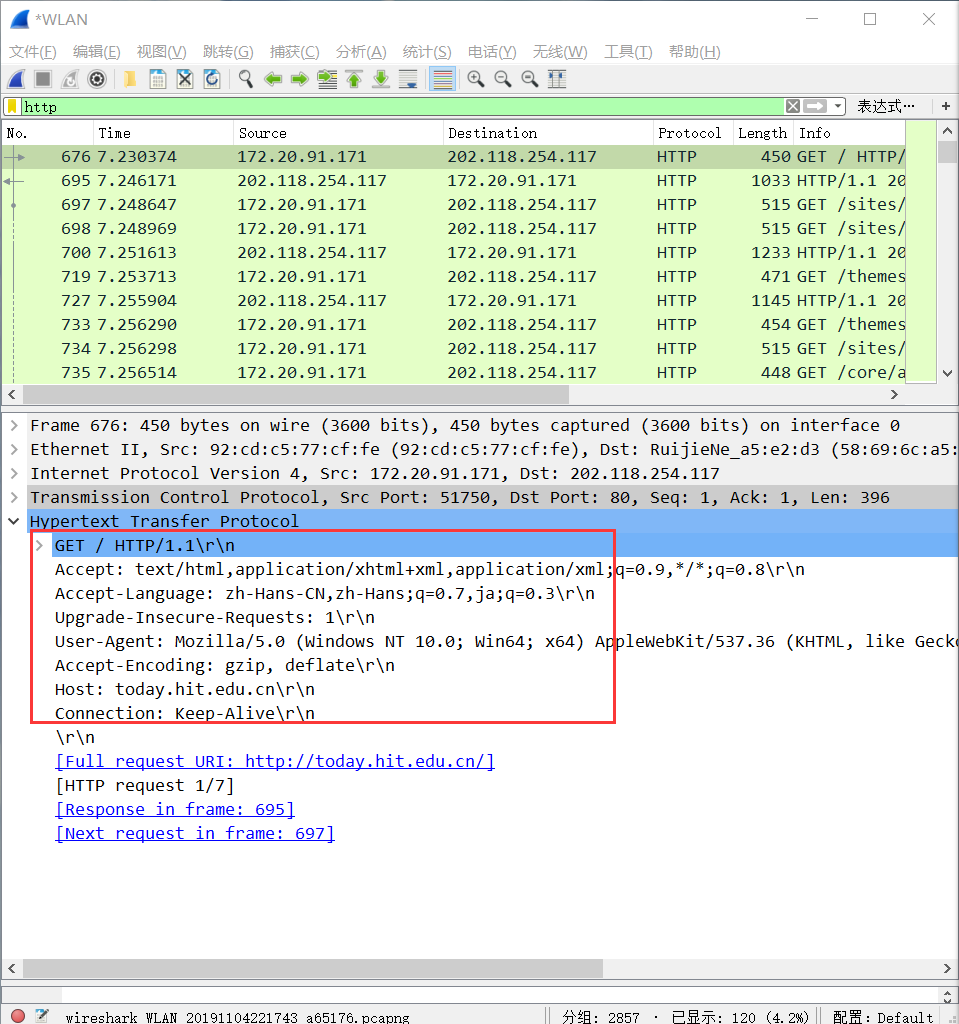
服务器向我的浏览器返回的状态代码为200。

**2）HTTP 条件GET/response 交互**

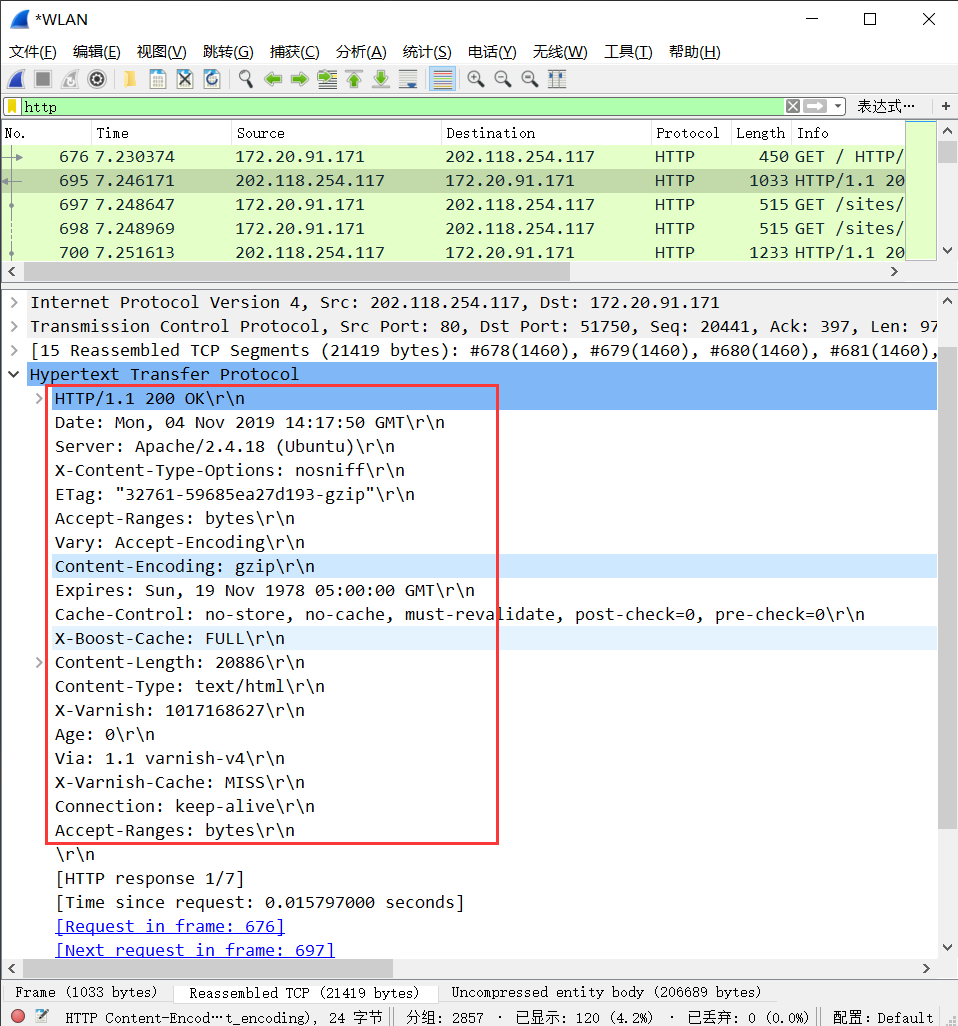
先清空浏览器的缓存，然后按1）的步骤进行HTTP报文的捕获。先输入http://today.hit.edu.cn，再点击刷新按钮。

**思考问题**

找到我的浏览器向服务器发出的第一个HTTP GET请求



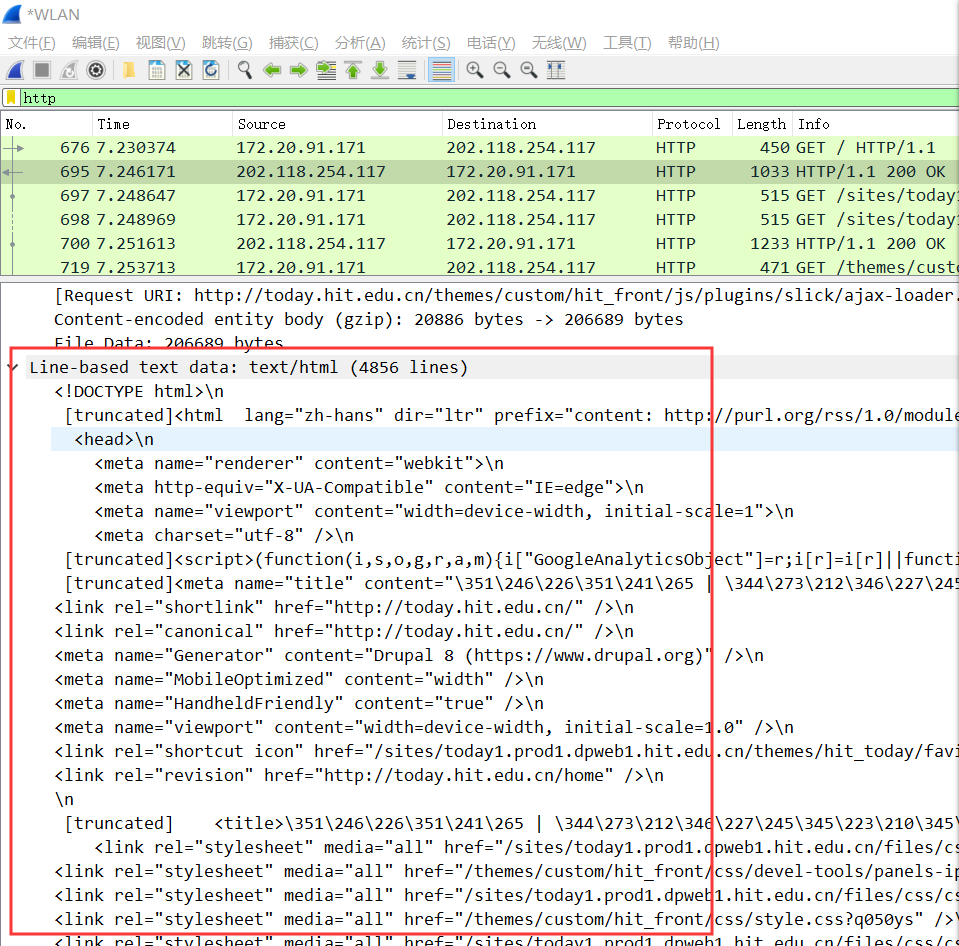
可以看到第一次发出的报文中有以下信息：请求方法为GET，使用的协议号为HTTP1.1，请求的主机为today.hit.edu.cn，请求建立的连接为keep-alive， User-Agent为用户使用的浏览器及版本等信息，Accept后为请求接收的对象类型、编码以及语言。其中没有一行是If-Modified-Since。



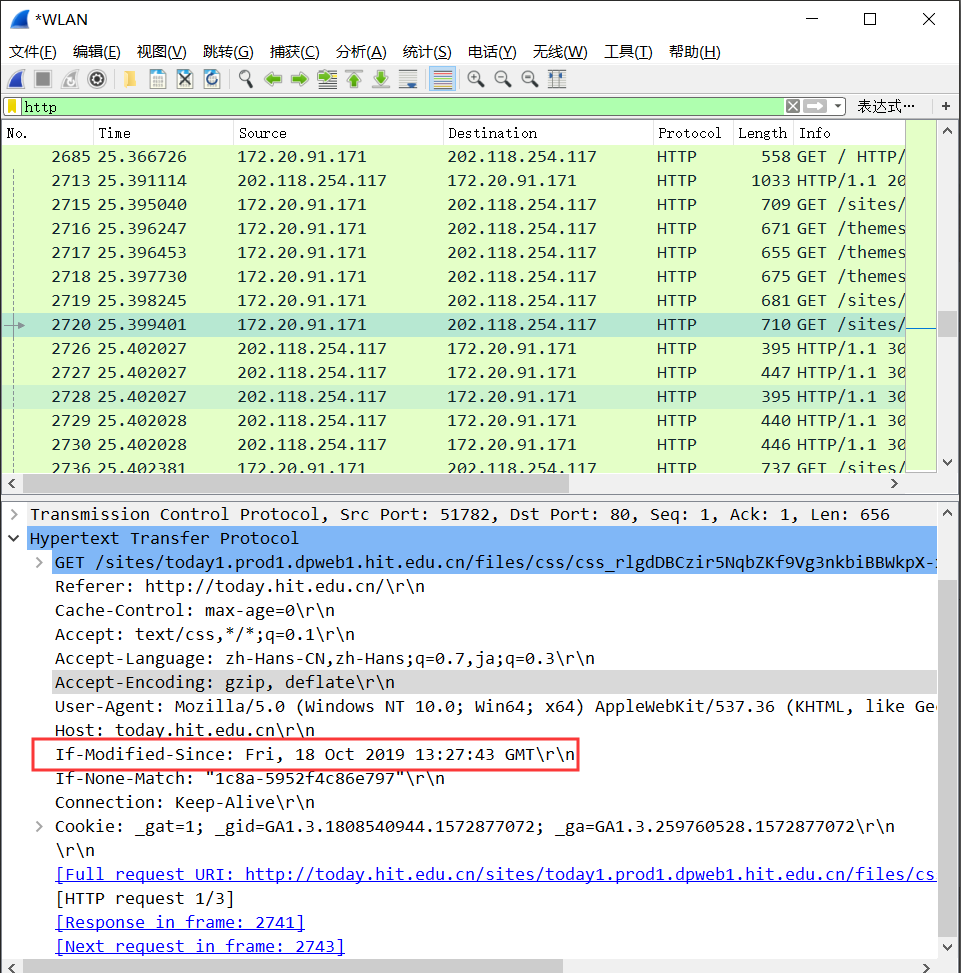
服务器第一次返回的请求报文如上图所示。可以看到服务器使用的HTTP协议版本也是HTTP1.1，接下来对HTTP响应报文的各字段进行分析

