****



**WireShark TCP实验报告**

**学院： 智能与计算学部**

**专业： 计算机科学与技术**

**姓名： 陈秋澄**

**学号： 3022244290**

**2024年4月5日**

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

按照要求依次访问http://gaia.cs.umass.edu/wiresharklabs/alice.txt与<http://gaia.cs.umass.edu/wireshark-labs/TCP-wireshark-file1.html>

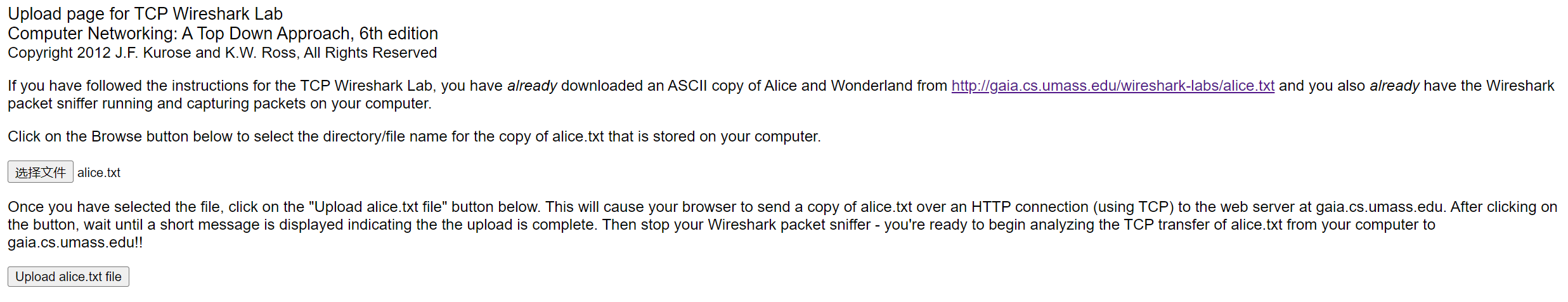


图1-a 操作佐证a

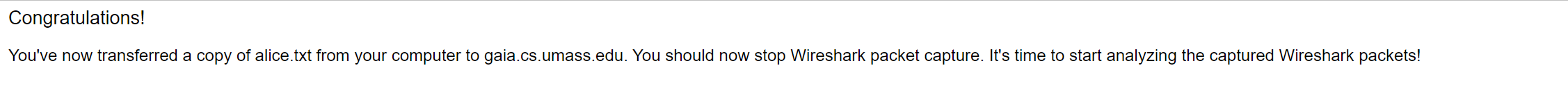


图1-b 操作佐证b

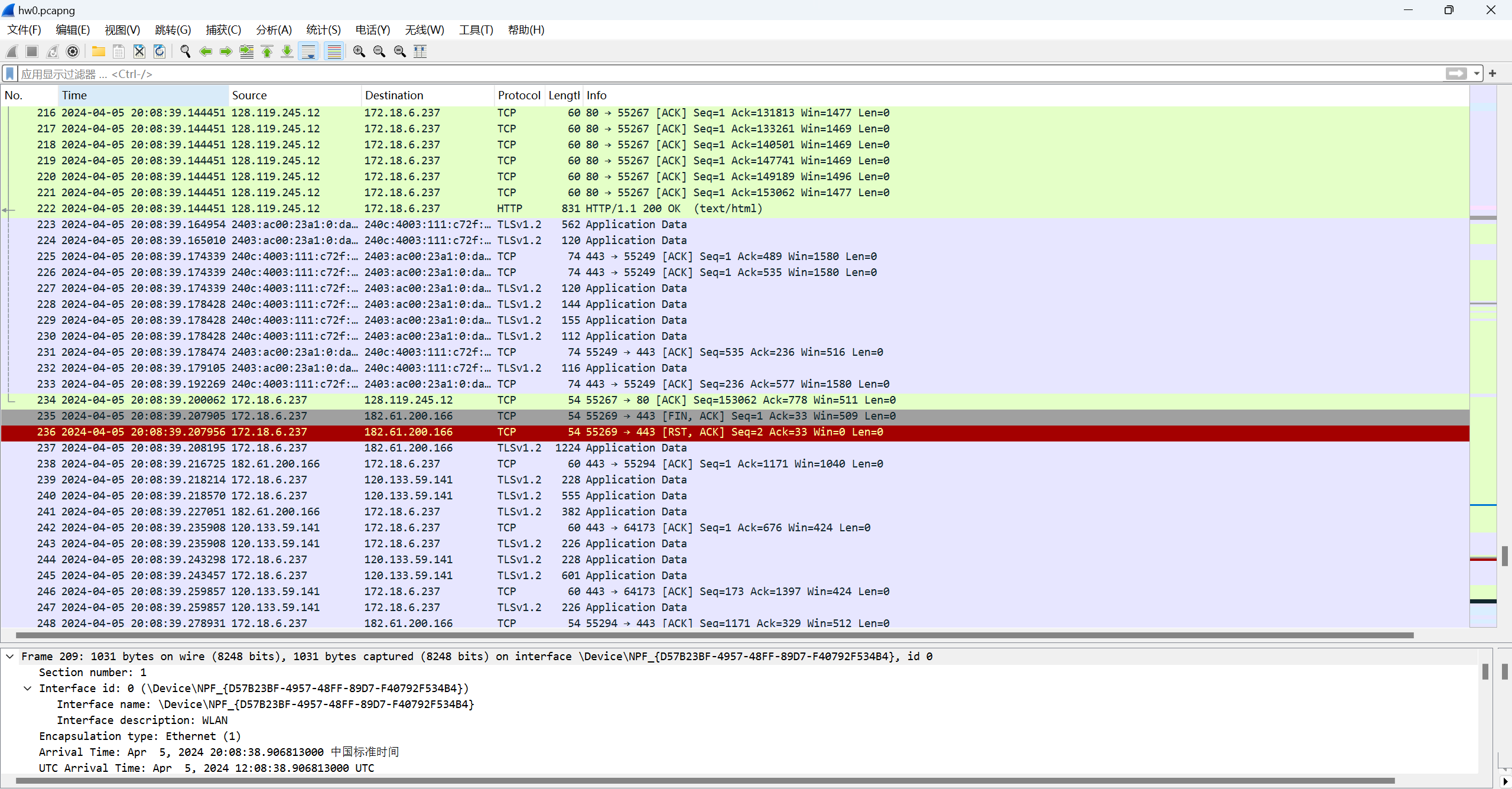


图1-c 操作佐证c

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

**1. 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? To answer this question, it’s probably easiest to select an HTTP message and explore the details of the TCP packet used to carry this HTTP message, using the “details of the selected packet header window” (refer to Figure 2 in the “Getting Started with Wireshark” Lab if you’re uncertain about the Wireshark windows.**

