Shaun Vulaj

SER 321

Assignment 1.2

01/27/2023

==========

Task 1.1 (Explore the Data Link Layer with ARP)

==========

Task 1.1(Step 1)

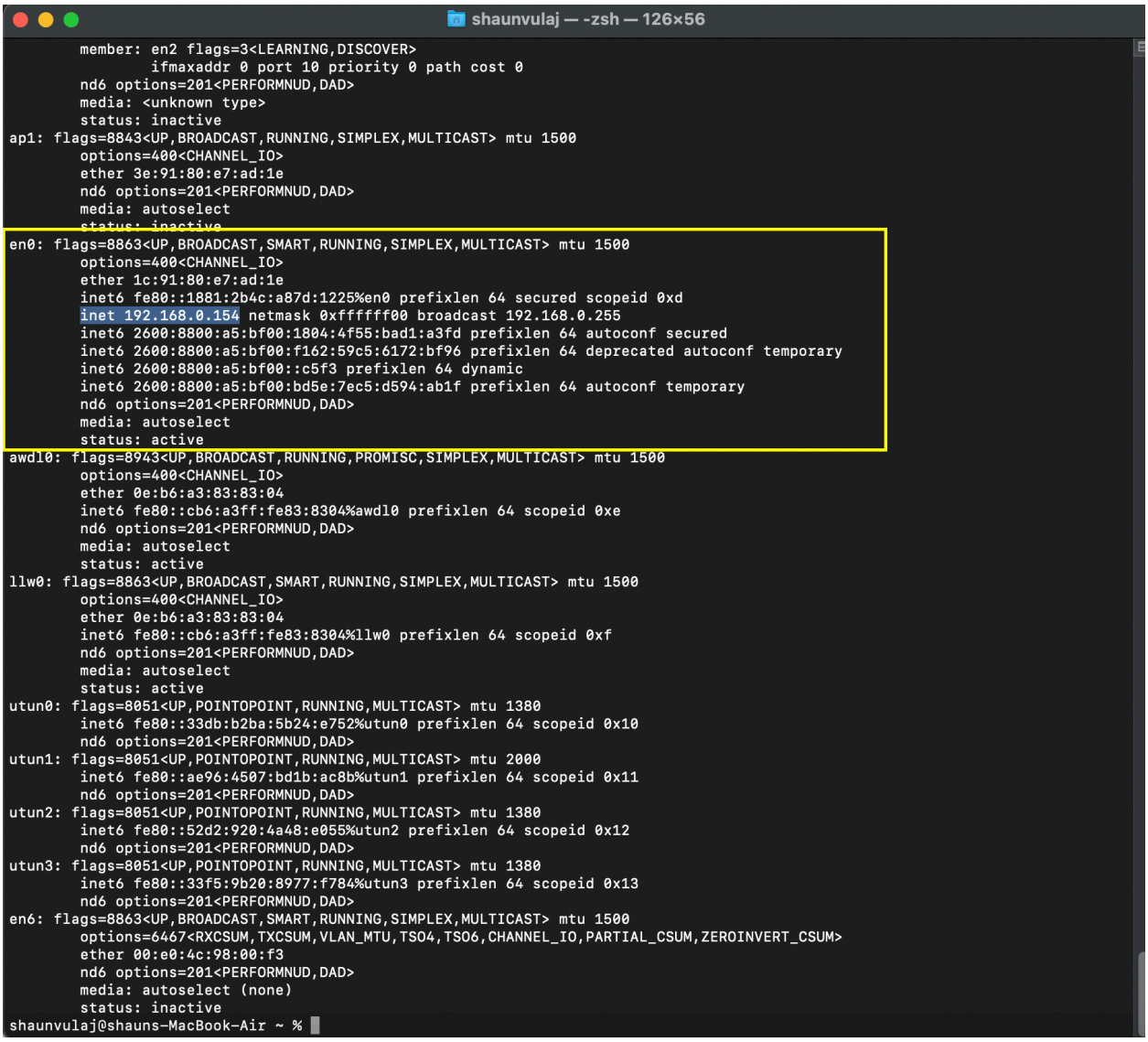


Figure 1 This image is for Task 1-1. You cannot see my command but I used "ifconfig" and then used "ipconfig getifaddr en0" to confirm.*

Task 1.1(Step 1)

