**计算机网络TCP实验报告**

**姓名：陈鹤影 学号：PB21061287 日期：2022.10.23**

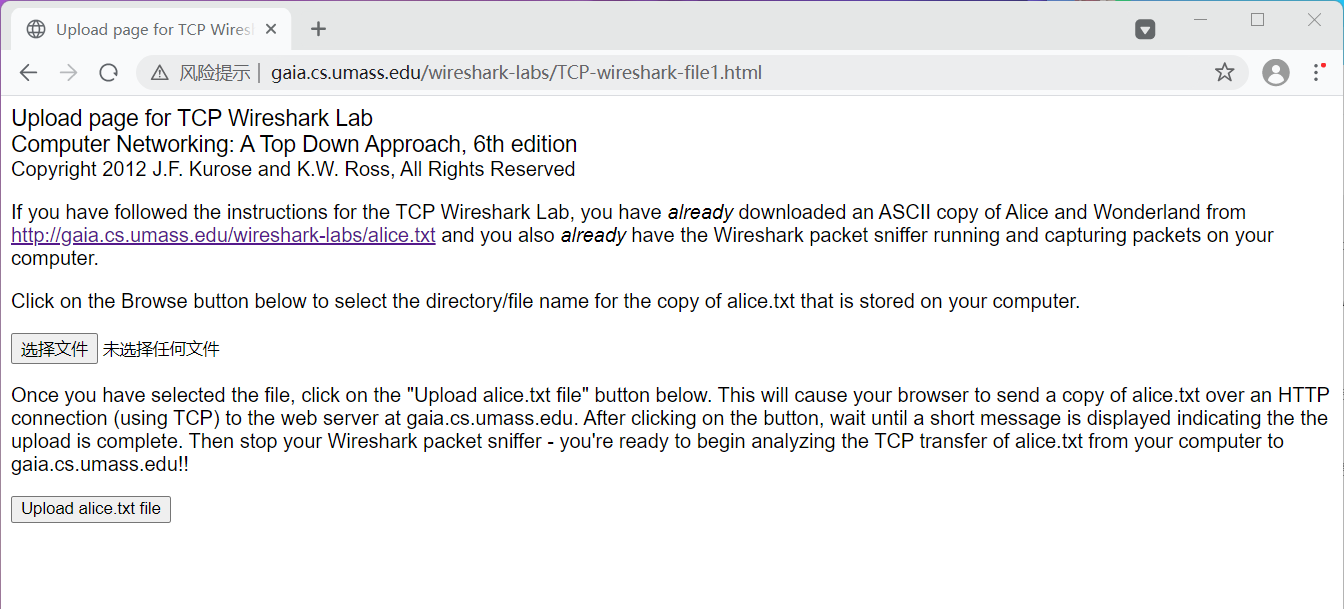
1. **实验目的：**
2. **熟悉并了解TCP报文的具体格式，掌握Seq、ACK、WIN等。**
3. **掌握TCP的建立过程——三次握手。**
4. **通过对TCP流的分析，掌握TCP的拥塞控制机制。**
5. **学习利用Wireshark绘制TCP流图形。**
6. **掌握TCP连接性能的相关计算。**
7. **实验流程及问题回答：**

**Part 1：Capturing a bulk TCP transfer from your computer to a remote server**

1. **实验内容：**

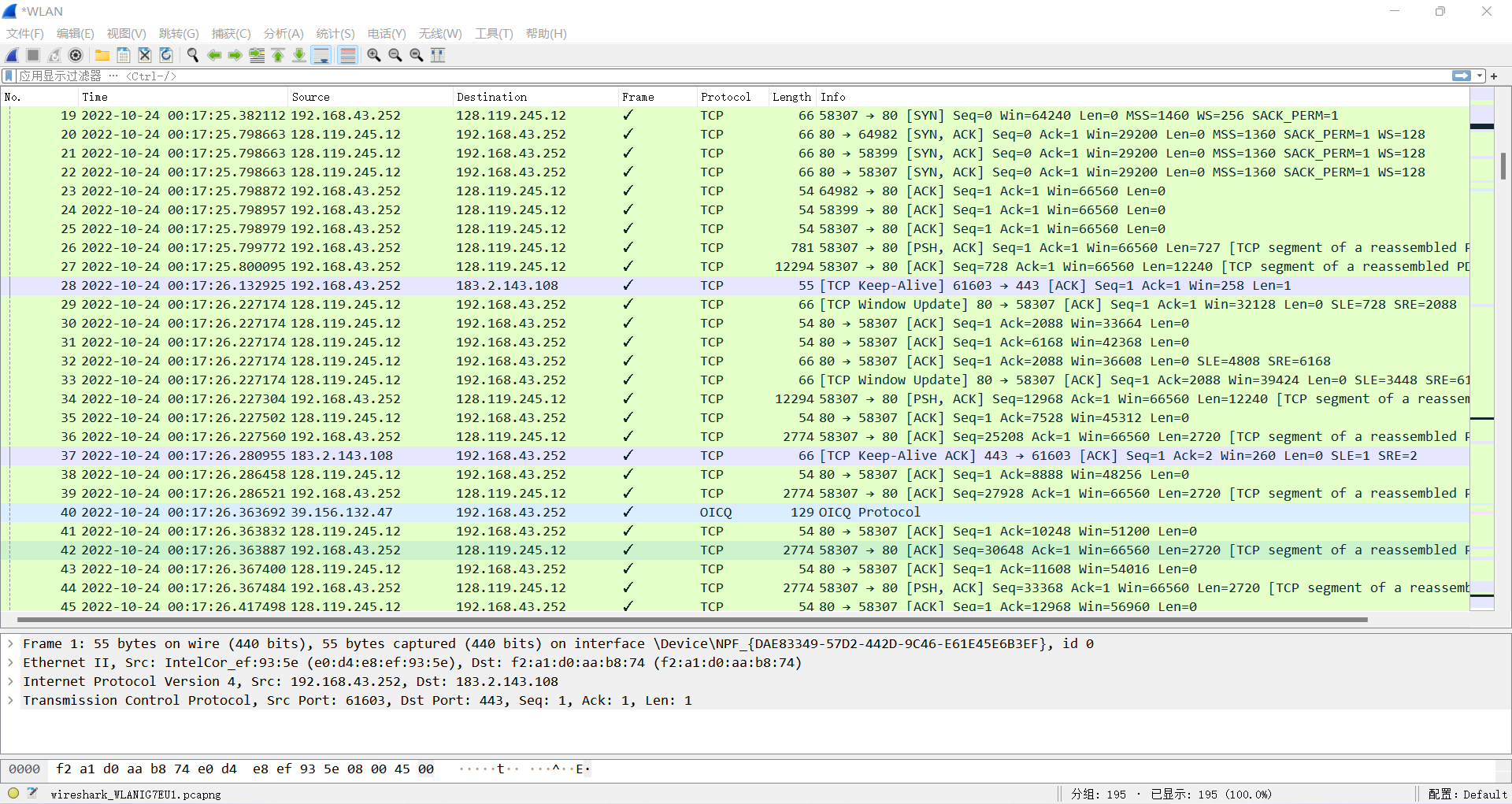
使用Wireshark跟踪文件从主机到远程服务器的TCP传输，利用HTTP POST方法将文件发送到Web服务器（传输较大的数据量）。

1. **实验流程：**
2. 获取需要传输的文件（获取文件储存在alice.txt中）
3. 登入<http://gaia.cs.umass.edu/wireshark-labs/TCP-wireshark-file1.html> 网页。界面如图所示：



**（图1-1 文件上传登入界面）**

1. 浏览并选择对应的文件（alice.txt），启动Wireshark抓包，上传文件。
2. 上传成功，结束Wireshark抓包。抓包结果如图所示：



**（图1-2 Wireshark对大文件上传过程的抓包）**

**Part 2：A first look at the captured trace**

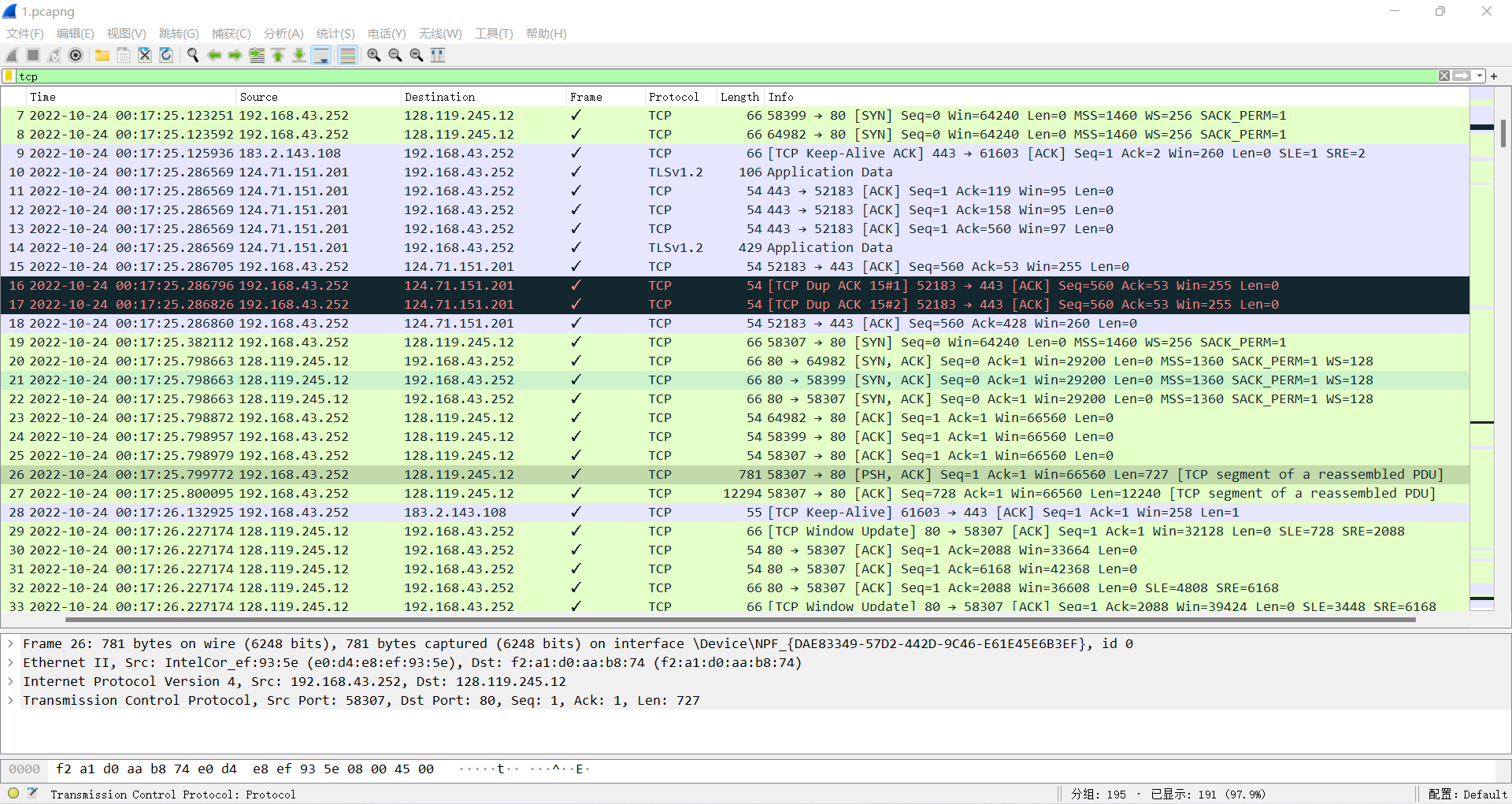
1. **实验内容：**

初步分析获得的trace。

1. **实验流程：**
2. 过滤出“tcp”数据包
3. 分析数据可以观察到TCP的三次握手过程：
4. 客户端TCP首先向服务器端TCP发送SYN报文段（SYN = 1）。客户随机选择初始序列号（Seq = 0）。
5. 服务器提取TCP SYN报文段，向客户TCP发送允许连接的SYNACK报文段（SYN = 1,ACK = 客户TCP Seq+1，）。服务器随机选择初始序列号（Seq = 0）。
6. 接收到SYNACK报文段后，客户给该TCP连接分配缓存和变量。客户主机向服务器主机发送报文段（SYN = 0,ACK = 服务器TCP Seq+1）。

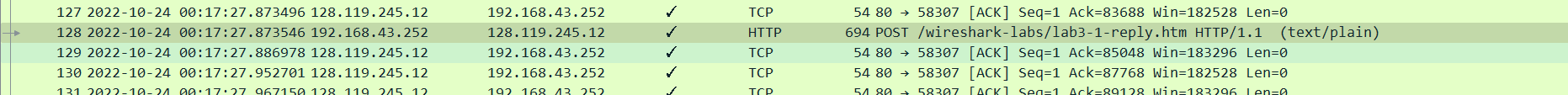
观察到过程如下，其中我们可以观察到客户TCP向服务器发送了多个报文段，同时收到了多个回复。

