重庆大学大数据与软件学院

上 机 实 验 报 告

上机实践项目	TCP 协议分析		
课程名称	计算机网络		
姓名	XXX	成绩	
学号	2021XXXX	教师	**XXX**
班级	软工 X 班	日期	2023/4/13

《计算机网络》上机实验报告

开课实验室：DS1502 **2023 年 4 月 13 日**

姓　名	XXX	年级、班级	2021 级软件工程 X 班	学号	**2021XXXX**
上机（项目）名称		TCP 协议分析		指导教师	XXX
教师评语	教师签名： **年　月　日**				

一、上机目的

In this lab, we'll investigate the behavior of the celebrated TCP protocol in detail.

• we'll do so by analyzing a trace of the TCP segments sent and received in transferring a 150KB file (containing the text of Lewis Carrol's Alice's Adventures in Wonderland) from your computer to a remote server.

• we'll study TCP's use of sequence and acknowledgement numbers for providing reliable data transfer; • Retrieving large HTML files；

• we'll see TCP's congestion control algorithm – slow start and congestion avoidance – in action;

• we'll look at TCP's receiver-advertised flow control mechanism.

• we'll also briefly consider TCP connection setup and we'll investigate the performance (throughput and round-trip time) of the TCP connection between your computer and the server.

Before beginning this lab, you'll probably want to review sections 3.5 and 3.7 in the text.

二、基本原理

Wireshark: Wireshark is a popular network protocol analysis tool that captures network traffic and analyzes the communication process of various protocols. It is a free and open-source software that can run on multiple operating systems including Windows, Mac OS X, and Linux. Wireshark captures packets from a network interface and can analyze them in real-time or offline. It supports various protocols such as TCP, UDP, IP, HTTP, FTP, DNS, ICMP, and can also decode TLS and SSL encrypted traffic. By using Wireshark, users can perform deep analysis of network communication processes, including understanding the data type, protocol header information, packet transmission time, and relationships between different packets. This is extremely useful for network administrators and security professionals.

TCP: TCP (Transmission Control Protocol) is a reliable, connection-oriented protocol that allows for the transmission of data across networks. It is responsible for establishing and maintaining a connection between two devices, ensuring that data is correctly acknowledged and re-transmitted if necessary. TCP operates by breaking data into smaller segments, each of which is sent across the network separately. These segments are then reassembled at the receiving device to recreate the original data. It also includes mechanisms for flow control, congestion control, and error detection, which help to ensure that data is transmitted efficiently and accurately. Due to its reliable nature, TCP is commonly used for applications such as email, file transfer, and web browsing, where the accuracy and completeness of data are critical. Overall, TCP is a fundamental protocol for modern networking that has played a key role in enabling the internet to grow and evolve over the past few decades.

Windows: Here are the steps to use Wireshark software on Windows for analyzing TCP protocol:

- Download and install Wireshark software on your Windows 11 computer.
- Open Wireshark and click on the "Start Capture" button to begin capturing network traffic.
- From the drop-down menu, select "Capture Filters" and choose the TCP protocol.
- Once you have selected the TCP protocol, click on the "Start" button to begin the capture process.
- As network traffic is captured, you can view the data in the Wireshark console.
- To further analyze the captured data, use the Wireshark tools to filter and sort the data in various ways.
- You can use the packet details window to view detailed information about each individual packet captured.
- You can also export the captured data to other applications or save them for later analysis.

三、使用的软件、硬件

Software: Windows11 && Wireshark && Firefox;

Hardware: Lenovo Legion R9000P2021H.

四、上机操作步骤

1. Capturing a bulk TCP transfer from your computer to a remote server

Before beginning our exploration of TCP, we'll need to use Wireshark to obtain a packet trace of the TCP transfer of a file from your computer to a remote server. You'll do so by accessing a Web page that will allow you to enter the name of a file stored on your computer (which contains the ASCII text of *Alice in Wonderland*), and then transfer the file to a Web server using the HTTP POST method (see section 2.2.3 in the text). We're using the POST method rather than the GET method as we'd like to transfer a large amount of data *from* your computer to another computer. Of course, we'll be running Wireshark during this time to obtain the trace of the TCP segments sent and received from your computer.

Do the following:

1) Start up your web browser. Go the http://gaia.cs.umass.edu/wireshark¬labs/alice.txt and retrieve an ASCII copy of Alice in Wonderland. Store this file somewhere on your computer.

2) Next go to http://gaia.cs.umass.edu/wireshark-labs/TCP-wireshark-file1.html.

3) You should see a screen that looks like:

4) Use the Browse button in this form to enter the name of the file (full path name) on your computer containing Alice in Wonderland (or do so manually). Don't yet press the "Upload alice.txt file" button.

5) Now start up Wireshark and begin packet capture (Capture->Start) and then press OK on the Wireshark Packet Capture Options screen (we'll not need to select any options here).

6) Returning to your browser, press the "Upload alice.txt file" button to upload the file to the gaia.cs.umass.edu server. Once the file has been uploaded, a short congratulations message will be displayed in your browser window.

7) Stop Wireshark packet capture. Your Wireshark window should look similar to the window shown below.

2. A first look at the captured trace

Before analyzing the behavior of the TCP connection in detail, let's take a high level view of the trace.

- First, filter the packets displayed in the Wireshark window by entering "tcp" (lowercase, no quotes, and don't forget to press return after entering!) into the display filter specification window towards the top of the Wireshark window.

What you should see is series of TCP and HTTP messages between your computer and gaia.cs.umass.edu. You should see the initial three-way handshake containing a SYN message. You should see an HTTP POST message. Depending on the version of Wireshark you are using, you might see a series of "HTTP Continuation" messages being sent from your computer to gaia.cs.umass.edu. Recall from our discussion in the earlier HTTP Wireshark lab, that is no such thing as an HTTP Continuation message – this is Wireshark's way of indicating that there are multiple TCP segments being used to carry a single HTTP message. In more recent versions of Wireshark, you'll see "[TCP segment of a reassembled PDU]" in the Info column of the Wireshark display to indicate that this TCP segment contained data that belonged to an upper layer protocol message (in our case here, HTTP). You should also see TCP ACK segments being returned from gaia.cs.umass.edu to your computer.

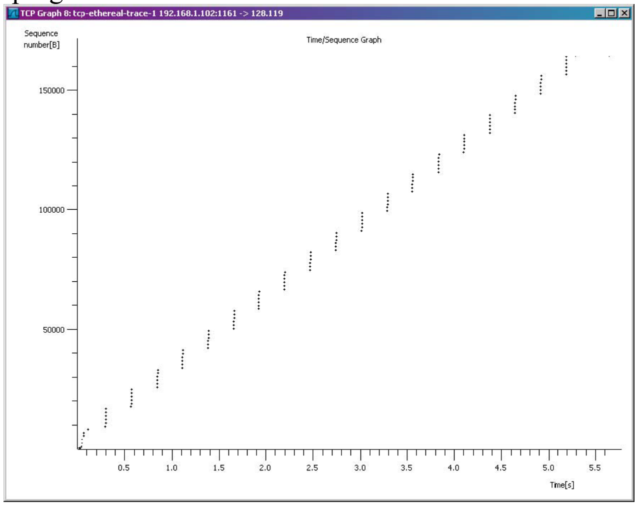
3. TCP Basics

Answer the questions.

4. TCP congestion control in action

Let's now examine the amount of data sent per unit time from the client to the server. Rather than (tediously!) calculating this from the raw data in the Wireshark window, we'll use one of Wireshark's TCP graphing utilities -*Time-Sequence-Graph(Stevens*) -to plot out data.

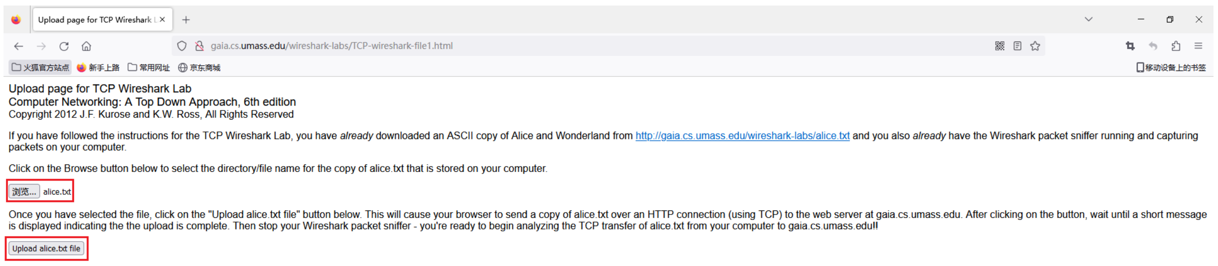
- Select a TCP segment in the Wireshark's "listing of captured-packets" window. Then select the menu: Statistics->TCP Stream Graph-> Time-Sequence-Graph(Stevens). You should see a plot that looks similar to the following plot, which was created from the captured packets in the packet trace tcp ethereal-trace-1 in http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.zip (see earlier footnote):



Here, each dot represents a TCP segment sent, plotting the sequence number of the segment versus the time at which it was sent. Note that a set of dots stacked above each other represents a series of packets that were sent back-to-back by the sender.

五、过程原始记录(数据、图表、计算等)

1. Capturing a bulk TCP transfer from your computer to a remote server

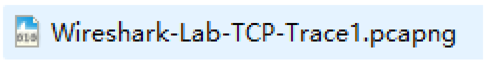

Upload page for TCP Wireshark L
gaia.cs.umass.edu/wireshark-labs/TCP-wireshark-file1.html
火狐官方站点
新手上路
常用网址
京东商城
移动设备上的书签

Upload page for TCP Wireshark Lab
Computer Networking: A Top Down Approach, 6th edition
Copyright 2012 J.F. Kurose and K.W. Ross, All Rights Reserved

If you have followed the instructions for the TCP Wireshark Lab, you have *already* downloaded an ASCII copy of Alice and Wonderland from http://gaia.cs.umass.edu/wireshark-labs/alice.txt and you also *already* have the Wireshark packet sniffer running and capturing packets on your computer.

Click on the Browse button below to select the directory/file name for the copy of alice.txt that is stored on your computer.

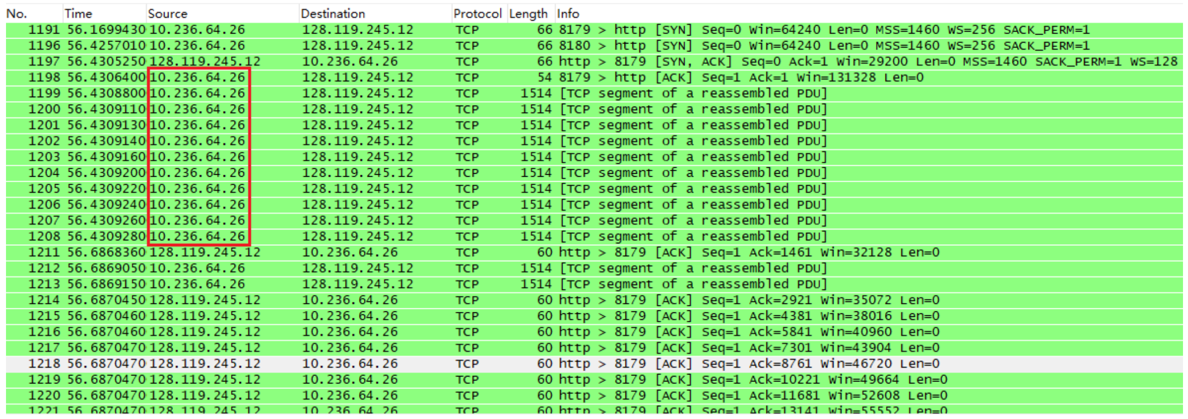
Once you have selected the file, click on the "Upload alice.txt file" button below. This will cause your browser to send a copy of alice.txt over an HTTP connection (using TCP) to the web server at gaia.cs.umass.edu. After clicking on the button, wait until a short message is displayed indicating the the upload is complete. Then stop your Wireshark packet sniffer - you're ready to begin analyzing the TCP transfer of alice.txt from your computer to gaia.cs.umass.edu!!



2. A first look at the captured trace

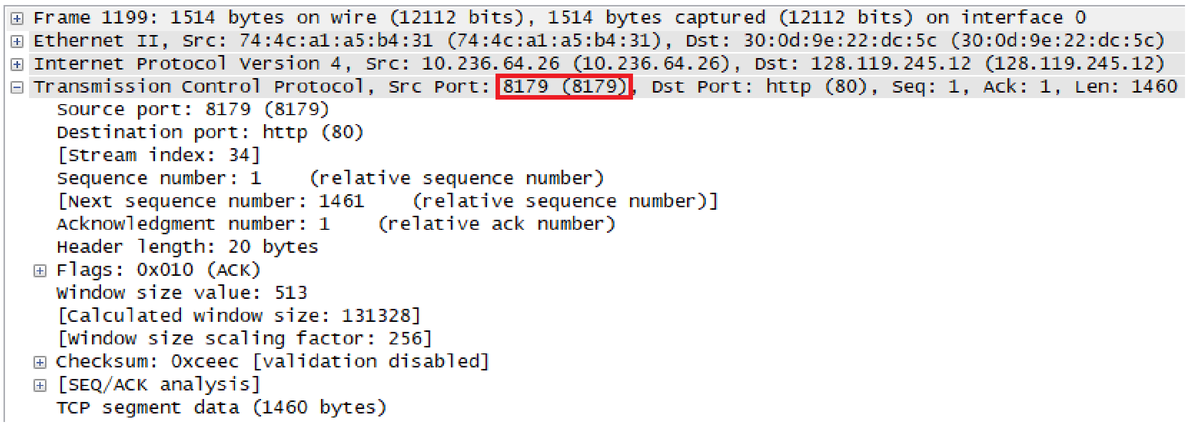
Q1: What is the IP address and TCP port number used by the client computer (source) that is transferring the file to gaia.cs.umass.edu? To answer this question, it’s probably easiest to select an HTTP message and explore the details of the TCP packet used to carry this HTTP message, using the “details 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.

A1: 将文件传输到 gaia.cs.umass.edu 的客户端计算机使用的 IP 地址为：10.236.64.26；



```
No.    Time       Source          Destination     Protocol Length Info
  1191 56.1699430 10.236.64.26    128.119.245.12  TCP        66 8179 > http [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
  1196 56.4257010 10.236.64.26    128.119.245.12  TCP        66 8180 > http [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
  1197 56.4305250 128.119.245.12  10.236.64.26    TCP        66 http > 8179 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=128
  1198 56.4306400 10.236.64.26    128.119.245.12  TCP        54 8179 > http [ACK] Seq=1 Ack=1 Win=131328 Len=0
  1199 56.4308800 10.236.64.26    128.119.245.12  TCP      1514 [TCP segment of a reassembled PDU]
  1200 56.4309110 10.236.64.26    128.119.245.12  TCP      1514 [TCP segment of a reassembled PDU]
  1201 56.4309130 10.236.64.26    128.119.245.12  TCP      1514 [TCP segment of a reassembled PDU]
  1202 56.4309140 10.236.64.26    128.119.245.12  TCP      1514 [TCP segment of a reassembled PDU]
  1203 56.4309160 10.236.64.26    128.119.245.12  TCP      1514 [TCP segment of a reassembled PDU]
  1204 56.4309200 10.236.64.26    128.119.245.12  TCP      1514 [TCP segment of a reassembled PDU]
  1205 56.4309220 10.236.64.26    128.119.245.12  TCP      1514 [TCP segment of a reassembled PDU]
  1206 56.4309240 10.236.64.26    128.119.245.12  TCP      1514 [TCP segment of a reassembled PDU]
  1207 56.4309260 10.236.64.26    128.119.245.12  TCP      1514 [TCP segment of a reassembled PDU]
  1208 56.4309280 10.236.64.26    128.119.245.12  TCP      1514 [TCP segment of a reassembled PDU]
  1211 56.6868360 128.119.245.12  10.236.64.26    TCP        60 http > 8179 [ACK] Seq=1 Ack=1461 Win=32128 Len=0
  1212 56.6869050 10.236.64.26    128.119.245.12  TCP      1514 [TCP segment of a reassembled PDU]
  1213 56.6869150 10.236.64.26    128.119.245.12  TCP      1514 [TCP segment of a reassembled PDU]
  1214 56.6870450 128.119.245.12  10.236.64.26    TCP        60 http > 8179 [ACK] Seq=1 Ack=2921 Win=35072 Len=0
  1215 56.6870460 128.119.245.12  10.236.64.26    TCP        60 http > 8179 [ACK] Seq=1 Ack=4381 Win=38016 Len=0
  1216 56.6870460 128.119.245.12  10.236.64.26    TCP        60 http > 8179 [ACK] Seq=1 Ack=5841 Win=40960 Len=0
  1217 56.6870470 128.119.245.12  10.236.64.26    TCP        60 http > 8179 [ACK] Seq=1 Ack=7301 Win=43904 Len=0
  1218 56.6870470 128.119.245.12  10.236.64.26    TCP        60 http > 8179 [ACK] Seq=1 Ack=8761 Win=46720 Len=0
  1219 56.6870470 128.119.245.12  10.236.64.26    TCP        60 http > 8179 [ACK] Seq=1 Ack=10221 Win=49664 Len=0
  1220 56.6870470 128.119.245.12  10.236.64.26    TCP        60 http > 8179 [ACK] Seq=1 Ack=11681 Win=52608 Len=0
  1221 56.6870470 128.119.245.12  10.236.64.26    TCP        60 http > 8179 [ACK] Seq=1 Ack=13141 Win=55552 Len=0
```

