



**Shaun Vulaj  
SER 321  
Assignment 1.2  
01/27/2023**

---

---

## Task 1.1 (Explore the Data Link Layer with ARP)

---

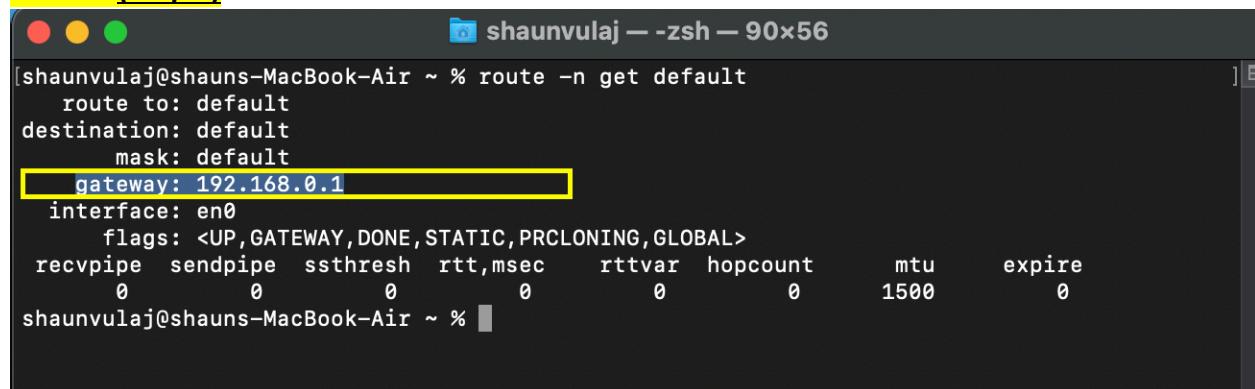
---

### Task 1.1(Step 1)

```
shaunvulaj -- zsh -- 126x56
member: en2 flags=3<LEARNING,DISCOVER>
          ifmaxaddr 0 port 10 priority 0 path cost 0
nd6 options=201<PERFORMNUD,DAD>
media: <unknown type>
status: inactive
ap1: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
      options=400<CHANNEL_IO>
      ether 3e:91:80:e7:ad:1e
      nd6 options=201<PERFORMNUD,DAD>
      media: autoselect
      status: inactive
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
      options=400<CHANNEL_IO>
      ether 1c:91:80:e7:ad:1e
      inet6 fe80::1881:2b4c:a87d:1225%en0 prefixlen 64 secured scopeid 0xd
      inet 192.168.0.154 netmask 0xffffffff broadcast 192.168.0.255
      inet6 2000:8800:a5:bf00:1804:4f55:bad1:a3fd prefixlen 64 autoconf secured
      inet6 2000:8800:a5:bf00:f162:59c5:6172:b9f6 prefixlen 64 deprecated autoconf temporary
      inet6 2000:8800:a5:bf00:c5f3 prefixlen 64 dynamic
      inet6 2000:8800:a5:bf00:bd5e:7ec5:d594:ab1f prefixlen 64 autoconf temporary
      nd6 options=201<PERFORMNUD,DAD>
      media: autoselect
      status: active
awdl0: flags=8943<UP,BROADCAST,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 1500
      options=400<CHANNEL_IO>
      ether 0e:b6:a3:83:83:04
      inet6 fe80::cb6:a3ff:fe83:8304%awdl0 prefixlen 64 scopeid 0xe
      nd6 options=201<PERFORMNUD,DAD>
      media: autoselect
      status: active
llw0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
      options=400<CHANNEL_IO>
      ether 0e:b6:a3:83:83:04
      inet6 fe80::cb6:a3ff:fe83:8304%llw0 prefixlen 64 scopeid 0xf
      nd6 options=201<PERFORMNUD,DAD>
      media: autoselect
      status: active
utun0: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 1380
      inet6 fe80::33db:b2ba:5b24:e752%utun0 prefixlen 64 scopeid 0x10
      nd6 options=201<PERFORMNUD,DAD>
utun1: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 2000
      inet6 fe80::ae96:4507:bd1b:ac8b%utun1 prefixlen 64 scopeid 0x11
      nd6 options=201<PERFORMNUD,DAD>
utun2: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 1380
      inet6 fe80::52d2:920:4a48:055%utun2 prefixlen 64 scopeid 0x12
      nd6 options=201<PERFORMNUD,DAD>
utun3: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 1380
      inet6 fe80::33f5:9b20:8977:f784%utun3 prefixlen 64 scopeid 0x13
      nd6 options=201<PERFORMNUD,DAD>
en6: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
      options=6467<RXCSUM,TXCSUM,VLAN_MTU,TS04,TS06,CHANNEL_IO,PARTIAL_CSUM,ZEROINVERT_CSUM>
      ether 00:e0:4c:98:00:f3
      nd6 options=201<PERFORMNUD,DAD>
      media: autoselect (none)
      status: inactive
shaunvulaj@shauns-MacBook-Air ~ %
```

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.

