

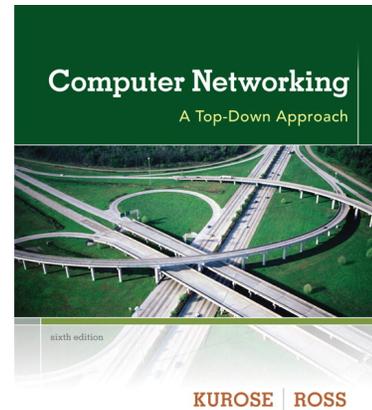
Wireshark Lab: TCP v6.2

Supplement to *Computer Networking: A Top-Down Approach, 6th ed.*, J.F. Kurose and K.W. Ross

With modification by Rong Zheng@McMaster

“Tell me and I forget. Show me and I remember. Involve me and I understand.” Chinese proverb

© 2005-21012, 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 1.2Mb file 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.

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, 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 <http://www.cas.mcmaster.ca/~rzheng/course/wireshark/TCP/hallway.jpg>

¹ References to figures and sections are for the 6th edition of our text, *Computer Networks, A Top-down Approach, 6th ed.*, J.F. Kurose and K.W. Ross, Addison-Wesley/Pearson, 2012.

and retrieve a picture of ITB hallway. Store this file somewhere on your computer.

- Next go to

<http://www.cas.mcmaster.ca/~rzheng/course/wireshark/TCP/TCP-wireshark-file1.html>

- You should see a screen that looks like:

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

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 hallway

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 www.cas.mcmaster.ca. 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 hallway.jpg from your computer to www.cas.mcmaster.ca!

Upload Hallway Pic file

- Use the *Browse* button in this form to enter the name of the file (full path name) on your computer containing *hallway.jpg* (or do so manually). Don't yet press the "Upload Hallway pic" 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 Hallway pic" button to upload the file to the www.cas.mcmaster.ca 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.

The screenshot shows the Wireshark interface with a packet capture list and a detailed view of a selected packet. The packet list shows multiple TCP segments from source 130.113.68.10 to destination 130.113.70.196. The detailed view shows the following information:

- Frame 204: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
- Ethernet II, Src: Vmware_07:89:bd (00:50:56:07:89:bd), Dst: Apple_50:63:0f (08:20:66:50:63:0f)
- Internet Protocol Version 4, Src: 130.113.68.10, Dst: 130.113.70.196
 - 0100 = Version: 4
 - 0101 = Header Length: 20 bytes (5)
 - Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
 - Total Length: 52
 - Identification: 0x3e7f (15999)
 - Flags: 0x02 (Don't Fragment)
 - Fragment offset: 0
 - Time to live: 64
 - Protocol: TCP (6)
 - Header checksum: 0x6c94 [validation disabled]
 - [Header checksum status: Unverified]
 - Source: 130.113.68.10
 - Destination: 130.113.70.196
 - [Source GeoIP: Unknown]
 - [Destination GeoIP: Unknown]
- Transmission Control Protocol, Src Port: http (80), Dst Port: 62818 (62818), Seq: 1, Ack: 110648, Len: 0

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 && ip.addr == 130.113.68.10” (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 www.cas.mcmaster.ca. 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 www.cas.mcmaster.ca. 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 www.cas.mcmaster.ca to your computer.

Whenever possible, when answering a question, you should include a screenshot of the information of the packet(s) within the trace that you used to answer the question asked. Highlight relevant areas (e.g, using circles, arrows) of the screenshots 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 www.cas.mcmaster.ca? 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 www.cas.mcmaster.ca? 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:

