

# Wireshark Lab 4b: TCP

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

## 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. 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://gaia.cs.umass.edu/wiresharklabs/alice.txt> 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 as shown in Fig1.
- 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 as shown Fig2.

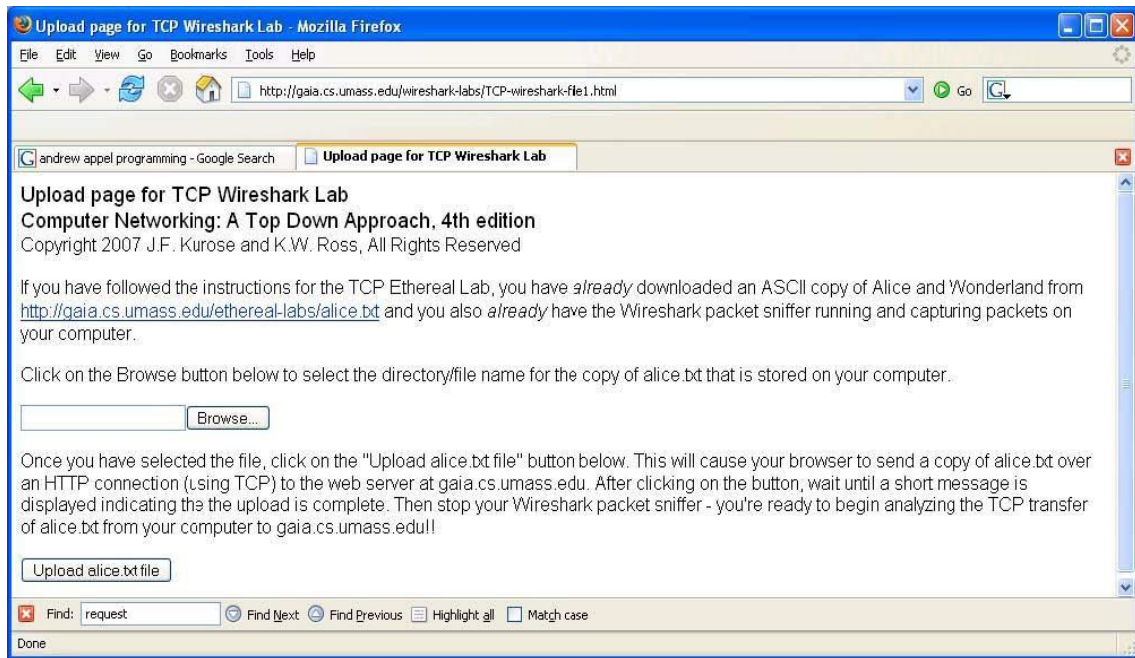


Fig 1 Upload page for TCP

No.	Time	Source	Destination	Protocol	Length	Info
67	1.484708	128.119.245.12	10.129.6.93	TCP	60	80 → 17646 [ACK] Seq=1 Ack=80767 Win=179584 Len=0
68	1.484709	128.119.245.12	10.129.6.93	TCP	60	80 → 17646 [ACK] Seq=1 Ack=87667 Win=179584 Len=0
69	1.484709	128.119.245.12	10.129.6.93	TCP	60	80 → 17646 [ACK] Seq=1 Ack=89047 Win=183296 Len=0
70	1.484709	128.119.245.12	10.129.6.93	TCP	60	80 → 17646 [ACK] Seq=1 Ack=91807 Win=181632 Len=0
71	1.484751	10.129.6.93	128.119.245.12	HTTP	199...	POST /wireshark-labs/lab3-1-reply.htm HTTP/1.1 (text/plain)
72	1.573367	128.119.245.12	10.129.6.93	TCP	60	80 → 17646 [ACK] Seq=1 Ack=93187 Win=183296 Len=0
73	1.573369	128.119.245.12	10.129.6.93	TCP	60	80 → 17646 [ACK] Seq=1 Ack=97327 Win=191488 Len=0
74	1.573559	128.119.245.12	10.129.6.93	TCP	60	80 → 17646 [ACK] Seq=1 Ack=102847 Win=202624 Len=0

> Frame 71: 19900 bytes on wire (159200 bits), 19900 bytes captured (159200 bits) on interface 0

> Ethernet II, Src: Dell\_23:f8:21 (c8:f7:50:23:f8:21), Dst: Cisco\_9f:f4:5e (00:00:0c:9f:f4:5e)

> Internet Protocol Version 4, Src: 10.129.6.93, Dst: 128.119.245.12

> Transmission Control Protocol, Src Port: 17646, Dst Port: 80, Seq: 133207, Ack: 1, Len: 19846

> [11 Reassembled TCP Segments (153052 bytes): #30(726), #31(12420), #44(1380), #48(24840), #50(2760), #52(5520), #56(41400), #58(2760), #60(15326)]

