

**Homework 2: HTTP, TCP, and Wireshark**  
**CSE 534, Fall 2021**  
**Instructor: Aruna Balasubramanian**  
**Due date: October 21 2021, 9.00pm**

The last few weeks we have been talking about HTTP, TCP, and Congestion Control. The goal of this assignment is to dissect the HTTP and TCP protocols using the Wireshark tool.

To do this, you should be familiar with the packet formats, PCAP files, TCPDump, and Wireshark. Briefly, TCPdump/Wireshark are both tools to capture packets going on the wire. When you exchange packets between A and B, Wireshark/TCPDump will let you capture the packets that are being sent out and received at A (or at B, depending on where you run the tool). PCAP is the file format used to store the captured packets. A PCAP library is available to parse the files.

**Part A Wireshark Programming Task (45 points)**

Your task is to write a program `analysis_pcap_tcp` that analyzes a Wireshark/TCPdump trace to characterize the TCP flows in the trace. A TCP flow starts with a TCP "SYN" and ends at a TCP "FIN".

You may use a PCAP library to analyze the trace. But you may only use the PCAP library to get each packet in byte format. The PCAP library can be used to figure out where the first packet ends and the second packet starts. You need to then write code to analyze the bytes to get the information about the packet.

[Hint: You can create your own packet structures and read the bytes into the structure. However, you cannot convert the PCAP file into text and perform the analysis. Similarly, you cannot use existing libraries to directly parse the TCP headers. This is important because the main goal of this homework is to learn how to parse network packets.]

Attached to the homework is a file `assignment2.pcap`. In this file, we have captured packets sent between 130.245.145.12 and 128.208.2.198. Node 130.245.145.12 establishes the connection (let's call it sender) with 128.208.2.198 (let's call it receiver) and then sends data. The trace was captured at the sender. Use your `analysis_pcap_tcp` code to analyze `assignment2.pcap` and answer the following questions (Ignore any traffic that is not TCP). Each of these needs to be done empirically:

1. Count the number of TCP flows initiated from the sender
2. For each TCP flow
  - (a) For the first 2 transactions after the TCP connection is set up (from sender to receiver), get the values of the Sequence number, Ack number, and Receive Window size. Explain these values.
  - (b) Compute the throughput at the receiver. You can make assumptions on what you want to include as part of the throughput estimation.
  - (c) Compute the loss rate for each flow. Loss rate is the number of packets not received divided by the number of packets sent. Loss rate is an application layer metric. So think about what makes sense when defining loss rate.

(d) Estimate the average RTT. Now compare your empirical throughput from (b) and the theoretical throughput (estimated using the formula derived in class). Explain your comparison.

Submit (i) the high level view of the `analysis_pcap_tcp` code, (ii) the `analysis_pcap_tcp` program, (iii) instructions on how to run and set up your code, and (iv) the answers to each question and a brief note about how you estimated each value

### **Part B Congestion control (15 points)**

Using the same `assignment2.pcap` file and your `analysis_pcap_tcp` program, answer the following questions about congestion control

For each TCP flow:

(1) Print the first ten congestion window sizes (or till the end of the flow, if there are less than five congestion windows). You need to decide whether the congestion window should be estimated at the sender or the receiver and explain your choice. Mention the size of the initial congestion window. You need to estimate the congestion window size empirically since the information is not available in the packet. Comment on how the congestion window size grows. Remember that your estimation may not be perfect, but that is ok. Congestion window sizes are typically estimated per RTT.

(2) Compute the number of times a retransmission occurred due to triple duplicate ack and the number of time a retransmission occurred due to timeout (as before, determine if you need to do it at the sender or the receiver).

Submit (i) the answers to each question and a brief note about how you estimated each value, (ii) the program if any you used to answer the two questions (including how to run the program. We need to be able to run your program without requiring any special software).

### **Part C HTTP Analysis task (30 points)**

You will now extend your tool made in part A to analyze various aspects of HTTP from TCP packets. Extend your previous program in Part A to create a new program called "`analysis_pcap_http`" to analyze HTTP in addition to TCP. This is similar to Part A in that you can use the `pcap` libraries to get the beginning and end of the packet, but cannot use the `pcap` libraries to completely decode the packet.

Your first task is to use *tcpdump*, a popular tool for capturing TCP/IP network packets. Connect to our server at <http://www.sbunetsyslabs.com> at port 1080 from your web browser and use *tcpdump* to capture the packets. Save the packet as `http_1080.pcap` for analysis. Remember that in this case, your browser's IP address is the client and <http://www.sbunetsyslabs.com> is the server. The client establishes the connection. The client requests data from the server, and server sends the data.

Do the same to capture the traffic over HTTP encrypted by TLS from <https://www.sbunetsyslabs.com> on ports 1081 and 1082 (name these files tcp\_1081.pcap and tcp\_1082.pcap respectively). Make sure to clear your browser cache after each run, so the resources are actually fetched over the network. Each port represents the same site delivered using a different version of HTTP (HTTP 1.0, HTTP 1.1, HTTP 2.0).

1. Reassemble each unique HTTP Request/Response for http\_1080.pcap (the other two are encrypted, so you will not be able to reassemble easily). The output of this part should be the Packet type (request or response) and the unique <source, dest, seq, ack> TCP tuple for all the TCP segments that contain data for that request.
2. Identify which HTTP protocol is being used for each PCAP file. Note that two of the sites are encrypted so you must use your knowledge of HTTP and TCP to programmatically solve this question. Include the logic behind your code in the write-up.
3. Finally, after you've labeled the PCAPs with their appropriate versions of HTTP, answer the following: Which version of the protocol did the site load the fastest under? The Slowest? Which sent the most number of packets and raw bytes? Which protocol sent the least? Report your results and write a brief explanation for your observations.

Submit the (i) tcpdump command you used (or what filters you used in Wireshark) and the high level view of the analysis\_pcap\_http code, (ii) the analysis\_pcap\_http code itself, (iii) answer to each question along with a brief description of how you estimated the answers to the question, (iv) any program you write to answer C.1, C.2, and C.3.

#### **Part D Fairness (10 points)**

Using the same technique discussed in class (transport-congestion-control.pdf, slide 31) explain why (1) Multiplicative Increase Additive Decrease, (2) Multiplicative Increase Multiplicative Decrease, and (3) Additive Increase, Additive Decrease, are not fair. Use a figure in each case, similar to slide 6 in lecture to illustrate your answer.

### **Submission instruction**

As before, you may write your programs in the following languages: Python and C/C++. If you want to write in any other language, please talk to me. Note that viewing these traces on Wireshark is helpful but may not always be always accurate. This is because Wireshark may sometimes parse HTTP/2 packets incorrectly.

You need to submit your homework in a single zip file as follows:

- The zip file and (the root folder inside) should be named using your last name, first name, and the homework name, all separated by a dash ('-') e.g. lastname-firstname-HW2.zip
- The zip file should contain all submissions requested in Parts A through D. Have a separate subfolder called Part A, Part B, Part C, and Part D in your zip file corresponding to each part.
- For each part that requires running a code, the entire code should be in a single file. Jupyter notebooks, Google Colaboratory, or any other Github repository will not be accepted. We

should require no special software to run your code. You will lose considerable portion of your grade otherwise

- You should provide a README.txt file describing how to run your programs in each part when applicable.
- Please include any requirements file if needed

Some example pcap libraries that you can use:

C/C++ - libpcap

Python - dpkt