由图2-a，打开tcp-ethereal-trace-1文件，客户机（src）使用的IP地址为：192.168.1.102；

通过图2-b的Source Port可知，客户机（src）使用的TCP端口号为：1161。

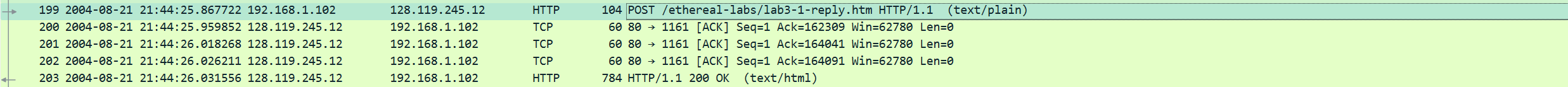


图2-a Question 1&2佐证a

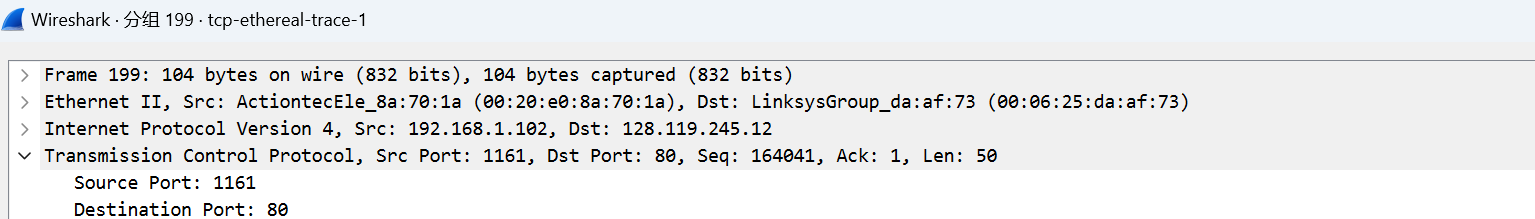
****

图2-b Question 1&2佐证b

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

由上图2-a可知gaia.cs.umass.edu（dst）使用的IP地址为：128.119.245.12；

由上图2-b 的Destination Port，gaia.cs.umass.edu（dst）使用的接收TCP段的端口号为：80。

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

通过图3-a的209行得知，客户机（src）使用的IP地址为：172.18.6.237；

通过图3-b 的Source Port: 55267得知，客户机（src）使用的TCP端口号为：55267。

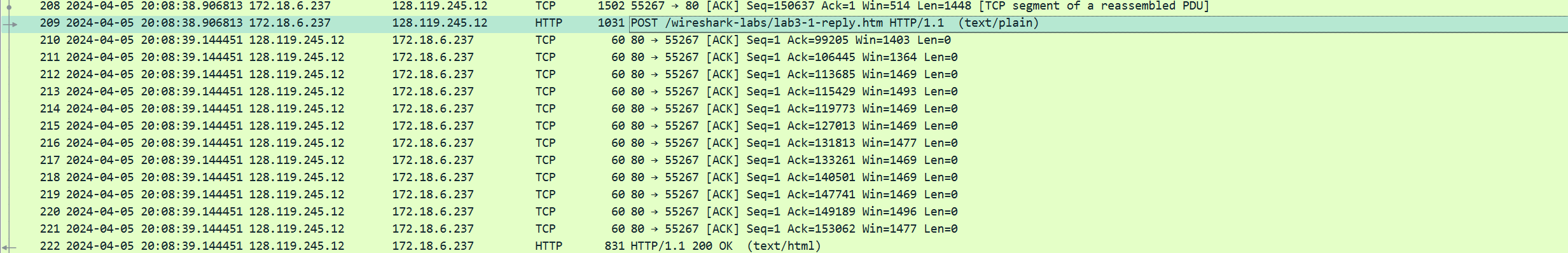


图3-a Question 3佐证a

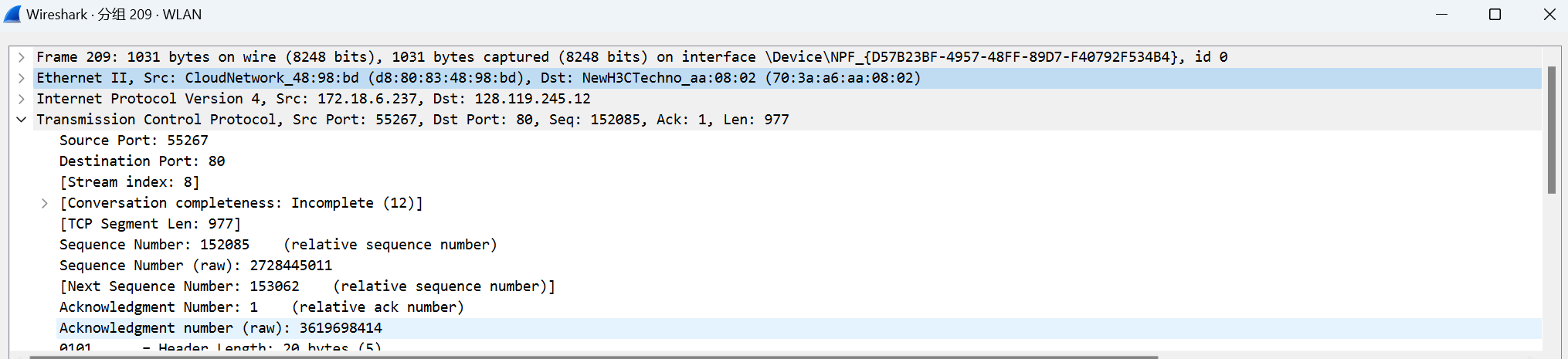


