Saad S Khan

Professor Mehlhase

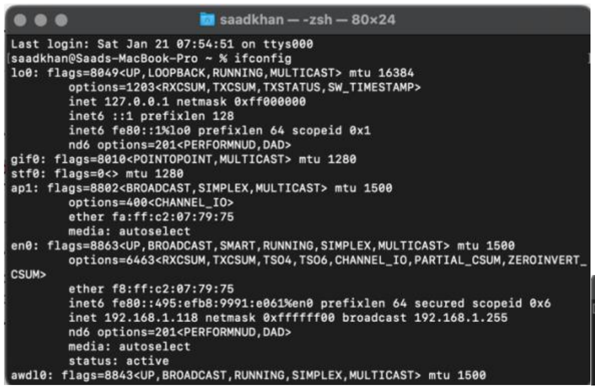
January 12st 2023

SER321

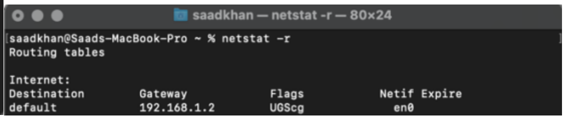
# Assignment 1 Part 2

Task 1:

1. Screen captures of my calls to identify your network interface and gateway.

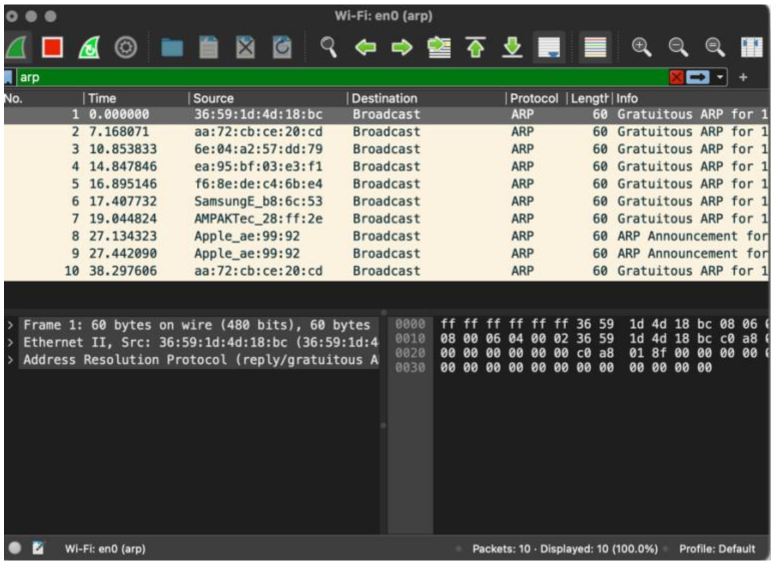


```
Last login: Sat Jan 21 07:54:51 on ttys000
saadkhan@Saads-MacBook-Pro ~ % ifconfig
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
        options=1203<RXCSUM,TXCSUM,TXSTATUS,SW_TIMESTAMP>
        inet 127.0.0.1 netmask 0xff000000
        inet6 ::1 prefixlen 128
        inet6 fe80::1%lo0 prefixlen 64 scopeid 0x1
        nd6 options=201<PERFORMNUD,DAD>
gif0: flags=8010<POINTOPOINT,MULTICAST> mtu 1280
stf0: flags=0<> mtu 1280
ap1: flags=8802<BROADCAST,SIMPLEX,MULTICAST> mtu 1500
        options=400<CHANNEL_IO>
        ether fa:ff:c2:07:79:75
        media: autoselect
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        options=6463<RXCSUM,TXCSUM,TSO4,TSO6,CHANNEL_IO,PARTIAL_CSUM,ZEROINVERT_
CSUM>
        ether f8:ff:c2:07:79:75
        inet6 fe80::495:efb8:9991:e061%en0 prefixlen 64 secured scopeid 0x6
        inet 192.168.1.118 netmask 0xffffff00 broadcast 192.168.1.255
        nd6 options=201<PERFORMNUD,DAD>
        media: autoselect
        status: active
awdl0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
```



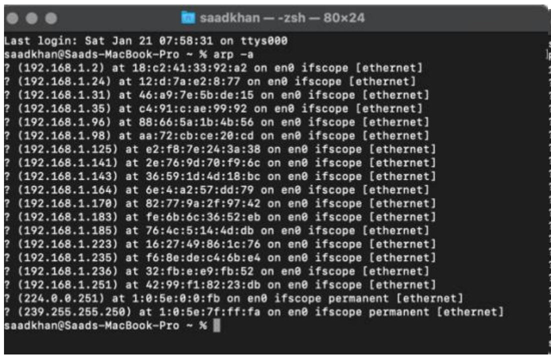
```
saadkhan@Saads-MacBook-Pro ~ % netstat -r
Routing tables

Internet:
Destination        Gateway            Flags           Netif Expire
default            192.168.1.2        UGScg             en0
```

A screen capture of my Wireshark instance with the appropriate filters also being visible.

