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

1. Download and save **alice.txt** on your computer [\_](http://gaia.cs.umass.edu/wireshark-labs/alice.txt)
2. Go to: 128.119.245.12/wireshark-labs/TCP-wireshark-file1.html [\_](http://gaia.cs.umass.edu/wireshark-labs/TCP-wireshark-file1.html)
3. “Browse” the alice.txt file
4. Start WireShark capture
5. “Upload” alice.txt file
6. Stop WireShark capture
7. Filter the “tcp” packets [\_](http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.zip)

**Questions**:

1. What is the IP address and TCP port number used by the client computer (source) that is transferring/uploading the file?

Solution: As Figure 1 shown, the IP address of source is 192.168.1.154, the source port number is 52267

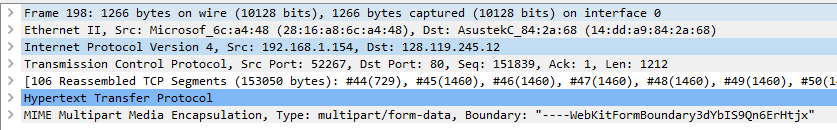


Figure 1.

1. What is the remote server IP address? On what port number is it sending and receiving TCP segments for this connection?

Solution: As Figure 1 shown, the IP address of destination is 128.119.245.12, the destination port number is 80

[Steps 3 to 13 are optional for non-tech people\*]

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 the segment that identifies the segment as a SYN segment?

Solution: As Figure 2 shown, the sequence number is 0; sequence number of TCP SYN segment is used to initiate the TCP connection between the client computer and gaia.cs.umass.edu. The SYN flag is set to 1 and that indicates this segment is a SYN segment.

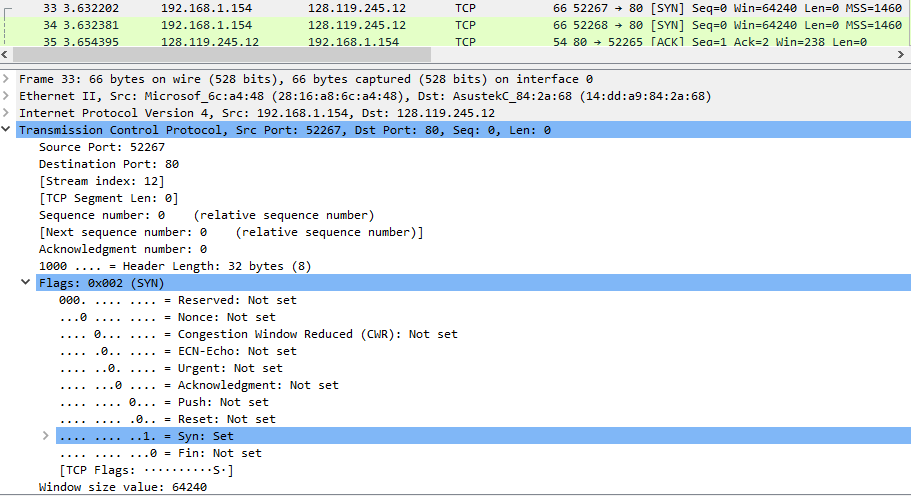


Figure 2.

1. 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 remote server determine that value? What is it in the segment that identifies the segment as a SYNACK segment?

Solution: As shown in Figure 3, the sequence number of the SYNACK segment from gaia.cs.umass.edu to the client computer in reply to the SYN is 0. The value of Acknowledgement field in the SYNACK segment is 1, this value is determined by the server by adding 1 to the sequence number of the SYN segment initiated by the client computer (which is equal to 0). As shown in Figure 3, both SYN flag and Acknowledgement flag in the segment are set to 1, that indicates this segment is a SYNACK segment.

