CS 315: Computer Networks Lab
Spring 2022-23, IIT Dharwad
Assignment-5

Wireshark Lab: TCP

January 31, 2023

Lab Instructions

- Please leave your bags on the Iron shelf near the SP16 entrance.
- Login to the Ubuntu OS on your machine. The login credentials are as follows:
  - Username: user
  - Password: 123456
- Mark your attendance in the attendance sheet before leaving the lab.
- Handle the lab resources with utmost care.
- Please go through the following exercises in today's lab.
- It is recommended that you complete all the following exercises during the lab slot itself.
- If you face any difficulties, please feel free to seek help online or from your peers or TAs.
- After finishing all exercises, please carry your solutions with you (via email/pen drive) for future reference, and delete the files from the desktop.

Introduction

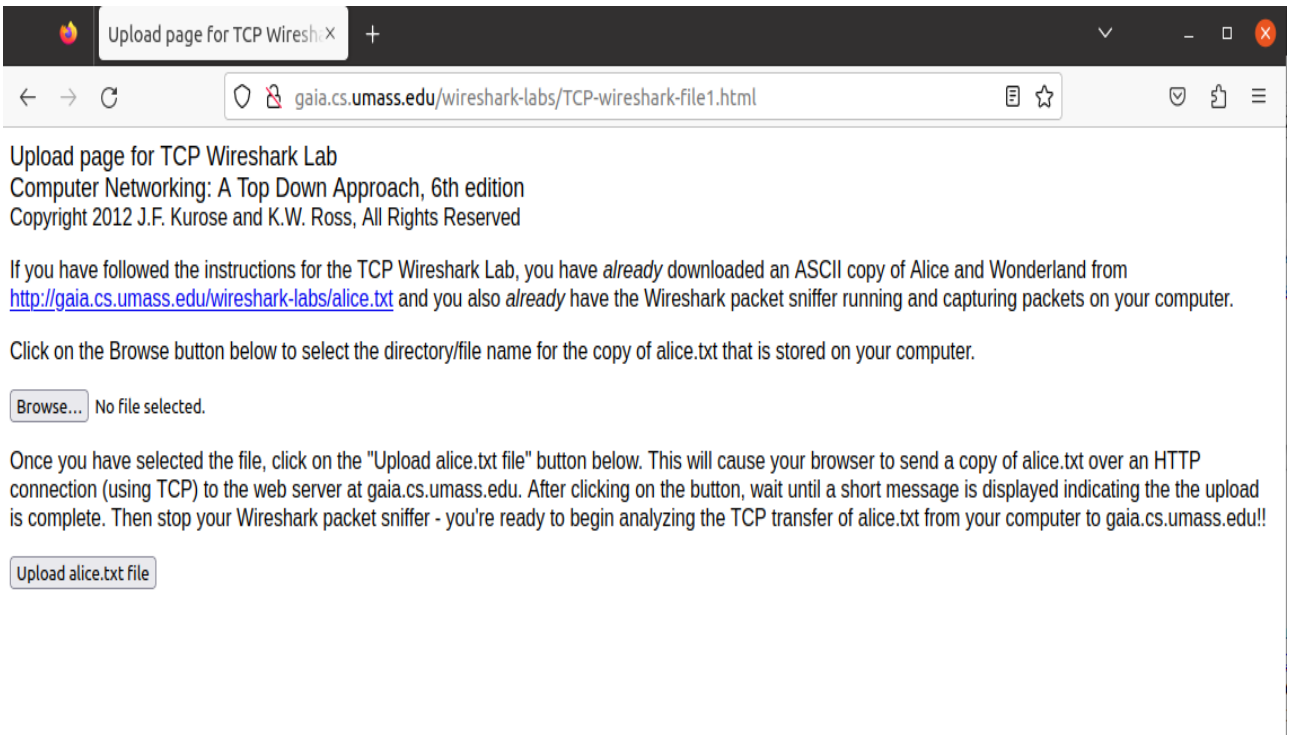
In this lab, we'll investigate the behavior of the celebrated TCP protocol in detail. We'll do so by analyzing a trace of the TCP segments sent and received in transferring a 150 KB file (containing the text of Lewis Carrol's *Alice's Adventures in Wonderland*) from your computer to a remote server. 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.

