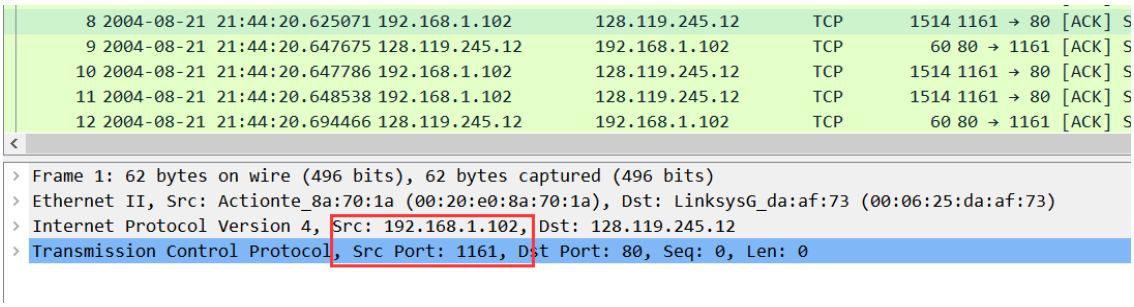
# A first look at the captured trace

实验步骤：用wireshark打开压缩包里的tcp-ethereal-trace-1即可。

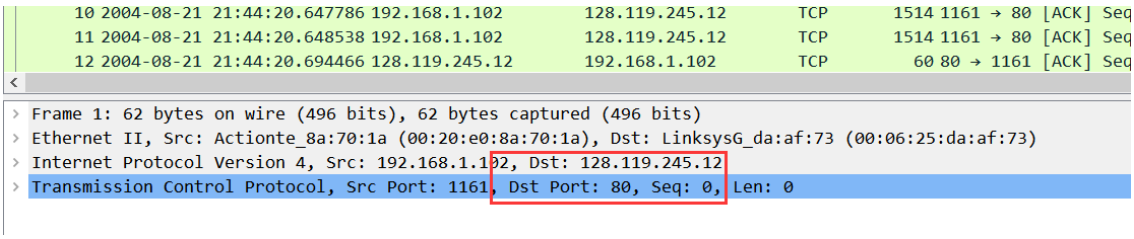
1. **What is the IP address and TCP port number used by the client computer (source) that is transferring the file to gaia.cs.umass.edu? To answer this question, it's probably easiest to select an HTTP message and explore the details of the TCP packet used to carry this HTTP message, using the "details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

   IP地址 : 192.168.1.102 端口号 : 1161




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?**

   IP地址 : 128.119.245.12　端口号 : 80

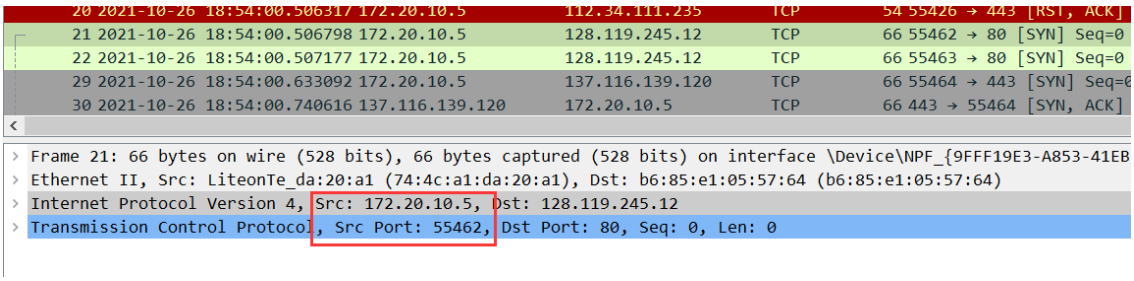



**If you have been able to create your own trace, answer the following question:**

实验步骤： 在网站http://gaia.cs.umass.edu/wireshark-labs/alice.txt上下载alice.txt，开始wireshark的捕获，再在http://gaia.cs.umass.edu/wireshark-labs/TCP-wireshark-file1.html上传文件alice.txt，停止捕获。