我们观察到在完成第三次握手后，TCP连接进入ESTABLISHED状态，随后客户和服务器开始传送数据。可以观察到一个携带应用数据的PSH包。



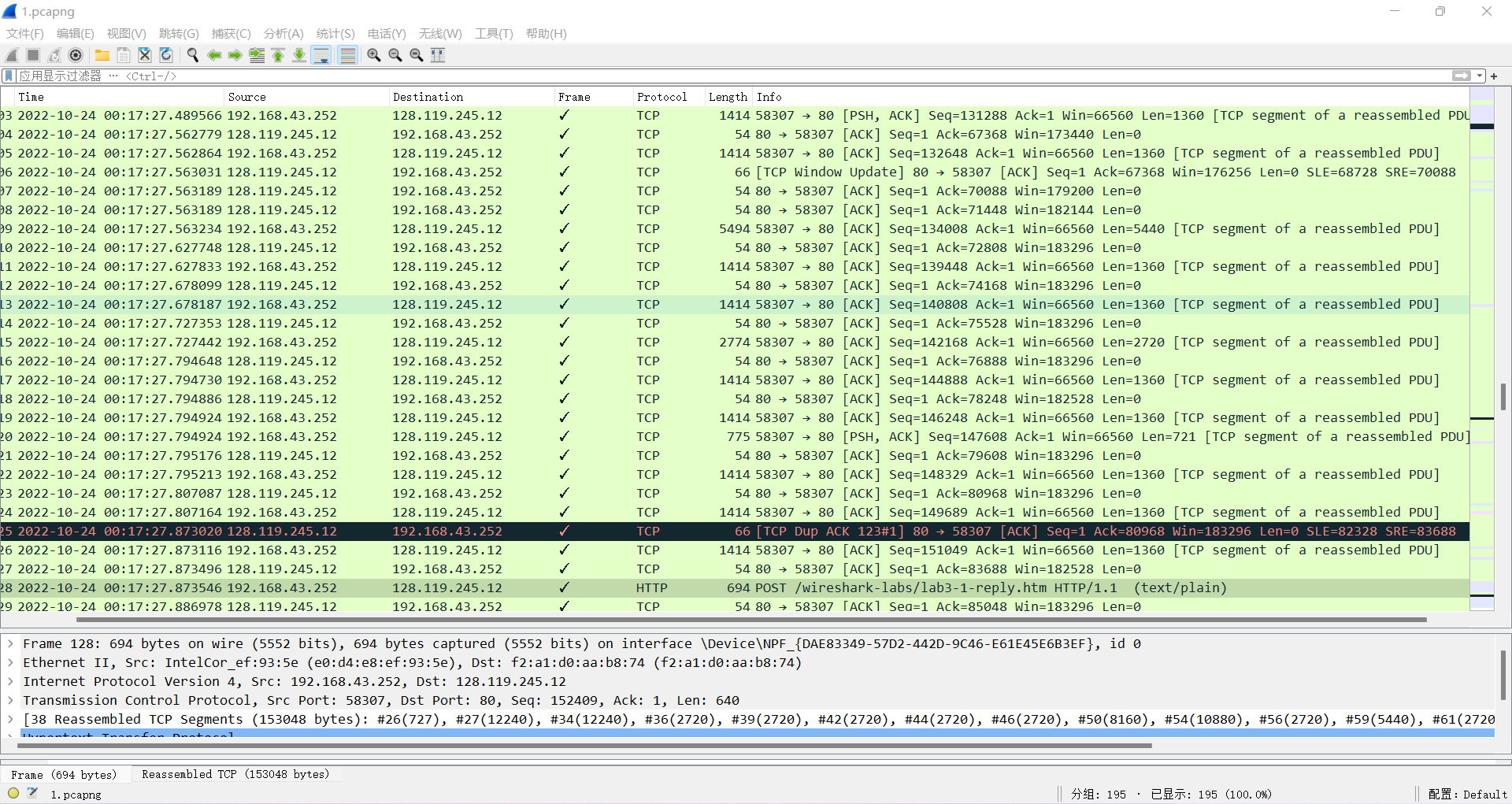
**（图1-3 三次握手过程）**

1. 可以观察到HTTP POST报文：



**（图1-4 HTTP POST报文）**

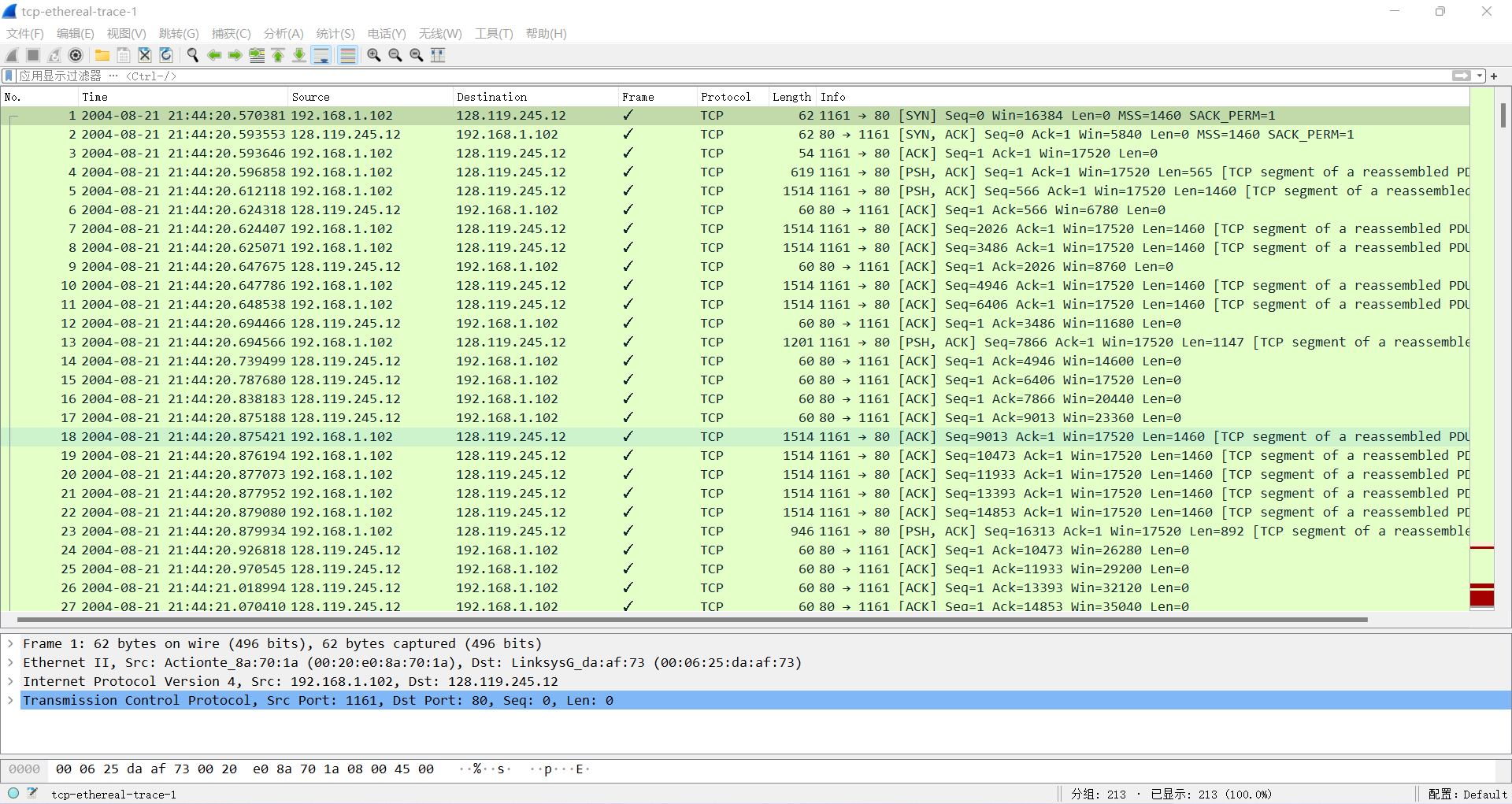
1. 同样也可以观察到大量带有“[TCP segment of a reassembled PDU]”信息的报文段。（在基于TCP的传输中，如果应用层消息过大（如超过MSS）TCP Segment不能一次包含全部的应用层PDU，而要把一个完整消息分成多个段，就会将除了最后一个报文段（segment）的所有其他报文段都打上“TCP segment of a reassembled PDU”。）同时也可以看到服务器返回的大量确认报文。



**（图1-5 [TCP segment of a reassembled PDU]and[ACK]）**

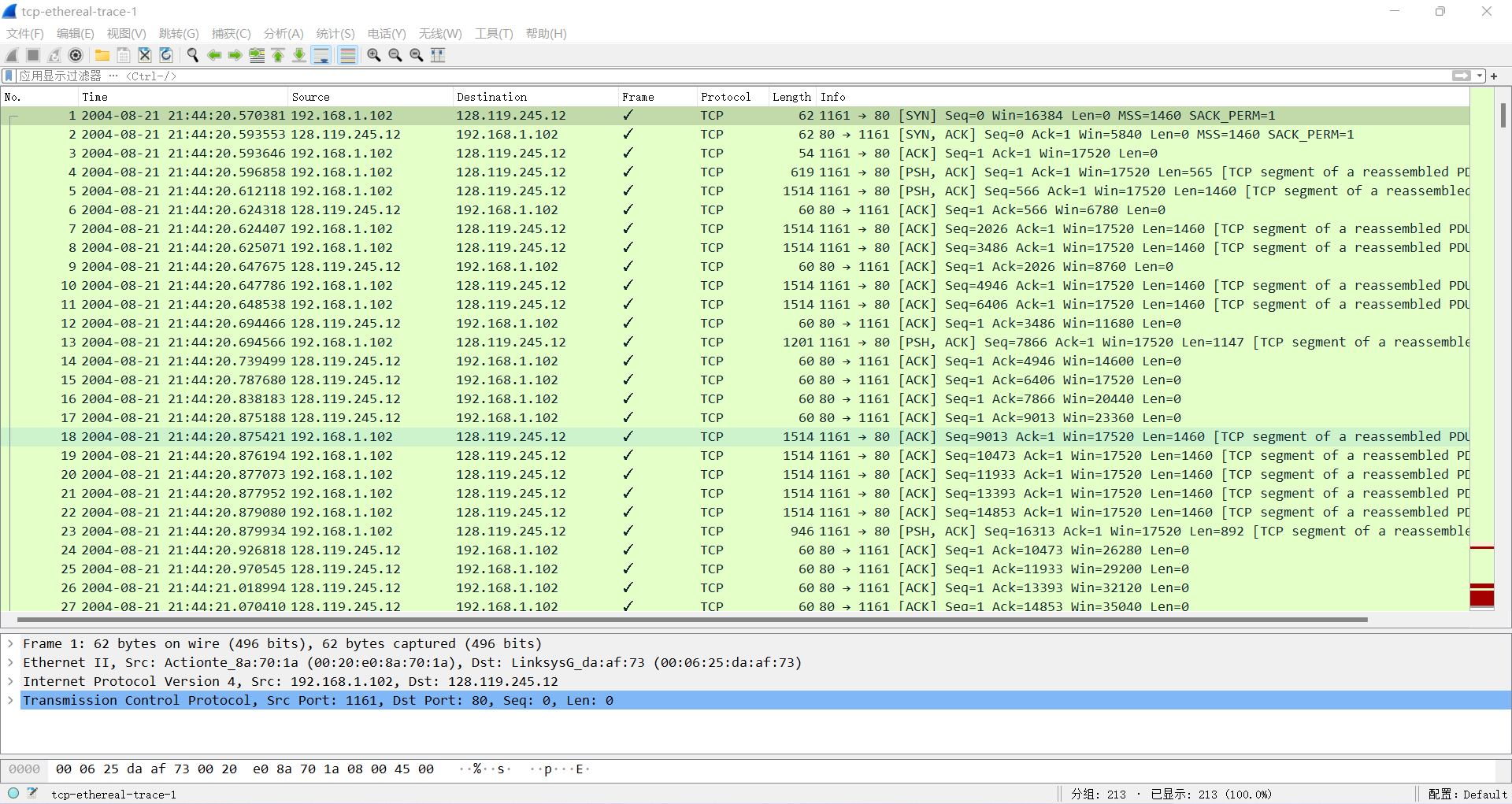
1. **Q&A(以下内容利用所给trace和本机trace分别分析)：**
2. **What is the IP address and TCP port number used by the client computer (source)that is transferring the file to gaia.cs.umass.edu?**