* Date表示消息发送的时间，
* Server: Apache/2.4.18(Ubuntu)表示服务器名称
* Accept-Ranges字段说明服务器通过byte serving支持的部分内容范围类型
* Vary: Accept-Encoding意为服务器会按照客户端发来的请求头的Accept-Encoding字段来对请求进行不同的服务
* X-Content-Type-Options: nosniff，这是一种安全功能，有助于防止基于 MIME 类型混淆的攻击（例如图片中嵌入代码）
* ETag：特定版本资源的标识符，通常是消息摘要
* Conteng-Encoding：gzip意为服务器设置数据使用的编码类型为gzip
* Expires设置响应体的过期时间，即该资源的过期时间，通常与Cache-Control一起使用
* Cache-Control——缓存控制，其中no-store意思是所有内容都不会被保存到缓存或Internet临时文件中；no-cache为不直接使用缓存，要向原服务器发起（新鲜度校验）请求；must-revalidate意为当前资源一定是向原服务器发去验证请求的，若请求失败会返回504；pre-check=0即在这段时间间隔之后，一个对象在显示给用户之前应被选中进行更新；post-check=0即在这段时间之后，在显示给用户之前，该对象被选中进行更新。
* Content-Length和Content-Type说明了网站返回内容的长度和类型
* Age：当代理服务器用自己缓存的实体去响应请求时，用该头部表明该实体从产生到现在经过多长时间了。如果某对象的age+Date与主请求中的Date一致，则缓存命中
* Contection: keep-alive让HTTP保持连接状态

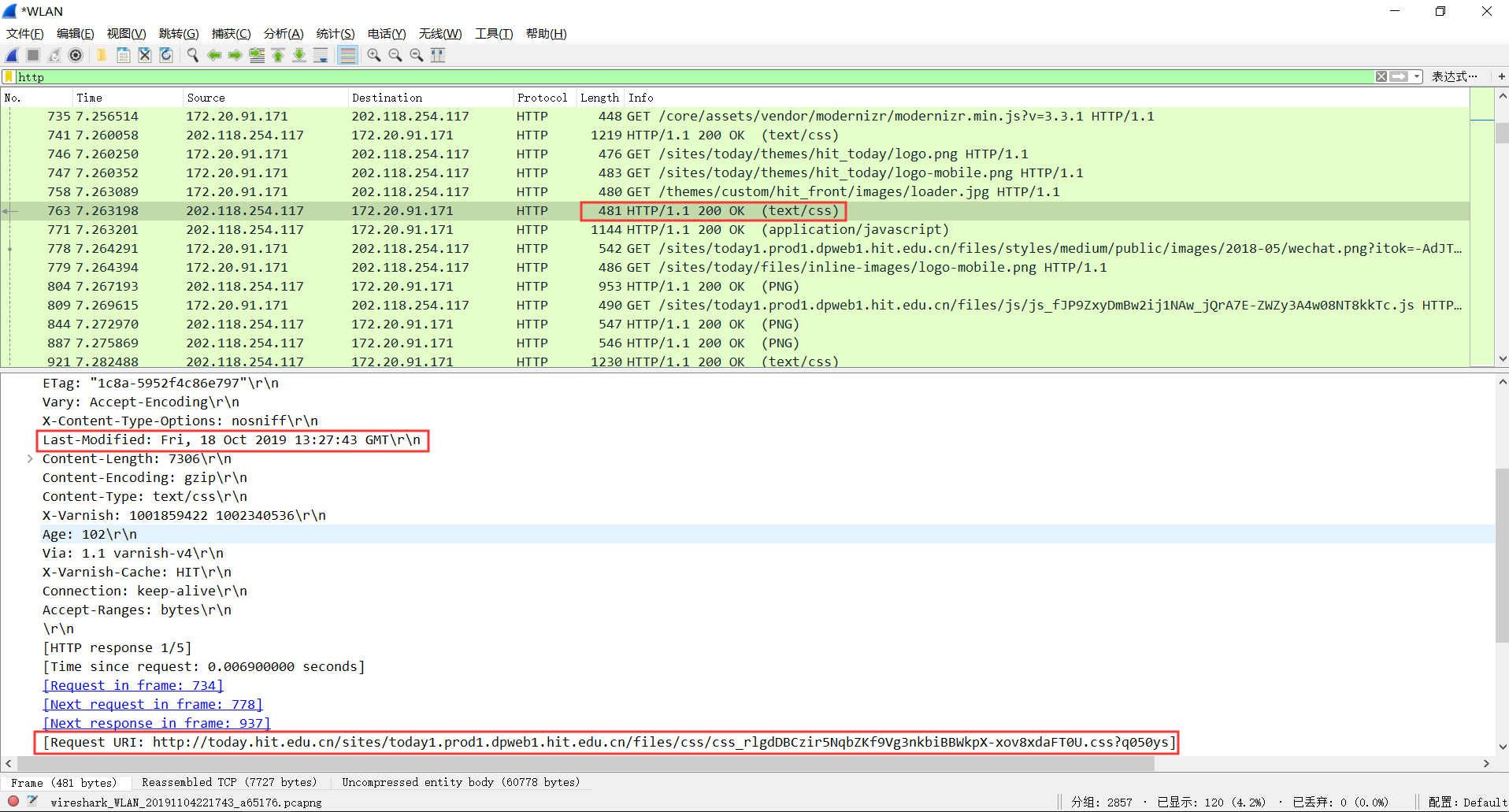
服务器明确返回了文件的内容，从Content-Length和Content-Type可以得知，并且在Wireshark中可以看到HTTP响应内容的文本格式：



再看刷新后的请求报文，可以发现其中多了一行If-Modified-Since字段：

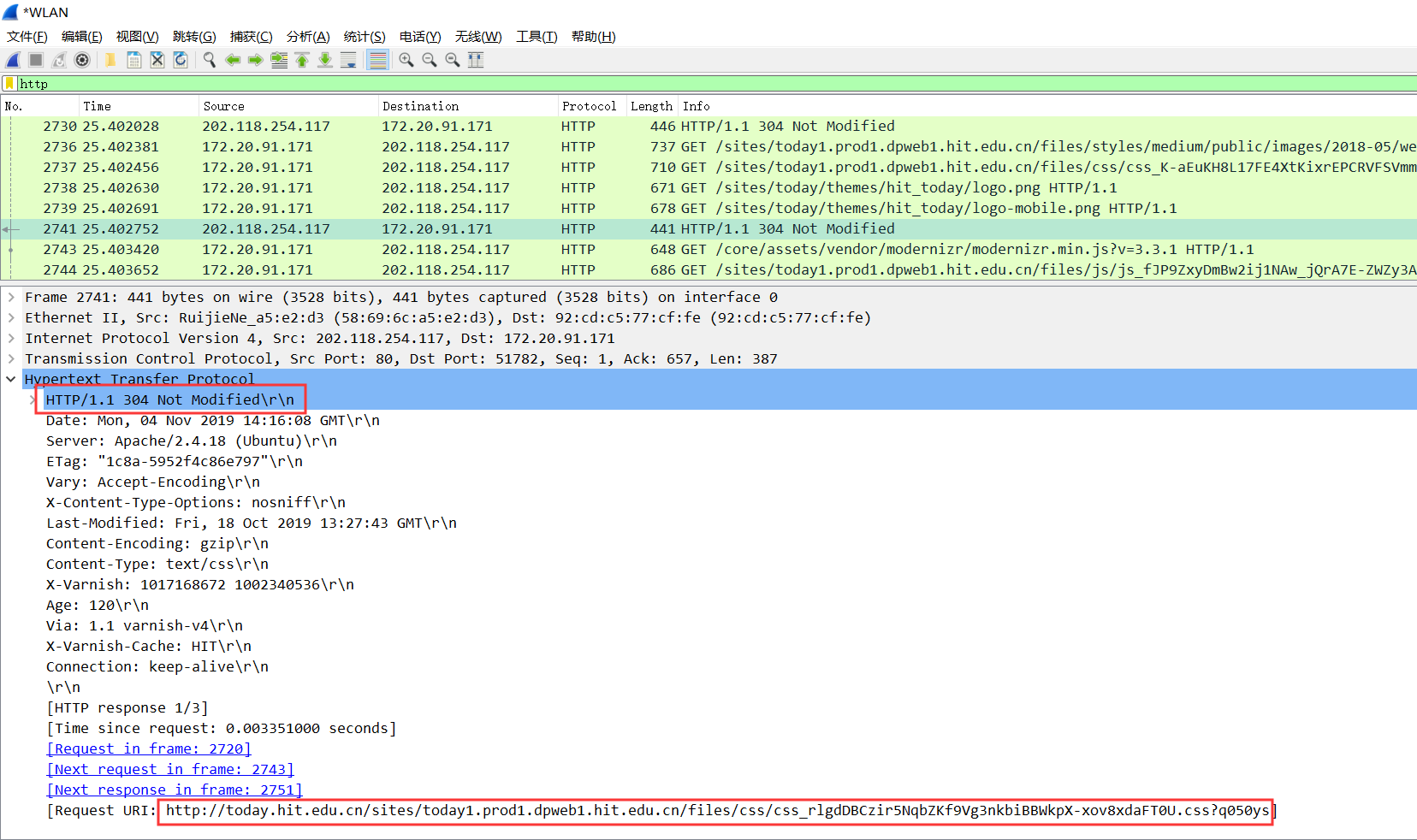


可以看到该字段后面跟了一个时间，找到之前第一次请求该对象时服务器给出的响应报文：



可以看出，我的浏览器发出第二次请求的If-Modified-Since字段后的时间就等于服务器第一次返回的响应报文中Last-Modified字段后的时间。Last-Modified说明了该对象最后一次更新的时间，第一次请求该对象使得我的浏览器缓存下了该对象。当浏览器再次请求该对象时，会向服务器询问在这个时间之后该对象有没有被修改，以此决定是否从缓存中读取内容。

