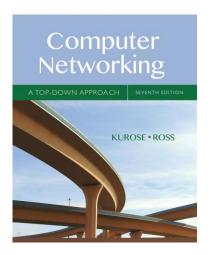
## Wireshark Lab: TCP v7.0

Supplement to *Computer Networking: A Top-Down Approach*, 7<sup>th</sup> 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

© 2005-2016, J.F Kurose and K.W. Ross, All Rights Reserved



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. 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.<sup>1</sup>

# 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 (which contains the ASCII text of *Alice in Wonderland*), 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.

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

Do the following:

- Start up your web browser. Go the <u>http://gaia.cs.umass.edu/wireshark-labs/alice.txt</u> and retrieve an ASCII copy of *Alice in Wonderland*. Store this file somewhere on your computer.
- Next go to <u>http://gaia.cs.umass.edu/wireshark-labs/TCP-wireshark-file1.html</u>.
- You should see a screen that looks like:

🕲 Upload page for TCP Wireshark Lab - Mozilla Firefox	
<u>Fi</u> le <u>E</u> dit <u>V</u> iew <u>G</u> o <u>B</u> ookmarks <u>T</u> ools <u>H</u> elp	
🔶 🗣 🚽 🤔 💿 🏠 🗋 http://gaia.cs.umass.edu/wireshark-labs/TCP-wireshark-fil	e1.html 🔽 🖸 Go 💽
G andrew appel programming - Google Search	le l
Upload page for TCP Wireshark Lab	
Computer Networking: A Top Down Approach, 4th edition	
Copyright 2007 J.F. Kurose and K.W. Ross, All Rights Reserved	
If you have followed the instructions for the TCP Ethereal Lab, you have a <u>http://gaia.cs.umass.edu/ethereal-labs/alice.bt</u> and you also <i>already</i> hav your computer.	
Click on the Browse button below to select the directory/file name for the	copy of alice.txt that is stored on your computer.
Browse	
Once you have selected the file, click on the "Upload alice.bt file" button an HTTP connection (using TCP) to the web server at gaia.cs.umass.ed displayed indicating the the upload is complete. Then stop your Wiresha of alice.bt from your computer to gaia.cs.umass.edu!!	u. After clicking on the button, wait until a short message is
Upload alice.txt file	
🔀 Find: request 💿 Find Next 💿 Find Previous 🗮 Highlight all 🗌 Match ca	se
Done	

- 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. Your Wireshark window should look similar to the window shown below.

0	0			D	🗙 tcp-et	herea	l-trace-	1 [Wire	eshark	1.6.7 (SV	'N Rev 419	973 from /	trunk-1.	6)]					
<u>F</u> ile	e <u>E</u> dit	<u>V</u> iew	<u>G</u> o <u>(</u>	<u>C</u> apture	<u>A</u> nalyz	e <u>S</u>	tatistic	s Tel	ephoi	ny <u>T</u> oo	ls <u>I</u> nter	nals <u>H</u> e	lp						
	<b>ä</b>	94 🚳			<b>X</b>	Z	8	o, <	þ 📦	» 💫 🖞	7 1		₹   ⊕	Θ	ŧ,	<b>→</b>		Y	•
Filt	er: <mark>tcp</mark>	)							▼ E×	pressio	n Cle	ear App	ly						
No.	· · · · · ·	Time		Source				De	stinat	ion		Proto	col Le	ngth	Info				4
		0.30669	-	192.168						245.12		TCP							easser
		0.30757		192.168						245.12		TCP							easser
		0.30869		192.168						245.12		TCP							eassem
		0.30955 0.35643		192.168	3.1.102 9.245.12				.119.2	245.12		TCP TCP							easser ng [AC
		0.35643 0.40016			9.245.12	_			.168.			TCP							.ng [AC
		0.44861			9.245.12				.168.3			TCP							.ng [AC
		0.50002			9.245.12				.168.3			TCP							ng [AC
	28 (	0.54505	2		9.245.12			192	.168.3	1.102		TCP							.ng [AC
	29 (	0.57641	7	128.119	9.245.12	2		192	.168.	1.102		TCP		60	http	> hea	alth-p	olli	ng [AC
		0.57667		192.168				128	.119.2	245.12		TCP							eassem
																			eassem
	32 (	0.57832	9	192.168	3.1.102			128	.119.2	245.12		TCP		1514	[TCP	segm	ent of	ar	easser 🔻
-																			<b>J</b> +
				on wire												- 1			
												a:af:73 ( 3.119.245							
~ 1	Nternet Versio		ot ver	rsion 4,	Src: IS	92.16	8.1.10	2 (192	.168.	1.102),	DST: 128	8.119.245	.12 (12	28.119	1.245	.12)			
		length	. 20	hvtes															
					eld: Ox(	00 (E	DSCP 0x	00: De	fault	: ECN: 0	0x00: No	t-ECT (No	t ECN-	Capab <sup>.</sup>	le Tr	anspo	-t))		
		Length:								,		,					- , ,		
				le2f (772	27)														
⊳	Flags:	0x02	(Don't	Fragment	t)														4
		ent offs																	- 1
		o live:																	
	Protoc	ol: TCF	P (6)																4
					e0 8a 70					s									4
				80 06 9 0d d6						/@									1
0020					6f 20 6														
				6c 69 (					eat	deli qh	nt it f								Ŧ
⊂  F	ile: "/l	Jsers/ki	urose/	/Umass/.	: Pac	kets	:213 C	Display	/ed: 2	02 Mark	ted: 0 Lo	oad time	: 0:00.0	009	Pro	file: D	efaul	t	

