CS331 Project 1 Write Up – Ben Webb

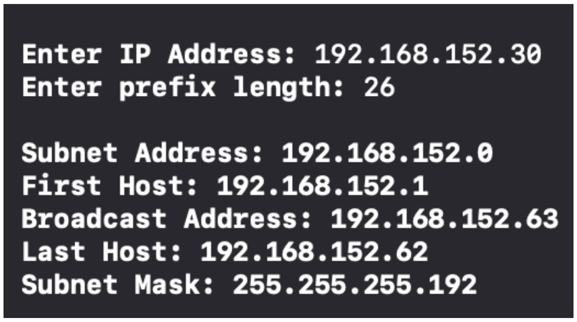
The purpose of this project was to create a program that could calculate IPv4 subnets by taking in the IPv4 address as well as the prefix length.

I decided to solve this problem in C++ by parsing an IPv4 char into bytes and using bitwise comparisons with the netmask that was created from the prefix length. The char was parsed using the command: `sscanf(ip, “%hu.%hu.%hu.%hu”, &a, &b, &c, &d)` which I found on the StackOverflow. ip is a char that is parsed into 4 unsigned shorts: a, b, c, and d.

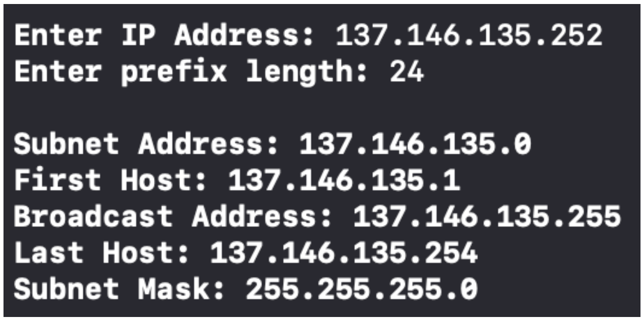
To create the subnet mask, I also created 4 integer mask variables to pair with the 4 IPv4 bytes: ma, mb, mc, and md. Each byte of the mask was calculated to find the number of left raised bits using the general formula: `mx = 256 – pow(2, 8 – std::min(prefix – x * 8,8))`. I first found how many bits the prefix raised of the IPv4byte (`prefix - x * 8`). With x representing the successive bytes. I also created a ceiling of 8 for the maximum number of bits that can be raised in one byte. In order to left shift the bits, I then found the total number of bits raised (`8 – std::min(prefix – x * 8,8))`. Using this, I found the decimal number corresponding to the number of bits using the pow() method. I finally, left shifted the byte by subtracting the decimal number from 256 resulting in a byte wise left shifted netmask.

The subnet address was found by doing a bitwise comparison for each IPv4 and mask byte `(a & ma, etc.)`. The first host was found by finding the address one larger than the subnet address in the final byte (`(d & md) + 1`). The broadcast address was found by finding the largest possible IPv4 address from the subnet address. This was done by inverting the last byte mask against a completely raised byte (`255 ^ md`). The last host was found by finding the address one smaller than the broadcast address (`(255 ^ md) - 1`).

The program prompts for an IPv4 address and a prefix length from which it will produce a fully calculated IPv4 subnet. Below is a demonstration of finding the IPv4 subnet of 192.168.152.30/26 as well as my IP address from the problem set of 137.146.135.252/24.

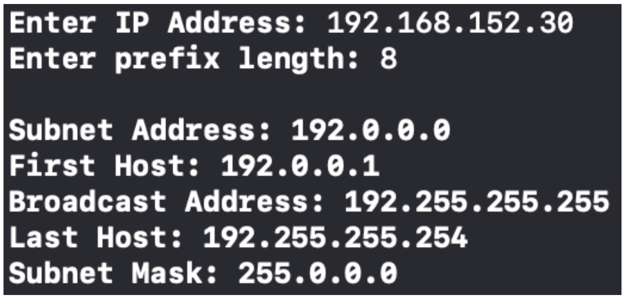


```
Enter IP Address: 192.168.152.30
Enter prefix length: 26

Subnet Address: 192.168.152.0
First Host: 192.168.152.1
Broadcast Address: 192.168.152.63
Last Host: 192.168.152.62
Subnet Mask: 255.255.255.192
```

