

# CS315: Lab Assignment 5

B Siddharth Prabhu

200010003@iitdh.ac.in

31 January 2023

## 1 Capturing bulk TCP transfer to a remote server

We begin by capturing TCP packets in Wireshark, as per the given instructions. We obtain a trace, as depicted in the below screenshot:

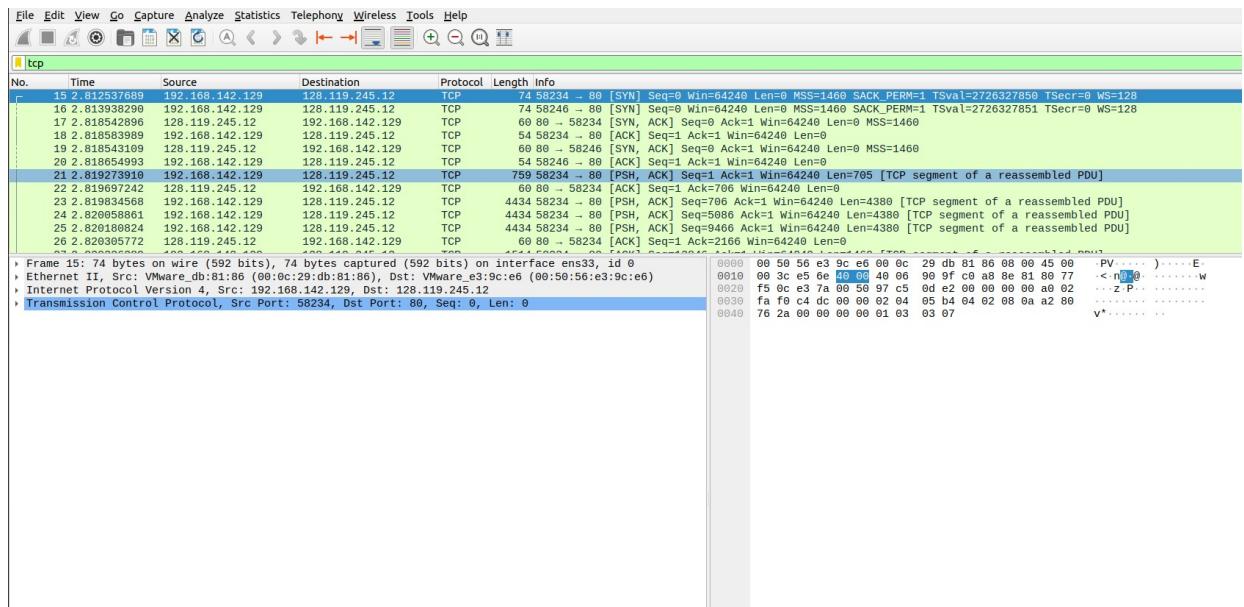


Figure 1: TCP Trace obtained

## 2 A first glance at the TCP trace

Before analyzing the behavior of the TCP connection in detail, let's take a high-level view of the trace. Let's start by looking at the HTTP POST message that uploaded the `alice.txt` file to `gaia.cs.umass.edu`. We shall find that file in the HTTP message so we can take a look at the HTTP POST message more carefully. We make note of the following:

- The body of the application-layer HTTP POST message contains the contents of the file `alice.txt`, which is a large file of more than 148K bytes.
- In fact, as shown in the Wireshark window in Figure (2) we see that the HTTP POST message was spread across 23 TCP segments.

The HTTP POST message that is sent when the file is uploaded, produces the following trace:

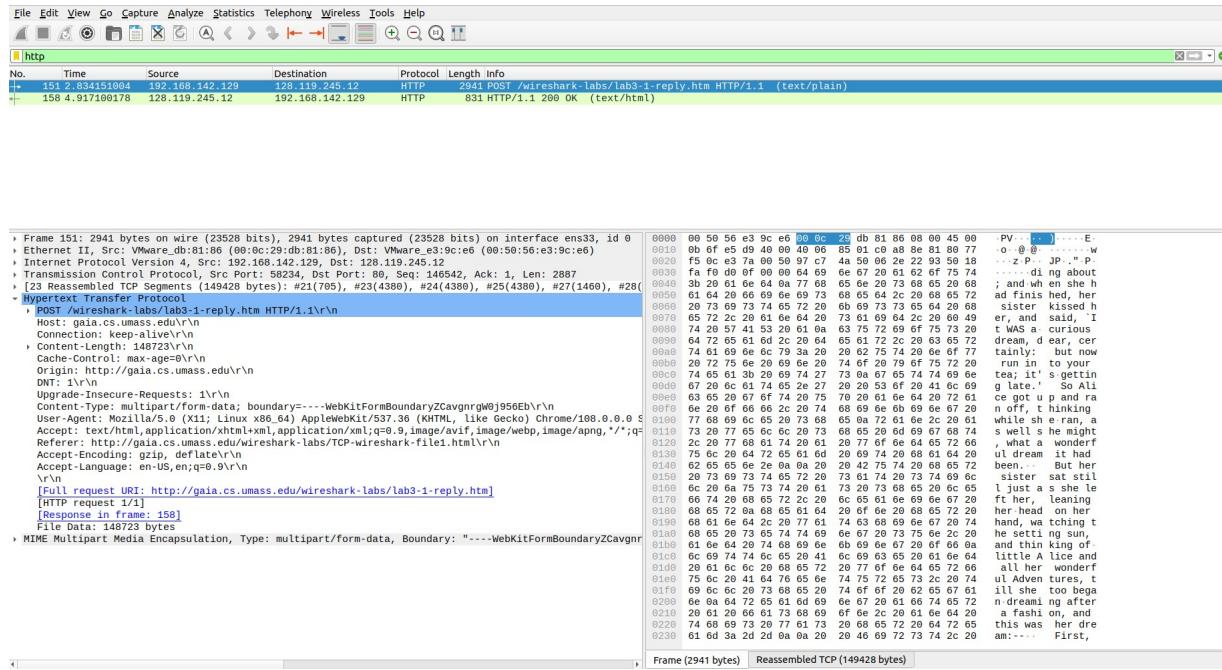


Figure 2: HTTP Trace obtained

Below, we shall answer some questions regarding the same:

## (1) What is the IP address and TCP port number used by the client computer (source) that is transferring the alice.txt file to gaia.cs.umass.edu?

As could be seen in figure (2),

- IP address of source = 192.168.142.129
- TCP Port number of source = 58234

**(Note:** The reason for the IP address above is that I'm running the browser (and Wireshark) on an Ubuntu VM with Network settings set to NAT (Network Address Translation). IP addresses beginning with 192.168 are for private networks, and in this case the private network is formed by the VM and the host system, via a network adapter. Hence, to communicate with the internet, this would take 1 hop more, than if this was run on the host.)

## (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?

As could be seen in figure (2), the IP address of gaia.cs.umass.edu is 128.119.245.12. Also, it sends and receives TCP segments on port 80.

### 3 TCP Basics

Let's explore TCP Segments by answering the following questions:

**(1) 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 this TCP segment that identifies the segment as a SYN segment?**

- For the TCP SYN segment that is used to initiate the TCP connection, the sequence number (raw form) is 2546273762, while its relative sequence number is 0.
- As seen in figure (3), the SYN flag is set to one. This flag identifies the segment as a SYN segment.

The screenshot shows a detailed view of a captured TCP SYN packet. The packet information pane at the top shows: Frame 15: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface ens33, id 0. The packet details pane shows the following fields:  
- Source Port: 58234  
- Destination Port: 80  
- Stream index: 0  
- Conversation completeness: Incomplete, DATA (15)  
- [TCP Segment Len: 0]  
- Sequence Number: 0 (relative sequence number)  
- Sequence Number (raw): 2546273762  
- [Next Sequence Number: 1 (relative sequence number)]  
- Acknowledgment Number: 0  
- Acknowledgment number (raw): 0  
- 1010 .... = Header Length: 40 bytes (10)  
- Flags: 0x002 (SYN)  
- 000. .... .0.. = Reserved: Not set  
- ...0 .... .0.. = Nonce: Not set  
- ....0.... .0.. = Congestion Window Reduced (CWR): Not set  
- ....0.... .0.. = ECN-Echo: Not set  
- ....0.... .0.. = Urgent: Not set  
- ....0.... .0.. = Acknowledgment: Not set  
- ....0.... .0.. = Push: Not set  
- ....0.... .0.. = Reset: Not set  
- ....0.... .1.. = Syn: Set  
- ....0.... .0.. = Fin: Not set  
- [TCP Flags: .....S.]  
- Window: 64240  
- [Calculated window size: 64240]  
- Checksum: 0xc4d4 [unverified]  
- [Checksum Status: Unverified]  
- Urgent Pointer: 0  
- Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window s...  
- [Timestamps]

Figure 3: TCP SYN Packet Details Window

**(2) 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 it in the segment that identifies the segment as a SYNACK segment? What is the value of the Acknowledgement field in the SYNACK segment? How did gaia.cs.umass.edu determine that value?**