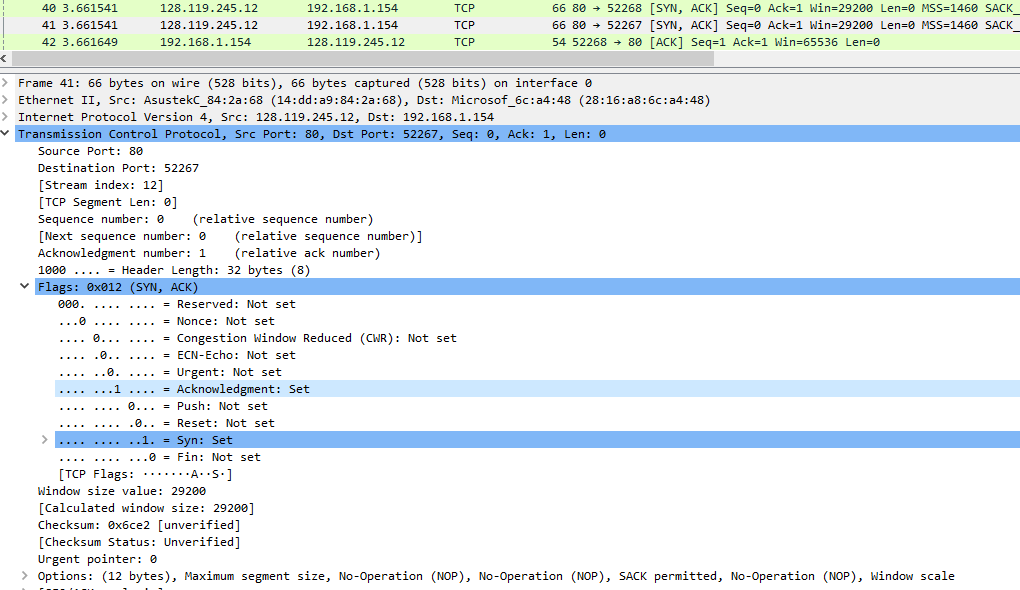


Figure 3.

1. 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.

Solution: As shown in Figure 4, the segment No. 44 contains the HTTP POST command which is with a “POST” in the DATA field.

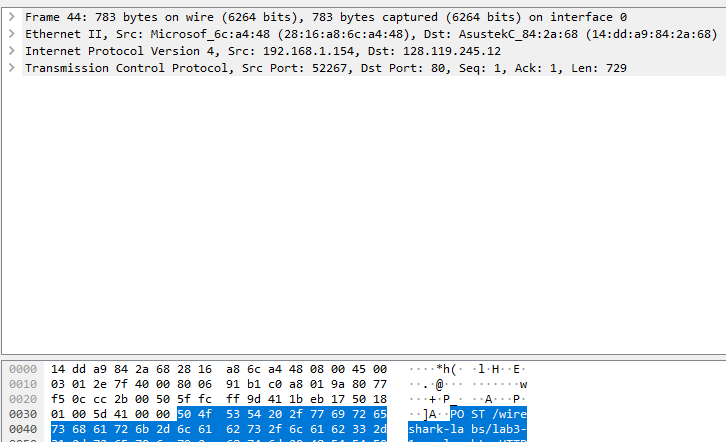


Figure 4.

1. 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 after the receipt of each ACK?

Solution: As shown in Figure 5, the first six segments are No. 44, 45, 46, 47, 48, and 49. The ACK of segments 1 to 6 are No. 57, 59, 60, 65, 68 and 73.

Segment 1st sequence number is 1;

Segment 2nd sequence number is 730;

Segment 3rd sequence number is 2190;

Segment 4th sequence number is 3650;

Segment 5th sequence number is 5110;

Segment 6th sequence number is 6570.