再找到浏览器响第二次请求时的响应报文：

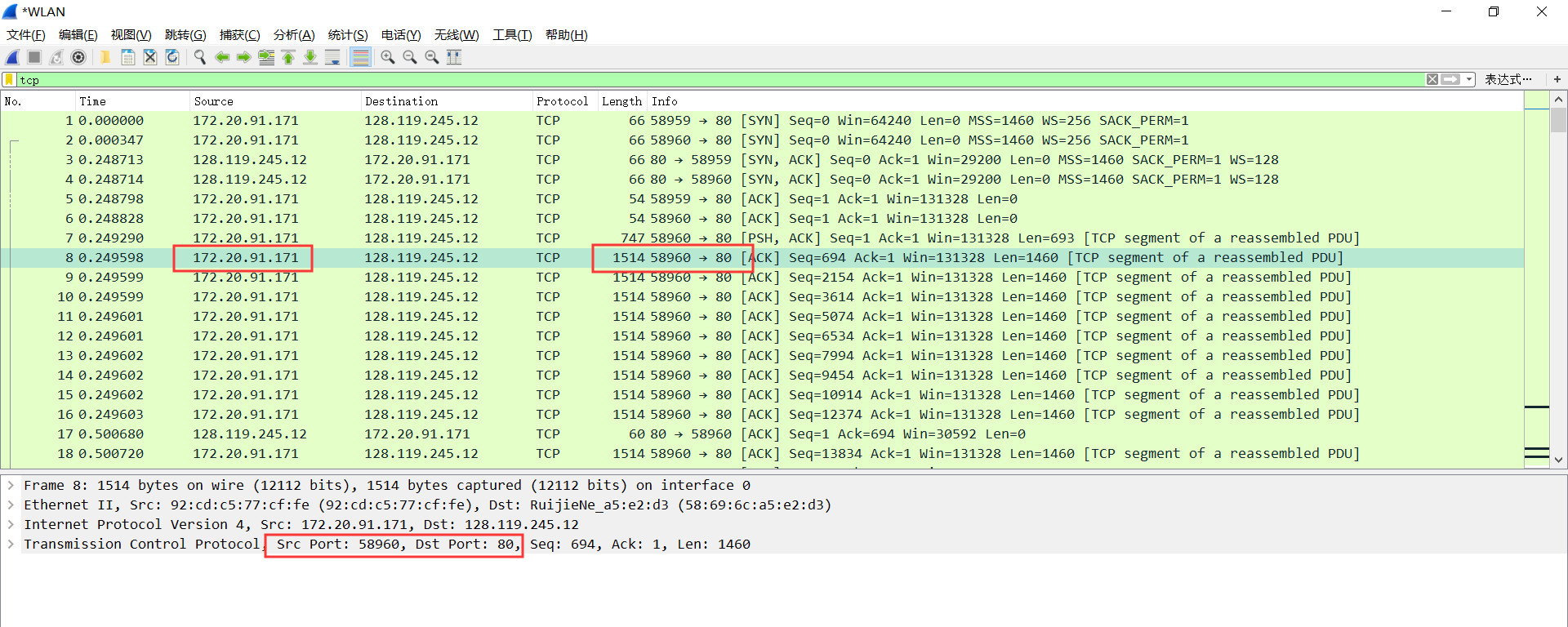


可以看到HTTP状态代码为304，意为没有更新该对象，这时服务器就没有再返回文件的内容（HTTP报文中没有该内容），而是浏览器从缓存中读取内容。

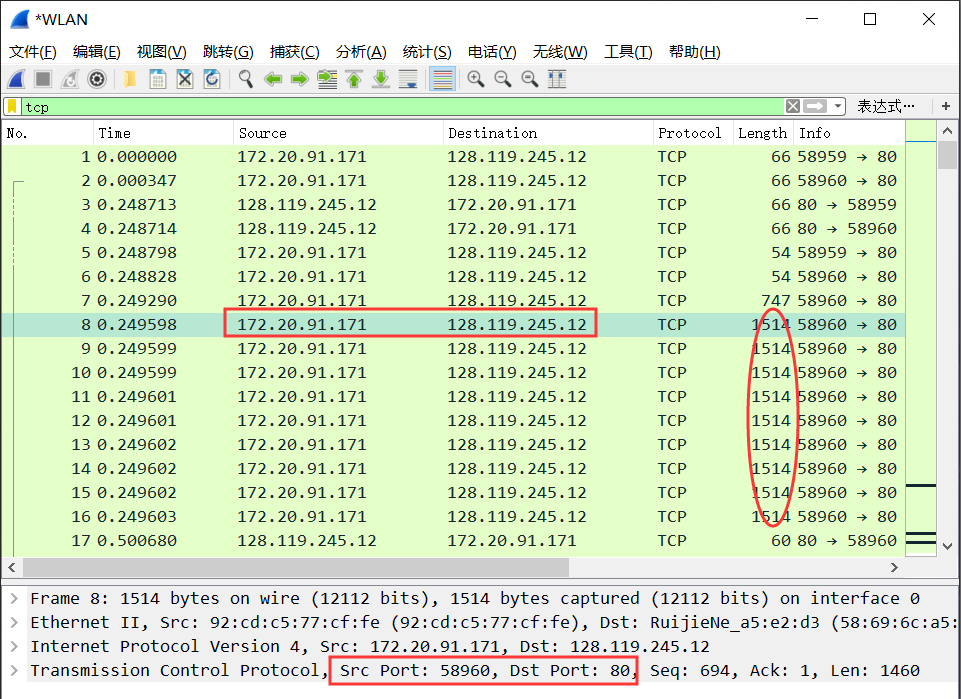
**（二）TCP分析**

先启动浏览器，打开http://gaia.cs.umass.edu/wireshark-labs/alice.txt 网页，获得alice.txt文本，然后打开http://gaia.cs.umass.edu/Wireshark-labs/TCP-Wireshark

-file1.html，打开本地的alice.txt文件，启动Wireshark开始捕获，回到浏览器点击Upload，停止捕获。在筛选规则中输入tcp，得到一系列tcp报文：



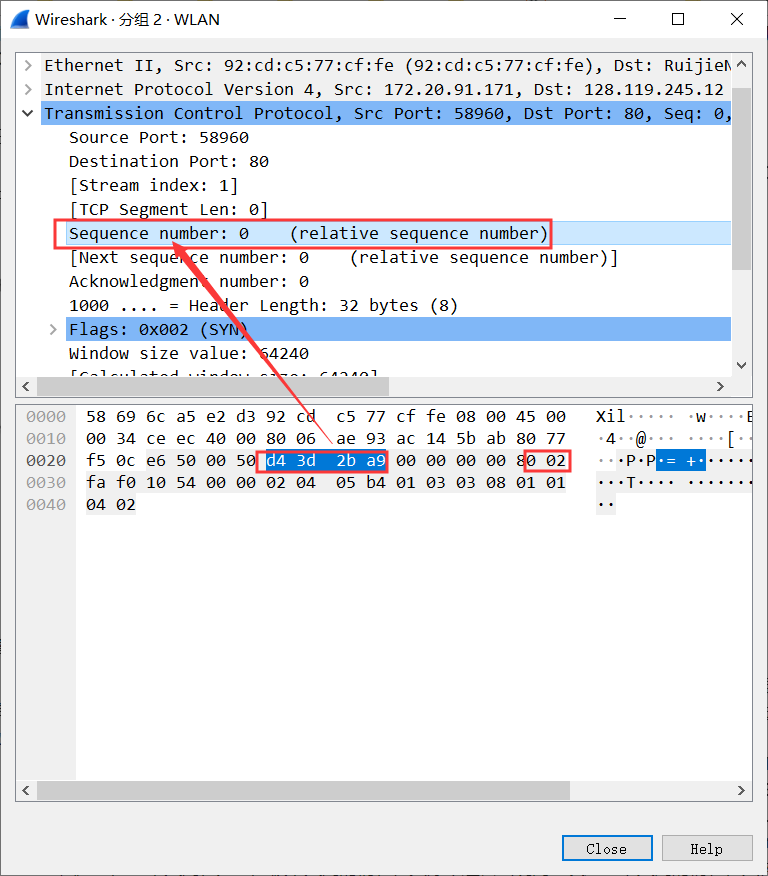
**思考问题**



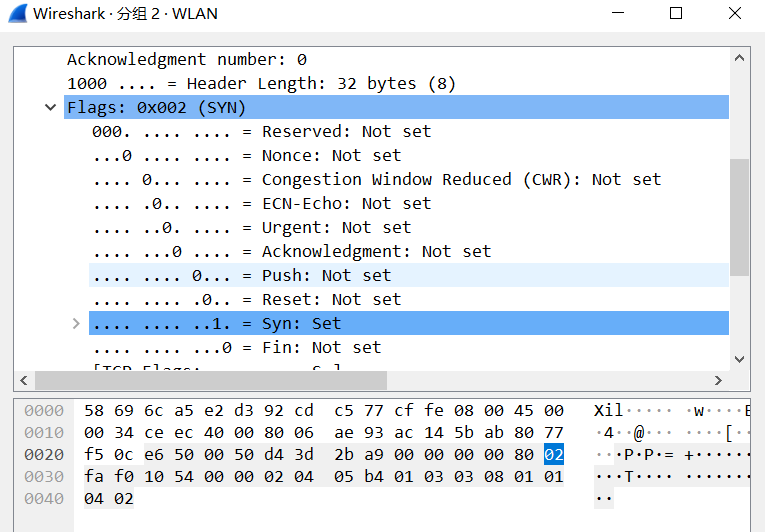
**1.**从上图可以得知，向gaia.cs.umass.edu服务器传送文件的客户端主机的IP地址为172.20.91.171，TCP端口号为58960，因为此时客户端主机向服务器发送大量TCP报文使用的是58960端口。

**2.**而gaia.cs.umass.edu服务器的IP地址为128.119.245.12，它用来发送和接收TCP报文的端口号是80。

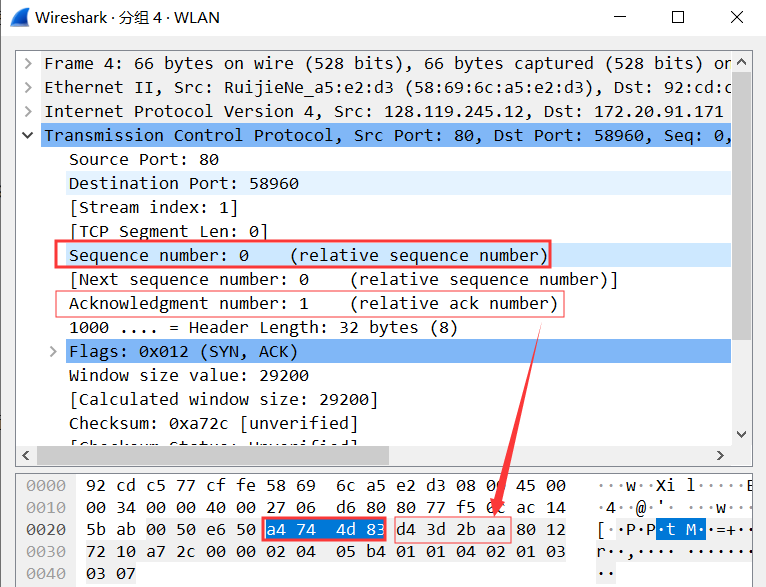
**3.**双击查看客户端发的第一条TCP报文，此时客户端请求建立TCP连接。



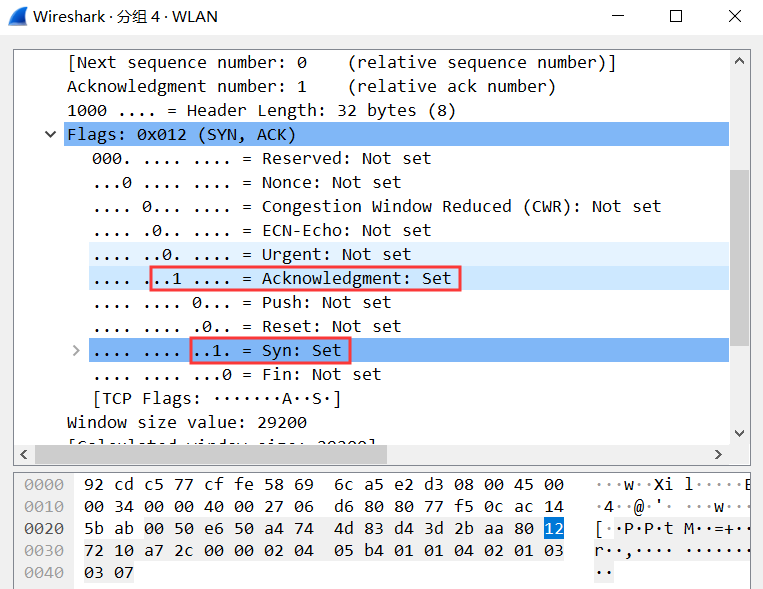
从图中可以看出，该报文段的序号为0xd43d2ba9，此为随机生成的序号，Wireshark显示相对序号为0。而该报文段的标志位说明了该报文段是SYN报文段。