**Ans：**IP address：192.168.1.102；TCP port number：1161；



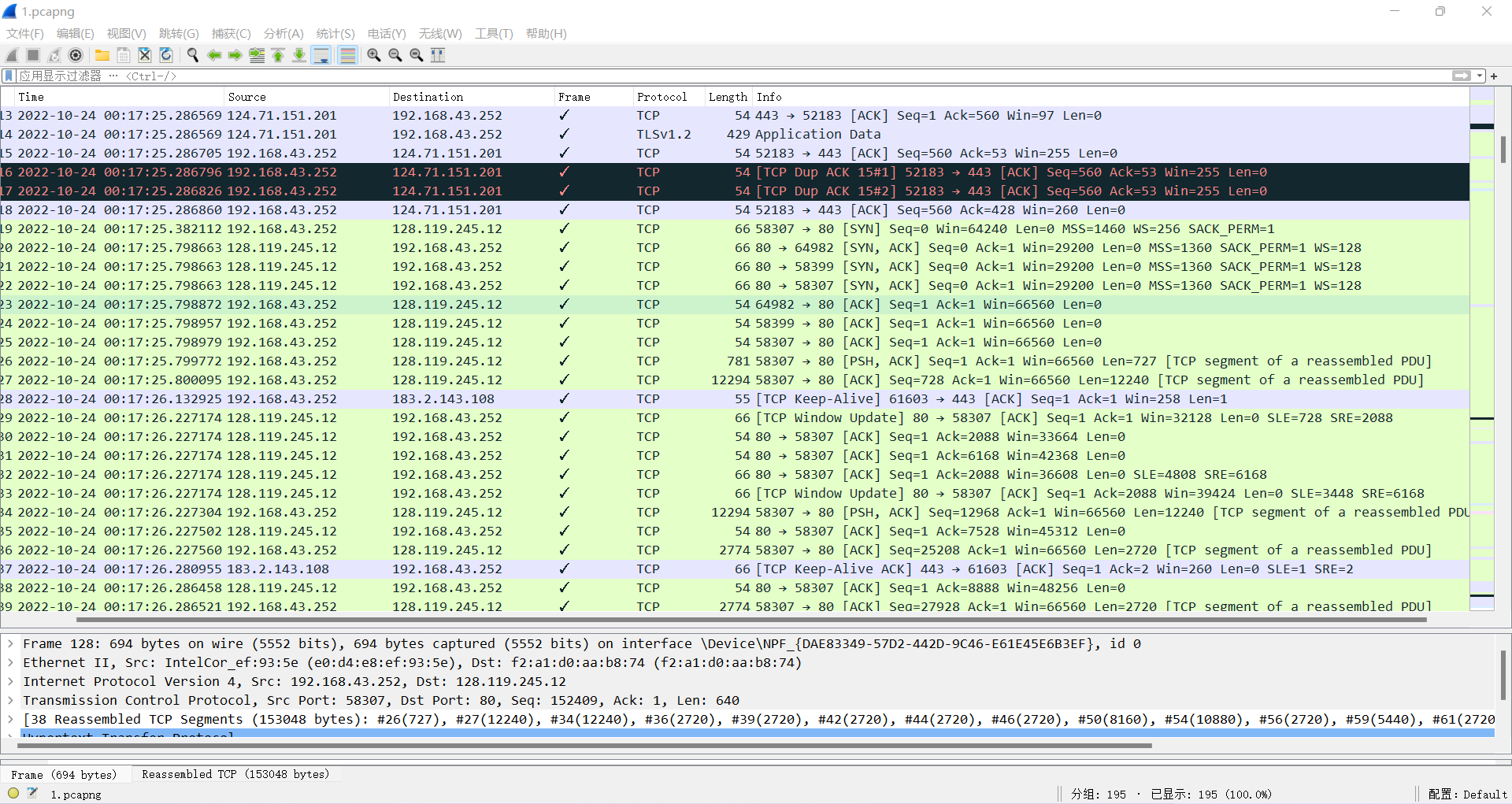
1. **What is the IP address of gaia.cs.umass.edu? On what port number is it sending and receiving TCP segments for this connection?**

**Ans：**IP address：128.119.245.12；TCP port number：80；



1. **What is the IP address and TCP port number used by your client computer (source) to transfer the file to gaia.cs.umass.edu?**

**Ans：**IP address：192.168.43.252；TCP port number：58307；



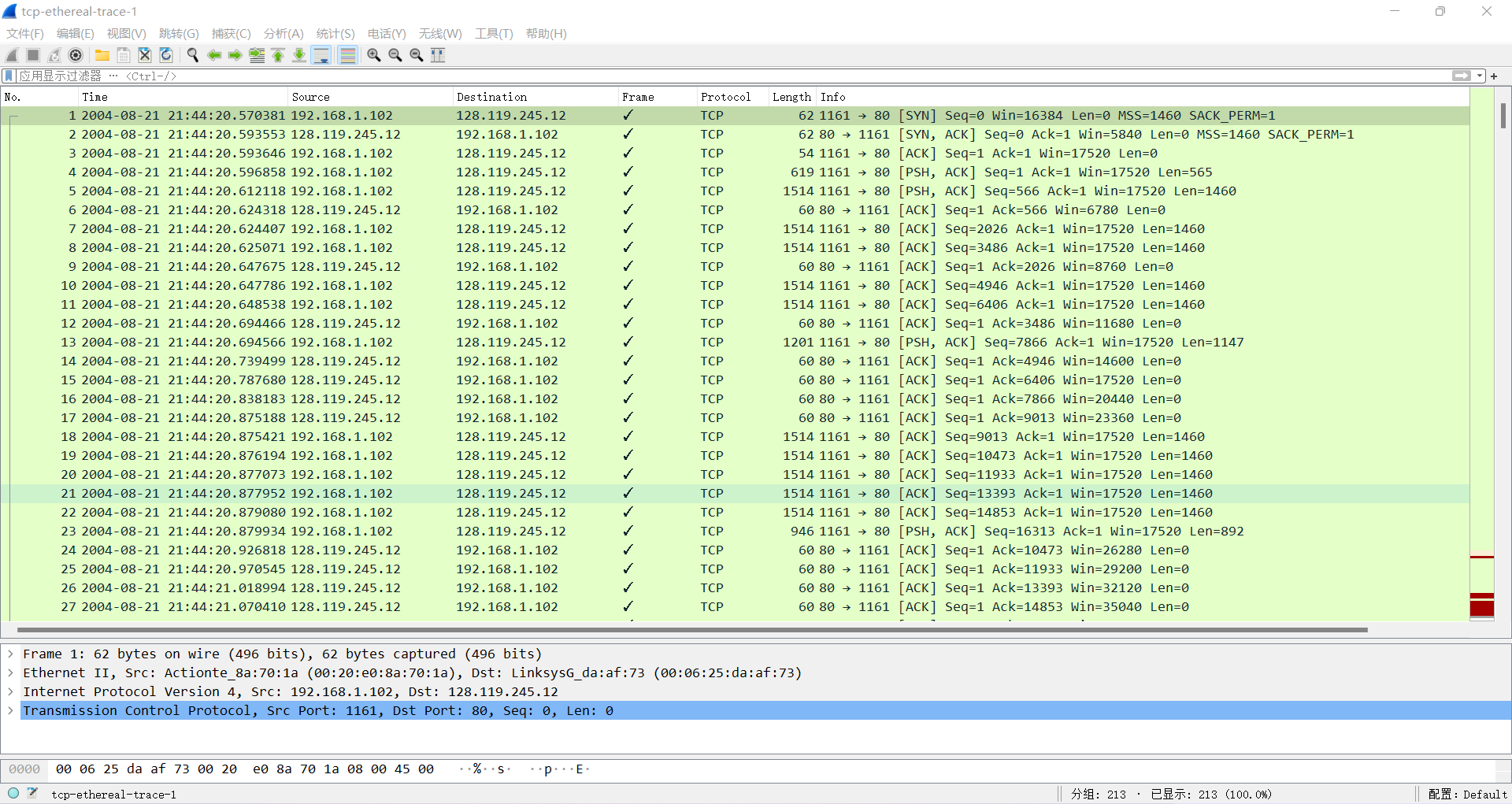
**Part 3：TCP Basics**

1. **实验内容：**

分析TCP数据流。

1. **实验流程：**

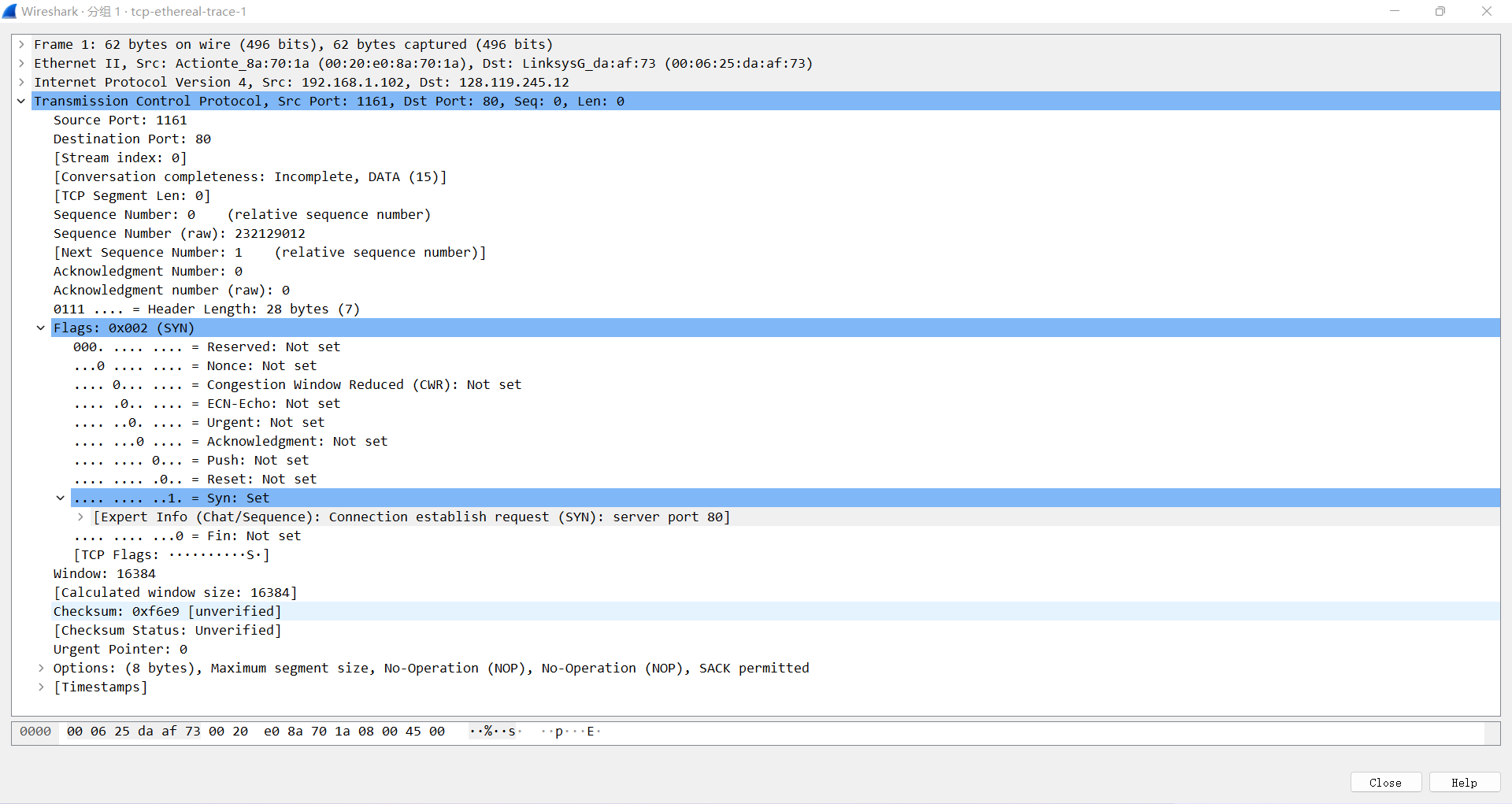
选择分析-关闭HTTP协议，关闭后Wireshark界面如图所示：



**（图1-6 在分析中关闭TCP协议后界面）**

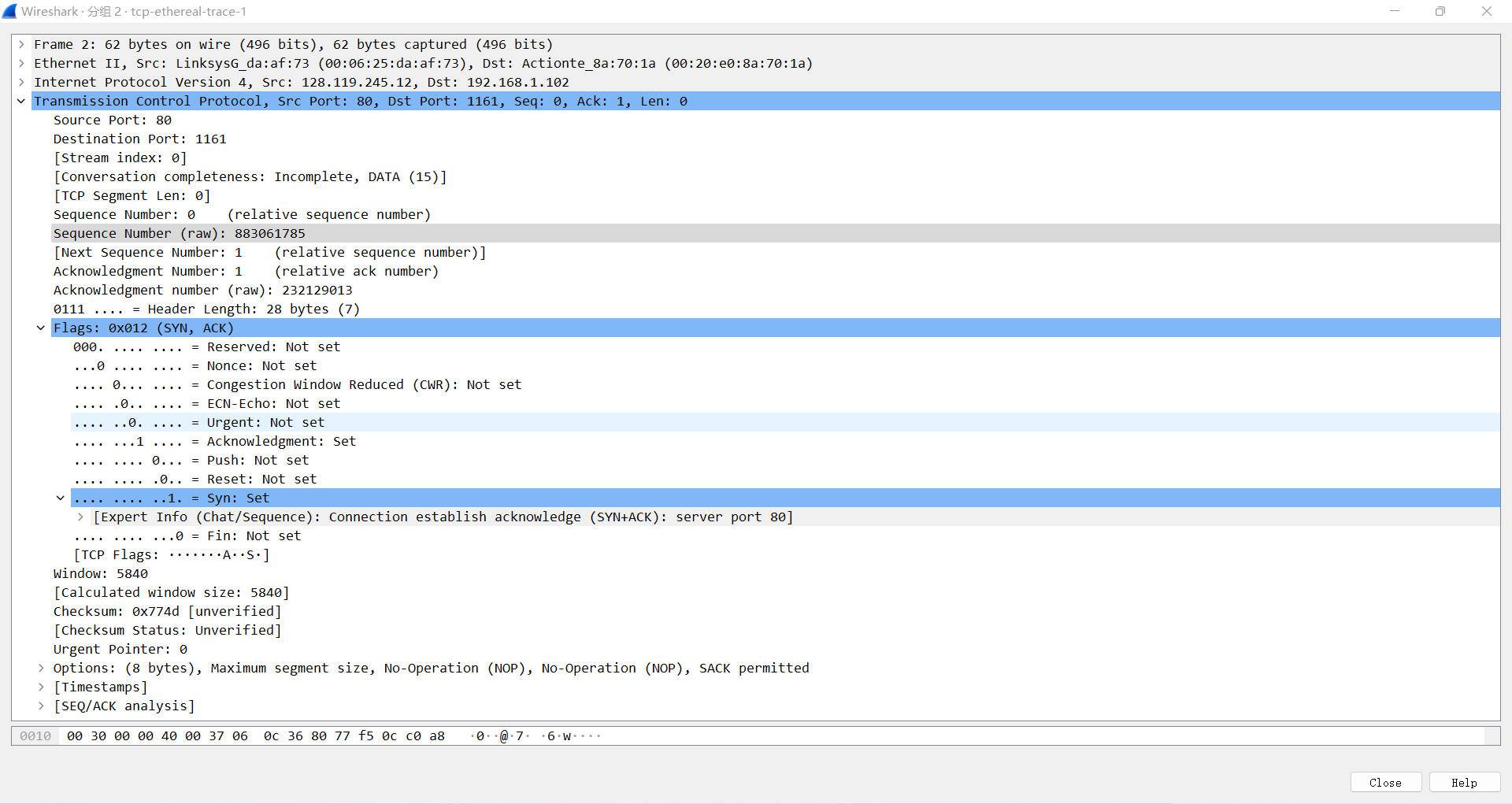
1. **Q&A(以下内容利用所给trace分析)：**
2. **What is the sequence number of the TCP SYN segment that is used to initiate the TCP connection between the client computer and gaia.cs.umass.edu? What is it in the segment that identifies the segment as a SYN segment?**