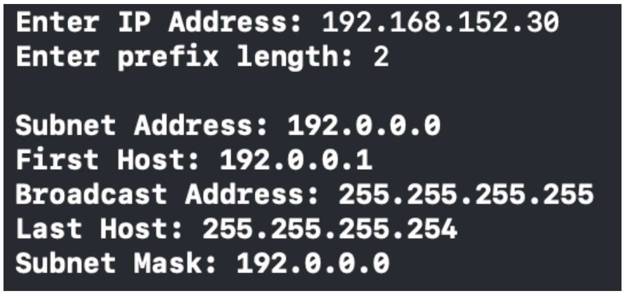


```
Enter IP Address: 137.146.135.252
Enter prefix length: 24

Subnet Address: 137.146.135.0
First Host: 137.146.135.1
Broadcast Address: 137.146.135.255
Last Host: 137.146.135.254
Subnet Mask: 255.255.255.0
```

The screenshot below also demonstrates the compatibility to find the subnet when the prefix results in multiple bytes existing within the subnet.

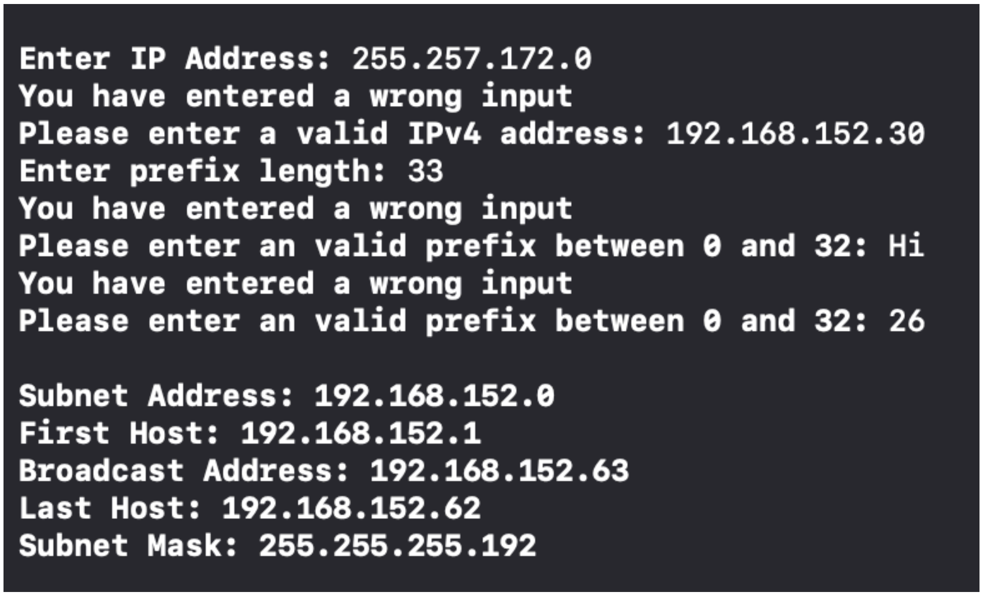


```
Enter IP Address: 192.168.152.30
Enter prefix length: 8

Subnet Address: 192.0.0.0
First Host: 192.0.0.1
Broadcast Address: 192.255.255.255
Last Host: 192.255.255.254
Subnet Mask: 255.0.0.0
```



```
Enter IP Address: 192.168.152.30
Enter prefix length: 2

Subnet Address: 192.0.0.0
First Host: 192.0.0.1
Broadcast Address: 255.255.255.255
Last Host: 255.255.255.254
Subnet Mask: 192.0.0.0
```

The command `std::cin >> prefix` at the end of the body of code was used to allow for readable printing into the terminal by preventing the exit message from appearing. Error checking for valid IPv4 and prefixes was derived from hackerearth. Below is a demonstration of an invalid IP address as well as an invalid prefix, eventually producing the IPv4 subnet of 192.168.152.30/26.