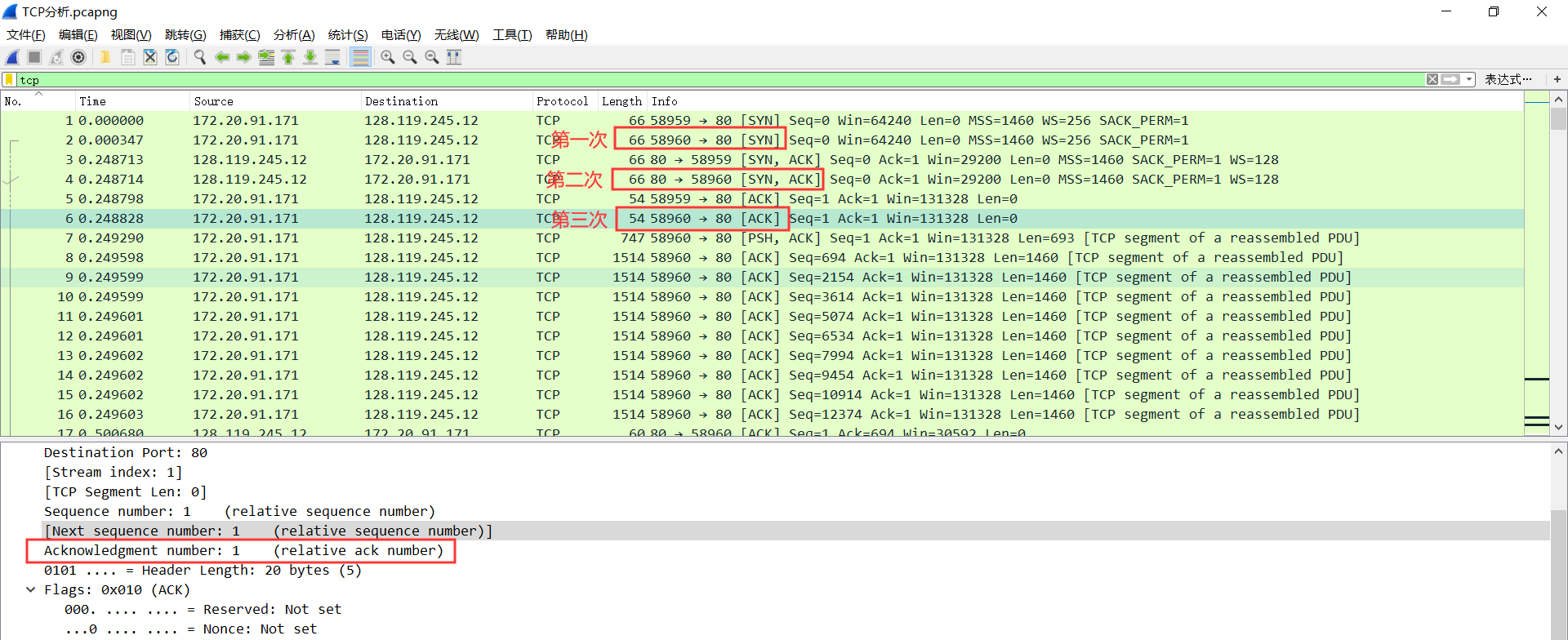
**4.**再查看服务器返回的TCP报文段：



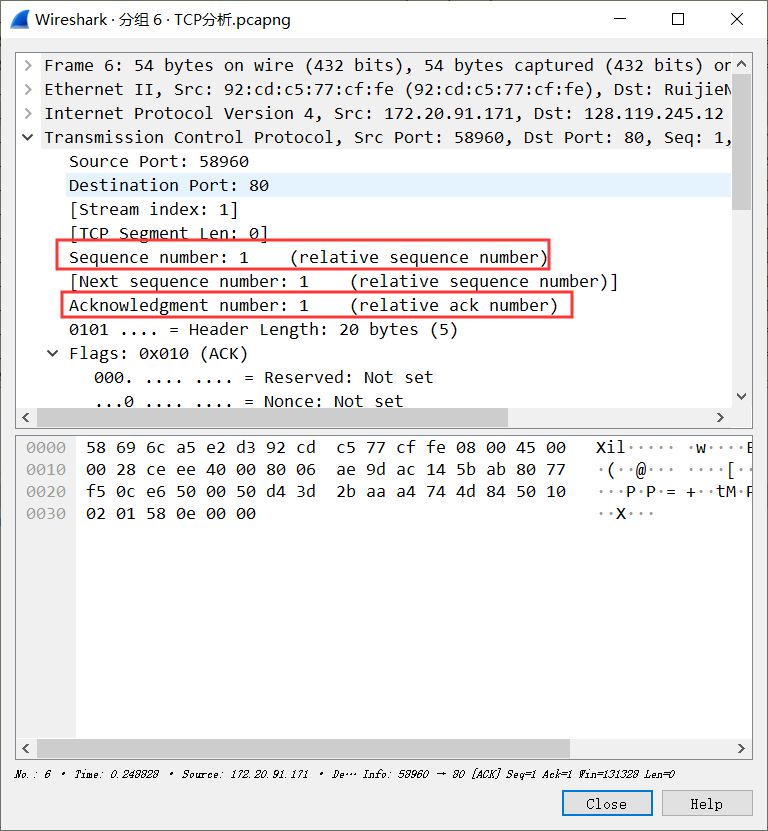
可以看到SYNACK报文段的序号为0xa4744d83，这也是随机生成的数，Wireshark同样也显示了它的相对序号为0。而该报文段中Acknowledgement字段的值为0xd43d2baa = 0xd43d2ba9 + 1，由于服务器收到了客户端发出的请求建立TCP连接的SYN报文，故向客户端发送了SYNACK报文，ACK的序号就是刚才SYN报文段的序号+1。该报文段用标志位中的两位标示了该报文段是SYNACK报文段：



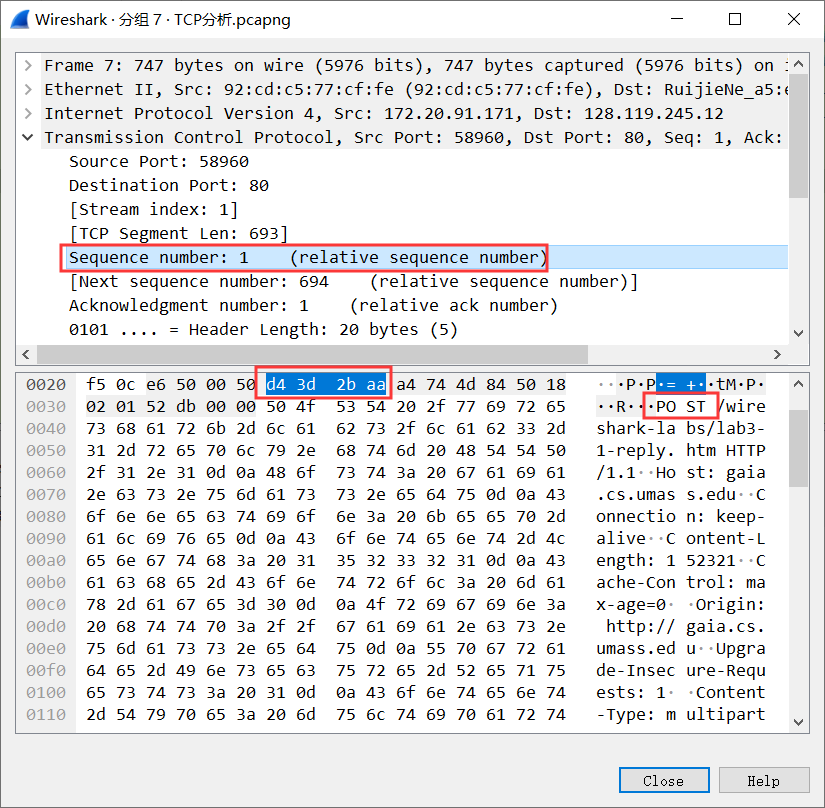
**5.**对于TCP的三次握手可以从下图看出：



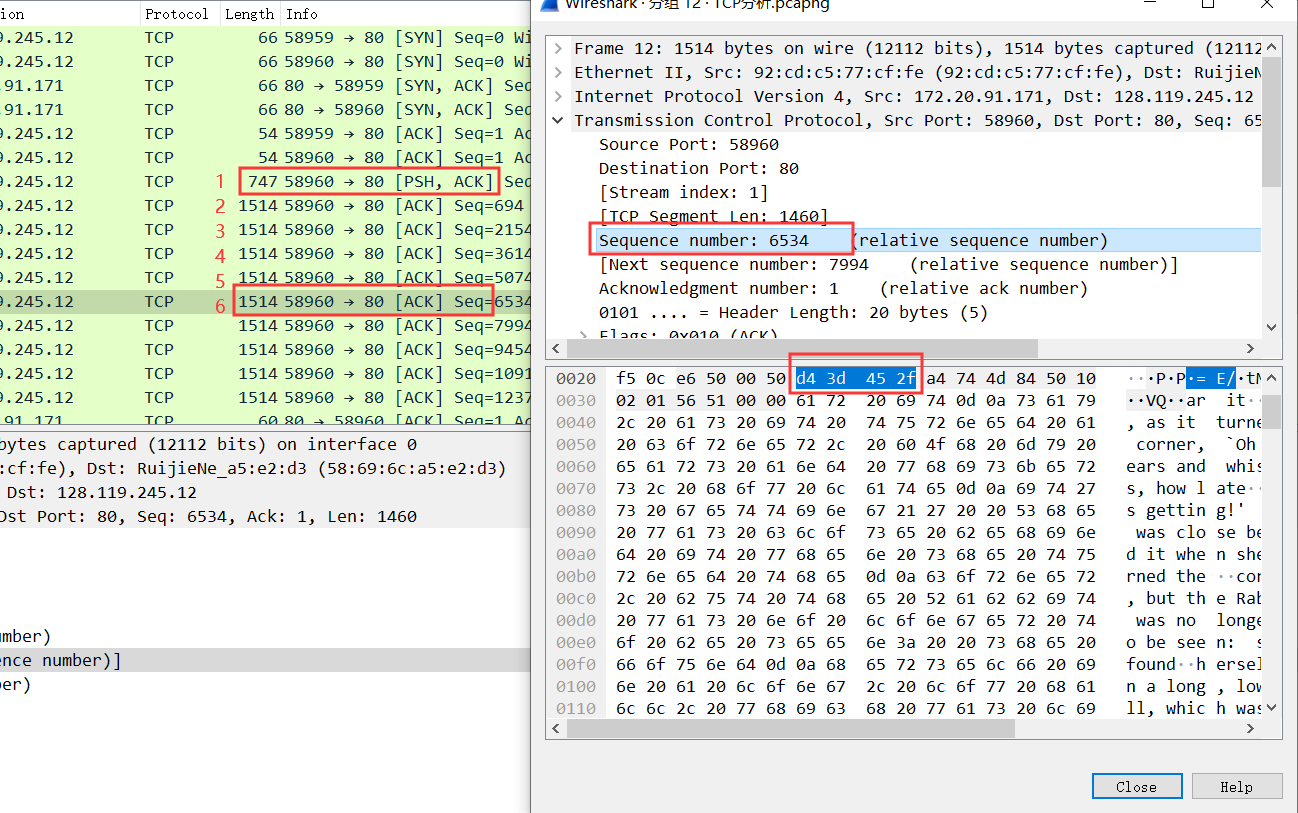
其中客户端向服务器发送SYN报文段请求建立连接为**第一次握手**，然后服务器返回SYNACK报文段是**第二次握手**，查看图上标出的第三个报文段：



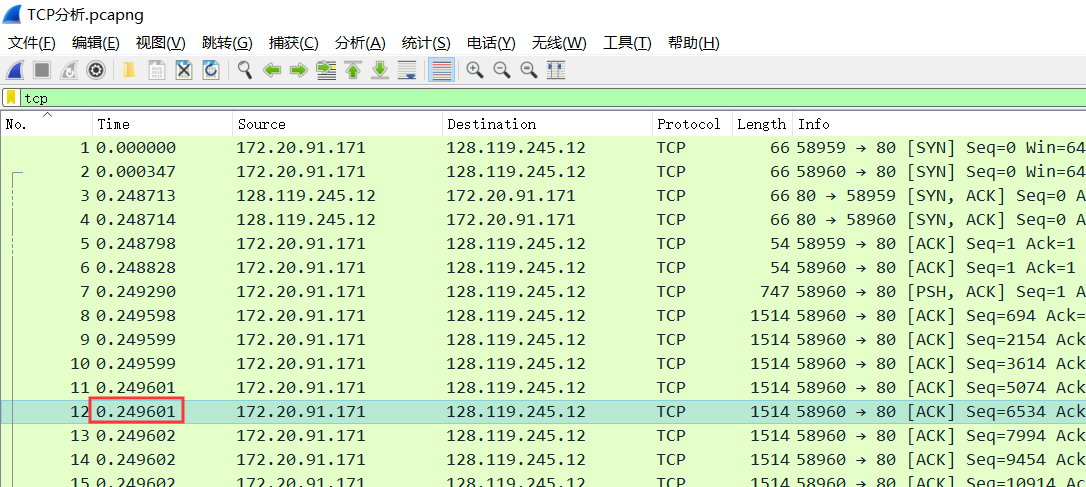
可以看到该报文段的序号为1，ACK序号也为1，可知这个报文段是对服务器发送的SYNACK报文段进行的确认，即**第三次握手**



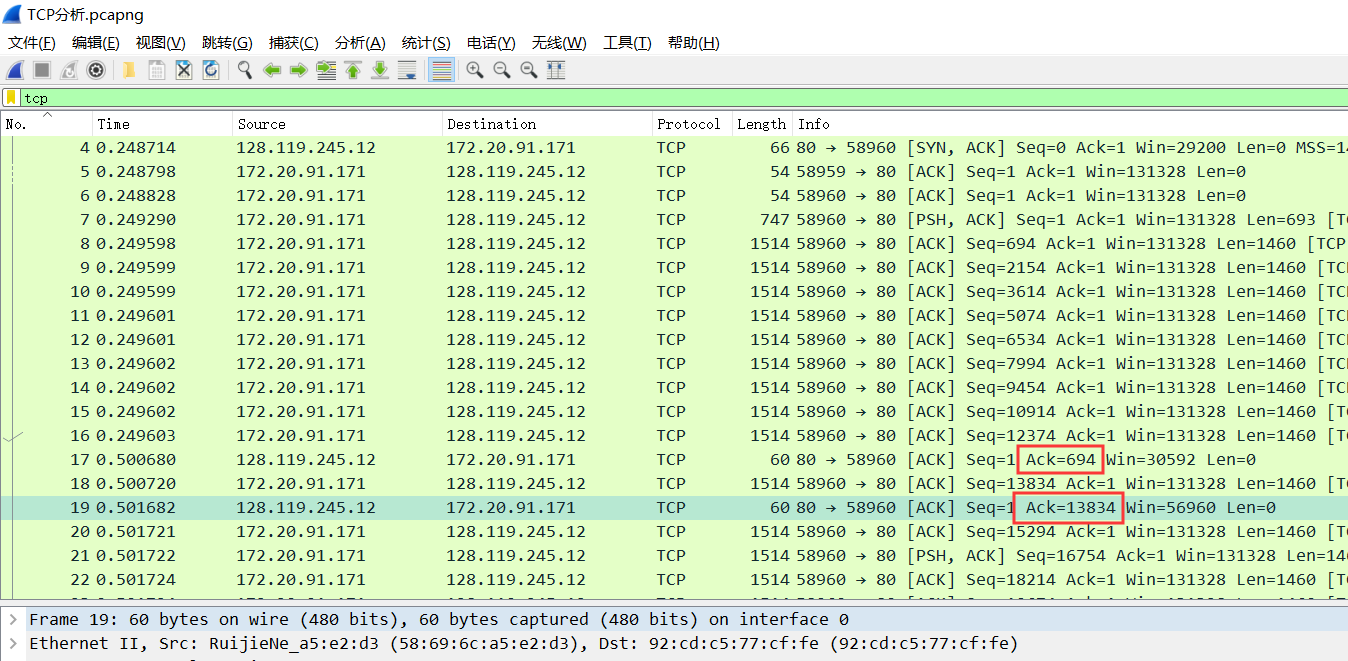
**6.**上图的报文端包含HTTP POST命令，可以看到该报文段的序号为0xd43d2baa，相对序号为1（第一次发送的报文段的相对序号为0）