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

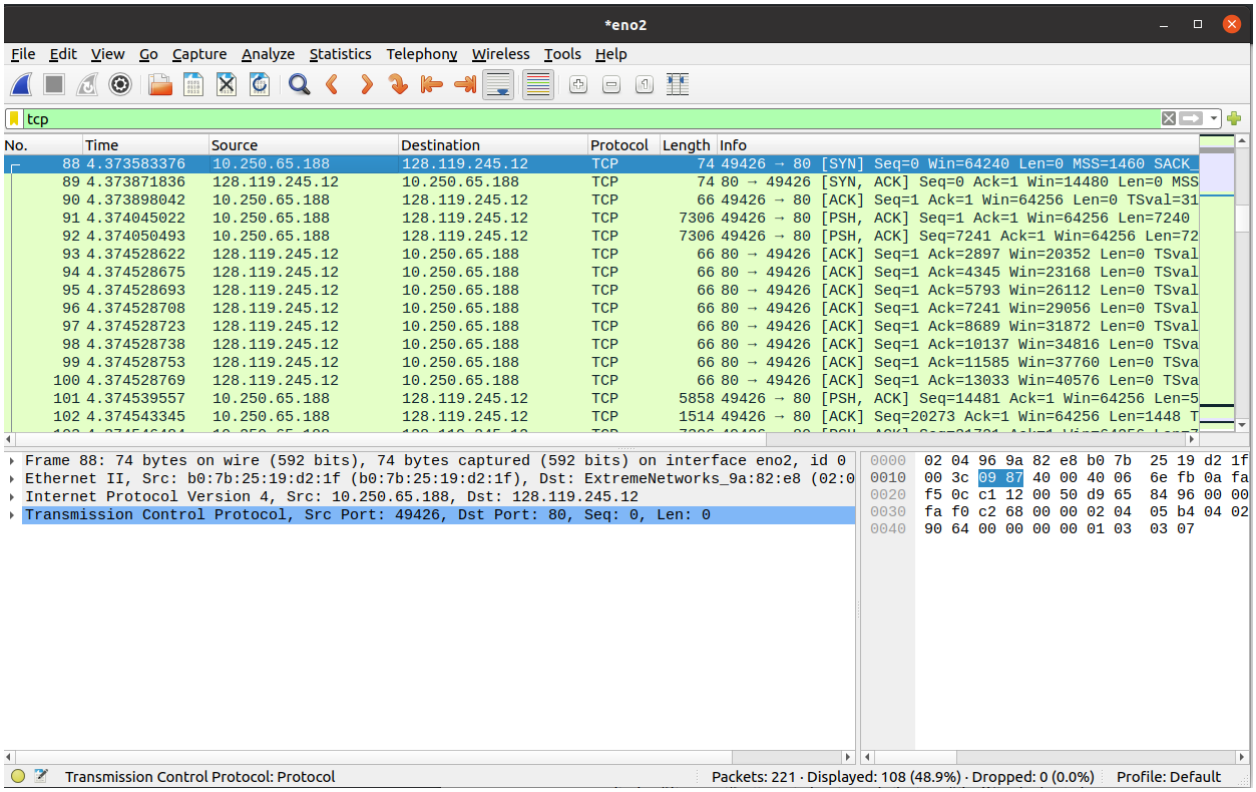
Before beginning our exploration of TCP, we'll need to use Wireshark to obtain a packet trace of the TCP transfer of a file from your computer to a remote server. You'll do so by accessing a Web page that will allow you to enter the name of a file stored on your computer (which contains the ASCII text 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/wireshark-labs/alice.txt and retrieve an ASCII copy of *Alice in Wonderland*. Store this as a .txt 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 the Figure below.