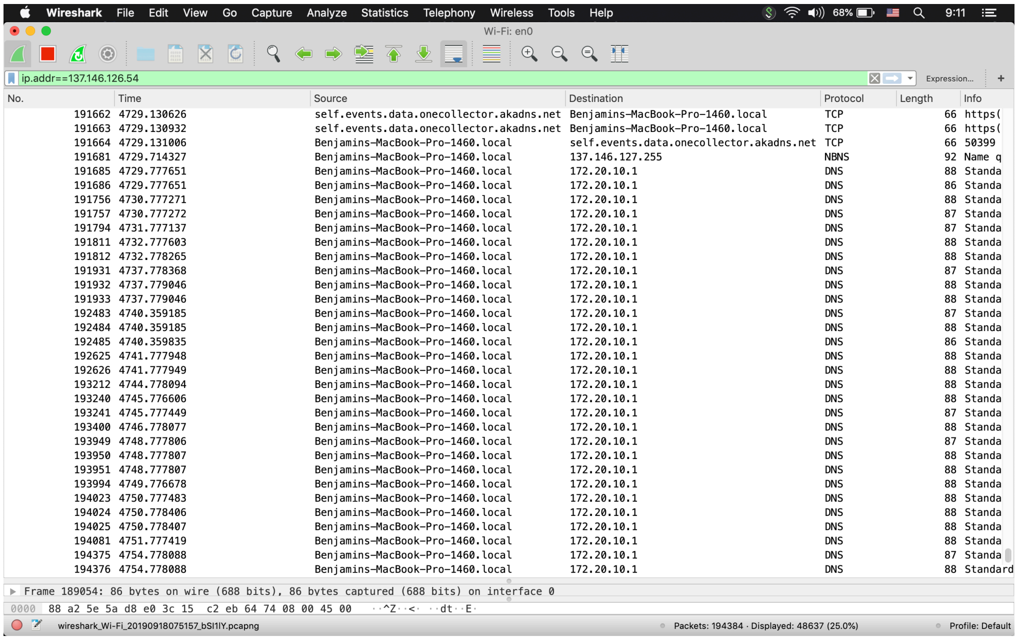


```
Enter IP Address: 255.257.172.0
You have entered a wrong input
Please enter a valid IPv4 address: 192.168.152.30
Enter prefix length: 33
You have entered a wrong input
Please enter an valid prefix between 0 and 32: Hi
You have entered a wrong input
Please enter an valid prefix between 0 and 32: 26

Subnet Address: 192.168.152.0
First Host: 192.168.152.1
Broadcast Address: 192.168.152.63
Last Host: 192.168.152.62
Subnet Mask: 255.255.255.192
```

Extensions

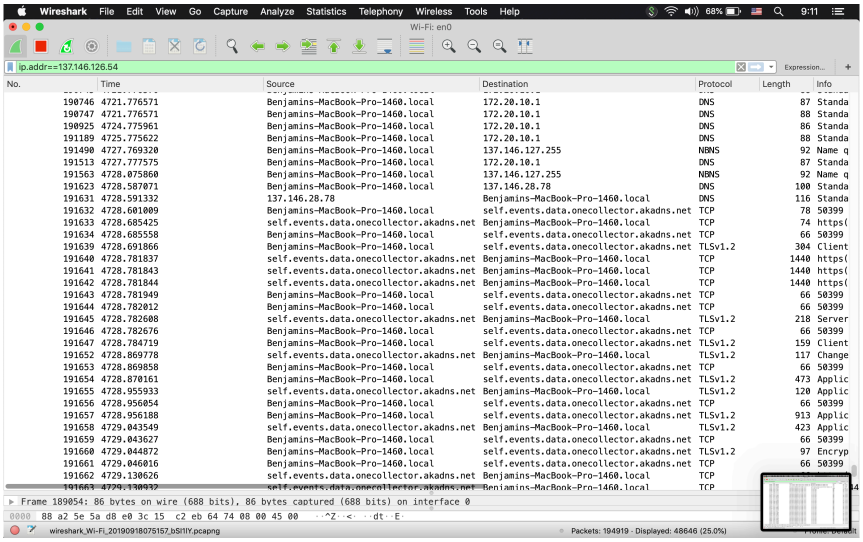
I decided to look at filters on wireshark as a part of my extensions.

I first decided to just view the traffic that is specific to my computer. To do this I typed ip.addr=137.146.126.54 into the display filter as I was receiving packets. I first closed all of the applications on my computer because I wanted to limit the amount and scope of broadcasts. The result can be seen in the first screenshot.