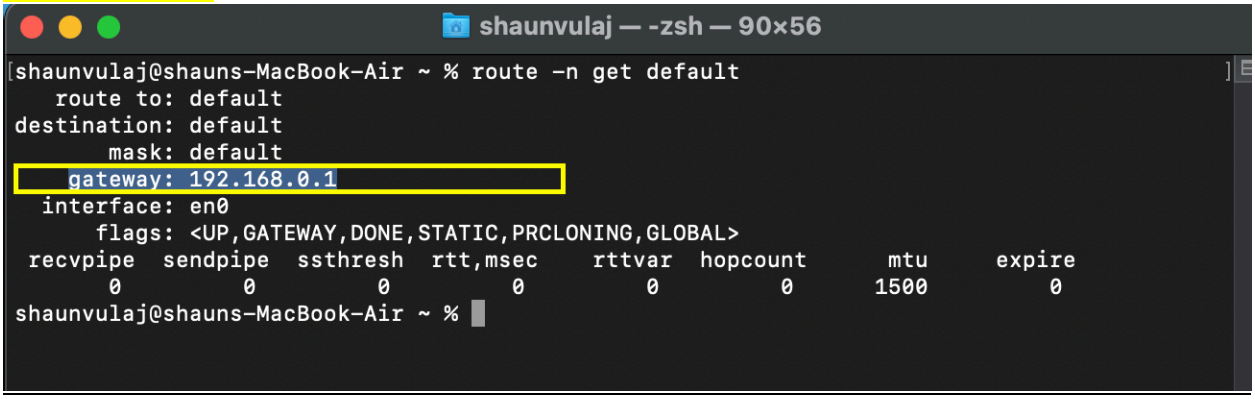


Figure 2 This image is for Task 1-2. Image displays the default gateway for my local computer using command "route -n get default".*

Task 1.1(Step 1)

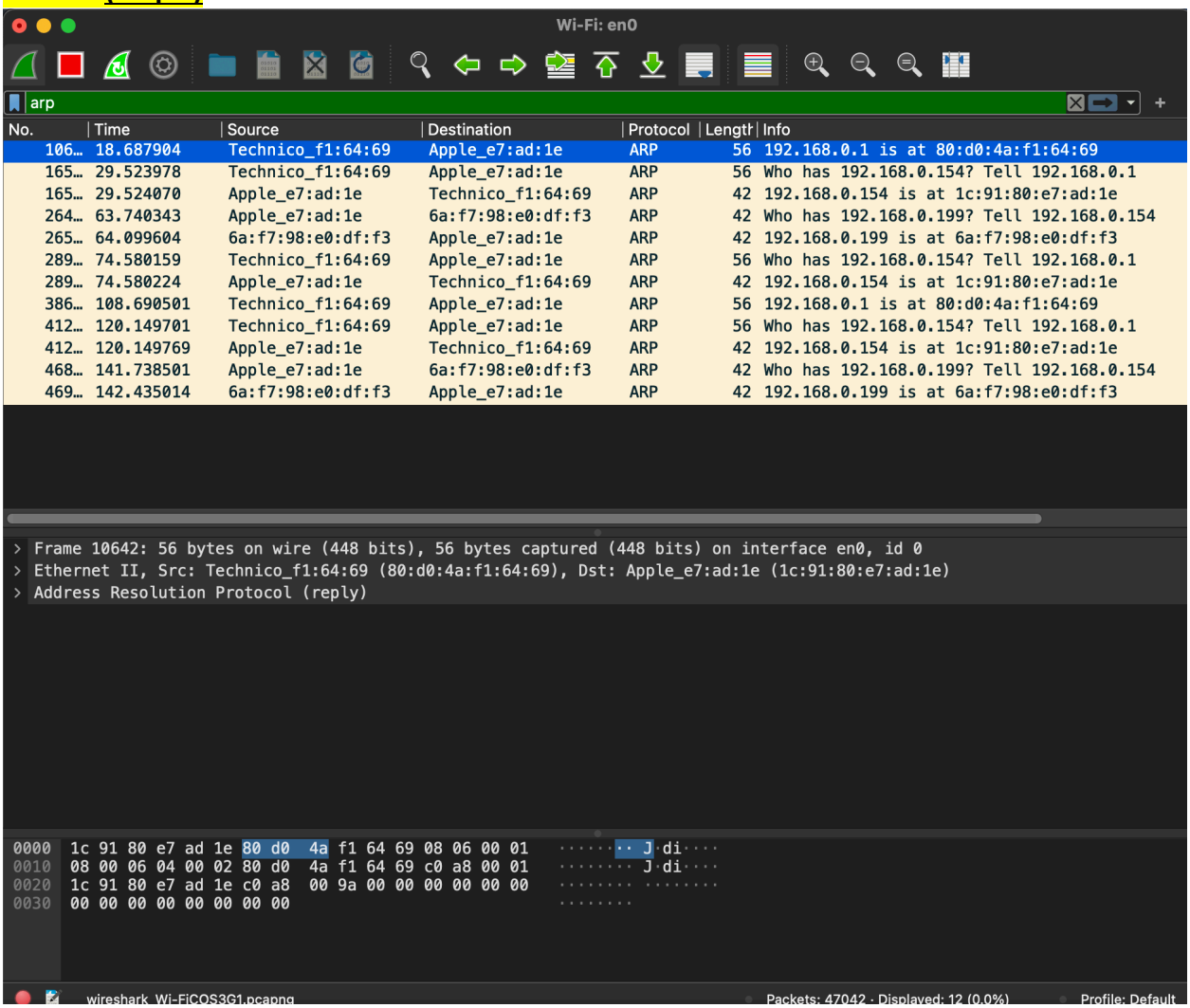


Figure 3 Image is for Task 1-3. This image displays the appropriate filter of network traffic on my local computer using the packet sniffing tool WireShark as described in Assignment 1.2 for SER 321.*