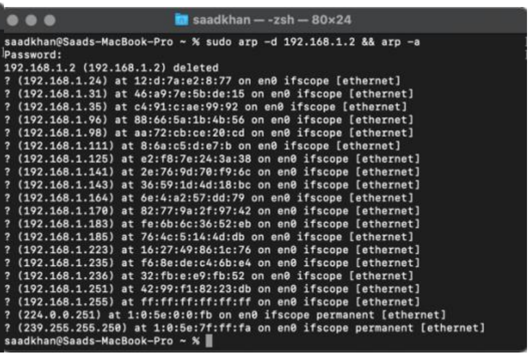


```
arp
No.    Time        Source              Destination   Protocol  Length Info
     1 0.000000    36:59:1d:4d:18:bc   Broadcast     ARP       60 Gratuitous ARP for 1
     2 7.168071    aa:72:cb:ce:20:cd   Broadcast     ARP       60 Gratuitous ARP for 1
     3 10.853833   6e:04:a2:57:dd:79   Broadcast     ARP       60 Gratuitous ARP for 1
     4 14.847846   ea:95:bf:03:e3:f1   Broadcast     ARP       60 Gratuitous ARP for 1
     5 16.895146   f6:8e:de:c4:6b:e4   Broadcast     ARP       60 Gratuitous ARP for 1
     6 17.407732   SamsungE_b8:6c:53   Broadcast     ARP       60 Gratuitous ARP for 1
     7 19.044824   AMPAKTec_28:ff:2e   Broadcast     ARP       60 Gratuitous ARP for 1
     8 27.134323   Apple_ae:99:92      Broadcast     ARP       60 ARP Announcement for
     9 27.442090   Apple_ae:99:92      Broadcast     ARP       60 ARP Announcement for
    10 38.297606   aa:72:cb:ce:20:cd   Broadcast     ARP       60 Gratuitous ARP for 1

> Frame 1: 60 bytes on wire (480 bits), 60 bytes     0000  ff ff ff ff ff ff 36 59  1d 4d 18 bc 08 06
> Ethernet II, Src: 36:59:1d:4d:18:bc (36:59:1d:4    0010  08 00 06 04 00 02 36 59  1d 4d 18 bc c0 a8
> Address Resolution Protocol (reply/gratuitous A    0020  00 00 00 00 00 00 c0 a8  01 8f 00 00 00 00
                                                     0030  00 00 00 00 00 00 00 00  00 00 00 00

Wi-Fi: en0 (arp)                         Packets: 10 · Displayed: 10 (100.0%)   Profile: Default
```

Screen captures of your arp -a and arp -d commands.

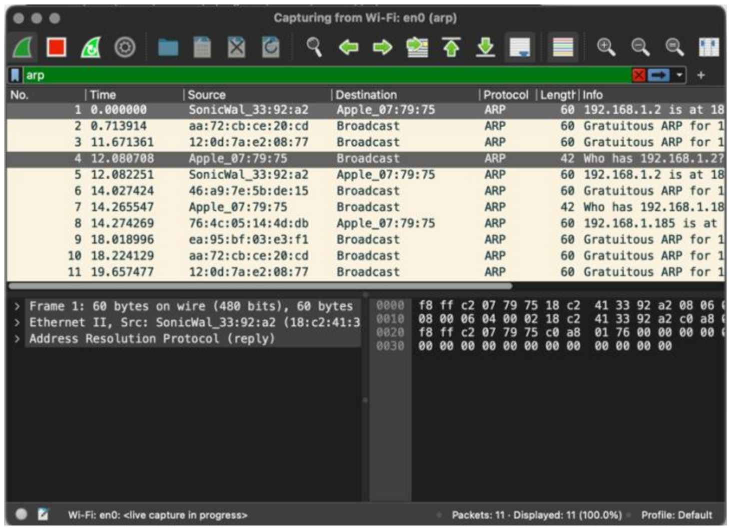


```
Last login: Sat Jan 21 07:58:31 on ttys000
saadkhan@Saads-MacBook-Pro ~ % arp -a
? (192.168.1.2) at 18:c2:41:33:92:a2 on en0 ifscope [ethernet]
? (192.168.1.24) at 12:d:7a:e2:8:77 on en0 ifscope [ethernet]
? (192.168.1.31) at 46:a9:7e:5b:de:15 on en0 ifscope [ethernet]
? (192.168.1.35) at c4:91:c:ae:99:92 on en0 ifscope [ethernet]
? (192.168.1.96) at 88:66:5a:1b:4b:56 on en0 ifscope [ethernet]
? (192.168.1.98) at aa:72:cb:ce:20:cd on en0 ifscope [ethernet]
? (192.168.1.125) at e2:f8:7e:24:3a:38 on en0 ifscope [ethernet]
? (192.168.1.141) at 2e:76:9d:70:f9:6c on en0 ifscope [ethernet]
? (192.168.1.143) at 36:59:1d:4d:18:bc on en0 ifscope [ethernet]
? (192.168.1.164) at 6e:4:a2:57:dd:79 on en0 ifscope [ethernet]
? (192.168.1.170) at 82:77:9a:2f:97:42 on en0 ifscope [ethernet]
? (192.168.1.183) at fe:6b:6c:36:52:eb on en0 ifscope [ethernet]
? (192.168.1.185) at 76:4c:5:14:4d:db on en0 ifscope [ethernet]
? (192.168.1.223) at 16:27:49:86:1c:76 on en0 ifscope [ethernet]
? (192.168.1.235) at f6:8e:de:c4:6b:e4 on en0 ifscope [ethernet]
? (192.168.1.236) at 32:fb:e:e9:fb:52 on en0 ifscope [ethernet]
? (192.168.1.251) at 42:99:f1:82:23:db on en0 ifscope [ethernet]
? (224.0.0.251) at 1:0:5e:0:0:fb on en0 ifscope permanent [ethernet]
? (239.255.255.250) at 1:0:5e:7f:ff:fa on en0 ifscope permanent [ethernet]
saadkhan@Saads-MacBook-Pro ~ %
```