Wireshark File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
68% 9:11
Wi-Fi: en0
Expression...

```
ip.addr==137.146.126.54

No.     Time         Source                                Destination                           Protocol  Length  Info
191662  4729.130626  self.events.data.onecollector.akadns.net  Benjamins-MacBook-Pro-1460.local  TCP       66 https(
191663  4729.130932  self.events.data.onecollector.akadns.net  Benjamins-MacBook-Pro-1460.local  TCP       66 https(
191664  4729.131006  Benjamins-MacBook-Pro-1460.local  self.events.data.onecollector.akadns.net  TCP       66 50399
191681  4729.714327  Benjamins-MacBook-Pro-1460.local  137.146.127.255   NBNS      92 Name q
191685  4729.777651  Benjamins-MacBook-Pro-1460.local  172.20.10.1       DNS       88 Standa
191686  4729.777651  Benjamins-MacBook-Pro-1460.local  172.20.10.1       DNS       86 Standa
191756  4730.777271  Benjamins-MacBook-Pro-1460.local  172.20.10.1       DNS       88 Standa
191757  4730.777272  Benjamins-MacBook-Pro-1460.local  172.20.10.1       DNS       87 Standa
191794  4731.777137  Benjamins-MacBook-Pro-1460.local  172.20.10.1       DNS       87 Standa
191811  4732.777603  Benjamins-MacBook-Pro-1460.local  172.20.10.1       DNS       88 Standa
191812  4732.778265  Benjamins-MacBook-Pro-1460.local  172.20.10.1       DNS       88 Standa
191931  4737.778368  Benjamins-MacBook-Pro-1460.local  172.20.10.1       DNS       87 Standa
191932  4737.779046  Benjamins-MacBook-Pro-1460.local  172.20.10.1       DNS       88 Standa
191933  4737.779046  Benjamins-MacBook-Pro-1460.local  172.20.10.1       DNS       88 Standa
192483  4740.359185  Benjamins-MacBook-Pro-1460.local  172.20.10.1       DNS       87 Standa
192484  4740.359185  Benjamins-MacBook-Pro-1460.local  172.20.10.1       DNS       88 Standa
192485  4740.359835  Benjamins-MacBook-Pro-1460.local  172.20.10.1       DNS       86 Standa
192625  4741.777948  Benjamins-MacBook-Pro-1460.local  172.20.10.1       DNS       88 Standa
192626  4741.777949  Benjamins-MacBook-Pro-1460.local  172.20.10.1       DNS       88 Standa
193212  4744.778094  Benjamins-MacBook-Pro-1460.local  172.20.10.1       DNS       88 Standa
193240  4745.776606  Benjamins-MacBook-Pro-1460.local  172.20.10.1       DNS       88 Standa
193241  4745.777449  Benjamins-MacBook-Pro-1460.local  172.20.10.1       DNS       87 Standa
193400  4746.778077  Benjamins-MacBook-Pro-1460.local  172.20.10.1       DNS       88 Standa
193949  4748.777806  Benjamins-MacBook-Pro-1460.local  172.20.10.1       DNS       87 Standa
193950  4748.777807  Benjamins-MacBook-Pro-1460.local  172.20.10.1       DNS       88 Standa
193951  4748.777807  Benjamins-MacBook-Pro-1460.local  172.20.10.1       DNS       88 Standa
193994  4749.776678  Benjamins-MacBook-Pro-1460.local  172.20.10.1       DNS       88 Standa
194023  4750.777483  Benjamins-MacBook-Pro-1460.local  172.20.10.1       DNS       88 Standa
194024  4750.778406  Benjamins-MacBook-Pro-1460.local  172.20.10.1       DNS       88 Standa
194025  4750.778407  Benjamins-MacBook-Pro-1460.local  172.20.10.1       DNS       88 Standa
194081  4751.777419  Benjamins-MacBook-Pro-1460.local  172.20.10.1       DNS       88 Standa
194375  4754.778088  Benjamins-MacBook-Pro-1460.local  172.20.10.1       DNS       87 Standa
194376  4754.778088  Benjamins-MacBook-Pro-1460.local  172.20.10.1       DNS       88 Standard

▶ Frame 189054: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface 0
0000  88 a2 5e 5a d8 e0 3c 15  c2 eb 64 74 08 00 45 00   ··^Z··<·  ··dt··E·
wireshark_Wi-Fi_20190918075157_bSI1lY.pcapng    Packets: 194384 · Displayed: 48637 (25.0%)    Profile: Default
```

