CS 447: Network and Data Communication Wireshark Lab #02: TCP

© 2005-2017, J.F Kurose and K.W. Ross, All Rights Reserved

<u>Note</u>: Make sure you produce your answers and any packet prints in PDF. Moodle will only accept PDF files. Networking A TOP-DOWN APPROACH SEVENTH EDITION KUROSE • ROSS

Computer

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 380KB file (containing the text of Robert Louis

Stevenson's *Treasure Island*) 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 text¹.

• 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 *Treasure Island*), 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://www.gutenberg.org/cache/epub/120/pg120.txt</u> and retrieve an ASCII copy of *Treasure Island*. 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 similar to Figure 01 below.
- Use the *Browse* button in this form to enter the name of the file (full path name) on your computer containing *Treasure Island* (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).

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

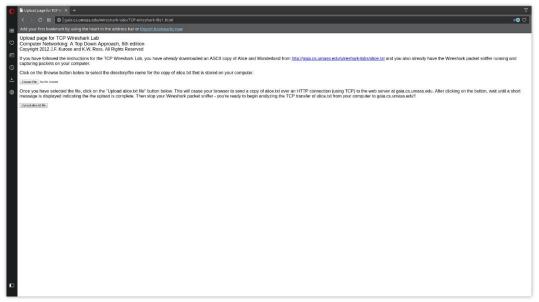


Figure 1

- 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.

	Capture Analyze Statistic	s Telephony W				
📶 🔲 🙇 🎯 🔳	📋 🔀 🙆 🔍 २		> 💶 📃	🔛		
📕 Apply a display filter <ct< th=""><th></th><th></th><th></th><th></th><th>9</th><th>Expression</th></ct<>					9	Expression
	Source D	estination	Protocol Len			
1.0.000000000		55.255.255.255		44688 → 7437 Len=173		
2 1.993634405	24.37.243.78 12	28.119.245.12	TCP	52676 → 80 [FIN, ACK] Seg=1 Ack=1 Win=229 Len=0 TSval=4	76468575 TSecr=40451053	
3 1,993705570		28.119.245.12		52674 → 80 [FIN, ACK] Seg=1 Ack=1 Win=237 Len=0 TSval=4		
4 1.998102025	24.37.243.78 12	28.119.245.12		52680 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PE		
5 2.040153609	128.119.245.12 24	4.37.243.78	TCP	80 → 52674 [ACK] Seg=1 Ack=2 Win=235 Len=0 TSval=404642	ð9 TSecr=476468575	
6 2.040157206	128.119.245.12 24	4.37.243.78	TCP	80 → 52676 [FIN, ACK] Seq=1 Ack=2 Win=227 Len=0 TSval=4	0464209 TSecr=476468575	
7 2.040201838	24.37.243.78 12	28.119.245.12	TCP	52676 → 80 [ACK] Seq=2 Ack=2 Win=229 Len=0 TSval=476468	522 TSecr=40464209	
8 2.044458954	128.119.245.12 24	4.37.243.78	TCP	80 → 52680 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1	460 SACK_PERM=1 TSval=40464213 TSecr=476468579 WS=128	
9 2.044505412	24.37.243.78 12	28.119.245.12	TCP	52680 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=4764	58626 TSecr=40464213	
10 2.044830243	24.37.243.78 12	28.119.245.12	TCP 7	52680 → 80 [PSH, ACK] Seq=1 Ack=1 Win=29312 Len=668 TSv	al=476468626 TSecr=40464213 [TCP segment of a reassembled PDU]	
11 2.044886694	24.37.243.78 12	28.119.245.12	TCP 15	52688 → 80 [ACK] Seq=669 Ack=1 Win=29312 Len=1448 TSval	=476468626 TSecr=40464213 [TCP segment of a reassembled PDU]	
[722 Reassembled TCP [Frame: 18, paylos [Frame: 11, paylos [Frame: 12, paylos [Frame: 13, paylos [Frame: 14, paylos [Frame: 15, paylos [Frame: 16, paylos	d: 0-667 (668 bytes)] d: 668-2115 (1448 bytes) d: 2116-3563 (1448 bytes) d: 3564-5011 (1448 bytes) d: 5612-6459 (1448 bytes) d: 6460-7907 (1448 bytes) d: 7908-9355 (1448 bytes)	: #10(668), #11 11 11 11 11 11 11 11 11 11 11 11 11	(1448), #12(1448		#12(1448), #19(1448), #21(1448), #22(1448), #24(1448), #25(1448), #26(1448), #27(1448), #	29(1448), #30(14
	0a 00 0f 00 42 0c 9b 08 10 40 05 24 d5 18 25 f3					
020 f5 0c cd c8 00 5	60 15 90 b4 c4 e0 85 f0	02 80 18 ···				
	00 01 01 08 0a 1c 66 56 10 73 74 61 66 66 2e 0d		ill st aff.			
	20/3/4 61 66 66 20 00 55 20/63 68 65 63 66 20		ase c heck the			
	52 28 78 61 67 65 73 28 55 6e 74 20 64 6f 6e 61		Web p ages for irrent donatio			
	74 68 6f 64 73 20 61 6e		metho ds and a			
	73 65 73 2e 20 20 44 6f	6e 61 74 ddi				
	51 72 65 20 61 63 63 65 51 28 6e 75 6d 62 65 72		is are accepte in a number of			
	72 0d 0a 77 61 79 73 20		her ways inc			
	57 28 63 68 65 63 6b 73		ling c hecks, o			
Frame (1388 bytes) Reasse	mbled TCP (392414 bytes)					
🔵 🗹 Text item (text)					Packets: 480 · Displayed: 480 (100.0%) · Dropped: 0 (0.0%)	

