Wireshark Lab for ECE374

Posted: 03/02/15

Due: 03/09/15

UDP and TCP

1. UDP

Let's begin with UDP. Since UDP is a streamlined, no-thrills protocol - simple and sweet – this part of this lab is quick and simple. Start capturing packets in Wireshark and then do something that will cause your host to send and receive several UDP segments (hint: you might want to recall that DNS messages are carried inside UDP segments, but you can do anything you want to ensure that there are UDP segments in your captured trace). 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, and select the minimum amount of packet detail that you need to answer the question.

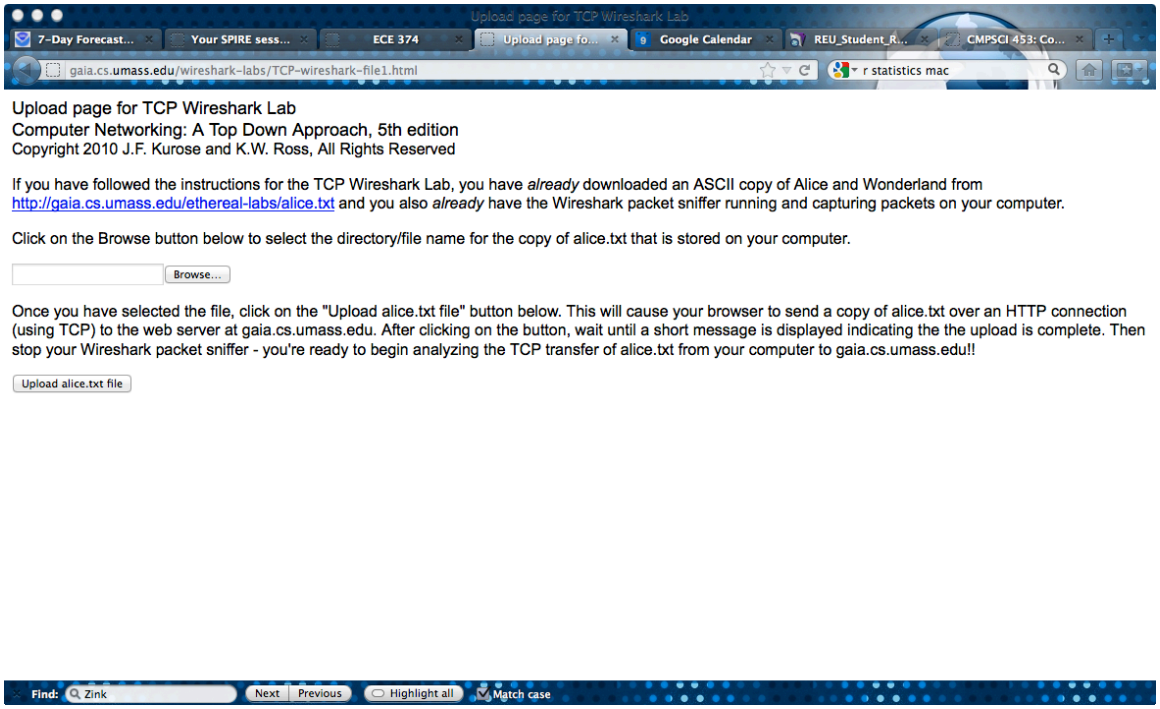
a. Select one packet. From this packet, determine how many fields there are in the UDP header. (Do not look in the textbook! Answer these questions directly from what you observe in the packet trace.) Name these fields as they are named in the Wireshark display of segment fields.
b. What are the source and destination port numbers, in both decimal and hexadecimal format. (Hint: the hexadecimal format is given in the data in the bottommost panel in the wireshark display, and so it's easier just to read it out from there rather than converting the decimal number to hex).
c. What is the value in the Length field in both decimal and hexadecimal format. What is the meaning of this value (i.e., this value is the length of what?)
d. What is the protocol number for UDP? Give your answer in both hexadecimal and decimal notation. (To answer this question, you'll need to look into the IP header.)
e. Examine a pair of UDP packets in which the first packet is sent by your host and the second packet is a reply to the first packet. Describe the relationship between the port numbers in the two packets.

2. 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 (see Chapter 2). 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 to http://gaia.cs.umass.edu/wireshark-labs/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:



- 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. (BTW,

Ethereal was the predecessor of Wireshark!)
- Stop Wireshark packet capture.

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 segments 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 inside a TCP segment and a series of tcp segments being sent from your computer to gaia.cs.umass.edu, as well as ACKs being returned from gaia.cs.umass.edu.

Answer the following questions for the TCP segments:
1. Print out a captured packet and indicate where you see the information that answers the following:
    a. What is the IP address and TCP port number used by your client computer (source) to transfer the file to gaia.cs.umass.edu?
    b. What is the IP address and TCP port number used by gaia.cs.umass.edu?
2. Print out a captured packet and indicate where you see the information that answers the following:
    a. 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?
    b. What is it in the segment that identifies the segment as a SYN segment?

3. Print out a captured packet and indicate where you see the information that answers the following:
    a. What is the sequence number of the SYNACK segment sent by gaia.cs.umass.edu to the client computer in reply to the SYN?
    b. What is the value of the ACKnowledgement field in the SYNACK segment?
    c. How did gaia.cs.umass.edu determine that value?
    d. What is it in the segment that identifies the segment as a SYNACK segment?
4. Print out a captured packet and indicate where you see the information that answers the following: 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.
    a. 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?

b. When was the ACK for each segment received?
c. Do you see evidence of the use of cumulative ACKs in your trace? Explain.
d. 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?
e. What is the length of each of the first six TCP segments?
f. 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?
g. Are there any retransmitted segments in the trace file? What did you check for (in the trace) in order to answer this question? How much data does the receiver typically acknowledge in an ACK? Can you identify cases where the receiver is ACKing every other received segment? Explain.