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?**

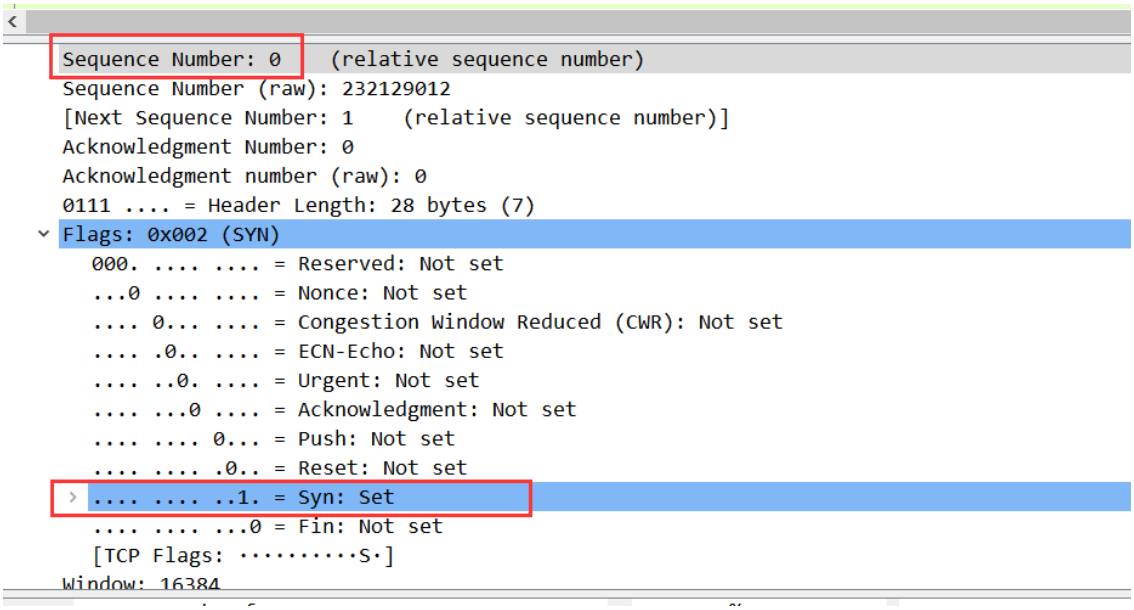
   IP地址 : 172.20.10.5 端口号 : 55462




# 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?**

   序列号：0　　SYN段被设为1




5. **What is the sequence number of the SYNACK segment sent by gaia.cs.umass.eduto 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?**

   序列号：0　　确认号：1，此处的确认号就是客户的初始序号加1。

   SYN段被设为1，这将该报文段标识为SYNACK报文段

```
Sequence Number: 0    (relative sequence number)
Sequence Number (raw): 883061785
[Next Sequence Number: 1    (relative sequence number)]
Acknowledgment Number: 1    (relative ack number)
Acknowledgment number (raw): 232129013
0111 .... = Header Length: 28 bytes (7)
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
   .... .... ...0 = Fin: Not set
   [TCP Flags: ·····A··S·]
Window: 5840
```

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.**

   序列号：1

```
[Stream index: 0]
[Conversation completeness: Incomplete, DATA (15)]
[TCP Segment Len: 565]
Sequence Number: 1    (relative sequence number)
Sequence Number (raw): 232129013
[Next Sequence Number: 566    (relative sequence number)]
Acknowledgment Number: 1    (relative ack number)
Acknowledgment number (raw): 883061786
0101 .... = Header Length: 20 bytes (5)
```

```
0000  00 06 25 da af 73 00 20  e0 8a 70 1a 08 00 45 00   ··%··s·  ··p···E·
0010  02 5d 1e 21 40 00 80 06  a2 e7 c0 a8 01 66 80 77   ·]·!@··· ·····f·w
0020  f5 0c 04 89 00 50 0d d6  01 f5 34 a2 74 1a 50 18   ·····P·· ··4·t·P·
0030  44 70 1f bd 00 00 50 4f  53 54 20 2f 65 74 68 65   Dp····PO ST /ethe
0040  72 65 61 6c 2d 6c 61 62  73 2f 6c 61 62 33 2d 31   real-lab s/lab3-1
0050  2d 72 65 70 6c 79 2e 68  74 6d 20 48 54 54 50 2f   -reply.h tm HTTP/
0060  31 2e 31 0d 0a 48 6f 73  74 3a 20 67 61 69 61 2e   1.1··Hos t: gaia.
0070  63 73 2e 75 6d 61 73 73  2e 65 64 75 0d 0a 55 73   cs.umass .edu··Us
```

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.**

   **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.***

| segment | sequence numbers | sent time | received time | RTT | EstimatedRTT |
|---|---|---|---|---|---|
| segment1 | 1 | 0.596858 | 0.624318 | 0.027460 | 0.027460 |
| segment2 | 566 | 0.612118 | 0.647675 | 0.035557 | 0.028472125 |
| segment3 | 2026 | 0.624407 | 0.694466 | 0.070059 | 0.033670484 |
| segment4 | 3486 | 0.625071 | 0.739499 | 0.114428 | 0.043765174 |
| segment5 | 4946 | 0.647786 | 0.787680 | 0.139894 | 0.055781277 |
| segment6 | 6406 | 0.648538 | 0.838183 | 0.189645 | 0.072514242 |

EstimatedRTT1 = 0.027460

EstimatedRTT2 =0.875 * 0.027460 + 0.125 * 0.035557 = 0.028472125

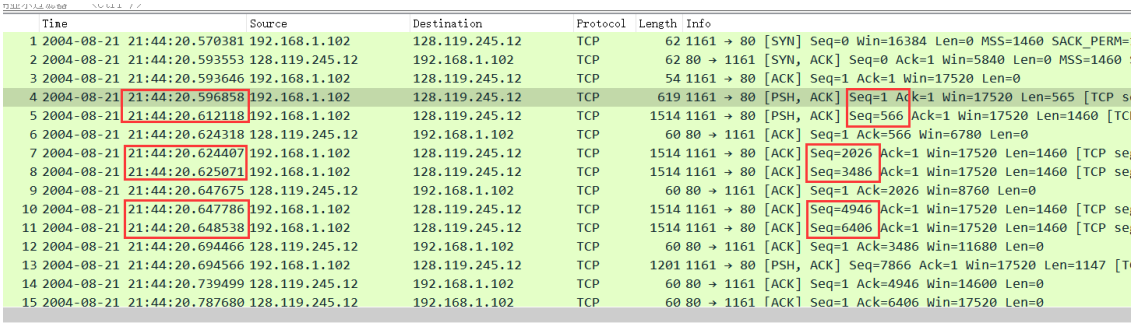
EstimatedRTT2 =0.875 * 0.028472125 + 0.125 * 0.070059 = 0.033670484

