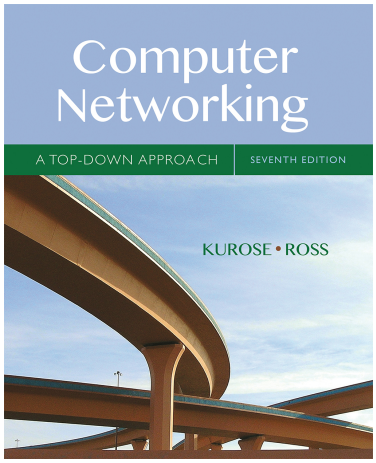
Wireshark Lab: TCP v7.3

Supplement to *Computer Networking: A Top-Down Approach, 7th ed.,* J.F. Kurose and K.W. Ross



"Tell me and I forget. Show me and I remember. Involve me and I understand." Chinese proverb

v7.3 includes changes by Steve Tarzia made in 2018-2020.

In this lab, we'll investigate the behavior of the celebrated TCP protocol in detail. We'll do so by analyzing a trace of the TCP segments sent and received in transferring a 5MB file from your computer to a remote server. We'll study TCP's use of sequence and acknowledgement numbers for providing reliable data transfer; we'll see TCP's congestion control algorithm – slow start and congestion avoidance – in action; and we'll look at TCP's receiver-advertised flow control mechanism. We'll also briefly consider TCP connection setup and we'll investigate the performance (throughput and round-trip time) of the TCP connection between your computer and the server.

Before beginning this lab, you'll probably want to review sections 3.5 and 3.7 in the text[1].

1. Capturing a bulk TCP transfer from your computer to a remote server

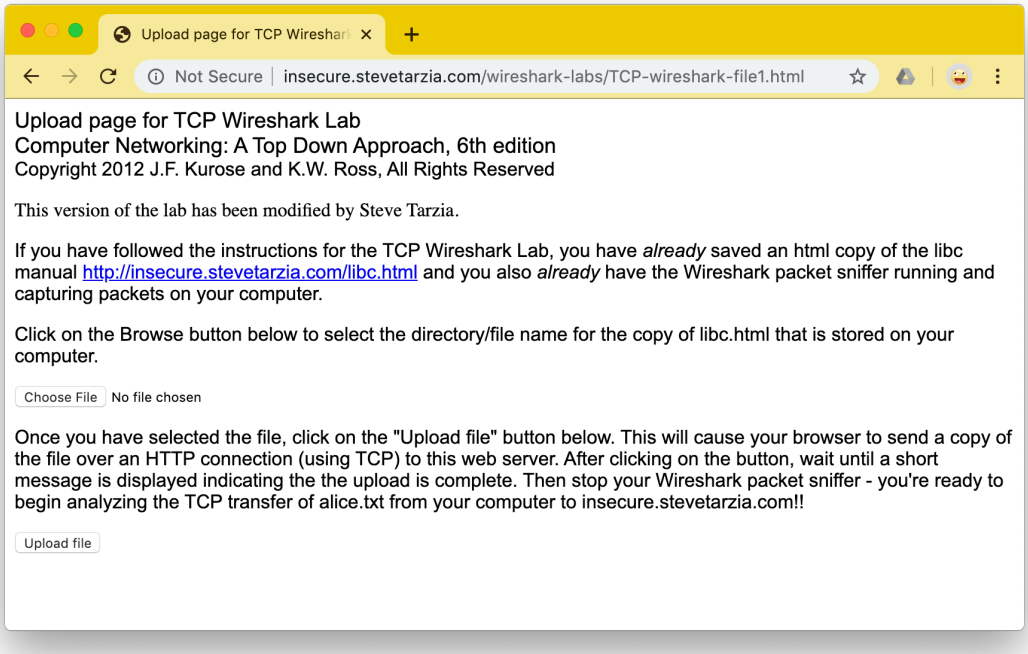
Before beginning our exploration of TCP, we'll need to use Wireshark to obtain a packet trace of the TCP transfer of a file from your computer to a remote server. You'll do so by accessing a Web page that will allow you to enter the name of a file stored on your computer, and then transfer the file to a Web server using the HTTP POST method (see section 2.2.3 in the text). We're using the POST method rather than the GET method as we'd like to transfer a large amount of data *from* your computer to another computer. Of course, we'll be running Wireshark during this time to obtain the trace of the TCP segments sent and received from your computer.