图3-b Question 3佐证b

**TCP Basics**

**4. 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?**



图4-a Question 4佐证a

由图4-a可知，用于启动客户端计算机和gaia.cs.umass.edu之间的TCP SYN段的序列号是0。

由图4- b可知，该段中的Syn标志可以确认该段为SYN段（通过设置Flags为0x002，即Syn位设为1）。

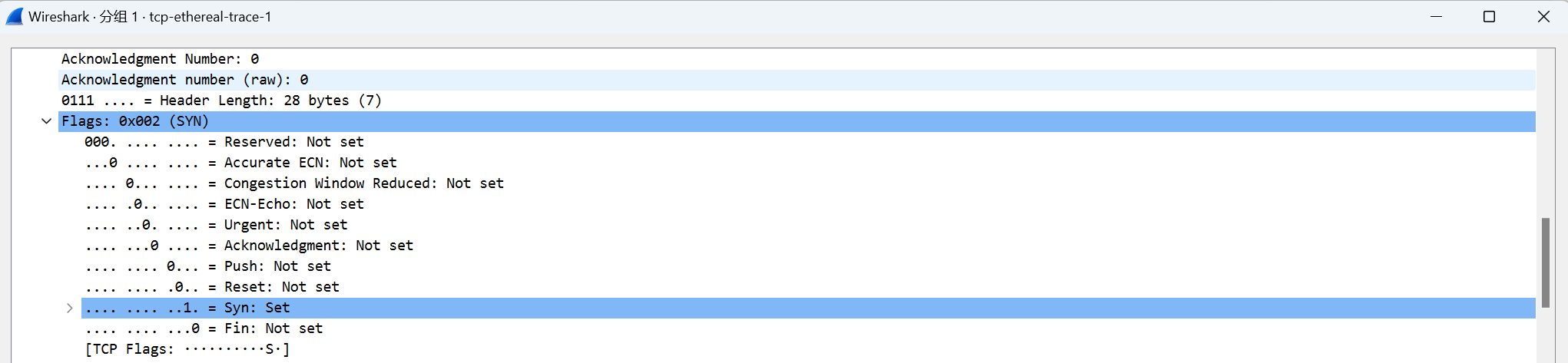


图4-b Question 4佐证b

**5. 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?**

由图5的Sequence Number可知，SYNACK的序列号是0；

由Acknowledgement Number可知，Acknowledgement字段的值为1；

服务器会根据客户端发来的SYN信息段确认该值，将客户端发来的末尾序列号加一（之前的连续数据必须全部收到），填充进报文段的确认号，是主机期望从对方主机收到的下一字节的序号；

从该段的标志位可以确认该段是一个SYNACK段（通过设置Flags为0x012，即Syn及Acknowledgment位均设为1）。

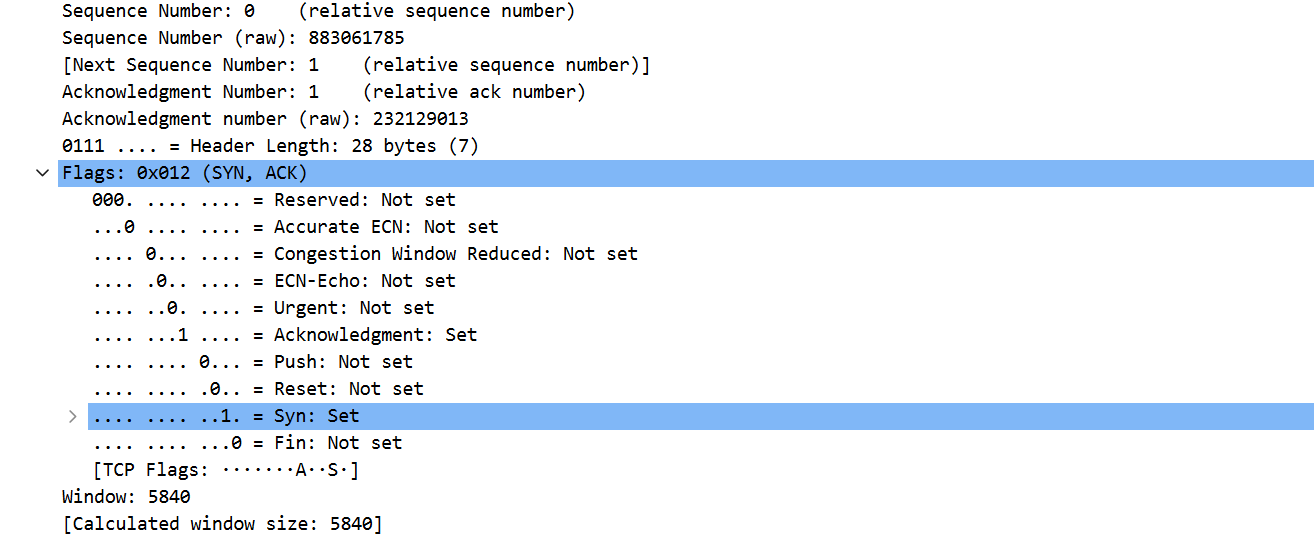
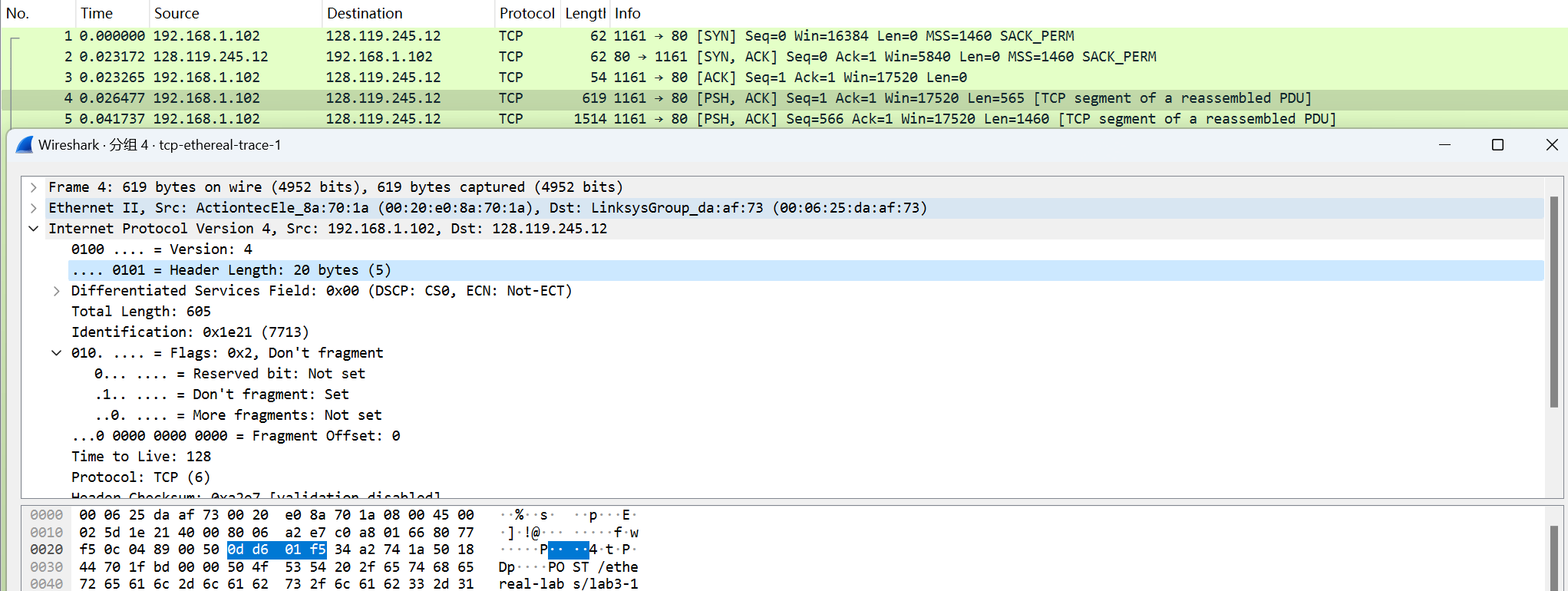