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 computers². You may

² 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

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 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 capture. 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 printout³ 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:

³ 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.

🔲 🔏 🎯 ± 🛅 🔀 🙆 º				Expressi	
Time Source	Destination		al Lengt Info		
10 4.813034555 24.37.243.78	128.119.245.12	TCP	n zwigy wino 1514 52714 → 80 [ACK] Seq=669 Ack=1 Win=220 Len=1448 TSval=476785866 TSecr=48773082		
11 4.813076360 24.37.243.78	128.119.245.12	TCP	1514 52714 → 80 [ACK] Seq=2117 Ack=1 Min=229 Len=1448 TSval=476785866 TSecr=40773082		
12 4.813104778 24.37.243.78	128.119.245.12	TCP	1514 52714 → 80 [ACK] Seg=3565 Ack=1 Min=229 Len=1448 TSval=476785866 TSecr=40773082		
13 4.813127964 24.37.243.78	128.119.245.12	TCP	1514 52714 → 80 [Ak1] 5eg-5913 Ack=1 Min-229 Len-1448 T5val-476785866 T5ecr-48773882		
14 4.813153234 24.37.243.78	128.119.245.12	TCP	1514 52714 → 80 [ACK] Seg=6461 Ack=1 Min=229 Len=1448 TSva]=476785866 TSecr=40773082		
15 4.813166578 24.37.243.78	128.119.245.12	TCP	1514 52714 → 80 (ACK) Seq-7909 Ack-1 Win-229 Len-1448 TSval-476785866 TSecr-40773082		
16 4.813255405 24.37.243.78	128.119.245.12	TCP	1514 52714 → 80 [ACK] Sequ=9357 Ack=1 Min=229 Len=1448 TSval=476785866 TSecr=40773082		
17 4.813272105 24.37.243.78	128.119.245.12	TCP	1514 52714 → 80 [ACK] Seq-10805 Ack-1 Win-229 Len-1448 T5val-476785866 TSecr-40773082		
18 4.813284767 24.37.243.78	128.119.245.12	TCP	1514 52714 → 80 [ACK] Seg=12253 Ack=1 Win=229 Len=1448 TSval=476785866 TSecr=40773082		
19 4.858857484 128.119.245.12	24.37.243.78	TCP	66 80 → 52714 [ACK] Seq=1 Ack=669 Win=237 Len=0 T5val=40781501 TSecr=476785866		
20 4.858902247 24.37.243.78	128.119.245.12	TCP	1514 52714 → 80 [ACK] Seq=13701 Ack=1 Win=229 Len=1448 T5val=476785912 T5ecr=40781501		
mission Control Protocol, Src Port			F.Bb.98 (ct.97:84:9F.85:0a) Act: 1, Len: 668		
smission Control Protocol, Src Port					
amission Control Protocol, Src Port (668 bytes) c4 e9 84 9f 8b 8a 00 8f 00 42 0c	7.243.78, Dst: 128.114 :: 52714, Dst Port: Bi	9.245.12 0, Seq: 1, A			
nisision Control Protocol, Src Port (668 bytes) c4 e9 84 97 86 02 80 07 60 02 40 07 22 09 74 93 46 08 42 06 42 96 13	7.243.76, Dst: 128.119 :: 52714, Dst Port: 04 95 08 00 45 00	9.245,12 8, Seq: 1, A t.@.@ B %	Act: 1, Lan: 668		
amistion control Protocol, Src Port (668 bytes) cc ao Ba of Bh do Ba of T Bh c2 Ba cc ao Ba of Bh do Ba of T Bh c2 cc ao Ba of the Ba of the Ba cc ao Ba of the Ba of the Ba cc ao Ba of the Ba of the Ba for the Ca of the Ba of the Ca Ba of the Ca of the Ba of the Ca Ba of the Ca ao Ba of the Ca Ba of the Ca ao Ba of the Ca Ba	243.78, Dst: 128.11 252714, Dst Port: 84 90 08 00 45 08 25 13 46 80 77 t a1 ac 29 80 18 0 2c ca 82 66	9,245,12 8, Seq: 1, A t @ @ B % t @ @ B % P 9 	Acto 1, Len: 668		