- The sequence number of the SYNACK segment is (raw form) 103686802, while its relative sequence number is 0.
- As seen in figure (4), the SYN flag and ACK flag are set to one. These flags identify the segment as a SYNACK segment.
- The value of the acknowledgement field in the SYNACK segment is 2546273763 in raw form, and 1 in relative form.
- gaia.cs.umass.edu calculated this value by adding one to the sequence number of the SYN message.

```

> Frame 17: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface ens33, id 0
> Ethernet II, Src: VMware_e3:9c:e6 (00:50:56:e3:9c:e6), Dst: VMware_db:81:86 (00:0c:29:db:81:86)
> Internet Protocol Version 4, Src: 128.119.245.12, Dst: 192.168.142.129
> Transmission Control Protocol, Src Port: 80, Dst Port: 58234, Seq: 0, Ack: 1, Len: 0
    Source Port: 80
    Destination Port: 58234
    [Stream index: 0]
    [Conversation completeness: Incomplete, DATA (15)]
    [TCP Segment Len: 0]
    Sequence Number: 0 (relative sequence number)
    Sequence Number (raw): 103686802
    [Next Sequence Number: 1 (relative sequence number)]
    Acknowledgment Number: 1 (relative ack number)
    Acknowledgment number (raw): 2546273763
    0110 .... = Header Length: 24 bytes (6)
    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: 64240
    [Calculated window size: 64240]
    Checksum: 0x2644 [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
    Options: (4 bytes), Maximum segment size
    [Timestamps]
    [SEQ/ACK analysis]

```

Figure 4: TCP SYNACK Packet Details Window

**(3) What is the sequence number of the TCP segment containing the header of the HTTP POST command? How many bytes of data are contained in the payload (data) field of this TCP segment? Did all of the data in the transferred file alice.txt fit into this single segment?**

- The sequence number of the TCP segment containing the HTTP POST command is 2546273763 in raw form, and 1 in relative form.
- As seen in figure (5), this TCP segment has 705 bytes of data in its payload field.
- No, not all of the data in the transferred file fit in this segment. This can be said by examining the 22 other segments that have been sent. We can also observe this breakup of file material in figure (6). In fact, the last part of the header seems to reach the second segment.

```

> Frame 21: 759 bytes on wire (6072 bits), 759 bytes captured (6072 bits) on interface ens33, id 0
> Ethernet II, Src: VMware_db:81:86 (00:0c:29:db:81:86), Dst: VMware_e3:9c:e6 (00:50:56:e3:9c:e6)
> Internet Protocol Version 4, Src: 192.168.142.129, Dst: 128.119.245.12
> Transmission Control Protocol, Src Port: 58234, Dst Port: 80, Seq: 1, Ack: 1, Len: 705
    Source Port: 58234
    Destination Port: 80
    [Stream index: 0]
    [Conversation completeness: Incomplete, DATA (15)]
    [TCP Segment Len: 705]
    Sequence Number: 1 (relative sequence number)
    Sequence Number (raw): 2546273763
    [Next Sequence Number: 706 (relative sequence number)]
    Acknowledgment Number: 1 (relative ack number)
    Acknowledgment number (raw): 103686803
    0101 .... = Header Length: 20 bytes (5)
    Flags: 0x018 (PSH, ACK)
    Window: 64240
    [Calculated window size: 64240]
    [Window size scaling factor: -2 (no window scaling used)]
    Checksum: 0xc789 [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
    Options: (4 bytes), Maximum segment size
    [Timestamps]
    [SEQ/ACK analysis]
    TCP payload (705 bytes)
    [Reassembled PDU in frame: 151]
    TCP segment data (705 bytes)

```

Figure 5: TCP segment containing HTTP POST header

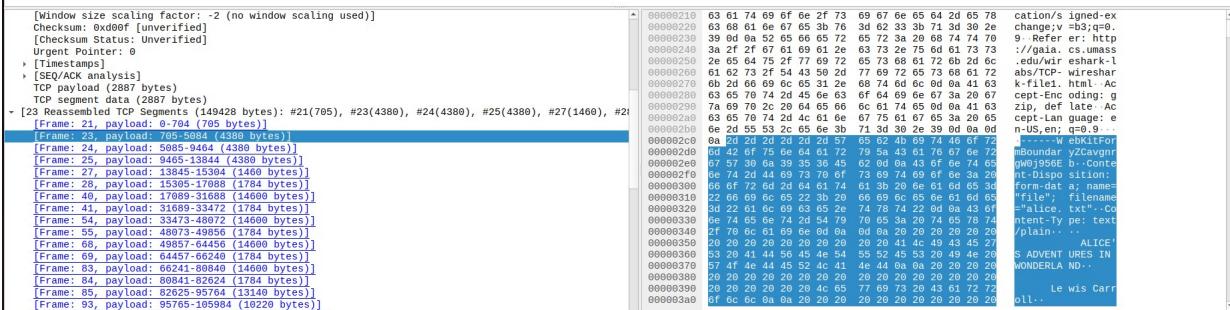


Figure 6: File fragmentation across segments

#### (4) Consider the TCP segment containing the HTTP “POST” as the first segment in the data transfer part of the TCP connection.