图5 Question 5佐证

**6. 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.**

包含HTTP POST命令的TCP段的序列号为1。

因为含有POST 字段的为最后的一次握手字段，作为客户机最后一次三次握手确认，还包含了要发送的字段，因此序列号为1（因为是连着上面的序列号0连续发的）。



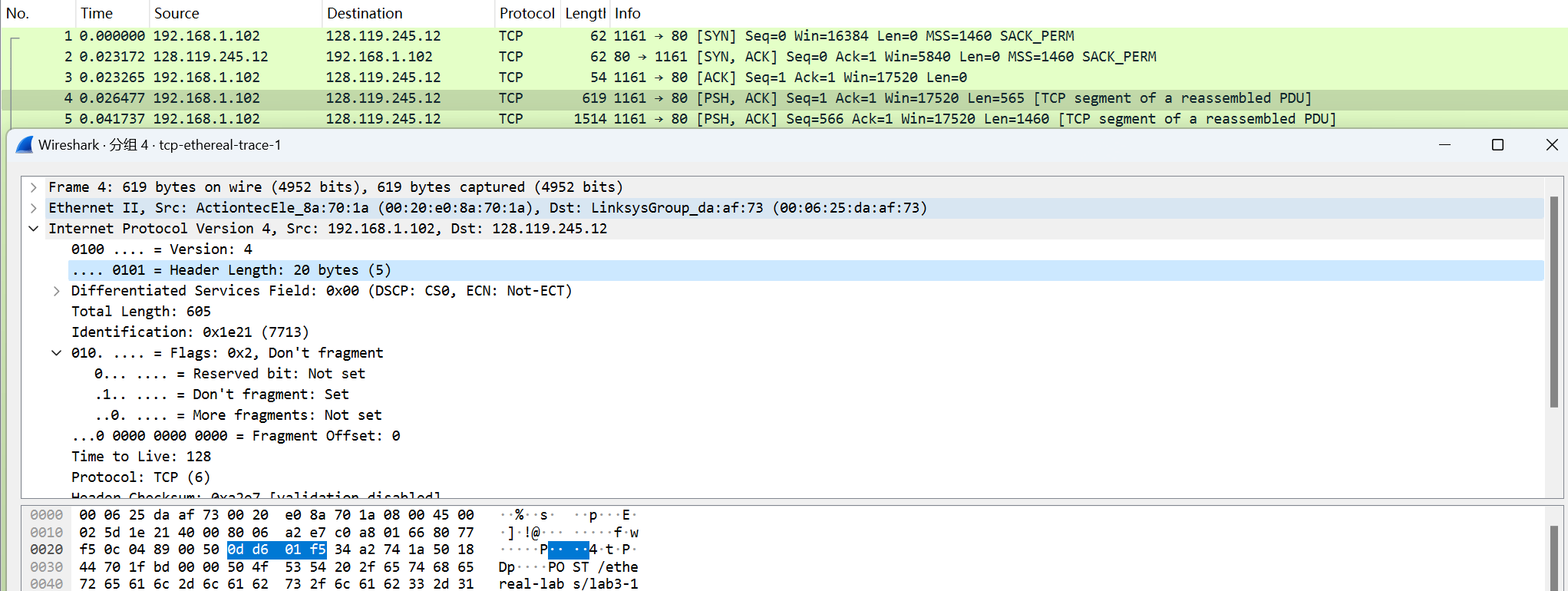


图6 Question 6佐证

**7. 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.**

如上图所示，TCP连接中前六个段的序号列是1、566、2026、3486、4946与6406。

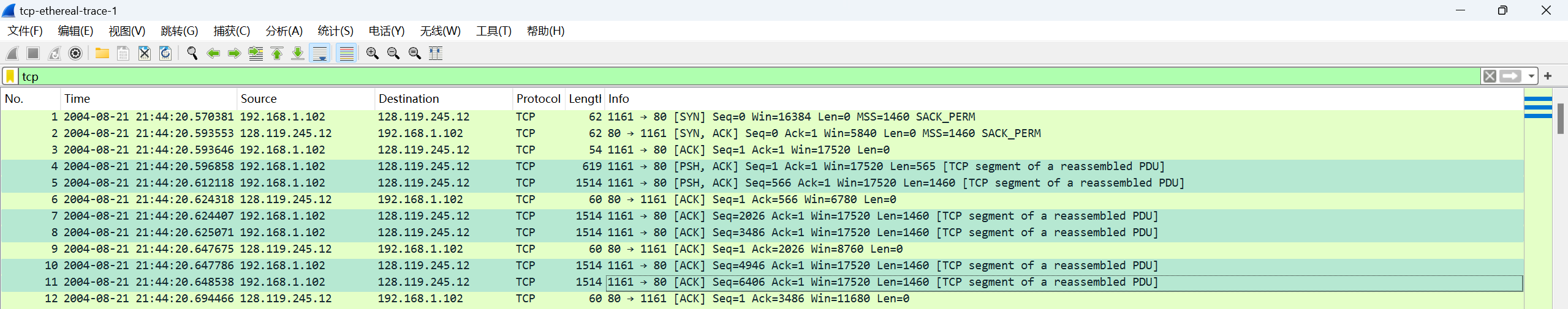
****

图7-a Question 7佐证a

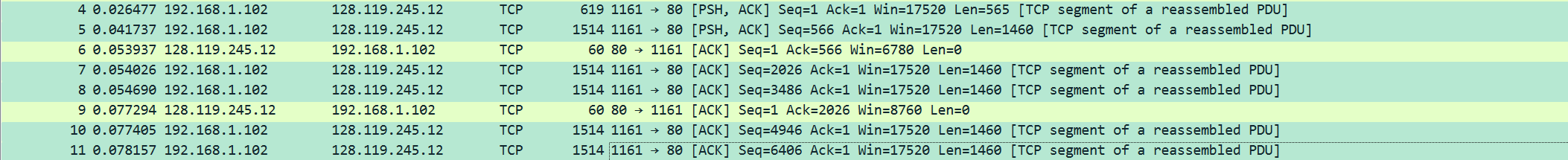


图7-b Question 7佐证b(发送时间)

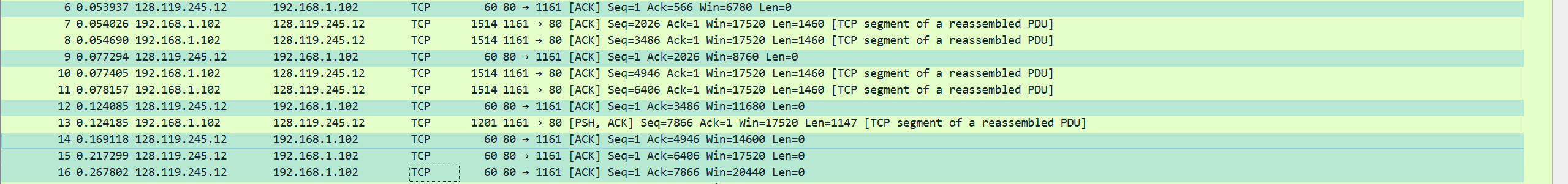


图7-c Question 7佐证c(ACK收到时间)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 序列号 | 发送时间 | ACK收到时间 | RTT |
| 1 | 1 | 0.026477 | 0.053937 | 0.027460 |
| 2 | 566 | 0.041737 | 0.077294 | 0.035557 |
| 3 | 2026 | 0.054026 | 0.124085 | 0.070059 |
| 4 | 3486 | 0.054690 | 0.169118 | 0.114428 |
| 5 | 4946 | 0.077405 | 0.217299 | 0.139894 |
| 6 | 6406 | 0.078157 | 0.267802 | 0.189645 |

根据公式：

取为[RFC 6298]中推荐的值：0.125，初始的为第一个网段测量的RTT为0.027460。则后续的计算公式为：

例如，= 0.875∗0.02746+0.125∗0.035557=0.028472

计算结果如下：

|  |  |
| --- | --- |
| 序号 | 时间 |
|  | 0.027460 |
|  | 0.028472 |
|  | 0.033670 |
|  | 0.0437651 |
|  | 0.0557812 |
|  | 0.07251424 |

