## Wireshark Lab: TCP

Version: 2.0 © 2009 J.F. Kurose, K.W. Ross. All Rights Reserved

KUROSE • ROSS

Computer Networking: A Topdown Approach, 5<sup>th</sup> edition.

In this lab, we'll investigate the behavior of TCP 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.

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> All references to the text in this lab are to *Computer Networking: A Top-down Approach*, 5<sup>th</sup> edition.

Do the following:

- Start up your web browser. Go the <u>http://gaia.cs.umass.edu/wireshark-</u> <u>labs/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:

🕲 Upload page for TCP Wireshark Lab - Mozilla Firefox		K
<u>File Edit V</u> iew <u>G</u> o <u>B</u> ookmarks <u>T</u> ools <u>H</u> elp		2000
🖕 • 🍦 - 🎯 📀 🏠 🗋 http://gaia.cs.umass.edu/wireshark-labs/TCP-wireshark-fil	e1.html 💟 🖸 Go 💽	ב
G andrew appel programming - Google Search		3
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.bd 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	š	~
🔀 Find: request 💿 Find <u>N</u> ext 🔕 Find <u>P</u> revious 📰 Highlight <u>a</u> ll 🗌 Mat <u>c</u> h 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.

📶 (Unl	titled	) - Wir	esharl	٢																		_ 🗆 ×
<u>F</u> ile I	<u>E</u> dit	⊻iew	<u>G</u> o <u>(</u>	apture	Analyze	<u>S</u> tatist	tics	<u>H</u> elp														
	۵	0	0	<b>)</b>	D		x	¢,	8	٩	\$	¢ 1	¢	Ŧ	⊉				€	Θ,	0	**
Eilter:											▼ Expl	ression	⊆le	ear <u>A</u> p	oply							
No. +		ime		Source				Destinat			Prot		info									A
		.000			.68.2.1 .19.24			128.1 192.1			TCF TCF						Seq=					-5940
		.040			.19.24.			128.1			TCF						ACK] Seg=					
		.046			.68.2.1			128.1			HTT						ibs/1a					
		.047			.68.2.1			128.1			HTT						ion-HT					
		.128			19.249 68.2.1			192.1 128.1:			TCF HTT						Seq=: Non-HT				1n=64	32 LE
		.128			.68.2.1			128.1			НТТ						ion-HT					
	9.0	.214	161	128.1	.19.245	5.12		192.1	68.2.	145	TCF	· F					Seq=:				√in=8	712 L
		.214			.68.2.1			128.1			HTT	P C	Ionti	inuat	ior	ı or r	ion-HT	TP 1	traff	ic		
		.214			.68.2.1			128.1			HTT			inuat			ion-HT				Jan 1	1616
		.298			19.249 68.2.1			192.1 128.1			TCF HTT						Seq=: 10n-HT				ATU=T	1010
		.381			19.245			192.1			TCF						Seq=:				√in=1	4520
		.382			.68.2.1			128.1			HTT						ion-HT					
		.382			.68.2.1			128.1			HTT						ion-HT					
		.382			.68.2.1			128.1			HTT						ion-HT		traff	ic		
		.421			.68.2.1 .19.24			192.1 192.1			NBN TCF						IOME<1 Seq=1		-k-62		din_1	7474
		. 552			19.24			192.1			TCF						Seq=					
		. 624			19.24			192.1			TCF						Seq=					
		.624			.68.2.1			128.1			HTT	P C	Ionti	inuat	ior	ı or r	ion-HT	TP 1	traff	ic		
		.624			.68.2.1			128.1			HTT						ion-HT					
		. 624			.68.2.1			128.1			HTT						ion-HT				1.15 10	26126
		.708 .794)			19.249 19.249			192.1 192.1			TCF TCF					[ACK]	Seq=: Seq=:					
		.866			19.245			192.1			TCF					TACK						31944
		.868			.68.2.1			128.1			HTT						ion-HT					
		.869			.68.2.1			128.1			НТТ			inuat			ion-HT					
		.869			.68.2.1			128.1			HTT			inuat			ion-HT					22262
		.950			19.249 19.249			192.1 192.1			TCF TCF					[ACK]						32767 32767
		.108			19.24			192.1			TCF					[ACK]						32767
1										145			ic op		-	[Active	Deq-				1	
	_		- 1	_	<u> </u>		_				_		_							_		F
		-			ı wire,		-			-												
											≘:6d),										0:a8)	
				•				-			145), c					-				2)		
⊕ Tra	ansm	issi	on Co	ontrol	Proto	ocol,	Src	Port	: 125	0 (125	50), Ds	t Por	rt:	http	(80	)), s	eq: O,	Lei	n: 0			
L																						
0000		0c 4			8 00 0					00 45		.AE										
0010					0800					91 80		0+k@.										
0020					0 c2 6 0 02 0			00 00		00 70 02		:://P		• • • •								
1000				50 V	0 V2 V		57	91 VI				• /••	••••		•							
Index Ref.	Inor		LO LU L		ana da						NO. D. C.	10.01		0	~					_		
File: "C:	RDOC	UME~1	(PAULA	.₩~1\LO	CALS~1\1	i emp\eth	ierXX)	xa0310	u 165 k	R 00:00:0	J9    P: 21	4 D: 214	4 M: O	Drops:	U							

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 computers<sup>2</sup>. 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.

<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.

#### 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 message. You should see an HTTP POST message and 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. You should also see TCP ACK segments being returned from gaia.cs.umass.edu to your computer.

Answer the following questions, 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 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?

If you have been able to create your own trace, answer the following question:

3. What is the IP address and TCP port number used by your client computer (source) to transfer the file to gaia.cs.umass.edu?