EstimatedRTT2 =0.875 * 0.033670484 + 0.125 * 0.114428 = 0.043765174

EstimatedRTT2 =0.875 * 0.043765174 + 0.125 * 0.139894 = 0.055781277

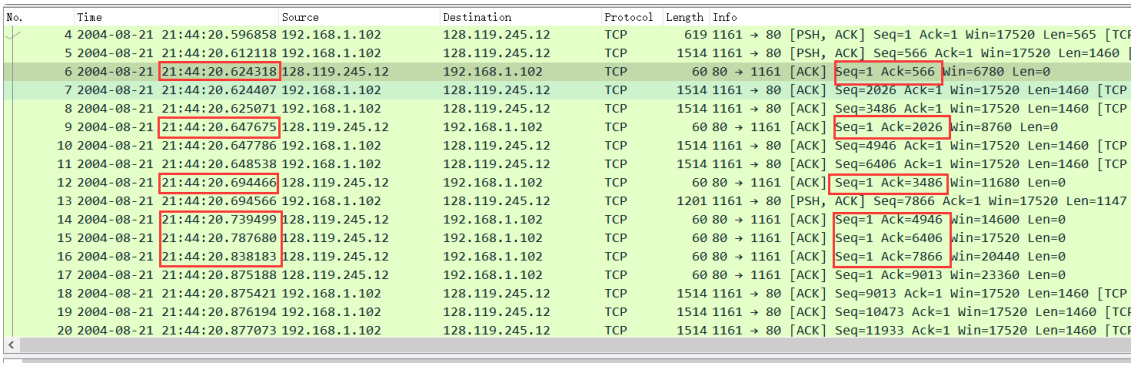
EstimatedRTT2 =0.875 * 0.055781277 + 0.125 * 0.189645 = 0.072514242