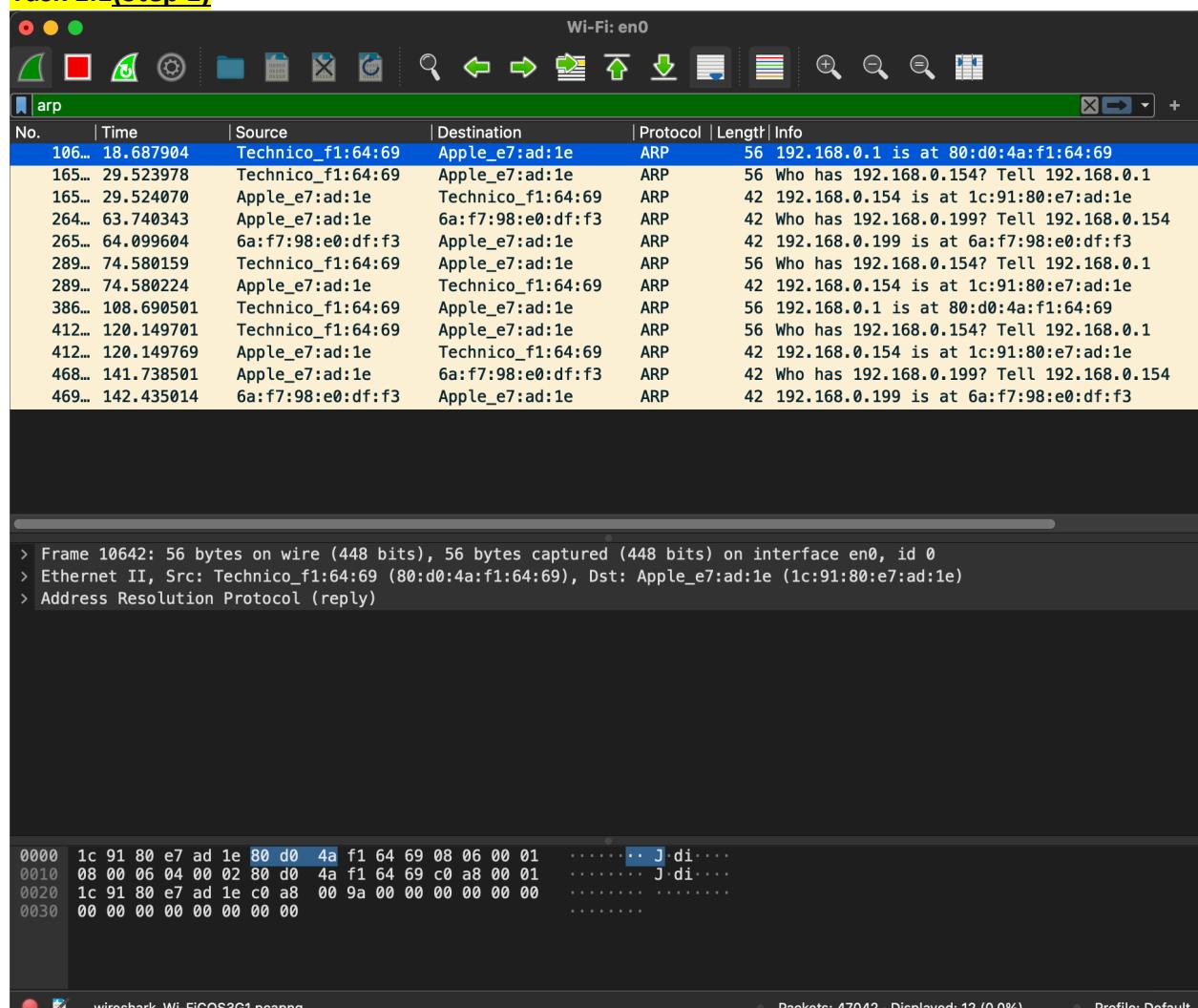
### Task 1.1(Step 1)



```
[shaunvulaj@shauns-MacBook-Air ~ % route -n get default
    route to: default
    destination: default
    mask: default
    gateway: 192.168.0.1
    interface: en0
    flags: <UP,GATEWAY,DONE,STATIC,PRCLONING,GLOBAL>
    recvpipe  sendpipe  ssthresh  rtt,msec      rttvar  hopcount      mtu      expire
              0          0          0          0          0          0        1500          0
shaunvulaj@shauns-MacBook-Air ~ %
```

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)