The segment sent time and ACK received time and RTT values are as following:

|  |  |  |  |
| --- | --- | --- | --- |
|  | segment sent time (s) | ACK received time (s) | RTT value (s) |
| Segment 1 | 3.662414 | 3.683973 | 0.021559 |
| Segment 2 | 3.662779 | 3.687715 | 0.024936 |
| Segment 3 | 3.662783 | 3.687715 | 0.024932 |
| Segment 4 | 3.662784 | 3.688037 | 0.025253 |
| Segment 5 | 3.662881 | 3.689886 | 0.027005 |
| Segment 6 | 3.662881 | 3.693118 | 0.030237 |

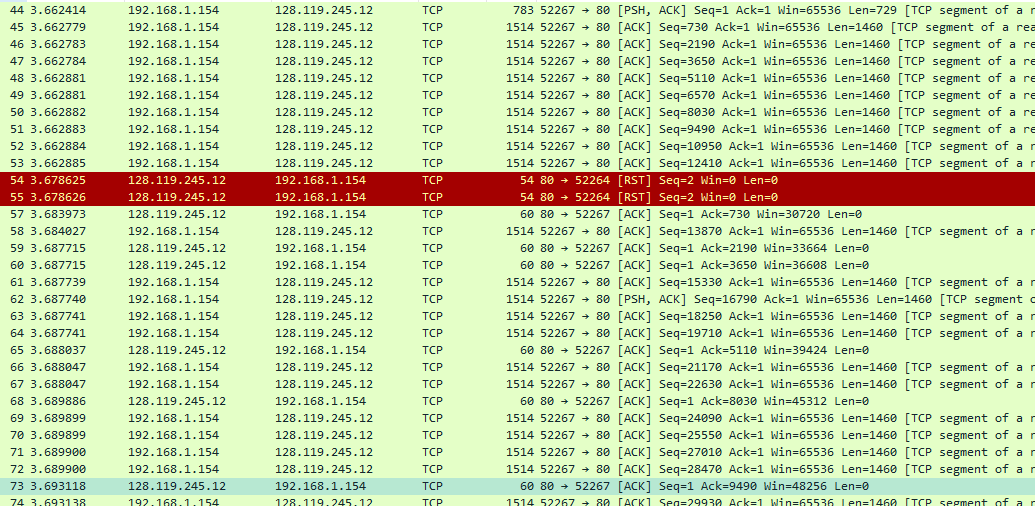


Figure 5.

According to the equation: **EstimatedRTT = 0.875 \* (previous) EstimatedRTT + 0.125 \* SampleRTT**

EstimatedRTT after the receipt of the ACK of segment 1 = RTT for Segment 1 = 0.021559 s

EstimatedRTT after the receipt of the ACK of segment 2 = 0.875\*0.021559 + 0.125\*0.024936

= 0.021981125 s

EstimatedRTT after the receipt of the ACK of segment 3 = 0.875\*0.021981125 + 0.125\*0.024932

= 0.022349984375 s

EstimatedRTT after the receipt of the ACK of segment 4 = 0.875\*0.022349984375 + 0.125\*0.025253

= 0.022712861328125 s

EstimatedRTT after the receipt of the ACK of segment 5 = 0.875\*0.022712861328125 + 0.125\*0.027005

= 0.0232493786621 s

EstimatedRTT after the receipt of the ACK of segment 6 = 0.875\*0.0232493786621 + 0.125\*0.030237

= 0.024122831329 s

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

Solution: As shown in Figure 6, the 1st TCP segment data is 729 bytes, and in the same way I find the 2nd TCP segment data is 1460 bytes, the 3rd TCP segment data is also 1460 bytes, the 4th TCP segment data is 1460 bytes, the 5th TCP segment data is 1460 bytes, and the 6th TCP segment data is 1460 bytes.

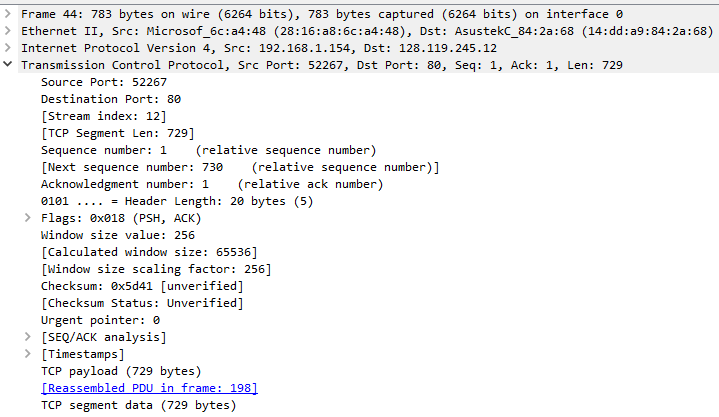


Figure 6.