> Hypertext Transfer Protocol

> POST /wireshark-labs/lab3-1-reply.htm HTTP/1.1\r\n

Host: gaia.cs.umass.edu\r\n

Connection: keep-alive\r\n

Content-Length: 152326\r\n

Cache-Control: max-age=0\r\n

<

00000030 48 f7 73 74 3a 20 67 61 69 61 2e 63 73 2e 75 6d Host: ga ia.cs.um  
00000040 61 73 73 2e 65 64 75 0d 0a 43 6f 6e 6e 65 63 74 ass.edu ·Connect  
00000050 69 6f 6e 3a 20 6b 65 65 70 2d 61 6c 69 76 65 0d ion: kee p-alive·  
00000060 0a 43 6f 6e 74 65 6e 74 2d 4c 65 6e 67 74 68 3a ·Content ·Length:  
00000070 20 31 35 32 33 32 36 0d 0a 43 61 63 68 65 2d 43 152326· ·Cache-C  
00000080 6f 6e 74 72 6f 6c 3a 20 6d 61 78 2d 61 67 65 3d ontrol: max-age=  
00000090 30 0d 0a 55 70 67 72 61 64 65 2d 49 6e 73 65 63 0·Upgra de-Insec

Fig 2 Captured TCP page

If you are unable to run Wireshark on a live network connection, you use the given TCP packet tracer file (TCP\_Trace.pcap).

## 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. 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, by opening the Wireshark captured packet file **TCP\_Trace.pcap**. 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 a Wireshark window that looks like as shown in Fig 3.

No.	Time	Source	Destination	Protocol	Length	Info
65	1.484708	128.119.245.12	10.129.6.93	TCP	60	80 → 17646 [ACK] Seq=1 Ack=66967 Win=163456 Len=0
66	1.484708	128.119.245.12	10.129.6.93	TCP	60	80 → 17646 [ACK] Seq=1 Ack=73867 Win=177280 Len=0
67	1.484708	128.119.245.12	10.129.6.93	TCP	60	80 → 17646 [ACK] Seq=1 Ack=80767 Win=179584 Len=0
68	1.484709	128.119.245.12	10.129.6.93	TCP	60	80 → 17646 [ACK] Seq=1 Ack=87667 Win=179584 Len=0
69	1.484709	128.119.245.12	10.129.6.93	TCP	60	80 → 17646 [ACK] Seq=1 Ack=89047 Win=183296 Len=0
70	1.484709	128.119.245.12	10.129.6.93	TCP	60	80 → 17646 [ACK] Seq=1 Ack=91807 Win=181632 Len=0
71	1.484751	10.129.6.93	128.119.245.12	TCP	199...	17646 → 80 [PSH, ACK] Seq=133207 Ack=1 Win=262144 Len=19846
72	1.573367	128.119.245.12	10.129.6.93	TCP	60	80 → 17646 [ACK] Seq=1 Ack=93187 Win=183296 Len=0

```

> Frame 71: 19900 bytes on wire (159200 bits), 19900 bytes captured (159200 bits) on interface 0
> Ethernet II, Src: Dell_23:f8:21 (c8:f7:50:23:f8:21), Dst: Cisco_9f:f4:5e (00:00:0c:9f:f4:5e)
> Internet Protocol Version 4, Src: 10.129.6.93, Dst: 128.119.245.12
▼ Transmission Control Protocol, Src Port: 17646, Dst Port: 80, Seq: 133207, Ack: 1, Len: 19846
  Source Port: 17646
  Destination Port: 80
  [Stream index: 1]
  [TCP Segment Len: 19846]
  Sequence number: 133207 (relative sequence number)
  [Next sequence number: 153053 (relative sequence number)]
  Acknowledgment number: 1 (relative ack number)
  0101 - Window Length: 20 bytes (E)
0000 00 00 0c 9f f4 5e c8 f7 50 23 f8 21 08 00 45 00 .....P# [...]E
0010 00 00 67 c2 40 00 80 06 00 00 0a 81 06 5d 80 77 ..g@... ..]w
0020 f5 0c 44 ee 00 50 29 94 07 eb 87 ea 02 a6 50 18 ..D·P)· .....P·
0030 04 00 86 68 00 00 20 20 20 20 54 68 65 20 4b 6e ...h... The Kn
0040 61 76 65 20 6f 66 20 48 65 61 72 74 73 2c 20 68 ave of H earts, h
0050 65 20 73 74 6f 6c 65 20 74 68 6f 73 65 20 74 61 e stole those ta
0060 72 74 73 2c 0d 0a 20 20 20 20 20 20 20 20 20 20 rts,..
0070 41 6e 64 20 74 6f 6f 6b 20 74 68 65 6d 20 71 75 And took them qu