amisision control frotocol, Src Port (668 bytes) cc en 84 9f 8b 8n 88 9f 98 42 8c 82 48 76 45 48 48 48 49 42 85 18 75 8c cd 8n 48 57 cd 81 82 78 18 76 8c cd 8n 48 57 cd 81 8c 18 81 78 64 50 48 20 48 81 81 80 83 12 20 45 20 45 80 81 81 80 83 12	90 08 00 45 00 25 714, Dst Port: 84 90 08 00 45 00 25 71 46 00 77 t 1 ac 20 80 65 73 66 61 72 % P	9.245.12 a, Seq: 1, A t e e B % bk, p 9.5 bk, P 9.5 bk, P 9.5	Act: 1, Lan: 668		
amistion control frontocol, Src Port (668 bytes) cc ex 84 97 8b 0a 00 0f 00 42 0c 80 0b 74 0b 44 00 42 00 18 76 cc et a 04 05 71 04 08 12 09 76 cc et a 04 05 71 04 08 12 09 25 da 54 45 35 42 07 76 07 20	243.78, Dst. 128.11 : 52714, Dst Port: Bi 2514, Dst Port: Bi 2514, Bi 77, t al ac 29 80 18 bi 26 ca 82 66 65 73 68 61 72, t 63 12 20 75 65, k-1	9.245.12 8, Seq: 1, A t.e.e. B t.e.e. B p. 9 .bk, POST / wires labs/1 ab3-1	Ack: 1, Len: 668		
amistion control Protocol, Src Port (668 bytes) cc4 e9 84 97 86 88 86 87 80 42 8c 22 60 72 6 84 48 88 49 89 42 96 11 15 8c cc 48 89 57 60 42 96 12 15 8c cc 48 85 87 76 46 81 43 98 85 91 42 86 86 91 48 86 42 96 11 15 8c cc 48 85 87 76 86 81 48 81 15 8c cc 48 85 87 76 86 81 48 81 25 28 28 46 12 97 97 97 80 10 81 26 28 28 46 12 97 97 97 80 10 81 26 28 28 46 12 97 97 97 80 10 81 26 28 28 46 12 97 97 97 80 10 81 26 28 28 46 12 97 97 97 80 10 81 26 28 28 46 12 97 97 80 10 81 26 28 28 46 12 97 97 80 10 81 26 28 28 46 12 97 97 80 10 81 26 28 28 46 12 97 97 80 10 81 26 28 28 46 12 97 97 80 10 10 10 10 10 10 10 10 10 10 10 10 10	1.243,76, Dot: 128,114 :: 52714, Dot Port: 87 25 f3 e8 65 76 25 f3 e8 67 7 15 m 25 80 18 60 26 c8 12 68 26 73 68 61 72 % F 2d 31 2d 72 % F 2d 31 2d 72 % F 35 71 26 81 72 % F	9.245.12 8, Seq: 1, A t.e.e. B t.e.e. B p. 9 .bk, POST / wires labs/1 ab3-1	Act: 1, Lan: 668		
amisision control Protocol, Src Port (668 bytes) cc ex 84 97 Bh 85 86 87 89 42 85 85 42 65 74 Bh 85 86 87 99 42 85 85 42 65 74 85 86 88 95 42 65 18 15 85 cc 66 88 95 85 cc 66 43 65 18 15 85 cc 66 88 95 8c 7 66 48 18 23 25 65 85 46 45 95 74 65 18 42 39 25 65 85 46 45 35 54 28 27 76 73 25 65 85 46 45 35 54 28 27 76 73 26 46 46 47 73 74 38 28 07 51 66 26 46 77 73 74 38 28 07 51 66	243.76, 0st 128.11 :: 52714, 0st Post : 52724, 0st Pos	9.245.12 8, Seq: 1, A 9, Seq: 1, C 9, Seq: 1, C 9, Seq: 1, A 9, Seq: 1, C 9, Seq: 1, A 9, Seq: 1, A 1, A 1, A 1, A 1, A 1, A 1, A 1, A	Act: 1, Len: 668		
Anisision Control Trotocol, 52c Port (668 bytes) cc 4 #0 &6 97 bb &m 40 97 00 42 0c 22 db 72 05 44 00 44 00 42 95 11 55 cc cd em 40 57 cd dd 12 39 80 45 31 56 cc de am 40 57 cd dd 12 39 80 45 31 56 cc de am 40 57 cd dd 12 39 80 45 31 24 25 27 37 26 56 47 57 34 30 26 53 32 47 32 46 55 55 75 48 48 37 46 44 66 77 37 37 32 85 66 47 57 48 48 37 46 49 66 46 67 57 36 48 29 48 69 56 55 37 46 61 67 48 20 46 65 55 57 48 48	243.7% Det 128.11 11 52714, Dat Pott B 23.13 49.88 17 23.43 48.98 17 40.8 17 16 50.74 80.77 17 60.2 68.7 16 60.2 82.9 17 60.2 7.68 81.7 61.2 7.2 8.7 24.3 7.2 8.7 25.7 6.8 17.2 26.7 6.8 17.2 23.2 1.3 2.7 24.3 2.7 1.7 25.7 6.8 1.7 25.0 7.6 6.6 24.6 6.6 9.7	9.245.12 8. Seq: 1, A 8. Seq: 1, A 9. Seq: 1	Act: 1, Lan: 008		