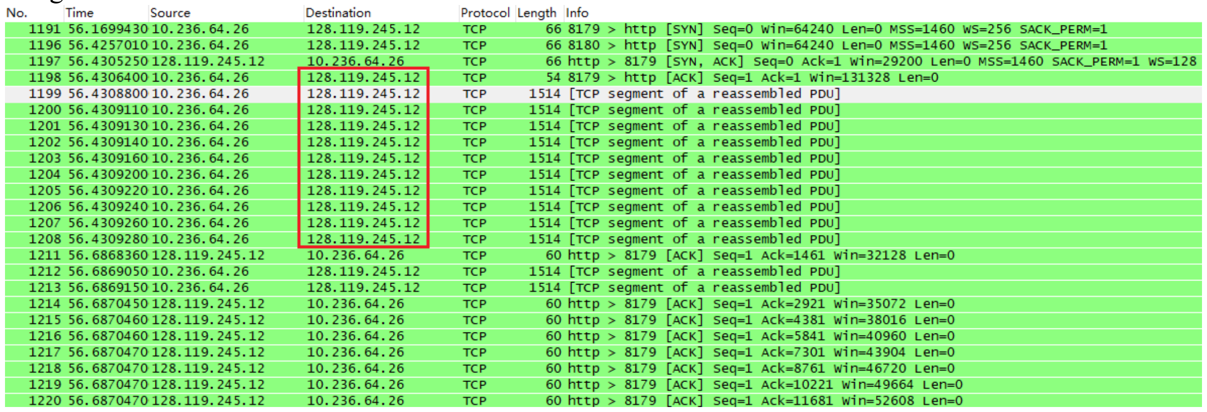
TCP 端口号为：8179；



```
⊞ Frame 1199: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface 0
⊞ Ethernet II, Src: 74:4c:a1:a5:b4:31 (74:4c:a1:a5:b4:31), Dst: 30:0d:9e:22:dc:5c (30:0d:9e:22:dc:5c)
⊞ Internet Protocol Version 4, Src: 10.236.64.26 (10.236.64.26), Dst: 128.119.245.12 (128.119.245.12)
⊟ Transmission Control Protocol, Src Port: 8179 (8179), Dst Port: http (80), Seq: 1, Ack: 1, Len: 1460
    Source port: 8179 (8179)
    Destination port: http (80)
    [Stream index: 34]
    Sequence number: 1    (relative sequence number)
    [Next sequence number: 1461    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    Header length: 20 bytes
  ⊞ Flags: 0x010 (ACK)
    Window size value: 513
    [Calculated window size: 131328]
    [Window size scaling factor: 256]
  ⊞ Checksum: 0xceec [validation disabled]
  ⊞ [SEQ/ACK analysis]
    TCP segment data (1460 bytes)
```

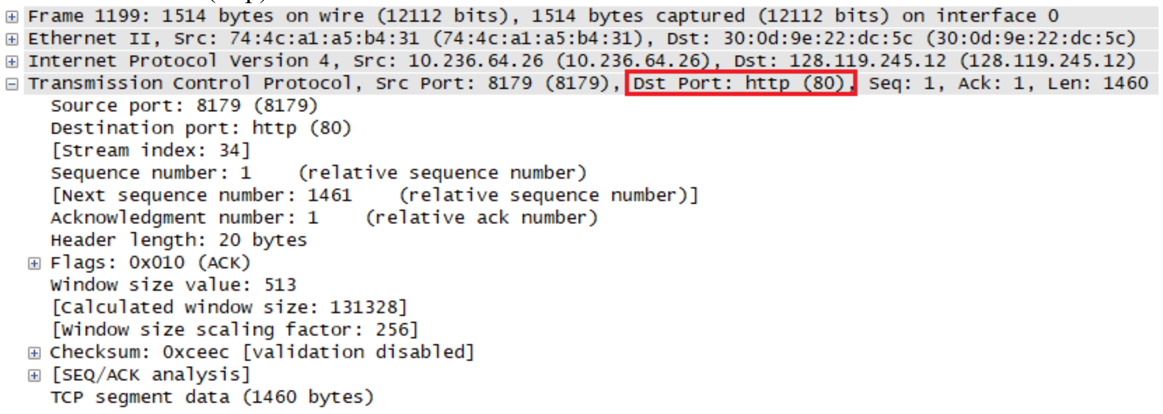
Q2: What is the IP address of gaia.cs.umass.edu? On what port number is it sending and receiving TCP segments for this connection?

A2: gaia.cs.umass.edu 的 IP 地址是：128.119.245.12；

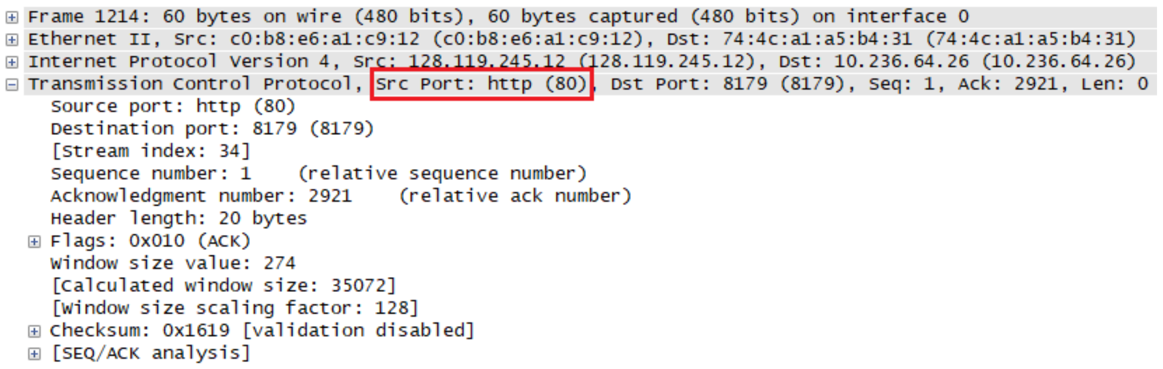


```
No.    Time       Source          Destination     Protocol Length Info
  1191 56.1699430 10.236.64.26    128.119.245.12  TCP        66 8179 > http [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
  1196 56.4257010 10.236.64.26    128.119.245.12  TCP        66 8180 > http [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
  1197 56.4305250 128.119.245.12  10.236.64.26    TCP        66 http > 8179 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=128
  1198 56.4306400 10.236.64.26    128.119.245.12  TCP        54 8179 > http [ACK] Seq=1 Ack=1 Win=131328 Len=0
  1199 56.4308800 10.236.64.26    128.119.245.12  TCP      1514 [TCP segment of a reassembled PDU]
  1200 56.4309110 10.236.64.26    128.119.245.12  TCP      1514 [TCP segment of a reassembled PDU]
  1201 56.4309130 10.236.64.26    128.119.245.12  TCP      1514 [TCP segment of a reassembled PDU]
  1202 56.4309140 10.236.64.26    128.119.245.12  TCP      1514 [TCP segment of a reassembled PDU]
  1203 56.4309160 10.236.64.26    128.119.245.12  TCP      1514 [TCP segment of a reassembled PDU]
  1204 56.4309200 10.236.64.26    128.119.245.12  TCP      1514 [TCP segment of a reassembled PDU]
  1205 56.4309220 10.236.64.26    128.119.245.12  TCP      1514 [TCP segment of a reassembled PDU]
  1206 56.4309240 10.236.64.26    128.119.245.12  TCP      1514 [TCP segment of a reassembled PDU]
  1207 56.4309260 10.236.64.26    128.119.245.12  TCP      1514 [TCP segment of a reassembled PDU]
  1208 56.4309280 10.236.64.26    128.119.245.12  TCP      1514 [TCP segment of a reassembled PDU]
  1211 56.6868360 128.119.245.12  10.236.64.26    TCP        60 http > 8179 [ACK] Seq=1 Ack=1461 Win=32128 Len=0
  1212 56.6869050 10.236.64.26    128.119.245.12  TCP      1514 [TCP segment of a reassembled PDU]
  1213 56.6869150 10.236.64.26    128.119.245.12  TCP      1514 [TCP segment of a reassembled PDU]
  1214 56.6870450 128.119.245.12  10.236.64.26    TCP        60 http > 8179 [ACK] Seq=1 Ack=2921 Win=35072 Len=0
  1215 56.6870460 128.119.245.12  10.236.64.26    TCP        60 http > 8179 [ACK] Seq=1 Ack=4381 Win=38016 Len=0
  1216 56.6870460 128.119.245.12  10.236.64.26    TCP        60 http > 8179 [ACK] Seq=1 Ack=5841 Win=40960 Len=0
  1217 56.6870470 128.119.245.12  10.236.64.26    TCP        60 http > 8179 [ACK] Seq=1 Ack=7301 Win=43904 Len=0
  1218 56.6870470 128.119.245.12  10.236.64.26    TCP        60 http > 8179 [ACK] Seq=1 Ack=8761 Win=46720 Len=0
  1219 56.6870470 128.119.245.12  10.236.64.26    TCP        60 http > 8179 [ACK] Seq=1 Ack=10221 Win=49664 Len=0
  1220 56.6870470 128.119.245.12  10.236.64.26    TCP        60 http > 8179 [ACK] Seq=1 Ack=11681 Win=52608 Len=0
```

端口号为：80 (http)；



```
⊞ Frame 1199: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface 0
⊞ Ethernet II, Src: 74:4c:a1:a5:b4:31 (74:4c:a1:a5:b4:31), Dst: 30:0d:9e:22:dc:5c (30:0d:9e:22:dc:5c)
⊞ Internet Protocol Version 4, Src: 10.236.64.26 (10.236.64.26), Dst: 128.119.245.12 (128.119.245.12)
⊟ Transmission Control Protocol, Src Port: 8179 (8179), Dst Port: http (80), Seq: 1, Ack: 1, Len: 1460
    Source port: 8179 (8179)
    Destination port: http (80)
    [Stream index: 34]
    Sequence number: 1    (relative sequence number)
    [Next sequence number: 1461    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    Header length: 20 bytes
  ⊞ Flags: 0x010 (ACK)
    Window size value: 513
    [Calculated window size: 131328]
    [Window size scaling factor: 256]
  ⊞ Checksum: 0xceec [validation disabled]
  ⊞ [SEQ/ACK analysis]
    TCP segment data (1460 bytes)
```

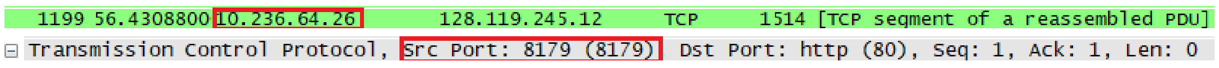


```
⊞ Frame 1214: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
⊞ Ethernet II, Src: c0:b8:e6:a1:c9:12 (c0:b8:e6:a1:c9:12), Dst: 74:4c:a1:a5:b4:31 (74:4c:a1:a5:b4:31)
⊞ Internet Protocol Version 4, Src: 128.119.245.12 (128.119.245.12), Dst: 10.236.64.26 (10.236.64.26)
⊟ Transmission Control Protocol, Src Port: http (80), Dst Port: 8179 (8179), Seq: 1, Ack: 2921, Len: 0
    Source port: http (80)
    Destination port: 8179 (8179)
    [Stream index: 34]
    Sequence number: 1    (relative sequence number)
    Acknowledgment number: 2921    (relative ack number)
    Header length: 20 bytes
  ⊞ Flags: 0x010 (ACK)
    Window size value: 274
    [Calculated window size: 35072]
    [Window size scaling factor: 128]
  ⊞ Checksum: 0x1619 [validation disabled]
  ⊞ [SEQ/ACK analysis]
```

Q3: What is the IP address and TCP port number used by your client computer (source) to transfer the file to gaia.cs.umass.edu?

A3: 因为我是使用的自己的计算机抓包得到的数据，所以答案同 Q1/A1，IP 地址为：10.236.64.26，

端口号为：8179；

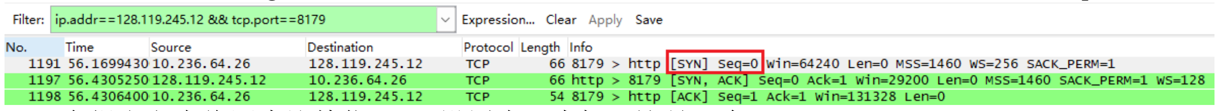


```
  1199 56.4308800 10.236.64.26       128.119.245.12     TCP      1514 [TCP segment of a reassembled PDU]
⊟ Transmission Control Protocol, Src Port: 8179 (8179), Dst Port: http (80), Seq: 1, Ack: 1, Len: 0
```

3. TCP Basics

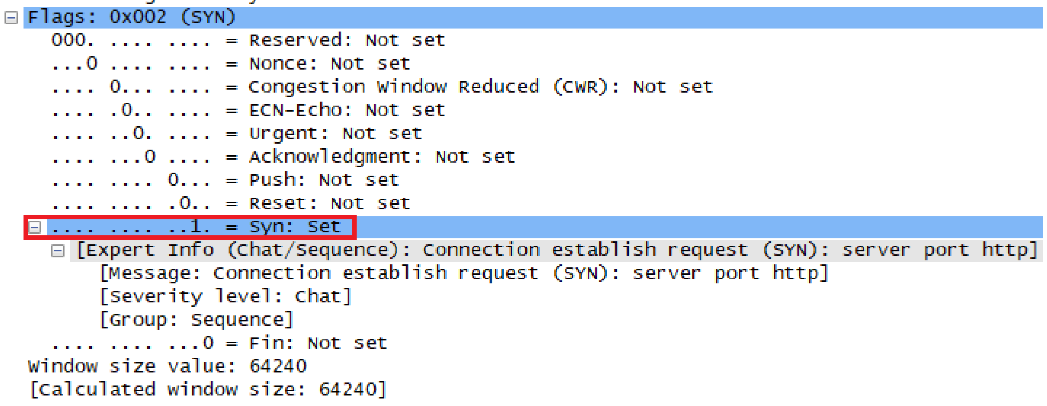
Q4: What is the sequence number of the TCP SYN segment that is used to initiate the TCP connection between the client computer and gaia.cs.umass.edu? What is it in the segment that identifies the segment as a SYN segment?

A4: 用于在客户端和 gaia.cs.umass.edu 之间发起 TCP 连接的 TCP SYN 段的序列号为 Seq = 0；



```
Filter: ip.addr==128.119.245.12 && tcp.port==8179

No.   Time       Source          Destination     Protocol Length Info
 1191 56.1699430 10.236.64.26    128.119.245.12  TCP       66 8179 > http [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
 1197 56.4305250 128.119.245.12  10.236.64.26    TCP       66 http > 8179 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=128
 1198 56.4306400 10.236.64.26    128.119.245.12  TCP       54 8179 > http [ACK] Seq=1 Ack=1 Win=131328 Len=0
```

在报文段中将同步比特位 SYN 设置为 1 来标识这是一个 SYN segment；

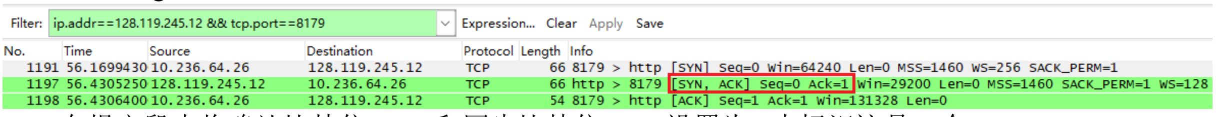


```
⊟ Flags: 0x002 (SYN)
    000. .... .... = Reserved: Not set
    ...0 .... .... = Nonce: Not set
    .... 0... .... = Congestion Window Reduced (CWR): Not set
    .... .0.. .... = ECN-Echo: Not set
    .... ..0. .... = Urgent: Not set
    .... ...0 .... = Acknowledgment: Not set
    .... .... 0... = Push: Not set
    .... .... .0.. = Reset: Not set
  ⊟ .... .... ..1. = Syn: Set
    ⊟ [Expert Info (Chat/Sequence): Connection establish request (SYN): server port http]
        [Message: Connection establish request (SYN): server port http]
        [Severity level: Chat]
        [Group: Sequence]
    .... .... ...0 = Fin: Not set
  Window size value: 64240
  [Calculated window size: 64240]
```

Q5: 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?

