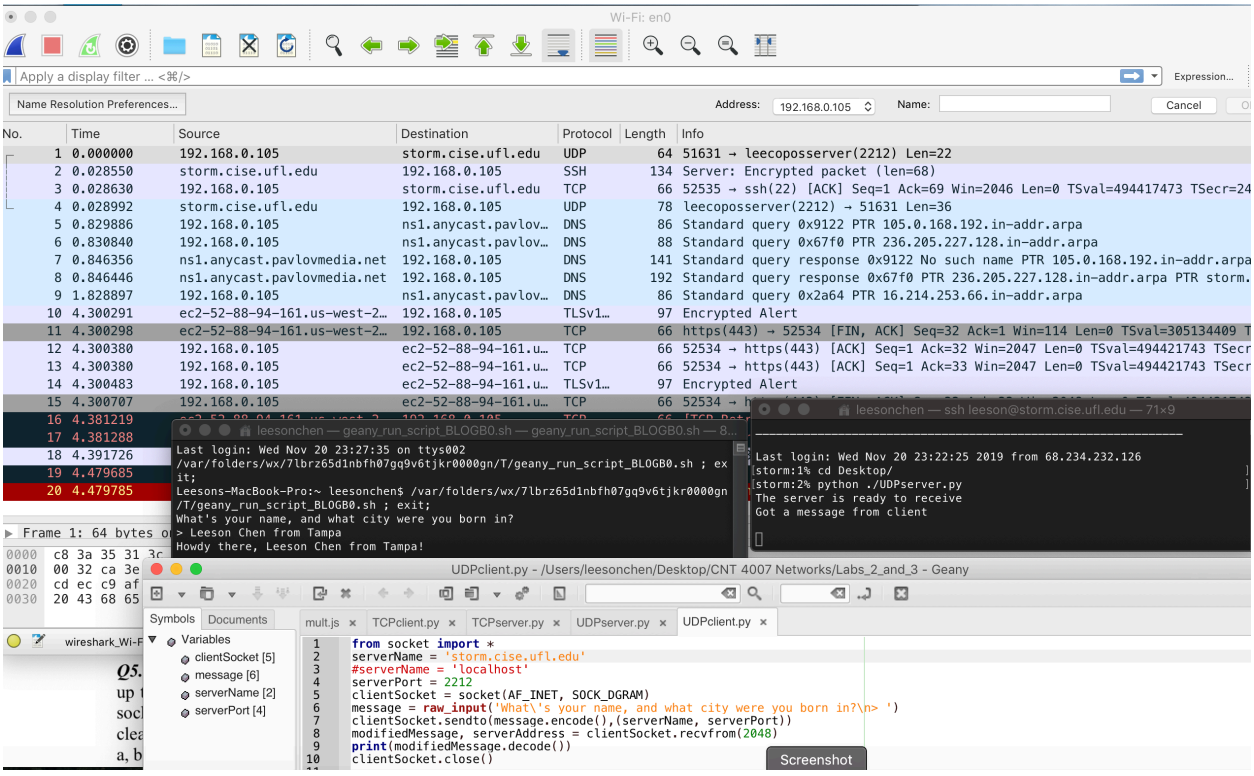
Leeson Chen
Labs 2 & 3 for CNT4007C Computer Network Fundamentals, Fall 2019
On UPD and TCP socket programming, MAC and IP addresses, wireshark Due Date: Nov 21th, 2019 on Canvas before midday 11:59am
Please also provide an exact hard copy (in class or during the TA office hours)
Perform the following experiments with a clear brief description of each experiment and its outcome:

Q1. *Write a program in python to open a UDP socket between a client and a server, then use wireshark to monitor the packets sent. [Hint: Write a small program for the UDP client side and another for the server side. Use parts of the code in your book pages 159- 164]. Include in the message your name, and the city of your birth place. Use port number that is the last 4 digits of your UFID. Explain the address and port numbers used.*

Screenshot:

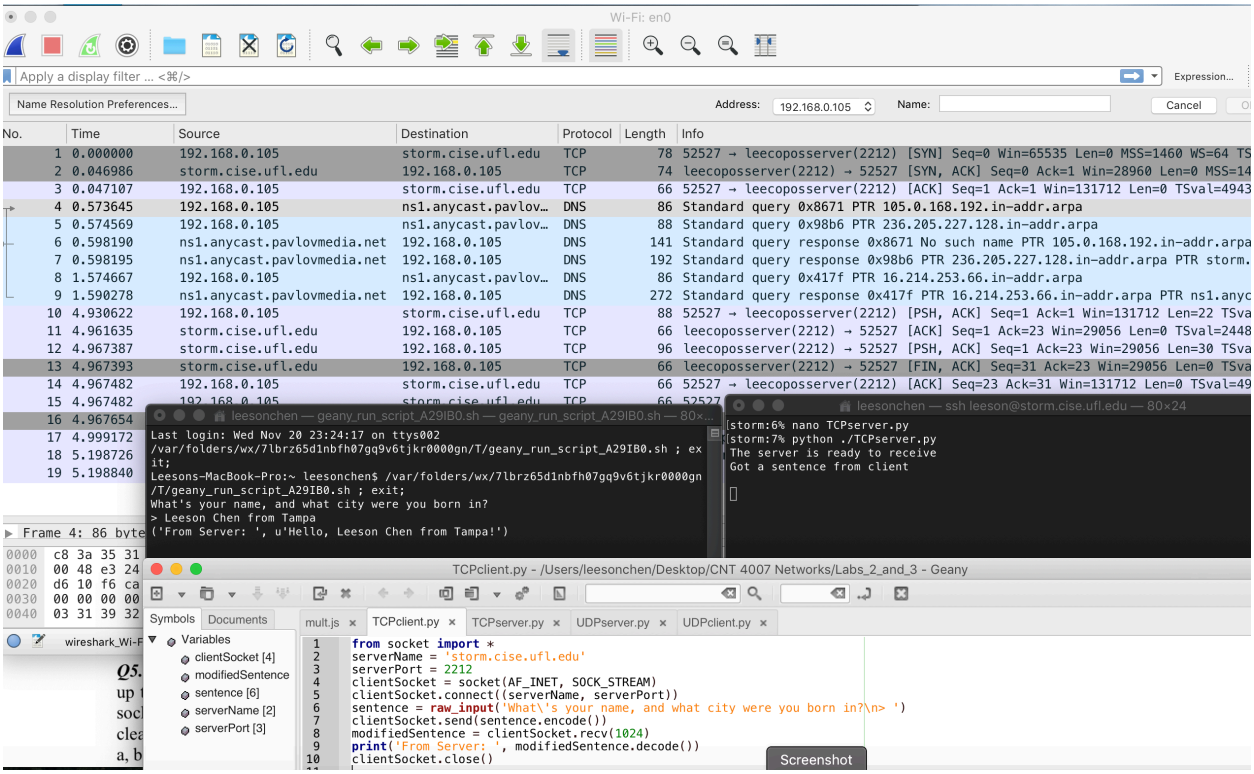


Explanation:
I ran the client-side code off my laptop (IP address 192.168.0.105) and ran the server-side code from my account on UF's storm.cise.ufl.edu server. Originally I was running both on my laptop by localhost but Wireshark wouldn't detect those localhost packets (since it wasn't sending over the network). The client side sends my name and city (the message "Leeson Chen from Tampa") and the server responds by saying howdy with my name and city ("Howdy there, Leeson Chen from Tampa!"). The server will also record it received a message by printing on it's side "Got a

message from client". The port number used is 2212, after the last four digits of my UFID. On the Wireshark display, we can see encrypted traffic going from my laptop to the storm server. Pavlov is my apartment complex's internet service provider.

Q2. *Write a program in python to open a TCP socket between a client and a server, then user wireshark to monitor the packets sent. [Hint: Write a small program for the TCP client side and another for the TCP server side. Use parts of the code in your book pages 164-170]. Include in the message your name, and the city of your birth place. Use port number that is the last 4 digits of your UFID. Explain the address and port numbers used.*

