

## Project 2: Transport Layer and ARQ Protocols Analysis with Wireshark

---

Baran Berkay Hökelek, 0060673

December 9, 2020

### Description of the Project

This project is a demonstration of our knowledge in TCP, Transport Layer and Wireshark. We were asked to perform some experiments regarding the Transport Layer of the Internet Protocol Suite using Wireshark and answer some questions.

### Part 1

The description of the project requirements for Part 1 was a mess and I didn't understand anything. After scratching my head for 2 days straight, I decided that it was not worth the effort and moved on.

### Part 2

- 5) There are 2 significant IP addresses that are used: 192.168.1.15 and 128.119.245.12. The former belongs to me, while the latter belongs to the address "gaia.cs.umass.edu", the URL which I uploaded the text to. Below are the screenshots showing the three-way and terminating handshakes.

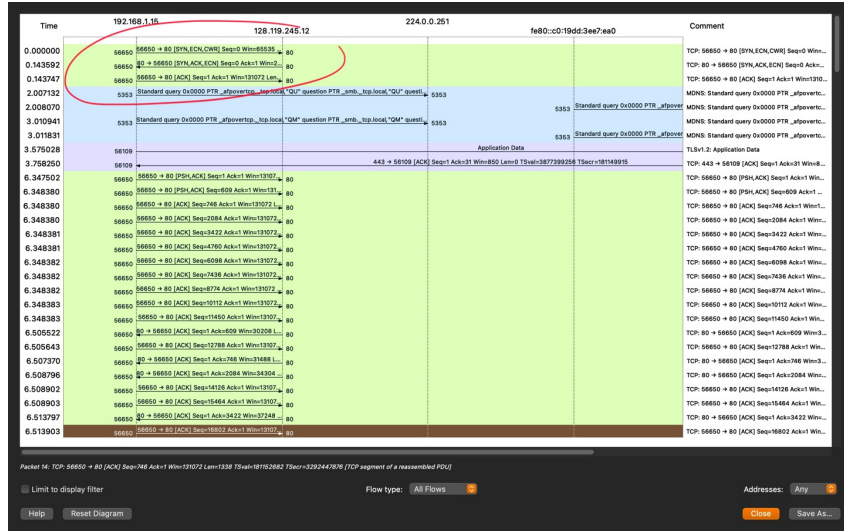


Figure 1: Screenshot of the flow graph, with the three-way handshake circled in red.

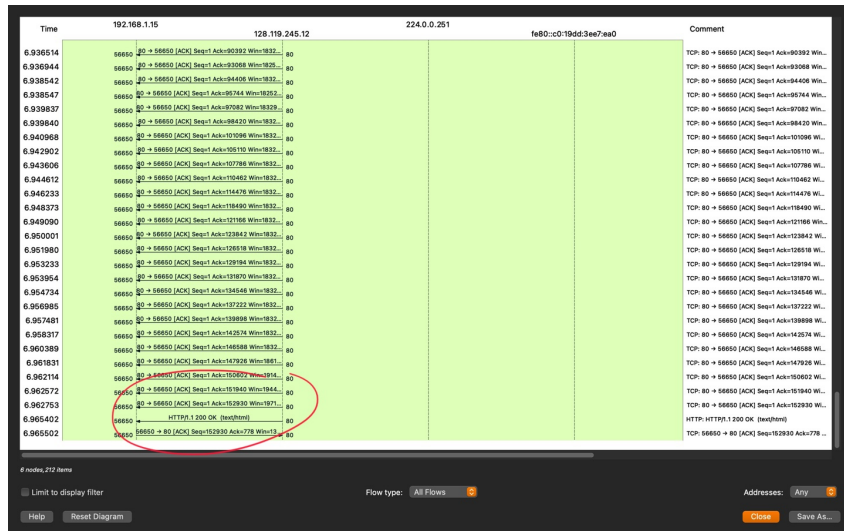


Figure 2: Screenshot of the flow graph, with the terminating handshake circled in red.