```

Fig 3 TCP captured page

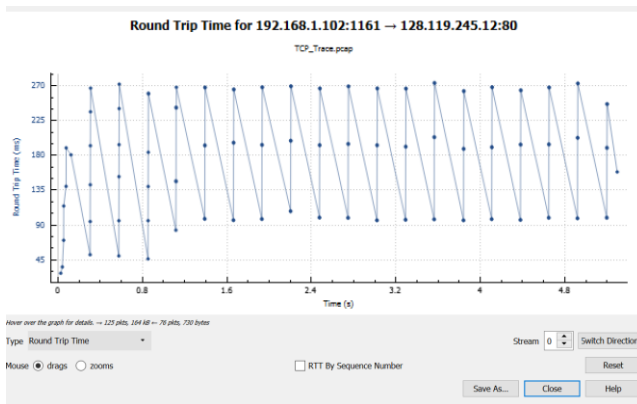
This is what we’re looking for -a series of TCP segments sent between your computer and gaia.cs.umass.edu. We will use the packet trace that you have captured (and/or the packet trace **TCP\_Trace.pcap**) to study TCP behavior in the rest of this lab.

### 3. TCP Basics

Answer the following questions for the TCP segments:

- 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?
- 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?
- 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 (see below) 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.

*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.*



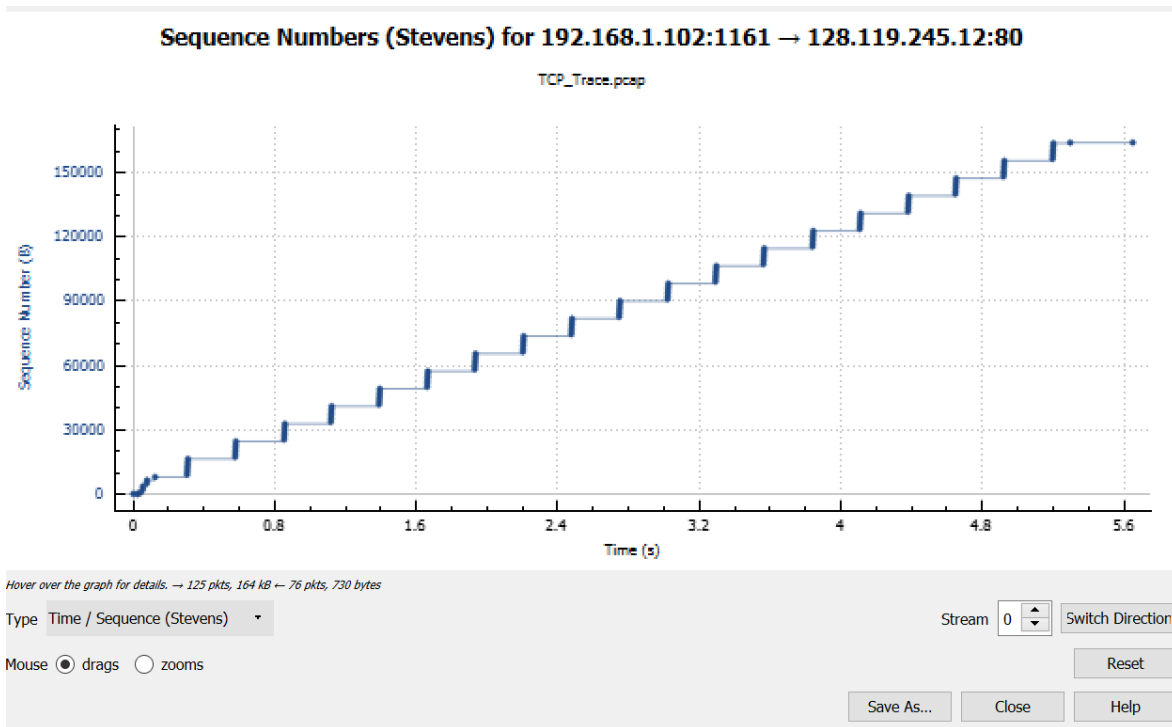
(Note:  $\text{EstimatedRTT} = 0.875 * \text{EstimatedRTT} + 0.125 * \text{SampleRTT}$ )

8. What is the length of each of the first six TCP segments?
9. 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?
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?
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 Wireshark window, we'll use one of Wireshark's TCP graphing utilities -*Time-Sequence-Graph (Stevens)* -to plot out data.

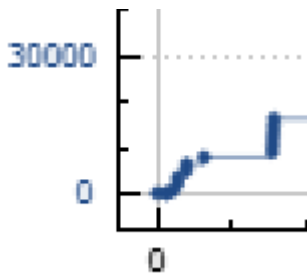
- Select a TCP segment in the Wireshark's "listing of captured-packets" window. Then select the menu: *Statistics->TCP Stream Graph-> Time-Sequence-Graph (Stevens)*. You should see a plot that looks similar to the following plot, which was created from the captured packets in the packet trace **TCP\_Trace.pcap**:



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.

Additional question: Answer the following questions for the TCP segments the packet **TCP\_Trace.pcap**.

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 slow start phase begins and ends?



Note: This lab is based on Kurose Book:

James F. Kurose and J. Rose, Computer Networks: A Top-Down Approach Featuring the Internet, 8<sup>th</sup> Edition (2016), ADDISON WESLEY, ISBN-13: 9780135928523.