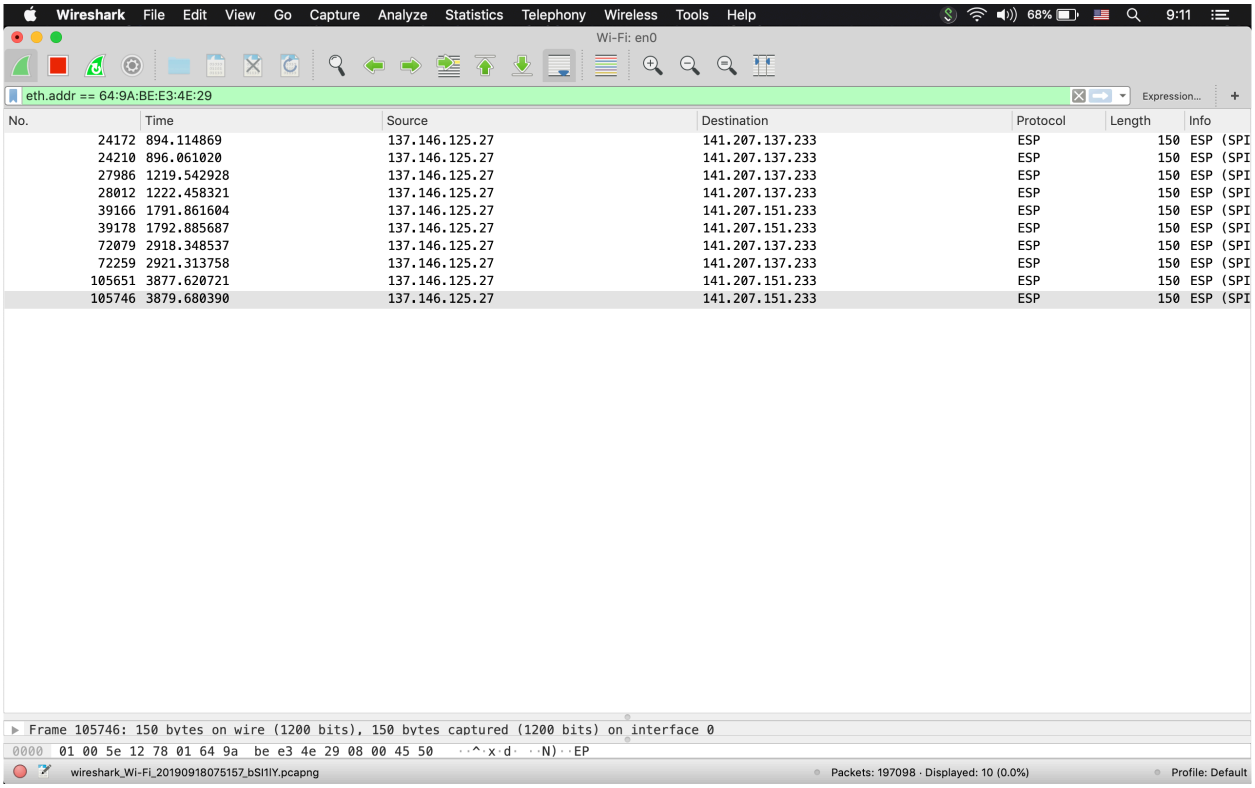
I noticed that my computer still continually pings the ip address 172.20.10.1. I tried to look it up but there is no information about it. The filter was specifically created to find traces where my computer was the source and the destination, but 172.20.10.1 never sent any responses. My computer however send two pings back to back and then would rest for 2-3 seconds.