**7.**从上图可以看出，该TCP连接上的第六个报文段的序号是0xd43d452f，相对序号为6534,



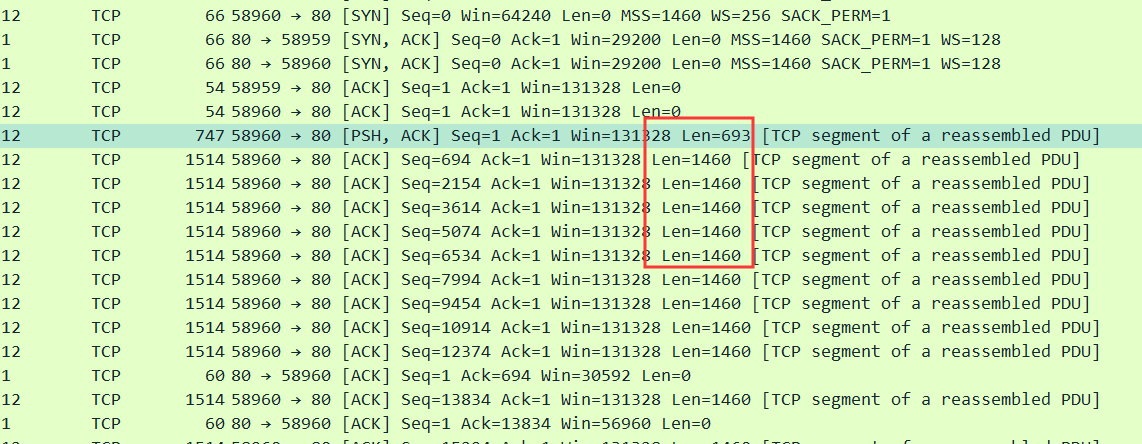
**约为**第一个SYN报文发出后**0.249**秒之后发出的。



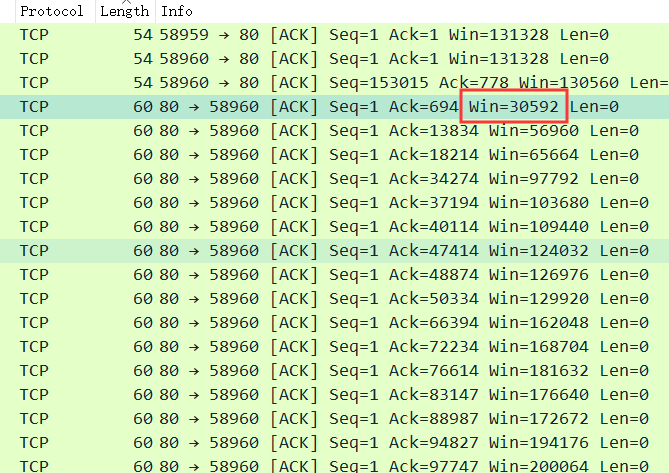
从之前的分析可以得知该报文段的下一个报文段的序号为7994，但由于TCP的累计确认机制，服务器并没有单独发送确认该报文段的ACK，而是直接发送了一个ACK 13834，当客户端收到该ACK时说明序号为6534的报文段也被成功接收，对应的时间**大约**为第一个SYN报文段发送后的**0.501**秒。

################################################

**8.**从之前的分析可以得知，前六个报文段中，除了第一个报文段的长度为**693**字节，其他第2~5个报文段的长度都是**1460**字节。

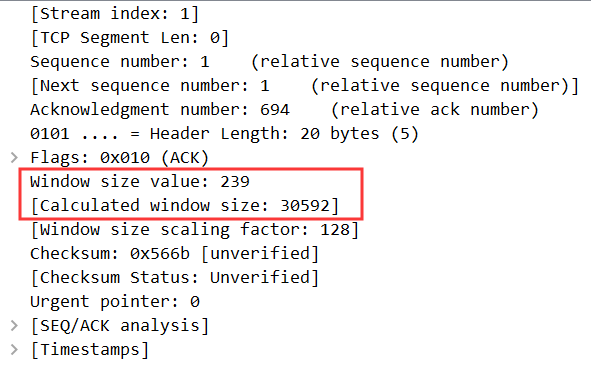


**9.**将所有报文按大小排序，主要看服务器返回的ACK报文段：

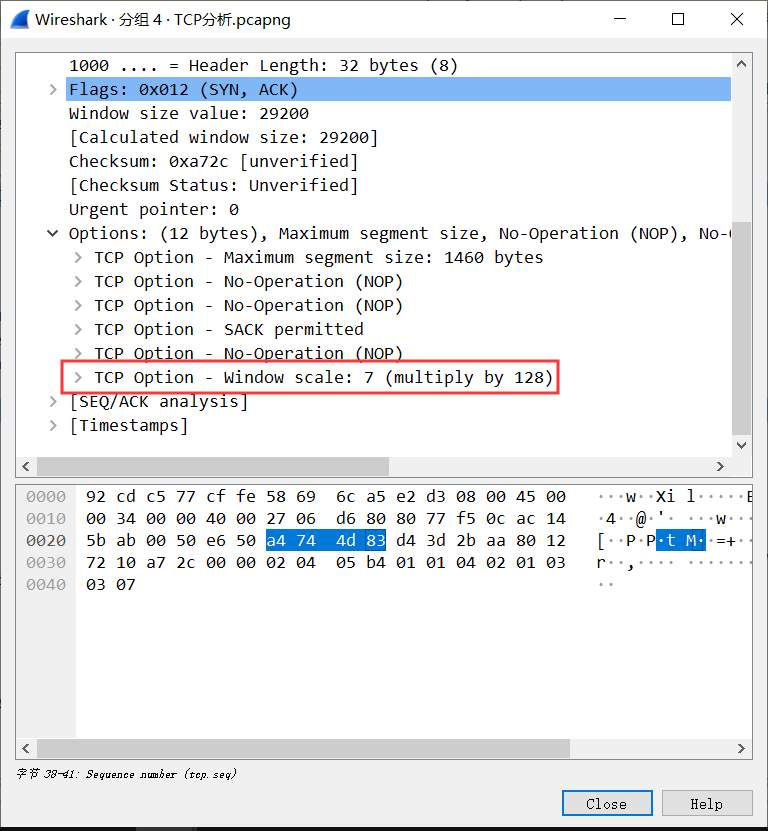


可以看到接收端公式的最小可用缓存空间为30592字节。

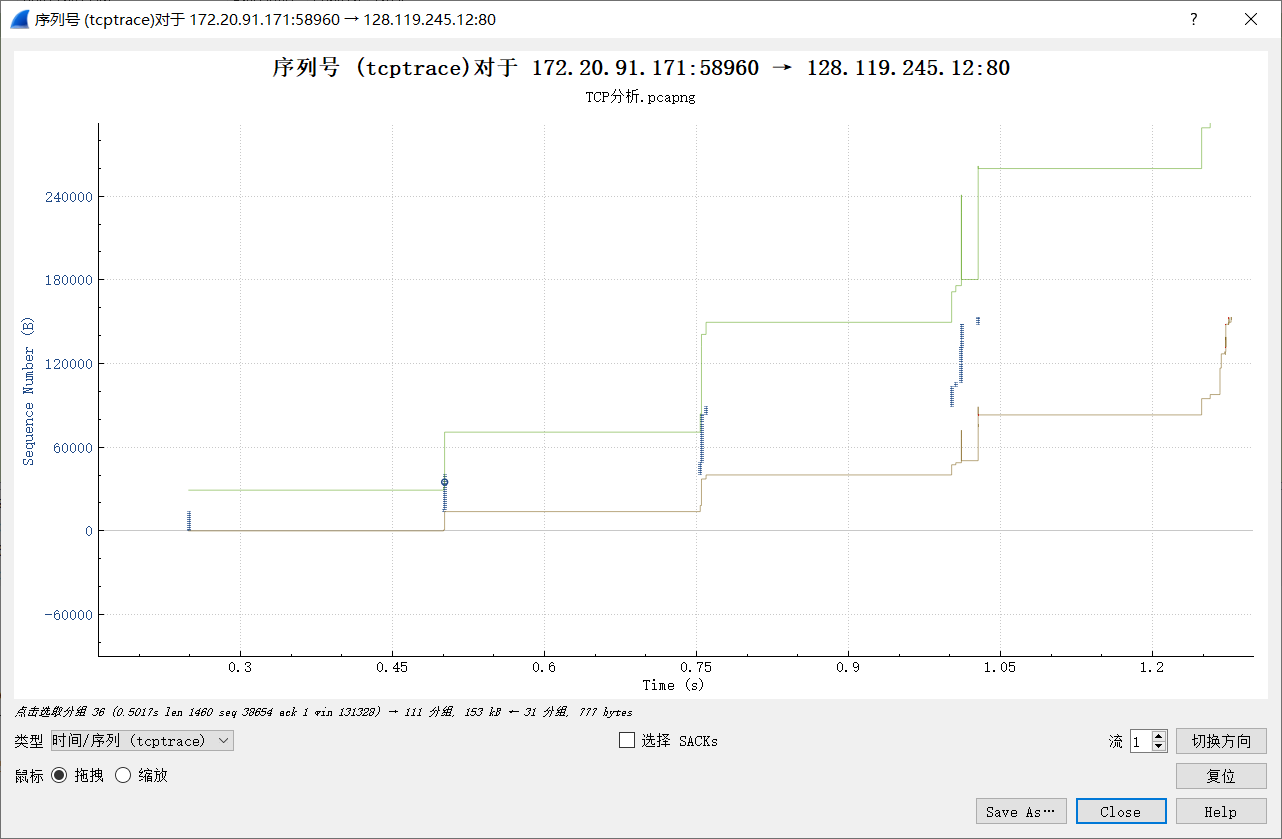
注意这里的窗口大小是Wireshark**经过计算后**得到的实际缓存空间大小，进入报文段详细查看，可以看到该报文段的窗口字段的值为239，经过计算后的窗口大小为**30592字节**。整个过程中接收端没有对发送端的传输进行限制。



那么为什么Wireshark会对窗口字段的值进行计算呢？查找资料后得知这是因为在TCP三次握手的过程中，连接双方会约定对窗口字段的值进行放大。找到SYNACK报文，可以看到其中的**选项**字段中约定了对窗口字段表示的实际窗口大小进行128倍的放大。