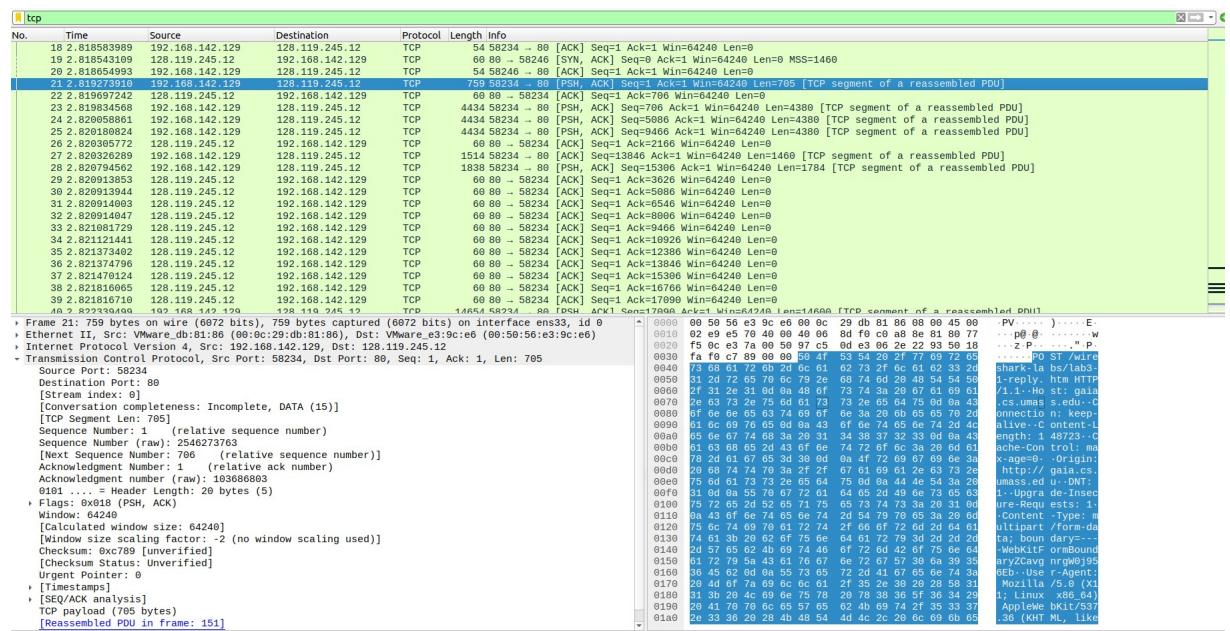


Figure 7: TCP segments containing HTTP POST

##### (i) At what time was the first segment (the one containing the HTTP POST) in the data-transfer part of the TCP connection sent?

The first segment in the data-transfer part of the TCP connection was sent at time 2.819273910.

##### (ii) At what time was the ACK for this first data-containing segment received?

The ACK for this first data-containing segment was received at time 2.819697242. This can be verified by looking at the acknowledgement number in Figure (8) and the next sequence number in Figure (7).

##### (iii) What is the RTT for this first data-containing segment?

The RTT for this first data-containing segment is 0.000423332, and is obtained by calculating the difference between the above two time quantities.

(iv) What is the RTT value of the second data-containing segment and its ACK?

The RTT for this second data-containing segment is 0.001079376, and is obtained by analyzing the ACK segment to the second segment. This is observed in Figure (10).

