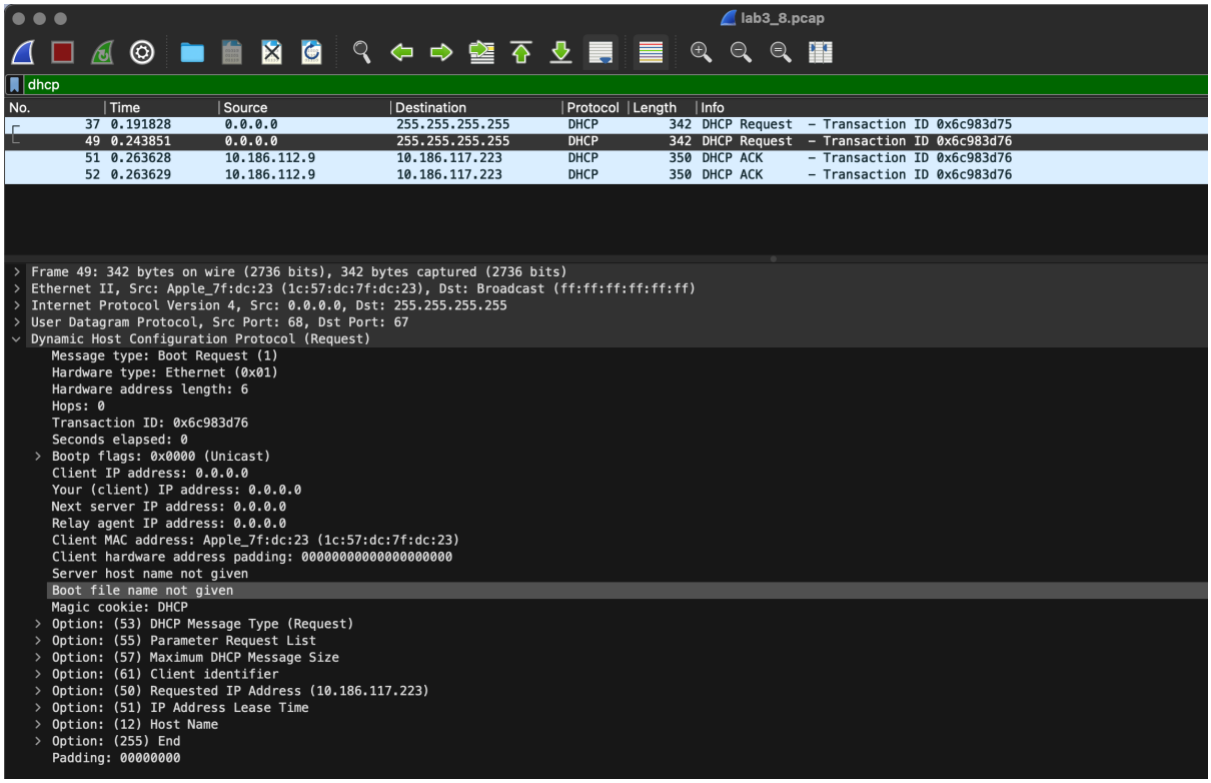
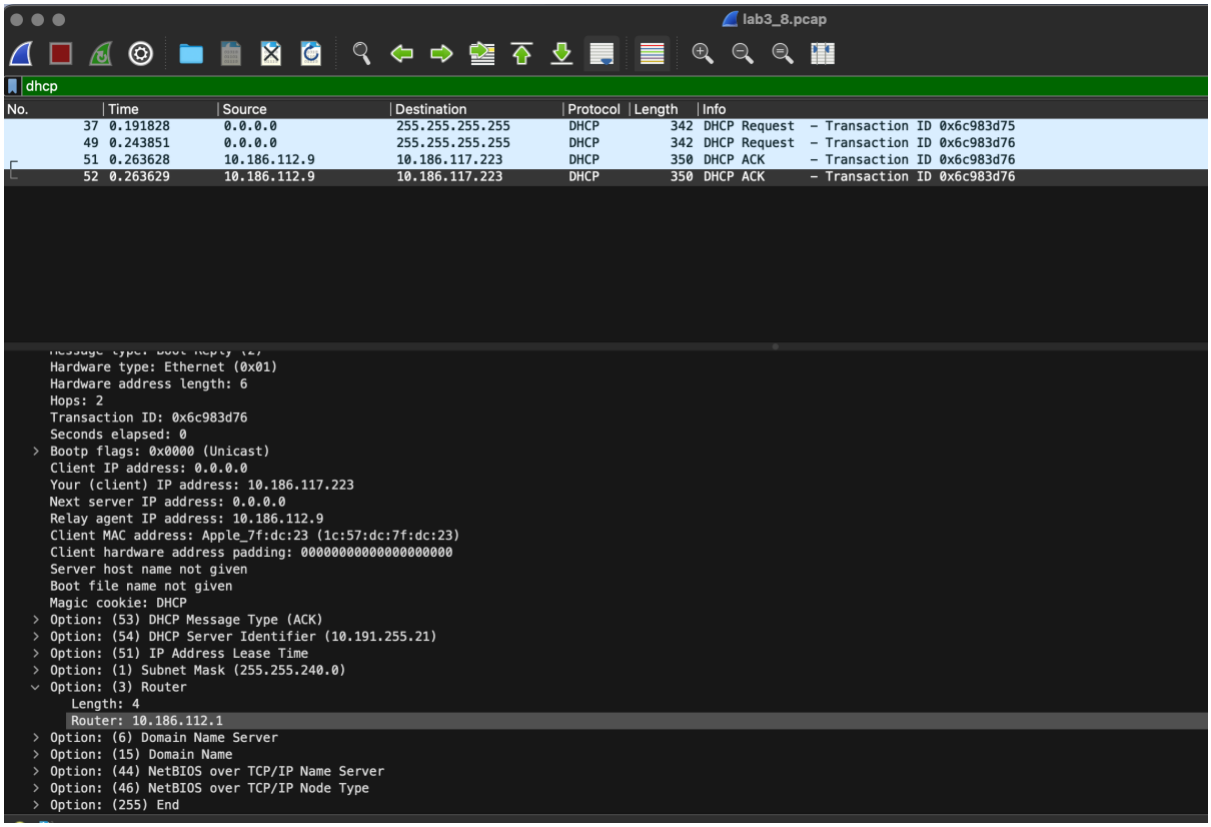
Shivam Bhat
bhat41@purdue.edu, 0033760929