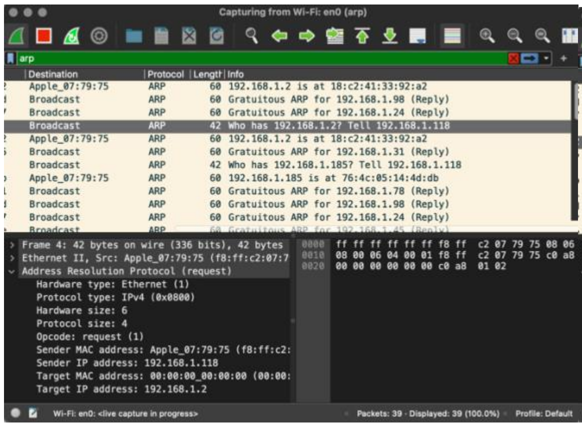
```
saadkhan@Saads-MacBook-Pro ~ % sudo arp -d 192.168.1.2 && arp -a
Password:
192.168.1.2 (192.168.1.2) deleted
? (192.168.1.24) at 12:d:7a:e2:8:77 on en0 ifscope [ethernet]
? (192.168.1.31) at 46:a9:7e:5b:de:15 on en0 ifscope [ethernet]
? (192.168.1.35) at c4:91:c:ae:99:92 on en0 ifscope [ethernet]
? (192.168.1.96) at 88:66:5a:1b:4b:56 on en0 ifscope [ethernet]
? (192.168.1.98) at aa:72:cb:ce:20:cd on en0 ifscope [ethernet]
? (192.168.1.111) at 8:6a:c5:d:e7:b on en0 ifscope [ethernet]
? (192.168.1.125) at e2:f8:7e:24:3a:38 on en0 ifscope [ethernet]
? (192.168.1.141) at 2e:76:9d:70:f9:6c on en0 ifscope [ethernet]
? (192.168.1.143) at 36:59:1d:4d:18:bc on en0 ifscope [ethernet]
? (192.168.1.164) at 6e:4:a2:57:dd:79 on en0 ifscope [ethernet]
? (192.168.1.170) at 82:77:9a:2f:97:42 on en0 ifscope [ethernet]
? (192.168.1.183) at fe:6b:6c:36:52:eb on en0 ifscope [ethernet]
? (192.168.1.185) at 76:4c:5:14:4d:db on en0 ifscope [ethernet]
? (192.168.1.223) at 16:27:49:86:1c:76 on en0 ifscope [ethernet]
? (192.168.1.235) at f6:8e:de:c4:6b:e4 on en0 ifscope [ethernet]
? (192.168.1.236) at 32:fb:e:e9:fb:52 on en0 ifscope [ethernet]
? (192.168.1.251) at 42:99:f1:82:23:db on en0 ifscope [ethernet]
? (192.168.1.255) at ff:ff:ff:ff:ff:ff on en0 ifscope [ethernet]
? (224.0.0.251) at 1:0:5e:0:0:fb on en0 ifscope permanent [ethernet]
? (239.255.255.250) at 1:0:5e:7f:ff:fa on en0 ifscope permanent [ethernet]
saadkhan@Saads-MacBook-Pro ~ %
```

Screen capture of the updated trace in Wireshark.

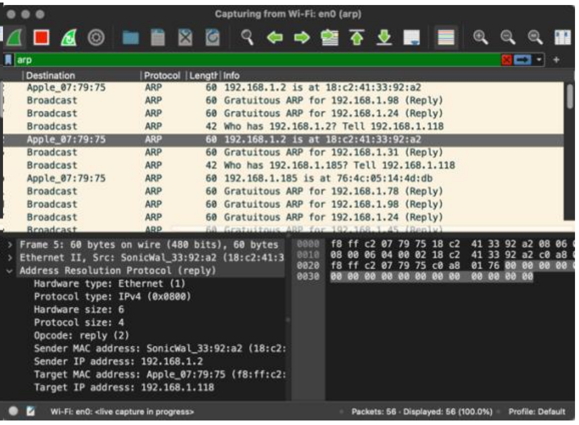


```
No.   Time        Source              Destination      Protocol Length Info
  1   0.000000    SonicWal_33:92:a2   Apple_07:79:75   ARP      60  192.168.1.2 is at 18
  2   0.713914    aa:72:cb:ce:20:cd   Broadcast        ARP      60  Gratuitous ARP for 1
  3   11.671361   12:0d:7a:e2:08:77   Broadcast        ARP      60  Gratuitous ARP for 1
  4   12.080708   Apple_07:79:75      Broadcast        ARP      42  Who has 192.168.1.2?
  5   12.082251   SonicWal_33:92:a2   Apple_07:79:75   ARP      60  192.168.1.2 is at 18
  6   14.027424   46:a9:7e:5b:de:15   Broadcast        ARP      60  Gratuitous ARP for 1
  7   14.265547   Apple_07:79:75      Broadcast        ARP      42  Who has 192.168.1.18
  8   14.274269   76:4c:05:14:4d:db   Apple_07:79:75   ARP      60  192.168.1.185 is at
  9   18.018996   ea:95:bf:03:e3:f1   Broadcast        ARP      60  Gratuitous ARP for 1
 10   18.224129   aa:72:cb:ce:20:cd   Broadcast        ARP      60  Gratuitous ARP for 1
 11   19.657477   12:0d:7a:e2:08:77   Broadcast        ARP      60  Gratuitous ARP for 1

> Frame 1: 60 bytes on wire (480 bits), 60 bytes     0000  f8 ff c2 07 79 75 18 c2  41 33 92 a2 08 06
> Ethernet II, Src: SonicWal_33:92:a2 (18:c2:41:3    0010  08 00 06 04 00 02 18 c2  41 33 92 a2 c0 a8
> Address Resolution Protocol (reply)                0020  f8 ff c2 07 79 75 c0 a8  01 76 00 00 00 00
                                                     0030  00 00 00 00 00 00 00 00  00 00 00 00

Wi-Fi: en0: <live capture in progress>        Packets: 11 · Displayed: 11 (100.0%)   Profile: Default
```