1. 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?

Solution: The minimum amount of available buffer space (receiver window) advertised at the received for the entire trace is 29200 bytes, which shows as in Figure 7. This receiver window grows steadily until a maximum receiver buffer size. The sender is never throttled due to lacking receiver buffer space.

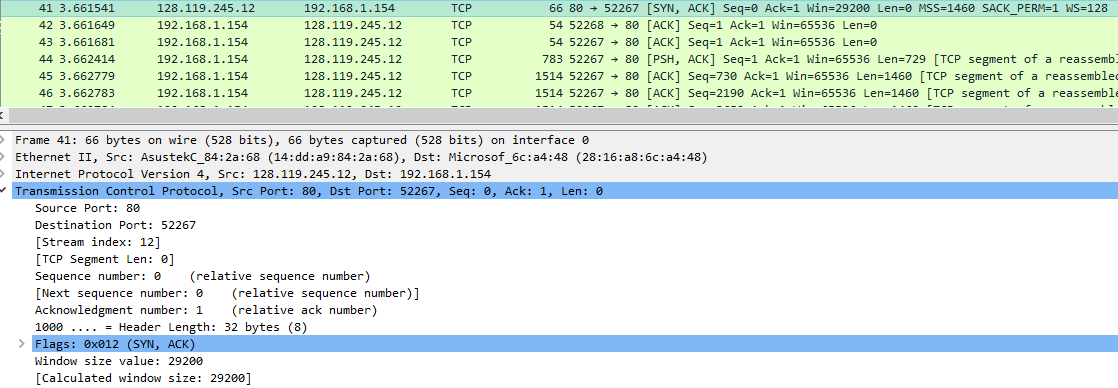


Figure 7.

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

Solution: Figure 8 shows Sequence numbers of the segments from the source (192.168.1.154) to the destination (128.119.245.12). There are no retransmitted segments in the trace file. This can be verified by checking the sequence number in the trace file. Figure 8 is the Sequence Numbers (Stevens) of this trace, all the sequence numbers are increasing with time. If there is a retransmitted segment, the sequence number of this retransmitted segment should be smaller than those of its neighboring segments.