Q1 Read Chapter

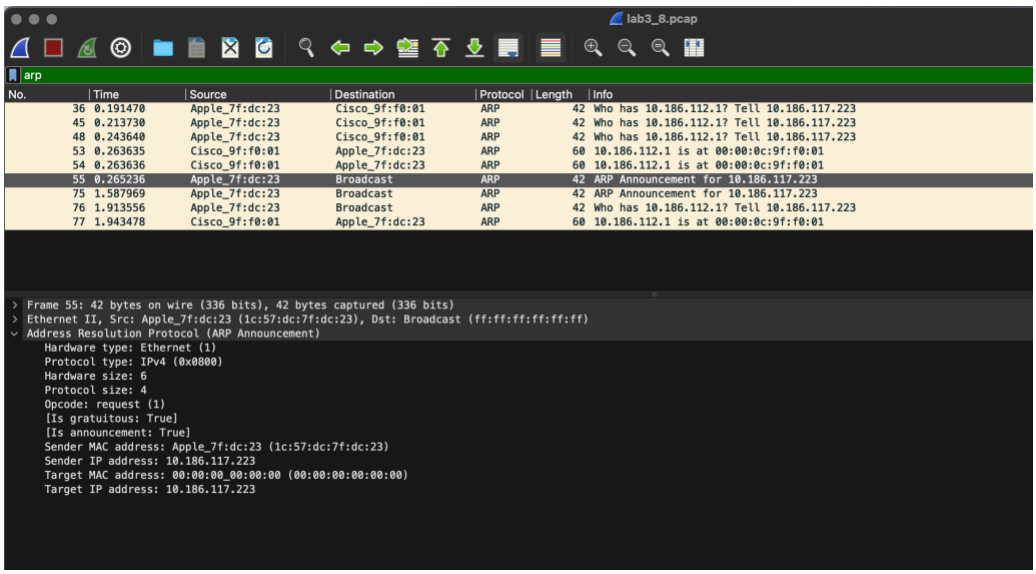
Q2 lab3.pcap file included in zip

Q3
DHCP

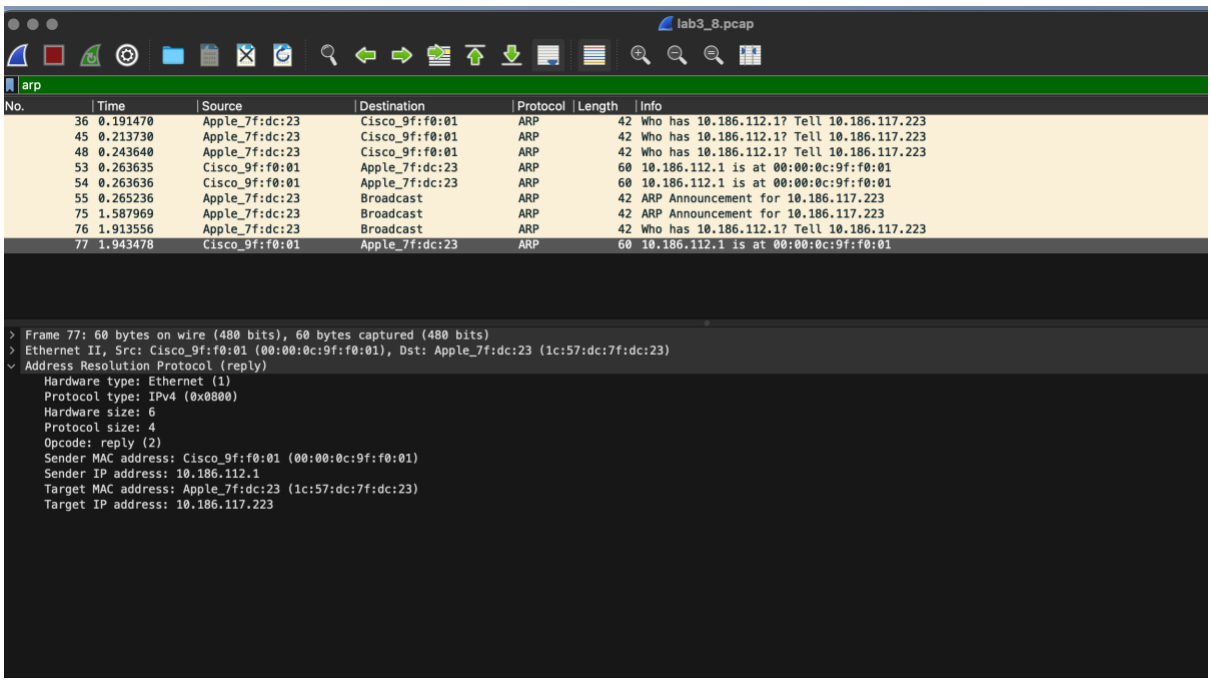




ARP Announcement



ARP Response



DHCP is used for IP address. DHCP stands for Dynamic Host Configuration Protocol, and it is a network protocol commonly used in IP networks to automatically assign IP addresses and other network configuration parameters to devices on a network. As part of this default gateways are also assigned.

We can see DHCPDISCOVER **(DHCP Discover)** messages where the device uses the **0.0.0.0** and **255.255.255.255** as the source address and destination address, respectively. The DCHP exchange close with the DHCKACK (after offer and request) where in the server acknowledges that the client can use the configuration. Here under the router we can see the gateway router ip.-**10.186.112.1** The offer and request are not show. This could be because of many reasons one of which could be encryption.