Task 1.1(Step 1)

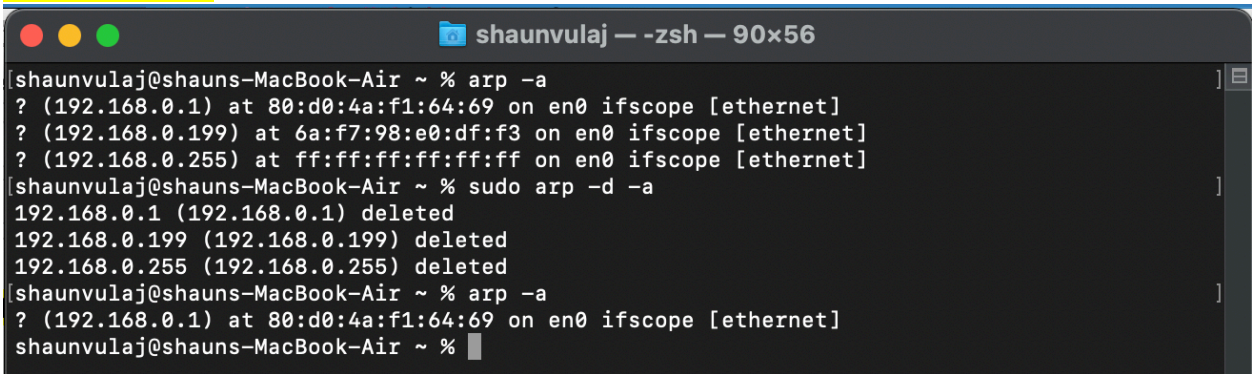


Figure 4 This image is for task 1-4. This image displays the arp cache using command "arp -a" then clearing the cache using command "sudo arp -d -a" and then again checking the arp cache to see if nodes have been deleted using "arp -a".*

Task 1.1(Step 1)

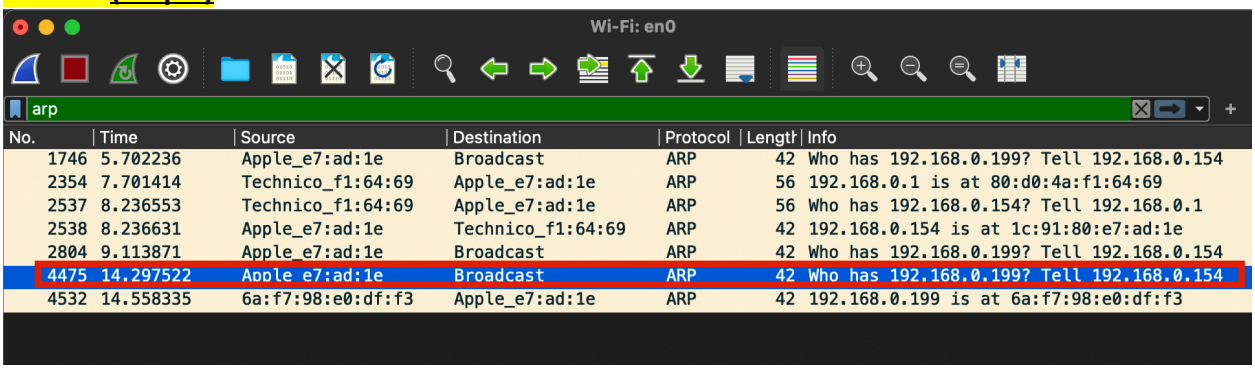


Figure 5 Image is for Task 1-5/6. This image displays an ARP traffic on my local computer.*

Task 1.1 (Step 2)