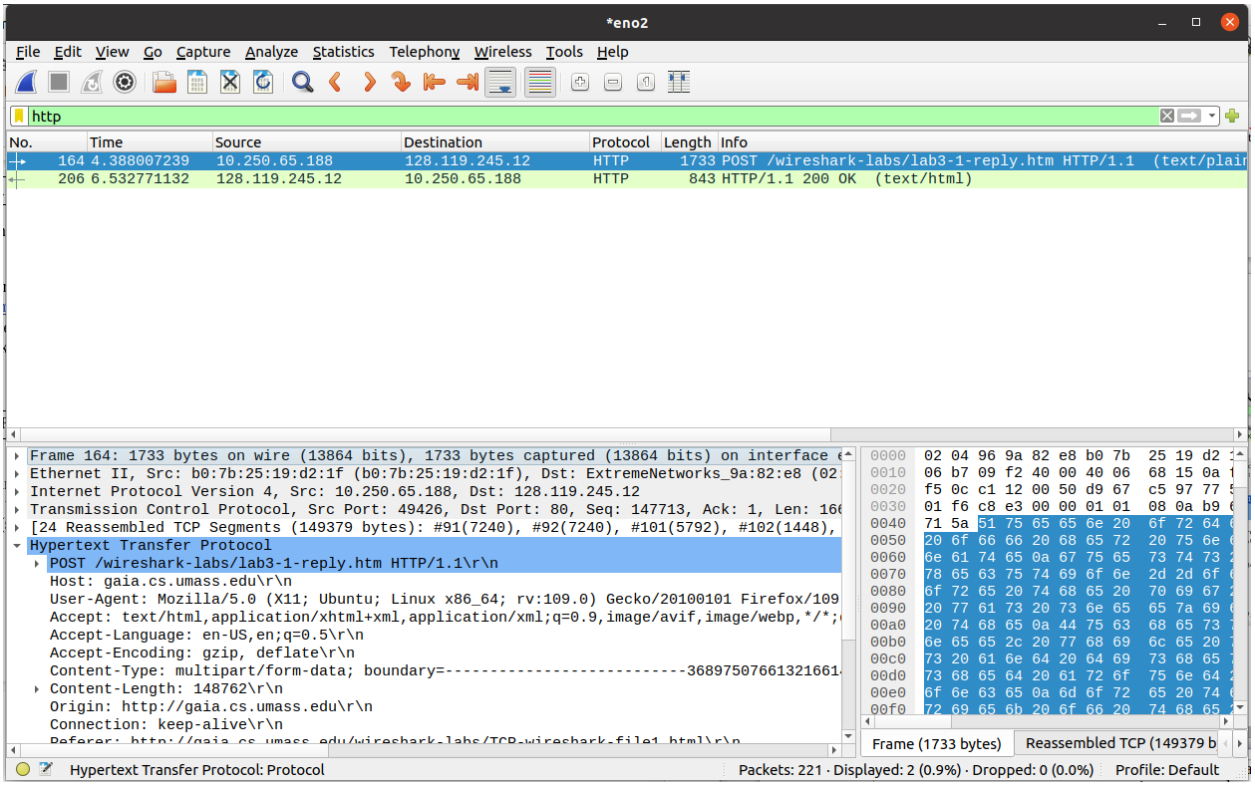
- Use the ***Browse*** button in this form to the file on your computer that you just created containing *Alice in Wonderland*. Don't press the "*Upload alice.txt file*" button yet.
- Now start up Wireshark and begin packet capture (see the earlier Wireshark labs if you need a refresher on how to do this).
- 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 in the Figure below.



Part 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.

Let's start by looking at the HTTP POST message that uploaded the alice.txt file to gaia.cs.umass.edu from your computer. Find that file in your Wireshark trace, and expand the HTTP message so we can take a look at the HTTP POST message more carefully. Your Wireshark screen should look something like the Figure below.