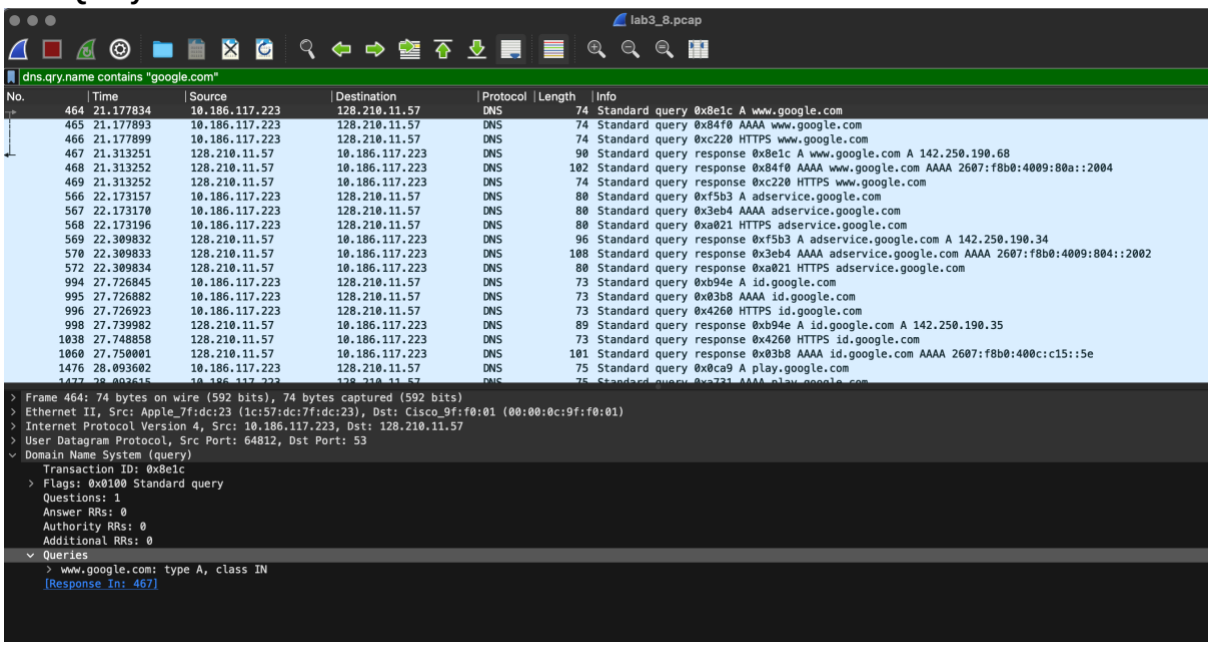
For mac addresses we use the ARP packets. ARP (Address Resolution Protocol) is a network protocol used to map IP (Internet Protocol) addresses to MAC (Media Access Control) addresses. The ARP request is broadcasted, and the device with the corresponding IP address responds with its MAC address. We can in line 77 my machine Apple_7f receives the resolved MAC address.

Hence First DHCP is used to obtain the ip address and gateway router ip following which ARP is used to obtain MAC addresses. We can see the DHCP ACK happens at 52 and MAC address is obtained via ARP at 53.

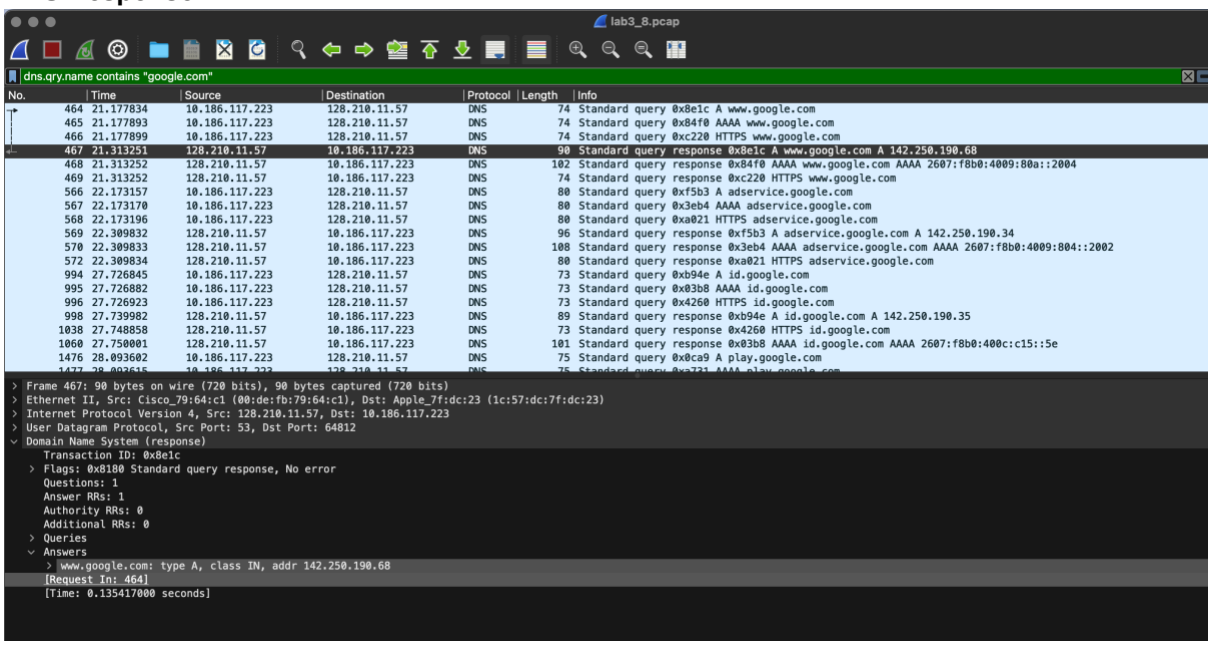
Also if we notice the packet sequence for this pcap we see that there is one ARP packet just before DCHP at 36,37 respectively. This could be a result of host already caching its previous IP and then probed to check if that's still unchanged or taken by someone else. Here it then sees through the DHCP reply that this IP is free and hence continues to take that for ARP. This use of caching speeds up the processes and helps handle failure fallbacks. None the less for new hosts , first DHCP will happen then ARP.