Do the following:

---

[1] References to figures and sections are for the 7th edition of our text, *Computer Networks, A Top-down Approach, 7th ed., J.F. Kurose and K.W. Ross, Addison-Wesley/Pearson, 2016.*

- Start up your web browser. Go to http://insecure.stevetarzia.com/libc.html and retrieve an copy of the GNU libc documentation. Store this file somewhere on your computer (file → save as).
- Next go to http://insecure.stevetarzia.com/wireshark-labs/TCP-wireshark-file1.html
- You should see a screen that looks like:

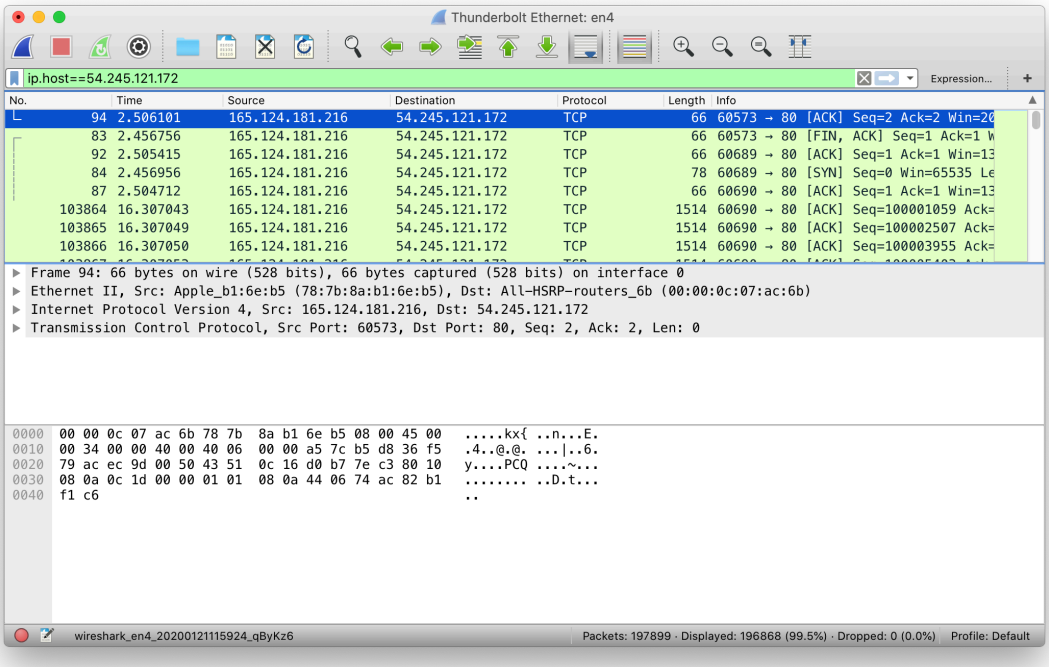


- Use the *Browse* button in this and choose *lib.html* (or whatever you named the downloaded file). Don't yet press the "*Upload file*" button.
- Now start up Wireshark and begin packet capture *(Capture->Start)* and then press *OK* on the Wireshark Packet Capture Options screen (we'll not need to select any options here).
- Returning to your browser, press the "*Upload file*" button to upload the file to the insecure.stevetarzia.com server. Once the file has been uploaded, a short congratulations message will be displayed in your browser window.

2. A first look at the captured trace

Before analyzing the behavior of the TCP connection in detail, let's take a high level view of the trace.

- First, filter the packets displayed in the Wireshark window by entering "ip.host==54.245.121.172" (lowercase, no quotes, and don't forget to press return after entering!) into the display filter specification window towards the top of the Wireshark window.



What you should see is series of TCP and HTTP messages between your computer and insecure.stevetarzia.com. You should see the initial three-way handshake containing a SYN message. You should see an HTTP POST message. Depending on the version of Wireshark you are using, you might see a series of "HTTP Continuation" messages being sent from your computer to insecure.stevetarzia.com. Recall from our discussion in the earlier HTTP Wireshark lab, that is no such thing as an HTTP Continuation message – this is Wireshark's way of indicating that there are multiple TCP segments being used to carry a single HTTP message. In more recent versions of Wireshark, you'll see "[TCP segment of a reassembled PDU]" in the Info column of the Wireshark display to indicate that this TCP segment contained data that belonged to an upper layer protocol message (in our case here, HTTP). You should also see TCP ACK segments being returned from gai.cs.umass.edu to your computer.