2. Screen capture of the ARP request and reply.



```
Destination      Protocol Length Info
Apple_07:79:75   ARP      60  192.168.1.2 is at 18:c2:41:33:92:a2
Broadcast        ARP      60  Gratuitous ARP for 192.168.1.98 (Reply)
Broadcast        ARP      60  Gratuitous ARP for 192.168.1.24 (Reply)
Broadcast        ARP      42  Who has 192.168.1.2? Tell 192.168.1.118
Apple_07:79:75   ARP      60  192.168.1.2 is at 18:c2:41:33:92:a2
Broadcast        ARP      60  Gratuitous ARP for 192.168.1.31 (Reply)
Broadcast        ARP      42  Who has 192.168.1.185? Tell 192.168.1.118
Apple_07:79:75   ARP      60  192.168.1.185 is at 76:4c:05:14:4d:db
Broadcast        ARP      60  Gratuitous ARP for 192.168.1.78 (Reply)
Broadcast        ARP      60  Gratuitous ARP for 192.168.1.98 (Reply)
Broadcast        ARP      60  Gratuitous ARP for 192.168.1.24 (Reply)
Broadcast        ARP      60  Gratuitous ARP for 192.168.1.45 (Reply)

> Frame 4: 42 bytes on wire (336 bits), 42 bytes     0000  ff ff ff ff ff ff f8 ff  c2 07 79 75 08 06
> Ethernet II, Src: Apple_07:79:75 (f8:ff:c2:07:7    0010  08 00 06 04 00 01 f8 ff  c2 07 79 75 c0 a8
v Address Resolution Protocol (request)              0020  00 00 00 00 00 00 c0 a8  01 02
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: Apple_07:79:75 (f8:ff:c2:
    Sender IP address: 192.168.1.118
    Target MAC address: 00:00:00_00:00:00 (00:00:
    Target IP address: 192.168.1.2

Wi-Fi: en0: <live capture in progress>        Packets: 39 · Displayed: 39 (100.0%)   Profile: Default
```



```
Destination      Protocol Length Info
Apple_07:79:75   ARP      60  192.168.1.2 is at 18:c2:41:33:92:a2
Broadcast        ARP      60  Gratuitous ARP for 192.168.1.98 (Reply)
Broadcast        ARP      60  Gratuitous ARP for 192.168.1.24 (Reply)
Broadcast        ARP      42  Who has 192.168.1.2? Tell 192.168.1.118
Apple_07:79:75   ARP      60  192.168.1.2 is at 18:c2:41:33:92:a2
Broadcast        ARP      60  Gratuitous ARP for 192.168.1.31 (Reply)
Broadcast        ARP      42  Who has 192.168.1.185? Tell 192.168.1.118
Apple_07:79:75   ARP      60  192.168.1.185 is at 76:4c:05:14:4d:db
Broadcast        ARP      60  Gratuitous ARP for 192.168.1.78 (Reply)
Broadcast        ARP      60  Gratuitous ARP for 192.168.1.98 (Reply)
Broadcast        ARP      60  Gratuitous ARP for 192.168.1.24 (Reply)
Broadcast        ARP      60  Gratuitous ARP for 192.168.1.45 (Reply)

> Frame 5: 60 bytes on wire (480 bits), 60 bytes     0000  f8 ff c2 07 79 75 18 c2  41 33 92 a2 08 06
> Ethernet II, Src: SonicWal_33:92:a2 (18:c2:41:3    0010  08 00 06 04 00 02 18 c2  41 33 92 a2 c0 a8
v Address Resolution Protocol (reply)                0020  f8 ff c2 07 79 75 c0 a8  01 76 00 00 00 00
    Hardware type: Ethernet (1)                      0030  00 00 00 00 00 00 00 00  00 00 00 00
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: reply (2)
    Sender MAC address: SonicWal_33:92:a2 (18:c2:
    Sender IP address: 192.168.1.2
    Target MAC address: Apple_07:79:75 (f8:ff:c2:
    Target IP address: 192.168.1.118

Wi-Fi: en0: <live capture in progress>        Packets: 56 · Displayed: 56 (100.0%)   Profile: Default
```

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

   1 is used to indicate a request and 2 is used to indicate a reply.

   2. How large is the ARP header for a request? What about for a reply? You will need to research this.

   The ARP header for both request and reply is 28 bytes for IPv4.

   Source: https://www.practicalnetworking.net/series/arp/traditional-arp/

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

   All zeros or 00:00:00:00:00:00

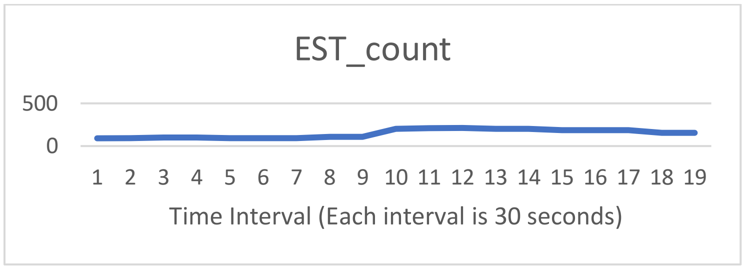
   4. What Ethernet Type value indicates that ARP is the higher layer protocol?

   0x806

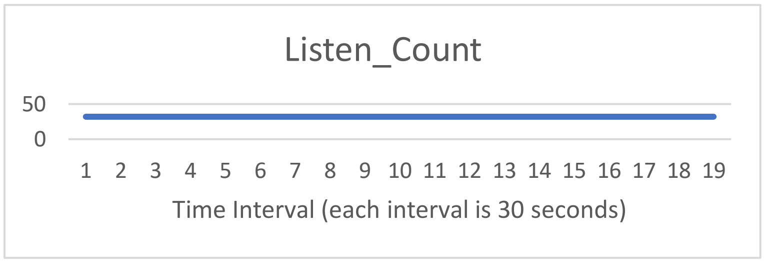
Task 1.2:

**Command:** watch -n 30 "date >> netstat_log.txt; netstat -an | grep -E 'ESTABLISHED|LISTEN' >> netstat_log.txt

Graphs:

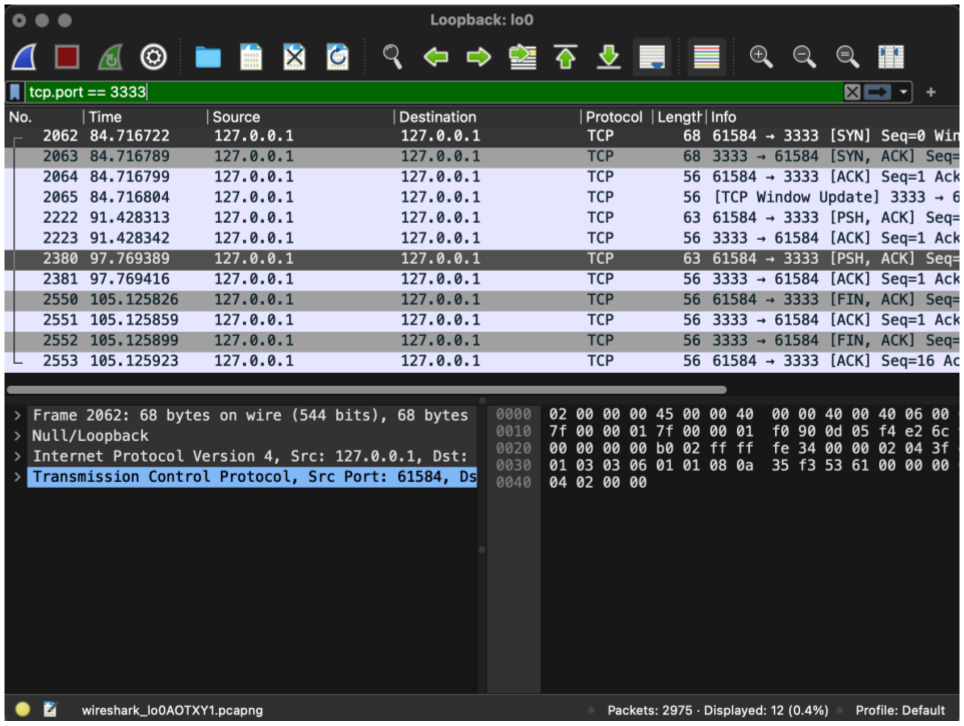


Established connection count



Listening connection count

Task 1.3:



Loopback: lo0

tcp.port == 3333

```
No.   Time        Source      Destination   Protocol Lengtl Info
 2062 84.716722   127.0.0.1   127.0.0.1     TCP        68 61584 → 3333 [SYN] Seq=0 Wi
 2063 84.716789   127.0.0.1   127.0.0.1     TCP        68 3333 → 61584 [SYN, ACK] Seq=
 2064 84.716799   127.0.0.1   127.0.0.1     TCP        56 61584 → 3333 [ACK] Seq=1 Ack
 2065 84.716804   127.0.0.1   127.0.0.1     TCP        56 [TCP Window Update] 3333 → 6
 2222 91.428313   127.0.0.1   127.0.0.1     TCP        63 61584 → 3333 [PSH, ACK] Seq=
 2223 91.428342   127.0.0.1   127.0.0.1     TCP        56 3333 → 61584 [ACK] Seq=1 Ack
 2380 97.769389   127.0.0.1   127.0.0.1     TCP        63 61584 → 3333 [PSH, ACK] Seq=
 2381 97.769416   127.0.0.1   127.0.0.1     TCP        56 3333 → 61584 [ACK] Seq=1 Ack
 2550 105.125826  127.0.0.1   127.0.0.1     TCP        56 61584 → 3333 [FIN, ACK] Seq=
 2551 105.125859  127.0.0.1   127.0.0.1     TCP        56 3333 → 61584 [ACK] Seq=1 Ack
 2552 105.125899  127.0.0.1   127.0.0.1     TCP        56 3333 → 61584 [FIN, ACK] Seq=
 2553 105.125923  127.0.0.1   127.0.0.1     TCP        56 61584 → 3333 [ACK] Seq=16 Ac
```

```
> Frame 2062: 68 bytes on wire (544 bits), 68 bytes
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst:
> Transmission Control Protocol, Src Port: 61584, Ds
```

```
0000  02 00 00 00 45 00 00 40  00 00 40 00 40 06 00
0010  7f 00 00 01 7f 00 00 01  f0 90 0d 05 f4 e2 6c
0020  00 00 00 00 b0 02 ff ff  fe 34 00 00 02 04 3f
0030  01 03 03 06 01 01 08 0a  35 f3 53 61 00 00 00
0040  04 02 00 00
```

wireshark_lo0AOTXY1.pcapng    Packets: 2975 · Displayed: 12 (0.4%)    Profile: Default

WireShark screenshot for TCP

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

   The command nc -k -l 3333 is used to start a network service to listen for incoming connections on port 3333. The -k flag tells the service to keep listening for new clients even after a client disconnects. The -l flag tells the service to listen for connections on port 3333 instead of starting an outbound connection

   The command nc 127.0.0.1 3333 is used to connect to port 3333 with the localhost IP address 127.0.0.1.

b) How many frames were sent back and forth to capture these 2 lines?

   2

c) How many packets were sent 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?