B)

DNS Query



DNS Response



To locate the google DNS resolution I used the following query **dns.qry.name contains "google.com".**

I then searched for all the packet where it looked for google.com. There we can see queries followed by responses. In the response we can check the answers of DNS responses(Answers) field to identify the ip which is **142.250.190.68**

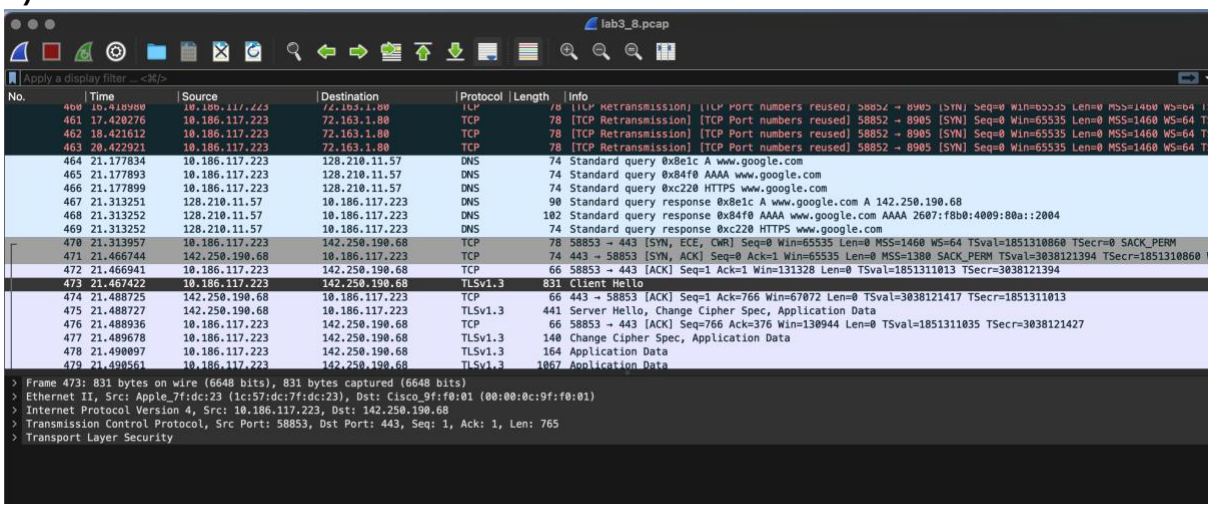
C)

I am not able to see any intra-AS or inter-AS routing. Wireshark records and examines network data at the packet level, enabling us to look inside packets as they are sent over a network. However, intra-AS and inter-AS routing are more complex higher-level network operations occurring at the network layer and are frequently invisible to Wireshark's packet-level analysis.

Routing information is exchanged between routers within and between autonomous systems, respectively, in intra-AS and inter-AS routing. The most efficient route for traffic to take both within and between networks is chosen using this information. This routing information is exchanged using routing protocols like OSPF, BGP which operate at a higher layer of network stack than what wireshark operates in.