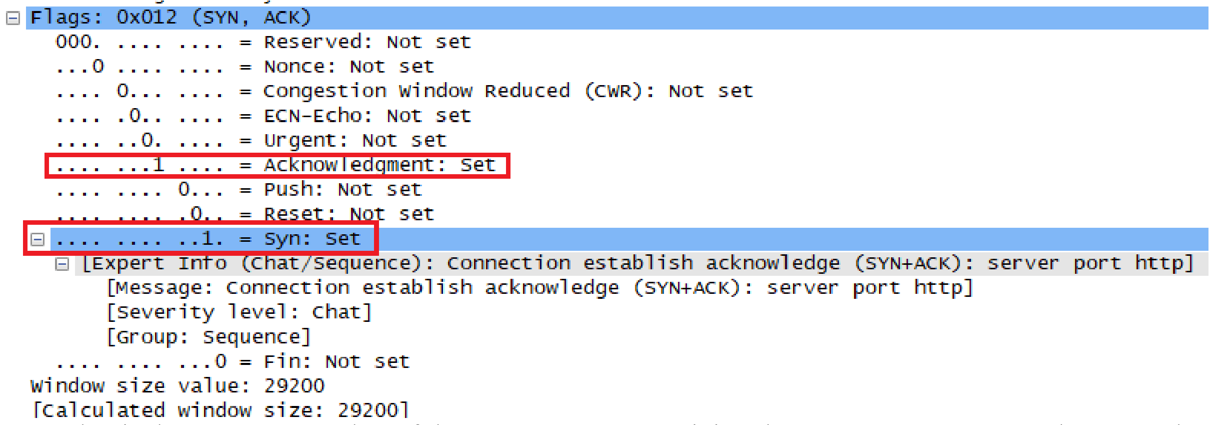
A5: gaia.cs.umass.edu 发送给客户端计算机应答 SYN 的 SYNACK 段的序列号为 Seq = 0，

SYNACK 段中 Acknowledgement 字段的值为 ACK = 1，该值是由服务器端在客户端计算机发来的 SYN segment 的序号上增加了 1 而产生的，该过程可以理解为 TCP 连接初始化的一部分；



```
Filter: ip.addr==128.119.245.12 && tcp.port==8179     Expression... Clear Apply Save

No.   Time        Source          Destination     Protocol Length Info
 1191 56.1699430 10.236.64.26    128.119.245.12  TCP      66 8179 > http [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
 1197 56.4305250 128.119.245.12  10.236.64.26    TCP      66 http > 8179 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=128
 1198 56.4306400 10.236.64.26    128.119.245.12  TCP      54 8179 > http [ACK] Seq=1 Ack=1 Win=131328 Len=0
```

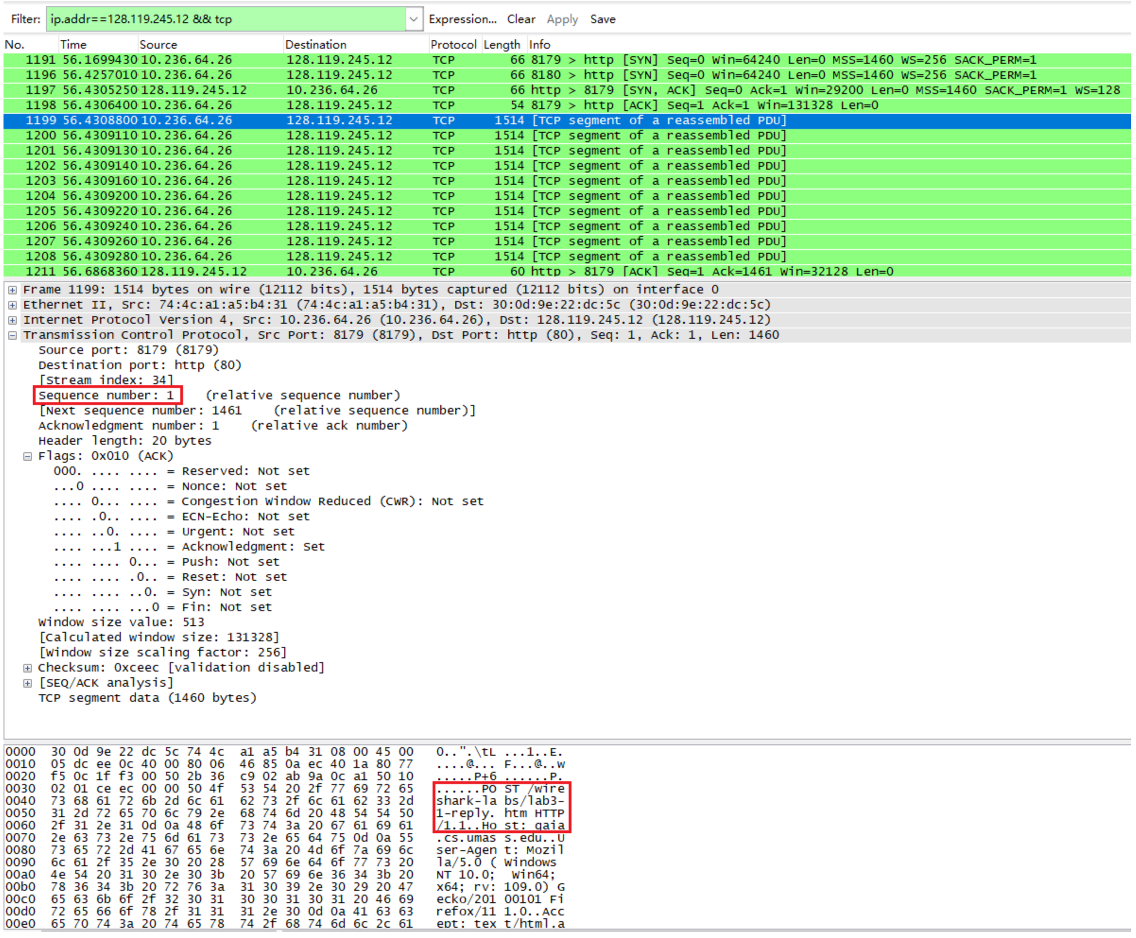
在报文段中将确认比特位 ACK 和同步比特位 SYN 设置为 1 来标识这是一个 SYNACK segment；



```
⊟ 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
    ⊟ [Expert Info (Chat/Sequence): Connection establish acknowledge (SYN+ACK): server port http]
        [Message: Connection establish acknowledge (SYN+ACK): server port http]
        [Severity level: Chat]
        [Group: Sequence]
    .... .... ...0 = Fin: Not set
  Window size value: 29200
  [Calculated window size: 29200]
```

Q6: 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.

A6:；包含 HTTP POST 命令的 TCP 段的序号为 Seq = 1；



```
Filter: ip.addr==128.119.245.12 && tcp     Expression... Clear Apply Save

No.   Time        Source          Destination     Protocol Length Info
 1191 56.1699430 10.236.64.26    128.119.245.12  TCP      66 8179 > http [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
 1196 56.4257010 10.236.64.26    128.119.245.12  TCP      66 8180 > http [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
 1197 56.4305250 128.119.245.12  10.236.64.26    TCP      66 http > 8179 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=128
 1198 56.4306400 10.236.64.26    128.119.245.12  TCP      54 8179 > http [ACK] Seq=1 Ack=1 Win=131328 Len=0
 1199 56.4308800 10.236.64.26    128.119.245.12  TCP    1514 [TCP segment of a reassembled PDU]
 1200 56.4309110 10.236.64.26    128.119.245.12  TCP    1514 [TCP segment of a reassembled PDU]
 1201 56.4309130 10.236.64.26    128.119.245.12  TCP    1514 [TCP segment of a reassembled PDU]
 1202 56.4309140 10.236.64.26    128.119.245.12  TCP    1514 [TCP segment of a reassembled PDU]
 1203 56.4309160 10.236.64.26    128.119.245.12  TCP    1514 [TCP segment of a reassembled PDU]
 1204 56.4309200 10.236.64.26    128.119.245.12  TCP    1514 [TCP segment of a reassembled PDU]
 1205 56.4309220 10.236.64.26    128.119.245.12  TCP    1514 [TCP segment of a reassembled PDU]
 1206 56.4309240 10.236.64.26    128.119.245.12  TCP    1514 [TCP segment of a reassembled PDU]
 1207 56.4309260 10.236.64.26    128.119.245.12  TCP    1514 [TCP segment of a reassembled PDU]
 1208 56.4309280 10.236.64.26    128.119.245.12  TCP    1514 [TCP segment of a reassembled PDU]
 1211 56.6868360 128.119.245.12  10.236.64.26    TCP      60 http > 8179 [ACK] Seq=1 Ack=1461 Win=32128 Len=0

⊞ Frame 1199: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface 0
⊞ Ethernet II, Src: 74:4c:a1:a5:b4:31 (74:4c:a1:a5:b4:31), Dst: 30:0d:9e:22:dc:5c (30:0d:9e:22:dc:5c)
⊞ Internet Protocol Version 4, Src: 10.236.64.26 (10.236.64.26), Dst: 128.119.245.12 (128.119.245.12)
⊟ Transmission Control Protocol, Src Port: 8179 (8179), Dst Port: http (80), Seq: 1, Ack: 1, Len: 1460
    Source port: 8179 (8179)
    Destination port: http (80)
    [Stream index: 34]
    Sequence number: 1    (relative sequence number)
    [Next sequence number: 1461    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    Header length: 20 bytes
  ⊟ Flags: 0x010 (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
      .... .... ..0. = Syn: Not set
      .... .... ...0 = Fin: Not set
    Window size value: 513
    [Calculated window size: 131328]
    [Window size scaling factor: 256]
  ⊞ Checksum: 0xceec [validation disabled]
  ⊞ [SEQ/ACK analysis]
    TCP segment data (1460 bytes)

0000  30 0d 9e 22 dc 5c 74 4c  a1 a5 b4 31 08 00 45 00   0..".\tL ...1..E.
0010  05 dc ee 0c 40 00 80 06  46 85 0a ec 40 1a 80 77   ....@... F...@..w
0020  f5 0c 1f f3 00 50 2b 36  c9 02 ab 9a 0c a1 50 10   .....P+6 ......P.
0030  02 01 ce ec 00 00 50 4f  53 54 20 2f 77 69 72 65   ......PO ST /wire
0040  73 68 61 72 6b 2d 6c 61  62 73 2f 6c 61 62 33 2d   shark-la bs/lab3-
0050  31 2d 72 65 70 6c 79 2e  68 74 6d 20 48 54 54 50   1-reply. htm HTTP
0060  2f 31 2e 31 0d 0a 48 6f  73 74 3a 20 67 61 69 61   /1.1..Ho st: gaia
0070  2e 63 73 2e 75 6d 61 73  73 2e 65 64 75 0d 0a 55   .cs.umas s.edu..U
0080  73 65 72 2d 41 67 65 6e  74 3a 20 4d 6f 7a 69 6c   ser-Agen t: Mozil
0090  6c 61 2f 35 2e 30 20 28  57 69 6e 64 6f 77 73 20   la/5.0 ( Windows
00a0  4e 54 20 31 30 2e 30 3b  20 57 69 6e 36 34 3b 20   NT 10.0;  Win64;
00b0  78 36 34 3b 20 72 76 3a  31 30 39 2e 30 29 20 47   x64; rv: 109.0) G
00c0  65 63 6b 6f 2f 32 30 31  30 30 31 30 31 20 46 69   ecko/201 00101 Fi
00d0  72 65 66 6f 78 2f 31 31  31 2e 30 0d 0a 41 63 63   refox/11 1.0..Acc
00e0  65 70 74 3a 20 74 65 78  74 2f 68 74 6d 6c 2c 61   ept: tex t/html,a
```

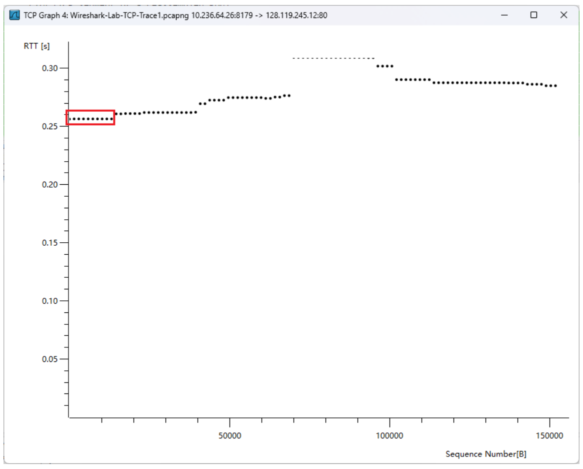
Q7: Consider the TCP segment containing the HTTP POST as the first segment in the TCP connection. What are the sequence numbers of the first six segments in the TCP connection (including the segment containing the HTTP POST)? At what time was each segment sent? When was the ACK for each segment received? 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 239 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 239 for all subsequent segments.
(Note: Wireshark has a nice feature that allows you to plot the RTT for each of the TCP segments sent. Select a TCP segment in the "listing of captured packets" window that is being sent from the client to the gaia.cs.umass.edu server. Then select: Statistics->TCP Stream Graph->Round Trip Time Graph.)

A7: TCP 连接中前六个段（包括包含 HTTP POST 的段）的序号、发送时间、收到 ACK 的时间（相同颜色的上下一一对应）、每个 TCP 段对应的 SampleRTT 和 EstimatedRTT 分别如下图：

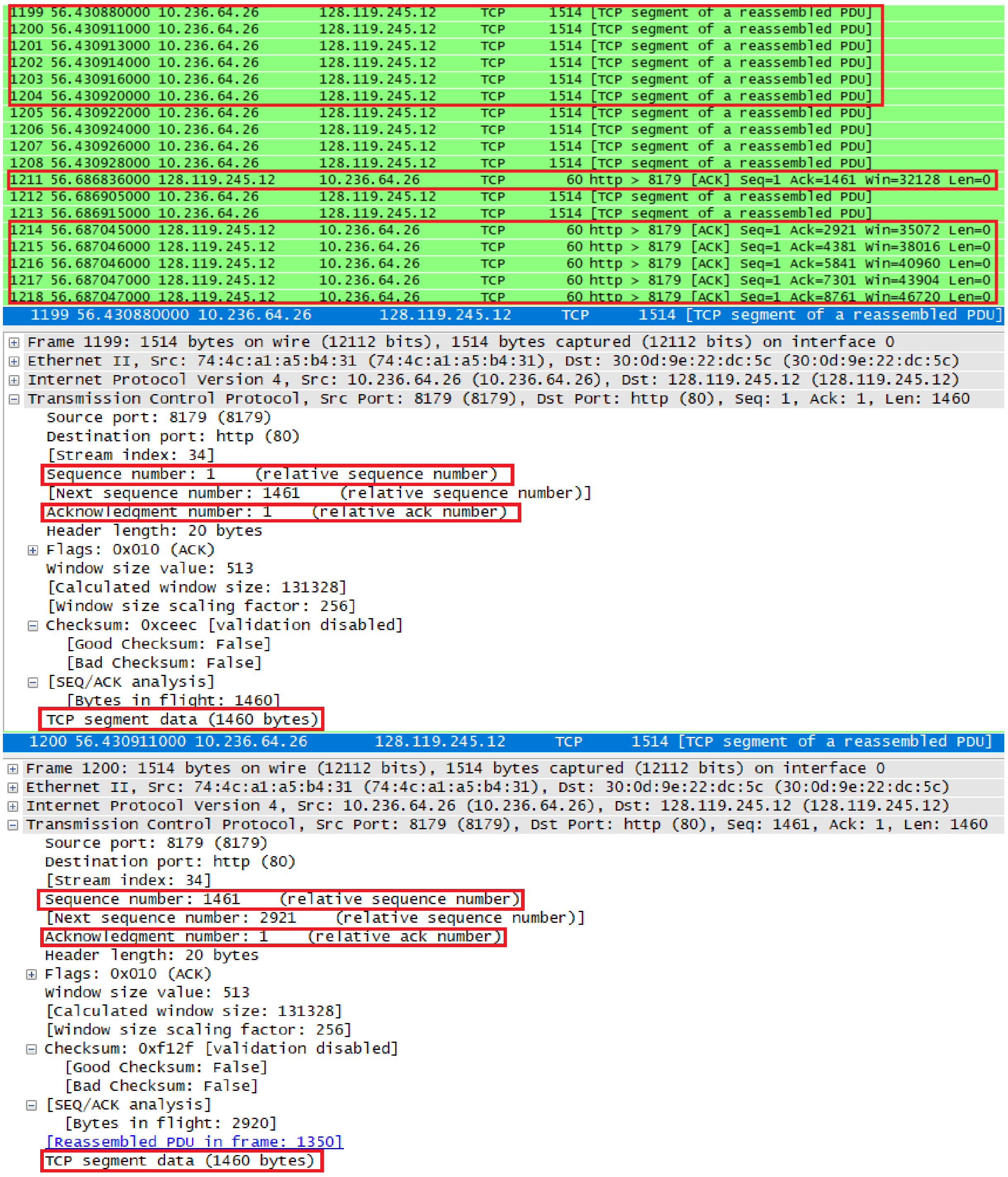
Number	Time	Size	PayloadSize	relative sn#	relative ack#	
1199	56.43088	1514	1460	1	1	🟥
1200	56.430911	1514	1460	1461	1	🟨
1201	56.430913	1514	1460	2921	1	🟩
1202	56.430914	1514	1460	4381	1	🟦
1203	56.430916	1514	1460	5841	1	⬛
1204	56.43092	1514	1460	7301	1	🟪
1211	56.686836	60	0	1	1461	🟥
1214	56.687045	60	0	1	2921	🟨
1215	56.687046	60	0	1	4381	🟩
1216	56.687046	60	0	1	5841	🟦
1217	56.687047	60	0	1	7301	⬛
1218	56.687047	60	0	1	8761	🟪