**Ans：**Sequence number：0；报文段的SYN标志被置1及“Connection establish request(SYN):sever port 80”；



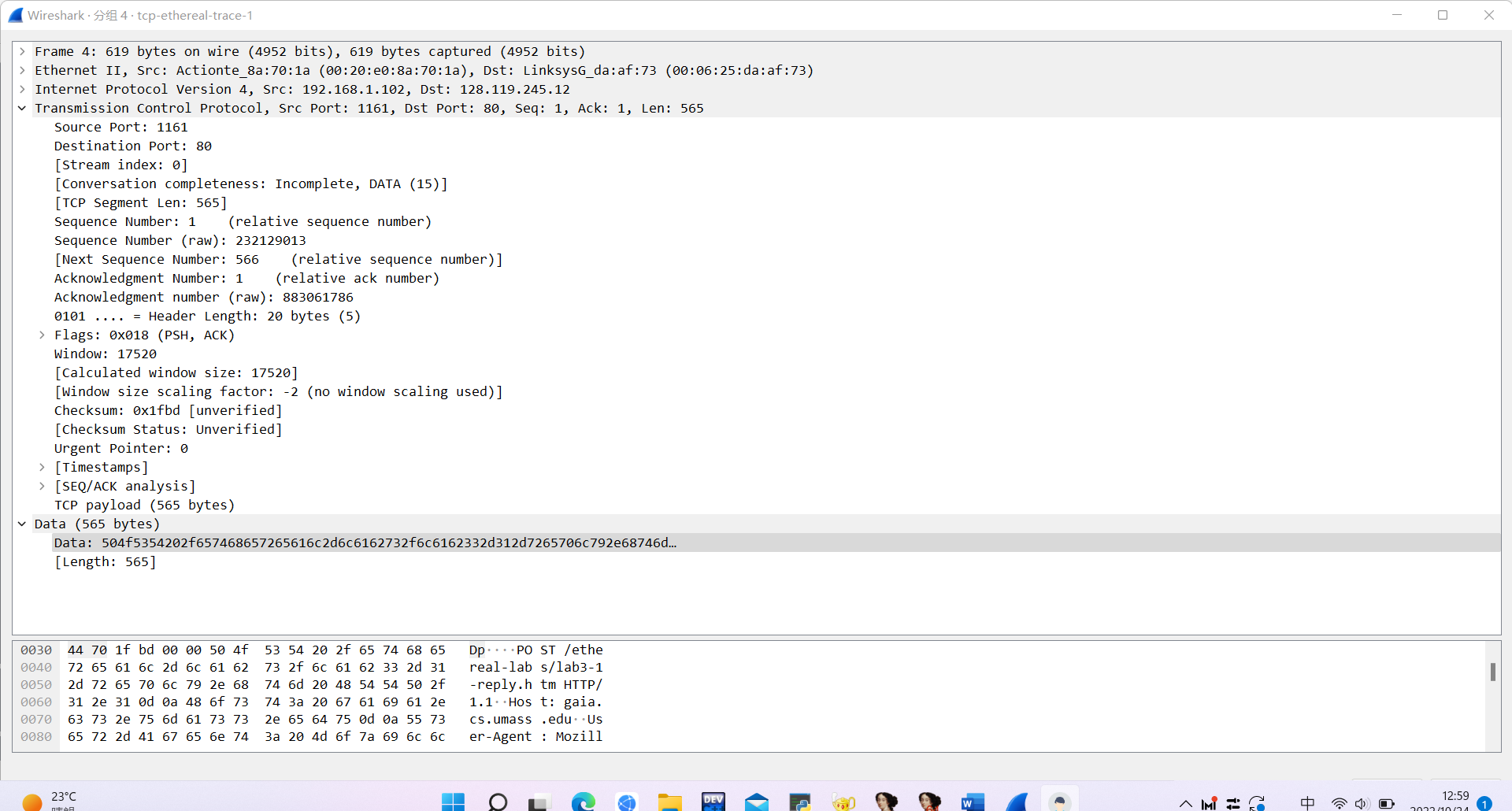
1. **What is the sequence number of the SYNACK segment sent by gaia.cs.umass.edu to the client computer in reply to the SYN? What is** **the value of the Acknowledgement field in the SYNACK segment? How did gaia.cs.umass.edu determine that value? What is it in the segment that identifies the segment as a SYNACK segment?**

**Ans：**Sequence number：0；Value of the Acknowledgement field：1；服务器通过接收到的SYN报文中Seq + 1 得到ACK的值；通过ACK和SYN均被置1证明这是一个SYNACK报文段。



1. **What is the sequence number of the TCP segment containing the HTTP POST command? Note that in order to find the POST command, you’ll need to dig into the packet content field at the bottom of the Wireshark window, looking for a segment with a “POST” within its DATA field.**

**Ans：**Sequence number：1；



1. **Consider the TCP segment containing the HTTP POST as the first segment in the TCP connection. What are the sequence numbers of the first six segments in the TCP connection (including the segment containing the HTTP POST)? At what time was each segment sent? When was the ACK for each segment received? Given the difference between when each TCP segment was sent, and when its acknowledgement was received, what is the** **RTT value for each of the six segments? What is the EstimatedRTT value (see Section 3.5.3, page 242 in text) after the receipt of each ACK? Assume that the value of the EstimatedRTT is equal to the measured RTT for the first segment, and then is computed using the EstimatedRTT equation on page 242 for all subsequent segments.**