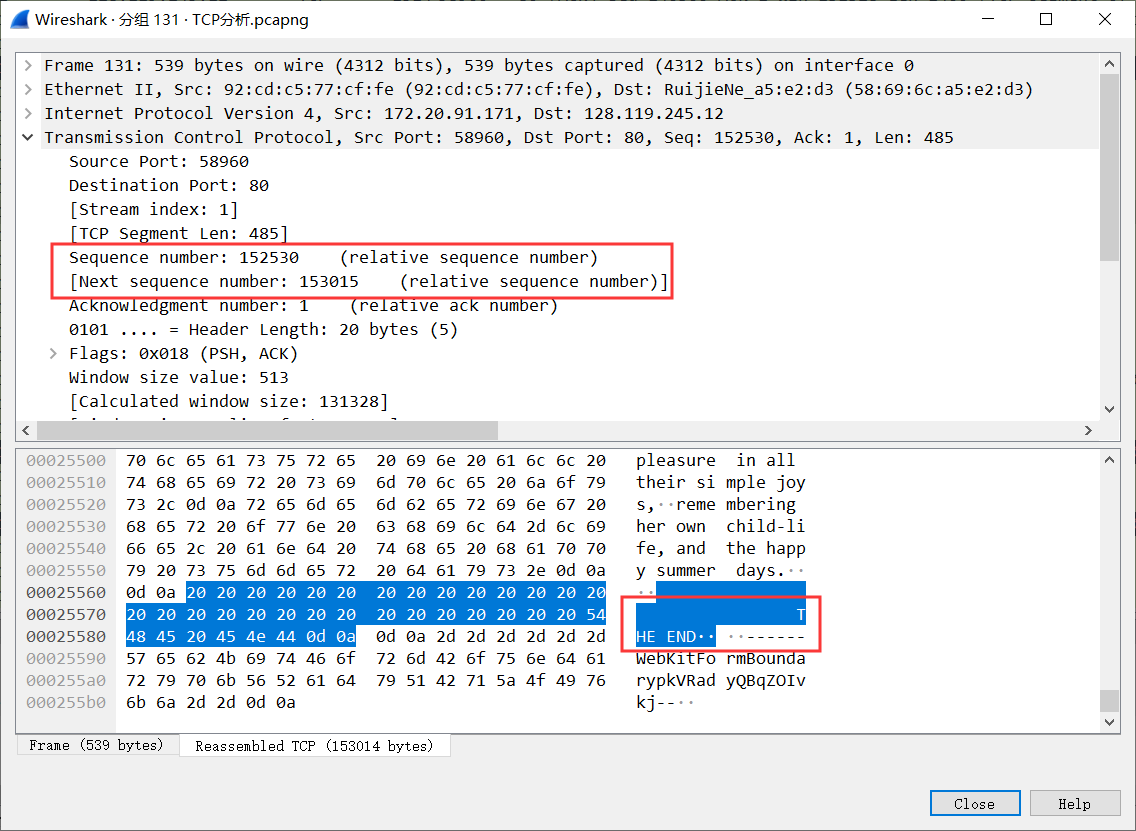
**10.**没有重传的报文段。使用Wireshark的统计功能，可以得到上传过程中的序列号-时间图像，可以看出整个过程中序列号随时间一直增长，而若有重传的报文段应该会出现序列号变小的情况，因此可以判断没有重传的报文段。



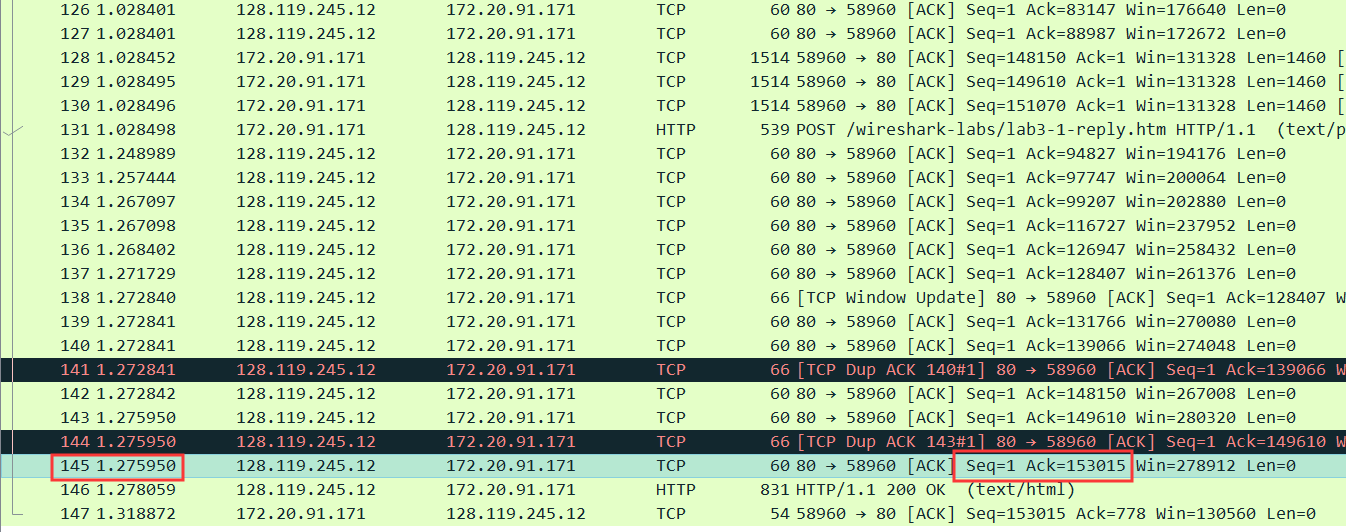
**11.**TCP建立连接后发送的第一个报文段为：



最后一个报文段为：



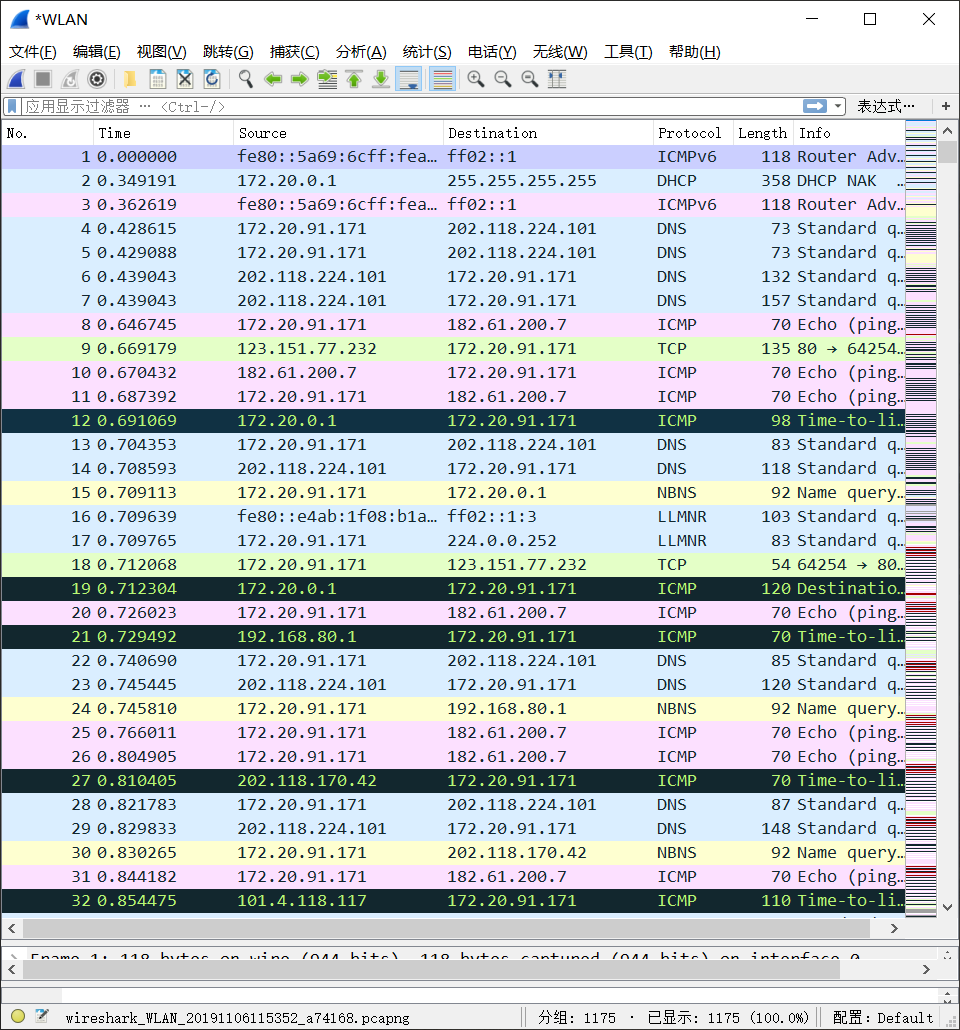
找到服务器对最后一个报文段的ACK：



可以看到这次传输共传输了(153015-1)=153014个字节，共用时1.275950-0.249290 = 1.02666秒，故TCP连接的throughput为：

**（三）IP分析**

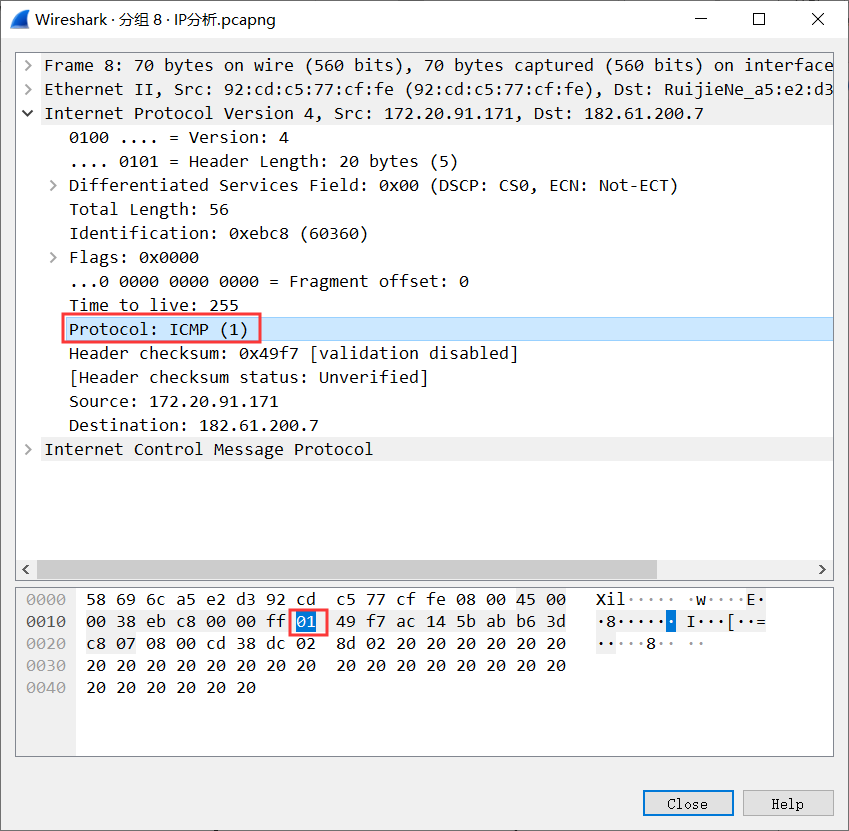
使用pingplotter对www.baidu.com发送长度分别为56字节、2000字节和3500字节的报，使用Wireshark俘获报文：



