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:

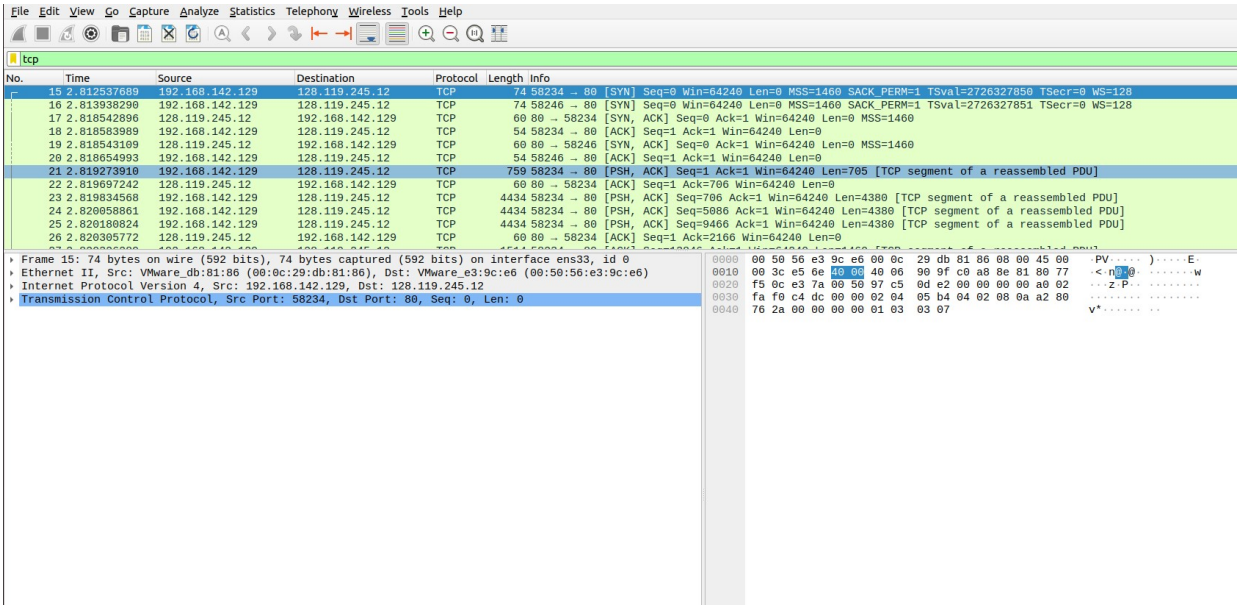

Figure 1: TCP Trace obtained

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:

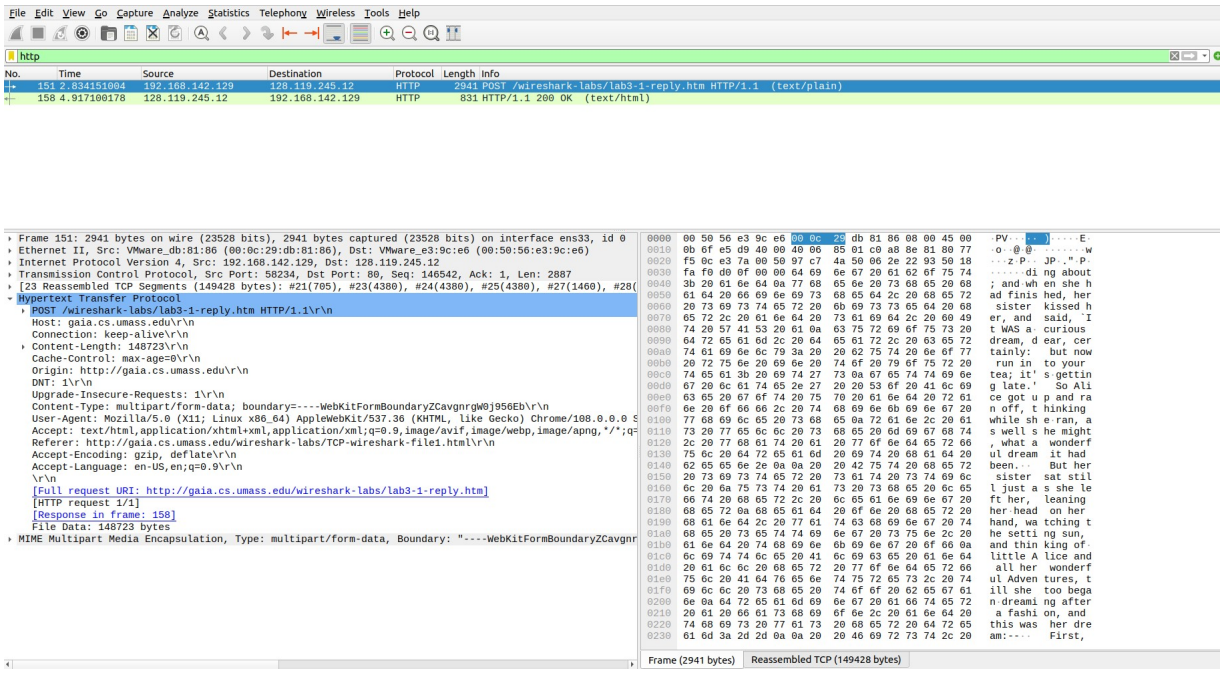

Figure 2: HTTP Trace obtained

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.

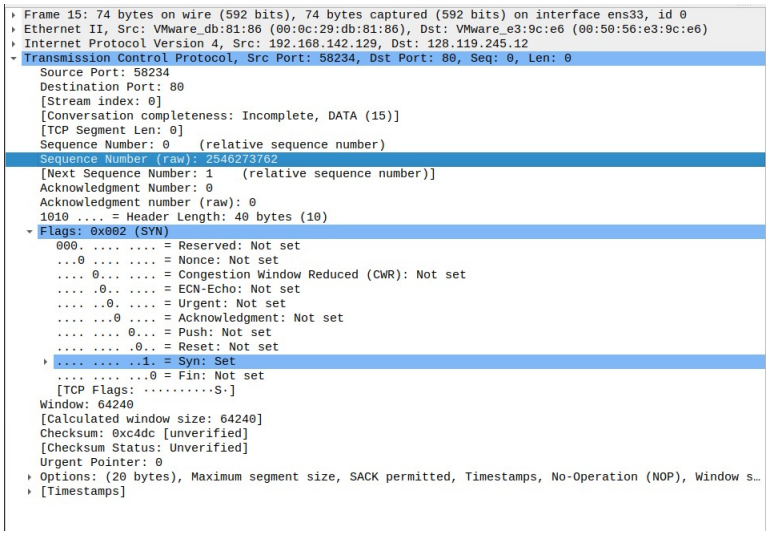

Figure 3: TCP SYN Packet Details Window

(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.

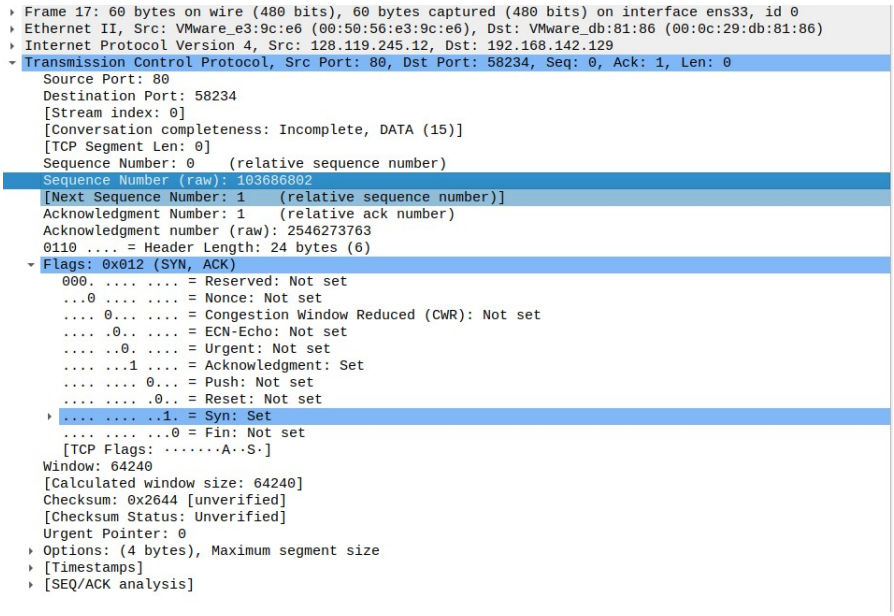

Figure 4: TCP SYNACK Packet Details Window

(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.

