Lab 2: Transport and Network Layer

Objective

In this lab, you will continue to use Wireshark, but now you will explore the transport and network layers. You will examine various UDP, TCP and ICMP transmissions. Write a report, to show you have executed the lab procedures. In this report, also answer any questions that are interleaved among the procedures. Feel free to also include questions, ponderings, and any interesting stuff you observed.

Step 1: UDP and DNS

Lets start by examining a few UDP segments. As you recall from the lecture and from Section 3.3 of the text, UDP is a streamlined, no-frills transport protocol. All state information is conveyed in each individual UDP segment. In Lab 1, we used dig to generate DNS traffic with the intent of examining the DNS protocol. In this lab, we will use dig to generate DNS traffic, but with the intent of examining the UDP protocol.

Procedures

1. Open Wireshark and set up our privacy filter so that you display only DNS traffic to or from your computer (Filter: **`dns && ip.addr==<your IP address>`**).
2. Use dig to generate a DNS query to lookup the domain name "codeschool.com." Then, stop the capture.
3. Before you take a look at the packets in Wireshark, think for a minute about what you expect to see as the UDP segment headers. What can you reasonably predict, and what could you figure out if you had some time and a calculator handy? Use your knowledge of UDP (see Chapter 3.3 if you need a refresher) to inform your predictions. [2 points, here and in the next question]
4. Take a look at the query packet on Wireshark. You'll see a bunch of bytes (70-75 bytes) listed as the actual packet contents in the bottom Wireshark window. The bytes at offsets up to number 33-34 are generated by the lower level protocols. If you click on the "User Datagram Protocol" line in the packet details window, you'll see the UDP contents get highlighted in the packet contents window. You will also see Wireshark interpret the header contents. Match up the bytes in the packet contents window with each field of the UDP header. Were your predictions correct?

5. Continue to examine the DNS request packet. Which fields does the UDP checksum cover? Wireshark probably shows the UDP checksum as "Unverified" (and the IP checksum as "Validation Disabled." Why is that (you'll have to do some Reading of The Fine Manual or testing your Google-fu)? [5 points]
6. Save your capture file. Restrict the range of saved packets to only those in the DNS query.

Step 2: TCP

Now, let's take a look at another transport protocol, TCP. We will use HTTP to invoke the sort of TCP behaviors we want to study -- I trust that you understand HTTP well enough by now.

Procedures

7. Download and save a copy of Geoffrey Chaucer's *Canterbury Tales and Other Poems* from the Project Gutenberg website[1]. Grab the Plain Text UTF-8 version:

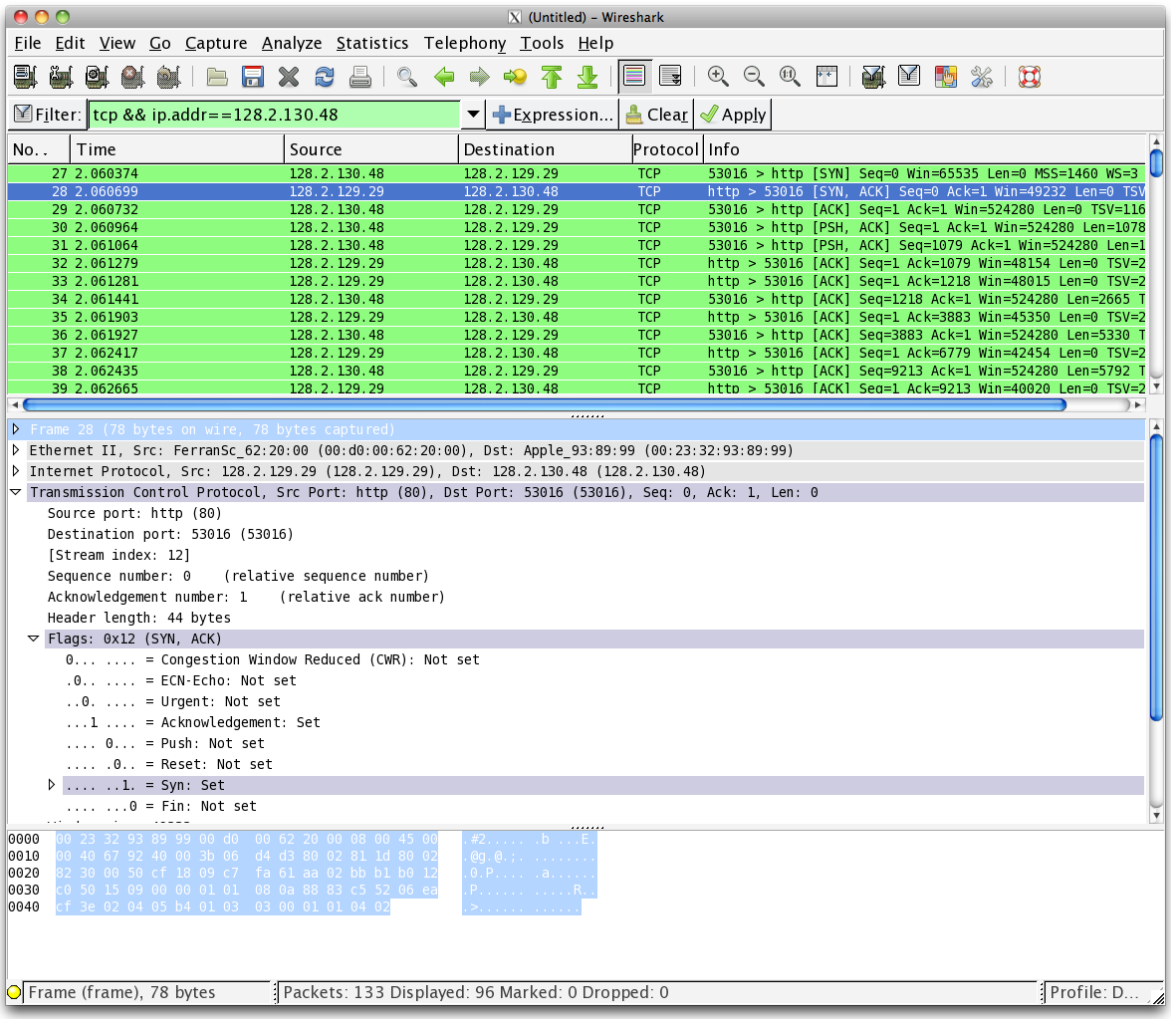
   http://www.gutenberg.org/ebooks/2383.txt.utf-8
8. Clear out Wireshark and start a new capture.
9. Go to the following website. When there, use the form to choose a file (the copy of the *Canterbury Tales* that you've stashed away somewhere on your hard drive) and upload the file. The point of this exercise is to capture a lengthy TCP stream which originates at your computer.

   http://www.ini740.rocks/Lab2/lab2a.html
10. Stop the Wireshark capture.
11. Let's take a look at what you captured. First, filter the results to look for TCP packets and to only look at those going to and from your computer with the filter "**`tcp && ip.addr == <your IP address>`**." If you have other services running on your computer, you might want to further filter so you only display TCP packets between your computer and the ECE webserver. What you should see is a series of TCP and HTTP messages between your computer and www.ece.cmu.edu. You should see the initial three-way handshake containing a SYN message. You should see an HTTP POST message and you may see a series of "HTTP Continuation" messages being sent from your computer to the server. Recall from our discussion in lab 1, that there 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. You should also see TCP ACK segments being returned from the server to your computer. Take a screenshot showing the three-way handshake. [8 points]

[1] You might need to do this twice. If you get the "Welcome Stranger" message about how many books Project Gutenberg supplies, rather than the text of this book, simply reload your browser window.

12. What is the IP address and TCP port number used by your computer (client) to transfer the file? What is the IP address of the server? On what port number is it sending and receiving TCP segments for this transfer of the file? [4 points]

13. 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 an Wireshark window that looks like:



This is what we're looking for - a series of TCP segments sent between your computer and www.ece.cmu.edu. We will use the packet trace that you have captured to study TCP behavior in the rest of this lab.

Step 2b: TCP Basics

Answer the following questions for the TCP segments:

14. What is the sequence number of the TCP SYN segment that is used to initiate the TCP connection? What element of the segment identifies it as a SYN segment?

Wireshark uses relative sequence numbers by default. Can you obtain absolute sequence numbers instead? How? You can use relative sequence numbers to answer the remaining questions. [4 points]

15. What is the sequence number of the SYNACK segment sent by the server in reply to the SYN? What is the value of the ACKnowledgement field in the SYNACK segment? How did the server determine that value? What element in the segment identifies it as a SYNACK segment? [4 points]

16. 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. [2 points]

17. Consider the TCP segment containing the HTTP POST as the first segment in the non-overhead part of the TCP connection. For the segments which follow, put together a table with one row per segment (and columns for whatever data you think is useful) until you have enough segments to calculate four SampleRTT values according to the RTT estimation techniques discussed in class. Calculate what those SampleRTT values are, as well as the EstimatedRTT after each Sample is collected. Discuss this calculation, including what your initial EstimatedRTT was, your choice of parameters, and any segments that weren't used in the calculation. [12 points]

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 server. Then select: Statistics → TCP Stream Graph → Round Trip Time Graph.