**Ans：**Sequence numbers：1，566， 2026，3486，4946，6406；

Send times： 2004-08-21 21:44:20.596858，

2004-08-21 21:44:20.612118，

2004-08-21 21:44:20.624407，

2004-08-21 21:44:20.625071，

2004-08-21 21:44:20.647786，

2004-08-21 21:44:20.648538；

Receive times：第1个包：2004-08-21 21:44:20.624318，

第2个包：2004-08-21 21:44:20.647675，

第3个包：2004-08-21 21:44:20.694466，

第4个包：2004-08-21 21:44:20.739499，

第5个包：2004-08-21 21:44:20.787680，

第6个包：2004-08-21 21:44:20.838183；

RTT value for each：第1个包：RTT=0.02746，

第2个包：RTT= 0.035557，

第3个包：RTT=0.070059，

第4个包：RTT=0.114428，

第5个包：RTT=0.139894，

第6个包：RTT=0.189645；

EstimatedRTT value：计算公式为

EstimatedRTT =0.875·EstimatedRTT+0.125 SampleRTT

（这里假设初始EstimatedRTT与第一次得到的SampleRTT相等）

第1个包：EstimatedRTT =0.02746，

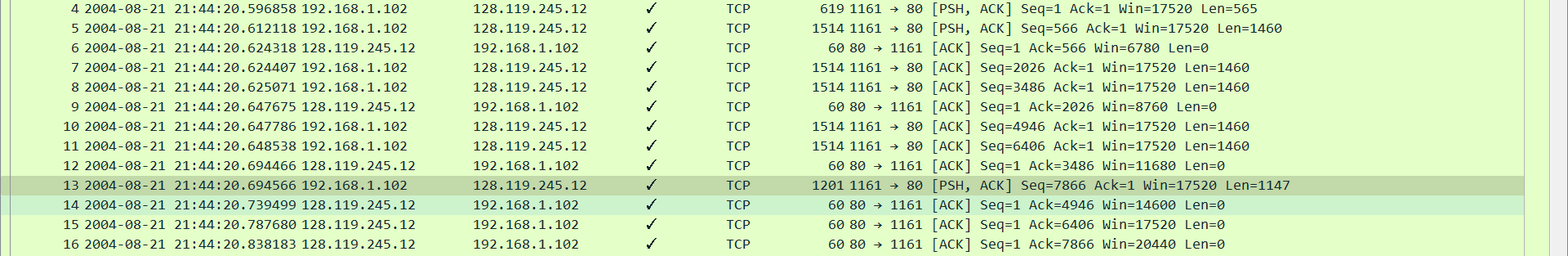
第2个包：EstimatedRTT = 0.028472125，

第3个包：EstimatedRTT =0.0336704844，

第4个包：EstimatedRTT =0.0437651738，

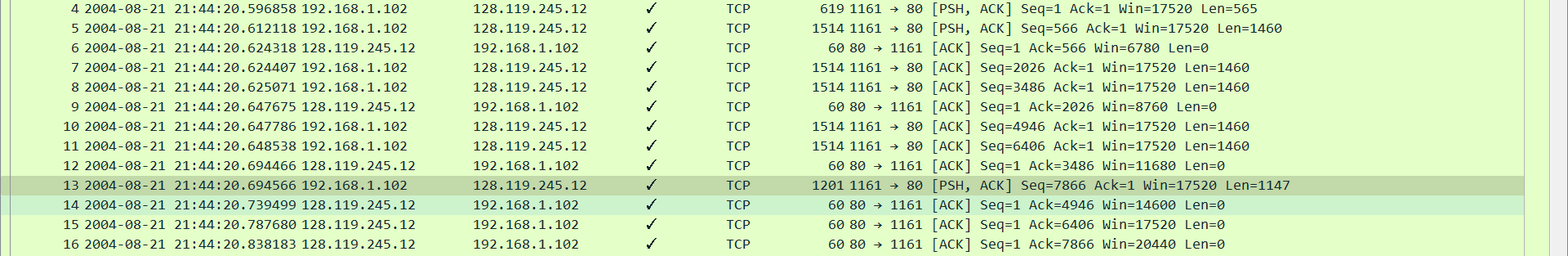
第5个包：EstimatedRTT =0.0557812771，

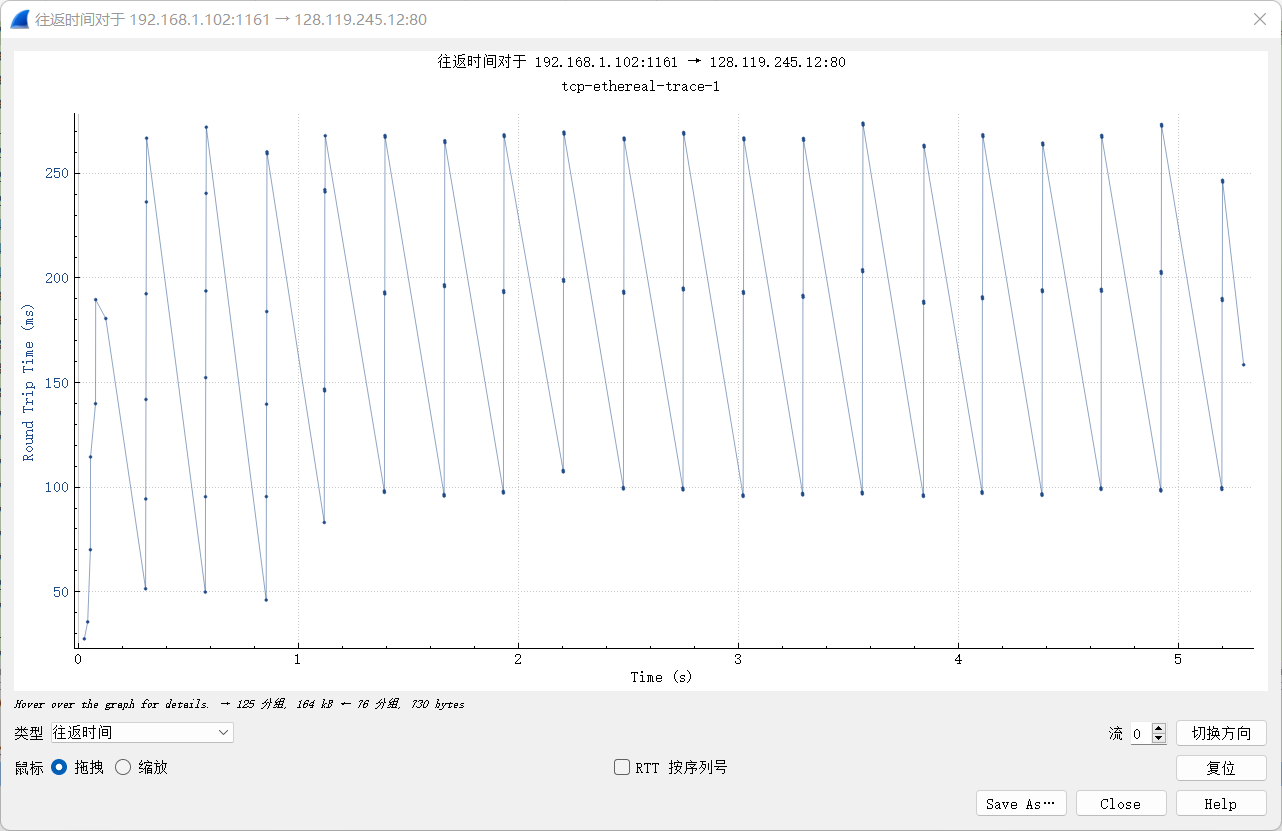
第6个包：EstimatedRTT =0.0725142425；



1. **What is the length of each of the first six TCP segments?**

**Ans：**Length：565，1460，1460，1460，1460，1460； 58307；

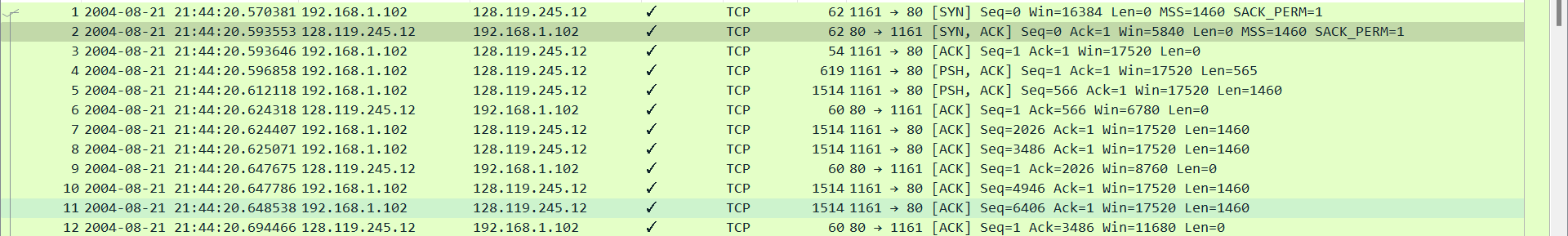




**（图1-7 TCP RTT流图形）**

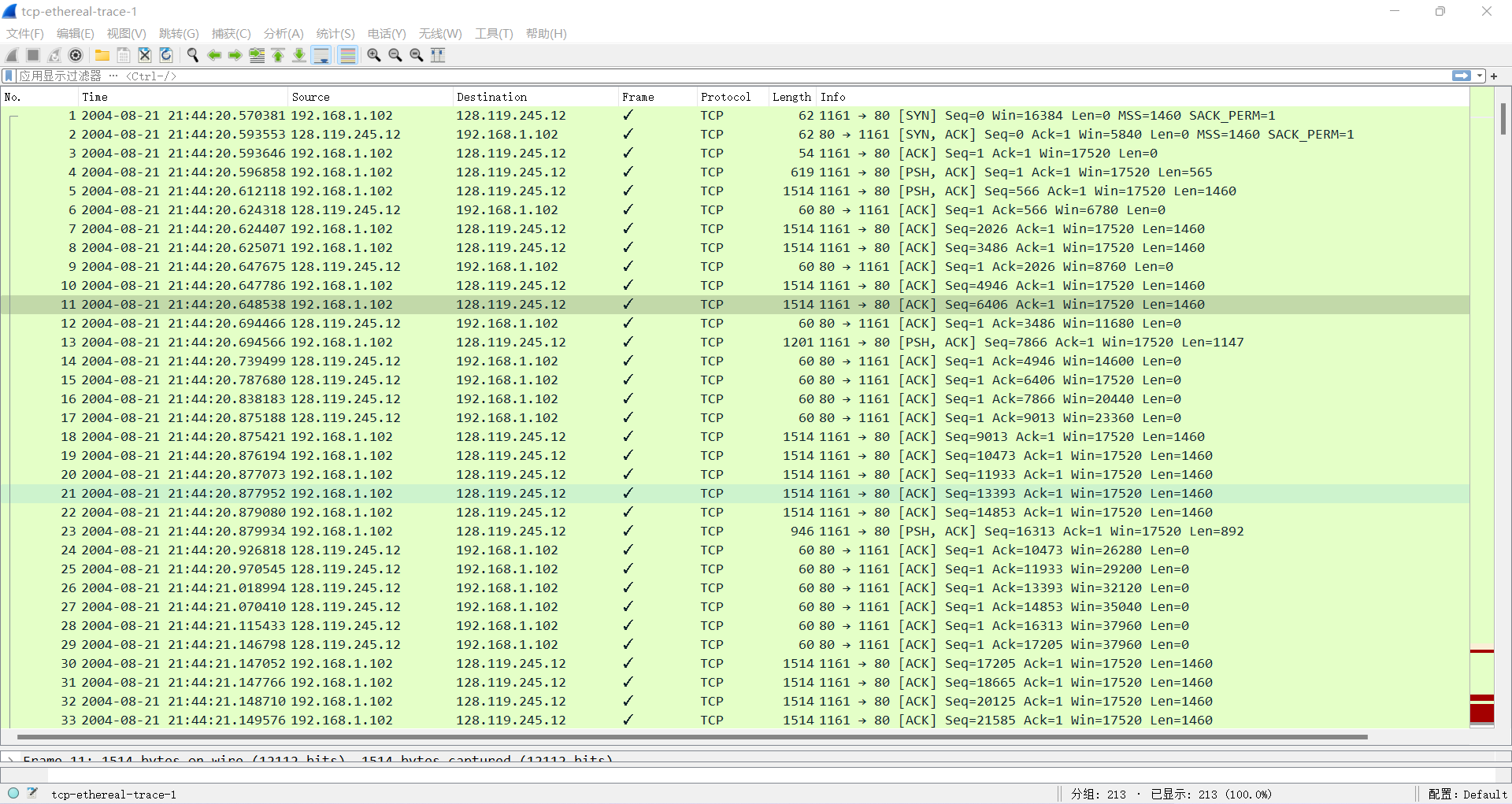
1. **What is the minimum amount of available buffer space advertised at the received for the entire trace? Does the lack of receiver buffer space ever throttle the sender?**

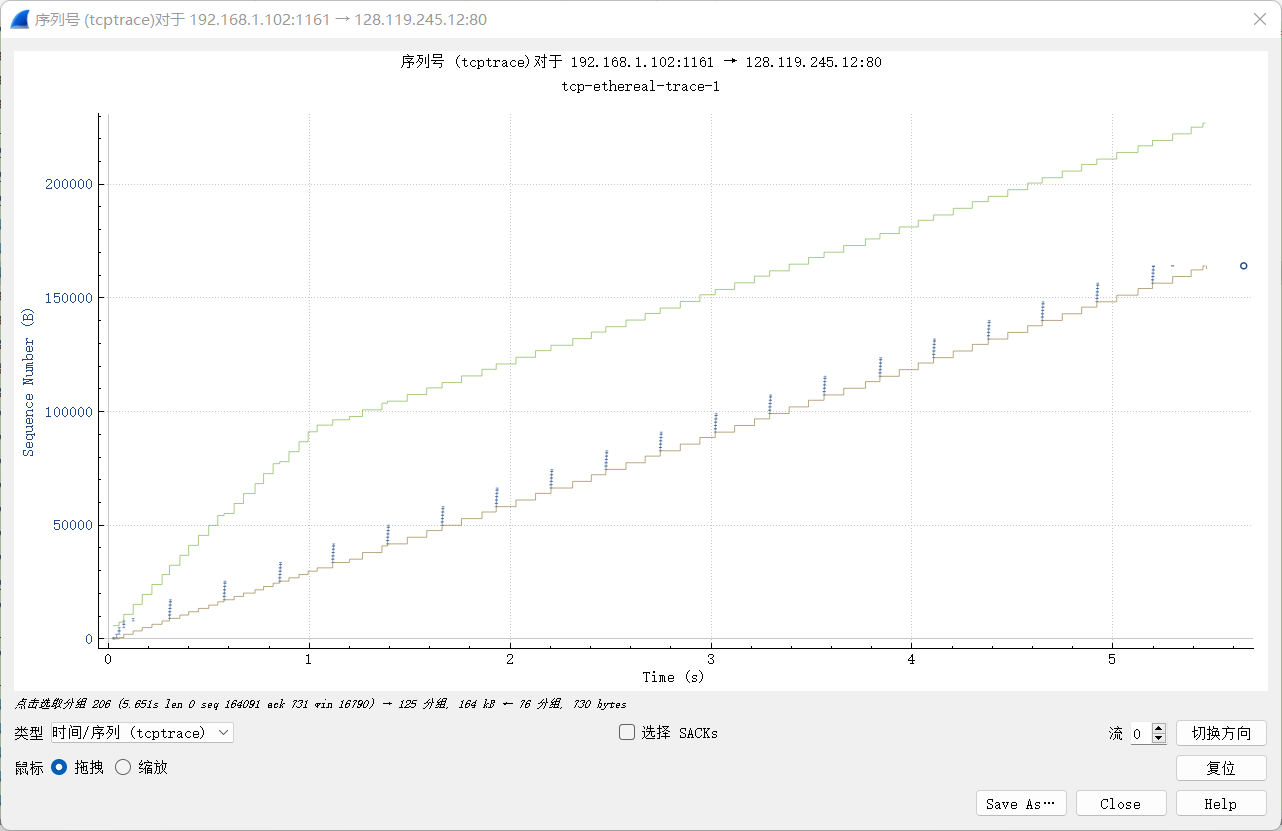
**Ans：**Minimum amount of available buffer：5840；由于TCP的流量控制机制，当接收窗口大小较小时发送方的传输速率会下降，但在上述过程中，接收窗口始终大于MSS，故不会限制发送方传输速率。



1. **Are there any retransmitted segments in the trace file? What did you check for (in the trace) in order to answer this question?**

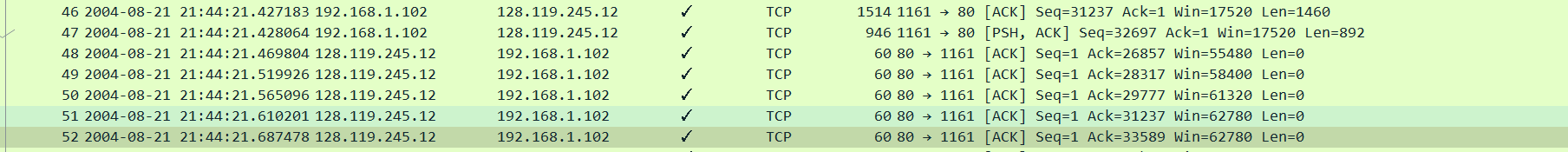
**Ans：**可以观察到TCP流中相应的序列号一直增大，并未出现重复的情况，因此可知并没有出现重传。同样也可以利用TCP 序列号流图形得到上述结论。





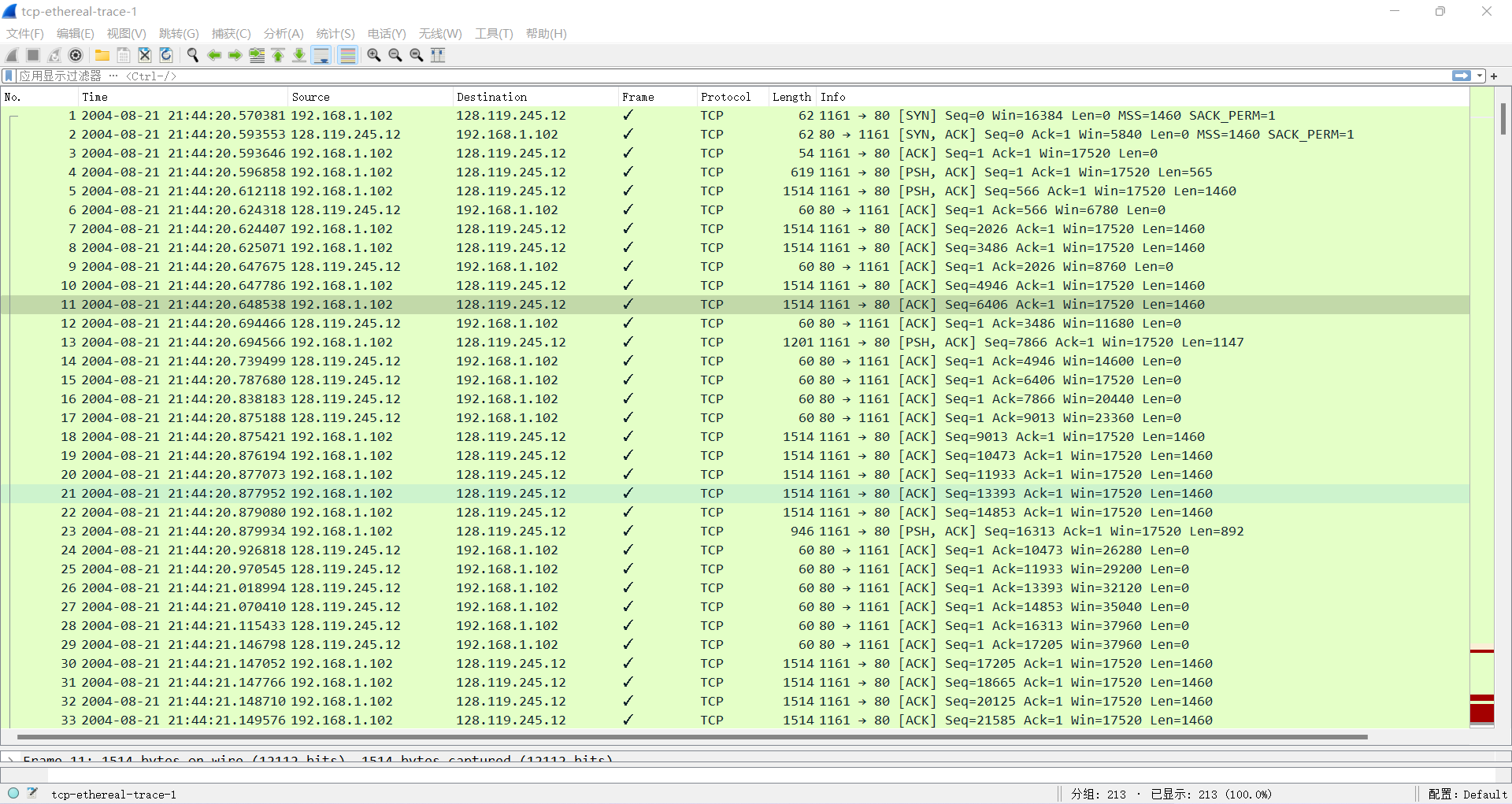
1. **How much data does the receiver typically acknowledge in an ACK? Can you identify cases where the receiver is ACKing every other received segment (see Table 3.2 on page 250 in the text)**