No.	Time	Source	Destination	Protocol	Length	Info
168	4.197289	130.113.70.196	130.113.68.10	TCP	1514	62818->http(80) [ACK] Seq=94880 Ack=1 Win=131744 Len=1448 TSval=3438240043 TSecr=1217176013
169	4.197291	130.113.70.196	130.113.68.10	TCP	1514	62818->http(80) [ACK] Seq=96328 Ack=1 Win=131744 Len=1448 TSval=3438240043 TSecr=1217176013
170	4.197292	130.113.70.196	130.113.68.10	TCP	1354	62818->http(80) [PSH, ACK] Seq=97776 Ack=1 Win=131744 Len=1288 TSval=3438240043 TSecr=1217176013
171	4.197374	130.113.70.196	130.113.68.10	TCP	1514	62818->http(80) [ACK] Seq=99064 Ack=1 Win=131744 Len=1448 TSval=3438240043 TSecr=1217176013
172	4.197375	130.113.70.196	130.113.68.10	TCP	1514	62818->http(80) [ACK] Seq=100512 Ack=1 Win=131744 Len=1448 TSval=3438240043 TSecr=1217176013
173	4.197377	130.113.70.196	130.113.68.10	TCP	1514	62818->http(80) [ACK] Seq=101960 Ack=1 Win=131744 Len=1448 TSval=3438240043 TSecr=1217176013
174	4.197378	130.113.70.196	130.113.68.10	TCP	1514	62818->http(80) [ACK] Seq=103408 Ack=1 Win=131744 Len=1448 TSval=3438240043 TSecr=1217176013
175	4.197379	130.113.70.196	130.113.68.10	TCP	1514	62818->http(80) [ACK] Seq=104856 Ack=1 Win=131744 Len=1448 TSval=3438240043 TSecr=1217176013
176	4.197382	130.113.70.196	130.113.68.10	TCP	1514	62818->http(80) [ACK] Seq=106304 Ack=1 Win=131744 Len=1448 TSval=3438240043 TSecr=1217176013
177	4.197383	130.113.70.196	130.113.68.10	TCP	1514	62818->http(80) [ACK] Seq=107752 Ack=1 Win=131744 Len=1448 TSval=3438240043 TSecr=1217176013
178	4.197386	130.113.70.196	130.113.68.10	TCP	1514	62818->http(80) [ACK] Seq=109200 Ack=1 Win=131744 Len=1448 TSval=3438240043 TSecr=1217176013
179	4.197685	130.113.68.10	130.113.70.196	TCP	66	http(80)->62818 [ACK] Seq=1 Ack=76056 Win=67072 Len=0 TSval=1217176013 TSecr=3438240042
180	4.197688	130.113.68.10	130.113.70.196	TCP	66	http(80)->62818 [ACK] Seq=1 Ack=77504 Win=69888 Len=0 TSval=1217176013 TSecr=3438240042
181	4.197693	130.113.68.10	130.113.70.196	TCP	66	http(80)->62818 [ACK] Seq=1 Ack=78952 Win=71296 Len=0 TSval=1217176013 TSecr=3438240042
182	4.197777	130.113.70.196	130.113.68.10	TCP	1514	62818->http(80) [ACK] Seq=110648 Ack=1 Win=131744 Len=1448 TSval=3438240043 TSecr=1217176013
183	4.197780	130.113.70.196	130.113.68.10	TCP	1514	62818->http(80) [ACK] Seq=112096 Ack=1 Win=131744 Len=1448 TSval=3438240043 TSecr=1217176013
184	4.197789	130.113.70.196	130.113.68.10	TCP	1514	62818->http(80) [ACK] Seq=113544 Ack=1 Win=131744 Len=1448 TSval=3438240043 TSecr=1217176013
185	4.197791	130.113.70.196	130.113.68.10	TCP	1514	62818->http(80) [ACK] Seq=114992 Ack=1 Win=131744 Len=1448 TSval=3438240043 TSecr=1217176013
186	4.197799	130.113.70.196	130.113.68.10	TCP	1514	62818->http(80) [ACK] Seq=116440 Ack=1 Win=131744 Len=1448 TSval=3438240043 TSecr=1217176013
187	4.197801	130.113.70.196	130.113.68.10	TCP	1514	62818->http(80) [ACK] Seq=117888 Ack=1 Win=131744 Len=1448 TSval=3438240043 TSecr=1217176013
188	4.197890	130.113.68.10	130.113.70.196	TCP	66	http(80)->62818 [ACK] Seq=1 Ack=81848 Win=74112 Len=0 TSval=1217176013 TSecr=3438240042
189	4.197914	130.113.70.196	130.113.68.10	TCP	1514	62818->http(80) [ACK] Seq=119336 Ack=1 Win=131744 Len=1448 TSval=3438240043 TSecr=1217176013
190	4.197917	130.113.70.196	130.113.68.10	TCP	1514	62818->http(80) [ACK] Seq=120784 Ack=1 Win=131744 Len=1448 TSval=3438240043 TSecr=1217176013
191	4.197918	130.113.70.196	130.113.68.10	TCP	1514	62818->http(80) [ACK] Seq=122232 Ack=1 Win=131744 Len=1448 TSval=3438240043 TSecr=1217176013
192	4.197920	130.113.70.196	130.113.68.10	TCP	1514	62818->http(80) [ACK] Seq=123680 Ack=1 Win=131744 Len=1448 TSval=3438240043 TSecr=1217176013