EstimatedRTT = (1- α)EstimatedRTT + α *SampleRTT (RFC 6298: α =0.125)

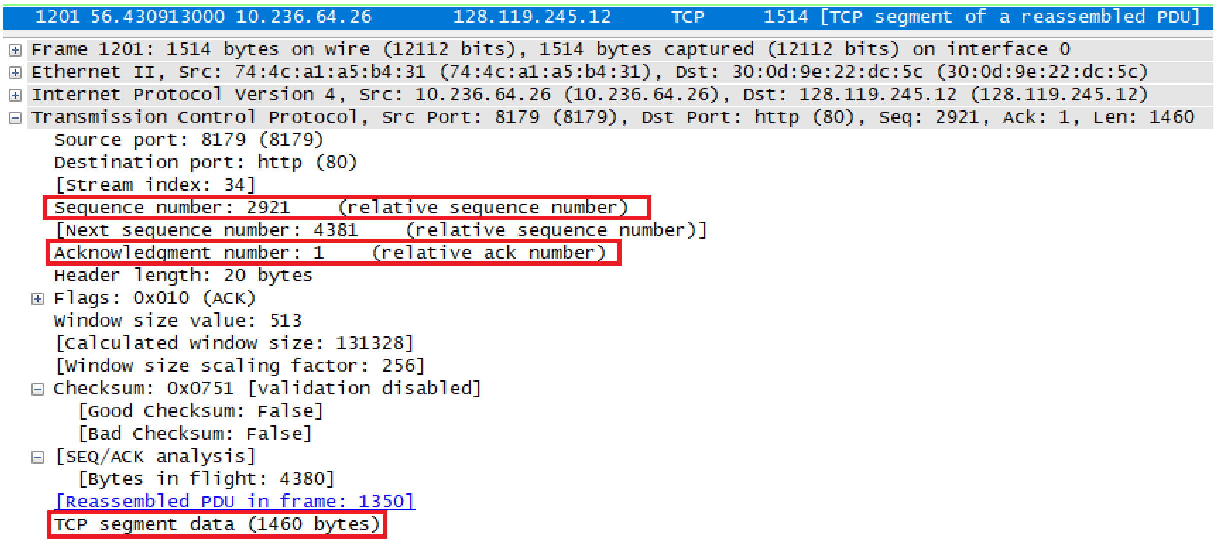
Numbel	SendTime	AckTime	SampleRTT	EstimatedRTT
1199	56.43088	56.68684	0.255956	0.255956
1200	56.43091	56.68705	0.256134	0.255978
1201	56.43091	56.68705	0.256133	0.255998
1202	56.43091	56.68705	0.256132	0.256014
1203	56.43092	56.68705	0.256131	0.256028
1204	56.43092	56.68705	0.256127	0.256041


TCP Graph 4: Wireshark-Lab-TCP-Trace1.pcapng 10.236.64.26:8179 -> 128.119.245.12:80
RTT [s]
0.30
0.25
0.20
0.15
0.10
0.05
50000
100000
150000
Sequence Number[B]

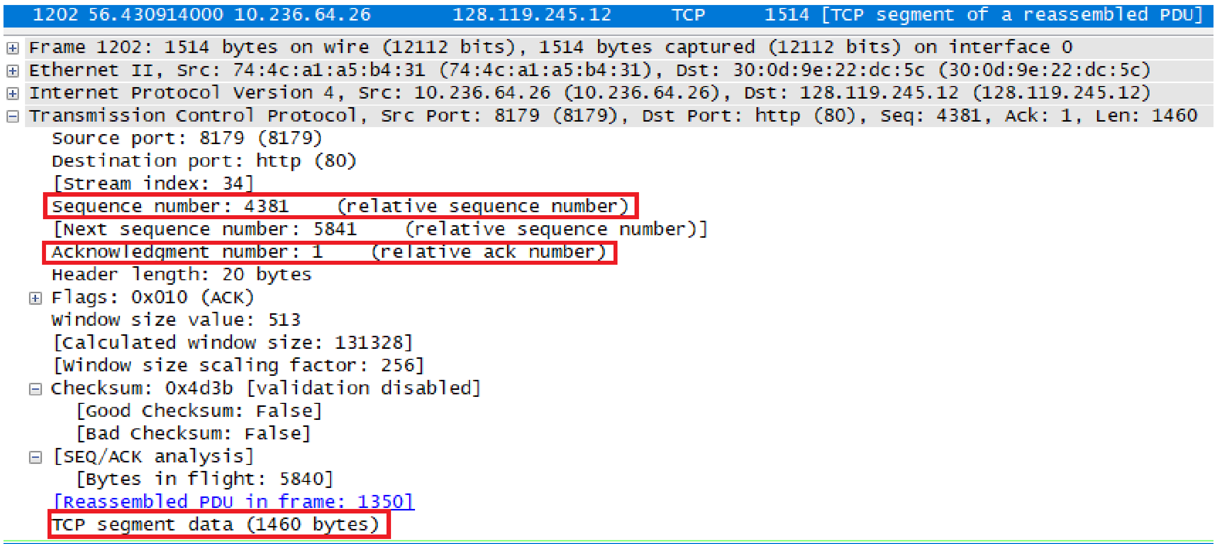
作证材料：



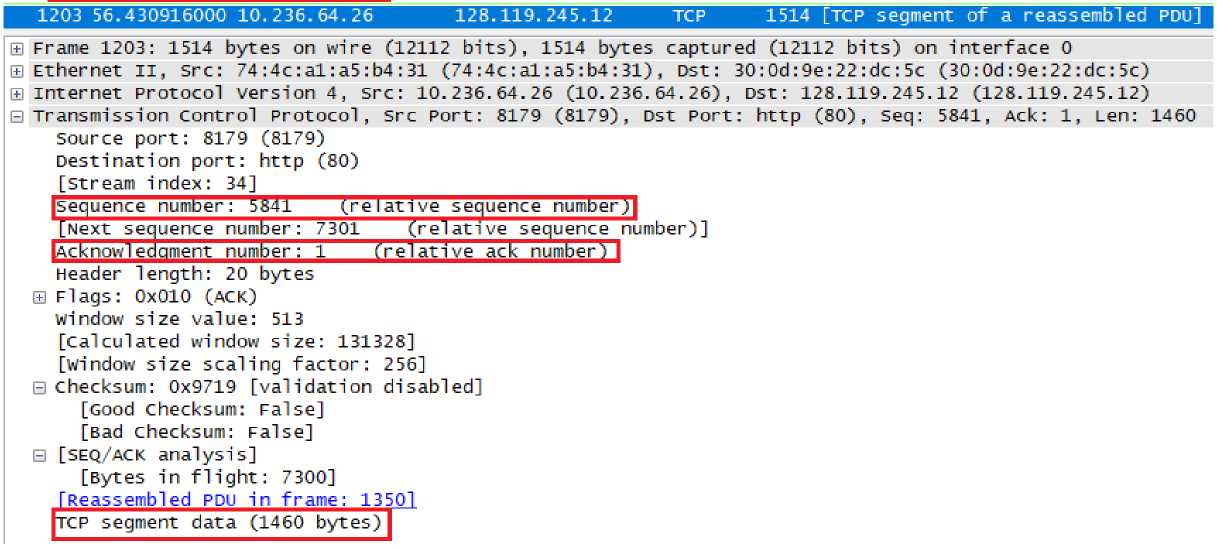
```
1199 56.430880000 10.236.64.26     128.119.245.12    TCP     1514 [TCP segment of a reassembled PDU]
1200 56.430911000 10.236.64.26     128.119.245.12    TCP     1514 [TCP segment of a reassembled PDU]
1201 56.430913000 10.236.64.26     128.119.245.12    TCP     1514 [TCP segment of a reassembled PDU]
1202 56.430914000 10.236.64.26     128.119.245.12    TCP     1514 [TCP segment of a reassembled PDU]
1203 56.430916000 10.236.64.26     128.119.245.12    TCP     1514 [TCP segment of a reassembled PDU]
1204 56.430920000 10.236.64.26     128.119.245.12    TCP     1514 [TCP segment of a reassembled PDU]
1205 56.430922000 10.236.64.26     128.119.245.12    TCP     1514 [TCP segment of a reassembled PDU]
1206 56.430924000 10.236.64.26     128.119.245.12    TCP     1514 [TCP segment of a reassembled PDU]
1207 56.430926000 10.236.64.26     128.119.245.12    TCP     1514 [TCP segment of a reassembled PDU]
1208 56.430928000 10.236.64.26     128.119.245.12    TCP     1514 [TCP segment of a reassembled PDU]
1211 56.686836000 128.119.245.12   10.236.64.26      TCP       60 http > 8179 [ACK] Seq=1 Ack=1461 Win=32128 Len=0
1212 56.686905000 10.236.64.26     128.119.245.12    TCP     1514 [TCP segment of a reassembled PDU]
1213 56.686915000 10.236.64.26     128.119.245.12    TCP     1514 [TCP segment of a reassembled PDU]
1214 56.687045000 128.119.245.12   10.236.64.26      TCP       60 http > 8179 [ACK] Seq=1 Ack=2921 Win=35072 Len=0
1215 56.687046000 128.119.245.12   10.236.64.26      TCP       60 http > 8179 [ACK] Seq=1 Ack=4381 Win=38016 Len=0
1216 56.687046000 128.119.245.12   10.236.64.26      TCP       60 http > 8179 [ACK] Seq=1 Ack=5841 Win=40960 Len=0
1217 56.687047000 128.119.245.12   10.236.64.26      TCP       60 http > 8179 [ACK] Seq=1 Ack=7301 Win=43904 Len=0
1218 56.687047000 128.119.245.12   10.236.64.26      TCP       60 http > 8179 [ACK] Seq=1 Ack=8761 Win=46720 Len=0
  1199 56.430880000 10.236.64.26       128.119.245.12     TCP      1514 [TCP segment of a reassembled PDU]

⊞ Frame 1199: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface 0
⊞ Ethernet II, Src: 74:4c:a1:a5:b4:31 (74:4c:a1:a5:b4:31), Dst: 30:0d:9e:22:dc:5c (30:0d:9e:22:dc:5c)
⊞ Internet Protocol Version 4, Src: 10.236.64.26 (10.236.64.26), Dst: 128.119.245.12 (128.119.245.12)
⊟ Transmission Control Protocol, Src Port: 8179 (8179), Dst Port: http (80), Seq: 1, Ack: 1, Len: 1460
    Source port: 8179 (8179)
    Destination port: http (80)
    [Stream index: 34]
    Sequence number: 1     (relative sequence number)
    [Next sequence number: 1461     (relative sequence number)]
    Acknowledgment number: 1     (relative ack number)
    Header length: 20 bytes
  ⊞ Flags: 0x010 (ACK)
    Window size value: 513
    [Calculated window size: 131328]
    [Window size scaling factor: 256]
  ⊟ Checksum: 0xceec [validation disabled]
      [Good Checksum: False]
      [Bad Checksum: False]
  ⊟ [SEQ/ACK analysis]
      [Bytes in flight: 1460]
    TCP segment data (1460 bytes)
  1200 56.430911000 10.236.64.26       128.119.245.12     TCP      1514 [TCP segment of a reassembled PDU]

⊞ Frame 1200: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface 0
⊞ Ethernet II, Src: 74:4c:a1:a5:b4:31 (74:4c:a1:a5:b4:31), Dst: 30:0d:9e:22:dc:5c (30:0d:9e:22:dc:5c)
⊞ Internet Protocol Version 4, Src: 10.236.64.26 (10.236.64.26), Dst: 128.119.245.12 (128.119.245.12)
⊟ Transmission Control Protocol, Src Port: 8179 (8179), Dst Port: http (80), Seq: 1461, Ack: 1, Len: 1460
    Source port: 8179 (8179)
    Destination port: http (80)
    [Stream index: 34]
    Sequence number: 1461     (relative sequence number)
    [Next sequence number: 2921     (relative sequence number)]
    Acknowledgment number: 1     (relative ack number)
    Header length: 20 bytes
  ⊞ Flags: 0x010 (ACK)
    Window size value: 513
    [Calculated window size: 131328]
    [Window size scaling factor: 256]
  ⊟ Checksum: 0xf12f [validation disabled]
      [Good Checksum: False]
      [Bad Checksum: False]
  ⊟ [SEQ/ACK analysis]
      [Bytes in flight: 2920]
    [Reassembled PDU in frame: 1350]
    TCP segment data (1460 bytes)
```



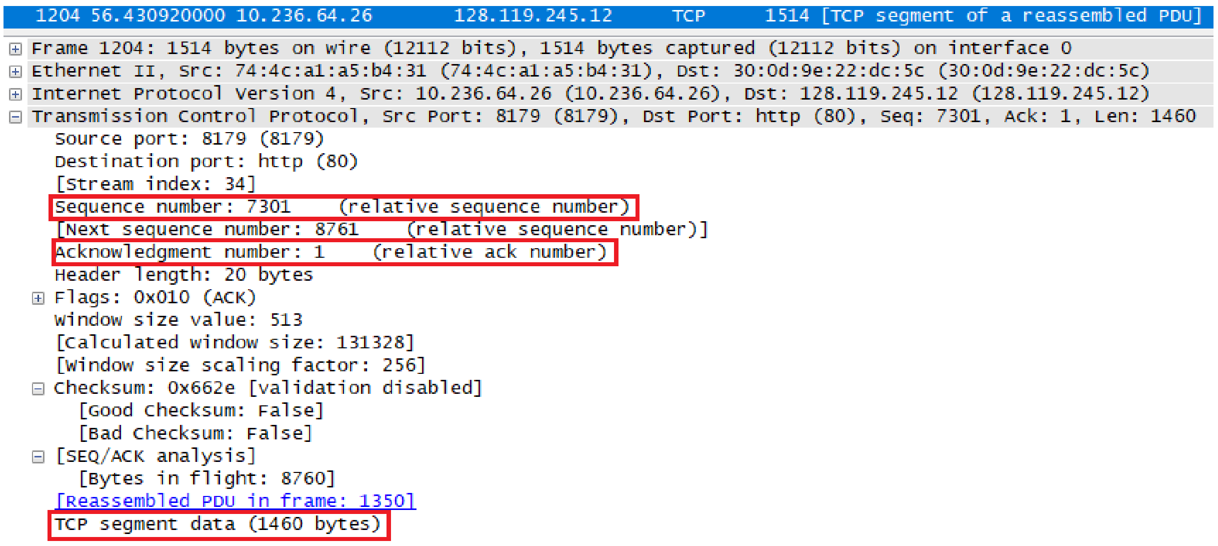
```
 1201 56.430913000 10.236.64.26        128.119.245.12     TCP      1514 [TCP segment of a reassembled PDU]
⊞ Frame 1201: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface 0
⊞ Ethernet II, Src: 74:4c:a1:a5:b4:31 (74:4c:a1:a5:b4:31), Dst: 30:0d:9e:22:dc:5c (30:0d:9e:22:dc:5c)
⊞ Internet Protocol Version 4, Src: 10.236.64.26 (10.236.64.26), Dst: 128.119.245.12 (128.119.245.12)
⊟ Transmission Control Protocol, Src Port: 8179 (8179), Dst Port: http (80), Seq: 2921, Ack: 1, Len: 1460
    Source port: 8179 (8179)
    Destination port: http (80)
    [Stream index: 34]
    Sequence number: 2921    (relative sequence number)
    [Next sequence number: 4381    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    Header length: 20 bytes
  ⊞ Flags: 0x010 (ACK)
    Window size value: 513
    [Calculated window size: 131328]
    [Window size scaling factor: 256]
  ⊟ Checksum: 0x0751 [validation disabled]
      [Good Checksum: False]
      [Bad Checksum: False]
  ⊟ [SEQ/ACK analysis]
      [Bytes in flight: 4380]
    [Reassembled PDU in frame: 1350]
    TCP segment data (1460 bytes)
```