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

根据图8的Len，前六个TCP段的长度分别为：565、1460、1460、1460、1460与1460。

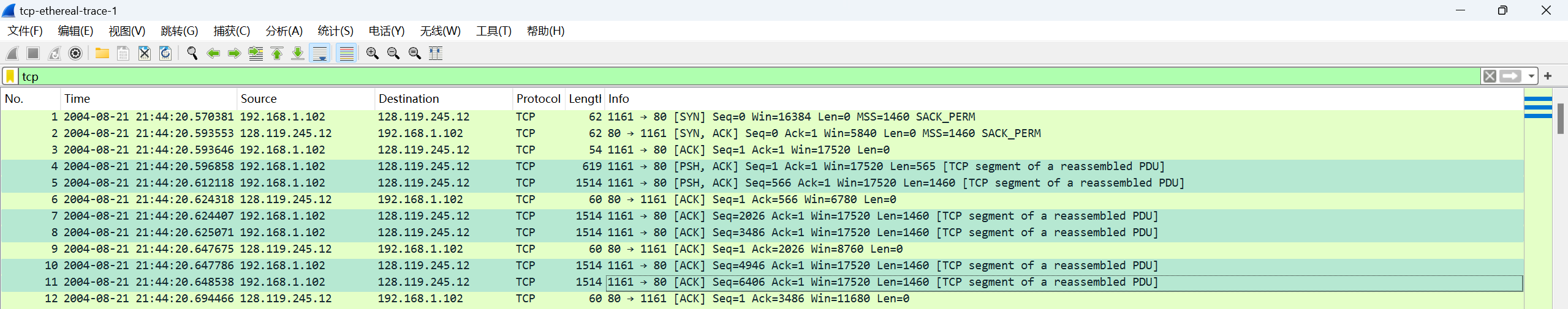
****

图8 Question 8佐证

**9. 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?**

浏览后可知，最小的缓冲区大小（Win）大小是第一次返回SYN时所规定的额，为5840bytes。

包的实际大小小于缓冲区大小，所以缓冲区的大小不会限制sender。

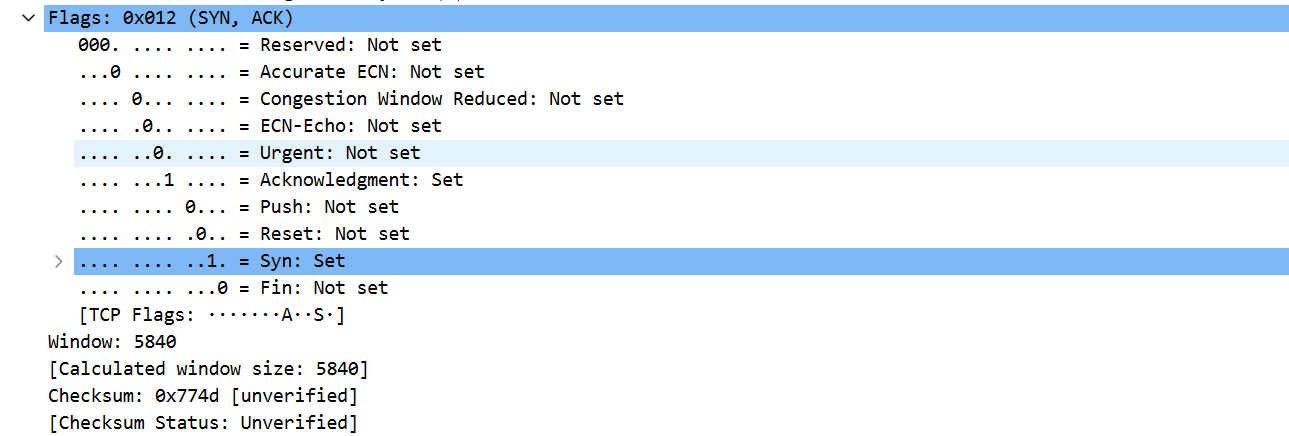


图9 Question 9佐证

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

在整个追踪文件中没有任何重传的片段。

为了回答这个问题，我检查了客户端发出的所有数据的序列号，发现序列号一直处于上涨状态，没有出现重复，说明没有片段被重传。

**11. 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).**

Sender通常在一个ACK 中确认2920长度的数据，在连接确认初始阶段每发送一条就返回一个ACK；在连接确认后client发送的一个段的数据大小一般为1460，在收发稳定之后可以看到通常client一次性发送6个段，服务器返回3个ACK，其中前两个的数值相差是2920。

两个连续ACK的确认序列号之间的差异表示服务器在这两个ACK之间接收的数据。通过检查每个ACK确认的数据量，可能会出现接收方每隔一个数据段进行ACK的情况。例如，第20号数据段，2920字节=1460\*2字节。

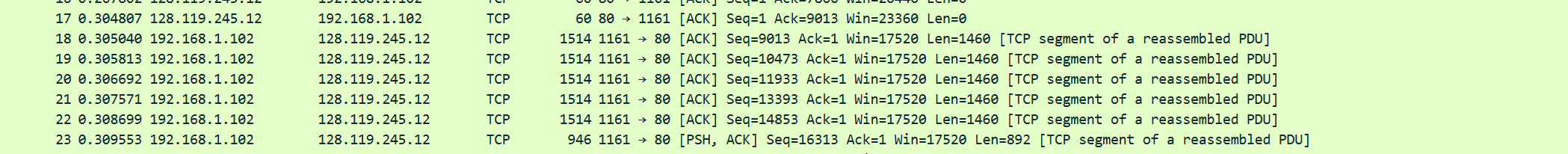


图10 Question 11佐证

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



图11-a Question 12佐证a



图11-b Question 12佐证b

总耗时为最后一个数据包发送时间减去第一个数据包发送时间.