- 6) The sequence numbers of the segments used for the 3-way handshake protocol are: 2397470178 for SYN, 3676586368 for SYN-ACK, and 2397470179 for ACK. The port number for the client is 56650, and the port number for the server is 80.
- 7) The relative sequence number is 0, the raw number is 3676586368.
- 8) The relative acknowledgement value is 1, the raw value is 2397470179. This value is determined by adding 1 to the sequence number determined by the SYN segment, as required by the protocol. This segment can be identified as a SYNACK segment because the SYN and ACK flags are turned on.
- 9) The sequence number of the TCP segment containing the HTTP POST command is 2397470179.
- 10) The first 6 segments in the TCP connection have the sequence numbers 2397470179, 2397470787, 2397470924, 2397472262, 2397473600 and 2397474938 respectively. The packets were sent at times 6.347502s, 6.348380s, 6.348380s, 6.348380s, 6.348381s and 6.348381s; and the acknowledgements came at times 6.505522s, 6.507370s, 6.508796s, 6.513797s, 6.517838s and 6.519821s. So, the RTT values are 158.02ms, 158.99ms, 160.42ms, 165.42ms, 169.46ms and 171.44ms. The estimated RTT according to the formula is 161.93ms.

## Part 3

- 11) `dns.qry.name == www.duke.edu`

No.	Time	Source	Destination	Protocol	Length	ACK/RTT	Info
421	1.462836	192.168.1.15	192.168.1.1	DNS	72		Standard query 0x967f A www.duke.edu
426	3.631285	192.168.1.1	192.168.1.15	DNS	118		Standard query response 0x967f A www.duke.edu CNAME duke.edu A 152.3.72.197

Figure 3: The filter that I used, and the results.

- 12) nslookup works on the DNS protocol on the application layer, and on UDP on the transport layer.

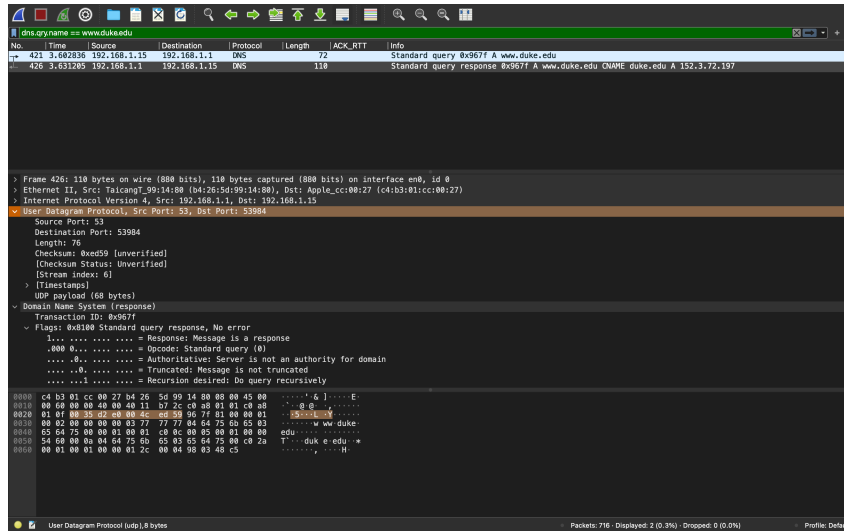


Figure 4: The protocol stack for the queries. UDP & DNS are clearly visible.

- 13) It turns out the DNS server that I'm connected to is unable to do recursive queries. I can find that out by looking at the query response packet and see whether the "Recursion Available" flag is 1 or 0.