Wireshark File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
68% 9:11
Wi-Fi: en0
Expression...

```
ip.addr==137.146.126.54

No.     Time         Source                                    Destination                               Protocol  Length  Info
190746  4721.776571  Benjamins-MacBook-Pro-1460.local          172.20.10.1                               DNS         87 Standa
190747  4721.776571  Benjamins-MacBook-Pro-1460.local          172.20.10.1                               DNS         88 Standa
190925  4724.775961  Benjamins-MacBook-Pro-1460.local          172.20.10.1                               DNS         86 Standa
191189  4725.775622  Benjamins-MacBook-Pro-1460.local          172.20.10.1                               DNS         88 Standa
191490  4727.769320  Benjamins-MacBook-Pro-1460.local          137.146.127.255                           NBNS        92 Name q
191513  4727.777575  Benjamins-MacBook-Pro-1460.local          172.20.10.1                               DNS         87 Standa
191563  4728.075860  Benjamins-MacBook-Pro-1460.local          137.146.127.255                           NBNS        92 Name q
191623  4728.587071  Benjamins-MacBook-Pro-1460.local          137.146.28.78                             DNS        100 Standa
191631  4728.591332  137.146.28.78                             Benjamins-MacBook-Pro-1460.local          DNS        116 Standa
191632  4728.601009  Benjamins-MacBook-Pro-1460.local          self.events.data.onecollector.akadns.net  TCP         78 50399
191633  4728.685425  self.events.data.onecollector.akadns.net  Benjamins-MacBook-Pro-1460.local          TCP         74 https(
191634  4728.685558  Benjamins-MacBook-Pro-1460.local          self.events.data.onecollector.akadns.net  TCP         66 50399
191639  4728.691866  Benjamins-MacBook-Pro-1460.local          self.events.data.onecollector.akadns.net  TLSv1.2    304 Client
191640  4728.781837  self.events.data.onecollector.akadns.net  Benjamins-MacBook-Pro-1460.local          TCP       1440 https(
191641  4728.781843  self.events.data.onecollector.akadns.net  Benjamins-MacBook-Pro-1460.local          TCP       1440 https(
191642  4728.781844  self.events.data.onecollector.akadns.net  Benjamins-MacBook-Pro-1460.local          TCP       1440 https(
191643  4728.781949  Benjamins-MacBook-Pro-1460.local          self.events.data.onecollector.akadns.net  TCP         66 50399
191644  4728.782012  Benjamins-MacBook-Pro-1460.local          self.events.data.onecollector.akadns.net  TCP         66 50399
191645  4728.782608  self.events.data.onecollector.akadns.net  Benjamins-MacBook-Pro-1460.local          TLSv1.2    218 Server
191646  4728.782676  Benjamins-MacBook-Pro-1460.local          self.events.data.onecollector.akadns.net  TCP         66 50399
191647  4728.784719  Benjamins-MacBook-Pro-1460.local          self.events.data.onecollector.akadns.net  TLSv1.2    159 Client
191652  4728.869778  self.events.data.onecollector.akadns.net  Benjamins-MacBook-Pro-1460.local          TLSv1.2    117 Change
191653  4728.869858  Benjamins-MacBook-Pro-1460.local          self.events.data.onecollector.akadns.net  TCP         66 50399
191654  4728.870161  Benjamins-MacBook-Pro-1460.local          self.events.data.onecollector.akadns.net  TLSv1.2    473 Applic
191655  4728.955933  self.events.data.onecollector.akadns.net  Benjamins-MacBook-Pro-1460.local          TLSv1.2    120 Applic
191656  4728.956054  Benjamins-MacBook-Pro-1460.local          self.events.data.onecollector.akadns.net  TCP         66 50399
191657  4728.956188  Benjamins-MacBook-Pro-1460.local          self.events.data.onecollector.akadns.net  TLSv1.2    913 Applic
191658  4729.043549  self.events.data.onecollector.akadns.net  Benjamins-MacBook-Pro-1460.local          TLSv1.2    423 Applic
191659  4729.043627  Benjamins-MacBook-Pro-1460.local          self.events.data.onecollector.akadns.net  TCP         66 50399
191660  4729.044872  Benjamins-MacBook-Pro-1460.local          self.events.data.onecollector.akadns.net  TLSv1.2     97 Encryp
191661  4729.046016  Benjamins-MacBook-Pro-1460.local          self.events.data.onecollector.akadns.net  TCP         66 50399
191662  4729.130626  self.events.data.onecollector.akadns.net  Benjamins-MacBook-Pro-1460.local          TCP
191663  4729.130932  self.events.data.onecollector.akadns.net  Benjamins-MacBook-Pro-1460.local          TCP

▶ Frame 189054: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface 0
0000  88 a2 5e 5a d8 e0 3c 15  c2 eb 64 74 08 00 45 00   ··^Z··<·  ··dt··E·
wireshark_Wi-Fi_20190918075157_bSI1lY.pcapng    Packets: 194919 · Displayed: 48646 (25.0%)
```

I then opened google chrome and googled colby college. As you can see, there is a significant spike in TCP requests. I did turn on hostname resolution so the IP of self.events.data.onecollector.akandns.net is not present. Here, you can see that there is a lot more activity with the intercepted packets being sent much more frequently.