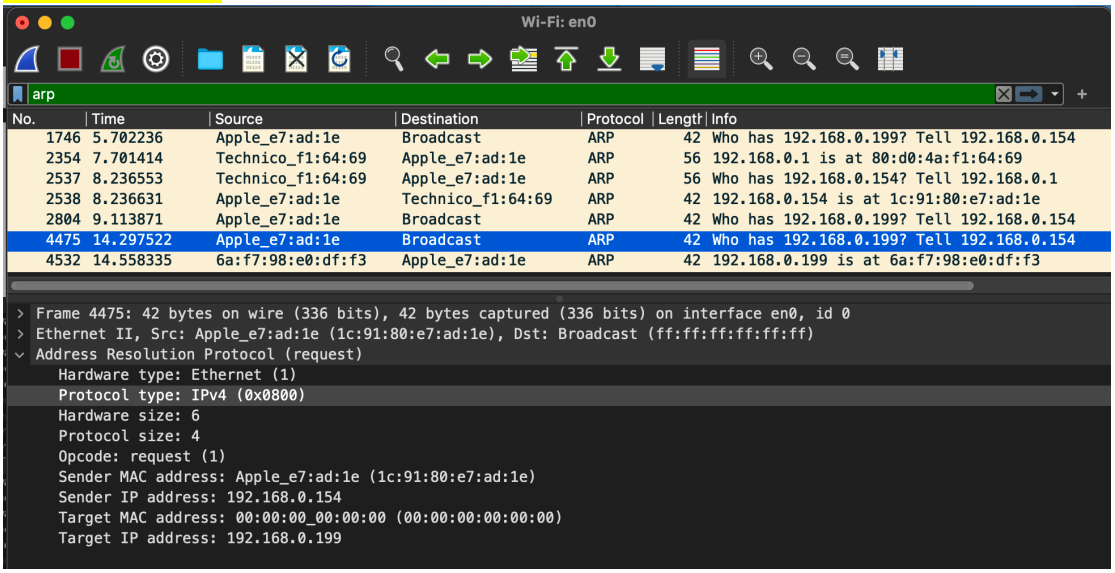


Figure 6 This image is for Task 1-7 of the SER 321 Assignment 1.2. This image displays the WireShark display for a request on my local network.*

Task 1.1 (Step 2)

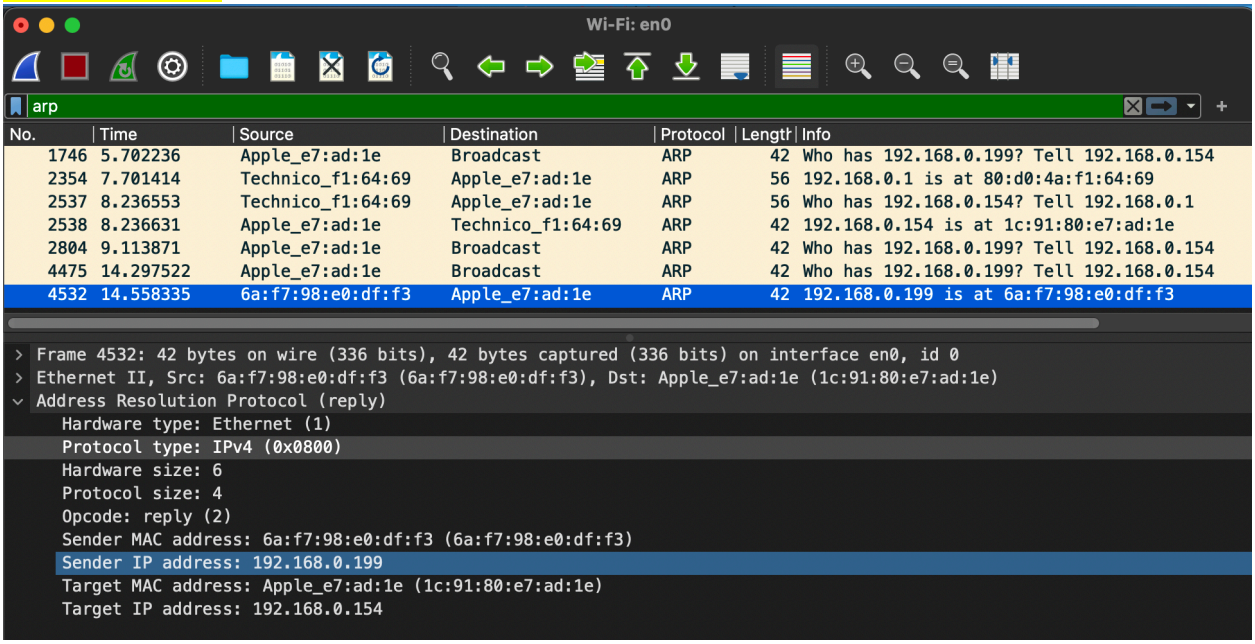


*Figure 7*This image is for Task 1-7 of the SER 321 Assignment 1.2. This image displays the WireShark display for a reply on my local network.*

Task 1.1(Step 3)

1. **What opcode is used to indicate a request? What about a reply?**

   Request Opcode: request (1) ~ 0001

   Reply Opcode: reply (2) ~ 0002

2. **How large is the ARP header for a request? What about for a reply? You will need to research this (hint: some sources define what belongs to the header differently, name which source you base your answer on)**

   Requests and Replies are 28bytes.

   Source:
   https://support.huawei.com/enterprise/en/doc/EDOC1100143232/eafab34b/arp-fundamentals

3. **What value is carried on a request for the unknown target MAC address?**

   00:00:00:00:00:00

4. **What Ethernet Type value indicates that ARP is the higher layer protocol?**

   Type: ARP (0x0806)

==========

Task 1.2 (Understanding TCP network sockets)

