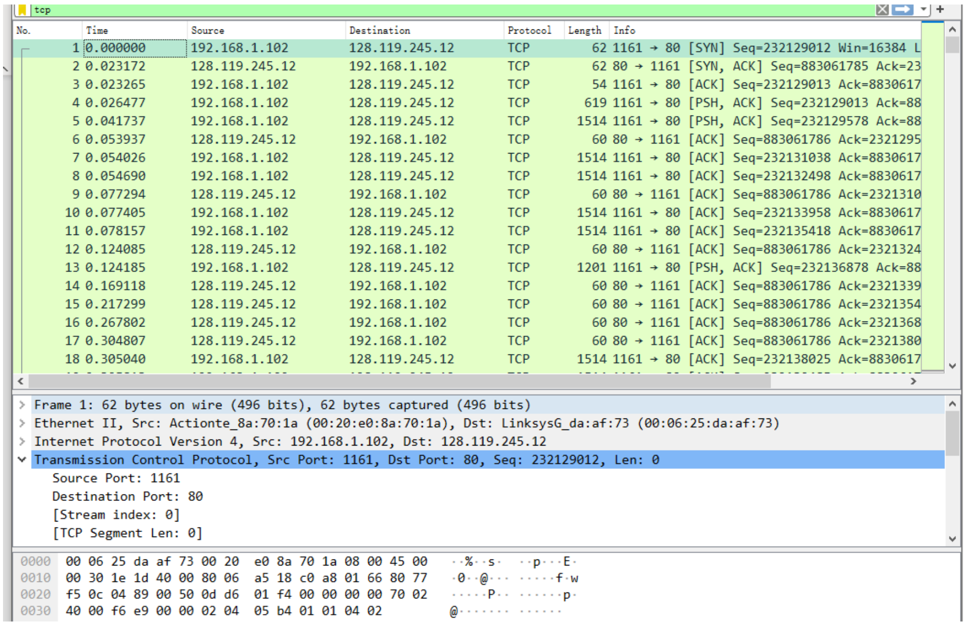
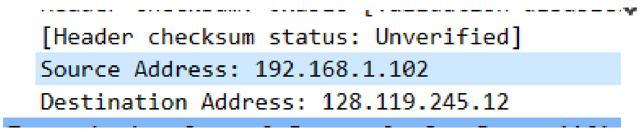
Lab Exercise 4: Exploring TCP

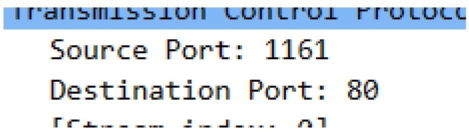
Exercise 1: Understanding TCP using Wireshark



***Question 1* . What is the IP address of gaia.cs.umass.edu? On what port number is it sending and receiving TCP segments for this connection? What are the IP address and TCP port number used by the client computer (source) that is transferring the file to gaia.cs.umass.edu?**

IP address of gaia.cs.umass.edu is 128.119.245.12, it is sending a receiving TCP segments on port number 80. The IP address and TCP port number used by the client computer (source) is 192.168.1.102 and 1161.





***Question 2.* 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.**

The sequence number of the TCP segment containing the HTTP POST command is the 4th packet: Seq = 232129013.

***Question** 3.* 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) sent from the client to the webserver** (Do not consider the ACKs received from the server as part of these six segments)? **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 relevant parts of Section 3.5 or lecture slides) after the receipt of each ACK?** Assume that the initial value of *EstimatedRTT* is equal to the measured RTT (*SampleRTT*) for the first segment, and then is computed using the *EstimatedRTT* equation for all subsequent segments. Set alpha to 0.125.

$$\text{EstimatedRTT} = (1-\alpha)*\text{EstimatedRTT} + \alpha*\text{SampleRTT}$$

Segment #	Seq #	Time sent(s)	ACK #	Time ACK(s)	Sample RTT	Estimated RTT
4	232129013	0.02648	6	0.05394	0.02746	0.02746
5	232129578	0.04174	9	0.07729	0.03555	0.02847
7	232131038	0.05403	12	0.12409	0.07006	0.03367
8	232132498	0.05469	14	0.16912	0.11443	0.04377
10	232133958	0.07741	15	0.21730	0.13989	0.05779
11	232135418	0.07816	16	0.26780	0.18964	0.07426