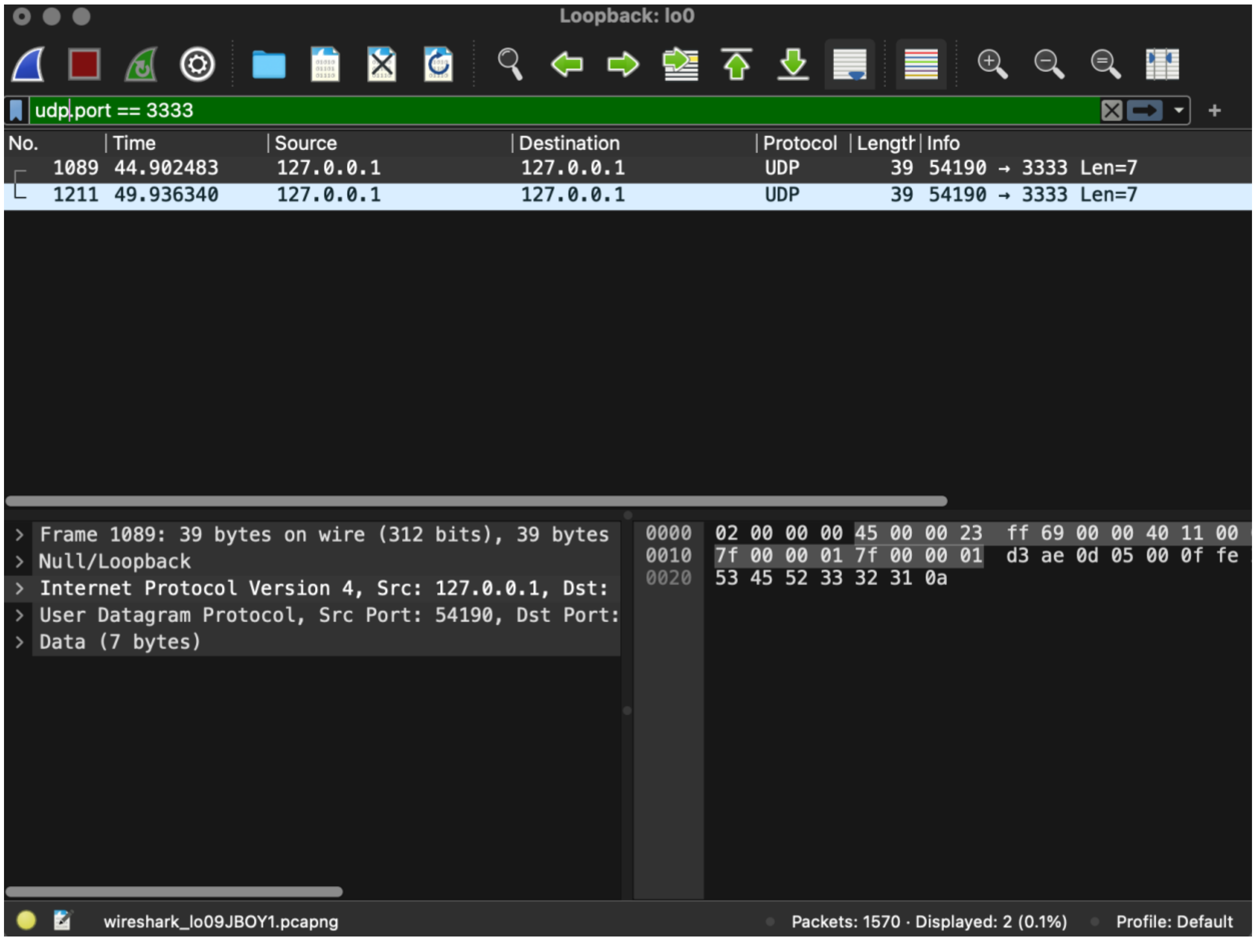
   14

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

   126

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

   112 bytes



```
udp.port == 3333

No.   Time       Source      Destination  Protocol Length Info
1089  44.902483  127.0.0.1   127.0.0.1    UDP          39 54190 → 3333 Len=7
1211  49.936340  127.0.0.1   127.0.0.1    UDP          39 54190 → 3333 Len=7

> Frame 1089: 39 bytes on wire (312 bits), 39 bytes
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst:
> User Datagram Protocol, Src Port: 54190, Dst Port:
> Data (7 bytes)

0000  02 00 00 00 45 00 00 23  ff 69 00 00 40 11 00
0010  7f 00 00 01 7f 00 00 01  d3 ae 0d 05 00 0f fe
0020  53 45 52 33 32 31 0a

wireshark_lo09JBOY1.pcapng    Packets: 1570 · Displayed: 2 (0.1%)    Profile: Default
```

WireShark screenshot for UDP.

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

The command "nc -k -l -u 3333" runs the netcat utility in listen mode on UDP port, and the "-k" option allows the server to continue running even after a client disconnects. It will wait for incoming connections on port 3333 and keep listening even if a client disconnects. The command "nc -u 127.0.0.1 3333" runs the netcat utility and connects to the specified IP address "127.0.0.1" using the UDP protocol on port 3333. It will initiate a connection to the specified IP address and port using UDP.

b) How many frames were needed to capture those 2 lines?

   2

c) How many packets were needed to capture those 2 lines?

   2

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

   2

e) How many total bytes went over the wire?

   78

f) How many bytes is the data (only the data) that was sent?

   14

g) Basically, how many bytes was the whole process compared to the actual data that we did send.?

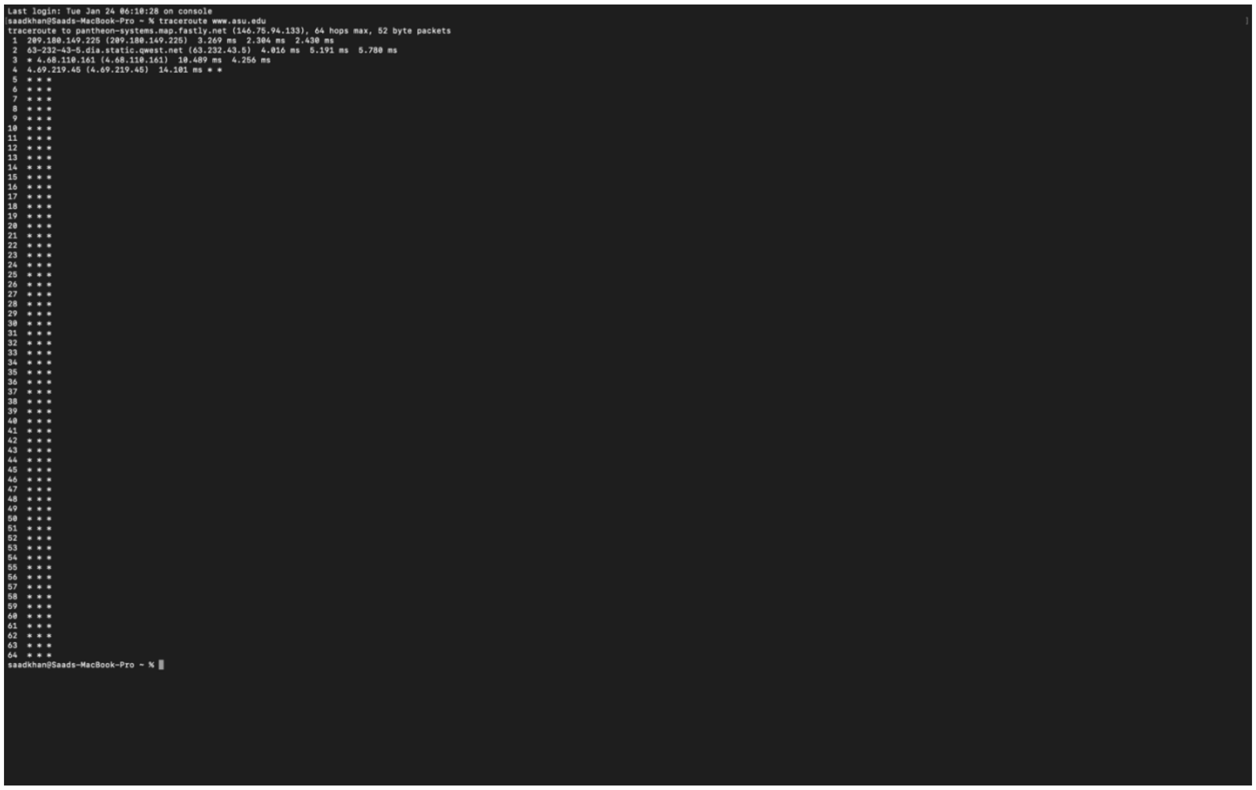
   64

h) What is the difference in relative overhead between UDP and TCP and why? Specifically, what kind of information was exchanged in TCP that was not exchanged in UDP? Show the relative parts of the packet traces.