18. 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? [3 points]

19. Are there any retransmitted segments? What did you check for (in the trace) in order to answer this question? [2 points]

20. How much data does the receiver typically acknowledge in an ACK? Can you identify cases where the receiver is *delayed ACKing* segments. Explain how or why not. [2 points]

21. What is the throughput (bytes transferred per unit time) for the TCP connection? Explain how you calculated this value. [2 points]

Step 2c: Statistics

Wireshark has some fairly robust reporting abilities, most of which are accessed via the Statistics menu. Spend a few minutes messing around with the options on that menu, trying to figure out what each report is telling you. Then, answer the following questions about the Canterbury Tales capture:

22. What is the most common TCP packet length range? What is the second most common TCP packet length range? Why is the ratio of TCP packets of length < 40

bytes equal to zero? Describe what actions you took to get answers to these questions from Wireshark. [4 points]

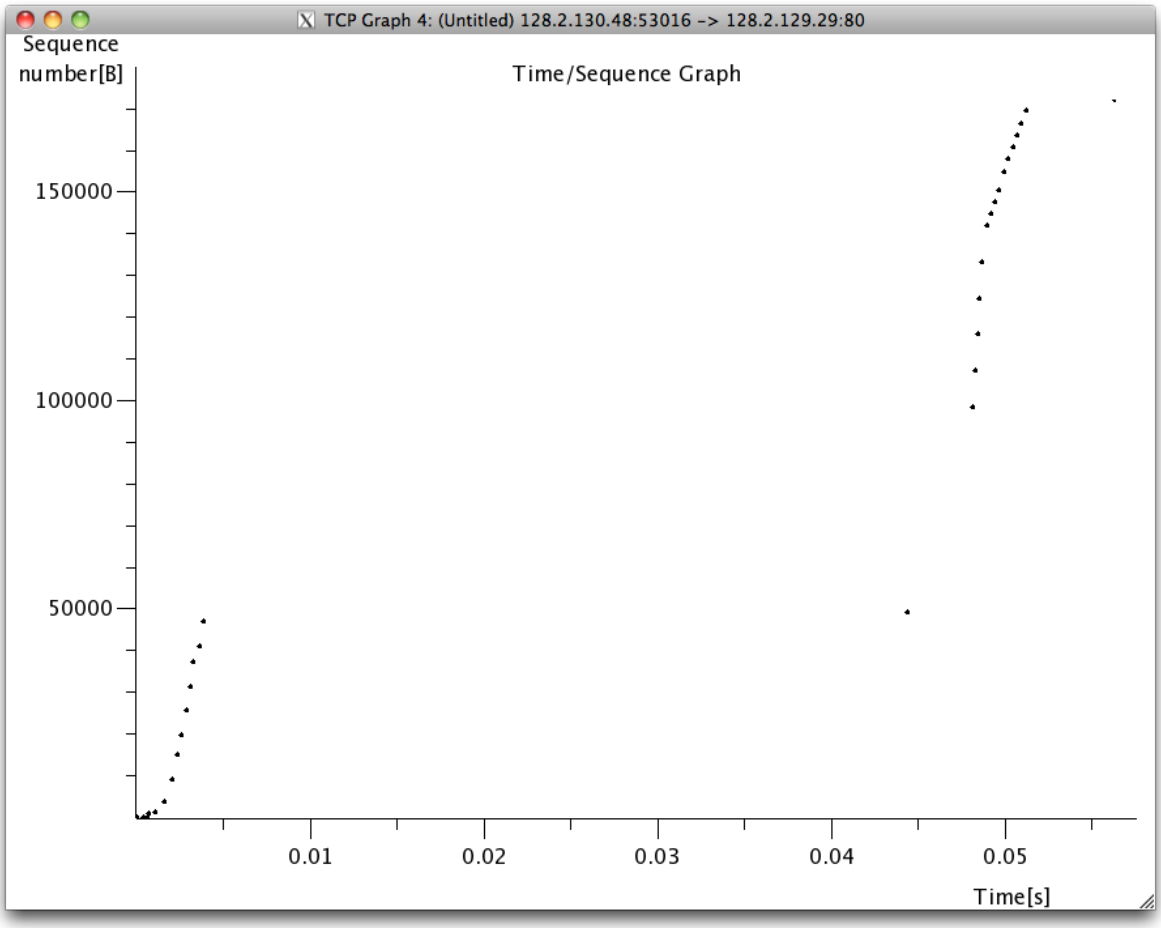
23. What average throughput did you use in Mbps? How many packets were captured in the packet capture session? How many bytes in total? Explain your methods. [4 points]

24. A conversation represents a traffic between two hosts. With which remote host did your local host converse the most (in bytes)? How many packets were sent from your host? How many packets were sent from the remote host? [4 points]

Step 3: Congestion Control

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 our data.