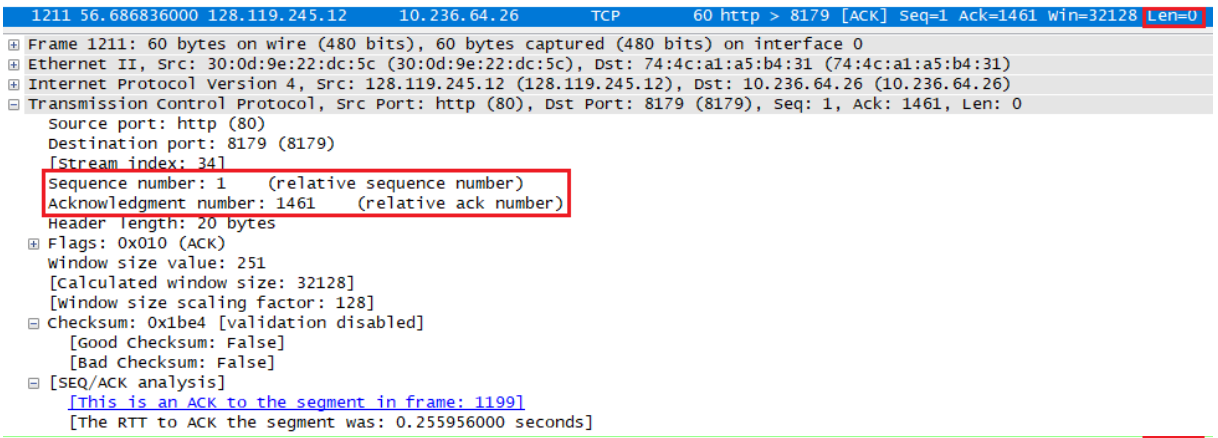
```
 1202 56.430914000 10.236.64.26        128.119.245.12     TCP      1514 [TCP segment of a reassembled PDU]
⊞ Frame 1202: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface 0
⊞ Ethernet II, Src: 74:4c:a1:a5:b4:31 (74:4c:a1:a5:b4:31), Dst: 30:0d:9e:22:dc:5c (30:0d:9e:22:dc:5c)
⊞ Internet Protocol Version 4, Src: 10.236.64.26 (10.236.64.26), Dst: 128.119.245.12 (128.119.245.12)
⊟ Transmission Control Protocol, Src Port: 8179 (8179), Dst Port: http (80), Seq: 4381, Ack: 1, Len: 1460
    Source port: 8179 (8179)
    Destination port: http (80)
    [Stream index: 34]
    Sequence number: 4381    (relative sequence number)
    [Next sequence number: 5841    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    Header length: 20 bytes
  ⊞ Flags: 0x010 (ACK)
    Window size value: 513
    [Calculated window size: 131328]
    [Window size scaling factor: 256]
  ⊟ Checksum: 0x4d3b [validation disabled]
      [Good Checksum: False]
      [Bad Checksum: False]
  ⊟ [SEQ/ACK analysis]
      [Bytes in flight: 5840]
    [Reassembled PDU in frame: 1350]
    TCP segment data (1460 bytes)
```



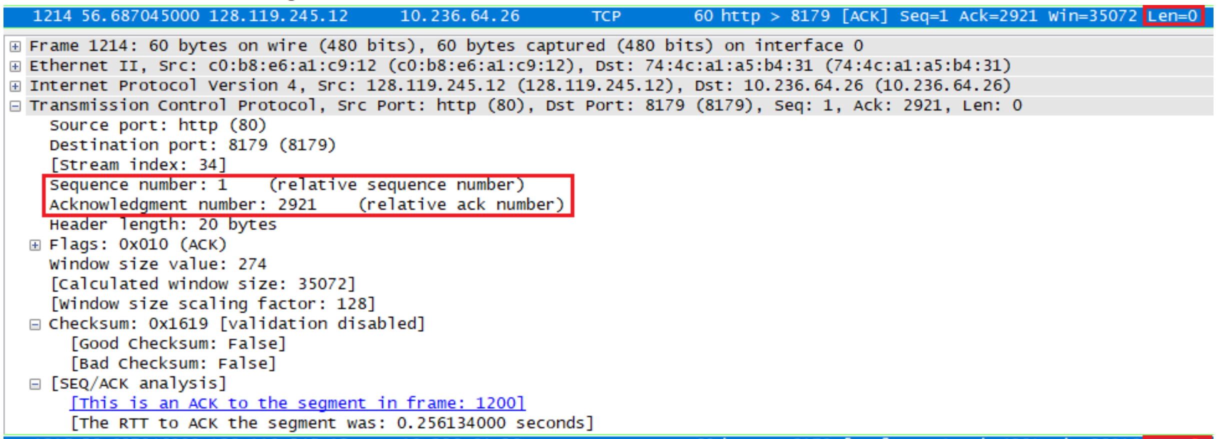
```
 1203 56.430916000 10.236.64.26        128.119.245.12     TCP      1514 [TCP segment of a reassembled PDU]
⊞ Frame 1203: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface 0
⊞ Ethernet II, Src: 74:4c:a1:a5:b4:31 (74:4c:a1:a5:b4:31), Dst: 30:0d:9e:22:dc:5c (30:0d:9e:22:dc:5c)
⊞ Internet Protocol Version 4, Src: 10.236.64.26 (10.236.64.26), Dst: 128.119.245.12 (128.119.245.12)
⊟ Transmission Control Protocol, Src Port: 8179 (8179), Dst Port: http (80), Seq: 5841, Ack: 1, Len: 1460
    Source port: 8179 (8179)
    Destination port: http (80)
    [Stream index: 34]
    Sequence number: 5841    (relative sequence number)
    [Next sequence number: 7301    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    Header length: 20 bytes
  ⊞ Flags: 0x010 (ACK)
    Window size value: 513
    [Calculated window size: 131328]
    [Window size scaling factor: 256]
  ⊟ Checksum: 0x9719 [validation disabled]
      [Good Checksum: False]
      [Bad Checksum: False]
  ⊟ [SEQ/ACK analysis]
      [Bytes in flight: 7300]
    [Reassembled PDU in frame: 1350]
    TCP segment data (1460 bytes)
```



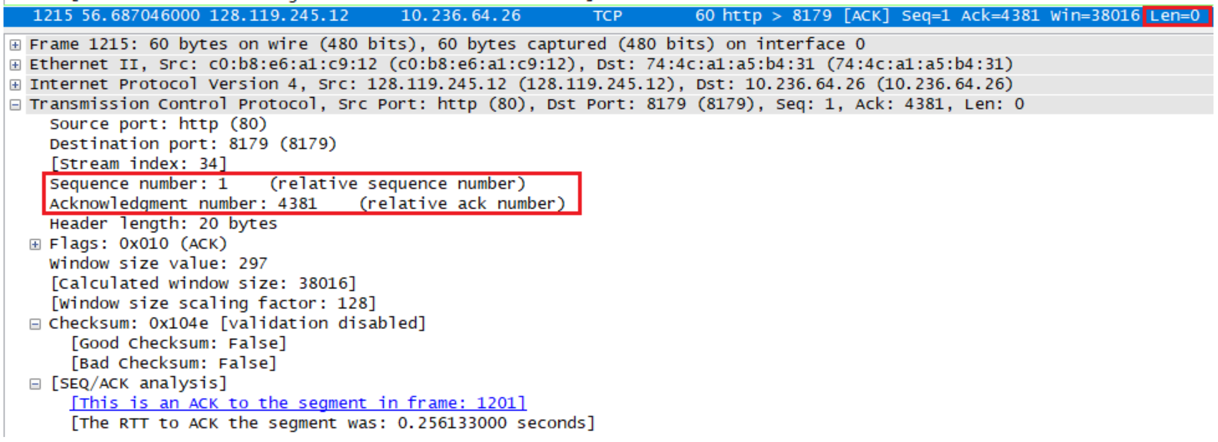
```
  1204 56.430920000 10.236.64.26        128.119.245.12     TCP      1514 [TCP segment of a reassembled PDU]
⊞ Frame 1204: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface 0
⊞ Ethernet II, Src: 74:4c:a1:a5:b4:31 (74:4c:a1:a5:b4:31), Dst: 30:0d:9e:22:dc:5c (30:0d:9e:22:dc:5c)
⊞ Internet Protocol Version 4, Src: 10.236.64.26 (10.236.64.26), Dst: 128.119.245.12 (128.119.245.12)
⊟ Transmission Control Protocol, Src Port: 8179 (8179), Dst Port: http (80), Seq: 7301, Ack: 1, Len: 1460
    Source port: 8179 (8179)
    Destination port: http (80)
    [Stream index: 34]
    Sequence number: 7301    (relative sequence number)
    [Next sequence number: 8761    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    Header length: 20 bytes
  ⊞ Flags: 0x010 (ACK)
    Window size value: 513
    [Calculated window size: 131328]
    [Window size scaling factor: 256]
  ⊟ Checksum: 0x662e [validation disabled]
      [Good Checksum: False]
      [Bad Checksum: False]
  ⊟ [SEQ/ACK analysis]
      [Bytes in flight: 8760]
    [Reassembled PDU in frame: 1350]
    TCP segment data (1460 bytes)
```



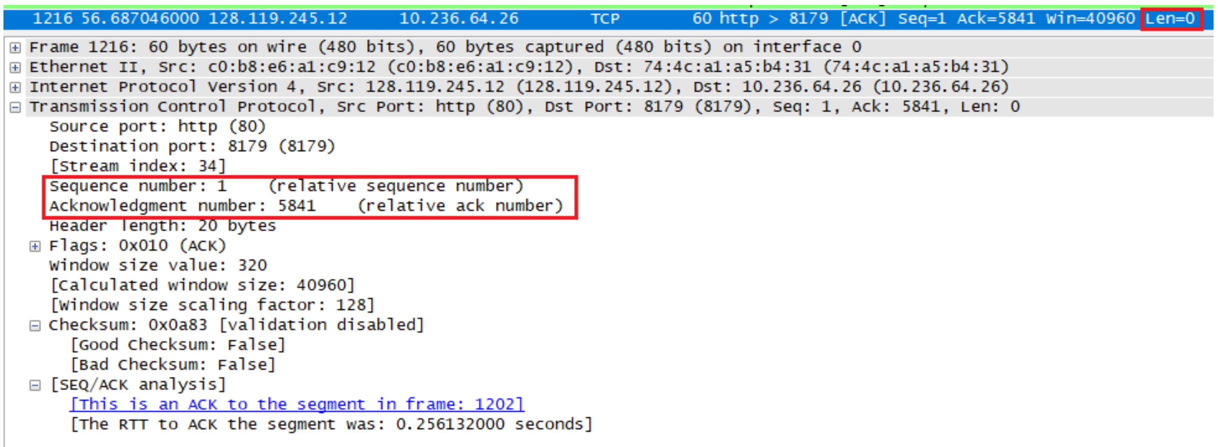
```
  1211 56.686836000 128.119.245.12      10.236.64.26       TCP       60 http > 8179 [ACK] Seq=1 Ack=1461 Win=32128 Len=0
⊞ Frame 1211: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
⊞ Ethernet II, Src: 30:0d:9e:22:dc:5c (30:0d:9e:22:dc:5c), Dst: 74:4c:a1:a5:b4:31 (74:4c:a1:a5:b4:31)
⊞ Internet Protocol Version 4, Src: 128.119.245.12 (128.119.245.12), Dst: 10.236.64.26 (10.236.64.26)
⊟ Transmission Control Protocol, Src Port: http (80), Dst Port: 8179 (8179), Seq: 1, Ack: 1461, Len: 0
    Source port: http (80)
    Destination port: 8179 (8179)
    [Stream index: 34]
    Sequence number: 1    (relative sequence number)
    Acknowledgment number: 1461    (relative ack number)
    Header length: 20 bytes
  ⊞ Flags: 0x010 (ACK)
    Window size value: 251
    [Calculated window size: 32128]
    [Window size scaling factor: 128]
  ⊟ Checksum: 0x1be4 [validation disabled]
      [Good Checksum: False]
      [Bad Checksum: False]
  ⊟ [SEQ/ACK analysis]
      [This is an ACK to the segment in frame: 1199]
      [The RTT to ACK the segment was: 0.255956000 seconds]
```



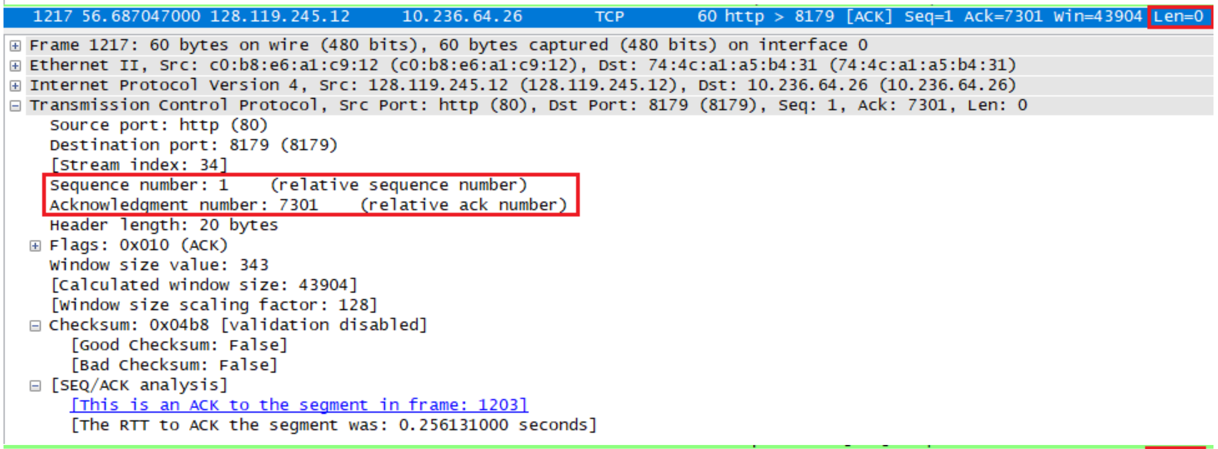
```
  1214 56.687045000 128.119.245.12      10.236.64.26       TCP       60 http > 8179 [ACK] Seq=1 Ack=2921 Win=35072 Len=0
⊞ Frame 1214: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
⊞ Ethernet II, Src: c0:b8:e6:a1:c9:12 (c0:b8:e6:a1:c9:12), Dst: 74:4c:a1:a5:b4:31 (74:4c:a1:a5:b4:31)
⊞ Internet Protocol Version 4, Src: 128.119.245.12 (128.119.245.12), Dst: 10.236.64.26 (10.236.64.26)
⊟ Transmission Control Protocol, Src Port: http (80), Dst Port: 8179 (8179), Seq: 1, Ack: 2921, Len: 0
    Source port: http (80)
    Destination port: 8179 (8179)
    [Stream index: 34]
    Sequence number: 1    (relative sequence number)
    Acknowledgment number: 2921    (relative ack number)
    Header length: 20 bytes
  ⊞ Flags: 0x010 (ACK)
    Window size value: 274
    [Calculated window size: 35072]
    [Window size scaling factor: 128]
  ⊟ Checksum: 0x1619 [validation disabled]
      [Good Checksum: False]
      [Bad Checksum: False]
  ⊟ [SEQ/ACK analysis]
      [This is an ACK to the segment in frame: 1200]
      [The RTT to ACK the segment was: 0.256134000 seconds]
```