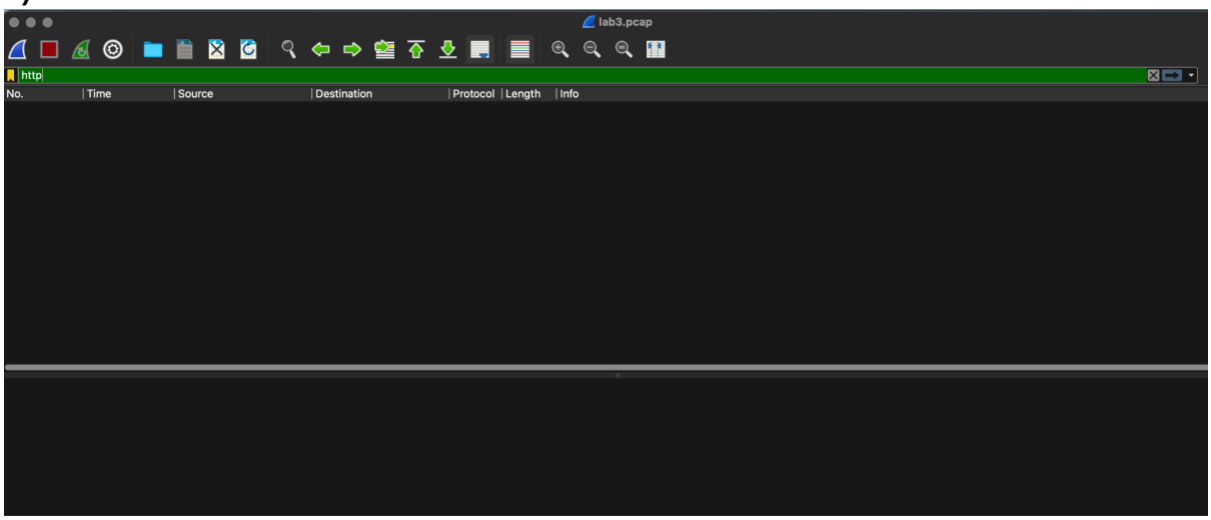
Also, when using wireshark we are using WiFi to capture the packets. My machine connects to an AP which then enables a connection to the internet. Hence there is no need for AS routing in the host as its function is just to transmit packets. AP takes care of the routing

D)

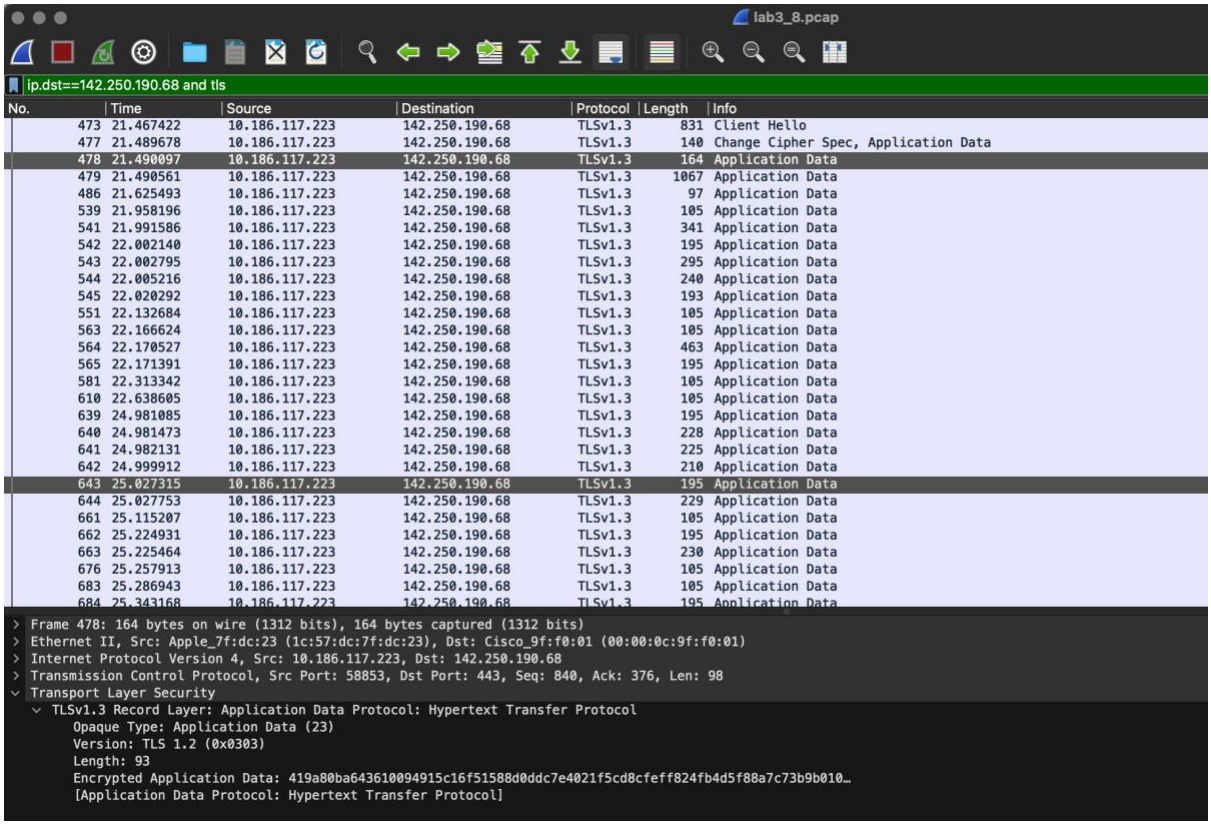


We can see the TCP handshake in the first 3 lines- 470-472.
We can also use the ‘tcp’ filter in search bar