**思考问题**

**1.**从上图可以看到我的主机的IP地址为172.20.91.171。

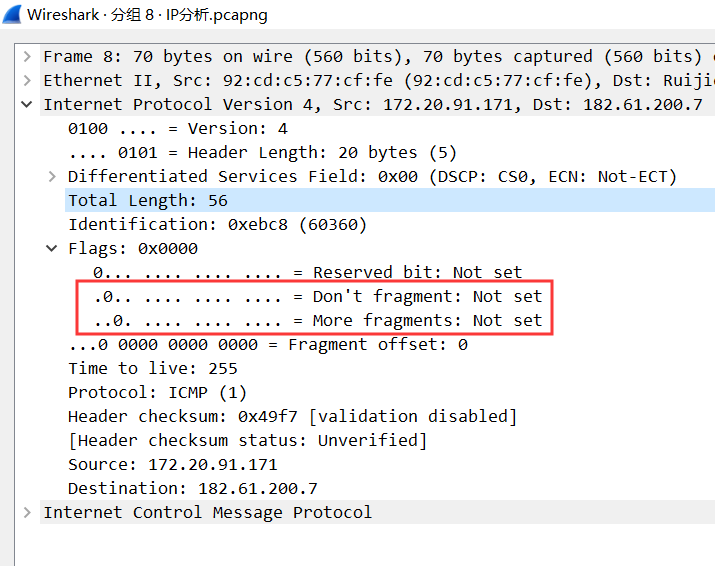
**2.**查看我的主机发送的第一个ICMP报文：



可以看到IP数据包头部中，上层协议字段的值为0x01，表示是ICMP协议

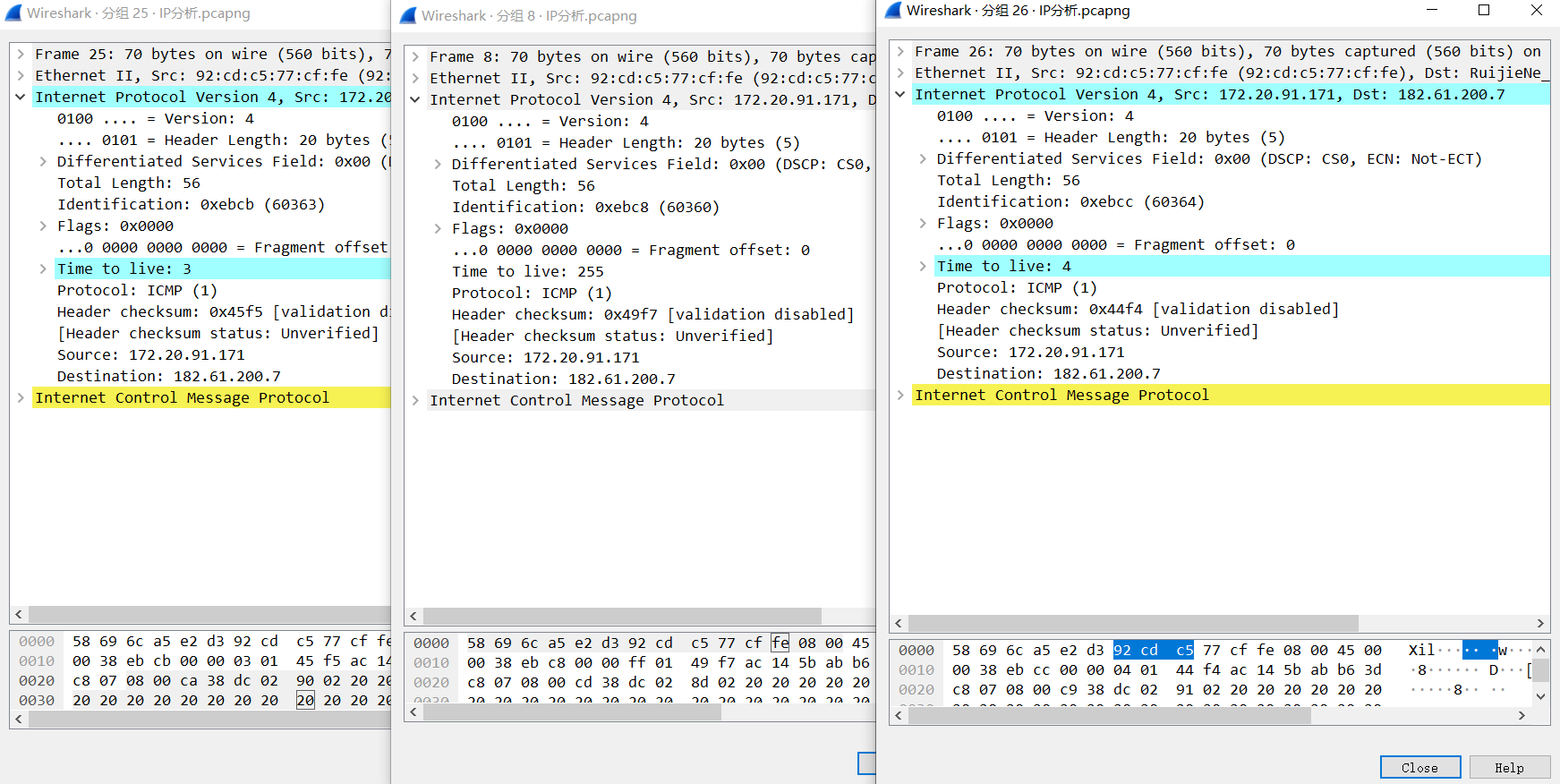
**3.**在上图中可以看到该IP数据包头部长度为20字节，而数据包总长度为56字节，故净载为36字节。

**4.**查看该数据包的flags字段



看到该数据包的MF为0，且该段的偏移为0，故该数据包没有分片。

**5.**



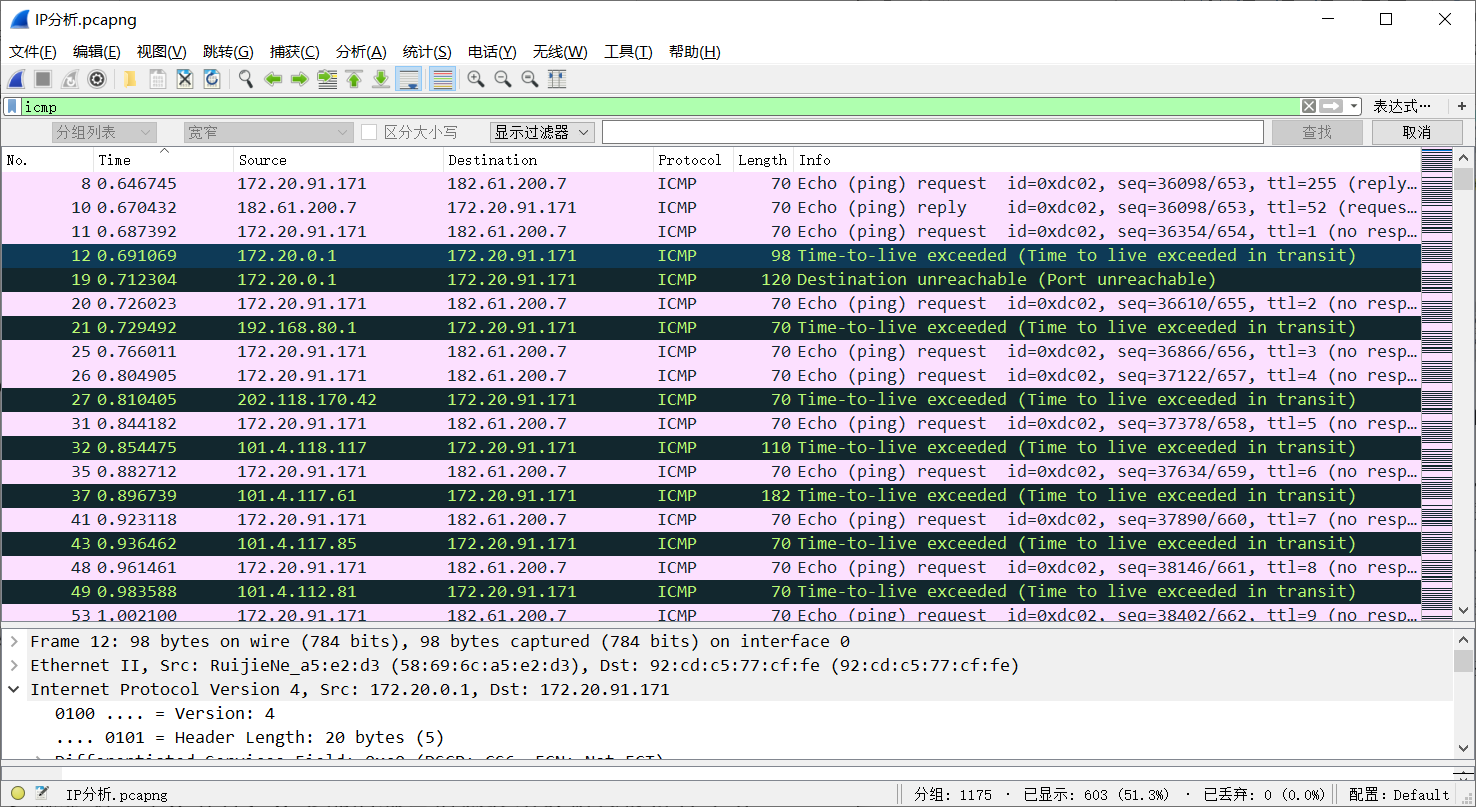
通过比较可以发现，这些IP数据包中的Identification、TTL和校验和字段总是发生改变。

**6.**必须保持常量的字段有：版本号、上层协议、源IP地址和目的IP地址，因为要使该IP数据包成功发送到目的地址，这些字段必须保持该值；

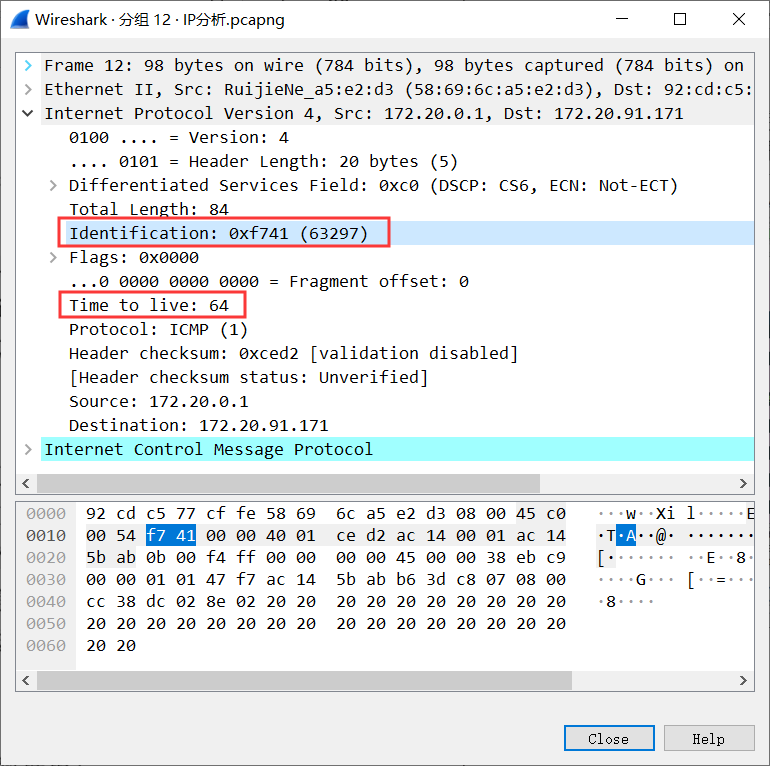
必须改变的字段有：Identification，TTL以及校验和，Identification就是IP数据包的序号，每个包的序号都不同，且根据traceroute的工作原理，每次主机发送的IP数据包的TTL都加一，而校验和为头部数据求和得出，这两者的变化都会使校验和发生改变。

**7.**Identification字段为两个字节，第一个IP数据包的Identification为随机产生，之后的IP数据包的Identification每次加1。

**8.**将俘获的报文按时间排序：



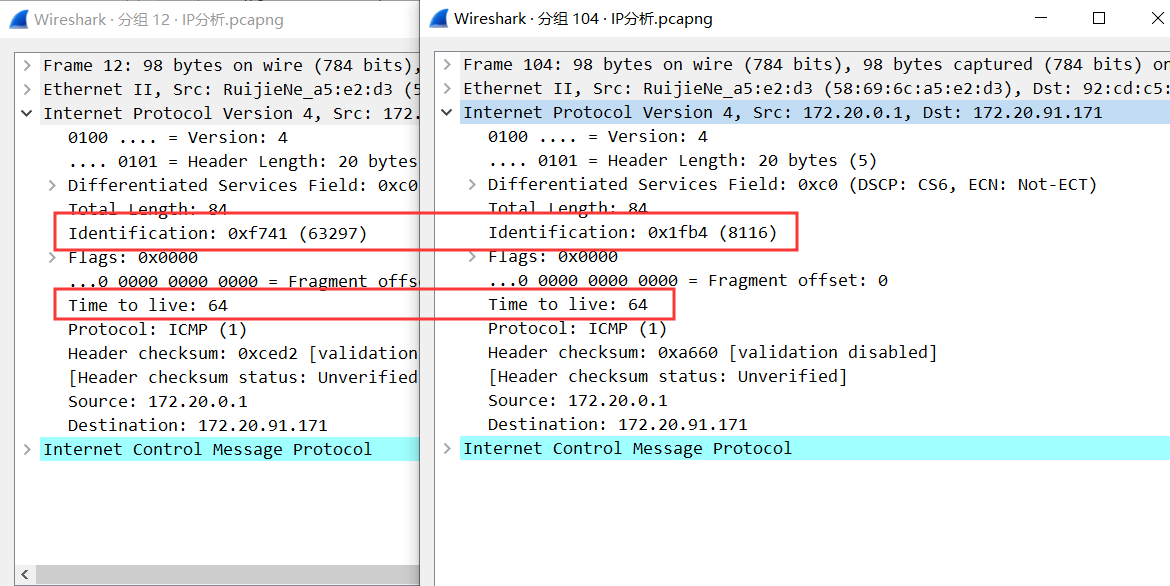
可以看到第四个报文即为最近的路由器返回的ICMP Time-to-lice exceeded消息，查看该报文：