吞吐量大约为164091\*8 /(26.221522-20.596858)=233387.81 bits/s

此外，我们通过统计->TCP流形图->吞吐量来绘制吞吐量曲线可知：在连接稳定阶段，TCP连接的吞吐量最大为 262144bits/s。

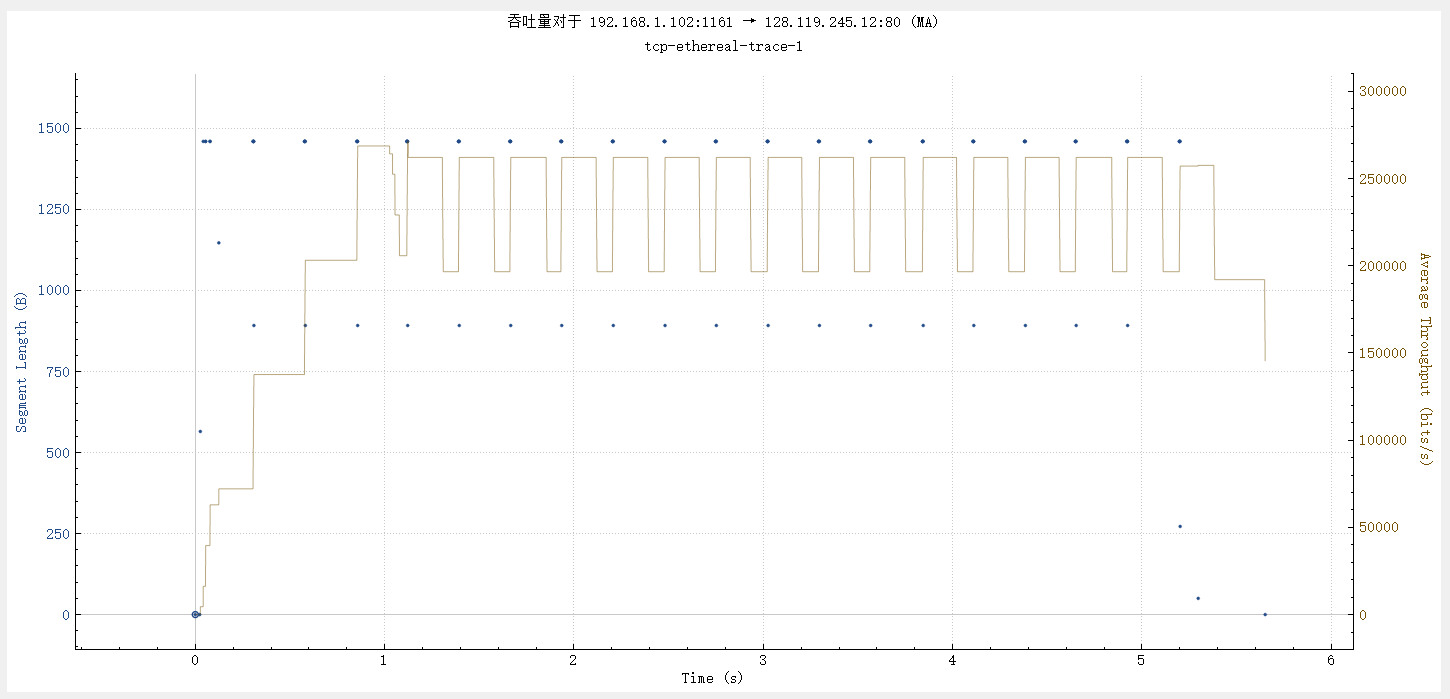


图11-c Question 12图c

**TCP congestion control in action**

**13. 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.**

使用Time-Sequence-Graph(Stevens)画图工具画出从client到server的发送的序列号随时间的变化图像如下所示：

