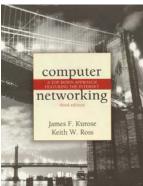
Ethereal Lab: TCP



Computer Networking: A Topdown Approach Featuring the Internet, 3rd edition.

Version: 1.0 © 2005 J.F. Kurose, K.W. Ross. All Rights Reserved

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.¹

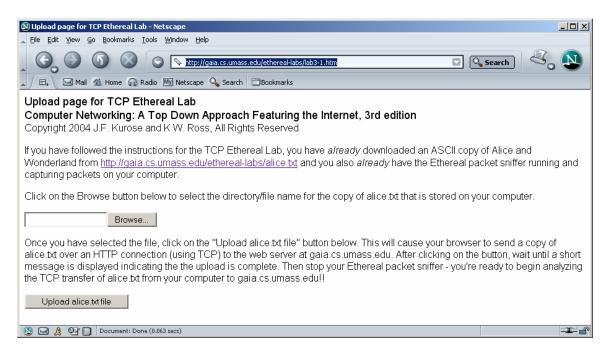
1. Capturing a bulk TCP transfer from your computer to a remote server

Before beginning our exploration of TCP, we'll need to use Ethereal 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 Ethereal during this time to obtain the trace of the TCP segments sent and received from your computer.

Do the following:

¹ All references to the text in this lab are to *Computer Networking: A Top-down Approach Featuring the Internet*, 3rd edition.

- Start up your web browser. Go the <u>http://gaia.cs.umass.edu/ethereal-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/ethereal-labs/TCP-ethereal-file1.html</u>.
- You should see a screen that looks like:



- 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 Ethereal and begin packet capture (*Capture->Start*) and then press *OK* on the Ethereal 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 Ethereal packet capture. Your Ethereal window should look similar to the window shown below.

🕑 lab3-1-trace - Ethereal								
Elle Edit View Go Capture Analyze Statistics Help								
B (🗁 🖪 🗙	e 🖓 📇 💽	\$ \$ \$ \$ \$ \$	⊕,	<			
Elter:								
No. +	Time	Source	Destination	Protocol	Info			
	0.000000	192.168.1.102 128.119.245.12	128.119.245.12 192.168.1.102	TCP TCP	1161 > http [SYN] seq=232129012 Ack=0 Win=16384 Len=0 MSS=1460 http > 1161 [SYN, ACK] seq=883061785 Ack=232129013 Win=5840 Len=			
		192.168.1.102	128.119.245.12	TCP	1161 > http [ACK] Seq=232129013 Ack=883061786 Win=17520 Len=0			
		192.168.1.102	128.119.245.12	HTTP	POST /ethereal-labs/lab3-1-reply.htm HTTP/1.1			
		192.168.1.102	128.119.245.12	HTTP	Continuation			
		128.119.245.12	192.168.1.102	TCP	http > 1161 [ACK] Seq=883061786 Ack=232129578 Win=6780 Len=0			
		192.168.1.102	128.119.245.12	HTTP	Continuation			
		192.168.1.102	128.119.245.12	HTTP	Continuation			
		128.119.245.12	192.168.1.102	TCP	http > 1161 [ACK] Seq=883061786 Ack=232131038 Win=8760 Len=0			
		192.168.1.102	128.119.245.12	HTTP	Continuation			
		192.168.1.102	128.119.245.12	HTTP	Continuation			
		128.119.245.12	192.168.1.102	TCP	http > 1161 [ACK] Seq=883061786 Ack=232132498 win=11680 Len=0			
		192.168.1.102	128.119.245.12	HTTP	Continuation			
		128.119.245.12	192.168.1.102	TCP	http > 1161 [ACK] Seq=883061786 Ack=232133958 Win=14600 Len=0			
		128.119.245.12	192.168.1.102	TCP	http > 1161 [ACK] Seq=883061786 Ack=232135418 Win=17520 Len=0			
		128.119.245.12	192.168.1.102	TCP	http > 1161 [ACK] Seq=883061786 Ack=232136878 Win=20440 Len=0			
		128.119.245.12	192.168.1.102	TCP	http > 1161 [ACK] seq=883061786 Ack=232138025 win=23360 Len=0			
		192.168.1.102	128.119.245.12	HTTP	Continuation			
		192.168.1.102	128.119.245.12	HTTP	Continuation			
		192.168.1.102	128.119.245.12	HTTP	Continuation			
		192.168.1.102	128.119.245.12	HTTP	Continuation	1		
		192.168.1.102	128.119.245.12	HTTP	Continuation	1		
		192.168.1.102	128.119.245.12	HTTP	Continuation	1		
		128.119.245.12	192.168.1.102	TCP	http > 1161 [ACK] Seq=883061786 Ack=232139485 Win=26280 Len=0	1		
		128.119.245.12	192.168.1.102	TCP	http > 1161 [ACK] Seq=883061786 Ack=232140945 Win=29200 Len=0			
		128.119.245.12	192.168.1.102	TCP	http > 1161 [ACK] Seq=883061786 Ack=232142405 Win=32120 Len=0	1		
		128.119.245.12	192.168.1.102	TCP	http > 1161 [ACK] Seq=883061786 Ack=232143865 Win=35040 Len=0	1		
		128.119.245.12	192.168.1.102	TCP	http > 1161 [ACK] Seq=883061786 Ack=232145325 Win=37960 Len=0			
		128.119.245.12	192.168.1.102	TCP	http > 1161 [ACK] Seq=883061786 Ack=232146217 Win=37960 Len=0			
30	0.576671	192.168.1.102	128.119.245.12	HTTP	Continuation	2		
		nt number: 88306	L/86			F		
		1: 20 bytes				H		
	aas: 0x0018	(PSH. ACK)				Ε		
					······			
	00 06 25 da	iaf 73 00 20 e0	8a 70 1a 08 00 45 00	%.	.spE.			
010 0)2 5d 1e 21	. 40 00 80 06 a2	e7 c0 a8 01 66 80 77	.].!	@f.w	Γ		
020 f	5 OC 04 89	00 50 0d d6 01	f5 34 a2 74 1a 50 18		.P4.t.P.	H		
030 4	14 70 1f bc	00 00 50 4f 53	54 20 2f 65 74 68 65	Dр. <u>.</u>	PO ST_/ethe			
	72 65 61 60 24 72 65 70	2d 6c 61 62 73	2f 6c 61 62 33 2d 31	real	-lab s/lab3-1	-		
		0:00:07 P: 213 D: 213 M				-		
ne; iabo-i	I-Grace 177 KB UU	300:07 JP: 213 D: 213 M	1: 0			1		