从客户发向服务器的报文段：

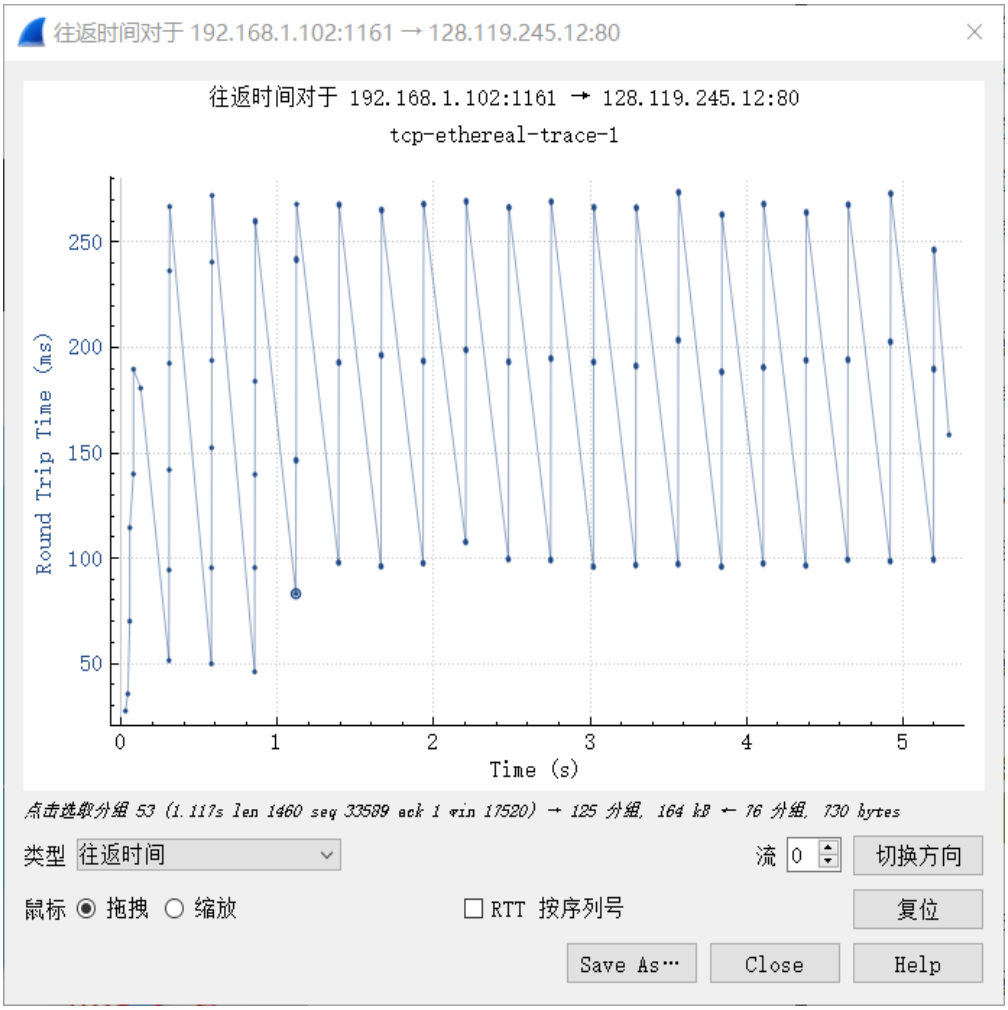


| | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 1 | 2004-08-21 21:44:20.570381 | 192.168.1.102 | 128.119.245.12 | TCP | 62 | 1161 → 80 [SYN] Seq=0 Win=16384 Len=0 MSS=1460 SACK_PERM= |
| 2 | 2004-08-21 21:44:20.593553 | 128.119.245.12 | 192.168.1.102 | TCP | 62 | 80 → 1161 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 |
| 3 | 2004-08-21 21:44:20.593646 | 192.168.1.102 | 128.119.245.12 | TCP | 54 | 1161 → 80 [ACK] Seq=1 Ack=1 Win=17520 Len=0 |
| 4 | 2004-08-21 21:44:20.596858 | 192.168.1.102 | 128.119.245.12 | TCP | 619 | 1161 → 80 [PSH, ACK] Seq=1 Ack=1 Win=17520 Len=565 [TCP s |
| 5 | 2004-08-21 21:44:20.612118 | 192.168.1.102 | 128.119.245.12 | TCP | 1514 | 1161 → 80 [PSH, ACK] Seq=566 Ack=1 Win=17520 Len=1460 [TC |
| 6 | 2004-08-21 21:44:20.624318 | 128.119.245.12 | 192.168.1.102 | TCP | 60 | 80 → 1161 [ACK] Seq=1 Ack=566 Win=6780 Len=0 |
| 7 | 2004-08-21 21:44:20.624407 | 192.168.1.102 | 128.119.245.12 | TCP | 1514 | 1161 → 80 [ACK] Seq=2026 Ack=1 Win=17520 Len=1460 [TCP se |
| 8 | 2004-08-21 21:44:20.625071 | 192.168.1.102 | 128.119.245.12 | TCP | 1514 | 1161 → 80 [ACK] Seq=3486 Ack=1 Win=17520 Len=1460 [TCP se |
| 9 | 2004-08-21 21:44:20.647675 | 128.119.245.12 | 192.168.1.102 | TCP | 60 | 80 → 1161 [ACK] Seq=1 Ack=2026 Win=8760 Len=0 |
| 10 | 2004-08-21 21:44:20.647786 | 192.168.1.102 | 128.119.245.12 | TCP | 1514 | 1161 → 80 [ACK] Seq=4946 Ack=1 Win=17520 Len=1460 [TCP se |
| 11 | 2004-08-21 21:44:20.648538 | 192.168.1.102 | 128.119.245.12 | TCP | 1514 | 1161 → 80 [ACK] Seq=6406 Ack=1 Win=17520 Len=1460 [TCP se |
| 12 | 2004-08-21 21:44:20.694466 | 128.119.245.12 | 192.168.1.102 | TCP | 60 | 80 → 1161 [ACK] Seq=1 Ack=3486 Win=11680 Len=0 |
| 13 | 2004-08-21 21:44:20.694566 | 192.168.1.102 | 128.119.245.12 | TCP | 1201 | 1161 → 80 [PSH, ACK] Seq=7866 Ack=1 Win=17520 Len=1147 [T |
| 14 | 2004-08-21 21:44:20.739499 | 128.119.245.12 | 192.168.1.102 | TCP | 60 | 80 → 1161 [ACK] Seq=1 Ack=4946 Win=14600 Len=0 |
| 15 | 2004-08-21 21:44:20.787680 | 128.119.245.12 | 192.168.1.102 | TCP | 60 | 80 → 1161 [ACK] Seq=1 Ack=6406 Win=17520 Len=0 |

从服务器发向客户的报文段：



| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 4 | 2004-08-21 21:44:20.596858 | 192.168.1.102 | 128.119.245.12 | TCP | 619 | 1161 → 80 [PSH, ACK] Seq=1 Ack=1 Win=17520 Len=565 [TCP |
| 5 | 2004-08-21 21:44:20.612118 | 192.168.1.102 | 128.119.245.12 | TCP | 1514 | 1161 → 80 [PSH, ACK] Seq=566 Ack=1 Win=17520 Len=1460 [ |
| 6 | 2004-08-21 21:44:20.624318 | 128.119.245.12 | 192.168.1.102 | TCP | 60 | 80 → 1161 [ACK] Seq=1 Ack=566 Win=6780 Len=0 |
| 7 | 2004-08-21 21:44:20.624407 | 192.168.1.102 | 128.119.245.12 | TCP | 1514 | 1161 → 80 [ACK] Seq=2026 Ack=1 Win=17520 Len=1460 [TCP |
| 8 | 2004-08-21 21:44:20.625071 | 192.168.1.102 | 128.119.245.12 | TCP | 1514 | 1161 → 80 [ACK] Seq=3486 Ack=1 Win=17520 Len=1460 [TCP |
| 9 | 2004-08-21 21:44:20.647675 | 128.119.245.12 | 192.168.1.102 | TCP | 60 | 80 → 1161 [ACK] Seq=1 Ack=2026 Win=8760 Len=0 |
| 10 | 2004-08-21 21:44:20.647786 | 192.168.1.102 | 128.119.245.12 | TCP | 1514 | 1161 → 80 [ACK] Seq=4946 Ack=1 Win=17520 Len=1460 [TCP |
| 11 | 2004-08-21 21:44:20.648538 | 192.168.1.102 | 128.119.245.12 | TCP | 1514 | 1161 → 80 [ACK] Seq=6406 Ack=1 Win=17520 Len=1460 [TCP |
| 12 | 2004-08-21 21:44:20.694466 | 128.119.245.12 | 192.168.1.102 | TCP | 60 | 80 → 1161 [ACK] Seq=1 Ack=3486 Win=11680 Len=0 |
| 13 | 2004-08-21 21:44:20.694566 | 192.168.1.102 | 128.119.245.12 | TCP | 1201 | 1161 → 80 [PSH, ACK] Seq=7866 Ack=1 Win=17520 Len=1147 |
| 14 | 2004-08-21 21:44:20.739499 | 128.119.245.12 | 192.168.1.102 | TCP | 60 | 80 → 1161 [ACK] Seq=1 Ack=4946 Win=14600 Len=0 |
| 15 | 2004-08-21 21:44:20.787680 | 128.119.245.12 | 192.168.1.102 | TCP | 60 | 80 → 1161 [ACK] Seq=1 Ack=6406 Win=17520 Len=0 |
| 16 | 2004-08-21 21:44:20.838183 | 128.119.245.12 | 192.168.1.102 | TCP | 60 | 80 → 1161 [ACK] Seq=1 Ack=7866 Win=20440 Len=0 |
| 17 | 2004-08-21 21:44:20.875188 | 128.119.245.12 | 192.168.1.102 | TCP | 60 | 80 → 1161 [ACK] Seq=1 Ack=9013 Win=23360 Len=0 |
| 18 | 2004-08-21 21:44:20.875421 | 192.168.1.102 | 128.119.245.12 | TCP | 1514 | 1161 → 80 [ACK] Seq=9013 Ack=1 Win=17520 Len=1460 [TCP |
| 19 | 2004-08-21 21:44:20.876194 | 192.168.1.102 | 128.119.245.12 | TCP | 1514 | 1161 → 80 [ACK] Seq=10473 Ack=1 Win=17520 Len=1460 [TC |
| 20 | 2004-08-21 21:44:20.877073 | 192.168.1.102 | 128.119.245.12 | TCP | 1514 | 1161 → 80 [ACK] Seq=11933 Ack=1 Win=17520 Len=1460 [TC |