Wi-Fi: en0

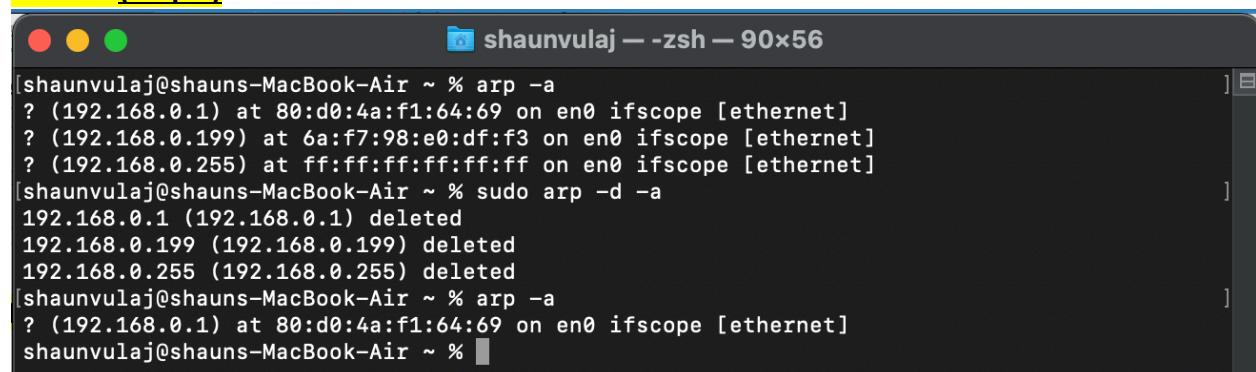
No.	Time	Source	Destination	Protocol	Length	Info
106...	18.687904	Technico_f1:64:69	Apple_e7:ad:1e	ARP	56	192.168.0.1 is at 80:d0:4a:f1:64:69
165...	29.523978	Technico_f1:64:69	Apple_e7:ad:1e	ARP	56	Who has 192.168.0.154? Tell 192.168.0.1
165...	29.524070	Apple_e7:ad:1e	Technico_f1:64:69	ARP	42	192.168.0.154 is at 1c:91:80:e7:ad:1e
264...	63.740343	Apple_e7:ad:1e	6a:f7:98:e0:df:f3	ARP	42	Who has 192.168.0.199? Tell 192.168.0.154
265...	64.099604	6a:f7:98:e0:df:f3	Apple_e7:ad:1e	ARP	42	192.168.0.199 is at 6a:f7:98:e0:df:f3
289...	74.580159	Technico_f1:64:69	Apple_e7:ad:1e	ARP	56	Who has 192.168.0.154? Tell 192.168.0.1
289...	74.580224	Apple_e7:ad:1e	Technico_f1:64:69	ARP	42	192.168.0.154 is at 1c:91:80:e7:ad:1e
386...	108.690501	Technico_f1:64:69	Apple_e7:ad:1e	ARP	56	192.168.0.1 is at 80:d0:4a:f1:64:69
412...	120.149701	Technico_f1:64:69	Apple_e7:ad:1e	ARP	56	Who has 192.168.0.154? Tell 192.168.0.1
412...	120.149769	Apple_e7:ad:1e	Technico_f1:64:69	ARP	42	192.168.0.154 is at 1c:91:80:e7:ad:1e
468...	141.738501	Apple_e7:ad:1e	6a:f7:98:e0:df:f3	ARP	42	Who has 192.168.0.199? Tell 192.168.0.154
469...	142.435014	6a:f7:98:e0:df:f3	Apple_e7:ad:1e	ARP	42	192.168.0.199 is at 6a:f7:98:e0:df:f3

> Frame 10642: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface en0, id 0  
> Ethernet II, Src: Technico\_f1:64:69 (80:d0:4a:f1:64:69), Dst: Apple\_e7:ad:1e (1c:91:80:e7:ad:1e)  
> Address Resolution Protocol (reply)

0000 1c 91 80 e7 ad 1e 80 d0 4a f1 64 69 08 06 00 01 . . . J.di...
0010 08 00 06 04 00 02 80 d0 4a f1 64 69 c0 a8 00 01 . . . J.di...
0020 1c 91 80 e7 ad 1e c0 a8 00 9a 00 00 00 00 00 00 . . .
0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 . . .

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.

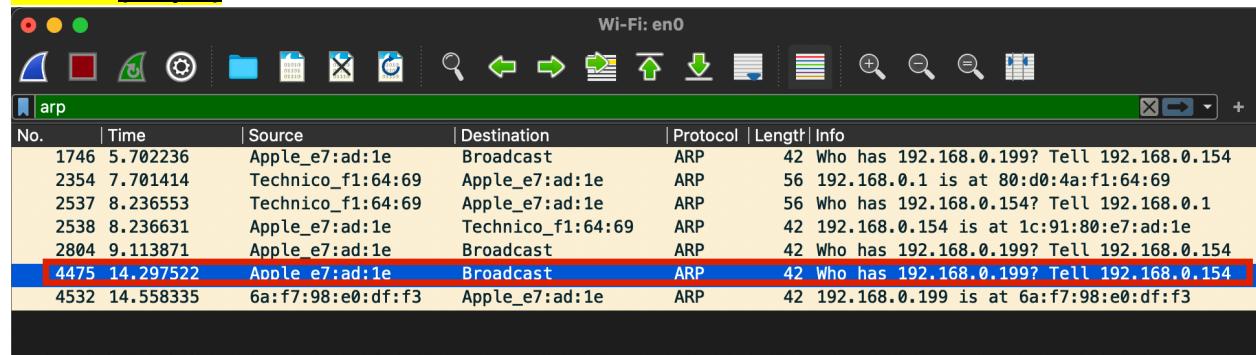
### Task 1.1(Step 1)



```
[shaunvulaj@shauns-MacBook-Air ~ % arp -a
? (192.168.0.1) at 80:d0:4a:f1:64:69 on en0 ifscope [ethernet]
? (192.168.0.199) at 6a:f7:98:e0:df:f3 on en0 ifscope [ethernet]
? (192.168.0.255) at ff:ff:ff:ff:ff:ff on en0 ifscope [ethernet]
[shaunvulaj@shauns-MacBook-Air ~ % sudo arp -d -a
192.168.0.1 (192.168.0.1) deleted
192.168.0.199 (192.168.0.199) deleted
192.168.0.255 (192.168.0.255) deleted
[shaunvulaj@shauns-MacBook-Air ~ % arp -a
? (192.168.0.1) at 80:d0:4a:f1:64:69 on en0 ifscope [ethernet]
shaunvulaj@shauns-MacBook-Air ~ % ]
```