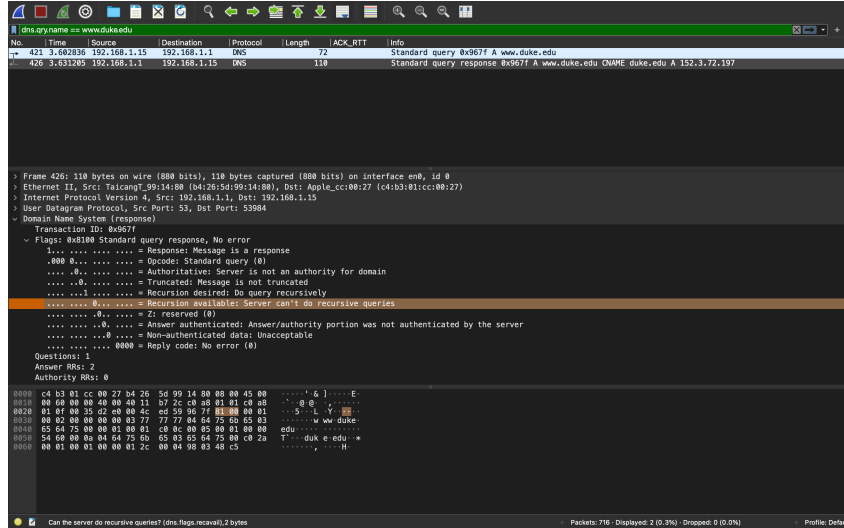
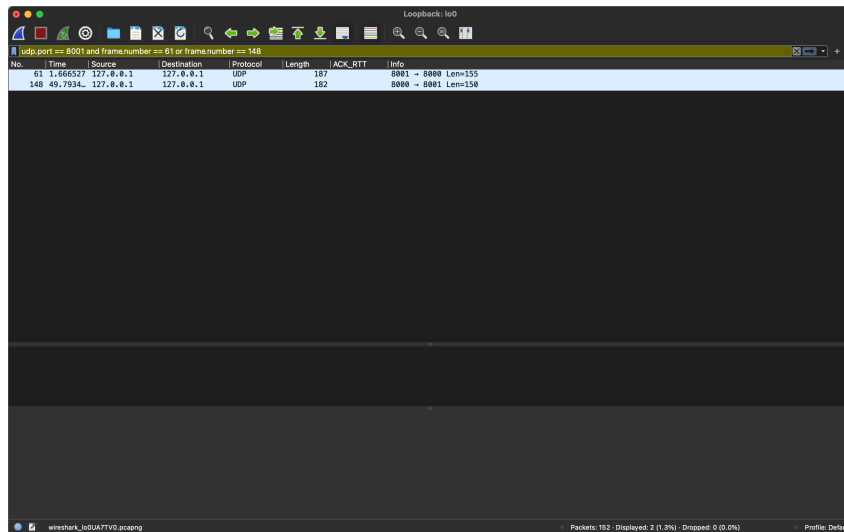


Figure 5: The "Recursion Available" flag.

- 14) UDP Layer had a header length of 8 bytes, and DNS has a header length of 12 bytes.  
 15) A UDP segment has only a single checksum.

## Part 4

- 16) There are a total of 50 packets that are sent from the sender. Considering that the message is made up of 32 packets, there have been 28 re-transmissions.
- 17) This result, though unnecessarily inefficient, makes sense. The reason why is that my timeout value for ReceiveWithARQ was too high (otherwise it got stuck on an infinite loop for some reason), and there were too many retransmissions (almost the same as the number of original transmissions) which added to the time cost.



The image shows a Wireshark packet capture window. The filter bar at the top is set to 'udp port == 8001 and frame.number == 61 or frame.number == 148'. The packet list shows two packets:

No.	Time	Source	Destination	Protocol	Length	ACK_RTT	Info
61	1.666527	127.0.0.1	127.0.0.1	UDP	187		8001 → 8000 Len=155
148	49.79342	127.0.0.1	127.0.0.1	UDP	182		8000 → 8001 Len=158

The status bar at the bottom indicates 'Packets: 152 · Displayed: 2 (1.3%) · Dropped: 0 (0.0%)'.

Figure 6: The first and last packet transmitted between the client and the server. Notice the 48 seconds of time difference!