Answer the following questions about your network trace. Whenever possible, when answering a question you should hand in a screenshot of the packet(s) within the trace that you used to answer the question asked. Annotate the screenshot to explain your answer.

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

Since this lab is about TCP rather than HTTP, let's change Wireshark's "listing of captured packets" window so that it shows information about the TCP segments containing the HTTP messages, rather than about the HTTP messages. To have Wireshark do this, select *Analyze->Enabled Protocols.* Then uncheck the HTTP box and select *OK*. You should now see the TCP details shown in the screenshot above.

You should see a series of TCP segments sent between your computer and insecure.stevetarzia.com. We will use the packet trace that you have captured to study TCP behavior in the rest of this lab.

Notice that the sequence numbers you see will always start at 1. By default, Wireshark will report "relative sequence numbers" – that is, it will subtract the random initial sequence number offset in order to make your analysis of a particular trace simpler. If you wish to see the real (absolute) sequence numbers you can change this behavior in the Wireshark preferences under *protocols -> TCP*.

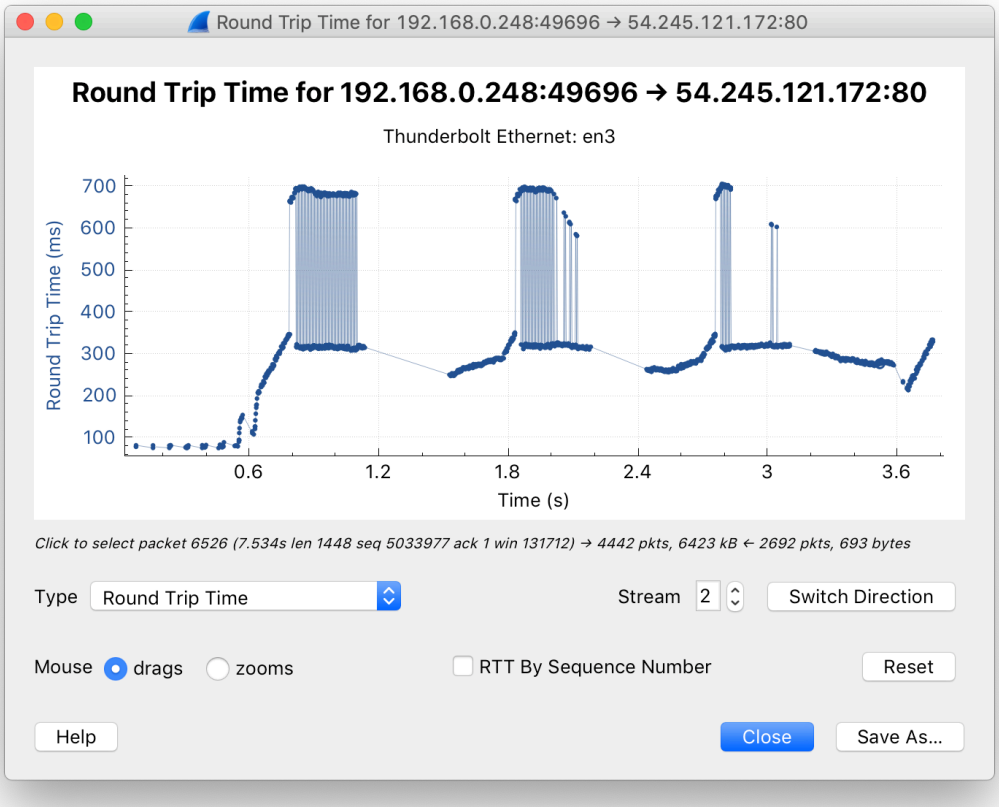
3. TCP Basics

Answer the following questions for the TCP segments:

3. What is the (absolute) sequence number of the TCP SYN segment that is used to initiate the TCP connection between the client computer and insecure.stevetarzia.com? What is it in the segment that identifies the segment as a SYN segment?
4. What is the (absolute) sequence number of the SYNACK segment sent by insecure.stevetarzia.com to the client computer in reply to the SYN? What is the value of the Acknowledgement field in the SYNACK segment? How did insecure.stevetarzia.com determine that value? What is it in the segment that identifies the segment as a SYNACK segment?
5. What is the (absolute) 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.
6. 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 242 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 242 for all subsequent segments.