E)



Since google.com is using the TLS protocol for encryptions, I can’t see any explicit HTTP messages. So there are no HTTP GET messages as well. I can see TLS packets with encrypted messages(payload) being sent from google.com


lab3_8.pcap
ip.dst==142.250.190.68 and tls
No. Time Source Destination Protocol Length Info
473 21.467422 10.186.117.223 142.250.190.68 TLSv1.3 831 Client Hello
477 21.489678 10.186.117.223 142.250.190.68 TLSv1.3 140 Change Cipher Spec, Application Data
478 21.490097 10.186.117.223 142.250.190.68 TLSv1.3 164 Application Data
479 21.490561 10.186.117.223 142.250.190.68 TLSv1.3 1067 Application Data
486 21.625493 10.186.117.223 142.250.190.68 TLSv1.3 97 Application Data
539 21.958196 10.186.117.223 142.250.190.68 TLSv1.3 105 Application Data
541 21.991586 10.186.117.223 142.250.190.68 TLSv1.3 341 Application Data
542 22.002140 10.186.117.223 142.250.190.68 TLSv1.3 195 Application Data
543 22.002795 10.186.117.223 142.250.190.68 TLSv1.3 295 Application Data
544 22.005216 10.186.117.223 142.250.190.68 TLSv1.3 240 Application Data
545 22.020292 10.186.117.223 142.250.190.68 TLSv1.3 193 Application Data
551 22.132684 10.186.117.223 142.250.190.68 TLSv1.3 105 Application Data
563 22.166624 10.186.117.223 142.250.190.68 TLSv1.3 105 Application Data
564 22.170527 10.186.117.223 142.250.190.68 TLSv1.3 463 Application Data
565 22.171391 10.186.117.223 142.250.190.68 TLSv1.3 195 Application Data
581 22.313342 10.186.117.223 142.250.190.68 TLSv1.3 105 Application Data
610 22.638605 10.186.117.223 142.250.190.68 TLSv1.3 105 Application Data
639 24.981085 10.186.117.223 142.250.190.68 TLSv1.3 195 Application Data
640 24.981473 10.186.117.223 142.250.190.68 TLSv1.3 228 Application Data
641 24.982131 10.186.117.223 142.250.190.68 TLSv1.3 225 Application Data
642 24.999912 10.186.117.223 142.250.190.68 TLSv1.3 210 Application Data
643 25.027315 10.186.117.223 142.250.190.68 TLSv1.3 195 Application Data
644 25.027753 10.186.117.223 142.250.190.68 TLSv1.3 229 Application Data
661 25.115207 10.186.117.223 142.250.190.68 TLSv1.3 105 Application Data
662 25.224931 10.186.117.223 142.250.190.68 TLSv1.3 195 Application Data
663 25.225464 10.186.117.223 142.250.190.68 TLSv1.3 230 Application Data
676 25.257913 10.186.117.223 142.250.190.68 TLSv1.3 105 Application Data
683 25.286943 10.186.117.223 142.250.190.68 TLSv1.3 105 Application Data
684 25.343168 10.186.117.223 142.250.190.68 TLSv1.3 195 Application Data
Frame 478: 164 bytes on wire (1312 bits), 164 bytes captured (1312 bits)
Ethernet II, Src: Apple_7f:dc:23 (1c:57:dc:7f:dc:23), Dst: Cisco_9f:f0:01 (00:00:0c:9f:f0:01)
Internet Protocol Version 4, Src: 10.186.117.223, Dst: 142.250.190.68
Transmission Control Protocol, Src Port: 58853, Dst Port: 443, Seq: 840, Ack: 376, Len: 98
Transport Layer Security
TLSv1.3 Record Layer: Application Data Protocol: Hypertext Transfer Protocol
Opaque Type: Application Data (23)
Version: TLS 1.2 (0x0303)
Length: 93
Encrypted Application Data: 419a80ba643610094915c16f51588d0ddc7e4021f5cd8cfeff824fb4d5f88a7c73b9b010…
[Application Data Protocol: Hypertext Transfer Protocol]