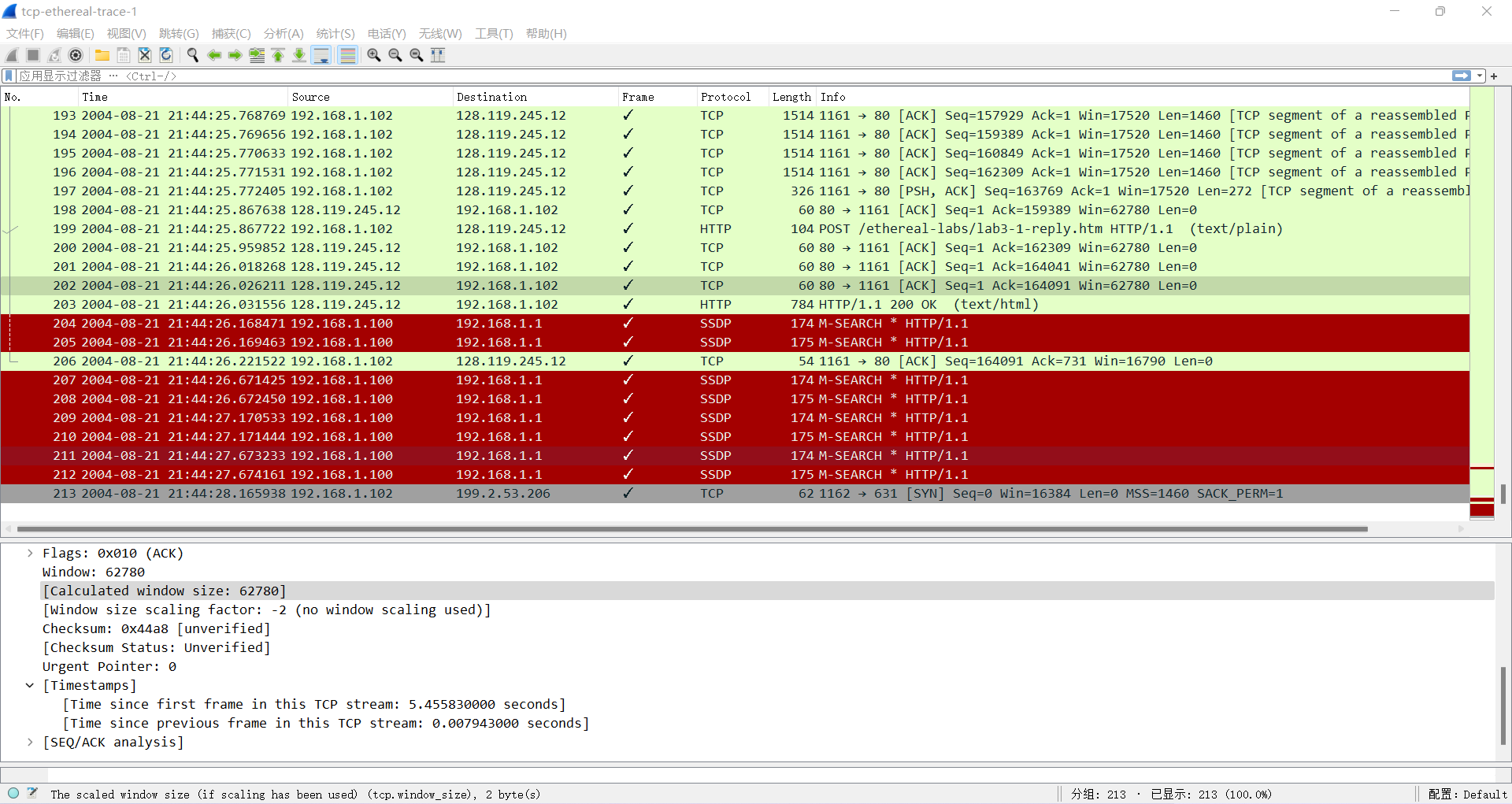
**Ans：**大部分确认为1460 bytes；如第52个报文段就是间隔确认的。

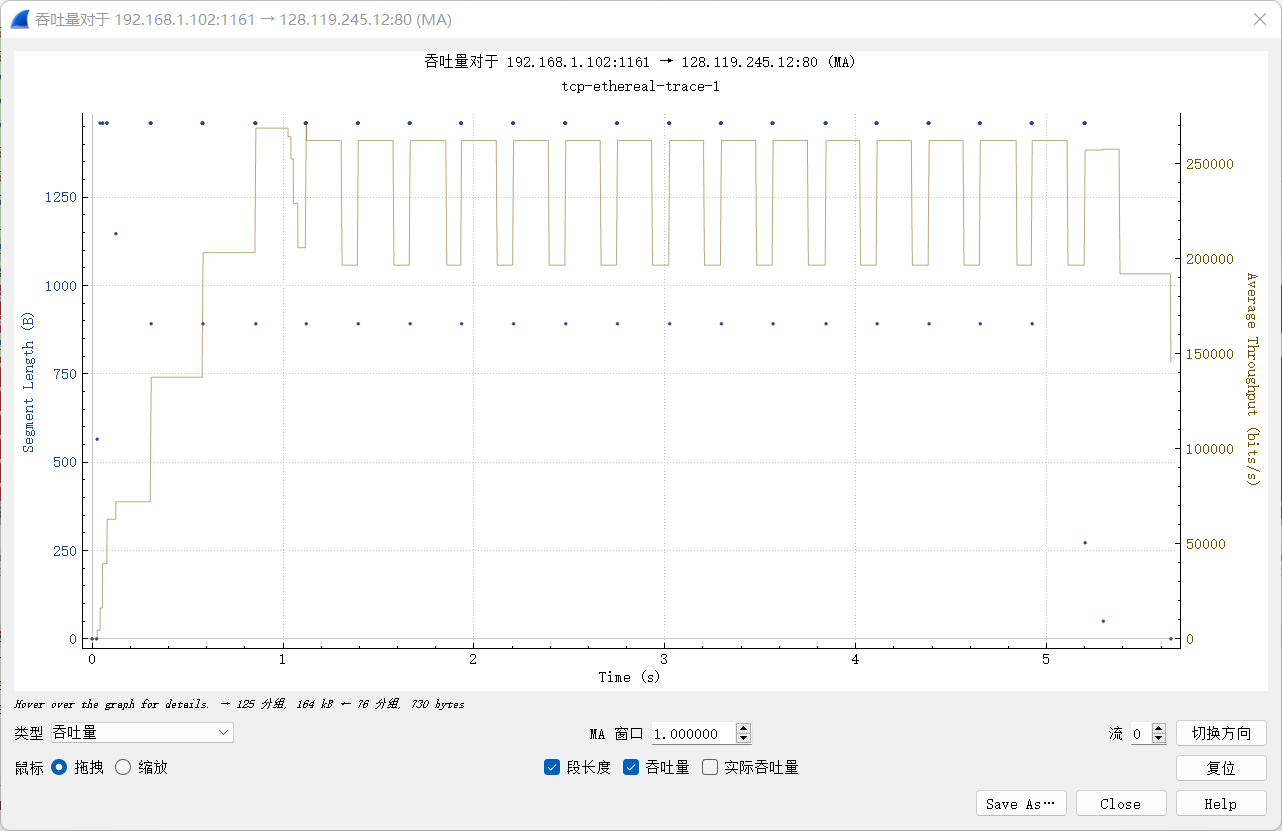


1. **What is the throughput (bytes transferred per unit time) for the TCP connection?Explain how you calculated this value**

**Ans：**传输数据总量为bytes，传输总时间为T= 26.026211- 20.596858=5.429353s。Byte/s。也可以利用图表直接得出。







**（图1-8 TCP 吞吐量流图形）**

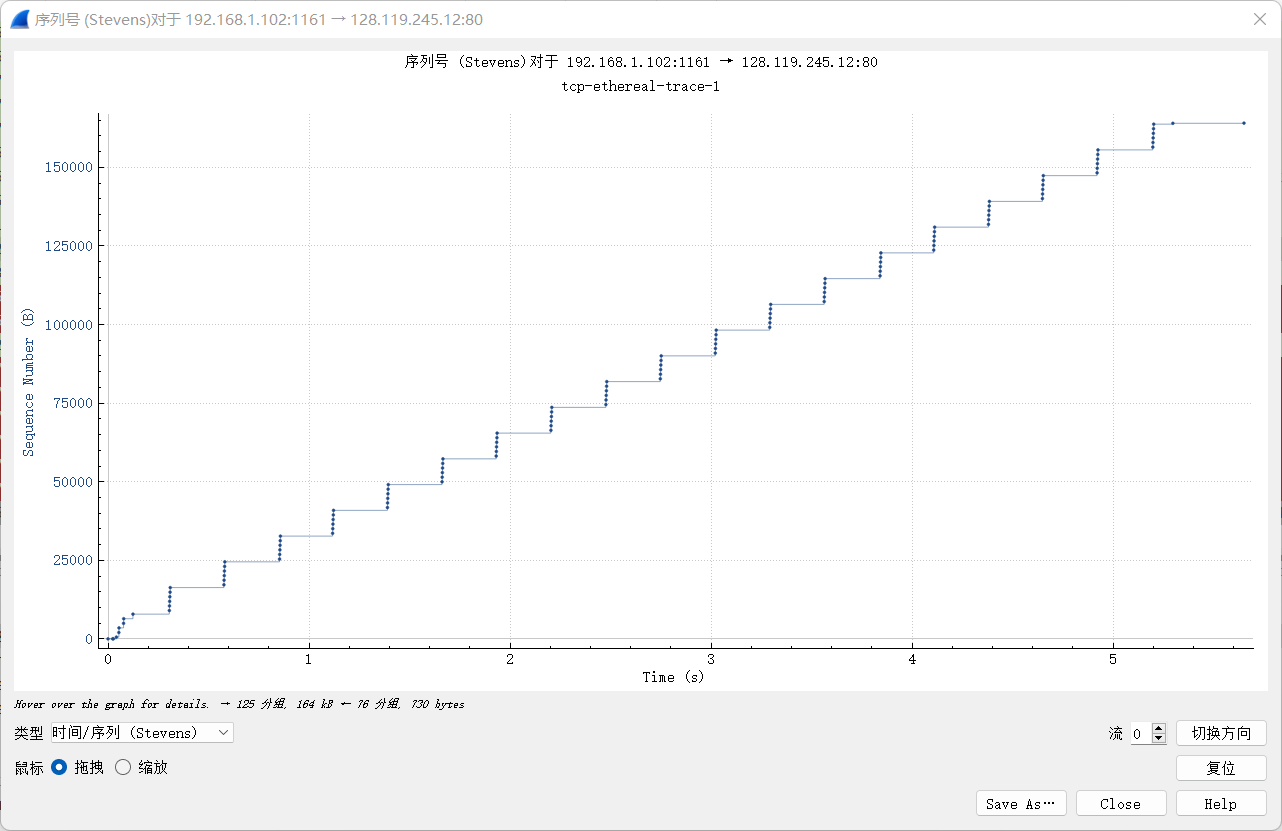
**Part 4：TCP congestion control in action**