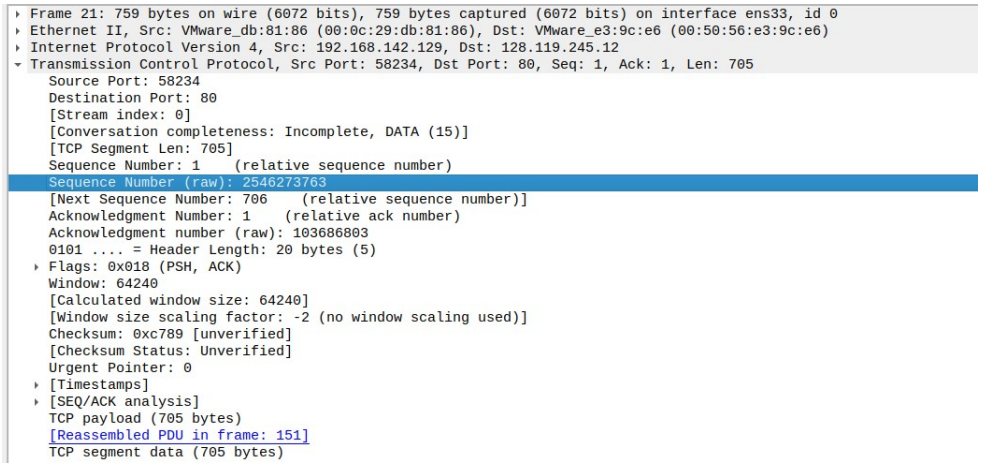

Figure 5: TCP segment containing HTTP POST header

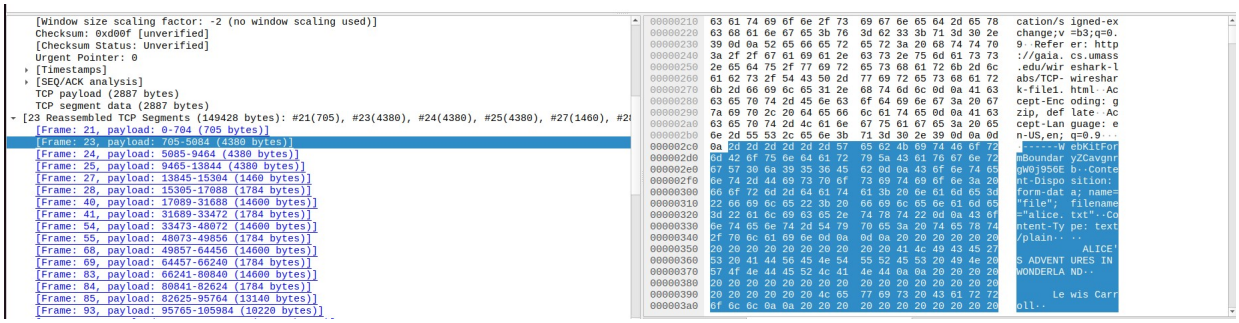

Figure 6: File fragmentation across segments

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

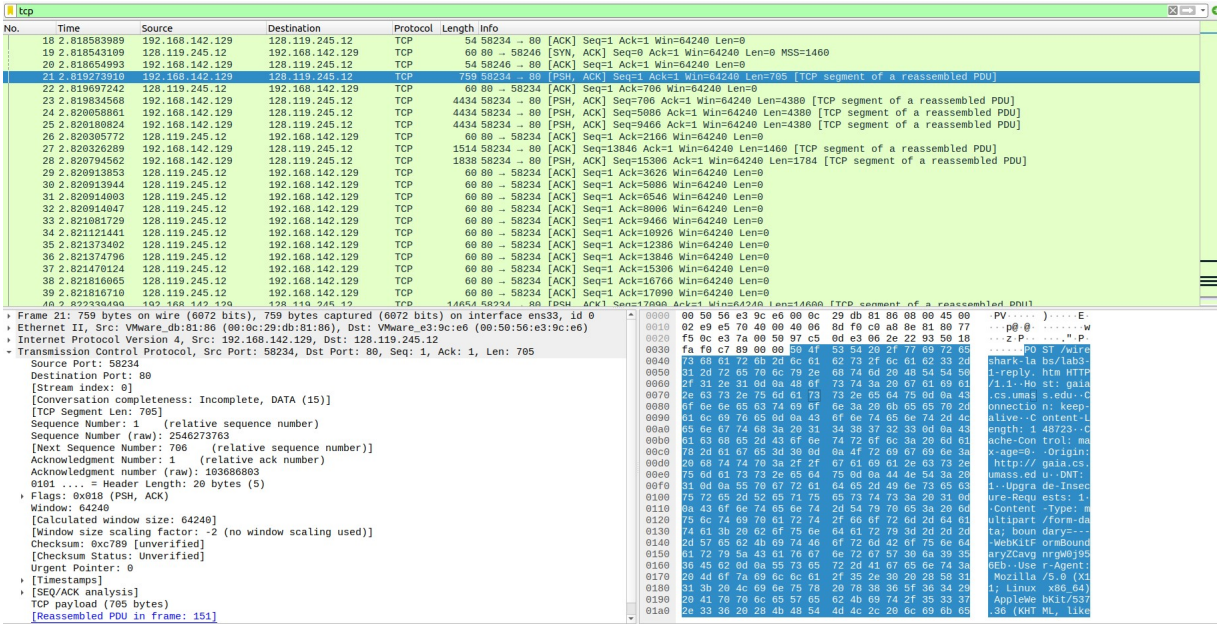

Figure 7: TCP segments containing HTTP POST

(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-carrying TCP 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).