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 an Wireshark window that looks like:

📶 tcj	o-eth	ereal-ti	ace-1	- Wires	hark															_ 🗆 ×
Eile	<u>E</u> dit	⊻iew	<u>G</u> o <u>C</u>	apture	<u>A</u> nalyze	<u>S</u> tatisti	ics <u>H</u> elp													
	2	0	<b>@</b> (	<b>)</b>	D	K	x 🖏	8	٩	\$	⇔	<b>©</b> 1	T	Ŧ		¥	Ð,	Θ	0	**
Eilter	-									▼ <u>E</u> ×i	pression.	<u>⊂</u> lea	r A	). Apply						
No		Time		Sour	te		Dest	ination			Protocol	Info								<u> </u>
		0.000			.168.1.				245.12		ТСР				[SYN]					
		0.023			.119.24				1.102		TCP									n=584(
		0.023 0.026			.168.1. $.168.1$ .				245.12		TCP TCP									20 Ler n=1752
		0.020 0.041			.168.1				245.12		TCP									Win=1:
		0.053			.119.24				1.102		TCP									5780 Le
		0.054			.168.1				245.12		TCP	-1161	. >	http	[ACK]	Seq=	2026	Ack=1	.Win=	17520
		0.054			.168.1.				245.12		ТСР									17520
		0.077			.119.24				1.102		TCP									8760 l
		0.077			.168.1.				245.12		TCP									17520
		0.078 0.124			.168.1. .119.24				245.12		TCP TCP									17520
		0.124			.168.1.				245.12		TCP									. Win=1
		0.169			.119.24				1.102		TCP									14600
		0.217			.119.24				1.102		TCP	http	>	1161	[ACK]	Seq=:	1 Ack	=6406	Win=	17520
L	16	0.267	802	128	.119.24	45.12	192	.168.	1.102		ТСР	http	) >	1161	[ACK]	Sed=:	1 ACk	=7866	win=	20440 💌
4																				Þ
E FI	ame	7 (1	514 k	ytes	on wir	e, 15	14 bytes	capt	tured)											A
. E1	her	net I	I, Sr	c: Ac	tionte	_8a:7	0:1a (00	:20:	e0:8a:7	'0:1a)	, Dst	: Lin	ksy	′sG_da	:af:73	3 (00:	06:2	5:da:a	af:73	)
							.1.102 (													
E TI	ans	missi	on Co	ntrol	Proto	col.	Src Port	: 110	61 (116	51), D	st Po	rt: ht	ttp	(80)	, Seq	2026	, AC	<: 1.	Len:	1460
E					(1161)	,									· ·		·	,		
					http (	801														
				er: 2			ative se	nuen	ce numh	ner)										
					ber: 34		(relat				her )]									
					mber: :		(relativ				1001 71									
			-	n: 20		L	(I E I a C I V		K Humbe											
					bytes															
		gs: O																		
				-			ow Reduc	.ea (0	LWRJ: N	NUT SE	2C									
					Echo: I		ет													
					nt: No															
	•	1 .		= Ackn	owleda	nent:	Set													<u>-</u>
4																				
0000					3 00 ZC		8a 70 1				%s			E.						A
0010					) 80 06 ) 0d d6		66 c0 a de 34 a				#@.									
0020					) Od Oa		0a 57 6				 Dp									
0040	- 20	) 6e 6	if 77	20 74	17279	69	6e 67 2	0 74	6f 20 0		now t	ry in	ig t	tor						
0050			(5 A1		5 70 61 Janua - Ma				77 70 1				<u>_</u>	in h						<u> </u>
rile: "		uments	and set	ungstPat	iia wing\M	y Docum	ents\Wiresh	arkitrac	es - ethere	ai   Pi	213 D: 2	.13 M: U								

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 <u>http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.zip</u>; see footnote 2) to study TCP behavior in the rest of this lab.

#### 3. TCP Basics

Answer the following questions for the TCP segments:

- 4. 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?
- 5. 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?

- 6. 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.
- 7. 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 page 249 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 249 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*.

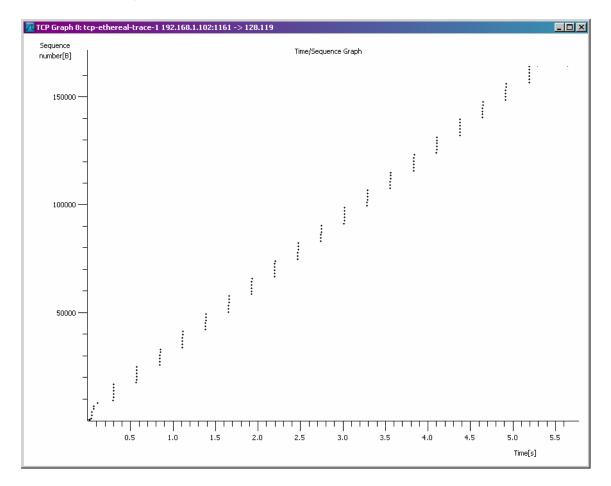
- 8. What is the length of each of the first six TCP segments?<sup>3</sup>
- 9. What is the minimum amount of available buffer space advertised at the received for the entire trace? Does the lack of receiver buffer space ever throttle the sender?
- 10. Are there any retransmitted segments in the trace file? What did you check for (in the trace) in order to answer this question?
- 11. 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 257 in the text).
- 12. What is the throughput (bytes transferred per unit time) for the TCP connection? Explain how you calculated this value.

<sup>&</sup>lt;sup>3</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 report edsegment 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.

### 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 Graph-> Time-Sequence-Graph(Stevens)*. You should see a plot that looks similar to the following plot, 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 footnote 2):



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 the packet trace *tcp-ethereal-trace-1* in <u>http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.zip</u>

- 13. 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.
- 14. 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