**Author**: Jeremy Vuong

**Course**: CSC 138-03

**Professor**: Jun Dai

**Date**: 03/05/2023

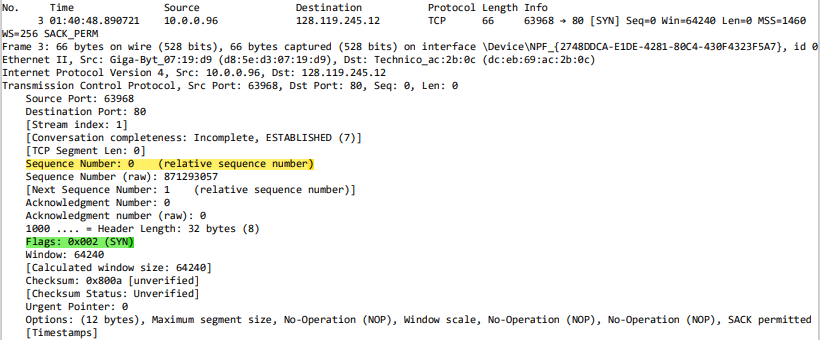
**Wireshark Lab 2-TCP**

1. 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? To answer this question, it’s probably easiest to select an HTTP message and explore the details of the TCP packet used to carry this HTTP message, using the “details of the selected packet header window” (refer to Figure 2 in the“Getting Started with Wireshark” Lab if you’re uncertain about the Wireshark windows.
   1. IP address: 10.0.0.96
   2. TCP port number: 63955
2. What is the IP address of gaia.cs.umass.edu? On what port number is it sending and receiving TCP segments for this connection?
   1. IP address of gaia.cs.umass.edu: 128.119.245.12
   2. Port number: 80

Text

Description automatically generated

1. What is the IP address and TCP port number used by your client computer (source) to transfer the file to gaia.cs.umass.edu?
2. 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?
   1. 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 is 0.
   2. Flags: 0x002 (SYN) identifies the segment as a SYN segment.



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 gaia.cs.umass.edu determine that value? What is it in the segment that identifies the segment as a SYNACK segment?
   1. The sequence number of the SYNACK segment sent by gaia.cs.umass.edu to the client computer in reply to the SYN is 0.
   2. The value of the Acknowledgement field in the SYNACK segment is 1.
   3. gaia.cs.umass.edu determines that value by adding 1 to the initial sequence number of the SYN segment from the client computer.
   4. The SYN flag and the Acknowledgement flag being set to 1 identifies the segment as a SYNACK segment.

Graphical user interface, text

Description automatically generated

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.
   1. The sequence number of the TCP segment containing the HTTP POST command is 1.

Text

Description automatically generated

***Note: I needed to create a new trace for the following questions because I noticed that there were some missing packets that I needed in order to complete the following questions.***

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 (see Section 3.5.3, page 239 in text) after the receipt of each ACK? Assume that the value of the EstimatedRTT is equal to the measured RTT for the first segment, and then is computed using the EstimatedRTT equation on page 239 for all subsequent segments.
   1. The first six segments:
      1. Packet 29, **sequence number**: 1;

**Sent at** 22:19:10.325873;

* + 1. Packet 34, **sequence number**: 14601;

**Sent at** 22:19:10.421928; **received at** 22:19:10.422037

**RTT** = 0.000109 seconds

* + 1. Packet 36, **sequence number**: 29201;

**Sent at** 22:19:10.422042; **received at** 22:19:10.520643

**RTT** = 0.098601 seconds

* + 1. Packet 40, **sequence number**: 43801;

**Sent at** 22:19:10.520654; **received at** 22:19:10.520911

**RTT** = 0.000257 seconds

* + 1. Packet 42, **sequence number**: 58401;

**Sent at** 22:19:10.520782; **received at** 22:19:10.615208

**RTT** = 0.094426 seconds

* + 1. Packet 44, **sequence number**: 73001;

**Sent at** 22:19:10.520852; **received at** 22:19:10.615786

**RTT** = 0.094934seconds

Table

Description automatically generated with medium confidence

Graphical user interface, application

Description automatically generated

Graphical user interface, text

Description automatically generated

Graphical user interface, text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Note: Wireshark has a nice feature that allows you to plot the RTT for each  
of the TCP segments sent. Select a TCP segment in the “listing of  
captured packets” window that is being sent from the client to the  
gaia.cs.umass.edu server. Then select: Statistics->TCP Stream Graph-  
>Round Trip Time Graph.

Chart, line chart

Description automatically generated

1. What is the length of each of the first six TCP segments?
   1. The length of each of the first six TCP segments are 14600 bytes

A picture containing table

Description automatically generated

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?
   1. The minimum amount of available buffer space advertised at the receiver for the entire trace was 29200 bytes.
   2. No, the lack of receiver buffer space did not ever throttle the sender. The buffer continued to grow.

Graphical user interface, text, application

Description automatically generated

1. Are there any retransmitted segments in the trace file? What did you check for (in the trace) in order to answer this question?
   1. No, there aren’t any retransmitted segments. I referred to the RTT graph in order to answer this question.
2. How much data does the receiver typically acknowledge in an ACK? Can you identify cases where the receiver is ACKing every other received segment (see Table 3.2 on page 247 in the text).
   1. The receiver is typically acknowledging 14,592 bytes of data in an ACK. If we observe the first several ACKs, we can see that 14,592 bytes of data are acknowledged each time.
   2. Yes, the screen shot below shows the receiver ACKing every other received segment.

A picture containing table

Description automatically generated

1. What is the throughput (bytes transferred per unit time) for the TCP connection? Explain how you calculated this value.
   1. calculated window size final ok = 281856 bytes

time difference: 0.713099 - 0.615789 = 0.09731 seconds

throughput = 281856/0.09731 = 2,896,475.1824067413421025588325969 bps

throughput = ~2.9 Mbps

* 1. I figured out the total amount of data transferred and divided it by the time it took to transfer the data.



Graphical user interface, text, application

Description automatically generated

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.

Chart

Description automatically generated

1. Answer each of two questions above for the trace that you have gathered when you  
   transferred a file from your computer to gaia.cs.umass.edu
   1. It looks like TCP’s slow start phase begins at around 0-0.1173 seconds and ends at around 0.2 seconds. Congestion avoidance takes over at around 1 second.