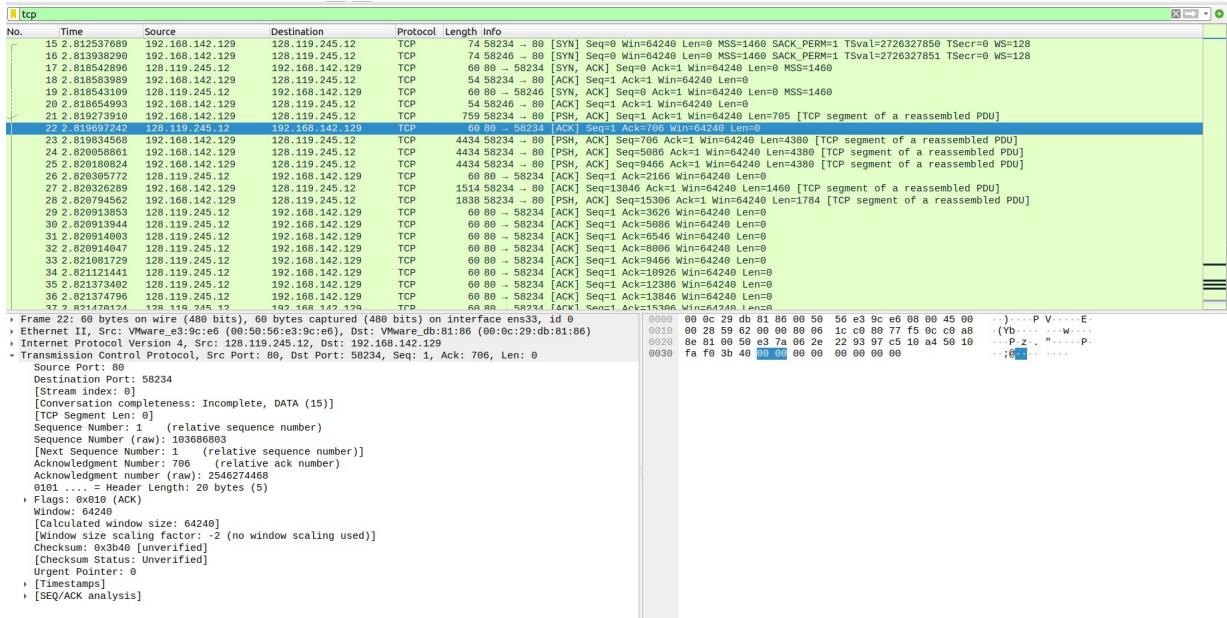


Figure 8: TCP ACK

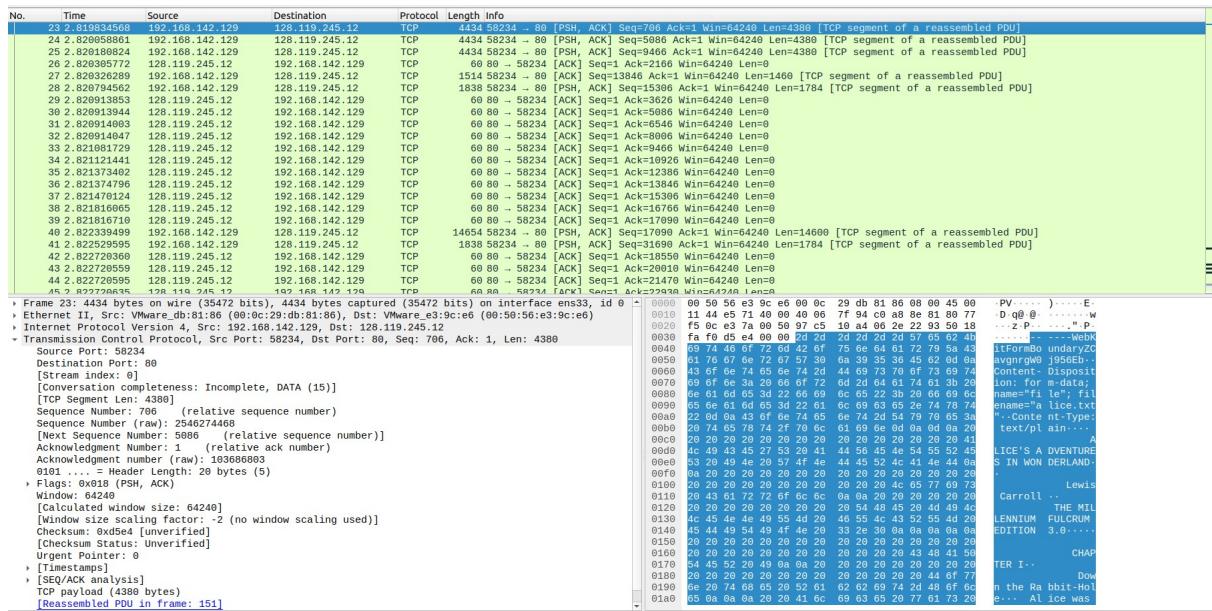


Figure 9: TCP second data-containing segment

```

> Frame 30: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface ens33, id 0
> Ethernet II, Src: VMware_e3:9c:e6 (00:50:56:e3:9c:e6), Dst: VMware_db:81:86 (00:0c:29:db:81:86)
> Internet Protocol Version 4, Src: 128.119.245.12, Dst: 192.168.142.129
> Transmission Control Protocol, Src Port: 80, Dst Port: 58234, Seq: 1, Ack: 5086, Len: 0
    Source Port: 80
    Destination Port: 58234
    [Stream index: 0]
    [Conversation completeness: Incomplete, DATA (15)]
    [TCP Segment Len: 0]
    Sequence Number: 1 (relative sequence number)
    Sequence Number (raw): 103686803
    [Next Sequence Number: 1 (relative sequence number)]
    Acknowledgment Number: 5086 (relative ack number)
    Acknowledgment number (raw): 2546278848
    0101 .... = Header Length: 20 bytes (5)
    Flags: 0x010 (ACK)
    Window: 64240
    [Calculated window size: 64240]
    [Window size scaling factor: -2 (no window scaling used)]
    Checksum: 0xa24 [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
    > [Timestamps]
    > [SEQ/ACK analysis]
        [This is an ACK to the segment in frame: 23]
        [The RTT to ACK the segment was: 0.001079376 seconds]

```

Figure 10: TCP ACK to second data-carrying segment

**(5) What is the length (header plus payload) of each of the first four data-carrying TCP segments?**

As visible in Figure (6), the length of the first four data-carrying TCP segments are **705, 4380, 4380, 4380** respectively.