If you are unable to run Ethereal 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 computers². 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.

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 Ethereal 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 Ethereal 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

² Download the zip file <u>http://gaia.cs.umass.edu/ethereal-labs/ethereal-traces.zip</u> and extract the file tcpethereal-trace-1. The traces in this zip file were collected by Ethereal running on one of the author's computers, while performing the steps indicated in the Ethereal lab. Once you have downloaded the trace, you can load it into Ethereal 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 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 Ethereal lab, that is no such thing as an HTTP Continuation message – this is Ethereal'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 Ethereal captured packet file *tcp-ethereal-trace-1* in <u>http://gaia.cs.umass.edu/ethereal-labs/ethereal-traces.zip</u> (that is download the trace and open that trace in Ethereal; 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 Ethereal" Lab if you're uncertain about the Ethereal 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 Ethereal'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 Ethereal do this, select *Analyze->Enabled Protocols*. Then uncheck the HTTP box and select *OK*. You should now see an Ethereal window that looks like:

🖉 lab3-1-trace - Ethereal								
<u>File Edit View Go Capture Analyze Statistics H</u> elp								
💼 🗁 🏹 × 🕺 📇 🔍 🗢 🗘 🟠								
Elter: tcp	← Expression 🦗 Glear 🗸 Apply							
No Time Source Destination 1 0.000000 192.168.1.102 128.119.245.12 2 0.023172 128.119.245.12 128.119.245.12 3 0.023265 192.168.1.102 128.119.245.12 4 0.026477 192.168.1.102 128.119.245.12 5 0.041737 192.168.1.102 128.119.245.12 6 0.053937 128.119.245.12 192.168.1.102 7 0.054026 192.168.1.102 128.119.245.12 9 0.077294 128.119.245.12 192.168.1.102 10 0.077405 192.168.1.102 128.119.245.12 11 0.076157 192.168.1.102 128.119.245.12 12 0.124485 192.168.1.102 128.119.245.12 13 0.124485 192.168.1.102 128.119.245.12 14 0.169118 128.119.245.12 192.168.1.102 15 0.217299 128.119.245.12 192.168.1.102 16 0.267802 128.119.245.12 192.168.1.102	<pre>TCP http > 1161 [svm, Ack] seq=83061785 Ack=232129013 win=5840 Len= TCP 1161 > http [Ack] seq=232129013 Ack=83061786 win=17520 Len= 1161 > http [P5H, Ack] seq=232129013 Ack=83061786 win=17520 Len= 1161 > http [P5H, Ack] seq=232129578 Ack=83061786 win=17520 Len= 1161 > http [P5H, Ack] seq=232129578 Ack=83061786 win=17520 Len= 1161 > http [Ack] seq=83061786 Ack=833061786 win=17520 Len= 1161 > http [Ack] seq=232131038 Ack=833061786 win=17520 Len= 1161 > http [Ack] seq=232131038 Ack=833061786 win=17520 Len= 1161 > http [Ack] seq=232131058 Ack=833061786 win=17520 Len= 1161 > http [Ack] seq=23213598 Ack=833061786 win=17520 Len= 1161 > http [Ack] seq=23213678 Ack=23213398 win=1680 Len=0 TCP 1161 > http [Ack] seq=833061786 Ack=23213398 win=1680 Len=0 TCP http > 1161 [Ack] seq=833061786 Ack=232133518 win=17520 Len=0 TCP http > 1161 [Ack] seq=833061786 Ack=23213358 win=17520 Len=0 TCP http > 1161 [Ack] seq=833061786 Ack=23213358 win=17520 Len=0 TCP http > 1161 [Ack] seq=833061786 Ack=23213358 win=17520 Len=0 TCP http > 1161 [Ack] seq=833061786 Ack=23213525 