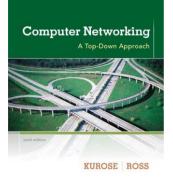
CS 447: Network and Data Communication

Wireshark Lab #01: TCP

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



You may refer to sections 3.5 and 3.7 in the textbook as a review, before beginning the lab.

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 Carrol'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.

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.

Do the following:

- Start up your web browser. Go the <u>http://gaia.cs.umass.edu/wiresharklabs/alice.txt</u> 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.
- You should see a screen that looks like:

C Upload page for TCP Wire ×
< 🕨 S 🏠 🗋 gaia.cs.umass.edu/wireshark-labs/TCP-wireshark-file1.html 🚽 🦁 🗄 🎈 📻 🗖
🗰 Apps 📴 🔁 🌝 🕒 🗅 Safari 🔄 📓 🏠 Moodle 🧾 Banner 🖪 447 🖪 490 🖾 ownCloud 🥹 Deterlab 🛛 » 🗋 Other bookma
Upload page for TCP Wireshark Lab Computer Networking: A Top Down Approach, 6th edition Copyright 2012 J.F. Kurose and K.W. Ross, All Rights Reserved
If you have followed the instructions for the TCP Wireshark Lab, you have <i>already</i> downloaded an ASCII copy of Alice and Wonderland from http://gaia.cs.umass.edu/wireshark-labs/alice.txt and you also <i>already</i> have the Wireshark packet sniffer running and capturing packets on 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.
Choose File No file chosen
Once you have selected the file, click on the "Upload alice.txt file" button below. This will cause your browser to send a copy of alice.txt over an HTTP connection (using TCP) to the web server at gaia.cs.umass.edu. After clicking on the button wait until a short message is displayed indicating the the upload is complete. Then stop your Wireshark packet sniffer - you're ready to begin analyzing the TCP transfer of alice.txt from your computer to gaia.cs.umass.edu!!
Upload alice.txt file

- 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.

00		ereal-trace-1 [Wireshark 1.6.7 (S			
Eile <u>E</u> dit ⊻iew	<u>Go</u> <u>C</u> apture <u>A</u> nalyze	e <u>S</u> tatistics Telephony <u>T</u> oc	ols <u>I</u> nternals <u>H</u> elp		
R R R 8	i 🖗 i 🖻 🖬 🗶 i	3 🔒 🔍 🔶 🏟 🥺	7 1 3) Đ, Q, Q, 🖭 🌌	V .
Filter: tcp		▼ Expression	on Clear Apply		
No. Time	Source	Destination	Protoco	ol Length Info	
20 0.3066	92 192.168.1.102	128.119.245.12	TCP	1514 [TCP segment of	a reassem
21 0.3075	71 192.168.1.102	128.119.245.12	TCP	1514 [TCP segment of	a reassem
22 0.3086	99 192.168.1.102	128.119.245.12	TCP	1514 [TCP segment of	a reassem
23 0.3095	53 192.168.1.102	128.119.245.12	TCP	946 [TCP segment of	
24 0.3564		192.168.1.102	TCP	60 http > health-po	
25 0.4001	128.119.245.12	192.168.1.102	TCP	60 http > health-po	
26 0.4486		192.168.1.102	TCP	60 http > health-po	
27 0.5000		192.168.1.102	TCP	60 http > health-po	
28 0.5450		192.168.1.102	TCP	60 http > health-po	
29 0.5764		192.168.1.102	TCP	60 http > health-po	
30 0.5766		128.119.245.12	TCP	1514 [TCP segment of	
31, 0, 5773		128.119.245.12	TOP	1514 [TCP segment of	
32 0.5783	192.168.1.102	128.119.245.12	TCP	1514 [TCP segment of	
C				_) •
 Ethernet II, S Internet Proto Version: 4 Header lengt Differentiat Total Length Identificati 	rc: Actionte_Ba:70:1a col Version 4, Src: 19: h: 20 bytes eed Services Field: 0x0 :: 1500 on: 0x1e2f (7727) (Don't Fragment) set: 0 : 128	oits), 1514 bytes captured ((00:20:e0:8a:70:1a), Dst: Li 2.168.1.102 (192.168.1.102), 0 (DSCP 0x00: Default; ECN:	nksysG_da:af:73 (00 Dst: 128.119.245.1	2 (128.119.245.12)	
010 05 dc 1e 2f 020 f5 0c 04 89	40 00 80 06 9f 5a c0 00 50 0d d6 4a dd 34 1 00 00 20 74 6f 20 68	la 08 00 45 00%s. a8 01 66 80 77/@ a2 74 1a 50 10P J 65 72 20 67 72 Dpto 20 69 74 20 66 eat deli	.Zf.w J.4.t.P. 5 her gr		