aminision control Protocol, Sic Port (666 bytes) cc = 0 84 07 86 69 69 67 00 42 00 20 74 05 48 69 84 69 67 00 43 20 74 05 48 69 84 69 64 79 51 15 60 cc 06 88 65 67 04 01 23 04 50 74 05 48 69 48 69 47 95 11 55 60 cc 06 88 65 67 04 01 23 04 51 42 04 04 10 24 05 74 05 10 02 10 24 05 74 05 10 02 10 25 05 05 05 05 05 05 05 05 10 05 05 05 05 05 05 10 05 05 05 05 10 05 05 05 05 10 05 05 05 05 10 05 100 05 10 05 100 05 100 05 100 05 100 05 10000000000	243.78, 041.128.11 :: 52714, 051.Post.87 :: 52714, 051.Post.97 :: 52714, 051.Post.97 <td :<="" td=""><td>9.245.12 8, Seq: 1, A 9, Seq: 1, A 1, A 1, A 1, A 1, A 1, A 1, A 1, A</td><td>Act: 1, Lan: 668</td><td></td></td>	<td>9.245.12 8, Seq: 1, A 9, Seq: 1, A 1, A 1, A 1, A 1, A 1, A 1, A 1, A</td> <td>Act: 1, Lan: 668</td> <td></td>	9.245.12 8, Seq: 1, A 9, Seq: 1, A 1, A 1, A 1, A 1, A 1, A 1, A 1, A	Act: 1, Lan: 668	
Anistian Control Protocol, 52c Port (668 bytes) cc e0 84 07 86 82 00 07 00 42 0c 82 00 74 06 07 00 42 0c 82 00 74 06 07 00 42 0c 82 00 74 0c 84 05 00 00 00 00 00 00 00 00 00 90 00 00 00 00 00 00 00 00 90 00 00 00 00 00 00 00 90 00 00 00 00 00 00 90 00 00 00 00 00 90 00 00 00 00 00 90 00 00 00 00 90 00 00 00 00 90 00 00 00 00 90 00 00 00 90 00 00 00 90 00 00 00 90 000	-241 78. Det. 128. 11' :: 52714. Det. Post Bet :: 52714. Det. Bet Post Bet :: 52714. Det. Bet	9.245.12 8. Seq: 1, A 8. Seq: 1, A 8. Seq: 1, A 9. Seq	Adt: 1, Len: 608		
amisision control Protocol, Src Port (668 bytes) cc op Bc of Bb dp db df db cc bc db cc op Bc of Bb db db df db cc bc db cc op Bc of Bb db db db db cc bc db cc op Bc of Bb db db db db cc bc db cc op Bc of Bb db db db cc bc db cc op Bc of Bb db db db cc bc db cc op Bc of Bb cc bc db cc bc db cc op Bc of Bb cc bc db cc bc db cc op Bc of Bb cc bc db cc op Bc of Bb cc bc db cc bc db cc op Bc of Bb cc bc db cc bc db cc op Bc of Bb cc bc db cc bc db cc op Bc of Bb cc bc db cc bc db cc op Bc of Bb cc bc db cc op Bc of Bc of Bb cc bc db cc op Bc of Bc of Bb cc	-243.76, 0st 128.11 :: 52714, 0st Post : 52714, 0st Po	9.245.12 #, Seq: 1, A #, Seq: 1, A # P 9 P 9 B - 9 K MICS 1 0012 MICS 1 00	Act: 1, Len: 668		
render Freidencol Version 4.5, Src. 24.73 maintainno (mattrine) Freidencol, Src. 24.73 maintainno (mattrine) (6668 bytes) c 4 40 84 97 bb 6a 80 87 el 44 20 61 22 40 74 6b 44 20 61 30 44 20 44 20 44 20 61 31 65 44 40 44 42 06 11 55 44 40 44 20 44 20 61 30 44 55 44 40 64 44 20 61 31 65 44 40 64 45 64 44 20 61 30 44 55 44 40 64 44 20 61 30 44 55 44 40 64 44 20 61 30 44 55 44 40 64 44 20 61 30 44 55 44 45 44 44 56 44 50 44 56 44 56 44 76 46 37 46 45 14 51 64 56 44 56 44 76 46 37 46 45 14 51 64 56 44 56 44 77 46 56 46 74 46 51 53 15 31 53 15 31 57 34 68 46 53 53 44 46 36 44 77 46 55 47 44 56 53 53 44 46 36 44 77 46 55 47 44 56 53 53 44 46 36 44 77 46 55 47 44 56 53 53 44 46 36 44 77 46 55 47 44 56 53 53 44 46 36 44 77 46 55 47 44 56 53 53 54 66 35 57 46 77 50 55 53 54 54 55 54 55 57 66 57 78 55 55 53 55 56 55 55 56 55 57 66 57 57 56 55 53 55 56 55 55 56 55 57 66 57 57 56 55 54 55 55 55 55 55 55 55 55 55 55 55 55 5	243 78, 041 128, 111 11 22714, 051 Post 28 23 13 40 80 77 23 14 40 80 77 40 20 12 12 12 12 40 20 80 70 61 10 10 40 20 81 90 20 11 10	9.245.12 8. Seq: 1, A 8. Seq: 1, A 8. Seq: 1, A 9. Seq	Act: 1, Lan: 668		