1. **实验内容：**

分析TCP数据包，观察TCP拥塞控制机制。

1. **实验流程：**

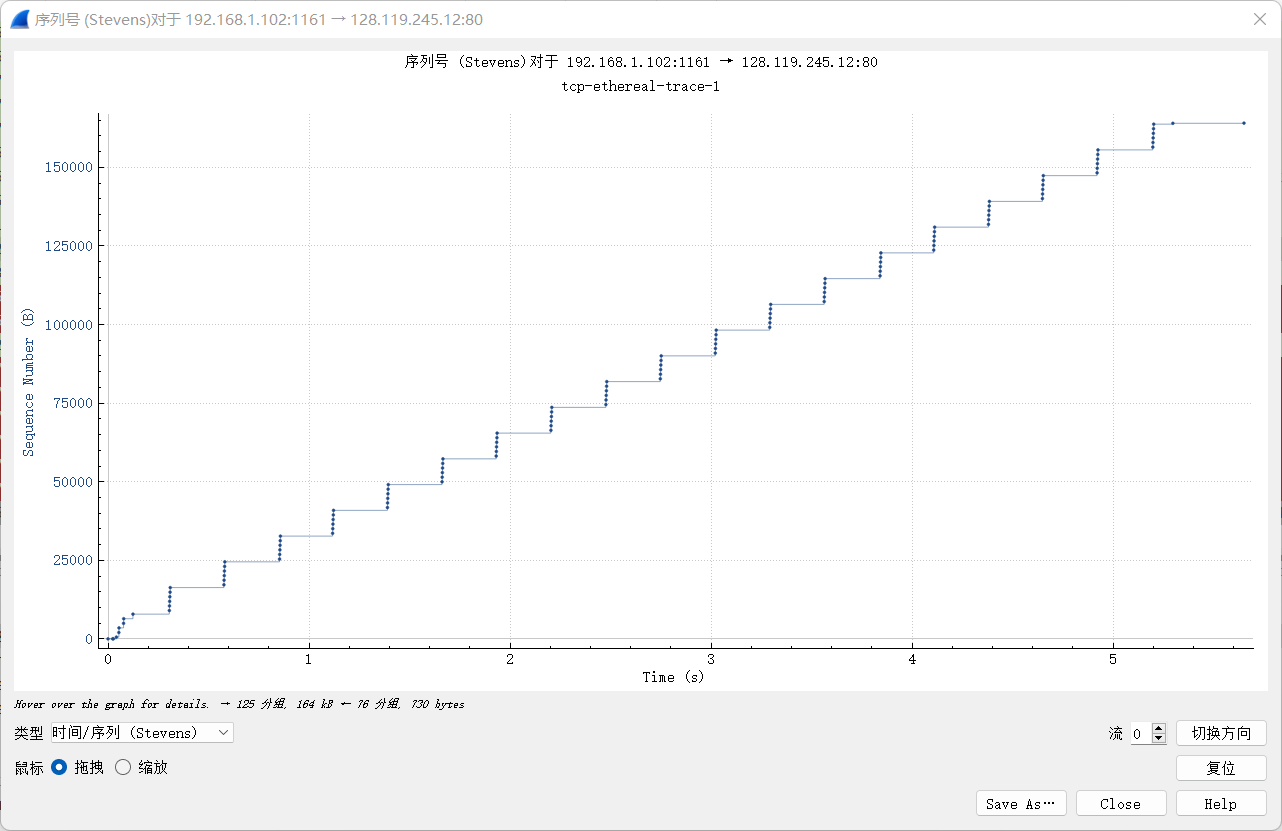
利用Wireshark制图，得到如图所示时间序列：

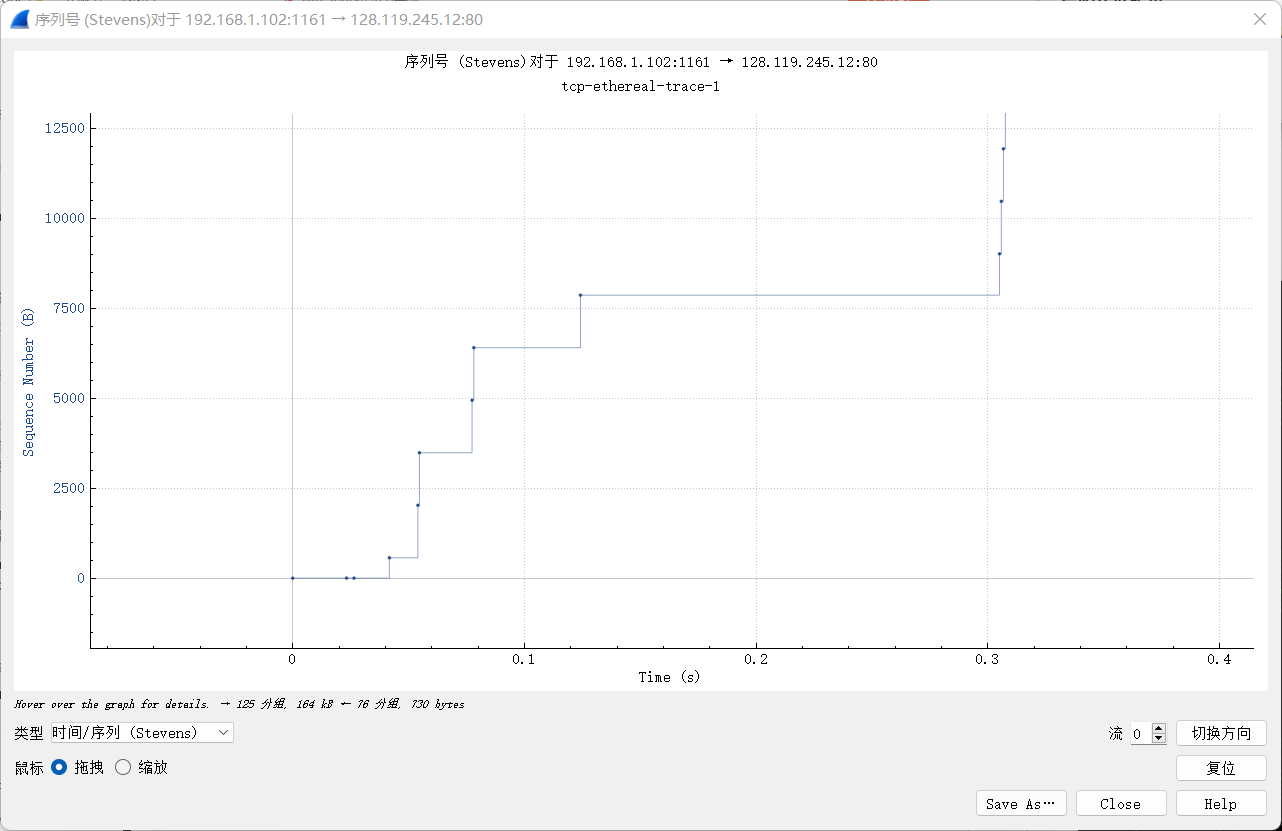


**（图1-9 Stevens TCP流图形）**

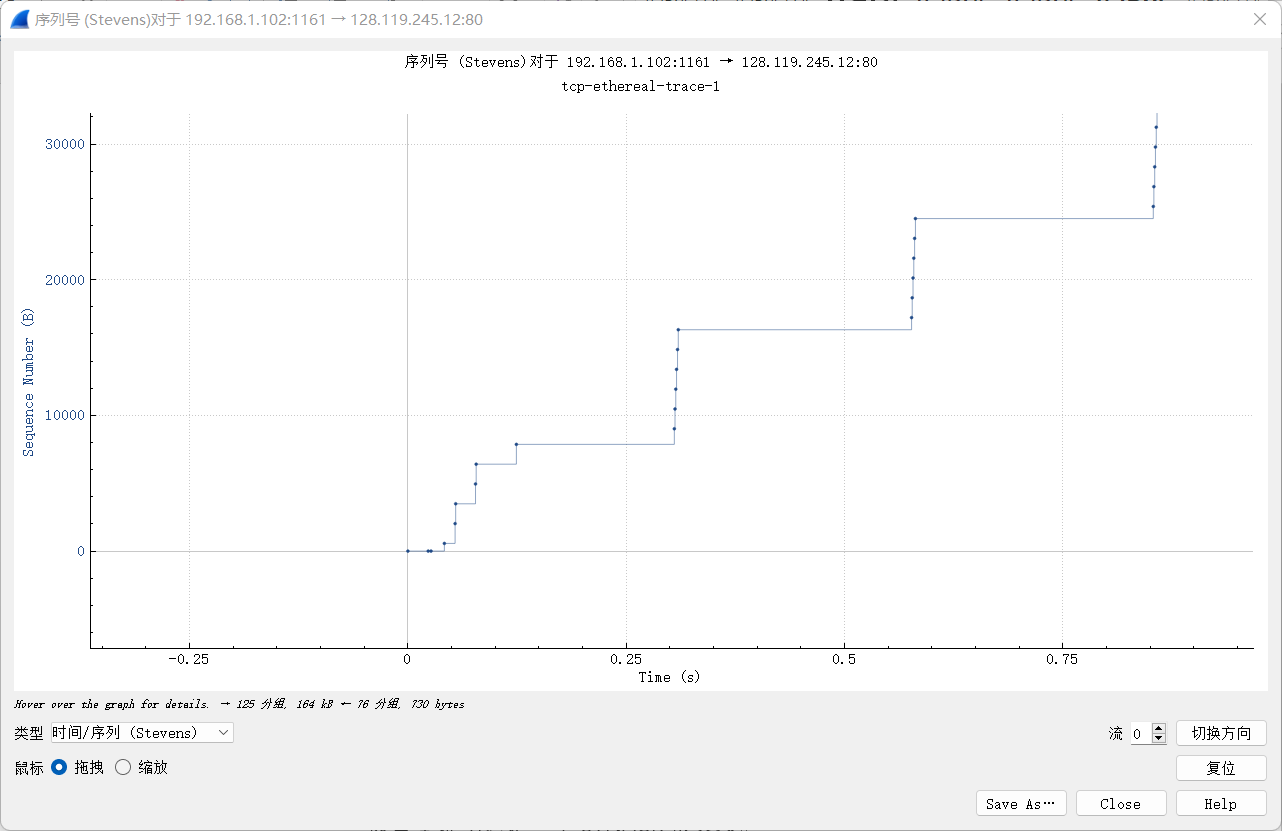
1. **Q&A(以下内容利用所给trace分析)：**
2. **Use the Time-Sequence-Graph(Stevens) plotting tool to view the sequence number versus time plot of segments being sent from the client to the gaia.cs.umass.edu server. Can you identify where TCP’s slowstart phase begins and ends, and where congestion avoidance takes over? Comment on ways in which the measured data differs from the idealized behavior of TCP that we’ve studied in the text?**

**Ans：**慢启动从TCP连接建立完成后发送第一个分组（分组5）开始，在第13个分组之后不再呈指数增长，慢启动结束，拥塞避免开始。观察可知在后续传输过程中每次均发送六个报文段，说明cwnd遇到了ssthresh，进入congestion avoidance阶段，cwnd不再增加，但是书上写得理想机制是会线性增加直到遇到超时，可能这里存在不同的控制机制。此外，在慢启动阶段cwnd并不是严格按照指数增长的，与课本上的理想情况存在一定的差别。