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)



No.	Time	Source	Destination	Protocol	Length	Info
1746	5.702236	Apple_e7:ad:1e	Broadcast	ARP	42	Who has 192.168.0.199? Tell 192.168.0.154
2354	7.701414	Technico_f1:64:69	Apple_e7:ad:1e	ARP	56	192.168.0.1 is at 80:d0:4a:f1:64:69
2537	8.236553	Technico_f1:64:69	Apple_e7:ad:1e	ARP	56	Who has 192.168.0.154? Tell 192.168.0.1
2538	8.236631	Apple_e7:ad:1e	Technico_f1:64:69	ARP	42	192.168.0.154 is at 1c:91:80:e7:ad:1e
2804	9.113871	Apple_e7:ad:1e	Broadcast	ARP	42	Who has 192.168.0.199? Tell 192.168.0.154
4475	14.297522	Apple_e7:ad:1e	Broadcast	ARP	42	Who has 192.168.0.199? Tell 192.168.0.154
4532	14.558335	6a:f7:98:e0:df:f3	Apple_e7:ad:1e	ARP	42	192.168.0.199 is at 6a:f7:98:e0:df:f3

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

### Task 1.1 (Step 2)

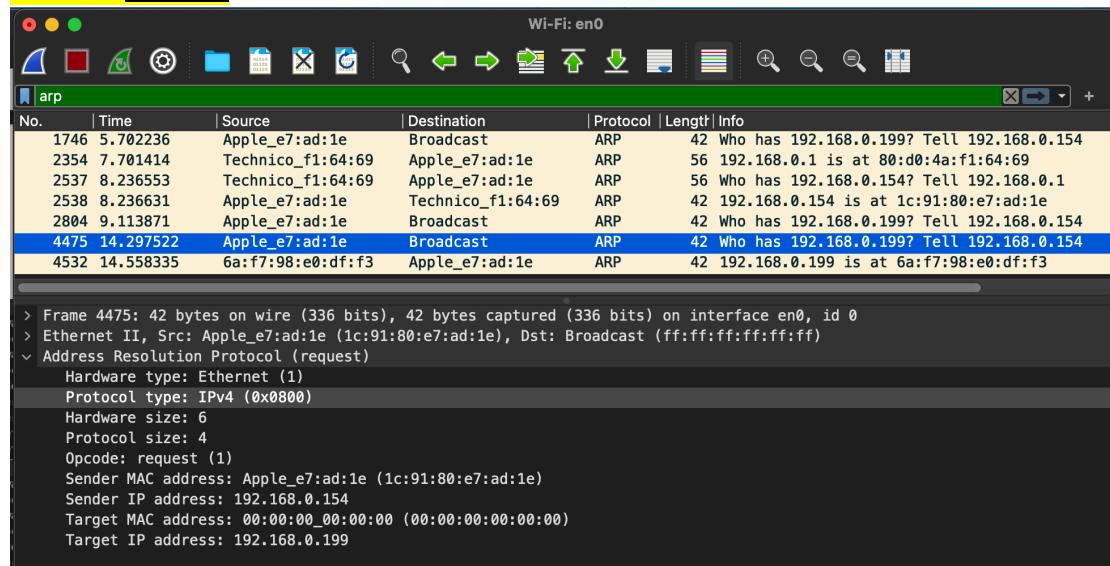


Figure 6\* This image is for Task 1-7 of the SER 321 Assignment 1.2. This image displays the Wireshark display for a request on my local network.