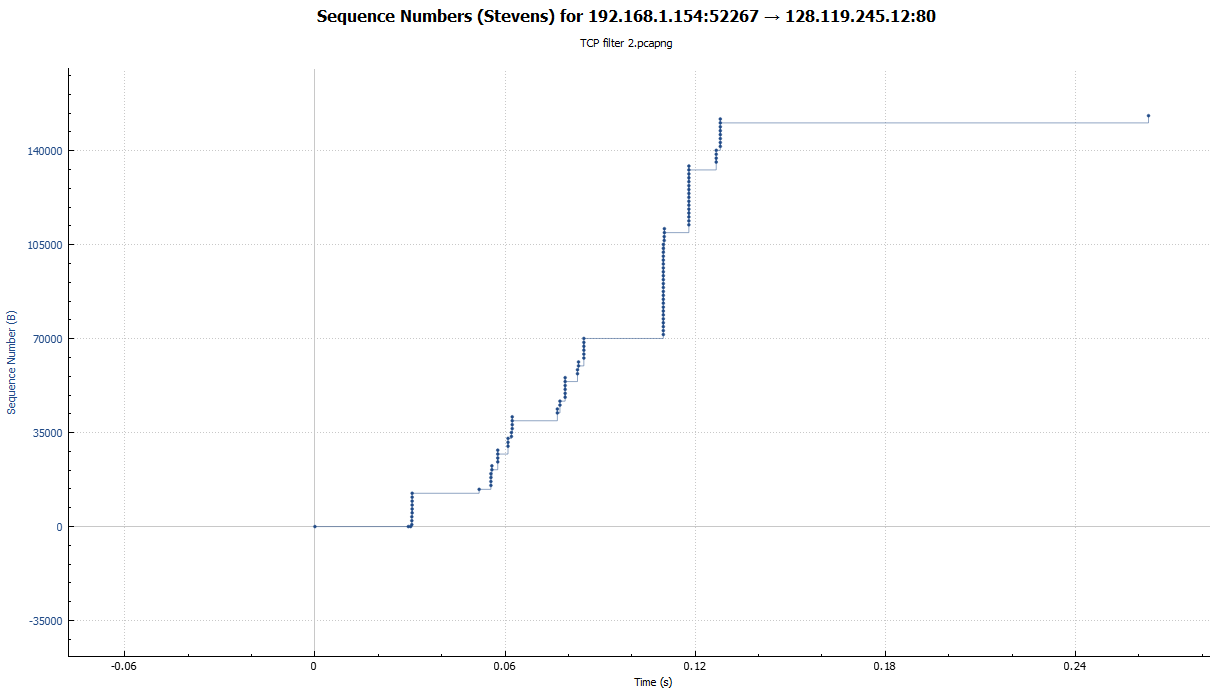


Figure 8.

1. How much data does the receiver typically acknowledge in an ACK? Can you identify cases where the receiver is ACKing every other received segment.

Solution: As shown in Figure 9, the ACK numbers increase in the sequence of 730 (segment 57), 2190 (segment 59), 3650 (segment 60), 5110 (segment 65), and so on. The ACK typically increases by 1460 each time, indicating the receiver usually acknowledges 1460 bytes in an ACK. By inspecting the amount of acknowledged data by each ACK, there are cases where the receiver ACKs every other segment. For example, as shown in Figure 9, the ACK number of segment 65 is 5110, the ACK number of segment 68 is 8030, hence the segment of No. 65 acknowledged data with 8030 - 5110 = 2920 bytes, which is equal to 1460 bytes \* 2 mathematically.

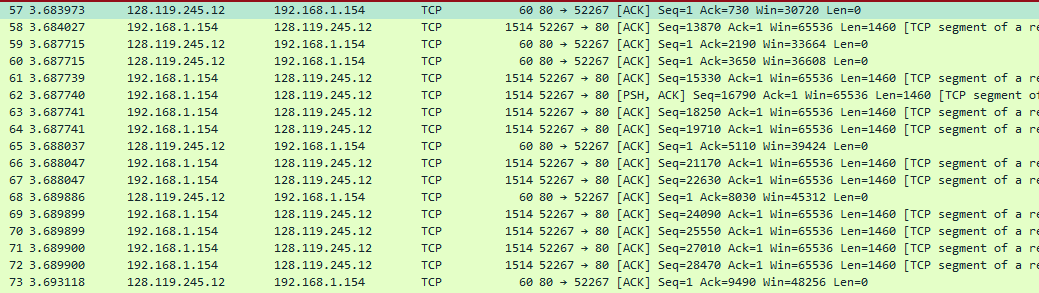


Figure 9.

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

Solution: The average throughput for TCP connection is computed as the ratio of the total amount data and the total transmission time. The total amount data transmitted is the size of alice.txt (152, 138 bytes).

I select the average time period as the whole connection time. The total transmission time is the difference of the time of the first TCP segment (3.662414 for No.44 segment) and the time of the last ACK (3.841475 for No.248 segment), which equals 3.841475 - 3.662414 = 0.179061 s.

The throughput for the TCP connection is 152, 138 bytes/ 0.179061 s = 849, 643 bytes/s

1. 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.

Solution: Figure 8 is the Time-Sequence-Graph (Stevens), by observing Figure 8, we can see the slow-start phase begins at about 0.03 s and ends at about 0.05 s. After that the congestion avoidance takes over, the TCP session is always in congestion avoidance state. The measured data uses only a fraction of the window size. The TCP transmit window does not grow linearly during the phase, this may not be caused by flow control, but may be since HTTP server enforces a rate-limit.