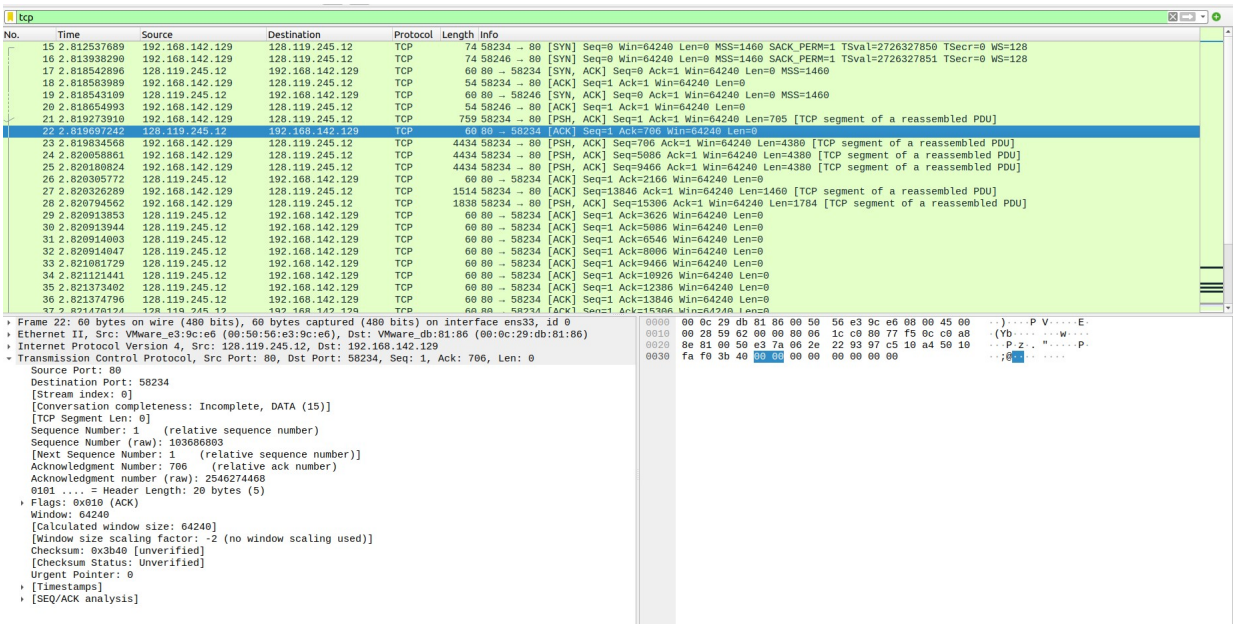

Figure 8: TCP ACK

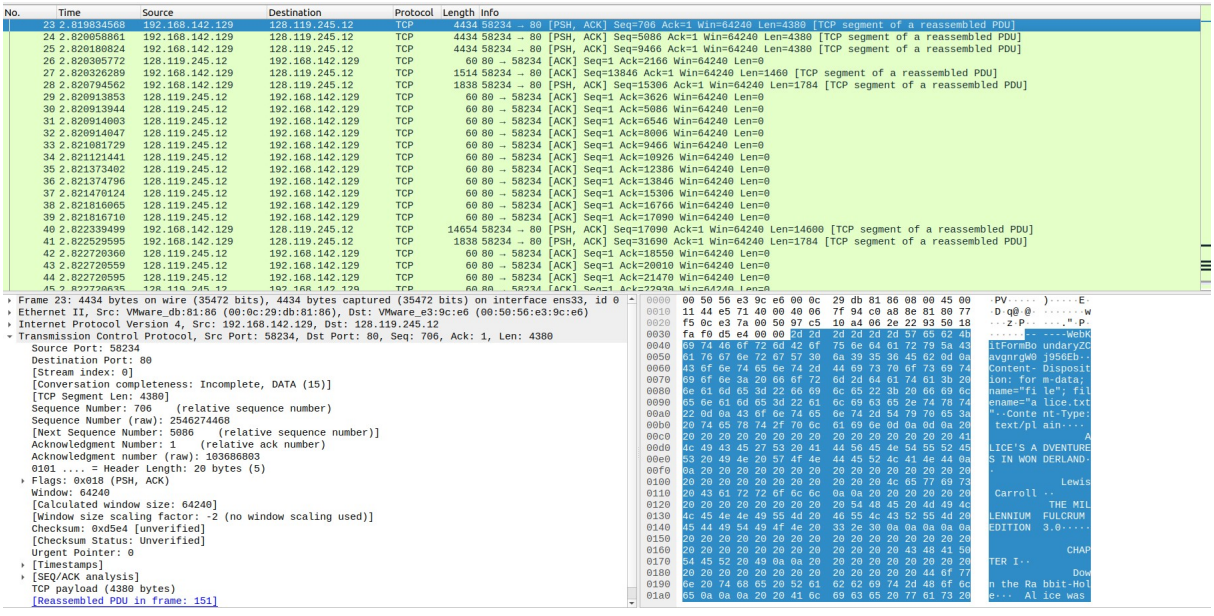

Figure 9: TCP second data-carrying segment