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

   分别为：565、1460、1460、1460、1460、1460

| Protocol | Length | Info |
|---|---|---|
| TCP | 62 | 1161 → 80 [SYN] Seq=0 Win=16384 Len=0 MSS=1460 SACK_PERM=1 |
| TCP | 62 | 80 → 1161 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 SACK_PERM=1 |
| TCP | 54 | 1161 → 80 [ACK] Seq=1 Ack=1 Win=17520 Len=0 |
| TCP | 619 | 1161 → 80 [PSH, ACK] Seq=1 Ack=1 Win=17520 Len=565 [TCP segment of a reassembled P |
| TCP | 1514 | 1161 → 80 [PSH, ACK] Seq=566 Ack=1 Win=17520 Len=1460 [TCP segment of a reassemble |
| TCP | 60 | 80 → 1161 [ACK] Seq=1 Ack=566 Win=6780 Len=0 |
| TCP | 1514 | 1161 → 80 [ACK] Seq=2026 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PD |
| TCP | 1514 | 1161 → 80 [ACK] Seq=3486 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PD |
| TCP | 60 | 80 → 1161 [ACK] Seq=1 Ack=2026 Win=8760 Len=0 |
| TCP | 1514 | 1161 → 80 [ACK] Seq=4946 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PD |
| TCP | 1514 | 1161 → 80 [ACK] Seq=6406 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PD |
| TCP | 60 | 80 → 1161 [ACK] Seq=1 Ack=3486 Win=11680 Len=0 |
| TCP | 1201 | 1161 → 80 [PSH, ACK] Seq=7866 Ack=1 Win=17520 Len=1147 [TCP segment of a reassembl |
| TCP | 60 | 80 → 1161 [ACK] Seq=1 Ack=4946 Win=14600 Len=0 |
| TCP | 60 | 80 → 1161 [ACK] Seq=1 Ack=6406 Win=17520 Len=0 |
| TCP | 60 | 80 → 1161 [ACK] Seq=1 Ack=7866 Win=20440 Len=0 |
| TCP | 60 | 80 → 1161 [ACK] Seq=1 Ack=9013 Win=23360 Len=0 |

bits)