***Question 4.* What is the length of each of the first six TCP segments? (same six segments as Q3)**

Segment #	Length (Frame and Captured) (bytes)	Payload Length Segment data (bytes)
4	619	565
5	1514	1460
7	1514	1460
8	1514	1460
10	1514	1460
11	1514	1460

***Question 5.* What is the minimum amount of available buffer space advertised at the receiver for the entire trace? Does the lack of receiver buffer space ever throttle the sender?**

The minimum amount of available buffer space advertised at the receiver for the entire trace is the first ACK from the server which is the 2nd packet: 5840bytes.

And the receiver buffer space was keep growing till it reached the max receiver buffer space 62780bytes. Besides, the senders sent at most 1514 bytes at once which is less than the min amount of available buffer space. Thus, the lack of receiver buffer space never throttles the sender.

```
  Source Port: 80
  Destination Port: 1161
  [Stream index: 0]
  [TCP Segment Len: 0]
  Sequence Number: 883061785
  [Next Sequence Number: 883061786]
  Acknowledgment Number: 232129013
  0111 .... = Header Length: 28 bytes (7)
> Flags: 0x012 (SYN, ACK)
  Window: 5840
  [Calculated window size: 5840]
  Checksum: 0x774d [unverified]
  [Checksum Status: Unverified]
  Urgent Pointer: 0
> Options: (8 bytes), Maximum segment size, No-Opera
> [SEQ/ACK analysis]
```

***Question* 6. Are there any retransmitted segments in the trace file? What did you check for (in the trace) in order to answer this question?**

We can check retransmitted segments in the trace file by checking for repeated sequence number + ACK number. And since there are no repeated sequence + ACK number, there are no retransmitted segments in the trace file.

***Question 7.* How much data does the receiver typically acknowledge in an ACK? Can you identify cases where the receiver is ACKing every other received segment (recall the discussion about delayed acks from the lecture notes or Section 3.5 of the text).**

1460 bytes of data are typically acknowledged by the receiver in an ACK, we can tell it from the length(segment data) of most of the TCP segments sent from sender to the receiver. The cases where the receiver is ACKING every other received segment.

One case the receiver is ACKing every other received segment would would be segment # 60, since from segment #59 we known the receiver has ACKed the segment #53. Instead of just acking #54, receiver has acked both segment #54 and #55 in segment #60.

Question 8. **What is the throughput (bytes transferred per unit time) for the TCP connection? Explain how you calculated this value.**

Throughput of the TCP connection = Total bytes transferred/ Total Time

Total bytes transferred = end of the seq #(#206) - start of the seq #(#4) = 232293103 – 232129013 = 164090 bytes

Total time = Time sent of the last segment #206 – Time sent of the first segment #4 = 5.65114s - 0.02648s = 5.62466s

Throughput = 164090 bytes/5.62466s = 29173.32 bytes/s

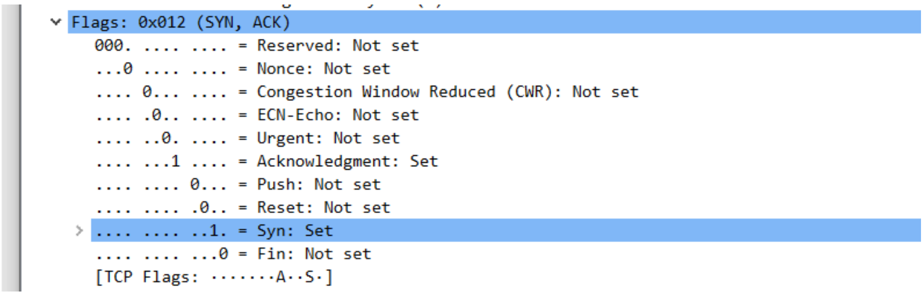
Exercise 2: TCP Connection Management

Question 1 **. What is the sequence number of the TCP SYN segment that is used to initiate the TCP connection between the client computer and server?**