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 earlier footnote) to study TCP behavior in 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 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->TCP Stream Graph->Round Trip Time Graph.*

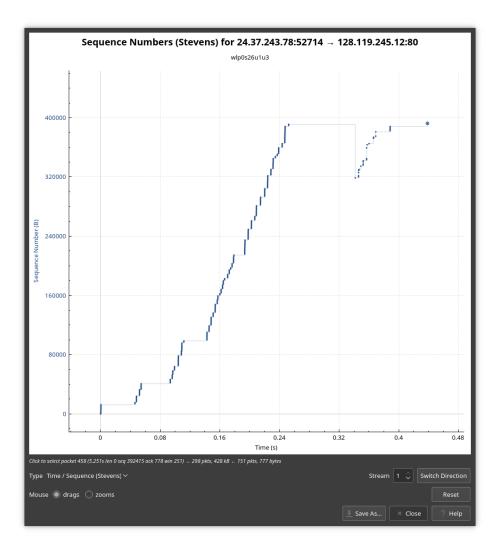
- 7. What is the length of each of the first six TCP segments?⁴
- 8. What is the minimum amount of available buffer space advertised at the received for the entire trace? Does the lack of receiver buffer space ever throttle the sender?
- 9. Are there any retransmitted segments in the trace file? What 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 250 in the text).
- 11. What is the throughput (bytes transferred per unit time) for the TCP connection? Explain how you calculated this value.
- 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.

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.

⁴ 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.



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 during your file transfer earliers. Answer the following questions.

- 12. Can you identify where TCP's slowstart phase begins and ends, and where congestion avoidance takes over?
- 13. Comment on ways in which the measured data differs from the idealized behavior of TCP that we've studied in the text.