

COMP9331/3331 LAB4

z5219960 Heng-Chuan Lin

EX1:

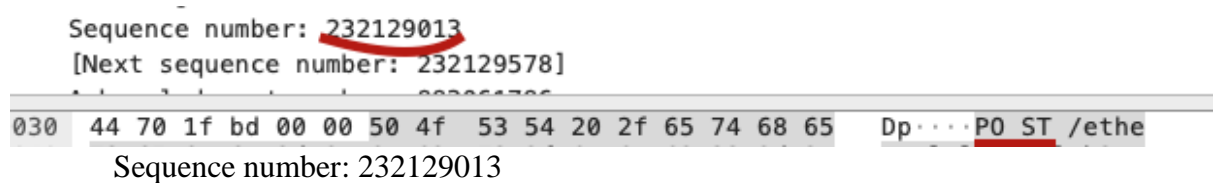
1. What is the IP address of gaia.cs.umass.edu? On what port number is it sending and receiving TCP segments for this connection? What is the IP address and TCP port number used by the client computer (source) that is transferring the file to gaia.cs.umass.edu?

Ans:

gaia.cs.umass.edu: 128.119.245.12 with port # 80
client: 192.168.1.102 with port # 1161

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.

Ans:



Sequence number: 232129013
[Next sequence number: 232129578]
030 44 70 1f bd 00 00 50 4f 53 54 20 2f 65 74 68 65 Dp... P0 ST /ethe
Sequence number: 232129013

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.

Ans:

segment	1	2	3	4	5	6
Seq #	232129013	232129578	232131038	232132498	232133958	232135418
Time sent(sec)	0.026477	0.041737	0.054026	0.054690	0.077405	0.078157
ACK-Time Received(sec)	0.053937	0.077294	0.124085	0.169118	0.217299	0.267802
RTT (sec)	0.02746	0.035557	0.070059	0.114428	0.139894	0.189645
<i>EstimatedRTT</i> (sec)	0.02746	0.02847	0.03367	0.04376	0.05577	0.07250

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

Ans:

segment	1	2	3	4	5	6
Length(byte)	565	1460	1460	1460	1460	1460

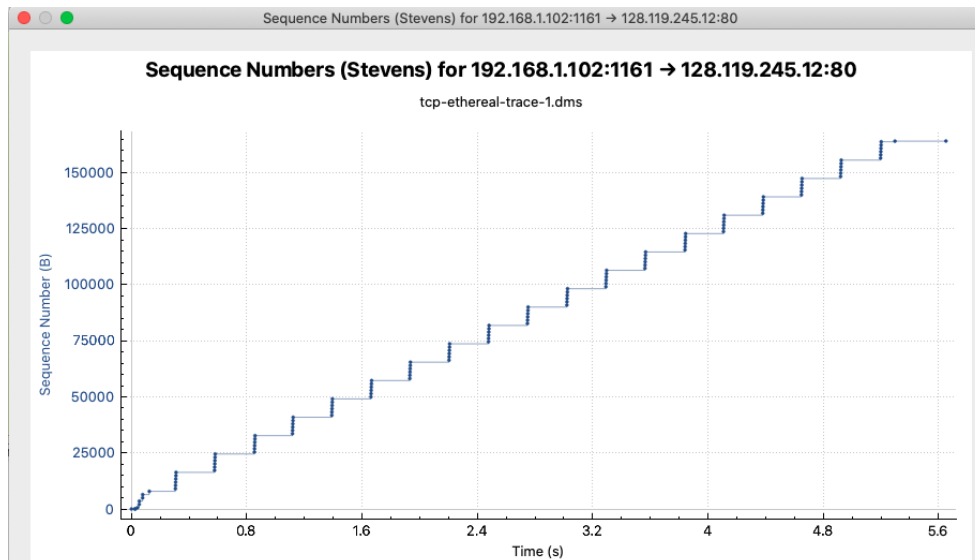
5. What is the minimum amount of available buffer space advertised at the receiver for the entire trace? Does the lack of receiver buffer space ever throttle the sender?

Ans:

The minimum amount of available buffer space advertised at the receiver for the entire trace is 5840 bytes. Then, we could observe that the maximum amount of available buffer space at receiver is 67280 bytes.

Hence, it doesn't throttle the sender because the size of segments is less than this value.

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



To check the retransmission, TCP Time Sequence (Stevens) on Wireshark is applied. The plot indicates that sequence # never go down which means there is no retransmission occurred.

- 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).