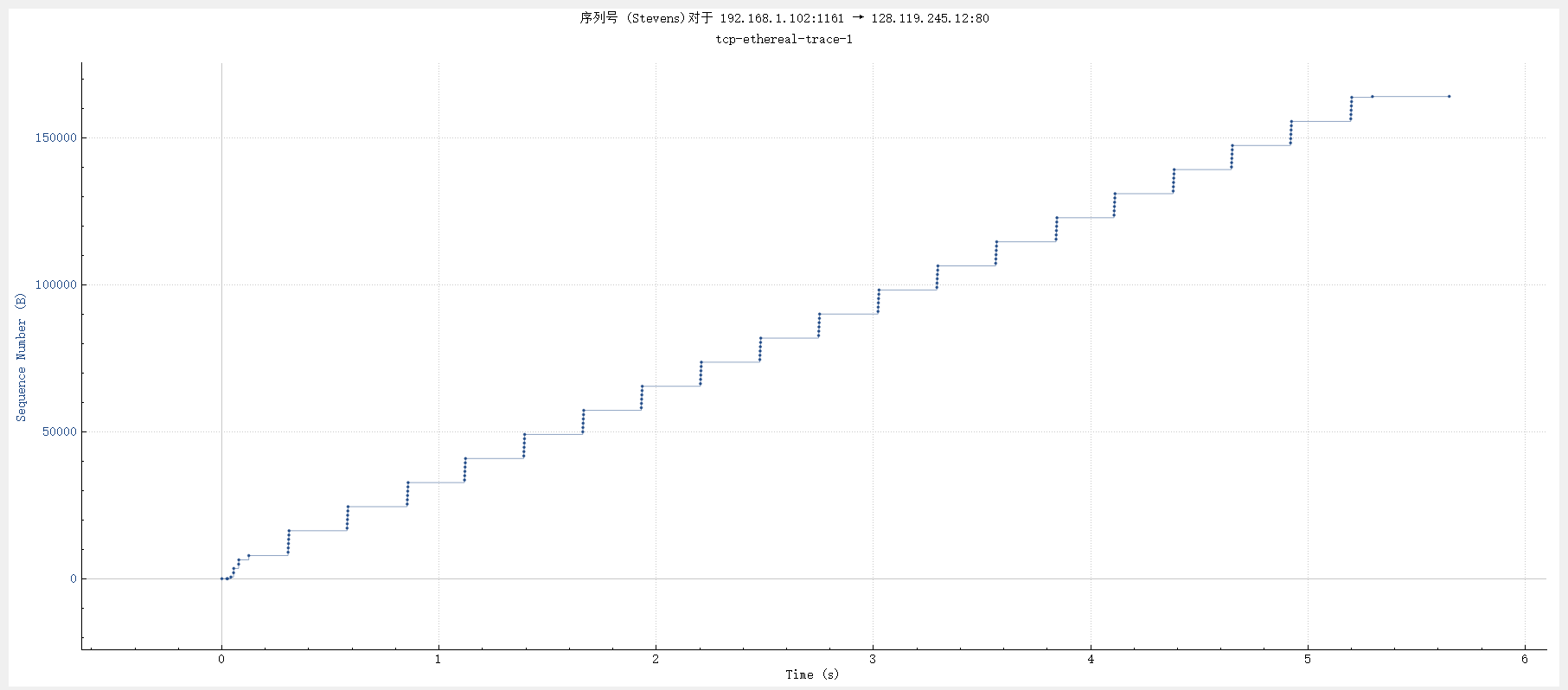


图12-a Question 13佐证a

慢启动阶段如下所示：慢启动从发出了HTTP POST报文段后开始(在第0秒开始)，大概在第0.3秒结束(无法看出慢启动结束都精确时间)。

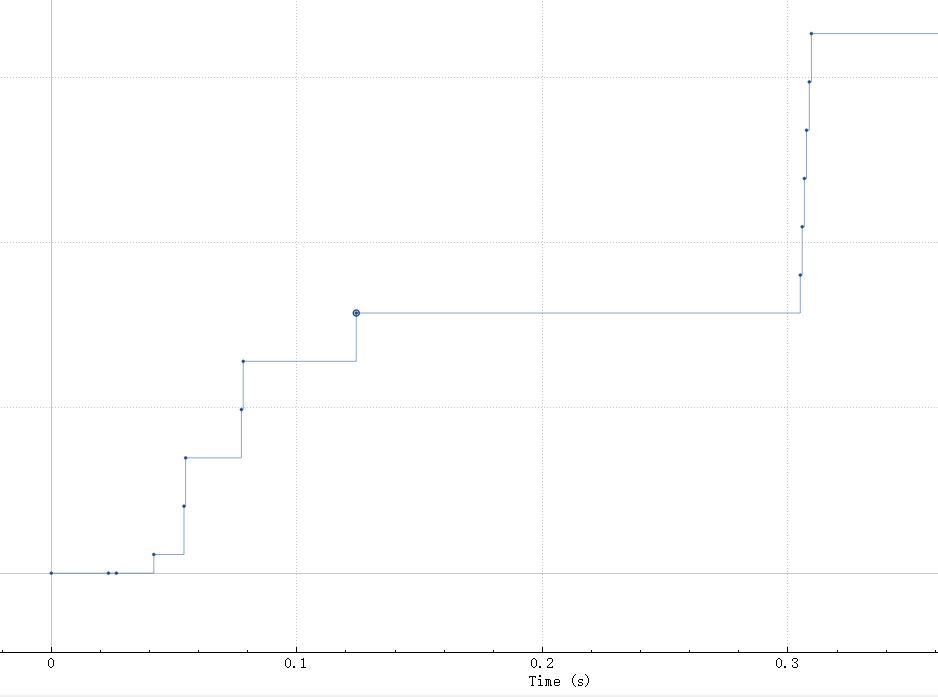


图12-b Question 13佐证b

拥塞避免阶段如下所示，即每发送六个序列号之后进入平台区等待，防止拥塞，停止发送。

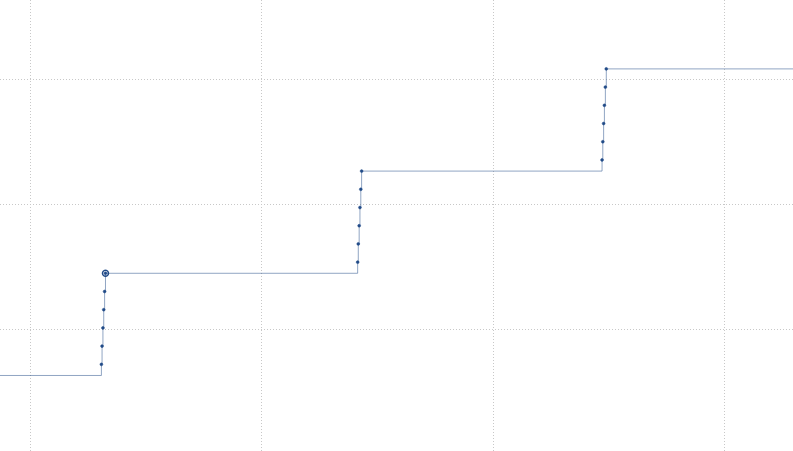


图12-c Question 13佐证c

由于存在拥塞避免机制，实际客户端每发送6个数据段，就会等待一段较长的时间且等待服务器回应的时间远长于发送数据的用时，远远达不到教材中的理想化的流水线式传输。