==========

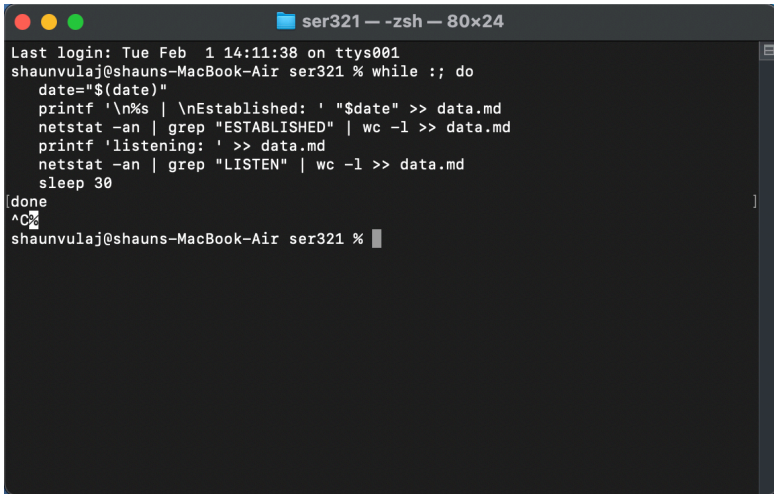


Figure 8 script used to monitor Listening & Established connection on my network*

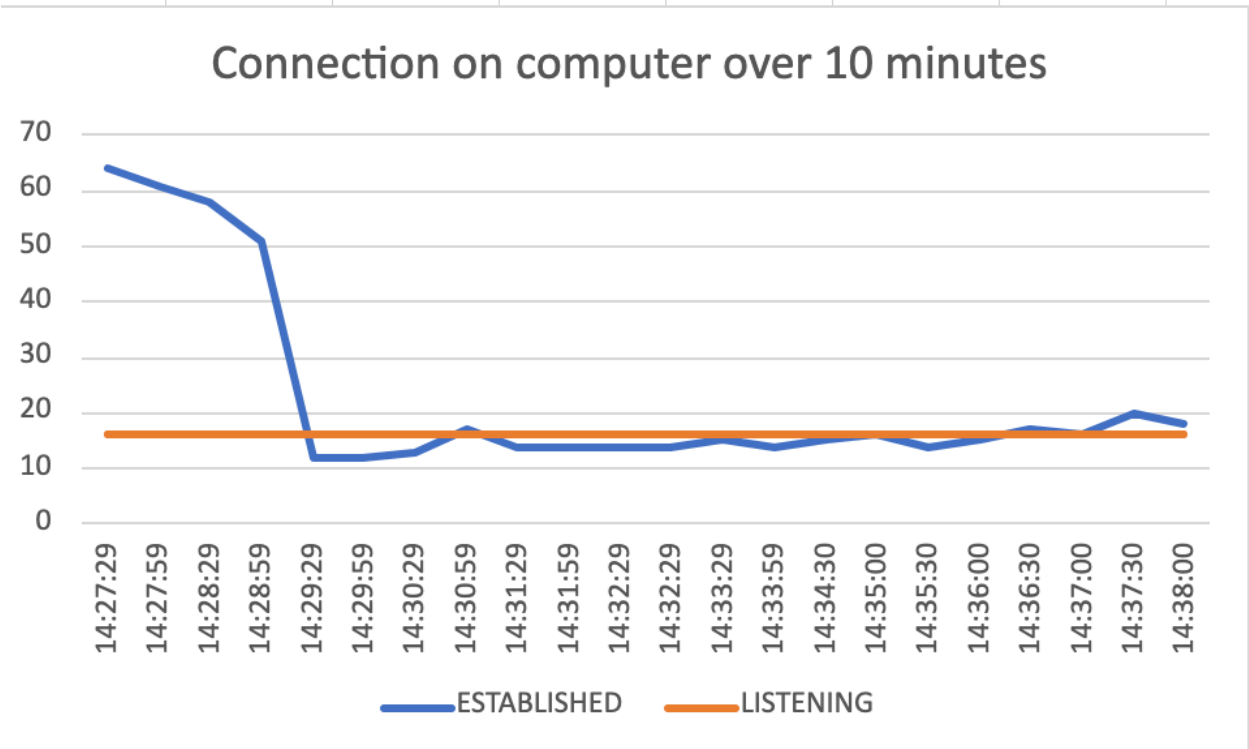


Figure 9 Graph displaying read data from 10 minutes of activity on my local computers network*

==========

Task 1.3 (Sniffing TCP/UDP traffic)

==========

Step 1 (TCP):

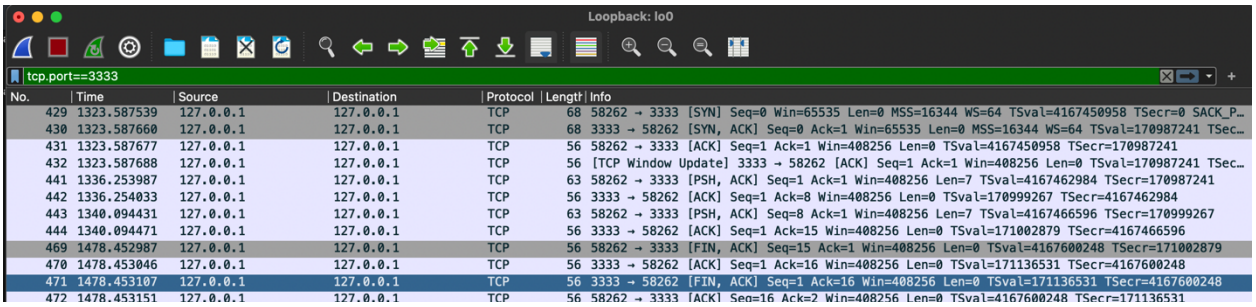


Figure 10 This image displays the network traffic with TCP filter applied in wireshark*

a) Explain both the commands you used in detail. What did they actually do?