### Task 1.1 (Step 2)

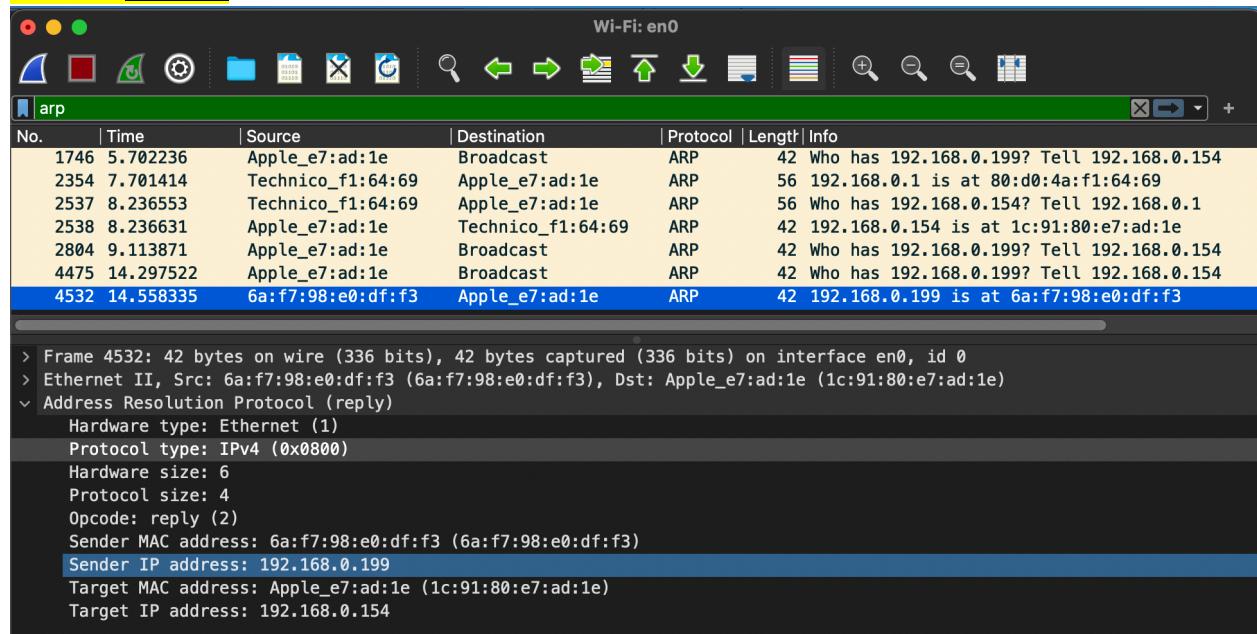


Figure 7\*This image is for Task 1-7 of the SER 321 Assignment 1.2. This image displays the Wireshark display for a reply on my local network.

### **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

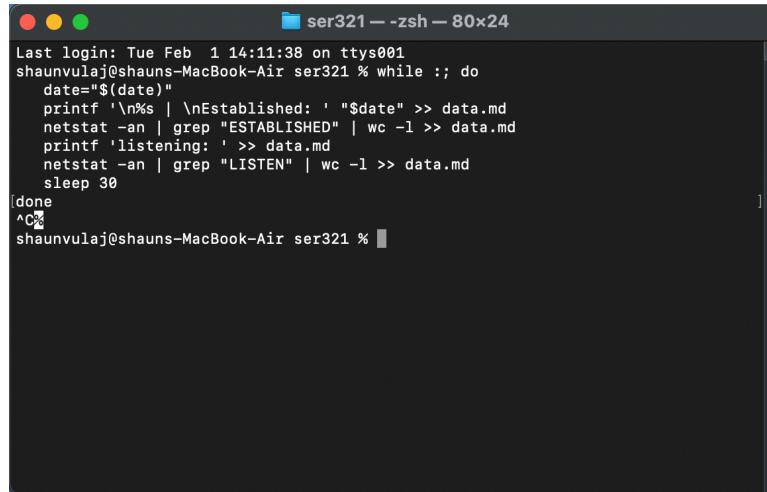
- 4. What Ethernet Type value indicates that ARP is the higher layer protocol?**

Type: ARP (0x0806)

---

## Task 1.2 (Understanding TCP network sockets)

---



A screenshot of a terminal window titled "ser321 -- zsh -- 80x24". The window shows a shell script being run. The script starts with "Last login: Tue Feb 1 14:11:38 on ttys001 shaunvulaj@shauns-MacBook-Air ser321 %". It then contains a loop: "while :; do". Inside the loop, it prints the current date, filters netstat output for established connections, counts them, and then does the same for listening connections. It then sleeps for 30 seconds. The script ends with "[done]" and a control-C interrupt (^C). The user then types "ser321 %".