   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 insecure.stevetarzia.com server. Then select: *Statistics->TCP Stream Graph->Round Trip Time Graph:*

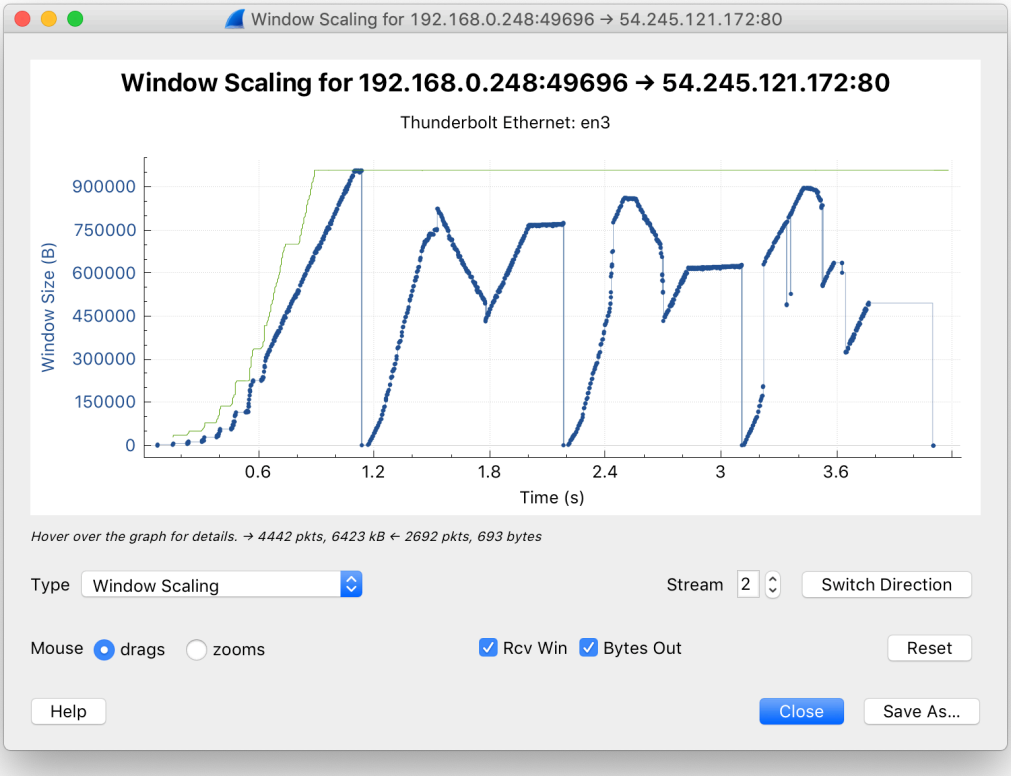


7. What is the length of each of the first six TCP segments?
8. What is the minimum amount of available buffer space advertised at the received for the entire trace? Does the lack of receiver buffer space ever throttle the sender?
9. Are there any retransmitted segments in the trace file? What did you check for (in the trace) in order to answer this question?
10. 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 250 in the text).
11. What is the throughput (bytes transferred per unit time) for the TCP connection? Explain how you calculated this value.

4. TCP congestion control in action

Let's now examine the amount of data sent per unit time from the client to the server. Rather than (tediously!) calculating this from the raw data in the Wireshark window, we'll use one of Wireshark's TCP graphing utilities – *Window Scaling* - to plot out data.

- Select a TCP segment in the Wireshark's "listing of captured-packets" window. Then select the menu : *Statistics->TCP Stream Graph-> Window Scaling*. You should see a plot that looks similar to the following plot:



12. Use the *Window Scaling* plotting tool to view the TCP window size versus time plot for segments being sent from the client to the insecure.stevetarzia.com server. Include a screenshot of your plot. Can you identify in your plot 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.