Lab Exercise 4: Exploring TCP

zid: z5228006 name: MINGLANG XIE

Exercise 1: Understanding TCP using Wireshark

For this particular experiment download the trace file: tcp-wireshark-trace-1 .

The following indicate the steps for this experiment:

Step 1: Start Wireshark by typing *wireshark* at the command prompt.

Step 2: Load the trace file *tcp-ethereal-trace-1* by using the *File* pull down menu, choosing *Open* and selecting the appropriate trace file. This file captures the sequence of messages exchanged between a host and a remote server (gaia.cs.umass.edu). The host transfers a 150 KB text file, which contains the text of Lewis Carrol's *Alice's Adventure in Wonderland* to the server. Note that the file is being transferred from the host to the server using a HTTP POST message.

Step 3: Now filter out all non-TCP packets by typing "tcp" (without quotes) in the filter field towards the top of the Wireshark window. You should see a series of TCP segments between the host in MIT and gaia.cs.umass.edu. The first three segments of the trace consist of the initial three-way handshake containing the SYN, SYN ACK and ACK messages. You should see an HTTP POST message in the 4 th segment of the trace being sent from the host in MIT to gaia.cs.umass.edu (check the contents of the payload of this segment). You should observe that the text file is transmitted as multiple TCP segments (i.e. a single POST message has been split into several TCP segments) from the client to the server (gaia.cs.umass.edu). You should also see several TCP ACK segments been returned in the reverse direction.

IMPORTANT NOTE: Do the sequence numbers for the sender and receiver start from zero? The reason for this is that Wireshark by default scales down all real sequence numbers such that the first segment in the trace file always starts from 0. To turn off this feature, you have to click Edit->Preferences>Protocols->TCP (or Wireshark->Preferences->Protocols->TCP) and then disable the "Relative Sequence Numbers" option. Note that the answers in the solution set will reflect this change. If you conduct the experiment without this change, the sequence numbers that you observe will be different from the ones in the answers. Also, set the time shown in the 2nd column as the "Seconds since beginning of capture" under view->Time display format.

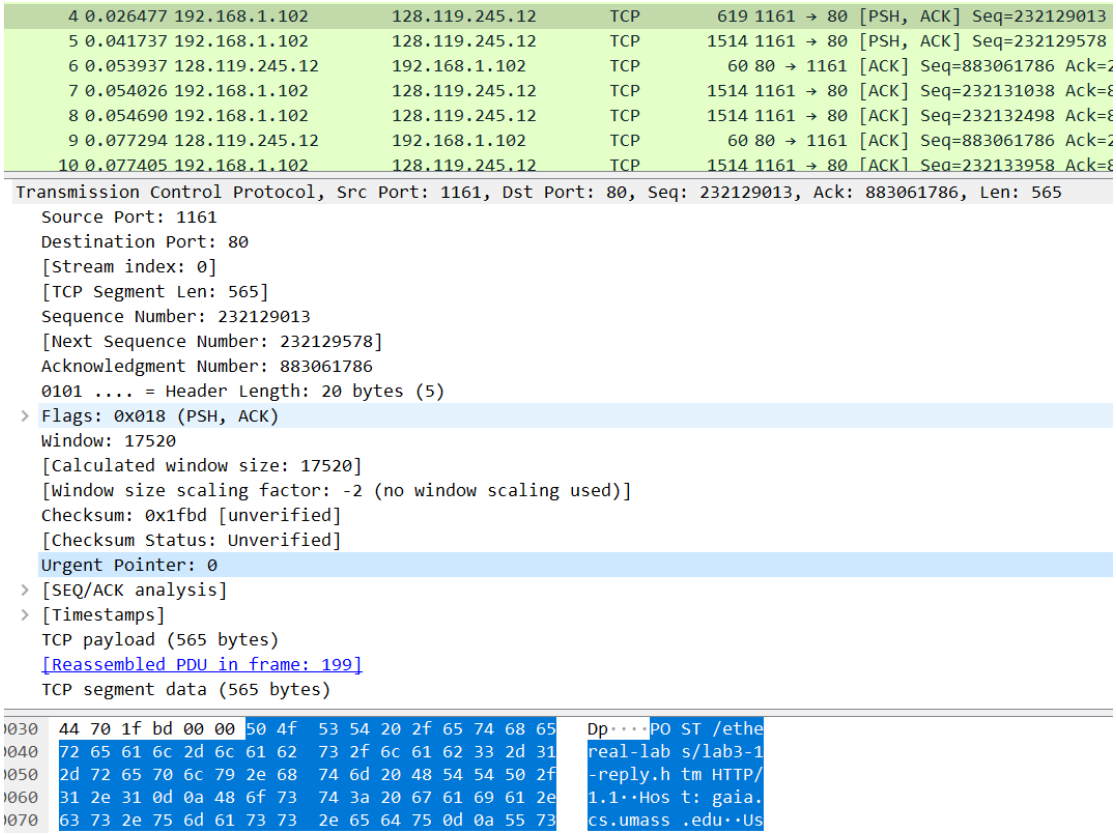
Answer:

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

1	0.000000	192.168.1.102	128.119.245.12	TCP	62 1161 → 80 [SYN] Seq=232129012 Win=16384 Len=0 MSS=1460 SACK_PERM=1
2	0.023172	128.119.245.12	192.168.1.102	TCP	62 80 → 1161 [SYN, ACK] Seq=883061785 Ack=232129013 Win=5840 Len=0 MSS=1460 SACK_PERM=1

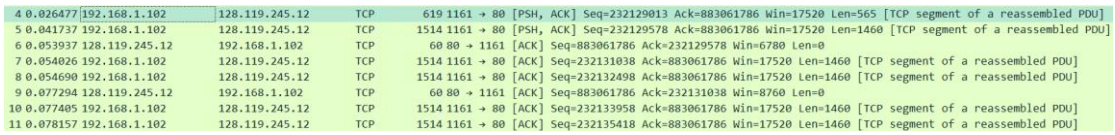
IP address and port number of gaia.cs.umass.edu is 128.119.245.12:80
IP address and port number used by the client computer is 192.168.1.102:1161

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 232129013.

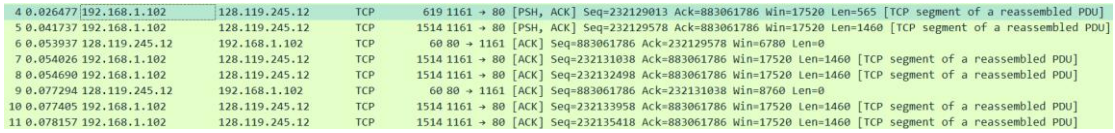
Question 3: Consider the TCP segment containing the HTTP POST as the first segment in the TCP connection. What are the sequence numbers of the first six segments in the TCP connection (including the segment containing the HTTP POST) sent from the client to the web server (Do not consider the ACKs received 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.



No.	Sequence	Time sent	Ack receive	RTT	Estimated RTT
4	232129013	0.026477	0.053937	0.02746	0.02746
5	232129578	0.041737	0.077294	0.035557	0.028472
7	232131038	0.054026	0.124085	0.070059	0.03367
8	232132498	0.054690	0.169118	0.114428	0.043765
10	232133958	0.077405	0.217299	0.139894	0.055781
11	232135418	0.078157	0.267802	0.189645	0.072514

$$Estimated\ RTT\ =\ (1-alpha)*Estimated\ RTT\ +\ alpha*Sample\ RTT$$

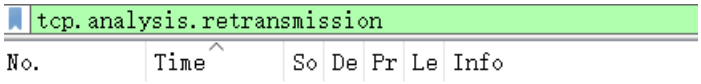
Question 4: What is the length of each of the first six TCP segments? (same six segments as Q3)



No.	Len
4	565
5	1460
7	1460
8	1460
10	1460
11	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 advertise window is 5830 and this is advertised in the SYNACK segment. The lack of receiver buffer space does not throttle the sender, since the window will grow to a reasonable size, the receiver advertised window is very large (mostly 62780 bytes).