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

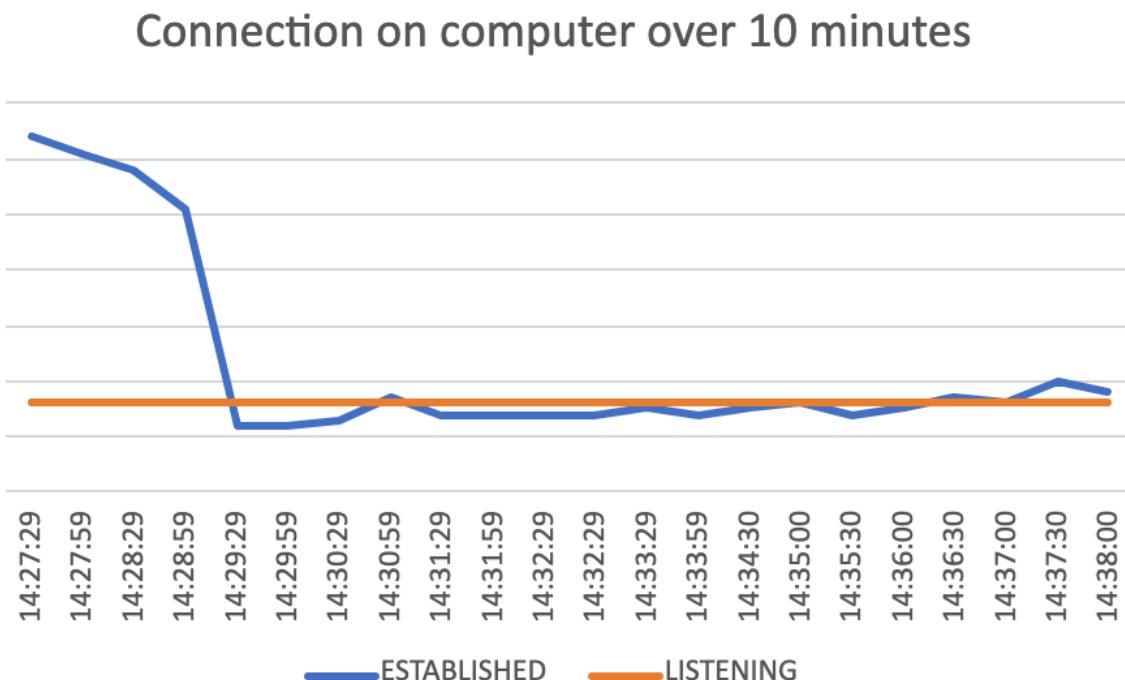


Figure 9\* Graph displaying read data from 10 minutes of activity on my local computers network

---

---

### Task 1.3 (Sniffing TCP/UDP traffic)

---

---

#### Step 1 (TCP):

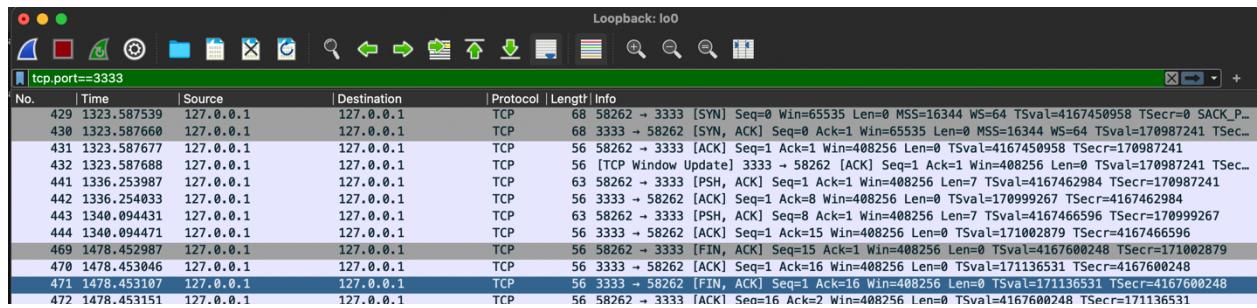


Figure 10\* This image displays the network traffic with TCP filter applied in wireshark

**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):

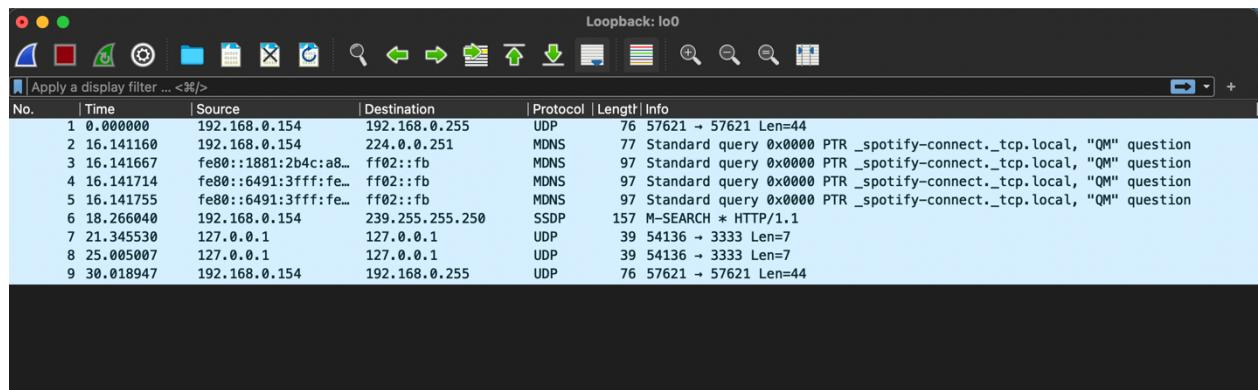


Figure 11\* This image displays the network traffic with UDP filter applied in Wireshark

- 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

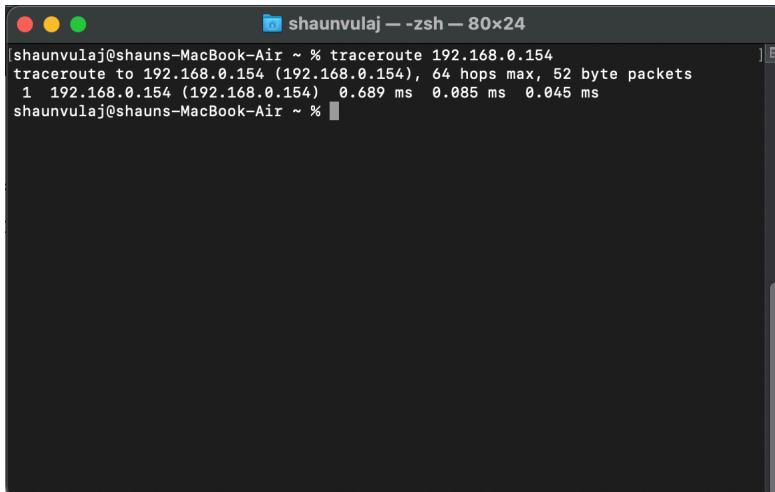
g) How much overhead was there. Basically, how many bytes was the whole process compared to the actually data that we did send.