   UDP is a connectionless protocol, while TCP is a connection-oriented protocol. In UDP, there is no overhead for establishing a connection before sending data, whereas in TCP, a three-way handshake is required to establish a connection before any data can be exchanged. a UDP packet will have a smaller header as compared to TCP. The header in a UDP packet contains source and destination port, length and checksum. While in TCP, the header contains source and destination port, sequence and acknowledgement numbers, control flags and window size.

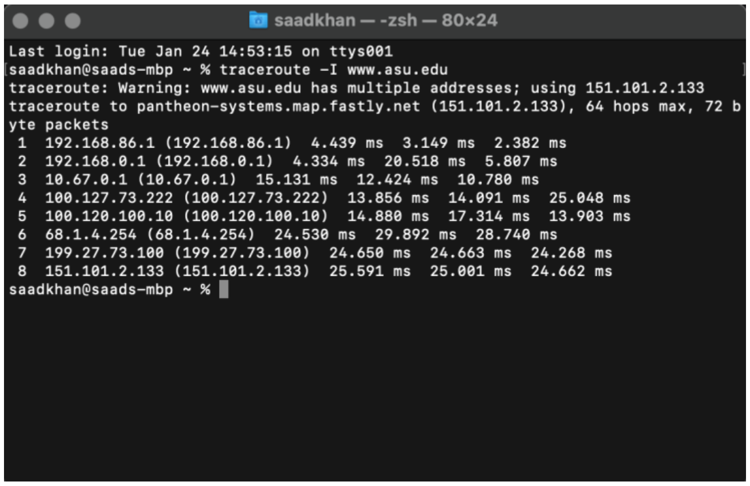
Task 1.4:

Route 1: Subaru dealership network



```
Last login: Tue Jan 24 06:10:28 on console
saadkhan@Saads-MacBook-Pro ~ % traceroute www.asu.edu
traceroute to pantheon-systems.map.fastly.net (146.75.94.133), 64 hops max, 52 byte packets
 1  209.180.149.225 (209.180.149.225)  3.269 ms  2.304 ms  2.430 ms
 2  63-232-43-5.dia.static.qwest.net (63.232.43.5)  4.016 ms  5.191 ms  5.780 ms
 3  * 4.68.110.161 (4.68.110.161)  10.489 ms  4.256 ms
 4  4.69.219.45 (4.69.219.45)  14.101 ms * *
 5  * * *
 6  * * *
 7  * * *
 8  * * *
 9  * * *
10  * * *
11  * * *
12  * * *
13  * * *
14  * * *
15  * * *
16  * * *
17  * * *
18  * * *
19  * * *
20  * * *
21  * * *
22  * * *
23  * * *
24  * * *
25  * * *
26  * * *
27  * * *
28  * * *
29  * * *
30  * * *
31  * * *
32  * * *
33  * * *
34  * * *
35  * * *
36  * * *
37  * * *
38  * * *
39  * * *
40  * * *
41  * * *
42  * * *
43  * * *
44  * * *
45  * * *
46  * * *
47  * * *
48  * * *
49  * * *
50  * * *
51  * * *
52  * * *
53  * * *
54  * * *
55  * * *
56  * * *
57  * * *
58  * * *
59  * * *
60  * * *
61  * * *
62  * * *
63  * * *
64  * * *
saadkhan@Saads-MacBook-Pro ~ %
```

Route 2: Home network



saadkhan — -zsh — 80×24

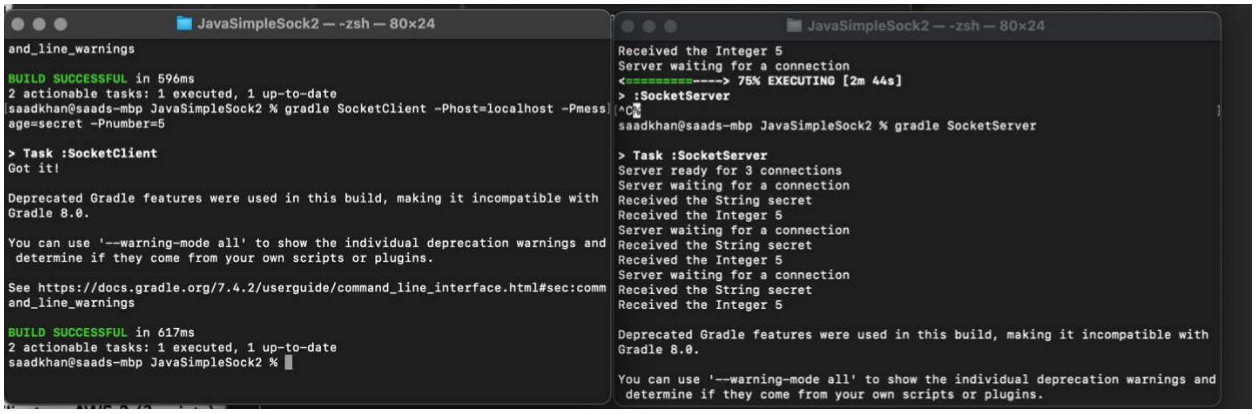
```
Last login: Tue Jan 24 14:53:15 on ttys001
saadkhan@saads-mbp ~ % traceroute -I www.asu.edu
traceroute: Warning: www.asu.edu has multiple addresses; using 151.101.2.133
traceroute to pantheon-systems.map.fastly.net (151.101.2.133), 64 hops max, 72 b
yte packets
 1  192.168.86.1 (192.168.86.1)  4.439 ms  3.149 ms  2.382 ms
 2  192.168.0.1 (192.168.0.1)  4.334 ms  20.518 ms  5.807 ms
 3  10.67.0.1 (10.67.0.1)  15.131 ms  12.424 ms  10.780 ms
 4  100.127.73.222 (100.127.73.222)  13.856 ms  14.091 ms  25.048 ms
 5  100.120.100.10 (100.120.100.10)  14.880 ms  17.314 ms  13.903 ms
 6  68.1.4.254 (68.1.4.254)  24.530 ms  29.892 ms  28.740 ms
 7  199.27.73.100 (199.27.73.100)  24.650 ms  24.663 ms  24.268 ms
 8  151.101.2.133 (151.101.2.133)  25.591 ms  25.001 ms  24.662 ms
saadkhan@saads-mbp ~ %
```

a) Home network is fastest.