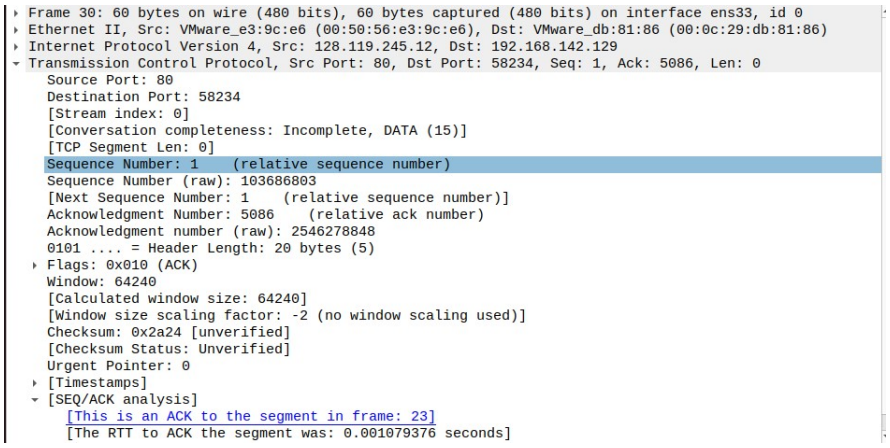

Figure 10: TCP ACK to second data-carrying segment

(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.