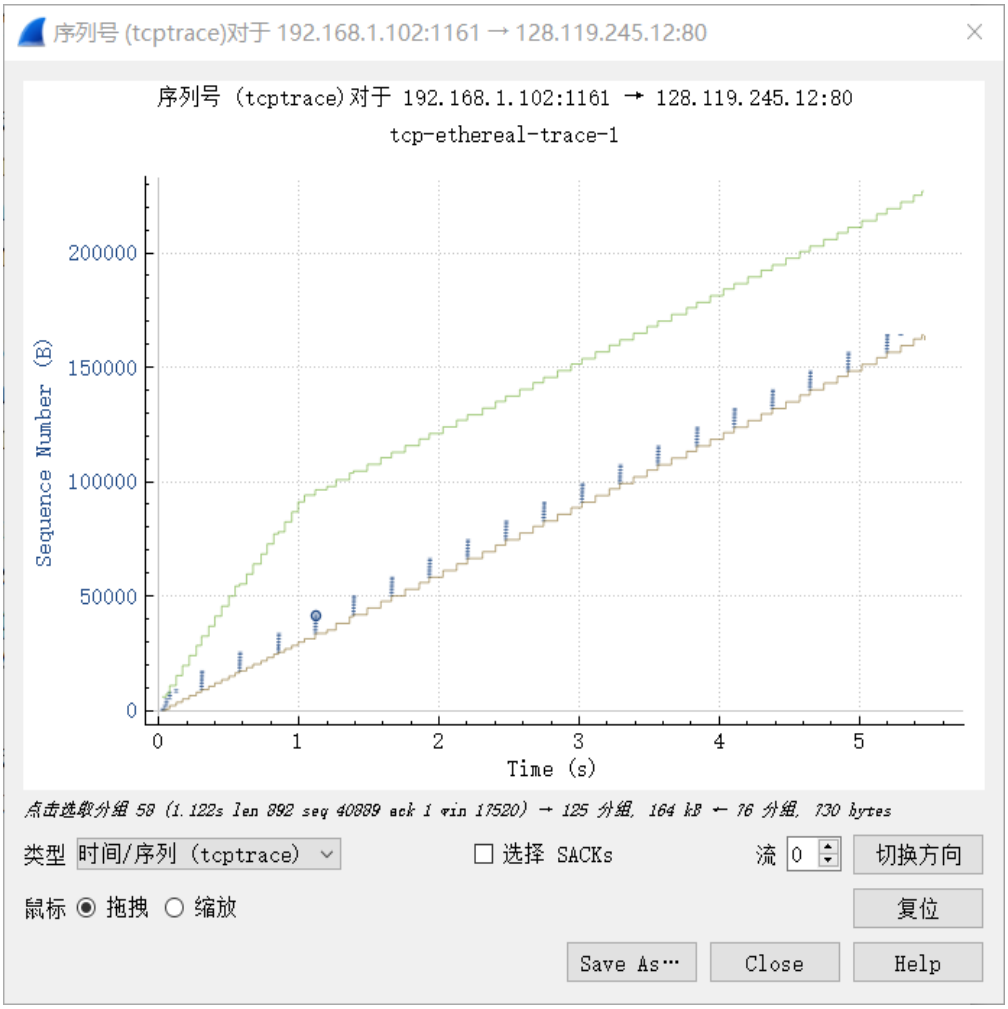
9. **What is the minimum amount of available buffer space advertised at the received for the entire trace? Does the lack of receiver buffer space ever throttle the sender?**

   最小为5840，在整个窗口中并未因缓冲区不足限制发送。

```
th  Info
62 1161 → 80 [SYN] Seq=0 Win=16384 Len=0 MSS=1460 SACK_PERM=1
62 80 → 1161 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 SACK_PER
54 1161 → 80 [ACK] Seq=1 Ack=1 Win=17520 Len=0
19 1161 → 80 [PSH, ACK] Seq=1 Ack=1 Win=17520 Len=565 [TCP segment o
14 1161 → 80 [PSH, ACK] Seq=566 Ack=1 Win=17520 Len=1460 [TCP segmen
60 80 → 1161 [ACK] Seq=1 Ack=566 Win=6780 Len=0
14 1161 → 80 [ACK] Seq=2026 Ack=1 Win=17520 Len=1460 [TCP segment of
14 1161 → 80 [ACK] Seq=3486 Ack=1 Win=17520 Len=1460 [TCP segment of
60 80 → 1161 [ACK] Seq=1 Ack=2026 Win=8760 Len=0
14 1161 → 80 [ACK] Seq=4946 Ack=1 Win=17520 Len=1460 [TCP segment of
14 1161 → 80 [ACK] Seq=6406 Ack=1 Win=17520 Len=1460 [TCP segment of
```

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

    没有，通过不断增加的序列号可知



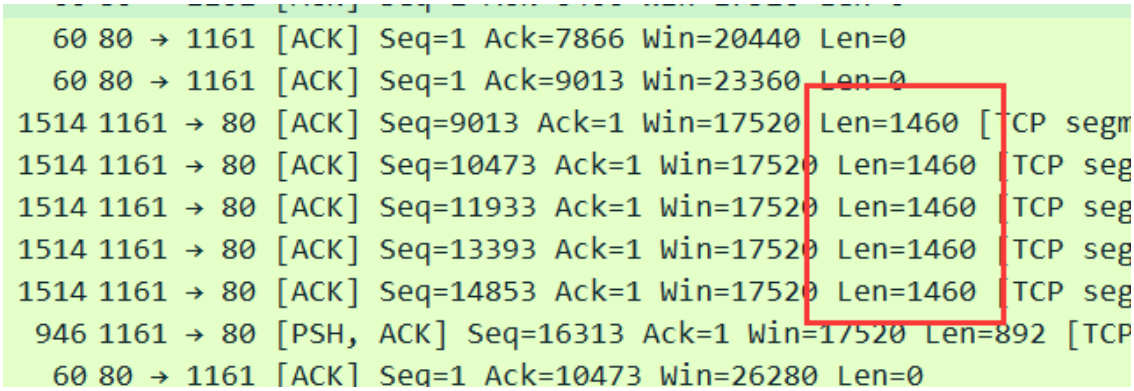

11. **How much data does the receiver typically acknowledge in an ACK? Can you identify cases where the receiver is ACKing every other received segment (see Table 3.2 on page 250 in the text).**