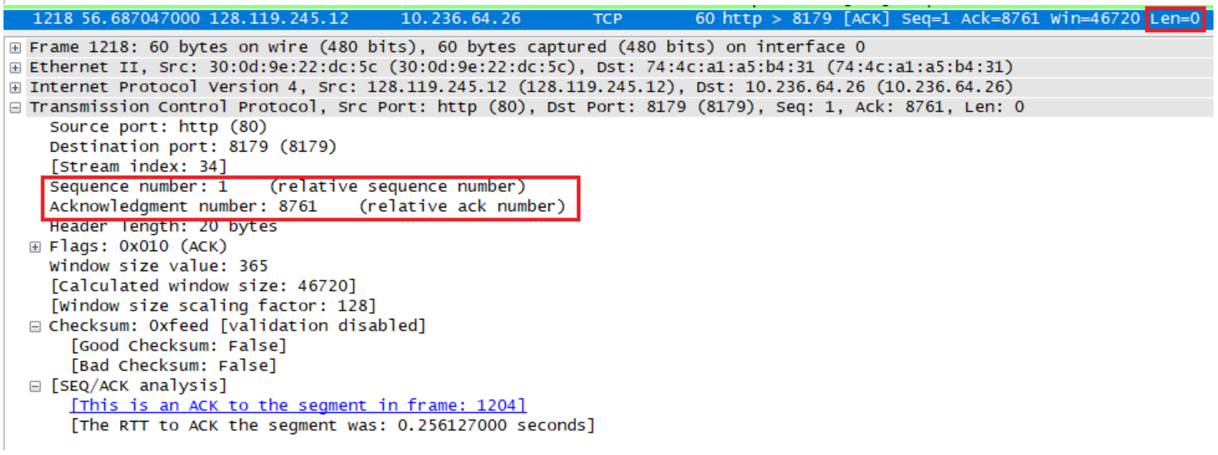
```
  1215 56.687046000 128.119.245.12      10.236.64.26       TCP       60 http > 8179 [ACK] Seq=1 Ack=4381 Win=38016 Len=0
⊞ Frame 1215: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
⊞ Ethernet II, Src: c0:b8:e6:a1:c9:12 (c0:b8:e6:a1:c9:12), Dst: 74:4c:a1:a5:b4:31 (74:4c:a1:a5:b4:31)
⊞ Internet Protocol Version 4, Src: 128.119.245.12 (128.119.245.12), Dst: 10.236.64.26 (10.236.64.26)
⊟ Transmission Control Protocol, Src Port: http (80), Dst Port: 8179 (8179), Seq: 1, Ack: 4381, Len: 0
    Source port: http (80)
    Destination port: 8179 (8179)
    [Stream index: 34]
    Sequence number: 1    (relative sequence number)
    Acknowledgment number: 4381    (relative ack number)
    Header length: 20 bytes
  ⊞ Flags: 0x010 (ACK)
    Window size value: 297
    [Calculated window size: 38016]
    [Window size scaling factor: 128]
  ⊟ Checksum: 0x104e [validation disabled]
      [Good Checksum: False]
      [Bad Checksum: False]
  ⊟ [SEQ/ACK analysis]
      [This is an ACK to the segment in frame: 1201]
      [The RTT to ACK the segment was: 0.256133000 seconds]
```



```
 1216 56.687046000 128.119.245.12      10.236.64.26        TCP        60 http > 8179 [ACK] Seq=1 Ack=5841 Win=40960 Len=0
⊞ Frame 1216: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
⊞ Ethernet II, Src: c0:b8:e6:a1:c9:12 (c0:b8:e6:a1:c9:12), Dst: 74:4c:a1:a5:b4:31 (74:4c:a1:a5:b4:31)
⊞ Internet Protocol Version 4, Src: 128.119.245.12 (128.119.245.12), Dst: 10.236.64.26 (10.236.64.26)
⊟ Transmission Control Protocol, Src Port: http (80), Dst Port: 8179 (8179), Seq: 1, Ack: 5841, Len: 0
    Source port: http (80)
    Destination port: 8179 (8179)
    [Stream index: 34]
    Sequence number: 1     (relative sequence number)
    Acknowledgment number: 5841     (relative ack number)
    Header length: 20 bytes
  ⊞ Flags: 0x010 (ACK)
    Window size value: 320
    [Calculated window size: 40960]
    [Window size scaling factor: 128]
  ⊟ Checksum: 0x0a83 [validation disabled]
      [Good Checksum: False]
      [Bad Checksum: False]
  ⊟ [SEQ/ACK analysis]
      [This is an ACK to the segment in frame: 1202]
      [The RTT to ACK the segment was: 0.256132000 seconds]
```



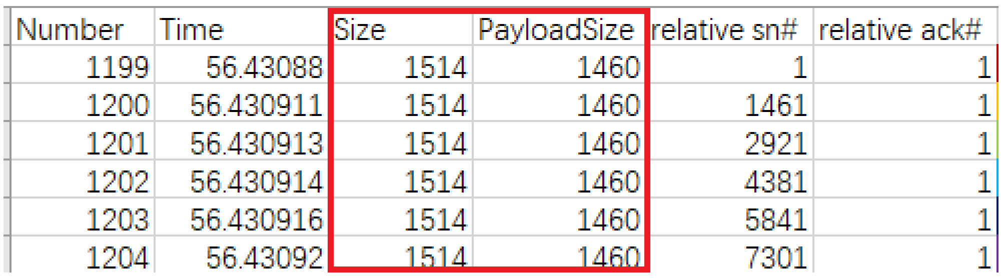
```
 1217 56.687047000 128.119.245.12      10.236.64.26        TCP        60 http > 8179 [ACK] Seq=1 Ack=7301 Win=43904 Len=0
⊞ Frame 1217: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
⊞ Ethernet II, Src: c0:b8:e6:a1:c9:12 (c0:b8:e6:a1:c9:12), Dst: 74:4c:a1:a5:b4:31 (74:4c:a1:a5:b4:31)
⊞ Internet Protocol Version 4, Src: 128.119.245.12 (128.119.245.12), Dst: 10.236.64.26 (10.236.64.26)
⊟ Transmission Control Protocol, Src Port: http (80), Dst Port: 8179 (8179), Seq: 1, Ack: 7301, Len: 0
    Source port: http (80)
    Destination port: 8179 (8179)
    [Stream index: 34]
    Sequence number: 1     (relative sequence number)
    Acknowledgment number: 7301     (relative ack number)
    Header length: 20 bytes
  ⊞ Flags: 0x010 (ACK)
    Window size value: 343
    [Calculated window size: 43904]
    [Window size scaling factor: 128]
  ⊟ Checksum: 0x04b8 [validation disabled]
      [Good Checksum: False]
      [Bad Checksum: False]
  ⊟ [SEQ/ACK analysis]
      [This is an ACK to the segment in frame: 1203]
      [The RTT to ACK the segment was: 0.256131000 seconds]
```



```
 1218 56.687047000 128.119.245.12      10.236.64.26        TCP        60 http > 8179 [ACK] Seq=1 Ack=8761 Win=46720 Len=0
⊞ Frame 1218: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
⊞ Ethernet II, Src: 30:0d:9e:22:dc:5c (30:0d:9e:22:dc:5c), Dst: 74:4c:a1:a5:b4:31 (74:4c:a1:a5:b4:31)
⊞ Internet Protocol Version 4, Src: 128.119.245.12 (128.119.245.12), Dst: 10.236.64.26 (10.236.64.26)
⊟ Transmission Control Protocol, Src Port: http (80), Dst Port: 8179 (8179), Seq: 1, Ack: 8761, Len: 0
    Source port: http (80)
    Destination port: 8179 (8179)
    [Stream index: 34]
    Sequence number: 1     (relative sequence number)
    Acknowledgment number: 8761     (relative ack number)
    Header length: 20 bytes
  ⊞ Flags: 0x010 (ACK)
    Window size value: 365
    [Calculated window size: 46720]
    [Window size scaling factor: 128]
  ⊟ Checksum: 0xfeed [validation disabled]
      [Good Checksum: False]
      [Bad Checksum: False]
  ⊟ [SEQ/ACK analysis]
      [This is an ACK to the segment in frame: 1204]
      [The RTT to ACK the segment was: 0.256127000 seconds]
```

Q8: What is the length of each of the first six TCP segments?

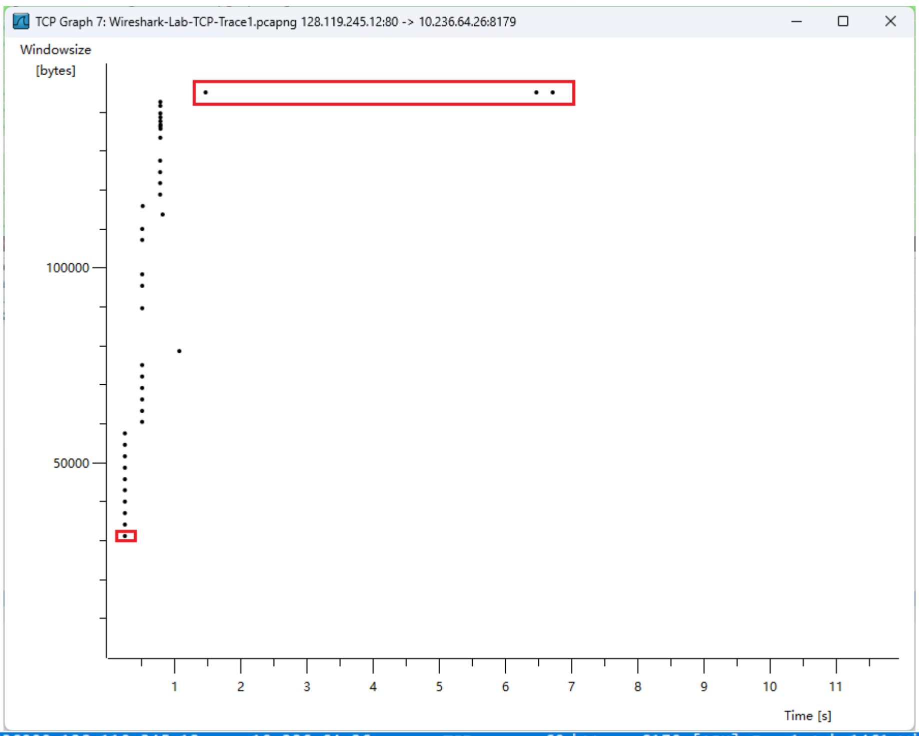
A8: 这六个 TCP 段的长度均为 1514，有效负载长度均为 1460；

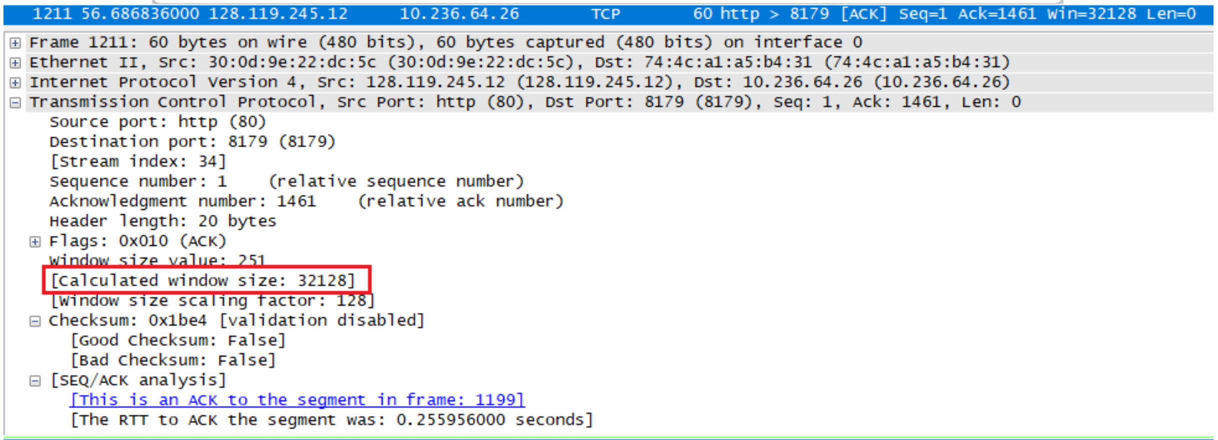


Number	Time	Size	PayloadSize	relative sn#	relative ack#
1199	56.43088	1514	1460	1	1
1200	56.430911	1514	1460	1461	1
1201	56.430913	1514	1460	2921	1
1202	56.430914	1514	1460	4381	1
1203	56.430916	1514	1460	5841	1
1204	56.43092	1514	1460	7301	1

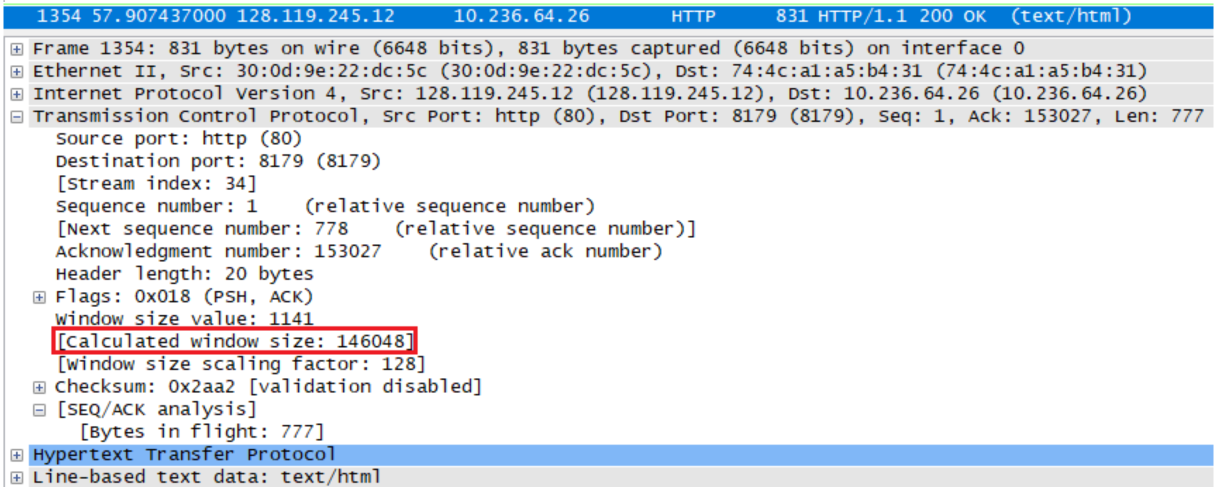
Q9: What is the minimum amount of available buffer space advertised at the received for the entire trace? Does the lack of receiver buffer space ever throttle the sender?