If you are unable to run Wireshark on a live network connection, you can download a packet trace file that was captured while following the steps above on one of the authors' computers.<sup>2</sup> You may well find it valuable to download this trace even if you've captured your own trace so that you can use it, as well as your own trace, when you explore the questions below.

### 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 "tcp" (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 gaia.cs.umass.edu. You should see the initial three-way handshake containing a SYN

<sup>&</sup>lt;sup>2</sup> Download the zip file <u>http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.zip</u> and extract the file tcpethereal-trace-1. The traces in this zip file were collected by Wireshark running on one of the author's computers, while performing the steps indicated in the Wireshark lab. Once you have downloaded the trace, you can load it into Wireshark and view the trace using the *File* pull-down menu, choosing *Open*, and then selecting the tcp-ethereal-trace-1 trace file.

message. You should also 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 gaia.cs.umass.edu. 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 gaia.cs.umass.edu to your computer.

Answer the following questions based on your own capture or by opening the Wireshark captured packet file *tcp-ethereal-trace-1* in <u>http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.zip</u> (that is, download the trace and open that trace in Wireshark; see footnote 2). Whenever possible, when answering a question you should hand in a printout of the packet(s) within the trace that you used to answer the question asked. Annotate the printout<sup>3</sup> to explain your answer. To print a packet, use *File*  $\rightarrow$ *Print*, choose *Selected packet only*, choose *Packet summary line*, and select the minimum amount of packet detail that you need to answer the question.

- 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).
- 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?

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*  $\rightarrow$  *Enabled Protocols*. Then uncheck the HTTP box and select *OK*. You should now see a Wireshark window that looks like this:

<sup>&</sup>lt;sup>3</sup> What do we mean by "annotate"? If you hand in a paper copy, please highlight where in the printout you've found the answer and add some text (preferably with a colored pen) noting what you found in what you've highlighted. If you hand in an electronic copy, it would be great if you could also highlight and annotate it as best you can.

0	0	🗙 tcp-ethereal-	trace-1 [Wireshark 1.6.7 (SVN Rev	41973 from /trun	k-1.6)]	
<u>F</u> ile	<u>E</u> dit ⊻iew <u>G</u> o	<u>C</u> apture <u>A</u> nalyze <u>S</u> ta	tistics Telephony <u>T</u> ools <u>I</u> r	nternals <u>H</u> elp		
	i ei ei e	(   🖿 🖪 🗙 😂 🖁	🖥 🔍 🔶 🌳 🖗 🏹		⊕, ⊖, ₪, 砰   🍑 🕅	•
Filter	tcp		▼ Expression	Clear Apply		
No.	Time	Source	Destination	Protocol	Length Info	<u></u>
	1 0.000000	192.168.1.102	128.119.245.12	TCP	62 health-polling > http	[SY
	2 0.023172	128.119.245.12	192.168.1.102	TCP	62 http > health-polling	[SY
	3 0.023265	192.168.1.102	128.119.245.12	TCP	54 health-polling > http	
	4 0.026477	192.168.1.102	128.119.245.12	TCP	619 health-polling > http	
	5 0.041737	192.168.1.102	128.119.245.12	TCP	1514 health-polling > http	
	6 0.053937	128.119.245.12	192.168.1.102	TCP	60 http > health-polling	
	7 0.054026	192.168.1.102	128.119.245.12	TCP	1514 health-polling > http	
	8 0.054690	192.168.1.102	128.119.245.12	TCP	1514 health-polling > http	
	9 0.077294	128.119.245.12	192.168.1.102	TCP	60 http > health polling	
	10 0.077405	192.168.1.102	128.119.245.12	TCP	1514 health-polling > http	
	11 0.078157	192.168.1.102	128.119.245.12	TCP	1514 health-polling > http	
	12 0.124085	128.119.245.12	192.168.1.102	TCP	60 http > health-polling	
_	13 0.124185	192.168.1.102	128.119.245.12	TCP	1201 health-polling > http	and the second se
•						)+
			bytes captured (496 bits)			
			0:e0:8a:70:1a), Dst: Linksys	G_da:af:73 (00:0	06:25:da:af:73)	
ΔD		ksysG_da:af:73 (00:06:2				U
		sysG_da:af:73 (00:06:25				- 11
			t: Individual address (unica			
		8a:70:1a (00:20:e0:8a:	t: Globally unique address (	Tactory default	)	- 11
× 5		nte 8a:70:1a (00:20:e0:8a:				- 11
			t: Individual address (unica	-+)		
			t: Globally unique address (unica		1	-
					/	11
		73 00 20 e0 8a 70 la 0 00 80 06 a5 18 c0 a8 0				
		50 od d6 01 f4 00 00 0				
0030	40 00 f6 e9 00	00 02 04 05 b4 01 01 0	4 02 @			