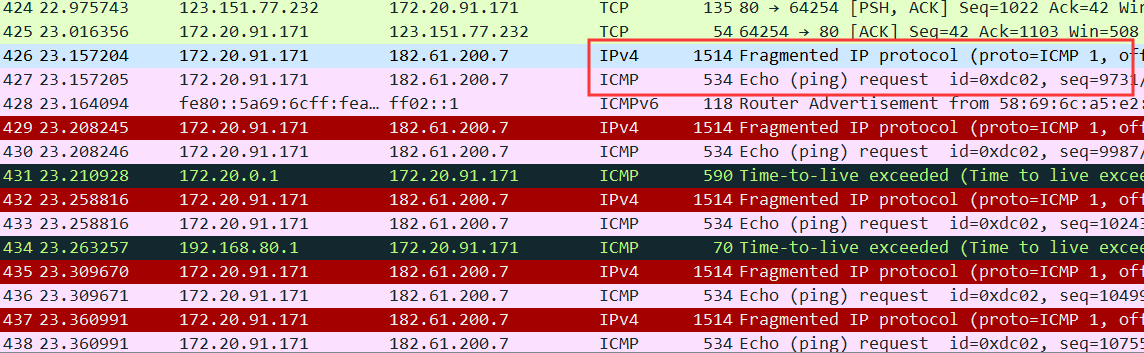
Identification字段为0xf741，TTL字段为64

**9.**最近的路由器返回的这些消息的Identification字段的值会改变，但是

TTL不变，都是64（见下图）。这是因为Identification字段用来区分不同的IP数据包，而TTL字段为默认设置的64。

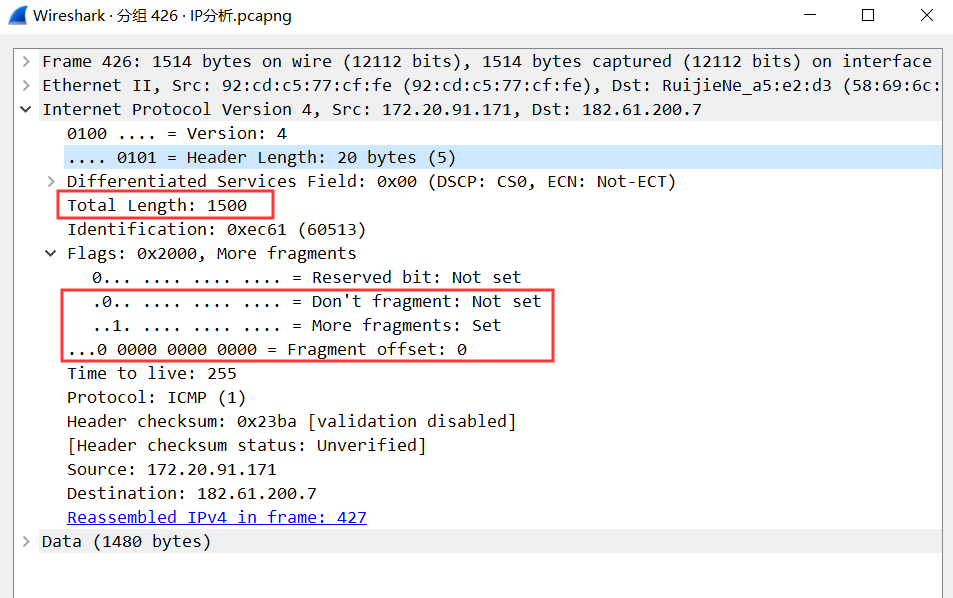


**10.**找到包大小改为2000字节后我的主机发送的第一个ICMP Echo Request消息：



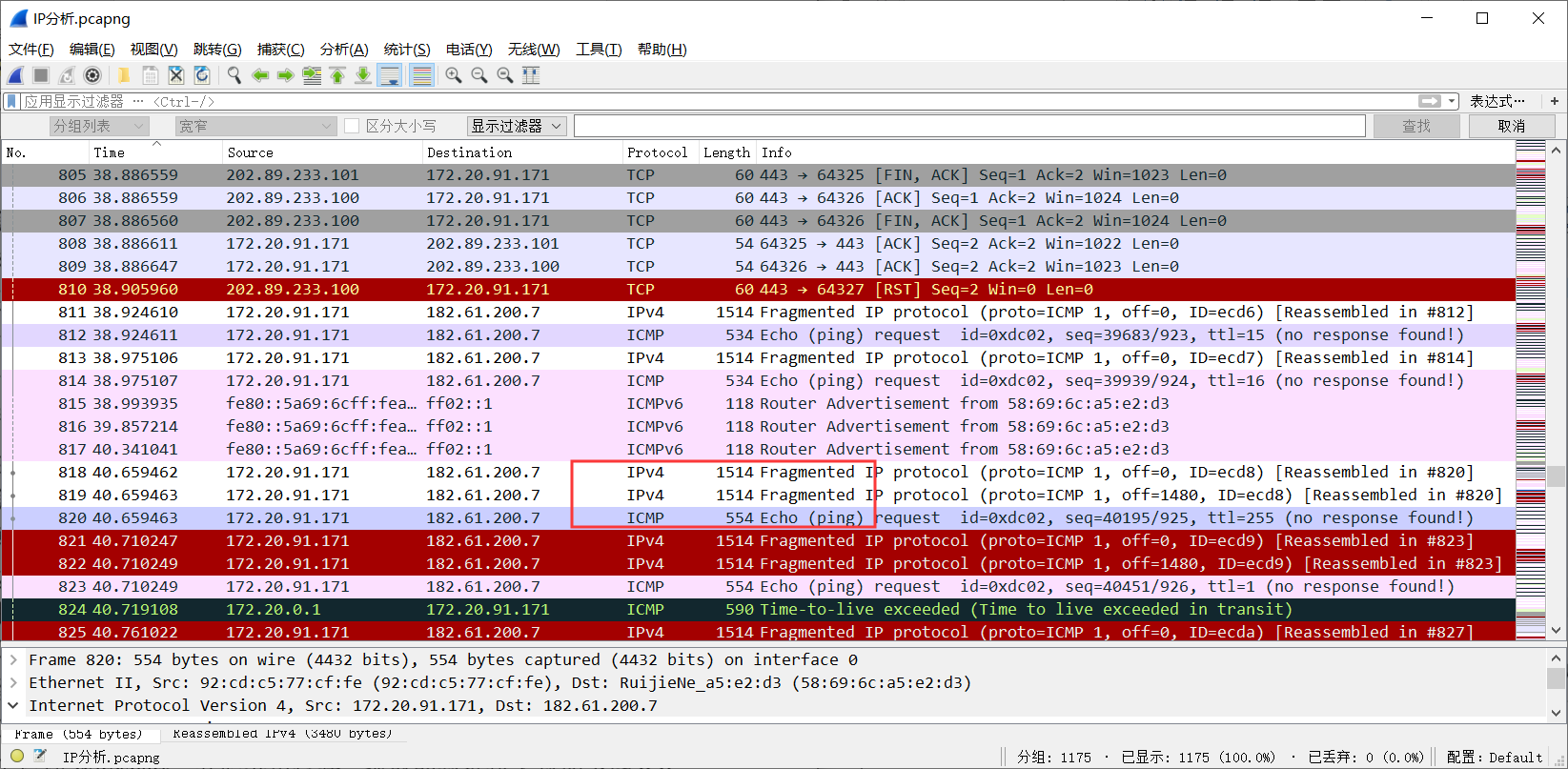
可以看到该消息被分解成了两个IP数据报。

**11.**



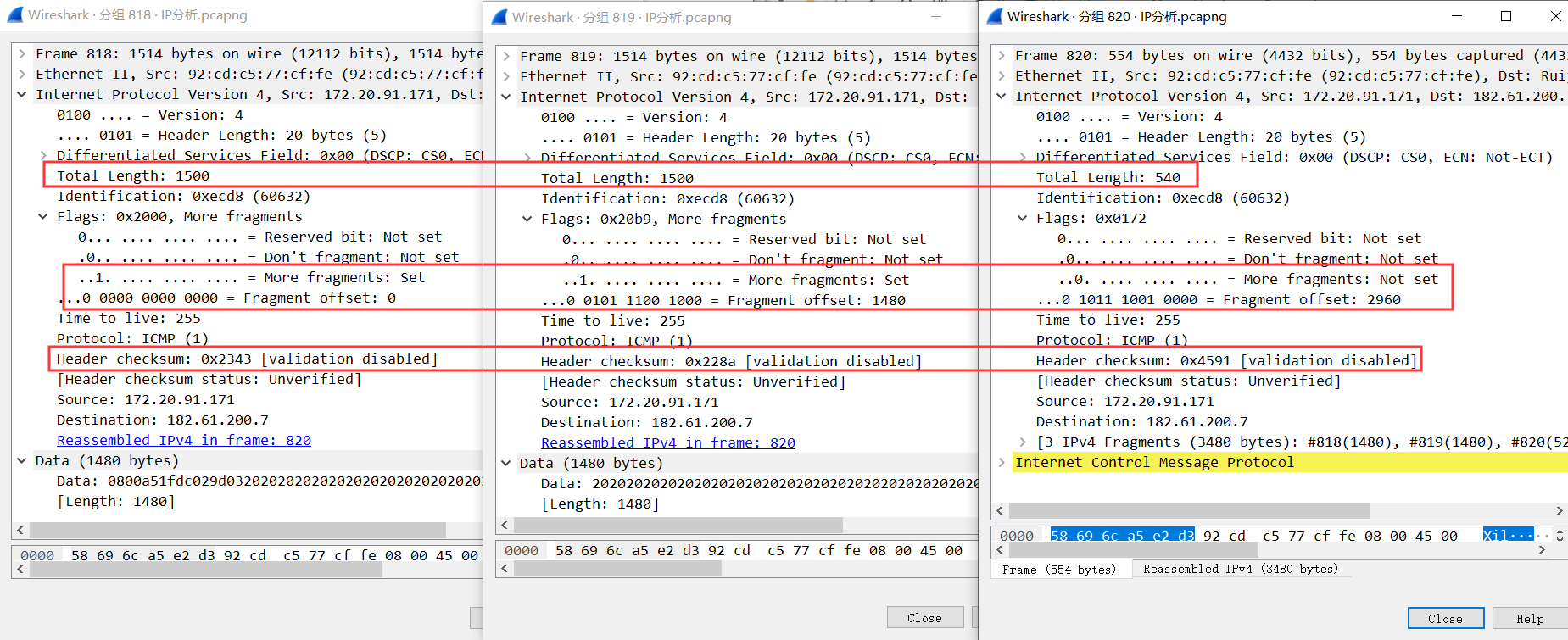
从第一个报文段的标志位可以看出：DF为0，MF为1，说明该数据包被进行了分片，且不是最后一个；而该包的片偏移为0，说明该包是第一个数据包。该分片的长度为1500字节。

**12.**找到将包大小改为3500字节后我的主机发送的第一个ICMP Echo Request消息：



可以看到原始数据包被分为了3片。

**13.**



观察这三个分片的IP数据报头部，可以看到前两个分片的总长度为1500字节，第三个的总长度为540字节；前两个的DF、MF分别为0、1，第三个的DF、MF为0、0。三个分片的片偏移分别为0、1480、2960。并且三个分片的头部检验和也不同。

**（四）以太网数据帧分析**

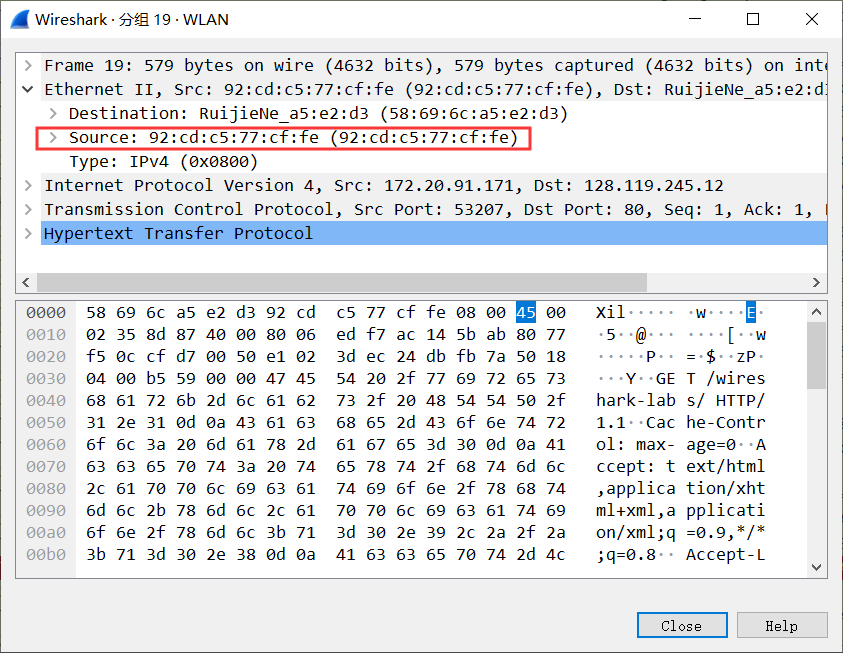
启动Wireshark分组捕获，访问http://gaia.cs.umass.edu/wireshark-labs

，停止俘获。在分组明细窗口中查看EthernetⅡ数据帧。

**思考问题**

**1.**你所在的主机的48-bit Ethernet地址（即MAC地址）是多少？

找到我的主机发出的HTTP GET请求，查看该报文：



看到我的主机的48位MAC地址为92:cd:c5:77:cf:fe。

**2.** Ethernet帧中目的地址是多少？这个目的地址是gaia.cs.umass.edu的Ethernet地址吗？

从上图可以看出目的地址为58:69:6c:a5:e2:d3，这不是gaia.cs.umass.edu的MAC地址，而是我所在的无线局域网的AP的MAC地址。

## 四、实验心得

https://www.cnblogs.com/s313139232/p/7830983.html

https://www.jianshu.com/p/f331d5f0b979

https://blog.csdn.net/u012375924/article/details/82806617

https://www.it610.com/article/2173439.htm