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?



There are no retransmitted segments in the trace file. Nothing show up when enter tcp.analysis.retransmission in the filter. You can also verify this by checking the sequence numbers of the TCP segment s in the trace file, all sequence numbers from the source to the destination are increasing monotonically with respect to time.

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).

In the early part of the trace file, we notice that each packet is individually being ACKed by the receiver. However, at segment # 60, the ACK with the acknowledgement field as 232166981is acknowledging two segments with sequence #232164061 and #232165521. After #60, there are several instances when the receiver sends an ACK for every other received segment. This is due to the TCP uses Delayed ACKs where the receiver waits for up to 500 msec for the arrival of another in-order segment and then sends a cumulative ACK for both of the received segemnts.

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

The throughput for the TCP connection is computed as the ratio between the total amount data and the total transmission time.

$$throughput = \frac{Total\ amount\ data}{Total\ transmission\ time}$$

The total amount data can be computed by the difference between the sequence number of the first TCP segment and the acknowledged sequence number of the last ACK. Thus, the total amount data are $232293103 - 232129013 = 164090$.
The total transmission time = #202 ack time (5.455830) - #4 ack time (0.026477) = 5.429353 seconds.

$$throughput = \frac{164090\ bytes}{5.429353\ \text{seconds}} = 30222.75\ bytes/s = \ \ 30.223\ KBytes/s$$

Exercise 2: TCP Connection Management

Consider the following TCP transaction between a client (10.9.16.201) and a server (10.99.6.175).

No	Source IP	Destination IP	Protocol	Info
295	10.9.16.201	10.99.6.175	TCP	50045 > 5000 [SYN] Seq=2818463618 win=8192 MSS=1460
296	10.99.6.175	10.9.16.201	TCP	5000 > 50045 [SYN, ACK] Seq=1247095790 Ack=2818463619 win=262144 MSS=1460
297	10.9.16.201	10.99.6.175	TCP	50045 > 5000 [ACK] Seq=2818463619 Ack=1247095791 win=65535
298	10.9.16.201	10.99.6.175	TCP	50045 > 5000 [PSH, ACK] Seq=2818463619 Ack=1247095791 win=65535
301	10.99.6.175	10.9.16.201	TCP	5000 > 50045 [ACK] Seq=1247095791 Ack=2818463652 win=262096
302	10.99.6.175	10.9.16.201	TCP	5000 > 50045 [PSH, ACK] Seq=1247095791 Ack=2818463652 win=262144
303	10.9.16.201	10.99.6.175	TCP	50045 > 5000 [ACK] Seq=2818463652 Ack=1247095831 win=65535
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

Answer:

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?

Seq=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?

The sequence number of the SYNACK segment sent by the server is 1247095790. The ACK in the SYNACK segment is 2818463619. The server has added 1 in the ISN from the client to arrive at the ACK number.

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 client sent Seq=2818463619 Ack=1247095791. There is no data included in this segment of the three-way handshake. Since the segment 298 as it is using the same Seq No.

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?

The connection is close by a Simultaneous close, which means both client and server have sent the FIN without receiving FIN from the other side (FINs No 304 and 305). ACK is not increasing by 1 in the FINs No 304 and 305.

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?

Client->Server =

$$final\ ACK - ISN - 2(SYN, FIN) = 2818463653 - 2818463618 - 2 = 33 \text{ Bytes}$$

Server->Client =

$$final\ ACK - ISN - 2(SYN, FIN) = 1247095832 - 1247095790 - 2 = 40 \text{ Bytes}$$

The relationship between total number of bytes sent with the initial sequence number and the final ack received.

$$Total\ number\ of\ bytes\ sent = final\ ack\ number - initial\ sequence\ number$$