"`nc -k -l 3333`" = continue listening after disconnection

`-k` = tells nc to keep listening for a connection even after its current connection finishes.

`-l` = Used to specify that nc should listen for an incoming connection rather than initiate a connection to a remote host.

"`nc 127.0.0.1 3333`" = is a default port scan (eg. NC [host] [port]). Connects to the machine and port being listened on.

b) How many frames were send back and forth to capture these 2 lines (Frames: 4 – I counted all frames that were sent)?

4

c) How many packets were send back and forth to capture only those 2 lines?

4

d) How many packets were needed to capture the whole "process" (starting the communication, ending the communication)?

12

e) How many bytes is the data (only the data) that was send?

14

f) How many total bytes went over the wire (back and forth) for the whole process?

710 Bytes

g) How much overhead was there. Basically how many bytes was the whole process compared to the actually data that we did send.

696 Bytes

Step 2 (UDP):

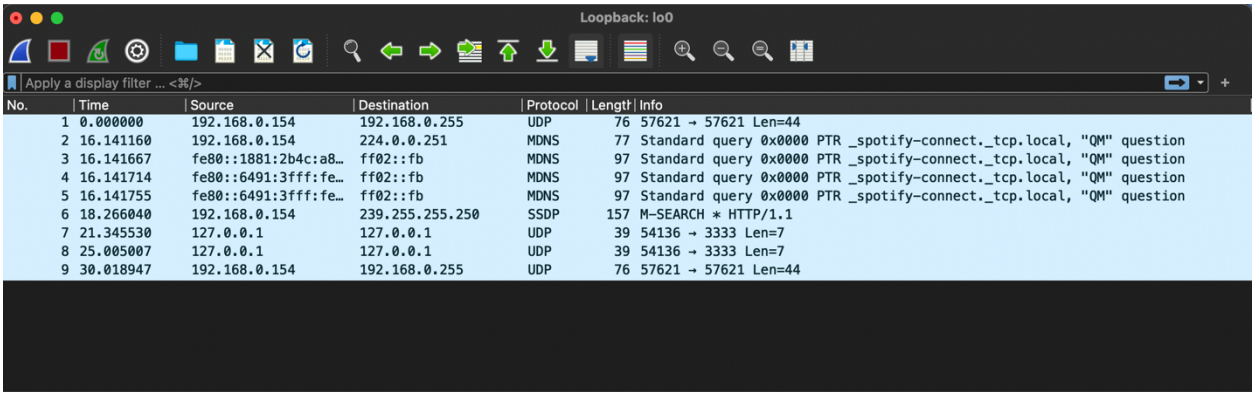


Figure 11 This image displays the network traffic with UDP filter applied in WireShark*

a) Explain both the commands you used in detail. What did they actually do?

"nc –k –l –u 3333" = continue listening after disconnection

–k = tells nc to keep listening for a connection even after its current connection finishes.

–l = Used to specify that nc should listen for an incoming connection rather than initiate a connection to a remote host.

–u = Connects to UDP

"nc -u 127.0.0.1 3333" = is a default port scan (eg. NC [host] [port]). Connects to the machine and port being listened on.

b) How many frames were send back and forth to capture these 2 lines (Frames: 4 – I counted all frames that were sent)?

2

c) How many packets were send back and forth to capture only those 2 lines?

2

d) How many packets were needed to capture the whole "process" (starting the communication, ending the communication)?

9

e) How many bytes is the data (only the data) that was send?

14

f) How many total bytes went over the wire (back and forth) for the whole process?

755

g) How much overhead was there. Basically, how many bytes was the whole process compared to the actually data that we did send.

741

==========

Task 1.4. Internet Protocol (IP) Routing

==========

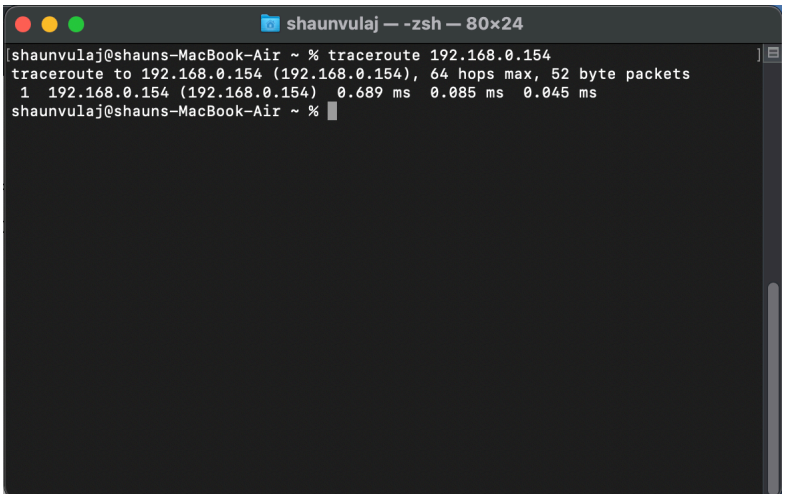


Figure 12 traceroute command in terminal to see the connection hops for my local computer*