b) Home network has fewest hops

Task 1.5.1:

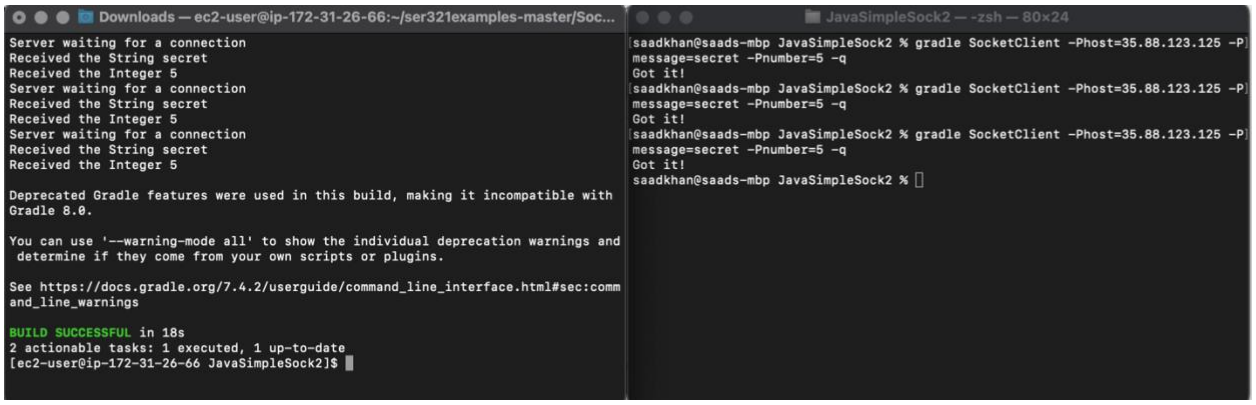
Screencast link: [https://youtu.be/Qsg8wvzCLq0](https://youtu.be/Qsg8wvzCLq0)


JavaSimpleSock2 — -zsh — 80×24
JavaSimpleSock2 — -zsh — 80×24

```
and_line_warnings

BUILD SUCCESSFUL in 596ms
2 actionable tasks: 1 executed, 1 up-to-date
[saadkhan@saads-mbp JavaSimpleSock2 % gradle SocketClient -Phost=localhost -Pmess
age=secret -Pnumber=5

> Task :SocketClient
Got it!

Deprecated Gradle features were used in this build, making it incompatible with
Gradle 8.0.

You can use '--warning-mode all' to show the individual deprecation warnings and
 determine if they come from your own scripts or plugins.

See https://docs.gradle.org/7.4.2/userguide/command_line_interface.html#sec:comm
and_line_warnings

BUILD SUCCESSFUL in 617ms
2 actionable tasks: 1 executed, 1 up-to-date
saadkhan@saads-mbp JavaSimpleSock2 %
```

```
Received the Integer 5
Server waiting for a connection
<=========----> 75% EXECUTING [2m 44s]
> :SocketServer
^C
saadkhan@saads-mbp JavaSimpleSock2 % gradle SocketServer

> Task :SocketServer
Server ready for 3 connections
Server waiting for a connection
Received the String secret
Received the Integer 5
Server waiting for a connection
Received the String secret
Received the Integer 5
Server waiting for a connection
Received the String secret
Received the Integer 5

Deprecated Gradle features were used in this build, making it incompatible with
Gradle 8.0.

You can use '--warning-mode all' to show the individual deprecation warnings and
 determine if they come from your own scripts or plugins.
```

Task 1.5.2:


Downloads — ec2-user@ip-172-31-26-66:~/ser321examples-master/Soc...
JavaSimpleSock2 — -zsh — 80×24

```
Server waiting for a connection
Received the String secret
Received the Integer 5
Server waiting for a connection
Received the String secret
Received the Integer 5
Server waiting for a connection
Received the String secret
Received the Integer 5

Deprecated Gradle features were used in this build, making it incompatible with
Gradle 8.0.

You can use '--warning-mode all' to show the individual deprecation warnings and
 determine if they come from your own scripts or plugins.

See https://docs.gradle.org/7.4.2/userguide/command_line_interface.html#sec:comm
and_line_warnings

BUILD SUCCESSFUL in 18s
2 actionable tasks: 1 executed, 1 up-to-date
[ec2-user@ip-172-31-26-66 JavaSimpleSock2]$
```

```
[saadkhan@saads-mbp JavaSimpleSock2 % gradle SocketClient -Phost=35.88.123.125 -P
message=secret -Pnumber=5 -q
Got it!
[saadkhan@saads-mbp JavaSimpleSock2 % gradle SocketClient -Phost=35.88.123.125 -P
message=secret -Pnumber=5 -q
Got it!
[saadkhan@saads-mbp JavaSimpleSock2 % gradle SocketClient -Phost=35.88.123.125 -P
message=secret -Pnumber=5 -q
Got it!
saadkhan@saads-mbp JavaSimpleSock2 %
```

To capture the traffic between my AWS server and my local machine, I had to switch from loop back to tracking the traffic on my wifi network. On gradle, I had to change the host value when sending data to the ipv4 address of the AWS server. On Wireshark, the number of packages sent between the server and client have increased and the message and number being sent were sent with some other data as well, unlike running locally where the message was the last thing sent in the package carrying it.

Task 1.5.3:

It is possible to run your server locally and your client on AWS. We need to configure your home computer to accept incoming connections from the internet by setting up a static IP and routing incoming connections to our home computer. Then we would need to run the server on our home computer and use the static IP to send data from the client to the server.

Task 1.5.4

The router plays the role of a gateway between your local network and the outside world. When a device wants to make a connection to the internet, the router sends the connection out with its out public IP address. When a request is sent from the internet, the router receives it and forwards it to the device it is meant for. The problem that arises when operating a server locally and attempting to connect to it from the outside is that the server's internal IP address cannot be accessed via the internet. We will need to set up your router to route incoming connections to the server's internal IP address if we want to connect to it from outside your network. The issue with this is that the connection to your local network can be used to gain access to your internal network for malicious purposes.