A9: 对于整个追踪，接收处通知发送方的最小窗口为 32128，最大窗口为 146048；
在本次追踪中，不存在接收方缓冲空间的缺乏限制发送方的情况；

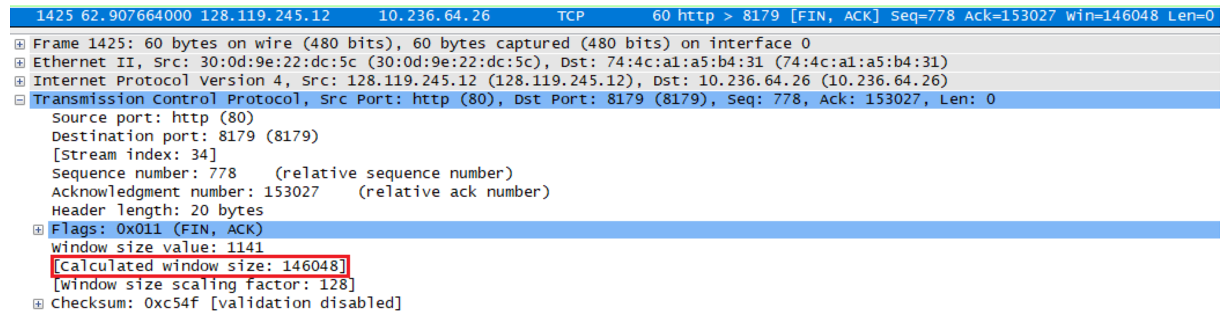




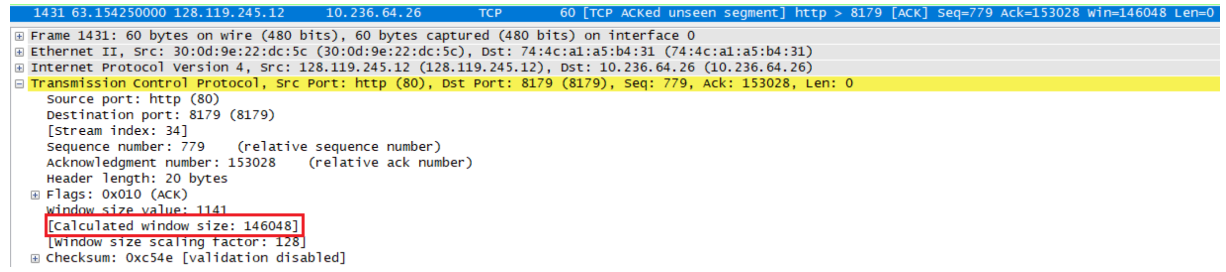
```
 1211 56.686836000 128.119.245.12      10.236.64.26        TCP        60 http > 8179 [ACK] Seq=1 Ack=1461 Win=32128 Len=0
⊞ Frame 1211: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
⊞ Ethernet II, Src: 30:0d:9e:22:dc:5c (30:0d:9e:22:dc:5c), Dst: 74:4c:a1:a5:b4:31 (74:4c:a1:a5:b4:31)
⊞ Internet Protocol Version 4, Src: 128.119.245.12 (128.119.245.12), Dst: 10.236.64.26 (10.236.64.26)
⊟ Transmission Control Protocol, Src Port: http (80), Dst Port: 8179 (8179), Seq: 1, Ack: 1461, Len: 0
    Source port: http (80)
    Destination port: 8179 (8179)
    [Stream index: 34]
    Sequence number: 1    (relative sequence number)
    Acknowledgment number: 1461    (relative ack number)
    Header length: 20 bytes
  ⊞ Flags: 0x010 (ACK)
    Window size value: 251
    [Calculated window size: 32128]
    [Window size scaling factor: 128]
  ⊟ Checksum: 0x1be4 [validation disabled]
      [Good Checksum: False]
      [Bad Checksum: False]
  ⊟ [SEQ/ACK analysis]
      [This is an ACK to the segment in frame: 1199]
      [The RTT to ACK the segment was: 0.255956000 seconds]
```



```
 1354 57.907437000 128.119.245.12      10.236.64.26        HTTP      831 HTTP/1.1 200 OK  (text/html)
⊞ Frame 1354: 831 bytes on wire (6648 bits), 831 bytes captured (6648 bits) on interface 0
⊞ Ethernet II, Src: 30:0d:9e:22:dc:5c (30:0d:9e:22:dc:5c), Dst: 74:4c:a1:a5:b4:31 (74:4c:a1:a5:b4:31)
⊞ Internet Protocol Version 4, Src: 128.119.245.12 (128.119.245.12), Dst: 10.236.64.26 (10.236.64.26)
⊟ Transmission Control Protocol, Src Port: http (80), Dst Port: 8179 (8179), Seq: 1, Ack: 153027, Len: 777
    Source port: http (80)
    Destination port: 8179 (8179)
    [Stream index: 34]
    Sequence number: 1    (relative sequence number)
    [Next sequence number: 778    (relative sequence number)]
    Acknowledgment number: 153027    (relative ack number)
    Header length: 20 bytes
  ⊞ Flags: 0x018 (PSH, ACK)
    Window size value: 1141
    [Calculated window size: 146048]
    [Window size scaling factor: 128]
  ⊞ Checksum: 0x2aa2 [validation disabled]
  ⊟ [SEQ/ACK analysis]
      [Bytes in flight: 777]
⊞ Hypertext Transfer Protocol
⊞ Line-based text data: text/html
```



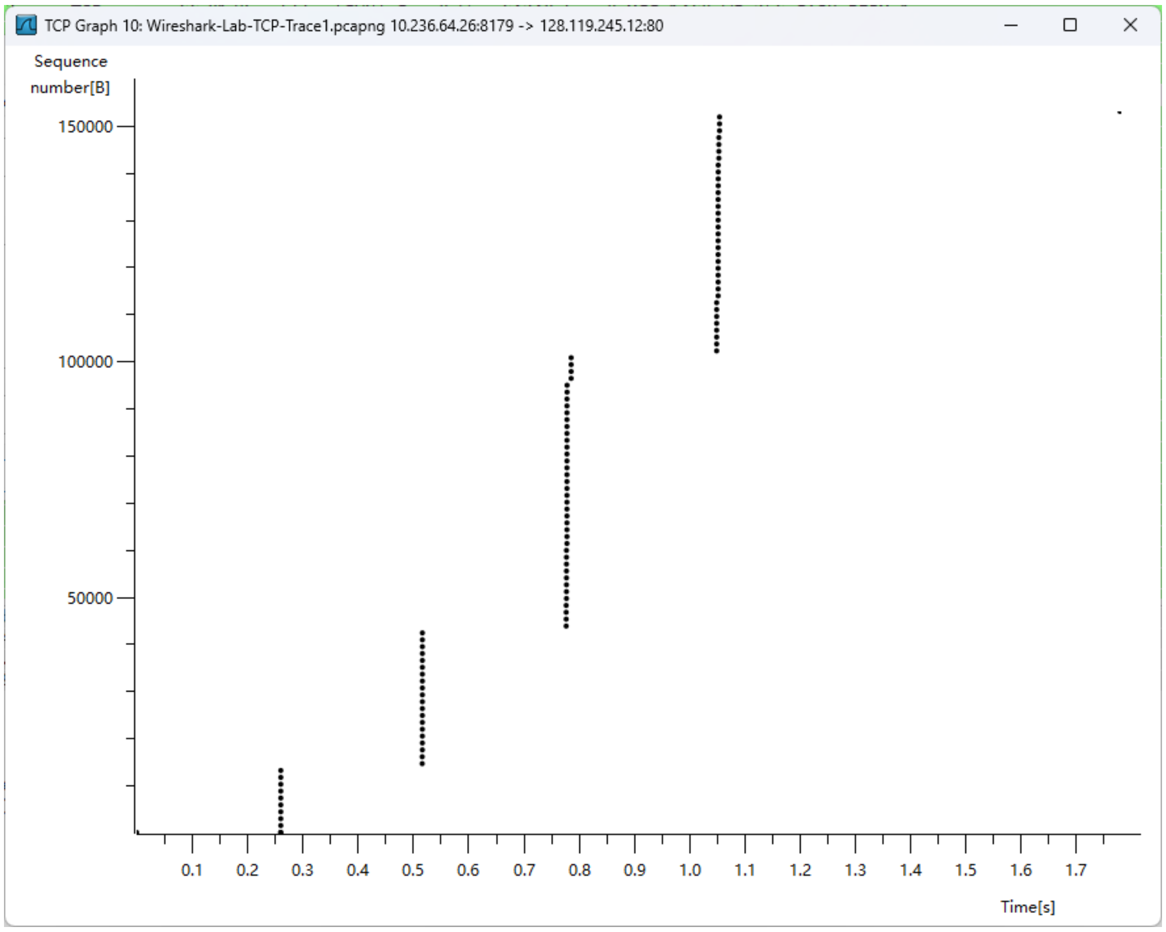
```
1425 62.907664000 128.119.245.12      10.236.64.26        TCP       60 http > 8179 [FIN, ACK] Seq=778 Ack=153027 Win=146048 Len=0
⊞ Frame 1425: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
⊞ Ethernet II, Src: 30:0d:9e:22:dc:5c (30:0d:9e:22:dc:5c), Dst: 74:4c:a1:a5:b4:31 (74:4c:a1:a5:b4:31)
⊞ Internet Protocol Version 4, Src: 128.119.245.12 (128.119.245.12), Dst: 10.236.64.26 (10.236.64.26)
⊟ Transmission Control Protocol, Src Port: http (80), Dst Port: 8179 (8179), Seq: 778, Ack: 153027, Len: 0
    Source port: http (80)
    Destination port: 8179 (8179)
    [Stream index: 34]
    Sequence number: 778    (relative sequence number)
    Acknowledgment number: 153027    (relative ack number)
    Header length: 20 bytes
  ⊞ Flags: 0x011 (FIN, ACK)
    Window size value: 1141
    [Calculated window size: 146048]
    [Window size scaling factor: 128]
  ⊞ Checksum: 0xc54f [validation disabled]
```



```
1431 63.154250000 128.119.245.12      10.236.64.26        TCP       60 [TCP ACKed unseen segment] http > 8179 [ACK] Seq=779 Ack=153028 Win=146048 Len=0
⊞ Frame 1431: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
⊞ Ethernet II, Src: 30:0d:9e:22:dc:5c (30:0d:9e:22:dc:5c), Dst: 74:4c:a1:a5:b4:31 (74:4c:a1:a5:b4:31)
⊞ Internet Protocol Version 4, Src: 128.119.245.12 (128.119.245.12), Dst: 10.236.64.26 (10.236.64.26)
⊟ Transmission Control Protocol, Src Port: http (80), Dst Port: 8179 (8179), Seq: 779, Ack: 153028, Len: 0
    Source port: http (80)
    Destination port: 8179 (8179)
    [Stream index: 34]
    Sequence number: 779    (relative sequence number)
    Acknowledgment number: 153028    (relative ack number)
    Header length: 20 bytes
  ⊞ Flags: 0x010 (ACK)
    Window size value: 1141
    [Calculated window size: 146048]
    [Window size scaling factor: 128]
  ⊞ Checksum: 0xc54e [validation disabled]
```

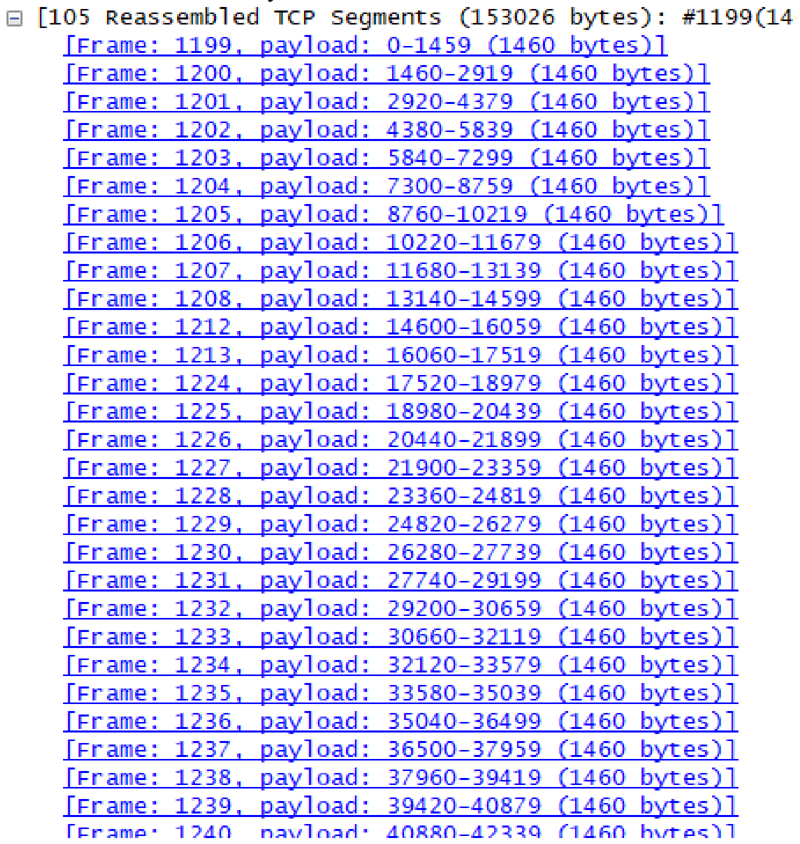
Q10: Are there any retransmitted segments in the trace file? What did you check for (in the trace) in order to answer this question?

A10: 对于整个追踪，并没有出现发送方重传的现象，从 Time-Sequence Graph (Stevens)可以看出，发送方每隔一定的时间发送一连串的分组，这些分组的序列号 Seq 没有重复，呈增长形势，没有出现重传的情况；

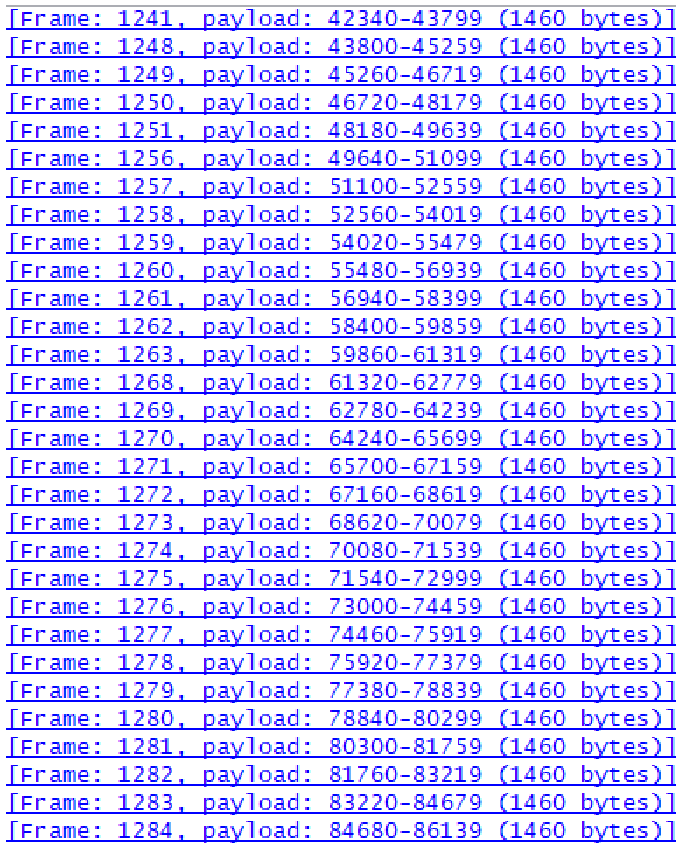


Q11: How much data does the receiver typically acknowledge in an ACK? Can you identify cases where the receiver is ACKing every other received segment (see Table 3.2 on page 247 in the text).