I also wanted to see if I could track my phone. To do this I used the filter eth.addr== which filters for when the MAC address is either the source or the destination. I am not sure if this is because of the Colby network or something else, but I left the filter on for over an hour and this was the only traffic that was captured despite very actively using my phone to create internet traffic.


Wireshark File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
68%
9:11
Wi-Fi: en0
Expression...

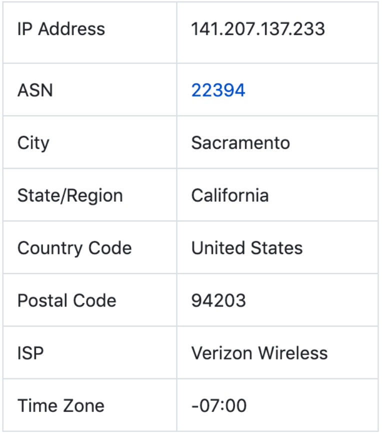
eth.addr == 64:9A:BE:E3:4E:29

```
No.      Time          Source           Destination       Protocol  Length  Info
   24172 894.114869    137.146.125.27   141.207.137.233   ESP          150  ESP (SPI
   24210 896.061020    137.146.125.27   141.207.137.233   ESP          150  ESP (SPI
   27986 1219.542928   137.146.125.27   141.207.137.233   ESP          150  ESP (SPI
   28012 1222.458321   137.146.125.27   141.207.137.233   ESP          150  ESP (SPI
   39166 1791.861604   137.146.125.27   141.207.151.233   ESP          150  ESP (SPI
   39178 1792.885687   137.146.125.27   141.207.151.233   ESP          150  ESP (SPI
   72079 2918.348537   137.146.125.27   141.207.137.233   ESP          150  ESP (SPI
   72259 2921.313758   137.146.125.27   141.207.137.233   ESP          150  ESP (SPI
  105651 3877.620721   137.146.125.27   141.207.151.233   ESP          150  ESP (SPI
  105746 3879.680390   137.146.125.27   141.207.151.233   ESP          150  ESP (SPI
```

```
▶ Frame 105746: 150 bytes on wire (1200 bits), 150 bytes captured (1200 bits) on interface 0
0000  01 00 5e 12 78 01 64 9a  be e3 4e 29 08 00 45 50   ··^·x·d· ··N)··EP
```

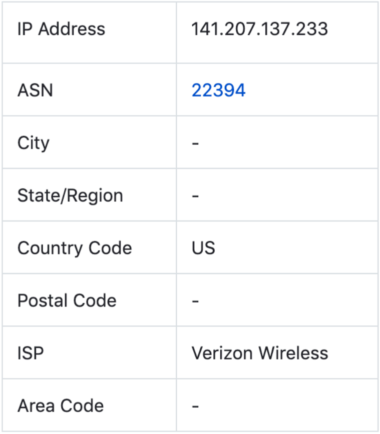
wireshark_Wi-Fi_20190918075157_bSI1lY.pcapng   Packets: 197098 · Displayed: 10 (0.0%)   Profile: Default

I looked up this IP and found that it was Verizon wireless which makes sense because they are my carrier. I have no clue why this was the only traffic that was filtered, but I find it interesting that the only packets were for my mobile service provider.



IP Address	141.207.137.233
ASN	22394
City	Sacramento
State/Region	California
Country Code	United States
Postal Code	94203
ISP	Verizon Wireless
Time Zone	-07:00

IP2Location.com Results



IP Address	141.207.137.233
ASN	22394
City	-
State/Region	-
Country Code	US
Postal Code	-
ISP	Verizon Wireless
Area Code	-

EurekAPI.com Results