25. 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 (though the individual plotted values may differ quite a bit).



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. Don't be distraught if your graph doesn't look like that shown above. Recall that the particular algorithms for managing congestion control can be implemented (or not) based on the OS you are running.

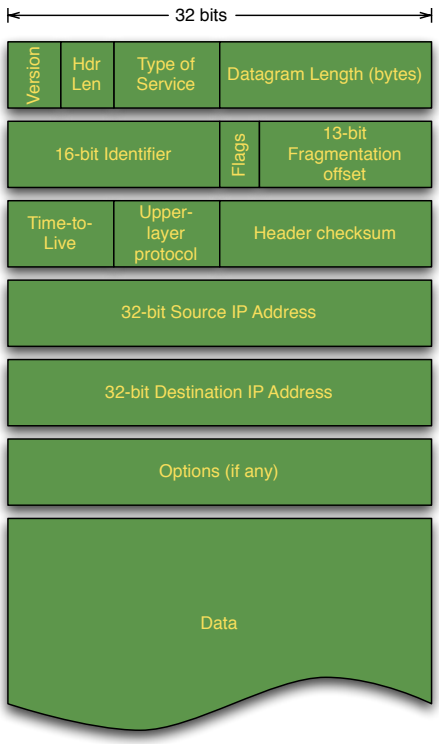
26. Use the Time-Sequence-Graph(Stevens) plotting tool to view the sequence number versus time plot of segments being sent. Can you identify where TCP's slowstart phase begins and ends, and where congestion avoidance takes over? Comment on ways in which the measured data differs from the idealized behavior of TCP that we've studied in the text. Make sure to include a copy of the plot in your report. [10 points]

Step 4: The Network Layer

Let's take this opportunity to check out a bit of IP traffic. We don't have to capture any additional traffic, as everything we've seen today is carried over IP packets.

27. Load the capture file that you saved in step 1. Recall that this was a simple DNS query, carried in a UDP packet.

28. Take a look at the IP section of the DNS query (the packet that was generated when you used dig to request the address of codeschool.com). Match up the header fields with the format we discussed in class (don't just look through Wireshark's display -- instead, match the raw bytes with the pictures we saw in lecture, which I've copied on the right). [4 points]



29. Most of the fields should match up and make perfect sense. Verify the Datagram Length, Upper-layer protocol and the IP address fields. [1 point]

30. Are there any interesting features of the data in the identifier/flags/offset fields? [2 points]

31. In class, we discussed the TTL field and determined that we didn't know a good way to set this. What does your OS set this field to? BTW, please document in this question what your OS and OS version are. [2 points]

Step 5: ICMP

The Network Layer uses ICMP to send information about the network. Some would say that ICMP is a higher-layer protocol, as the actual ICMP packet is carried inside an IP packet. Let's take a look at how that works.

32. Start a new capture, with the display filter showing only packets sent to or from your computer (i.e. "**`ip.addr==<ip address>`**")

33. In a terminal window, execute the traceroute utility to trace from your computer to **`www.cmuj.jp`** or **`www.regjeringen.no`** or some other far-away destination (like you did for Homework #1). If you are having trouble with the weird traceroutes, try this from a non-campus location (your home, a coffeshop, etc). Do whatever you can to get a traceroute consisting of about a dozen steps. [2 points]

34. Stop the capture and take a look at what you found.

35. What are the transmitted segments like? Describe the important features of the segments you observe. In particular, examine the destination port field. What characteristics do you observe about this port number and why would it be chosen so? [5 points]

36. What about the return packets? What are the values of the various header fields? [3 points]

37. The ICMP packets carry some interesting data. What is it? Can you show the relationship to the sent packets? [3 points]

38. HW#1 asserted that ping operates in a similar fashion to traceroute. Use Wireshark to show the degree to which this is true. What differences and similarities are there between the network traffic of ping versus traceroute. [5 points]

Finishing up

Deposit your report (in PDF format) in Gradescope. You've spent quite a bit of time working on this report, so ensure it is submitted properly before the deadline or it will not be graded.