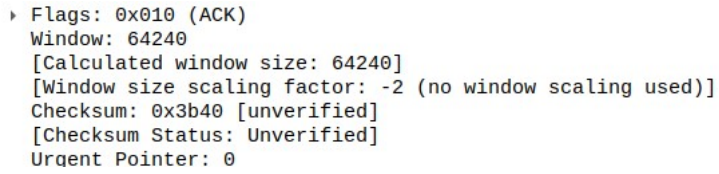

Figure 11: Window size advertised

(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).

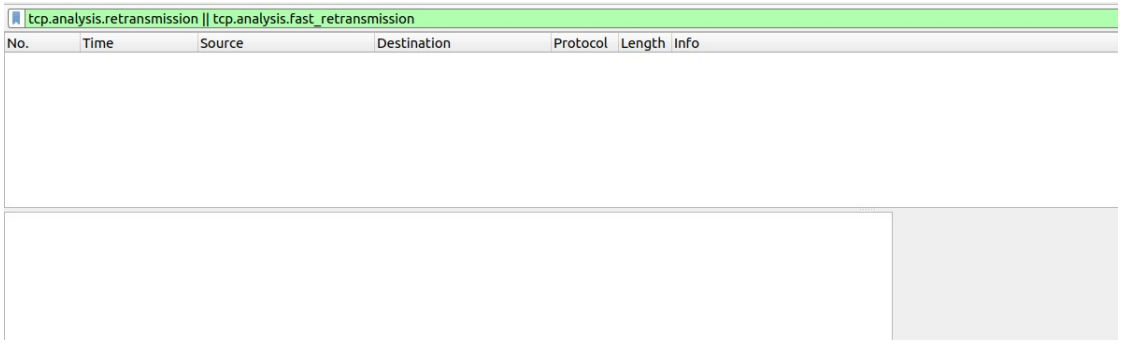

Figure 12: No retransmissions

(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, 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.

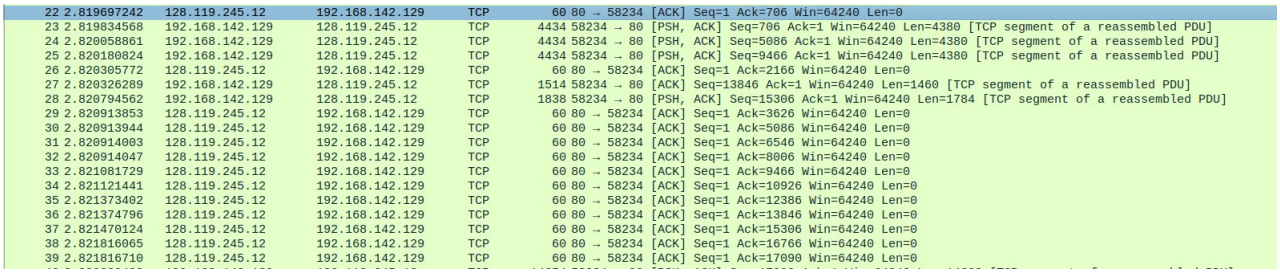

Figure 13: ACK bits

(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 (t1) = 2.819273910
- Time of last ACK (t2) = 2.834361649
- Bytes in the file transferred (F) = 149428
- Total time taken (t2 - t1) = 0.015087739

Hence, we calculate throughput as $\frac{F}{t2-t1} = \frac{149428}{0.015087739} = 9903935.90451 bytes/second$