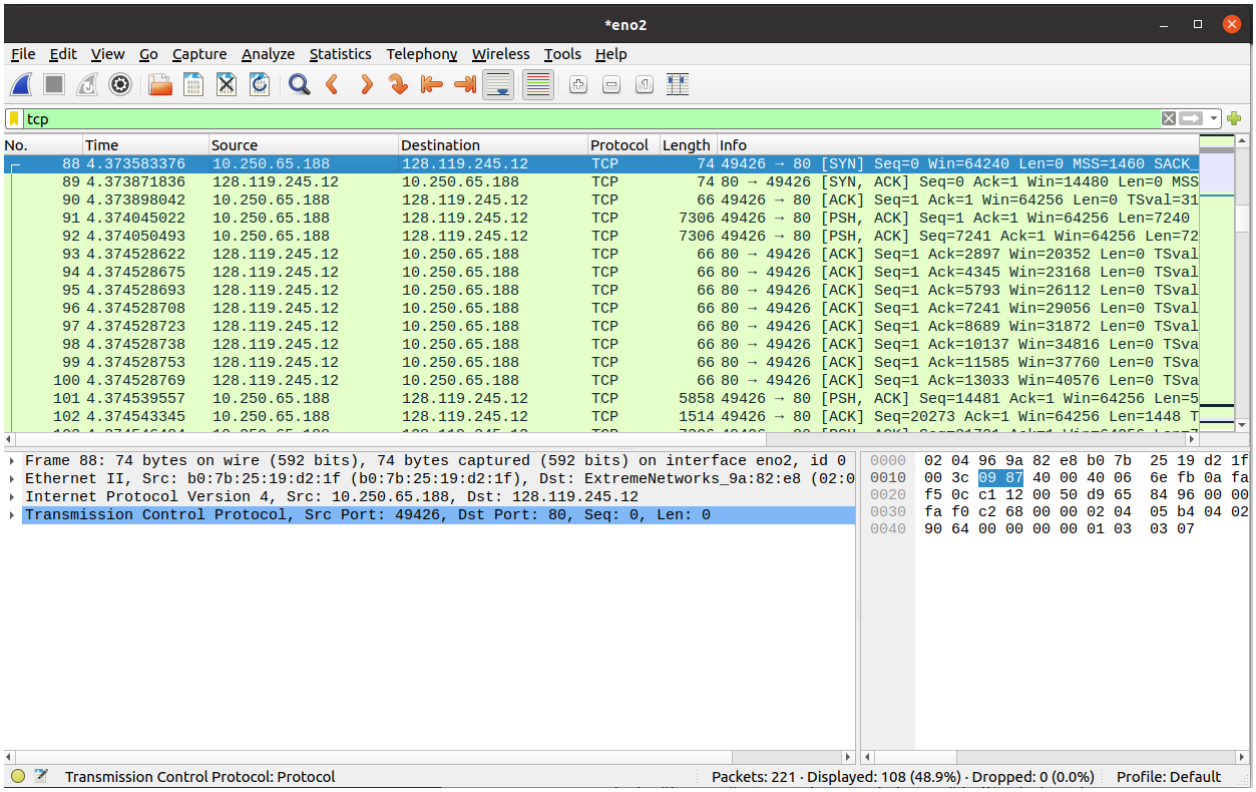


There are a few things to note here:

- The body of your application-layer HTTP POST message contains the contents of the file alice.txt, which is a large file of more than 148K bytes. OK – it's not *that* large, but it's going to be too large for this one HTTP POST message to be contained in just one TCP segment!
- In fact, as shown in the Wireshark window in the Figure above we see that the HTTP POST message was spread across 24 TCP segments.

Let's now "get our hands dirty" by looking at some TCP segments.

- 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. Your Wireshark display should look something like the Figure below.



Answer the following questions:

1. What is the IP address and TCP port number used by the client computer (source) that is transferring the alice.txt 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"
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?

Part 3: TCP Basics

Answer the following questions for the TCP segments:

1. 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 this TCP segment that identifies the segment as a SYN segment?
2. 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 it in the segment that identifies the segment as a SYNACK segment? What is the value of the Acknowledgement field in the SYNACK segment? How did gaia.cs.umass.edu determine that value?
3. What is the sequence number of the TCP segment containing the header of the HTTP POST command? Note that in order to find the POST message header, you'll need to dig