741

---

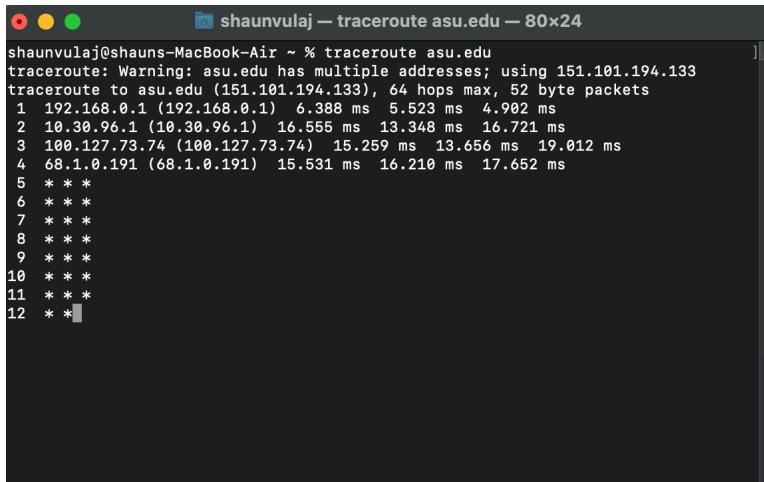
## Task 1.4. Internet Protocol (IP) Routing

---

A screenshot of a macOS terminal window titled "shaunvulaj -- zsh -- 80x24". The command "traceroute 192.168.0.154" is run, showing the path from the user's local machine to a destination at 192.168.0.154. The output shows one hop with a latency of 0.689 ms.

```
[shaunvulaj@shauns-MacBook-Air ~ % traceroute 192.168.0.154
traceroute to 192.168.0.154 (192.168.0.154), 64 hops max, 52 byte packets
1  192.168.0.154 (192.168.0.154)  0.689 ms  0.085 ms  0.045 ms
shaunvulaj@shauns-MacBook-Air ~ % ]
```

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

A screenshot of a macOS terminal window titled "shaunvulaj — traceroute asu.edu — 80x24". The command "traceroute asu.edu" is run, showing the path from the user's local machine to a website at asu.edu. The output shows 12 hops, with the first four being valid and the rest being asterisks indicating network unreachable or other issues.

```
shaunvulaj@shauns-MacBook-Air ~ % traceroute asu.edu
traceroute: Warning: asu.edu has multiple addresses; using 151.101.194.133
traceroute to asu.edu (151.101.194.133), 64 hops max, 52 byte packets
1  192.168.0.1 (192.168.0.1)  6.388 ms  5.523 ms  4.902 ms
2  10.30.96.1 (10.30.96.1)  16.555 ms  13.348 ms  16.721 ms
3  100.127.73.74 (100.127.73.74)  15.259 ms  13.656 ms  19.012 ms
4  68.1.0.191 (68.1.0.191)  15.531 ms  16.210 ms  17.652 ms
5  * * *
6  * * *
7  * * *
8  * * *
9  * * *
10 * * *
11 * * *
12 * *
```

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=l\\_SMshzFAPg&ab\\_channel=ShaunVulaj](https://www.youtube.com/watch?v=l_SMshzFAPg&ab_channel=ShaunVulaj)