A11: 接收方通常在 ACK 中确认 1460bytes 的数据；

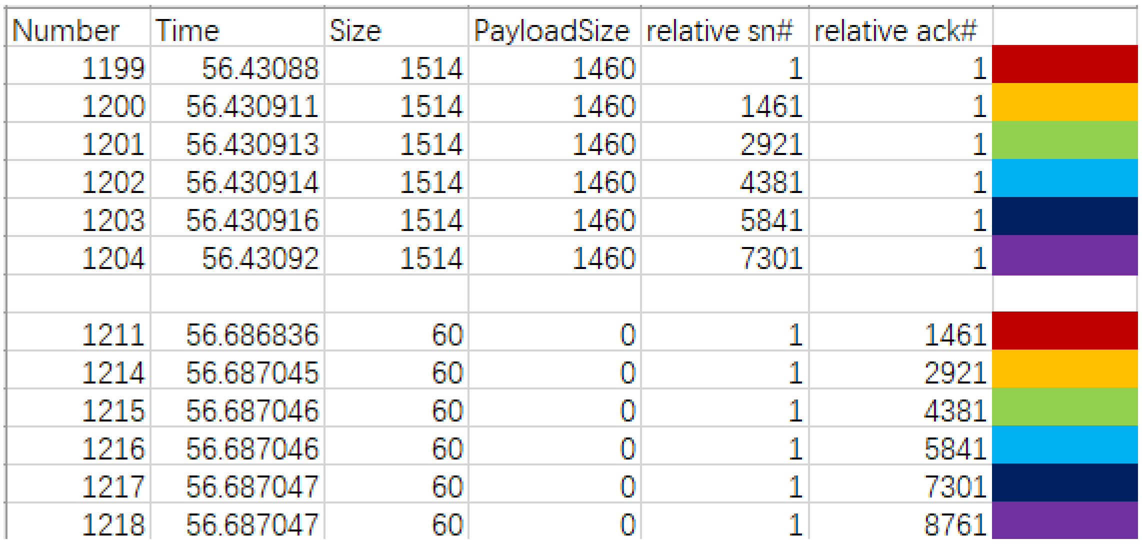


```
⊟ [105 Reassembled TCP Segments (153026 bytes): #1199(14
    [Frame: 1199, payload: 0-1459 (1460 bytes)]
    [Frame: 1200, payload: 1460-2919 (1460 bytes)]
    [Frame: 1201, payload: 2920-4379 (1460 bytes)]
    [Frame: 1202, payload: 4380-5839 (1460 bytes)]
    [Frame: 1203, payload: 5840-7299 (1460 bytes)]
    [Frame: 1204, payload: 7300-8759 (1460 bytes)]
    [Frame: 1205, payload: 8760-10219 (1460 bytes)]
    [Frame: 1206, payload: 10220-11679 (1460 bytes)]
    [Frame: 1207, payload: 11680-13139 (1460 bytes)]
    [Frame: 1208, payload: 13140-14599 (1460 bytes)]
    [Frame: 1212, payload: 14600-16059 (1460 bytes)]
    [Frame: 1213, payload: 16060-17519 (1460 bytes)]
    [Frame: 1224, payload: 17520-18979 (1460 bytes)]
    [Frame: 1225, payload: 18980-20439 (1460 bytes)]
    [Frame: 1226, payload: 20440-21899 (1460 bytes)]
    [Frame: 1227, payload: 21900-23359 (1460 bytes)]
    [Frame: 1228, payload: 23360-24819 (1460 bytes)]
    [Frame: 1229, payload: 24820-26279 (1460 bytes)]
    [Frame: 1230, payload: 26280-27739 (1460 bytes)]
    [Frame: 1231, payload: 27740-29199 (1460 bytes)]
    [Frame: 1232, payload: 29200-30659 (1460 bytes)]
    [Frame: 1233, payload: 30660-32119 (1460 bytes)]
    [Frame: 1234, payload: 32120-33579 (1460 bytes)]
    [Frame: 1235, payload: 33580-35039 (1460 bytes)]
    [Frame: 1236, payload: 35040-36499 (1460 bytes)]
    [Frame: 1237, payload: 36500-37959 (1460 bytes)]
    [Frame: 1238, payload: 37960-39419 (1460 bytes)]
    [Frame: 1239, payload: 39420-40879 (1460 bytes)]
    [Frame: 1240, payload: 40880-42339 (1460 bytes)]
```



```
    [Frame: 1241, payload: 42340-43799 (1460 bytes)]
    [Frame: 1248, payload: 43800-45259 (1460 bytes)]
    [Frame: 1249, payload: 45260-46719 (1460 bytes)]
    [Frame: 1250, payload: 46720-48179 (1460 bytes)]
    [Frame: 1251, payload: 48180-49639 (1460 bytes)]
    [Frame: 1256, payload: 49640-51099 (1460 bytes)]
    [Frame: 1257, payload: 51100-52559 (1460 bytes)]
    [Frame: 1258, payload: 52560-54019 (1460 bytes)]
    [Frame: 1259, payload: 54020-55479 (1460 bytes)]
    [Frame: 1260, payload: 55480-56939 (1460 bytes)]
    [Frame: 1261, payload: 56940-58399 (1460 bytes)]
    [Frame: 1262, payload: 58400-59859 (1460 bytes)]
    [Frame: 1263, payload: 59860-61319 (1460 bytes)]
    [Frame: 1268, payload: 61320-62779 (1460 bytes)]
    [Frame: 1269, payload: 62780-64239 (1460 bytes)]
    [Frame: 1270, payload: 64240-65699 (1460 bytes)]
    [Frame: 1271, payload: 65700-67159 (1460 bytes)]
    [Frame: 1272, payload: 67160-68619 (1460 bytes)]
    [Frame: 1273, payload: 68620-70079 (1460 bytes)]
    [Frame: 1274, payload: 70080-71539 (1460 bytes)]
    [Frame: 1275, payload: 71540-72999 (1460 bytes)]
    [Frame: 1276, payload: 73000-74459 (1460 bytes)]
    [Frame: 1277, payload: 74460-75919 (1460 bytes)]
    [Frame: 1278, payload: 75920-77379 (1460 bytes)]
    [Frame: 1279, payload: 77380-78839 (1460 bytes)]
    [Frame: 1280, payload: 78840-80299 (1460 bytes)]
    [Frame: 1281, payload: 80300-81759 (1460 bytes)]
    [Frame: 1282, payload: 81760-83219 (1460 bytes)]
    [Frame: 1283, payload: 83220-84679 (1460 bytes)]
    [Frame: 1284, payload: 84680-86139 (1460 bytes)]
```

可以识别接收方对每一个接收段进行 ack 的情况，如 Q7/A7 中，下表中上下颜色相同的相匹配；



Number	Time	Size	PayloadSize	relative sn#	relative ack#	
1199	56.43088	1514	1460	1	1	🟥
1200	56.430911	1514	1460	1461	1	🟨
1201	56.430913	1514	1460	2921	1	🟩
1202	56.430914	1514	1460	4381	1	🟦
1203	56.430916	1514	1460	5841	1	⬛
1204	56.43092	1514	1460	7301	1	🟪
1211	56.686836	60	0	1	1461	🟥
1214	56.687045	60	0	1	2921	🟨
1215	56.687046	60	0	1	4381	🟩
1216	56.687046	60	0	1	5841	🟦
1217	56.687047	60	0	1	7301	⬛
1218	56.687047	60	0	1	8761	🟪

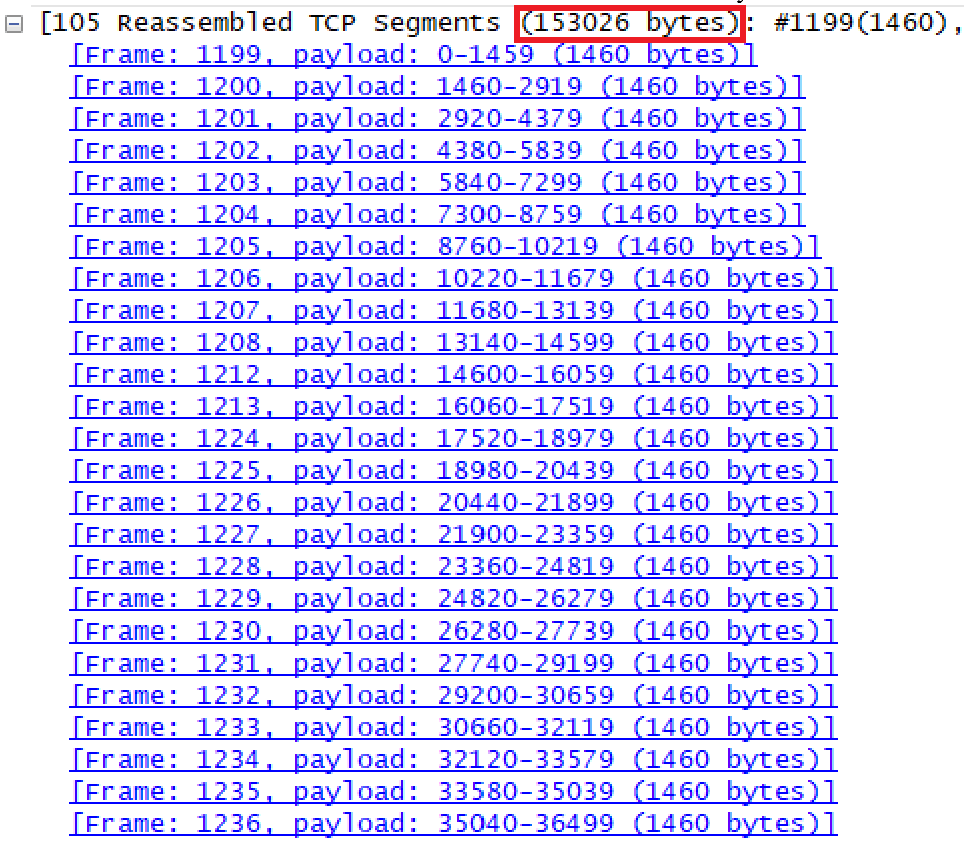
Q12: What is the throughput (bytes transferred per unit time) for the TCP connection? Explain how you calculated this value.

A12: TCP 吞吐量(单位时间内传输的字节数)由传输内容所耗费的时间决定。

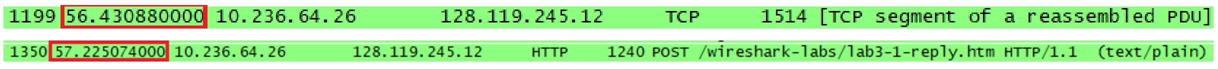
在本次追踪中，选取整个连接的时间作为参考时间段，则此次 TCP 连接的平均吞吐量为总的传输数据与总传输时间的比值。

传输的数据总量为 153026 bytes，传输花费的总时间为 57.225074-56.430880 = 0.794194 s

因此本次 TCP 连接的吞吐量为 153026/0.794194 = 192680.881497 bytes/s；



```
⊟ [105 Reassembled TCP Segments (153026 bytes): #1199(1460),
    [Frame: 1199, payload: 0-1459 (1460 bytes)]
    [Frame: 1200, payload: 1460-2919 (1460 bytes)]
    [Frame: 1201, payload: 2920-4379 (1460 bytes)]
    [Frame: 1202, payload: 4380-5839 (1460 bytes)]
    [Frame: 1203, payload: 5840-7299 (1460 bytes)]
    [Frame: 1204, payload: 7300-8759 (1460 bytes)]
    [Frame: 1205, payload: 8760-10219 (1460 bytes)]
    [Frame: 1206, payload: 10220-11679 (1460 bytes)]
    [Frame: 1207, payload: 11680-13139 (1460 bytes)]
    [Frame: 1208, payload: 13140-14599 (1460 bytes)]
    [Frame: 1212, payload: 14600-16059 (1460 bytes)]
    [Frame: 1213, payload: 16060-17519 (1460 bytes)]
    [Frame: 1224, payload: 17520-18979 (1460 bytes)]
    [Frame: 1225, payload: 18980-20439 (1460 bytes)]
    [Frame: 1226, payload: 20440-21899 (1460 bytes)]
    [Frame: 1227, payload: 21900-23359 (1460 bytes)]
    [Frame: 1228, payload: 23360-24819 (1460 bytes)]
    [Frame: 1229, payload: 24820-26279 (1460 bytes)]
    [Frame: 1230, payload: 26280-27739 (1460 bytes)]
    [Frame: 1231, payload: 27740-29199 (1460 bytes)]
    [Frame: 1232, payload: 29200-30659 (1460 bytes)]
    [Frame: 1233, payload: 30660-32119 (1460 bytes)]
    [Frame: 1234, payload: 32120-33579 (1460 bytes)]
    [Frame: 1235, payload: 33580-35039 (1460 bytes)]
    [Frame: 1236, payload: 35040-36499 (1460 bytes)]
```

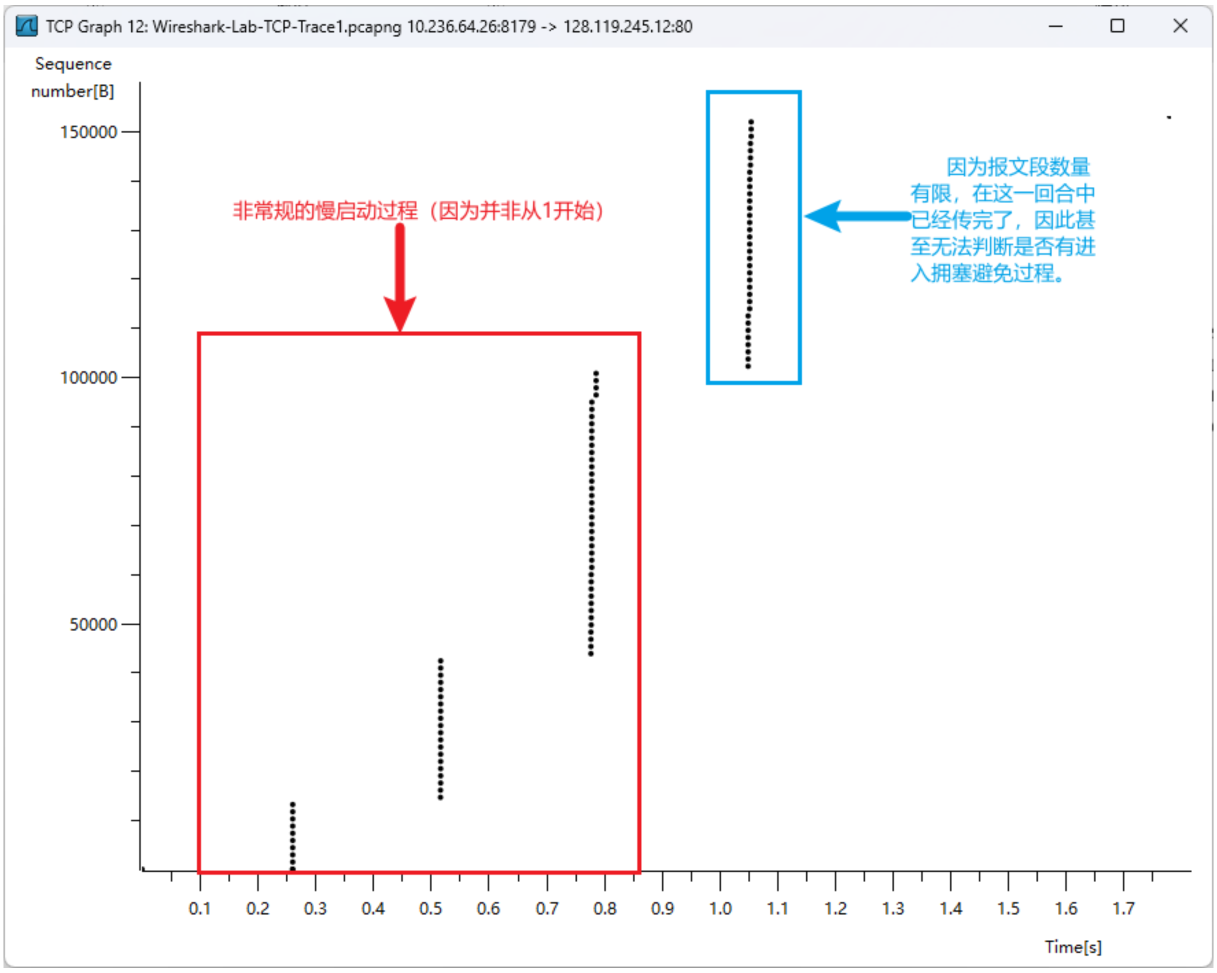


```
1199 56.430880000 10.236.64.26      128.119.245.12     TCP      1514 [TCP segment of a reassembled PDU]
1350 57.225074000 10.236.64.26      128.119.245.12     HTTP     1240 POST /wireshark-labs/lab3-1-reply.htm HTTP/1.1  (text/plain)
```

4. TCP congestion control in action

Q13: 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.

A13: 在本次追踪中，因为报文段数量有限，无法准确判定慢启动过程是否已经结束、进入拥塞避免过程；



Q14: 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

A14: 上述两个问题均已回答，见 Q13/A13；

六、结果及分析

1. Capturing a bulk TCP transfer from your computer to a remote server

我们将使用 POST 方法将存储在计算机上的文件（包含《爱丽丝梦游仙境》的 ASCII 文本）传输到 Web 服务器，在使用 Wireshark 捕获从计算机到远程服务器的文件的 TCP 传输的数据包。

2. A first look at the captured trace

我们通过 Wireshark 抓包分析 TCP 连接的行为和 HTTP 消息的传递。在抓包结果中会看到你的计算机与 gaia.cs.umass.edu 之间的一系列 TCP 和 HTTP 消息，包括 SYN 握手消息和 HTTP POST 消息等，帮助我们进一步熟悉 TCP 报文段的构成。

3. TCP Basics

在本部分，通过实践来回答问题，我们对序号、确认号的认识进一步深化，并且对于估计 RTT 有了更深刻的理解，可以识别接收方对每一个接收段进行 ACK 的情况，如 Q7/A7 中，帮助我们进一步熟悉 TCP 协议；

4. TCP congestion control in action

该部分是关于 TCP 拥塞控制的内容。通过使用 Wireshark 的 TCP 图形化工具 Time-Sequence Graph(Stevens)来绘制客户端向服务器发送的每个单位时间的数据量。通过对图片的分析，我们可以知道在传输过程中是否发生丢包、重传等情况，也可以分析哪些时间属于慢启动过程、哪些时间属于拥塞避免过程；

总的来讲，本次实验成功完成，熟悉了 TCP 协议的相关内容。

七、上机实验总结

通过本次实验，我学习了如何在 Windows11 系统上利用 Wireshark 软件对 TCP 协议进行抓包分析，实现了如下目标：

·可靠数据传输机制：TCP 使用序列号和确认号实现可靠数据传输；

·HTML 文件检索：实验中通过传输一个 150KB 的 HTML 文件，观察 TCP 的分段传输和接收；

·拥塞控制：观察 TCP 的慢启动和拥塞避免算法的运作，以避免网络拥塞；

·流量控制机制：TCP 的接收方通过窗口流量控制机制，控制发送端发送的数据量，以避免接收方负荷过重；

·TCP 连接性能：实验中还研究了计算机和服务器之间 TCP 连接的性能，包括吞吐量和往返时间。

总之，本实验通过对 TCP 协议的详细研究，深入理解了 TCP 协议的各种机制和特点，有助于我们更好地了解 TCP 协议，进而优化网络传输性能。