**(6) What is the minimum amount of available buffer space advertised to the client by gaia.cs.umass.edu among these first four data-carrying TCP segments? Does the lack of receiver buffer space ever throttle the sender for these first four data-carrying segments?**

The minimum amount of available buffer space advertised to the client is **64240**. This receiver window grows until it reaches the maximum receiver buffer size of 62780 bytes. According to the trace, the sender is never throttled due to lacking of receiver buffer space.

```

> Flags: 0x010 (ACK)
Window: 64240
[Calculated window size: 64240]
[Window size scaling factor: -2 (no window scaling used)]
Checksum: 0x3b40 [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0

```

Figure 11: Window size advertised

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

No, there are no retransmitted segments in the trace file. To answer this question, I typed `tcp.analysis.retransmission || tcp.analysis.fast_retransmission` into the filter entry field. This is visible in Figure (12).

tcp.analysis.retransmission    tcp.analysis.fast_retransmission						
No.	Time	Source	Destination	Protocol	Length	Info

Figure 12: No retransmissions

**(8) How much data does the receiver typically acknowledge in an ACK among the first ten data-carrying segments sent from the client to gaia.cs.umass.edu? Can you identify cases where the receiver is ACKing every other received segment among these first ten data-carrying segments?**

Based on the screenshot below in Figure, the ACK values are: 706, 2166, 3626, 5086, 6546, 8006, 9466, 10926, 12386, 13846, 15306. So, the bits of data acknowledged are: 1460, 1460, 1500, 1460, 1460, 1460, 1460, 1460, 1460. Hence, it is mostly acknowledging 1460 bits at a time.

There aren't cases where receiver ACKs every other received segment among these first ten data-carrying segments.