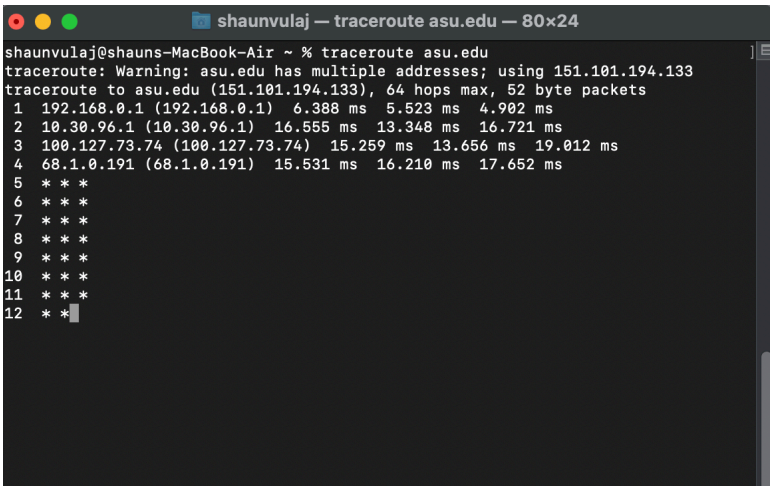


Figure 13 traceroute command in terminal to see the connection hops for my ASU's website*

a) Which is the fastest?

My local computer

b) Which has the fewest hops?

My computer

==========

Task 1.5 Running client servers in different ways

==========

Task 1.5.1

My video demonstrating local client server communication using WireShark for analysis: https://www.youtube.com/watch?v=I_SMshzFAPg&ab_channel=ShaunVulaj