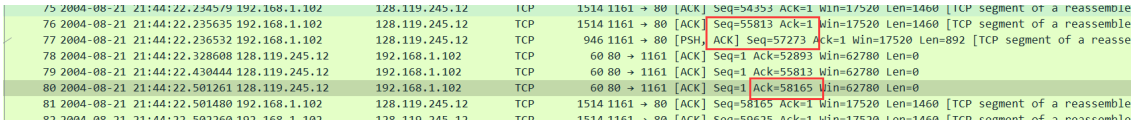
    大多数的ACK长度为1460 byte。

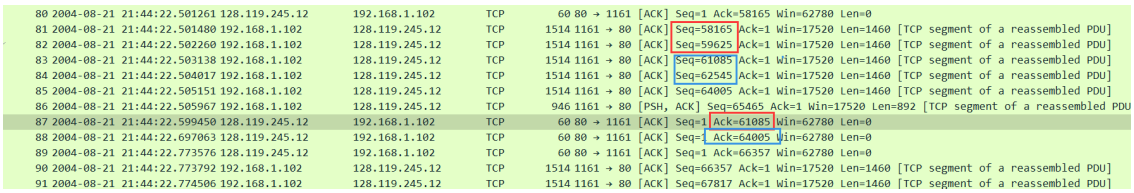
    第80个报文段是对第76、77条报文段的累积确认

第87个报文段是对第81、82条报文段的累积确认

第88个报文段是对第83、84条报文段的累积确认



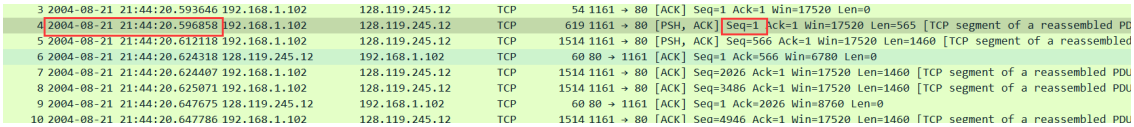



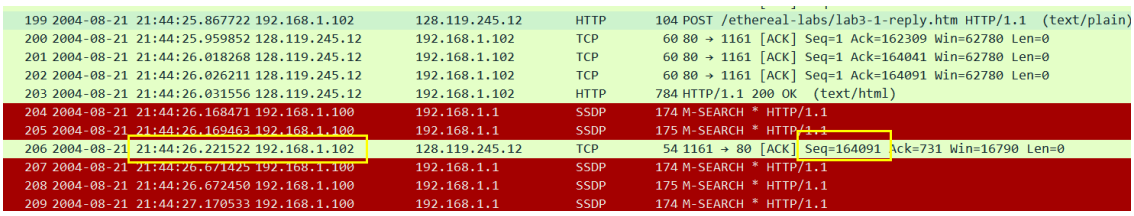




12. **What is the throughput (bytes transferred per unit time) for the TCP connection? Explain how you calculated this value.**

    最后一个序列号减去第一个序列号再除以总时间

    吞吐量 = $\frac{164091-1}{26.221522-20.596858} = 29173.29817\ byte/sec = 29.173\ kbyte/sec$