22 2.819697242	128.119.245.12	192.168.142.129	TCP	60 80 → 58234 [ACK] Seq=1 Ack=706 Win=64240 Len=0		
23 2.819834568	192.168.142.129	128.119.245.12	TCP	4434 58234 → 80 [PSH, ACK] Seq=706 Ack=1 Win=64240 Len=4380 [TCP segment of a reassembled PDU]		
24 2.820958861	192.168.142.129	128.119.245.12	TCP	4434 58234 → 80 [PSH, ACK] Seq=5086 Ack=1 Win=64240 Len=4380 [TCP segment of a reassembled PDU]		
25 2.8209180824	192.168.142.129	128.119.245.12	TCP	4434 58234 → 80 [PSH, ACK] Seq=9466 Ack=1 Win=64240 Len=4380 [TCP segment of a reassembled PDU]		
26 2.8209365772	128.119.245.12	192.168.142.129	TCP	60 80 → 58234 [ACK] Seq=2 Ack=2166 Win=64240 Len=0		
27 2.8209326289	192.168.142.129	128.119.245.12	TCP	1514 58234 → 80 [ACK] Seq=3 Ack=1460 Win=64240 Len=1460 [TCP segment of a reassembled PDU]		
28 2.8209794562	192.168.142.129	128.119.245.12	TCP	1838 58234 → 80 [PSH, ACK] Seq=15306 Ack=1 Win=64240 Len=1784 [TCP segment of a reassembled PDU]		
29 2.8209138582	128.119.245.12	192.168.142.129	TCP	60 80 → 58234 [ACK] Seq=2 Ack=3626 Win=64240 Len=0		
30 2.8209139442	128.119.245.12	192.168.142.129	TCP	60 80 → 58234 [ACK] Seq=1 Ack=5086 Win=64240 Len=0		
31 2.8209140983	128.119.245.12	192.168.142.129	TCP	60 80 → 58234 [ACK] Seq=1 Ack=6546 Win=64240 Len=0		
32 2.8209140947	128.119.245.12	192.168.142.129	TCP	60 80 → 58234 [ACK] Seq=2 Ack=8006 Win=64240 Len=0		
33 2.821081729	128.119.245.12	192.168.142.129	TCP	60 80 → 58234 [ACK] Seq=3 Ack=9466 Win=64240 Len=0		
34 2.8212121441	128.119.245.12	192.168.142.129	TCP	60 80 → 58234 [ACK] Seq=1 Ack=10926 Win=64240 Len=0		
35 2.8213734092	128.119.245.12	192.168.142.129	TCP	60 80 → 58234 [ACK] Seq=1 Ack=12386 Win=64240 Len=0		
36 2.821374796	128.119.245.12	192.168.142.129	TCP	60 80 → 58234 [ACK] Seq=1 Ack=13846 Win=64240 Len=0		
37 2.821470124	128.119.245.12	192.168.142.129	TCP	60 80 → 58234 [ACK] Seq=1 Ack=15306 Win=64240 Len=0		
38 2.821816665	128.119.245.12	192.168.142.129	TCP	60 80 → 58234 [ACK] Seq=1 Ack=16766 Win=64240 Len=0		
39 2.821816716	128.119.245.12	192.168.142.129	TCP	60 80 → 58234 [ACK] Seq=1 Ack=17090 Win=64240 Len=0		

Figure 13: ACK bits

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

For throughput, we make note of the following:

- Time of first SYN ( $t_1$ ) = 2.819273910
- Time of last ACK ( $t_2$ ) = 2.834361649
- Bytes in the file transferred ( $F$ ) = 149428
- Total time taken ( $t_2 - t_1$ ) = 0.015087739

Hence, we calculate throughput as  $\frac{F}{t_2 - t_1} = \frac{149428}{0.015087739} = 9903935.90451 \text{ bytes/second}$

No.	Time	Source	Destination	Protocol	Length	Info
141 2.828588216	128.119.245.12	192.168.142.129	TCP	68 80 - 58234	[ACK]	Seq=1 Ack=138106 Win=5516 Len=0
142 2.828588247	128.119.245.12	192.168.142.129	TCP	68 80 - 58234	[ACK]	Seq=1 Ack=139566 Win=4056 Len=0
143 2.829621829	128.119.245.12	192.168.142.129	TCP	68 80 - 58234	[ACK]	Seq=1 Ack=140926 Win=2596 Len=0
144 2.830621939	128.119.245.12	192.168.142.129	TCP	68 80 - 58234	[ACK]	Seq=1 Ack=142326 Win=1236 Len=0
145 2.830621938	128.119.245.12	192.168.142.129	TCP	68 [TCP ZeroWindow] 80 - 58234	[ACK]	Seq=1 Ack=143622 Win=8 Len=0
146 2.831098309	128.119.245.12	192.168.142.129	TCP	2974 [TCP Window Update]	80 - 58234	[ACK] Seq=1 Ack=143622 Win=2928 Len=8
147 2.831328080	128.119.245.12	192.168.142.129	TCP	68 [TCP Window Full] 58234 - 80	[PSH ACK]	Seq=143622 Ack=1 Win=64240 Len=2928 [TCP segment of a reassembled PDU]
148 2.831328529	128.119.245.12	192.168.142.129	TCP	68 80 - 58234	[ACK]	Seq=1 Ack=145082 Win=146 Len=0
149 2.831328680	128.119.245.12	192.168.142.129	TCP	68 [TCP ZeroWindow] 80 - 58234	[ACK]	Seq=1 Ack=146542 Win=8 Len=0
150 2.834151894	128.119.245.12	192.168.142.129	HTTP	294 POST /wireshark-1.8.0/label-reply.html	1	(text/plain)
152 2.834361595	128.119.245.12	192.168.142.129	TCP	68 80 - 58234	[ACK]	Seq=1 Ack=148902 Win=30984 Len=0
153 2.834361649	128.119.245.12	192.168.142.129	TCP	68 80 - 58234	[ACK]	Seq=1 Ack=149429 Win=7057 Len=0
154 2.841481867	128.119.245.12	192.168.142.129	TCP	68 [TCP Window Update]	80 - 58234	[ACK] Seq=1 Ack=149429 Win=64240 Len=0
Frame 153: 68 bytes on wire (480 bits), 68 bytes captured (480 bits) on interface ens33, id 0						
Ethernet II, Src: VMware_e3:9c:e6 (00:50:56:e3:9c:e6), Dst: VMware_db:81:86 (00:0c:29:db:81:86)						
Internet Protocol Version 4, Src: 128.119.245.12, Dst: 192.168.142.129						
Transmission Control Protocol, Src Port: 80, Dst Port: 58234, Seq: 1, Ack: 149429, Len: 0						
Sequence Number: 0						
Destination Port: 58234						
[Stream index: 8]						
[Conversation completeness: Incomplete, DATA (15)]						
[TCP Segment Len: 0]						
Sequence Number: 1 (relative sequence number)						
Sequence Number (raw): 103686893						
Acknowledgment Number: 149428 (relative ack number)						
Acknowledgment Number (raw): 2546423191						
0@1 ... = Header Length: 20 bytes (5)						
Flags: 0x010 (ACK)						
Window: 29557						
Calculated window size: 29557						
Window size scaling factor: -2 (no window scaling used)						
Checksum: 0x7de6 [unverified]						
Checksum Status: Unverified						
Urgent Pointer: 0						
[Timestamps]						
[SEQ/ACK analysis]						
[This is an ACK to the segment in frame: 151]						
[The RTO to ACK the segment was: 0.000218645 seconds]						
[RTT: 0.000046300 seconds]						
0000 00 00 29 db 81 86 00 50 56 e3 9c e6 00 00 45 00 ..) ..P V ..E						
0010 00 28 59 cf 00 00 00 1c 53 80 77 f5 0c c9 a8 ..(Y ..S ..W ..						
0020 8e 81 00 50 e3 7a 06 2e 22 93 97 c7 55 97 50 10 ..P z ..U P ..						
0030 73 75 7d c6 00 00 00 00 00 00 00 00 su) .. .. .. ..						