Slow-start

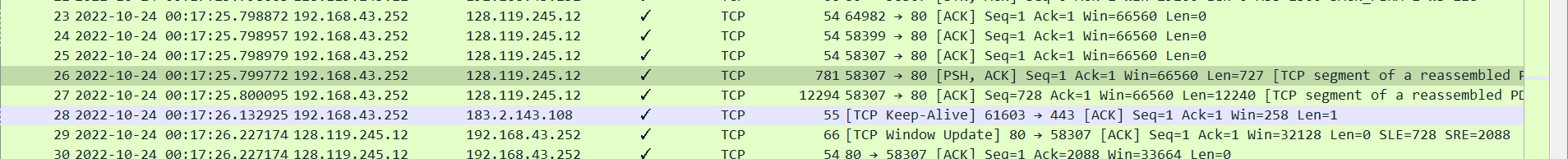


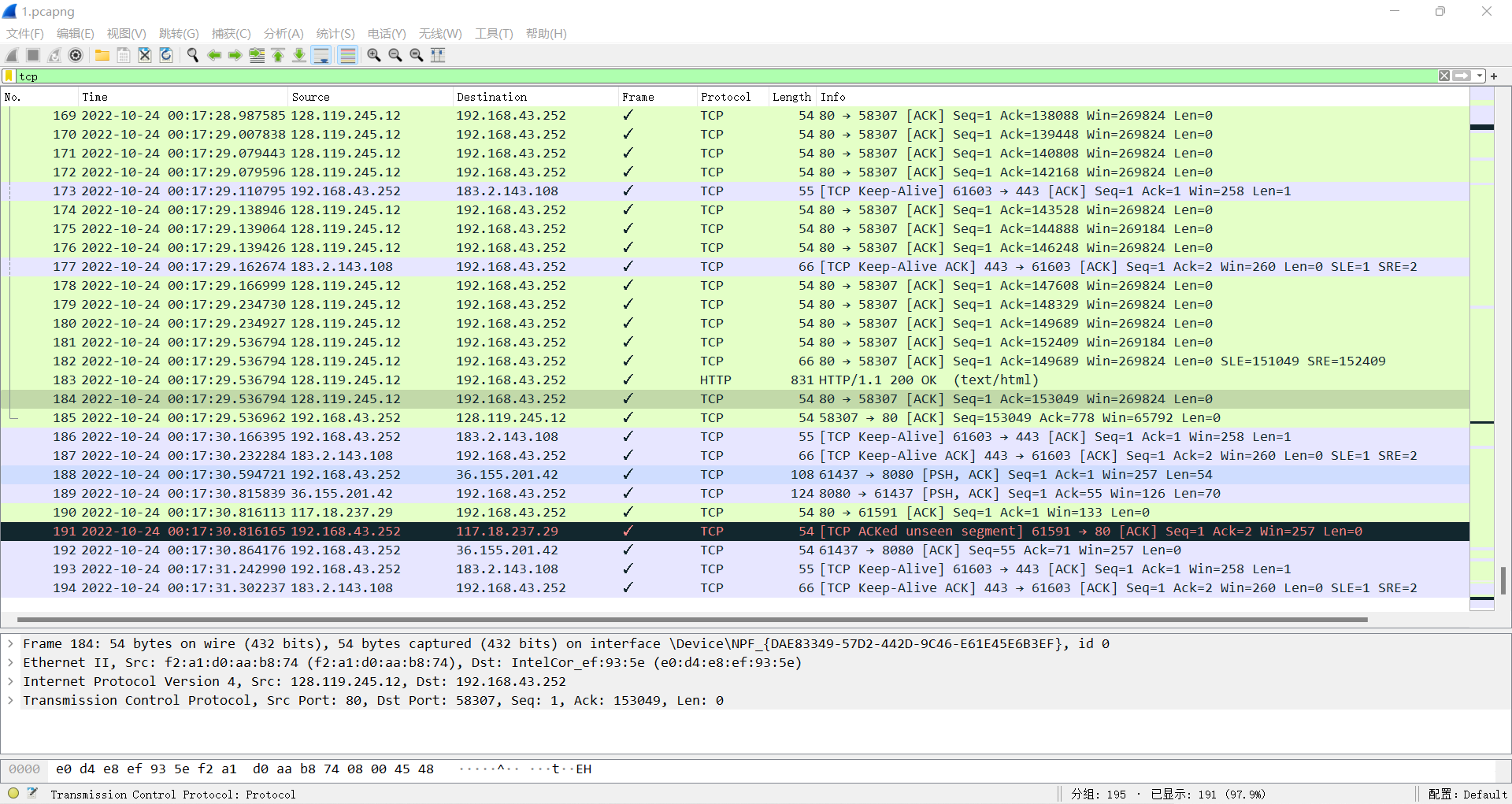
拥塞避免

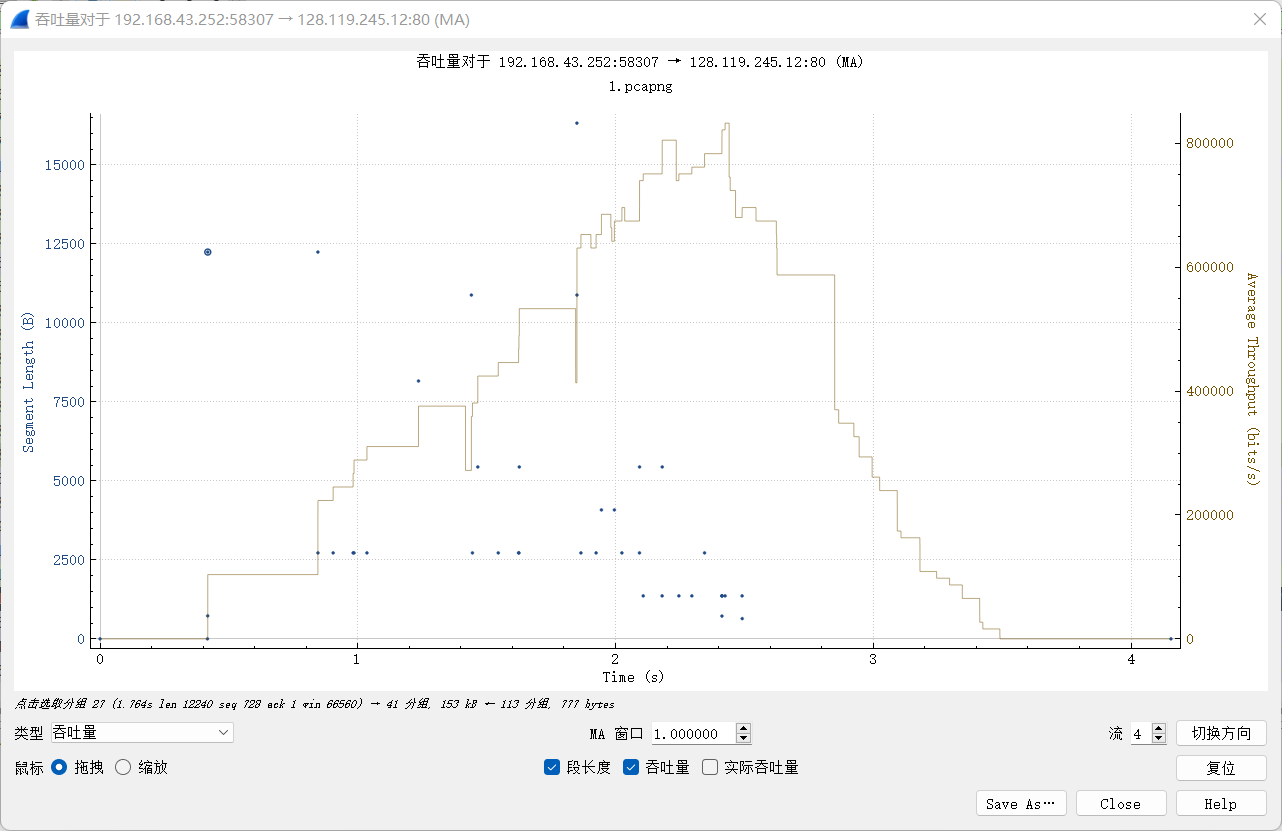
Slow-start end

1. **Answer each of two questions above for the trace that you have gathered when you transferred a file from your computer to gaia.cs.umass.edu**

**Ans：**传输数据总量为bytes，传输总时间为T= 29.536794-25.799772=3.737022s。Byte/s。也可以利用图表直接得出。

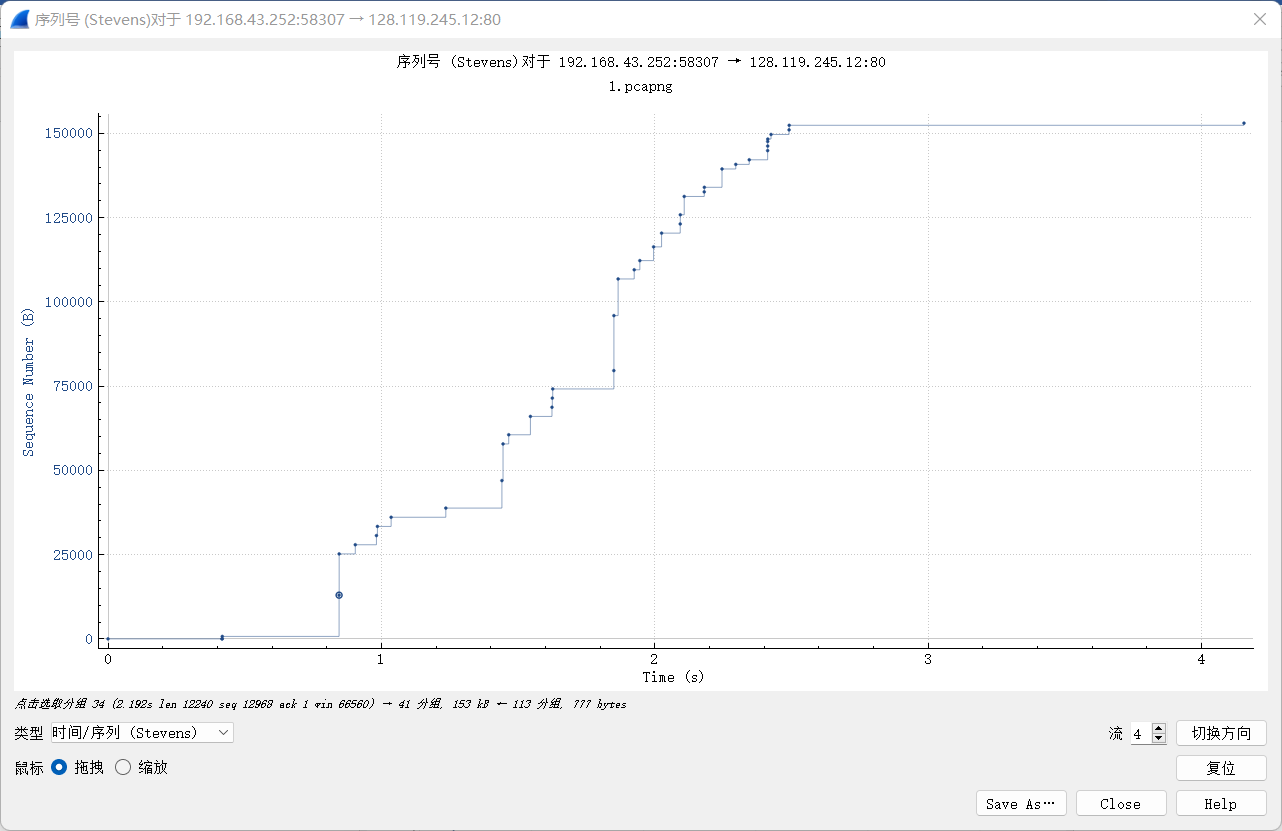






**（图1-10 TCP 吞吐量流图形）**

**Ans：**慢启动从TCP连接建立完成后发送第一个分组（分组27）开始，在第36个分组之后不再呈指数增长，慢启动结束，拥塞避免开始。可以比所给trace看到更为明显的拥塞控制过程。



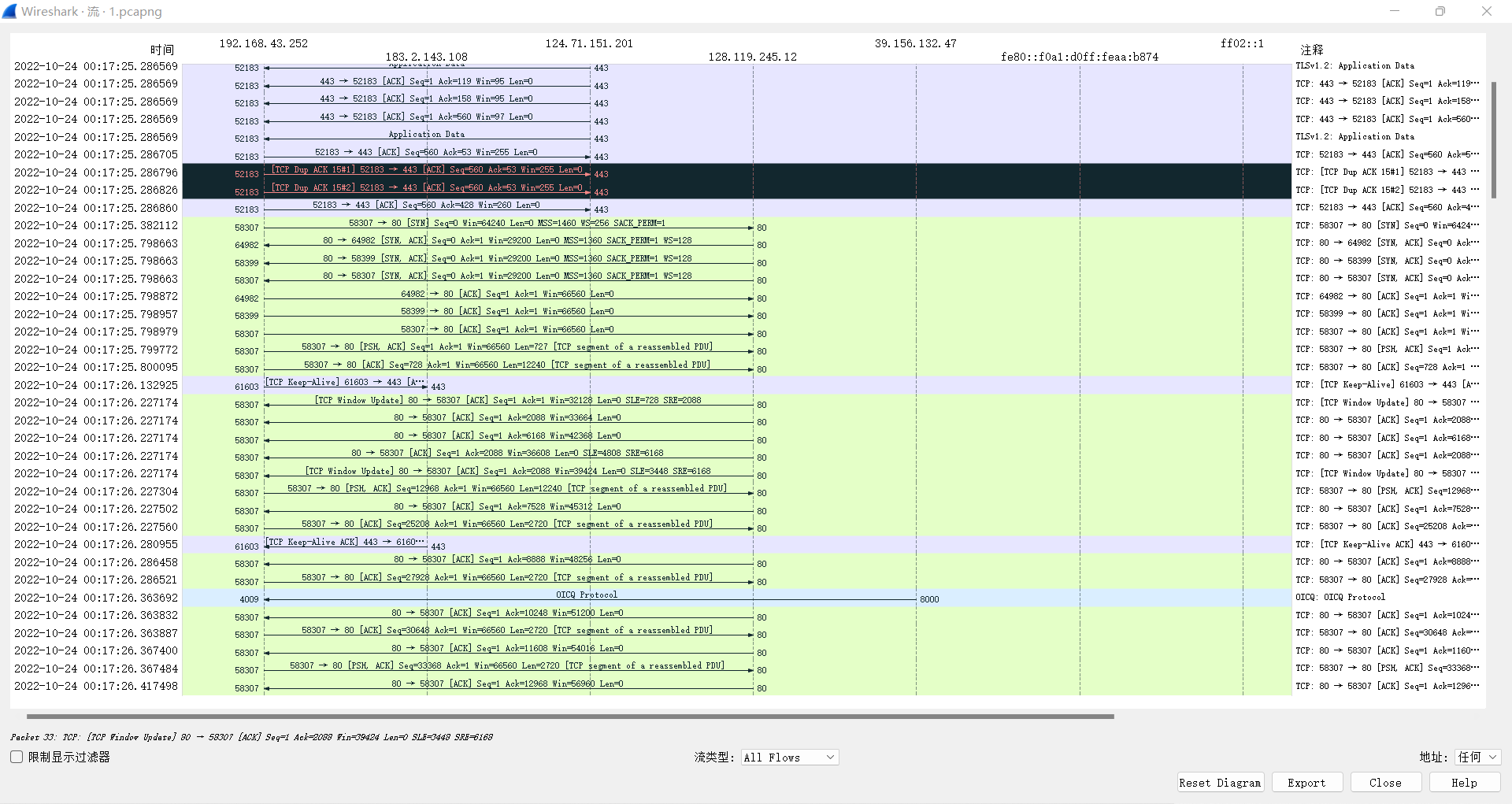
Slow-start

Slow-start end

拥塞避免

**补充内容：**

利用Wireshark我们也可以看到TCP流量图，以实验中所得数据为例，可以得到如下流量图：



**（图1-11 TCP流量图）**

除了三次握手，在结束连接时TCP也具有四次挥手过程：

由于TCP连接是全双工的，因此，每个方向都必须要单独进行关闭。在一方完成数据发送任务后，将会发送一个FIN来终止本方连接，但是在这个TCP连接上仍然能够发送数据，直到接收方也发送了FIN。首先进行关闭的一方将执行主动关闭，而另一方则执行被动关闭。

（1）第一次挥手：客户端发送一个FIN，用来关闭客户机到服务器的数据传送，客户端进入FIN\_WAIT\_1状态。

（2）第二次挥手：服务器收到FIN后，发送一个ACK给客户端，确认序号为收到序号+1，服务器进入CLOSE\_WAIT状态。

（3）第三次挥手：服务器发送一个FIN，用来关闭服务器到客户端的数据传送，服务器进入LAST\_ACK状态。

（4）第四次挥手：客户端收入FIN后，客户端进入TIME\_WAIT状态，接着发送一个ACK给服务器，确认序号为收到序号+ 1，服务器进入CLOSED状态，完成四次挥手。

**成功！您的提交显示在此页面上。该提交的确认编号为 476899eb-8f53-4d94-b8bc-793ae67d4af2。请复制并保存此编号以作为提交证明。**[**在“我的成绩”中查看您的所有提交确认。**](https://www.bb.ustc.edu.cn/webapps/bb-mygrades-BBLEARN/myGrades?course_id=_12565_1&stream_name=mygrades&is_stream=false)