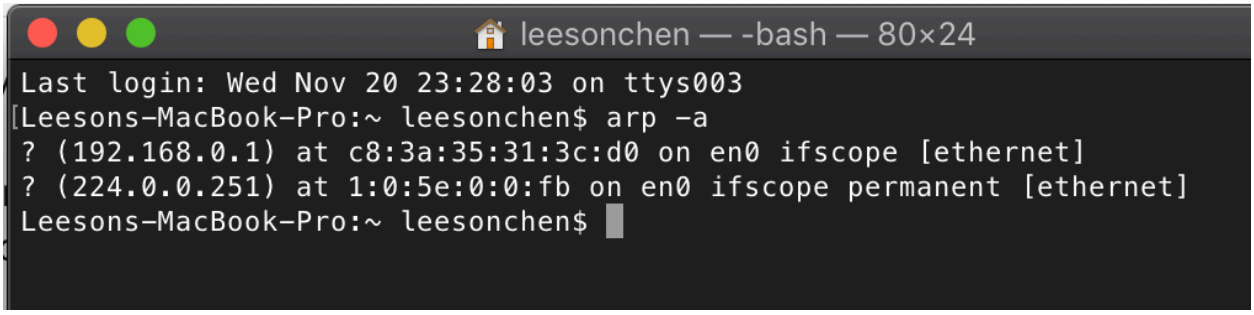
Screenshot:



Explanation:
This is pretty much the same code as before. The TCP client is running on my laptop which has the IP address of 192.168.0.105. The TCP server is running from my window ssh-ed into the storm.cise.ufl.edu server so that Wireshark can detect traffic. My client sends the message "Leeson Chen from Tampa" and the server returns "Hello, Leeson Chen from Tampa!" while also recording on its side "Got a sentence from client". Port number is again 2212. Wireshark shows the TCP traffic going between my laptop and the storm server.

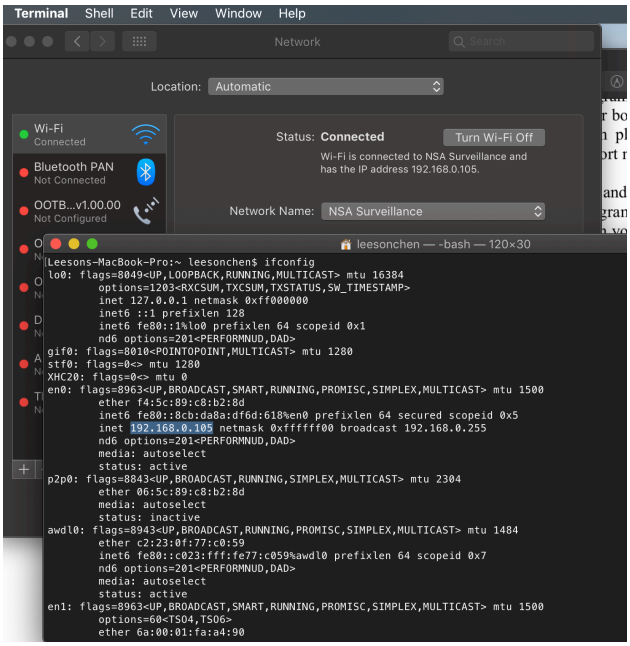
Q3. *Show the ARP table of your machine and describe it.*

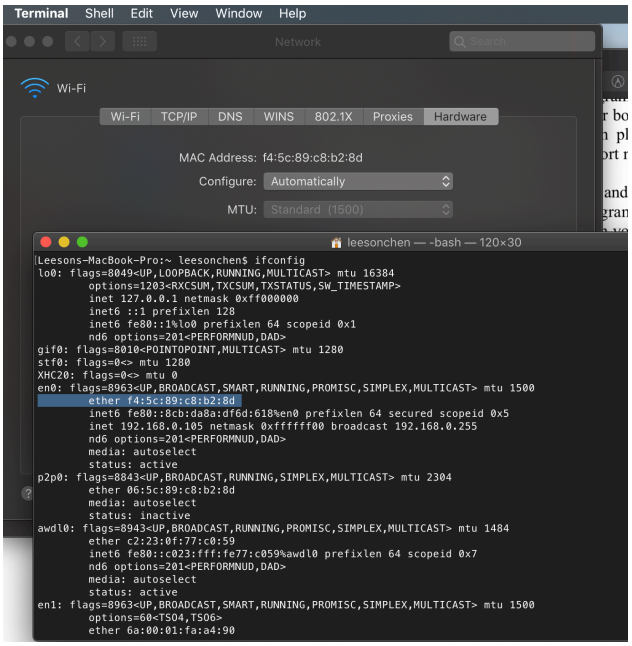
Screenshot:



Explanation:
I just typed “arp -a” on my command line to get the ARP table to show. ARP (address resolution protocol) associates an IP with a MAC address, so the IP 192.168.0.1 resolves into the MAC address of c8:3a:35:31:3c:d0. The same goes for the second line.