From the TCP SYN segment #295, we know the sequence number is 2818463618

Question 2. **What is the sequence number of the SYNACK segment sent by the server to the client computer in reply to the SYN? What is the value of the Acknowledgement field in the SYNACK segment? How did the server determine that value?**

From the SYNACK segment #296, we know the sequence number is 1247095790. The value of the ack filed in the SYNACK segment is 2818463619. The server determines that value by adding 1 byte (SYN) to the sequence number of the TCP SYN segment: 2818463618 + 1 = 2818463619.



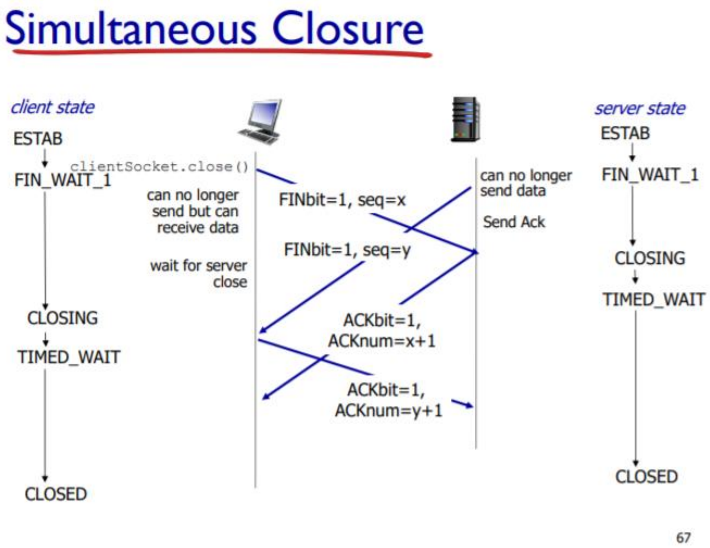
Question 3 **. What is the sequence number of the ACK segment sent by the client computer in response to the SYNACK? What is the value of the Acknowledgment field in this ACK segment? Does this segment contain any data?**

The sequence number of the ACK segment(#3) sent by the client in response to the SYNACK is 2818463619, and the ack value is 1247095791. It does not contain any data.

Question 4 **. Who has done the active close? client or the server? how you have determined this? What type of closure has been performed? 3 Segment (FIN/FINACK/ACK), 4 Segment (FIN/ACK/FIN/ACK) or Simultaneous close?**

304	10.9.16.201	10.99.6.175	TCP	50045 > 5000 [FIN, ACK] Seq=2818463652 Ack=1247095831 win=65535
305	10.99.6.175	10.9.16.201	TCP	5000 > 50045 [FIN, ACK] Seq=1247095831 Ack=2818463652 win=262144
306	10.9.16.201	10.99.6.175	TCP	50045 > 5000 [ACK] Seq=2818463652 Ack=1247095832 win=65535
308	10.99.6.175	10.9.16.201	TCP	5000 > 50045 [ACK] Seq=1247095831 Ack=2818463653 win=262144

From the segment #304 and #305, we know both the client and server has done the active close. The client initiates the active close with a FINACK segment, and the server also response with a FINACK segment to the active close. Simultaneous close type of closure has been performed, we can tell it from the [FIN,ACK] sent from both the client and the server.



Question 5 **. How many data bytes have been transferred from the client to the server and from the server to the client during the whole duration of the connection? What relationship does this have with the Initial Sequence Number and the final ACK received from the other side?**

Th total data bytes transferred from the client to the server during the connections is equal to the final ack sent from server to client – the initial sequence number of the client sent data to server.

Total data = final ack – initial sequence # = 2818463652 – 2818463619 = 33 bytes.

The total bytes transferred from the server to the client during the connection is equal to the final ack from client to server – the initial sequence number of the server.

Total data = final ack – initial sequence # = 1247095831 – 1247095791 = 40 bytes.

Relation: Total data bytes = final Ack received from the other side – initial sequence number