Ans:

1	0.000000	192.168.1.102	128.119.245.12	TCP	62	1161 → 80	[SYN, ACK] 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
3	0.023265	192.168.1.102	128.119.245.12	TCP	54	1161 → 80	[ACK] Seq=232129013 Ack=883061786 Win=17520 Len=0
4	0.026477	192.168.1.102	128.119.245.12	TCP	619	1161 → 80	[PSH, ACK] Seq=232129013 Ack=883061786 Win=17520 Len=565 [TCP segment of a reassembled ...]
5	0.041737	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80	[PSH, ACK] Seq=232129578 Ack=883061786 Win=17520 Len=1460 [TCP segment of a reassembled ...]
6	0.053937	128.119.245.12	192.168.1.102	TCP	60	80 → 1161	[ACK] Seq=883061786 Ack=232129578 Win=6780 Len=0
7	0.054026	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80	[ACK] Seq=232131038 Ack=883061786 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
8	0.054690	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80	[ACK] Seq=232132498 Ack=883061786 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
9	0.077294	128.119.245.12	192.168.1.102	TCP	60	80 → 1161	[ACK] Seq=883061786 Ack=232132498 Win=8760 Len=0
10	0.077405	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80	[ACK] Seq=232133958 Ack=883061786 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
11	0.078157	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80	[ACK] Seq=232135418 Ack=883061786 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
12	0.124885	128.119.245.12	192.168.1.102	TCP	60	80 → 1161	[ACK] Seq=883061786 Ack=232132498 Win=11600 Len=0
13	0.124185	192.168.1.102	128.119.245.12	TCP	1201	1161 → 80	[PSH, ACK] Seq=232136878 Ack=883061786 Win=17520 Len=1147 [TCP segment of a reassembled ...]
14	0.169118	128.119.245.12	192.168.1.102	TCP	60	80 → 1161	[ACK] Seq=883061786 Ack=232133958 Win=14600 Len=0
15	0.217299	128.119.245.12	192.168.1.102	TCP	60	80 → 1161	[ACK] Seq=883061786 Ack=232135418 Win=17520 Len=0
16	0.267802	128.119.245.12	192.168.1.102	TCP	60	80 → 1161	[ACK] Seq=883061786 Ack=232136878 Win=20440 Len=0
17	0.304807	128.119.245.12	192.168.1.102	TCP	60	80 → 1161	[ACK] Seq=883061786 Ack=232138025 Win=23360 Len=0

We can observe that receiver typically acknowledge 1460 bytes in an ACK (ACK # - SEQ# with same colour). For example, segment in No.5 row. is ACKed by ACK in No.9 row and segment in No.7 row is ACKed by ACK in No.12 row.

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

1	0.000000	192.168.1.102	128.119.245.12	TCP	62	1161 → 80	[SYN, ACK] 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
3	0.023265	192.168.1.102	128.119.245.12	TCP	54	1161 → 80	[ACK] Seq=232129013 Ack=883061786 Win=17520 Len=0
200	5.389471	128.119.245.12	192.168.1.102	TCP	60	80 → 1161	[ACK] Seq=883061786 Ack=232291321 Win=62780 Len=0
201	5.447887	128.119.245.12	192.168.1.102	TCP	60	80 → 1161	[ACK] Seq=883061786 Ack=232293053 Win=62780 Len=0
202	5.455830	128.119.245.12	192.168.1.102	TCP	60	80 → 1161	[ACK] Seq=883061786 Ack=232293103 Win=62780 Len=0

total transfer data = (ACK# of last segment - 1 for actual bytes) - 1st seq#

$$= (232293103 - 1) - 232129012 = 164090 \text{ bytes}$$

total transfer time = time(#202) - time(#4 start transmission)

$$= 5.455830 - 0.026477$$

$$= \sim 5.429$$

throughput = total transfer data / total transfer time

$$= 164090 / 5.429 \approx 30224 \text{ Bytes/sec}$$

EX2:

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

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?

Ans:

seq# is 2818463618

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?

Ans:

SYNACK = seq# is 1247095790

SYNACK = ACK# is 2818463619

Since SYN contains no data, the server just increases 1 to the ACK# and sent it back.

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?

Ans:

The sequence number of the ACK segment sent by the client computer in response to the SYNACK is 2818463619. The value of the Acknowledgment field in this ACK segment is 1247095791.

Since this the 3rd step of three-way handshake, this segment could include content. This segment contains

$$2818463652 - 2818463619 = 33 \text{ bytes}$$

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?

Ans:

both server and client sent the FIN request actively.
according to the table, server sent the FIN request before
ACKing FIN from client. And this is so called Simultaneous close.

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?

Ans:

The scenario is that client set up TCP connection then sent data to server. after Acking the transferred data, server sent data back to client then they close the connection simultaneously.

From client to server:

$$2818463652 - 2818463619 = 33 \text{ bytes}$$

From server to client:

$$1247095831 - 1247095791 = 40 \text{ bytes}$$

if we use final ACK# - initial seq #,
we would obtain client transfer 35 bytes, and server transfer
42 bytes. These values are total amount of transferred data without
subtraction of the FIN and SYN. To track the total transfer data, SYN
and FIN bit flag should be excluded (no data actually inside SYN and
FIN segment)