Q4. *Show the mac addresses for the interfaces of your machine and the ip addresses using ifconfig (or ipconfig), and the routing tables using netstat (or other) commands.*





Screenshot:
I used the command “ifconfig” to display this table, where my MAC and IP addresses are shown in the highlighted sections. I confirmed this by going into my laptop’s settings and getting the IP

and MAC addresses from there, which match what's displayed. Like the analogy in class, an IP address is like your machine's mailing address while the MAC is more similar to the machine's social security number.

Routing table screenshot:

```
Last login: Wed Nov 20 23:29:24 on ttys004
[Leesons-MacBook-Pro:~ leesonchen$ netstat
Active Internet connections
Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)
tcp4       0      0  192.168.0.105.52637    199.232.34.167.https   ESTABLISHED
tcp4       0      0  192.168.0.105.52636    ec2-34-208-17-53.https ESTABLISHED
tcp4       0      0  localhost.12443        localhost.52635        ESTABLISHED
tcp4       0      0  localhost.52635        localhost.12443        ESTABLISHED
tcp4       0      0  localhost.12443        localhost.52634        ESTABLISHED
tcp4       0      0  localhost.52634        localhost.12443        ESTABLISHED
tcp4       0      0  192.168.0.105.52631    ec2-34-218-190-1.https ESTABLISHED
tcp4       0      0  localhost.12443        localhost.52630        ESTABLISHED
tcp4       0      0  localhost.52630        localhost.12443        ESTABLISHED
tcp4       0      0  192.168.0.105.52558    40.97.124.226.https    ESTABLISHED
tcp4       0      0  192.168.0.105.52467    nyc04-018.ff.ava.http  ESTABLISHED
tcp4       0      0  localhost.12080        localhost.52466        ESTABLISHED
tcp4       0      0  localhost.52466        localhost.12080        ESTABLISHED
tcp4       0      0  192.168.0.105.52339    52.96.28.178.https     ESTABLISHED
tcp4       0      0  192.168.0.105.52330    17.57.144.148.5223     ESTABLISHED
tcp4       0      0  192.168.0.105.52633    atl26s14-in-f4.1.https TIME_WAIT
udp4       0      0  *.60054                *.*
udp4       0      0  *.53342                *.*
udp4       0      0  *.65404                *.*
udp4       0      0  *.51623                *.*
udp4       0      0  *.*                    *.*
udp4       0      0  *.*                    *.*
udp4       0      0  *.*                    *.*
udp4       0      0  *.49242                *.*
udp6       0      0  *.60582                *.*
udp4       0      0  *.60582                *.*
udp6       0      0  *.54471                *.*
udp4       0      0  *.54471                *.*
udp6       0      0  *.57742                *.*
udp4       0      0  *.57742                *.*
udp4       0      0  *.*                    *.*
udp4       0      0  *.*                    *.*
udp4       0      0  *.64954                *.*
udp4       0      0  localhost.57377        *.*
udp4       0      0  *.*                    *.*
udp4       0      0  *.*                    *.*
udp4       0      0  *.50274                *.*
udp4       0      0  *.*                    *.*
udp4       0      0  *.*                    *.*
udp4       0      0  *.*                    *.*
udp4       0      0  *.*                    *.*
udp4       0      0  *.*                    *.*
udp4       0      0  *.*                    *.*
udp4       0      0  *.*                    *.*
udp4       0      0  *.*                    *.*
udp4       0      0  *.*                    *.*
udp4       0      0  *.*                    *.*
udp4       0      0  *.*                    *.*
udp4       0      0  *.*                    *.*
udp4       0      0  *.*                    *.*
udp4       0      0  *.*                    *.*
udp4       0      0  *.*                    *.*
udp4       0      0  *.*                    *.*
udp4       0      0  *.*                    *.*
udp4       0      0  *.*                    *.*
udp4       0      0  *.*                    *.*
udp4       0      0  *.*                    *.*
udp4       0      0  *.*                    *.*
udp4       0      0  *.*                    *.*
```

