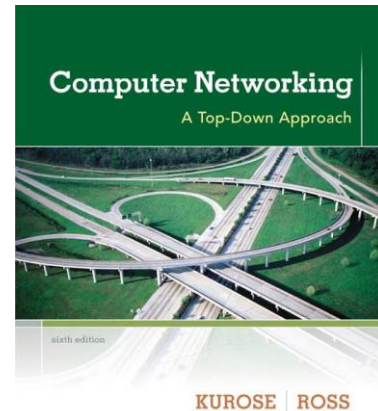


Wireshark Lab: TCP v6.0

Supplement to *Computer Networking: A Top-Down Approach*, 6th 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

© 2005-21012, J.F Kurose and K.W. Ross, All Rights Reserved



Introduction

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 150KB file (containing the text of Lewis Carroll's *Alice's Adventures in Wonderland*) from your computer to a remote server.

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

Access a Web page that will allow you to enter the name of a file stored on your computer (which contains the ASCII text of *Alice in Wonderland*), and then transfer the file to a Web server using the HTTP POST. 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.

Do the following:

- Start up your web browser. Go the <http://gaia.cs.umass.edu/wiresharklabs/alice.txt> and retrieve an ASCII copy of *Alice in Wonderland*. Store this file somewhere on your computer.
- Next go to <http://gaia.cs.umass.edu/wireshark-labs/TCP-wireshark-file1.html>.
- Use the *Browse* button in this form to enter the name of the file (full path name) on your computer containing *Alice in Wonderland* (or do so manually). Don't yet press the “*Upload alice.txt 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 alice.txt file*” button to upload the file to the gaia.cs.umass.edu server. Once the file has been uploaded, a short congratulations message will be displayed in your browser window.
- Stop Wireshark packet capture.

2. A first look at the captured trace

First, filter the packets displayed in the Wireshark window by entering **tcp**. What you should see is series of TCP and HTTP messages between your computer and `gaia.cs.umass.edu`. You should see the initial three-way handshake containing a SYN message. You should see an HTTP POST message. Depending on the version of 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 `gaia.cs.umass.edu` to your computer.

Answer the following questions. When answering, you should hand in a printout of the packet(s) within the trace that you used to answer the question asked. Annotate the printout³ to explain your answer.

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 IP address of `gaia.cs.umass.edu`? On what port number is it sending and receiving TCP segments for this connection?

3. TCP Basics

Answer the following questions for the TCP segments:

3. 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?

4. 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?

5. 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.

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 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 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 gaia.cs.umass.edu 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 receiver 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 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. Select a TCP segment in the Wireshark's "listing of captured-packets" window. Then select the menu : *Statistics->TCP Stream Graph-> Time-Sequence-Graph(Stevens)*.

In the graph, each dot represents a TCP segment sent, plotting the sequence number of the segment versus the time at which it was sent. Note that a set of dots stacked above each other represents a series of packets that were sent back-to-back by the sender.

12. 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.