win=2360 Len=0 TCP http > 1161 [Ack] seq=833061786 Ack=23213525 win=2360 Len=0 TCP http > 1161 [Ack] seq=232138057 Ack=833061786 win=17520 Len=0 TCP http > 1161 [Ack] seq=232138057 Ack=833061786 win=17520 Len=0 TCP 1161 > http [Ack] seq=232138057 Ack=833061786 win=17520 Len=0 TCP 1161 > http [Ack] seq=232138057 Ack=833061786 win=17520 Len=1460 TCP 1161 > http [Ack] seq=232138057 Ack=833061786 win=17520 Len=1460 TCP 1161 > http [Ack] seq=23213945 Ack=833061786 win=17520 Len=1460 TCP 1161 > http [Ack] seq=232134045 Ack=833061786 win=17520 Len=1460 TCP 1161 > http [Ack] seq=2321342455 Ack=833061786 win=17520 Len=1460 TCP 1161 > http [Ack] seq=2321342455 Ack=833061786 win=17520 Len=1460 TCP 1161 > http [Ack] seq=2321342455 Ack=833061786 win=17520 Len=1460 TCP 1161 > http [Ack] seq=2321342655</pre>							
<pre> Transmission Control Protocol, Src Port: 1161 (11 Source port: 1161 (1161) Destination port: http (80) Sequence number: 232131038 [Next sequence number: 232132498] Acknowledgement number: 883061786 Header length: 20 bytes Flags: 0x0010 (ACK) 0000 00 06 25 da af 73 00 20 e0 8a 70 1a 08 00 45 0010 05 dc 1e 23 40 00 80 06 9f 66 c0 a8 01 66 86 0020 f5 0c 48 90 05 00 dd 00 9d 57 65 20 61 76 0030 44 70 b9 8e 00 00 d0 a 00 4a 57 65 20 61 76 0040 20 66 6f 77 20 74 62 79 69 66 f7 20 74 6f 20 </pre>	2::da:af:73 168.1.102), Dst Addr: 128.119.245.12 (128.119.245.12) 161), Dst Port: http (80), Seq: 232131038, Ack: 883061786, Len: 1460 5 00%5pE. 0 77#@ ff.w 0 10P							

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/ethereal-labs/ethereal-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 Ethereal 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 237 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 237 for all subsequent segments.

Note: Ethereal 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.*

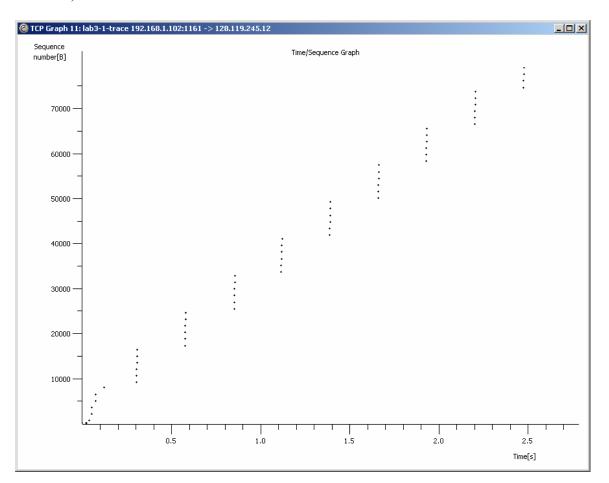
- 8. What is the length of each of the first six TCP segments?³
- 9. What is the minimum amount of available buffer space advertised at the received for the entire trace? Does the lack of receiver buffer space ever throttle the sender?
- 10. Are there any retransmitted segments in the trace file? What 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 245 in the text).
- 12. 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 Ethereal window, we'll use one of Ethereal's TCP graphing utilities - *Time-Sequence-Graph(Stevens)* - to plot out data.

• Select a TCP segment in the Ethereal'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-*

³ 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 Ethereal 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 Ethereal software. We recommend that if you have this inconsistency, that you perform this lab using the provided trace file.



trace-1 in <u>http://gaia.cs.umass.edu/ethereal-labs/ethereal-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/ethereal-labs/ethereal-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? Note that in this "real-world" trace, not everything is quite as neat and clean as in Figure 3.51 (also note that the y-axis labels for the *Time-Sequence-Graph(Stevens*) plotting tool and Figure 3.51 are different).
- 14. Comment on ways in which the measured data differs from the idealized behavior of TCP that we've studied in the text.
- 15. 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