Figure 14: Final ACK for the file transfer

## 4 TCP Congestion Control

### 4.1 (1) Can you identify where TCP's slow start phase begins and ends, and where congestion avoidance takes over?

TCP's slow start phase starts and ends from 0.006 to 0.0075 seconds, as seen in the graph. The congestion avoidance takes over at 0.01515, as visible in the graph.

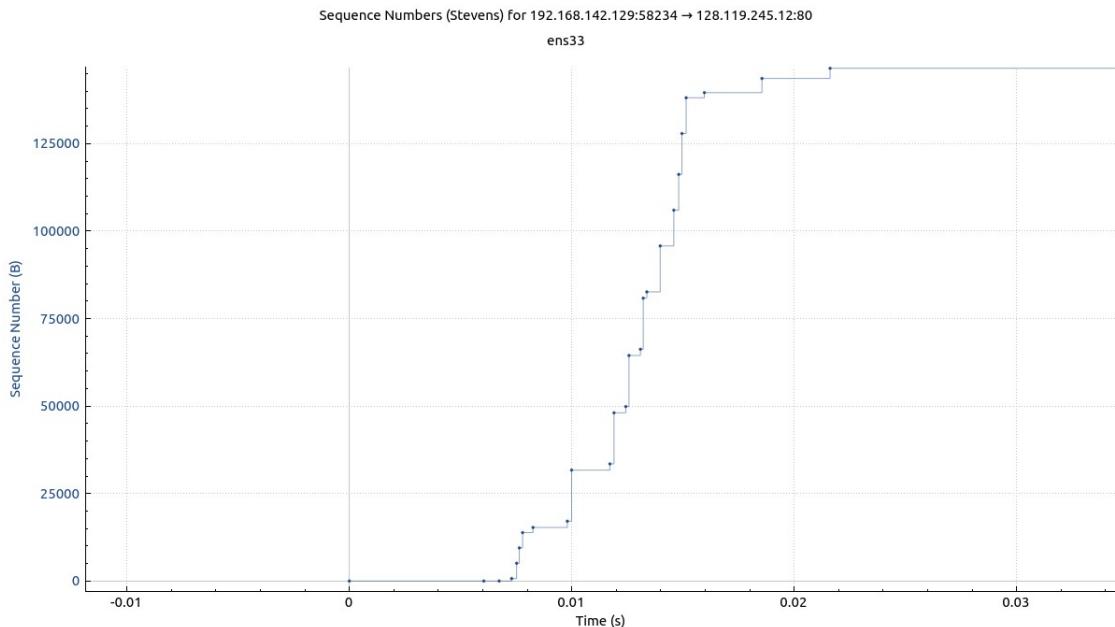


Figure 15: Plot for TCP transfer