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 author's computers¹. You may well find it valuable to download this trace even if you've captured your own trace and use it, as well as your own trace, when you explore the questions below.

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 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 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

¹ 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.

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 packet capture (or using *tcpethereal-trace-1* in author's Wireshark captured packet file if you ended up using that). 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 printout² to explain your answer. To print a packet, use *File->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->Enabled Protocols*. Then uncheck the HTTP box and select *OK*. You should now see a Wireshark window that looks like:

		🖻 🖪 🗙 😂 🖁	:isticsTelephony Iools Ir] 🔍 🦛 🏟 😜 賽 💈		€ € ₿ 🗗 🙀 🕅
ilter:	tcp		▼ Expression	Clear Apply	
10.	Time	Source	Destination	Protoco	l Length Info
	1 0.000000	192.168.1.102	128.119.245.12	TCP	62 health-polling > http is
	2 0.023172	128.119.245.12	192.168.1.102	TCP	62 http > health-polling [S
	3 0.023265	192.163.1.102	128.119.245.12	TCP	54 health-polling > http [A
	4 0.026477	192.163.1.102	128.119.245.12	TCP	619 health-polling > http [P
	5 0.041737	192.168.1.102	128.119.245.12	TCP	1514 health-polling > http [P
	6 0.053937	128.119.245.12	192.168.1.102	TCP	60 http > health-polling [A
	7 0.054026	192.163.1.102	128.119.245.12	TCP	1514 health-polling > http [A
	8 0.054690	192.163.1.102	128.119.245.12	TCP	1514 health-polling > http [A
	9 0.077294	128.119.245.12	192.168.1.102	TCP	60 http > health-polling [A
	10 0.077405	192.163.1.102	128.119.245.12	TCP	1514 health-polling > http [A
	11 0.078157	192.163.1.102	128.119.245.12	TCP	1514 health-polling > http [A
	12 0.124085	128.119.245.12	192.168.1.102	TCP	60 http > health-polling [A
	13 0.124185	192.163.1.102	128.119.245.12	TCP	1201 health-polling > http [P
)
Frame	e 1: 62 bytes o	n wire (496 bits), 62 b	ytes captured (496 bits)		
			:e0:8a:70:1a), Dst: Linksys0	G_da:af:73 (00:	.06:25:da:af:73)
		sysG_da:af:73 (00:06:25			
		/sG_da:af:73 (00:06:25:0			
			t: Individual address (unica		
			t: Globally unique address (factory defaul	t)
	urce: Actionte_	8a:70:1a (00:20:e0:8a:7	'0:la)		
		nte_8a:70:1a (00:20:e0:8			
			t: Individual address (unica		
	0	= LG bit	t: Globally unique address (factory defaul	t)
00 0	0 06 25 da af 7	73 00 20 e0 8a 70 1a 08	3 00 45 00%spl	Е.	
	0 30 1e 1d 40 0	0 80 06 a5 18 c0 a8 01	L 66 80 77 .0@f	. W	
		50 od d6 01 f4 00 00 00			

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 (and/or the packet trace *tcp-ethereal-trace-1* in http://gaia.cs.umass.edu/wireshark-labs/wiresharktraces.zip; see earlier footnote) to study TCP behavior in

 $^{^{2}}$ 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 highlight. If you hand in an electronic copy, it would be great if you could also highlight and annotate.

the rest of this lab.

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, 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.
 - a. *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 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 247 in the text).