● File: "/Users/kurose/Umass/... {Packets: 213 Displayed: 202 Marked: 0 Load time: 0:00.011 } Profile: Default

This is what we're looking for—a series of TCP segments sent between your computer and gaia.cs.umass.edu. We will use the packet trace that you have captured (or the packet trace *tcp-ethereal-trace-1* in <u>http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.zip</u>; see earlier footnote) to study TCP behavior in the rest of this lab.

For convenience, you can change the display filter to show only packets between your client and gaia (in case your computer or browser is generating other traffic). To do that, set the filter to "tcp and ip.host == ipaddr", where ipaddr is the IP address of gaia.

### 3. TCP Basics

By default, Wireshark converts actual TCP sequence numbers into *relative* sequence numbers that have been adjusted to start at zero. We don't want that for this lab, since it obscures an important part of the protocol. Choose  $Edit \rightarrow Preferences$  and then click to expand the *Protocols* option. From the long list of known protocols, choose *TCP*. In the options that appear, uncheck *Relative sequence numbers*. Then click *OK*. (Note: after you finish this lab, you'll probably want to re-check that option, since it's handy in other situations.)

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 about the segment identifies it as a SYN segment?
- 4. What is the sequence number of the SYN/ACK 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 SYN/ACK segment? How did gaia.cs.umass.edu determine that value? What is it in the segment that identifies the segment as a SYN/ACK 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 one 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 gaia.cs.umass.edu server. Then select: *Statistics*  $\rightarrow$ *TCP Stream Graphs*  $\rightarrow$ *Round Trip Time.* (You may have to play with some of the buttons to get the results you want.)

- 7. What is the length of each of the first six TCP segments?<sup>4</sup>
- 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?

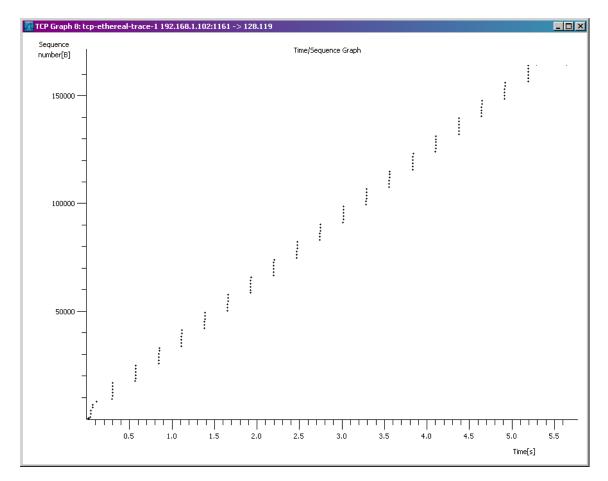
<sup>&</sup>lt;sup>4</sup> The TCP segments in the tcp-ethereal-trace-1 trace file are all less that 1460 bytes. This is because the computer on which the trace was gathered has an Ethernet card that limits the length of the maximum IP packet to 1500 bytes (40 bytes of TCP/IP header data and 1460 bytes of TCP payload). This 1500 byte value is the standard maximum length allowed by Ethernet. If your trace indicates a TCP length greater than 1500 bytes, and your computer is using an Ethernet connection, then Wireshark is reporting the wrong TCP segment length; it will likely also show only one large TCP segment rather than multiple smaller segments. Your computer is indeed probably sending multiple smaller segments, as indicated by the ACKs it receives. This inconsistency in reported segment lengths is due to the interaction between the Ethernet driver and the Wireshark software. We recommend that if you have this inconsistency, that you perform this lab using the provided trace file.

- 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 - *Time-Sequence-Graph(Stevens)* - to plot out data.

• Select a TCP segment in the Wireshark's "listing of captured-packets" window. Then select the menu : *Statistics→TCP Stream Graphs→Time Sequence* (*Stevens*). You should see a plot that looks similar to the following one, which was created from the captured packets in the packet trace *tcp-ethereal-trace-1* in <u>http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.zip</u> (see earlier footnote):



Here, 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.

Answer the following questions for the TCP segments in the packet trace:<sup>5</sup>

12. Use the *Time Sequence (Stevens)* plotting tool to view the sequence numberversus-time plot of segments being sent from the client to the gaia.cs.umass.edu server. Can you identify where TCP's slow-start 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.

<sup>&</sup>lt;sup>5</sup> If your answers seem unreasonable, you should download the file <u>http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.zip</u>, extract the file *tcp-ethereal-trace-1*, and use that to answer these questions.