**14. 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.**

使用Time-Sequence-Graph(Stevens)画图工具画出从client到server的发送的序列号随时间的变化图像如下所示：

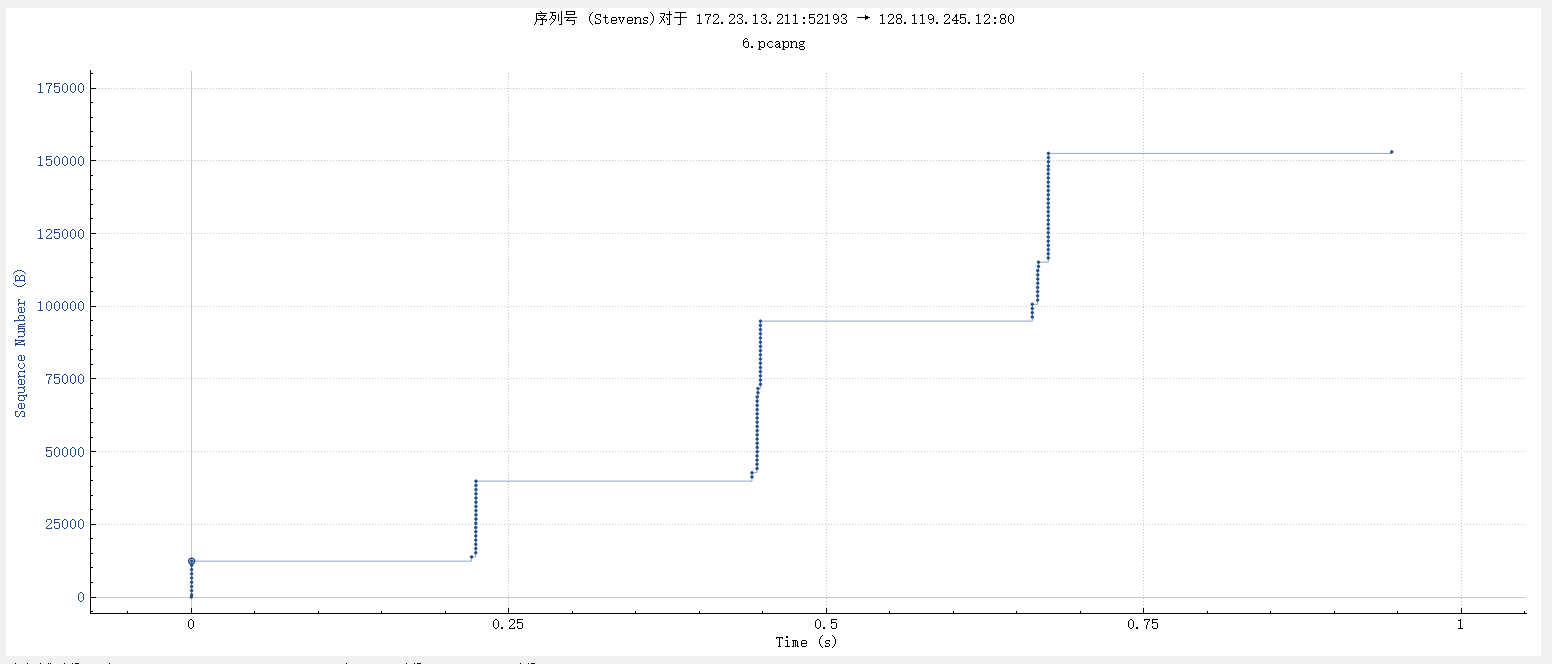


图13 Question 14佐证

吞吐量大约为153029 ∗ 8/( 3.541317 − 2.495695 )≈ 1.171 Mbps

慢启动阶段如下所示：在第0秒开始，大概在第0.25秒结束(无法看出慢启动结束都精确时间)。

拥塞避免阶段发送序列号数目不确定，之后也是进入平台区等待，防止拥塞，停止发送。