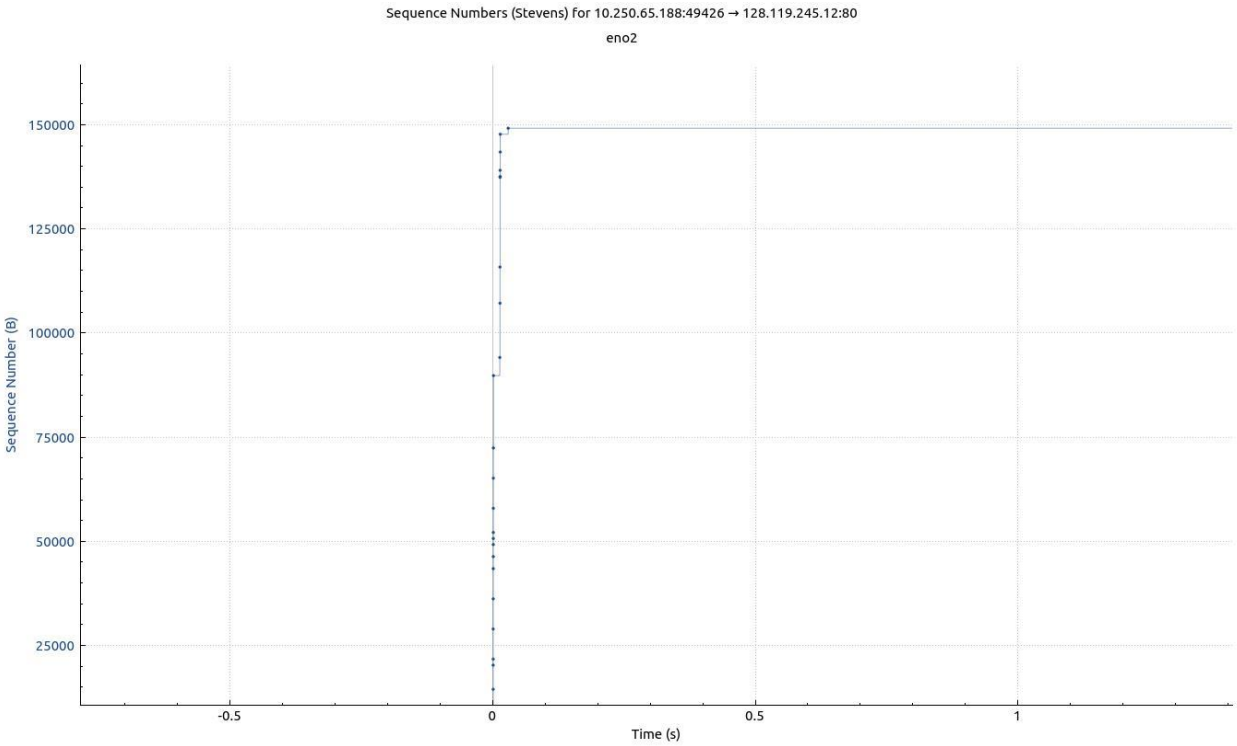
into the packet content field at the bottom of the Wireshark window, *looking for a segment with the ASCII text "POST" within its DATA field*. How many bytes of data are contained in the payload (data) field of this TCP segment? Did all of the data in the transferred file alice.txt fit into this single segment?

4. Consider the TCP segment containing the HTTP "POST" as the first segment in the data transfer part of the TCP connection.
    i. At what time was the first segment (the one containing the HTTP POST) in the data-transfer part of the TCP connection sent?
    ii. At what time was the ACK for this first data-containing segment received?
    iii. What is the RTT for this first data-containing segment?
    iv. What is the RTT value of the second data-carrying TCP segment and its ACK?
5. What is the length (header plus payload) of each of the first four data-carrying TCP segments?
6. What is the minimum amount of available buffer space advertised to the client by gaia.cs.umass.edu among these first four data-carrying TCP segments? Does the lack of receiver buffer space ever throttle the sender for these first four data-carrying segments?
7. Are there any retransmitted segments in the trace file? What did you check for (in the trace) in order to answer this question?
8. How much data does the receiver typically acknowledge in an ACK among the first ten data-carrying segments sent from the client to gaia.cs.umass.edu? Can you identify cases where the receiver is ACKing every other received segment among these first ten data-carrying segments?
9. What is the throughput (bytes transferred per unit time) for the TCP connection? Explain how you calculated this value.

Part 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 client-sent TCP segment in the Wireshark's "listing of captured-packets" window corresponding to the transfer of alice.txt from the client to gaia.cs.umass.edu. Then select the menu: *Statistics->TCP Stream Graph-> Time-Sequence-Graph(Stevens*). You should see a plot that looks similar to the plot in the Figure below,



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 (sometimes called a "fleet" of packets) that were sent back-to-back by the sender.

Answer the following questions:

1. 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, and where congestion avoidance takes over?

Submission Details

- Write your answers in a single doc/tex file, and submit its PDF named after your IIT Dharwad roll number, which contains all answers (with screenshots, if necessary).