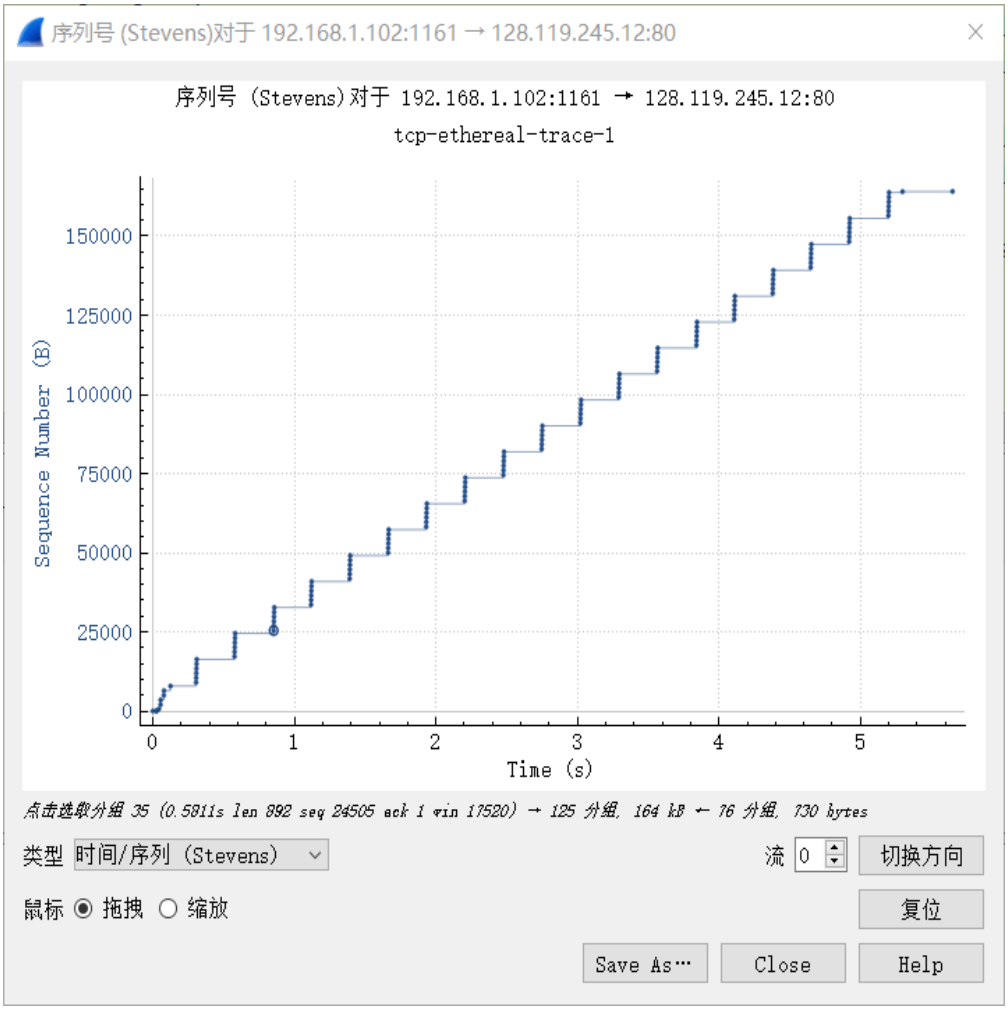



# TCP congestion control in action

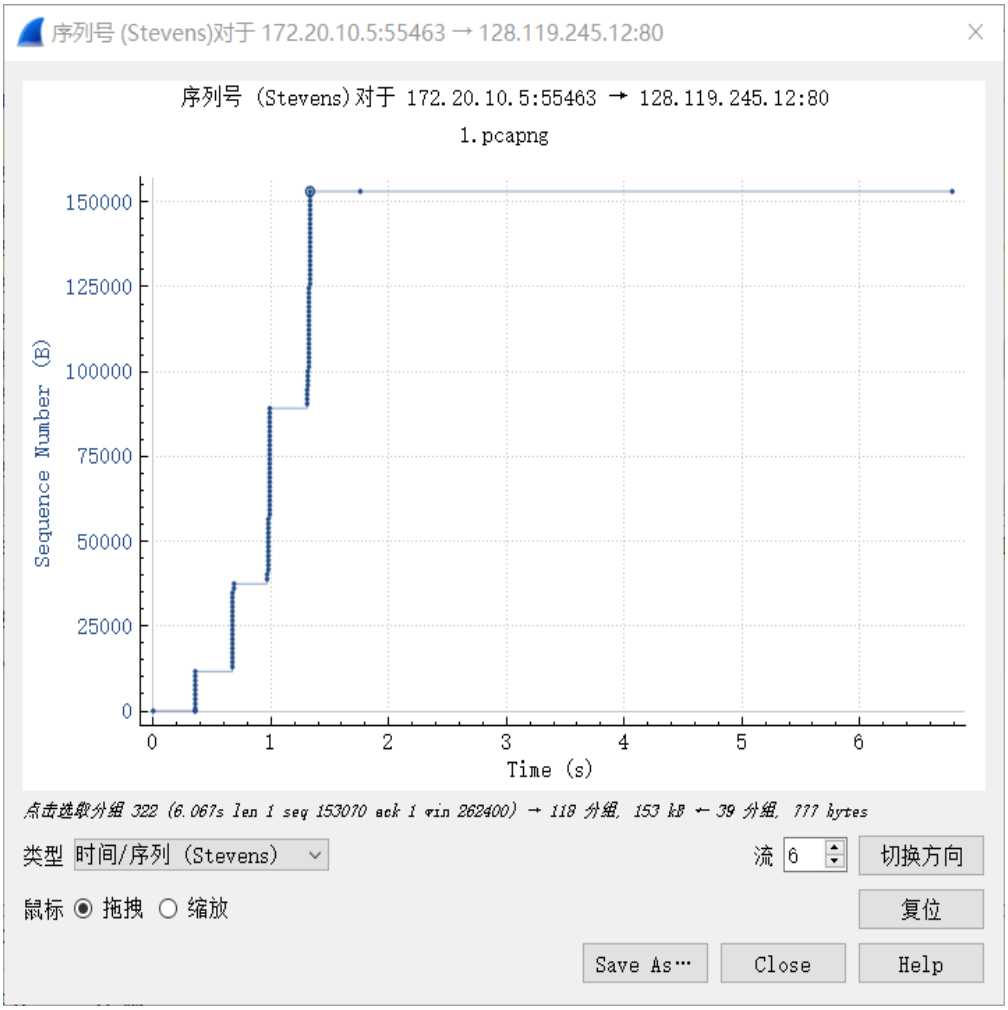
13. **Use the *Time-Sequence-Graph(Stevens)* plotting tool to view the sequence number versus time plot of segments being sent from the client to the gaia.cs.umass.edu server. Can you identify where TCP's slowstart phase begins and ends, and where congestion avoidance takes over? Comment on ways in which the measured data differs from the idealized behavior of TCP that we've studied in the text.**

    在发出了HTTP POST报文段后，慢启动开始，但由于拥塞窗口大小无法直接从时间—序列图直接得到，故无法知道什么时候慢启动结束，什么时候拥塞避免开始。




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**

    如下为自己抓的包的时间—序列图，同样也不能看出什么时候慢启动结束，什么时候拥塞避免开始，因为拥塞窗口大小无法直接从时间—序列图直接得到。


序列号 (Stevens)对于 172.20.10.5:55463 → 128.119.245.12:80
序列号 (Stevens)对于 172.20.10.5:55463 → 128.119.245.12:80
1.pcapng
Sequence Number (B)
150000
125000
100000
75000
50000
25000
0
0
1
2
3
4
5
6
Time (s)
点击选取分组 322 (6.067s len 1 seq 153070 ack 1 win 262400) → 118 分组, 153 kB ← 39 分组, 777 bytes
类型 时间/序列 (Stevens)
流 6
切换方向
鼠标 拖拽 缩放
复位
Save As…
Close
Help