Task 1.5.2

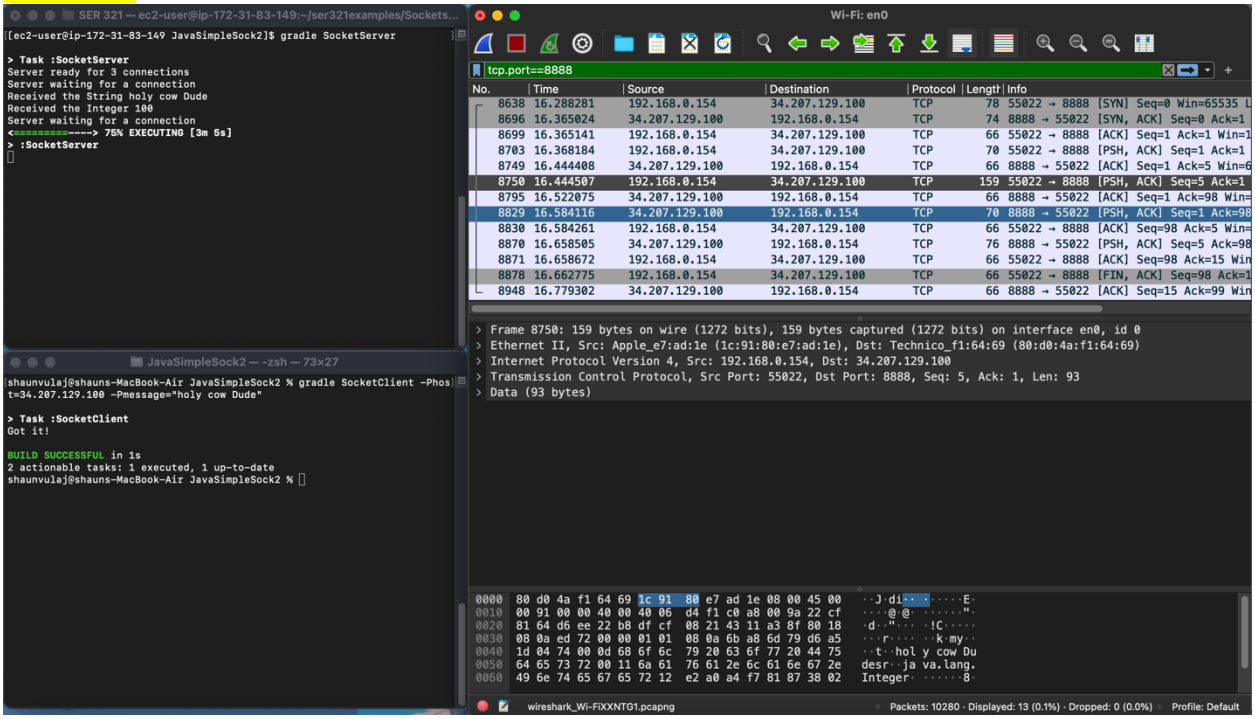
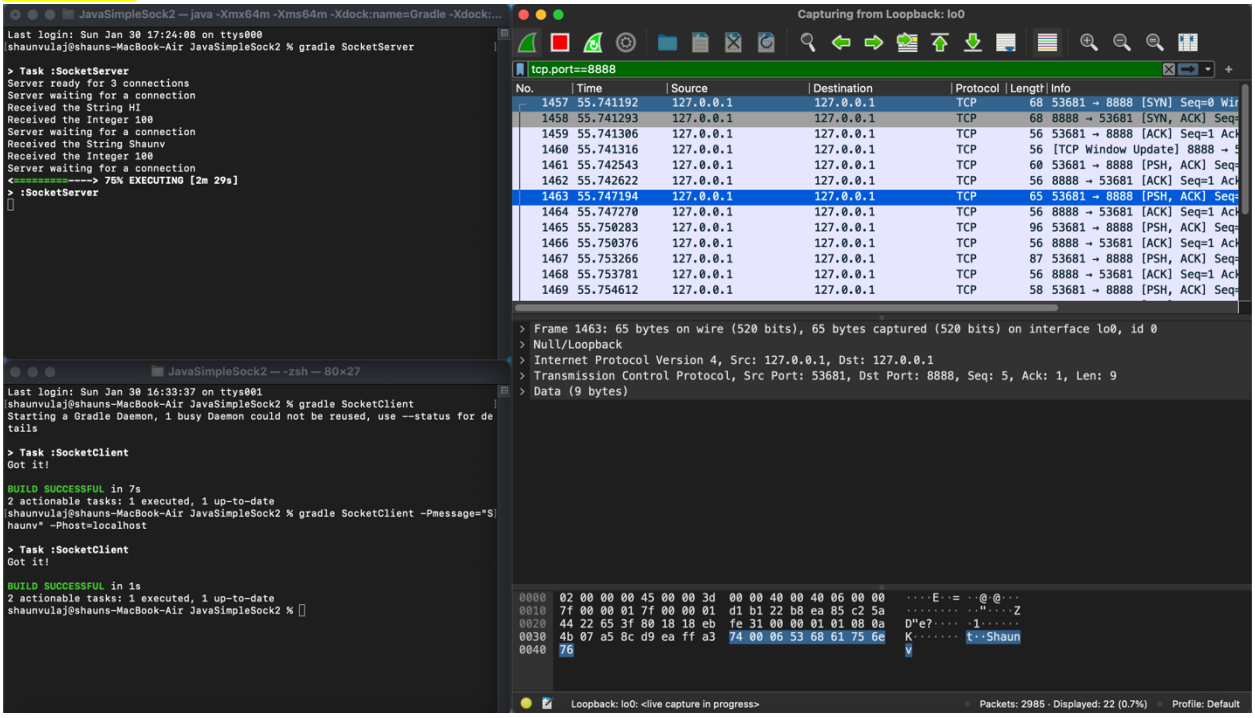


Figure 14 Image displaying the network activity between my local computer and my AWS server with Wirshark tool being used for analysis*

I did not need to change much on WireShark for this task. The only thing that was changed from the previous task was that I switched the kind of connection we were looking at so, I changed in the main menu from Loopback: lo0 to Wi-Fi: en0. In addition, I noticed in the decrypted section at the bottom of the WireShark now displayed a “sr” directly after my message also, I notice that now, in the same frame I see the message “ja va.lang”. Previously that was not the case, that message was in a different frame.

Task 1.5.2



Task 1.5.3

This does not seem to work and I believe that the code was written to accept and search for the specific port 8888. In our case we have assigned port 8888 specifically on our AWS webpage so when we run the client on AWS it tries to search for that 8888 port but our computer could use any of the appropriate ports to make that connection, we have not specifically set our computer to use 8888. So, when we are on AWS it cannot find us.

Task 1.5.4

Why can you easily reach your server on AWS with a client running in your local network but not as easily go the other direction?

I believe is works this way because local computer IP addresses are hidden from users outside of my local IP. Routers usually have this alias for my local computer for security reasons. So AWS cannot find my local computer as easily as my local computer can find AWS. My evidence for this is that we have the exact IP address of AWS from the webpage but you would think, “I have my local IP, it should work the same” but the way our local computer exists on the internet is as an alias which is not visible to us(I think) so when we put our IP address as our host from AWS it looks for our actual address but what it needs is our alias, which is a kinda of secret address.

And what can you do to reach your server in your local network if you want to reach it from outside your network (you do not have to do that)?

I believe that the solution to this issue is to find your alias.

What is the "issue" if you want to run your server locally and reach it from the "outside world"?

My computer is using a private IP address that the outside world does not have access to so you cannot establish the connection.