### Task 1.5.2

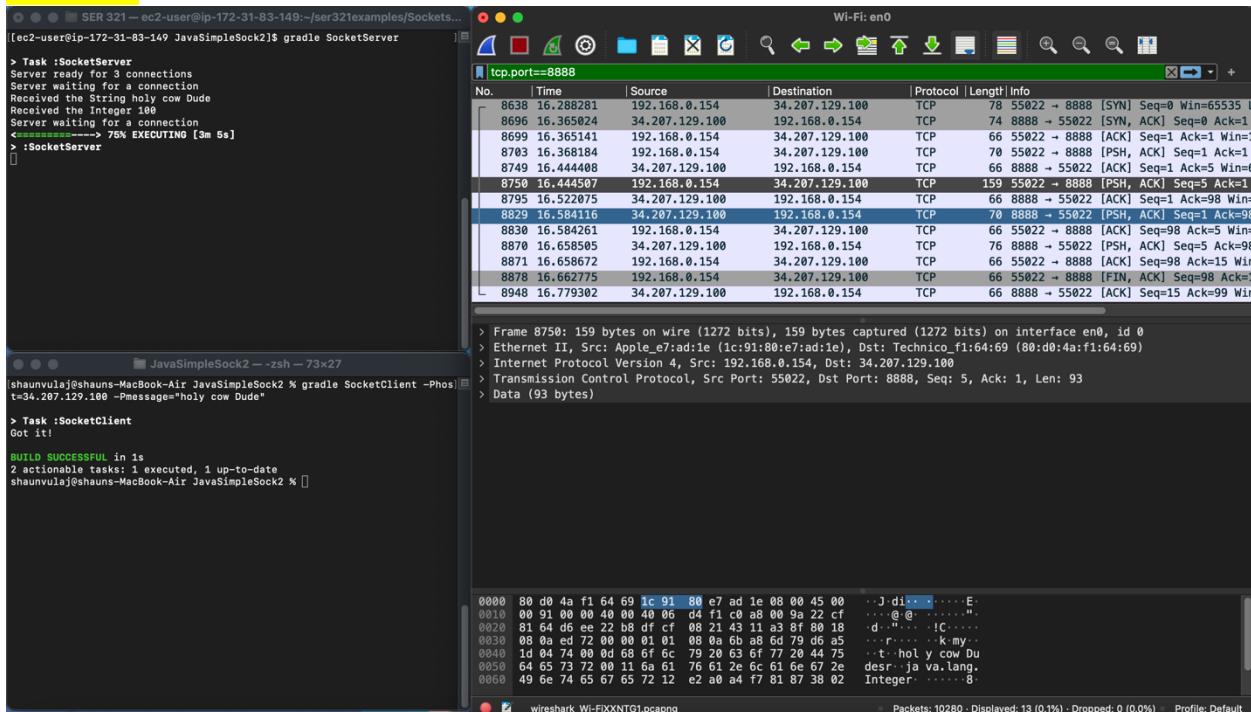


Figure 14\* Image displaying the network activity between my local computer and my AWS server with Wireshark tool being used for analysis

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

```
Last login: Sun Jan 30 17:24:08 on ttys000
shaunvalj@shauns-MacBook-Air JavaSimpleSock2 % gradle SocketServer

> Task :SocketServer
Server ready for 3 connections
Server waiting for a connection
Received the String HI
Received the Integer 100
Server waiting for a connection
Received the String Shaunv
Received the Integer 100
Server waiting for a connection
* * * * * 75% EXECUTING [2 m 29s]
:SocketServer

Last login: Sun Jan 30 16:33:07 on ttys001
shaunvalj@shauns-MacBook-Air JavaSimpleSock2 % gradle SocketClient
Starting a Gradle Daemon, 1 busy Daemon could not be reused, use --status for details
> Task :SocketClient
Got it!

BUILD SUCCESSFUL in 7s
2 actionable tasks: 1 executed, 1 up-to-date
shaunvalj@shauns-MacBook-Air JavaSimpleSock2 % gradle SocketClient -Pmessage="S"
shaunvalj@localhost:~ % 

> Task :SocketClient
Got it!

BUILD SUCCESSFUL in 1s
2 actionable tasks: 1 executed, 1 up-to-date
shaunvalj@shauns-MacBook-Air JavaSimpleSock2 % 
```

Capturing from Loopback: lo0

tcp.port==8888

No.	Time	Source	Destination	Protocol	Length	Info
1457	55.741192	127.0.0.1	127.0.0.1	TCP	68	53681 - 8888 [SYN] Seq=0 Win=
1459	55.741293	127.0.0.1	127.0.0.1	TCP	68	8888 - 53681 [SYN, ACK] Seq=1 Ack=1
1459	55.741306	127.0.0.1	127.0.0.1	TCP	56	53681 - 8888 [ACK] Seq=1 Ack=2
1460	55.741316	127.0.0.1	127.0.0.1	TCP	56	[TCP Window Update] 8888 - 53681
1461	55.742543	127.0.0.1	127.0.0.1	TCP	60	53681 - 8888 [PSH, ACK] Seq=1 Ack=2
1462	55.742622	127.0.0.1	127.0.0.1	TCP	56	8888 - 53681 [ACK] Seq=1 Ack=3
1463	55.747194	127.0.0.1	127.0.0.1	TCP	65	53681 - 8888 [PSH, ACK] Seq=1 Ack=2
1464	55.747270	127.0.0.1	127.0.0.1	TCP	56	8888 - 53681 [ACK] Seq=1 Ack=3
1465	55.750283	127.0.0.1	127.0.0.1	TCP	96	53681 - 8888 [PSH, ACK] Seq=1 Ack=4
1466	55.750376	127.0.0.1	127.0.0.1	TCP	56	8888 - 53681 [ACK] Seq=1 Ack=5
1467	55.753266	127.0.0.1	127.0.0.1	TCP	87	53681 - 8888 [PSH, ACK] Seq=1 Ack=6
1468	55.753781	127.0.0.1	127.0.0.1	TCP	56	8888 - 53681 [ACK] Seq=1 Ack=7
1469	55.754612	127.0.0.1	127.0.0.1	TCP	58	53681 - 8888 [PSH, ACK] Seq=1 Ack=8

> Frame 1463: 65 bytes on wire (520 bits), 65 bytes captured (520 bits) on interface lo0, id 0

> Null/Loopback

> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

> Transmission Control Protocol, Src Port: 53681, Dst Port: 8888, Seq: 5, Ack: 1, Len: 9

> Data (9 bytes)

0000 02 00 00 00 05 00 00 3d 00 00 40 00 04 00 06 00 00 .E. = @ @ ..

0010 7f 00 00 01 ff 00 00 01 d1 b1 22 b8 ea 85 c2 5a .D'e?.....1.....Z

0020 44 22 65 3f 00 18 1b fe 31 00 01 01 00 08 0a

0030 4b 07 a5 8c d9 ea ff a3 74 00 06 53 68 61 75 6e K t -Shaun

0040 76 V

Packets: 29985 - Displayed: 22 (0.7%) Profile: Default

## 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 it 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.