1. Answer each of two questions above for the trace that you have gathered when you transferred a file from your computer to the remote server.
2. Show the captured alice.txt file

Solution: Please see Figure 10 as following:

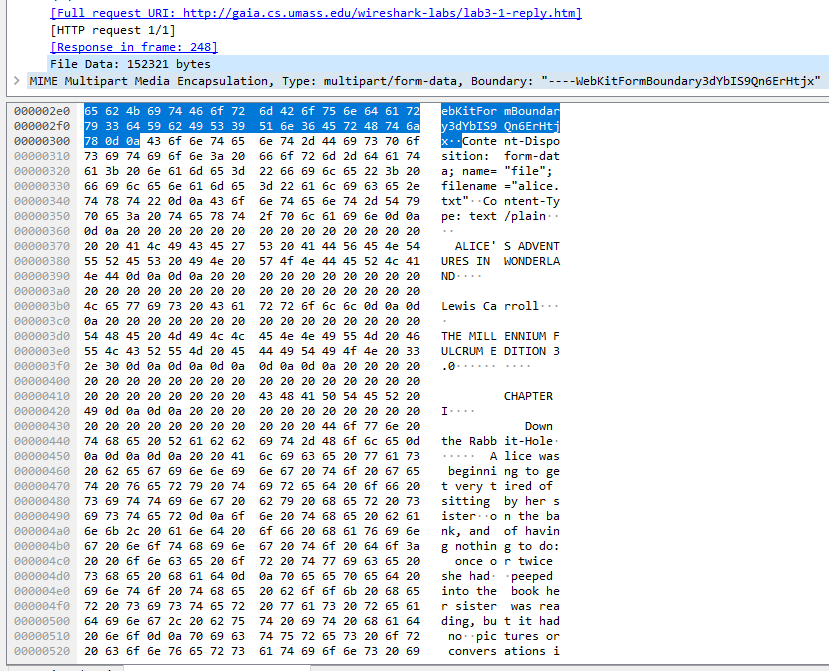


Figure 10.

**B: Capture a bulk TCP transfer from a remote server to your computer**

1. Start WireShark capture
2. Visit website <http://pneumann.com/sec_labs/url-spec.txt>
3. Stop capture

**Questions**:

1. Show the captured url-spec.txt (in plaintext).

Solution: Please see Figure 11 as following:

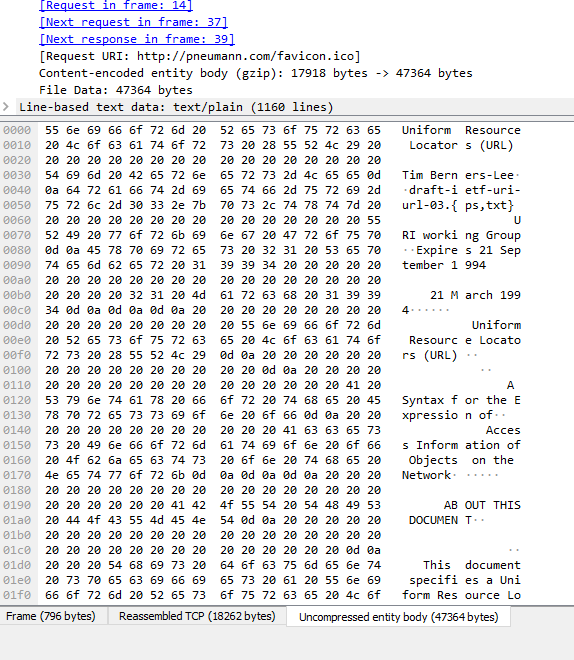


Figure 11.

1. When examining packets, what are the main difference between part A and part B?

Solution: The Hypertext Transfer Protocol (HTTP) is designed to enable communications between client and server, it works as a request-response protocol between a client computer and the server. The biggest difference between part A and B is part A uses POST to send data to a server to create a resource, whereas part B uses GET to request data from a specified resource. Hence I could inspect the sent data through post request and inspect the received data through get response.