▶ Frame 188: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
 ▶ Ethernet II, Src: Vmware_87:09:bd (00:50:56:87:09:bd), Dst: Apple_50:63:8f (a8:20:66:50:63:8f)
 ▼ Internet Protocol Version 4, Src: 130.113.68.10, Dst: 130.113.70.196
 0100 = Version: 4
 0101 = Header Length: 20 bytes (5)
 ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
 Total Length: 52

3. TCP Basics

Answer the following questions for the TCP segments:

- What is the actual sequence number of the TCP SYN segment that is used to initiate the TCP connection between the client computer and `www.cas.mcmaster.ca`? Which field in the segment identifies the segment as a SYN segment?
- What is the actual sequence number of the SYNACK segment sent by `www.cas.mcmaster.ca` to the client computer in reply to the SYN? What is the value of the Acknowledgement number field in the SYNACK segment? How did `www.cas.mcmaster.ca` determine that value? Which field in the segment identifies the segment as a SYNACK segment?
- 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.
- 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 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 in lecture slides 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

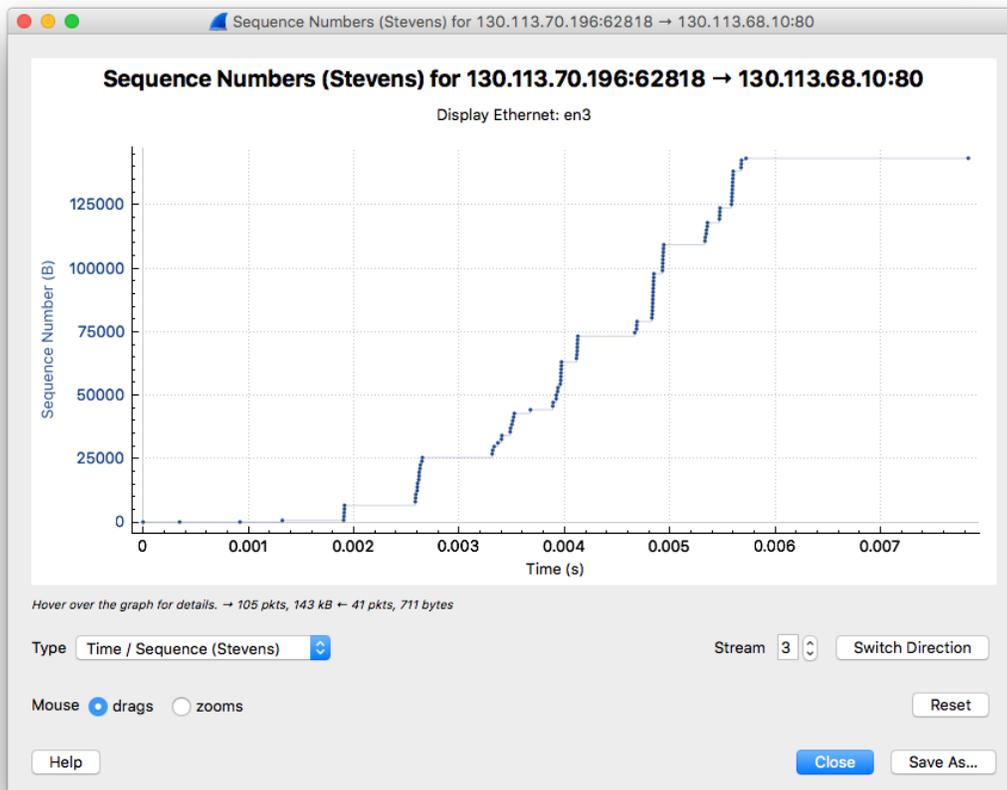
www.cas.mcmaster.ca server. Then select: *Statistics->TCP Stream Graph->Round Trip Time Graph*. **Make sure you choose the proper direction (by toggling “switch direction”)**

7. What is the segment length of each of the first six TCP segments from the client?
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?
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 (i.e., delayed ACK).
11. What is the throughput (bytes transferred per unit time) for the TCP connection? Explain how you arrived at 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 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.

Answer the following questions for the trace that you have gathered when you transferred a file from your computer to `www.cas.mcmaster.ca`

1. Use the -> *Time-Sequence-Graph(Stevens)* plotting tool to view the growth of sequence numbers for the TCP connection from the client to the `www.cas.mcmaster.ca` 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.

Submission

Include your answers and screen shots if any in a single PDF file named `YOUR_MAC_ID_A3.pdf`. Submission should be done through Avenue.