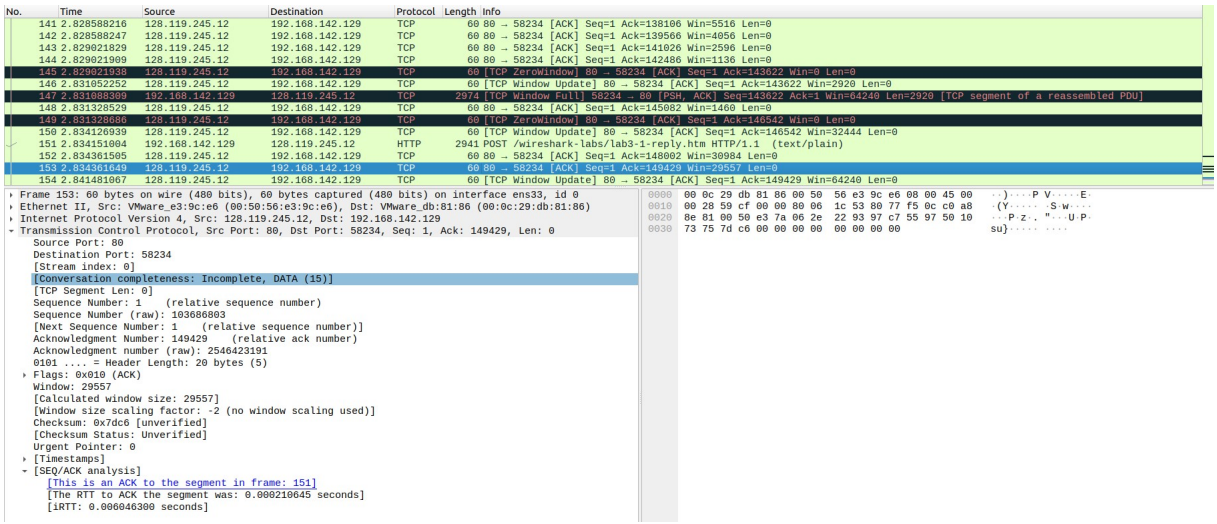

Figure 14: Final ACK for the file transfer

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.

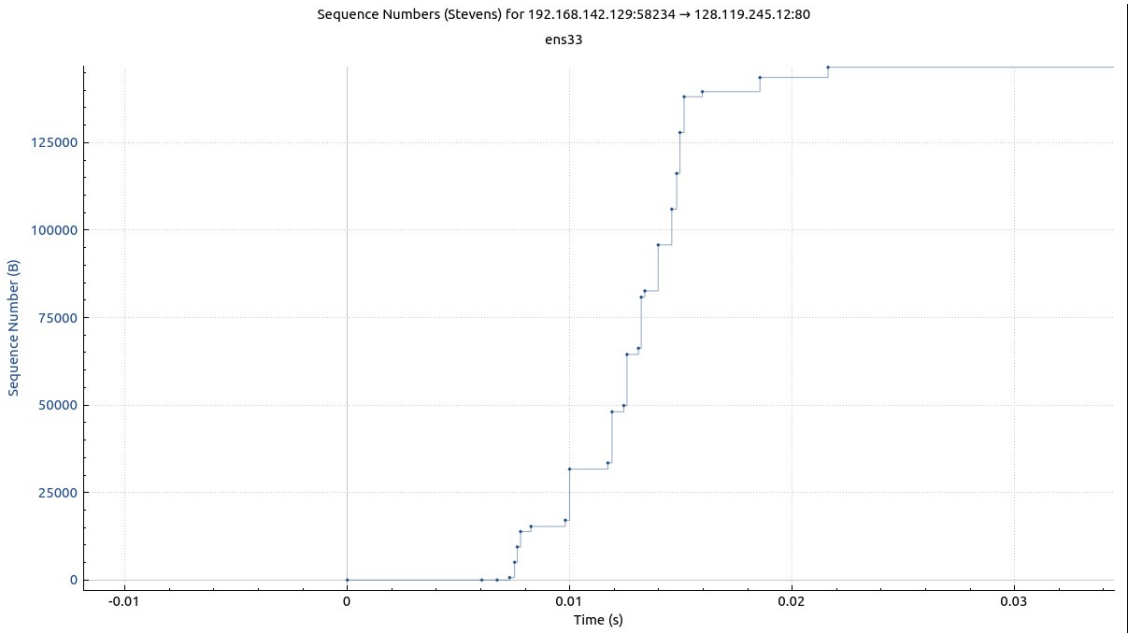

Figure 15: Plot for TCP transfer