# Lab 4 solutions

## Exercise 1

**Question 1**.

The IP address for the host (in MIT) is 192.168.1.102 and the port number is 1161.

**Question 2**.

The IP address of gaia.cs.umass.edu is 128.119.245.12 and the port number used for this connection is 80 (indicating a web server).

**Question 3**.

The sequence number of the SYN segment is 232129012. Clicking on the Flags field in Wireshark reveals the details of all the flag fields. Notice that the SYN flag for this segment has been set to 1, implying that this segment is a SYN segment.

**Question 4**.

The sequence number of the SYNACK segment is 883061785. The acknowledgement field is 232129013. The SYNACK packet is acknowledging the previously received SYN packet with the sequence number 232129012 and since this SYN segment does not contain any data, the TCP receiver on the web server simply increments this value by 1 and copies it to the acknowledgement field. The server is thus indicating to the client that the next segment, which it expects from the client, should contain the sequence number 232129013. On examining the flag fields of this TCP segment, we see that the SYN and ACK flag bits are both set to 1 implying that this is a SYNACK packet.

**Question 5**.

The sequence number of the ACK segment sent by the client is 232129013. Observe how the SYN packet uses up 1 sequence number even if it does not contain any data. The acknowledgment filed contains 883061786 (1 greater than the sequence # of the SYNACK). No this segment does not contain any data. The ACK flag is set to 1 indicating that this is an ACK segment.

**Question 6**.

The sequence number of the TCP segment containing the POST method is 232129013. Notice that the sequence number is same as the sequence number for the ACK segment that was sent prior to this segment. The reason for this is that the ACK segment contained no data and hence the sequence number was not incremented.

**Question 7**.

The details for the first six segments and the corresponding ACKs are provided below along with the calculation of the *EstimatedRTT* :

| No. | Sequence # | Length (in bytes) | Time sent | Time ACK received | Sample RTT | Estimated RTT |
|---|---|---|---|---|---|---|
| 1 | 232129013 | 565 | 0.026477 | 0.053937 | 0.02746 | 0.02746 |
| 2 | 232129578 | 1460 | 0.041737 | 0.077294 | 0.035557 | 0.028472 |
| 3 | 232131038 | 1460 | 0.054026 | 0.124085 | 0.070059 | 0.03367 |
| 4 | 232132498 | 1460 | 0.054690 | 0.169118 | 0.114428 | 0.043765 |
| 5 | 232133958 | 1460 | 0.077405 | 0.217299 | 0.139894 | 0.055781 |
| 6 | 232135418 | 1460 | 0.078157 | 0.267802 | 0.189645 | 0.072514 |

Recall that EstimatedRTT = 0.875 * EstimatedRTT + 0.125 * Sample RTT.

Be careful when you note down the times. Note that the ACK always contains the next expected sequence number in the acknowledgement field.

**Question 8**.

See the above table for the TCP segment lengths. Note that the length in the table indicates the payload contained in the TCP header. It does not include the 20 byte TCP header.

**Question 9**.

To answer this question examine all the ACK (and SYNACK) packets sent by the server to the client. The minimum advertised window is 5840 and this is advertised in the SYNACK segment. The receiver window does not seem to throttle the sender at all. Even when the advertised receiver window is at its lowest value (5840: equivalent to 4 MSS packets of 1460 bytes), the sender seems to be constrained by congestion window as opposed to the flow control window. Notice that in the later part of the trace, when the congestion window has grown to a reasonable size, the receiver advertised window is very large (mostly 62780 bytes: 43 MSS segments). Hence the lack of receiver buffer space does not seem to be an issue with this connection.

**Question 10**.

There are no retransmitted segments in the trace file. If at all a packet is retransmitted there would be a repeat entry for that segment which is retransmitted with the same sequence number. However there is no such occurrence in the entire trace file. <o:p style="line-height: 1.6em; background-color: initial;"></o:p>You can also verify this by checking the sequence numbers of the TCP segments in the trace file. In the *Time- Sequence-Graph (Stevens* ) of this trace, all sequence numbers from the source (192.168.1.102) to the destination (128.119.245.12) are increasing monotonically with respect to time. If there is a retransmitted segment, the sequence number of this retransmitted segment should be smaller than those of its neighbouring segments.

**Question 11**.

In the early part of the trace file (probably slow-start phase), we notice that each packet is individually being ACKed by the receiver. We observe a behaviour where the sender transmits a burst of packets and the receiver then sends back ACKs for each of them. However later on in the trace, in particular at segment # 60, you will notice that the ACK with the acknowledgement field as 232166981 is actually acknowledging two segments with sequence #232164061 and # 232165521. From this point on there are several instances when the receiver sends an ACK for every other received segment. The receiver is typically sending a cumulative ACK for two TCP segments that it receives. As explained earlier this is due to the fact that 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 segments. (Refer to the Receiver event and action table in Section 3.5.4 of the text or Week 6 Lecture notes).

**Question 12**.

The computation of TCP throughput largely depends on the selection of averaging time period. As a common throughput computation, in this question, we select the average time period as the whole connection time. Then, the average throughput for this TCP connection is computed as the ratio between the total amount data and the total transmission time. The total amount data transmitted can be computed by the difference between the sequence number of the first TCP segment (i.e. 1 byte for No. 4 segment) and the acknowledged sequence number of the last ACK (164091 bytes for No. 202 segment). Therefore, the total data are 164091 - 1 = 164090 bytes. The whole transmission time is the difference of the time instant of the first TCP segment (i.e., 0.026477 second for No.4 segment) and the time instant of the last ACK (i.e., 5.455830 second for No. 202 segment). Therefore, the total transmission time is 5.455830 - 0.026477 = 5.4294 seconds. Hence, the throughput for the TCP connection is computed as 164090/5.4294 = 30.222 KByte/sec.