Q4

On Amber

```
data 70 $ python3 lab3.py A "lab3.pcap"
CASE A:
IPAddr: 10.186.112.1
MACAddr: 00:00:0c:9f:f0:01
data 71 $ python3 lab3.py B "lab3.pcap" "www.google.com"
CASE B:
IPAddr-DEST: 142.250.190.68
data 72 $ python3 lab3.py C "lab3.pcap" "www.google.com"
CASE C:
IPAddr-SRC: 10.186.117.223
IPAddr-DEST: 142.250.190.68
Port-DEST: 443
SYN: 1
ACK: 0
IPAddr-SRC: 142.250.190.68
IPAddr-DEST: 10.186.117.223
Port-DEST: 443
SYN: 1
ACK: 1
IPAddr-SRC: 10.186.117.223
IPAddr-DEST: 142.250.190.68
Port-DEST: 443
SYN: 0
ACK: 1
data 73 $ python3 lab3.py ALL "lab3.pcap" "www.google.com"
CASE A:
IPAddr: 10.186.112.1
MACAddr: 00:00:0c:9f:f0:01
CASE B:
IPAddr-DEST: 142.250.190.68
CASE C:
IPAddr-SRC: 10.186.117.223
IPAddr-DEST: 142.250.190.68
Port-DEST: 443
SYN: 1
ACK: 0
IPAddr-SRC: 142.250.190.68
IPAddr-DEST: 10.186.117.223
Port-DEST: 443
SYN: 1
ACK: 1
IPAddr-SRC: 10.186.117.223
IPAddr-DEST: 142.250.190.68
```

On Local

```
⌥⌘1                                    -zsh
~/Doc/P/PurduePrivate/l/c/l3/Lab3_0033760929 devCNS *4 ?14 ❯ python3 lab3.py A "lab3.pcap"        base
CASE A:
IPAddr: 10.186.112.1
MACAddr: 00:00:0c:9f:f0:01
~/Doc/P/PurduePrivate/l/c/l3/Lab3_0033760929 devCNS *4 ?14 ❯ python3 lab3.py B "lab3.pcap" "www.google.com"
CASE B:
IPAddr-DEST: 142.250.190.68
~/Doc/P/PurduePrivate/l/c/l3/Lab3_0033760929 devCNS *4 ?14 ❯ python3 lab3.py C "lab3.pcap" "www.google.com"        base
CASE C:
IPAddr-SRC: 10.186.117.223
IPAddr-DEST: 142.250.190.68
Port-DEST: 443
SYN: 1
ACK: 0
IPAddr-SRC: 142.250.190.68
IPAddr-DEST: 10.186.117.223
Port-DEST: 443
SYN: 1
ACK: 1
IPAddr-SRC: 10.186.117.223
IPAddr-DEST: 142.250.190.68
Port-DEST: 443
SYN: 0
ACK: 1
~/Doc/P/PurduePrivate/l/c/l3/Lab3_0033760929 devCNS *4 ?14 ❯ python3 lab3.py ALL "lab3.pcap" "www.google.com"        base
CASE A:
IPAddr: 10.186.112.1
MACAddr: 00:00:0c:9f:f0:01
CASE B:
IPAddr-DEST: 142.250.190.68
CASE C:
IPAddr-SRC: 10.186.117.223
IPAddr-DEST: 142.250.190.68
Port-DEST: 443
SYN: 1
ACK: 0
IPAddr-SRC: 142.250.190.68
IPAddr-DEST: 10.186.117.223
Port-DEST: 443
SYN: 1
ACK: 1
IPAddr-SRC: 10.186.117.223
IPAddr-DEST: 142.250.190.68
Port-DEST: 443
SYN: 0
ACK: 1
```

<u>Instruction</u>

For handshakes I print one compete handshake **SYN =1, SYN =1 ACK=1, SYN=0 ACK =1 as mentioned by TA at https://campuswire.com/c/GC1863205/feed/381**

I have disables QUIC as mentioned by TA in https://campuswire.com/c/GC1863205/feed/377

Command : Python3 lab3.py MODE FILEPATH WEBSITE

Import requirements
Install scapy
import logging

import logging is used for supressing warning as mentioned by TA at
https://campuswire.com/c/GC1863205/feed/395