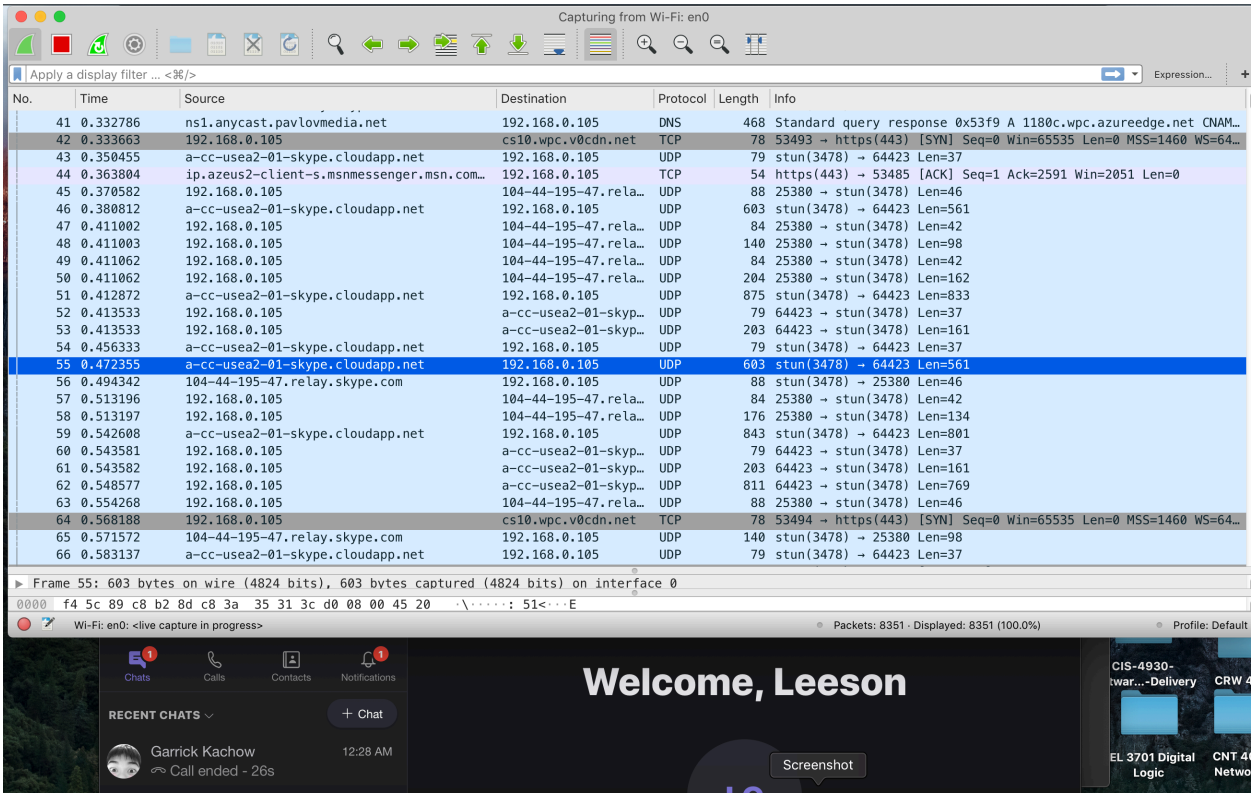
To display this routing table I just typed "netstat" into my command line. The table shows the routes where network traffic from my laptop is directed.

Q5. *Extra points (optional): suggest up to three related 'original' experiments (each with up to three short well-defined tasks) to use networking tools or commands to understand socket programming, UDP, TCP, MAC, IP addresses, or ARP/routing tables. Make them clear, brief and include short sample answers. Label them clearly as well (1. a, b, c. and 2. a, b, c, 3. ...). Provide references and/or websites as appropriate.*

Suggested Question 1:

During class, we mentioned that VoIP call services such as Skype might prefer to use UDP over TCP, because real-time conversation is more important than not losing packets. Open up a Skype call and monitor the traffic with Wireshark to determine whether Skype is UDP or TCP.

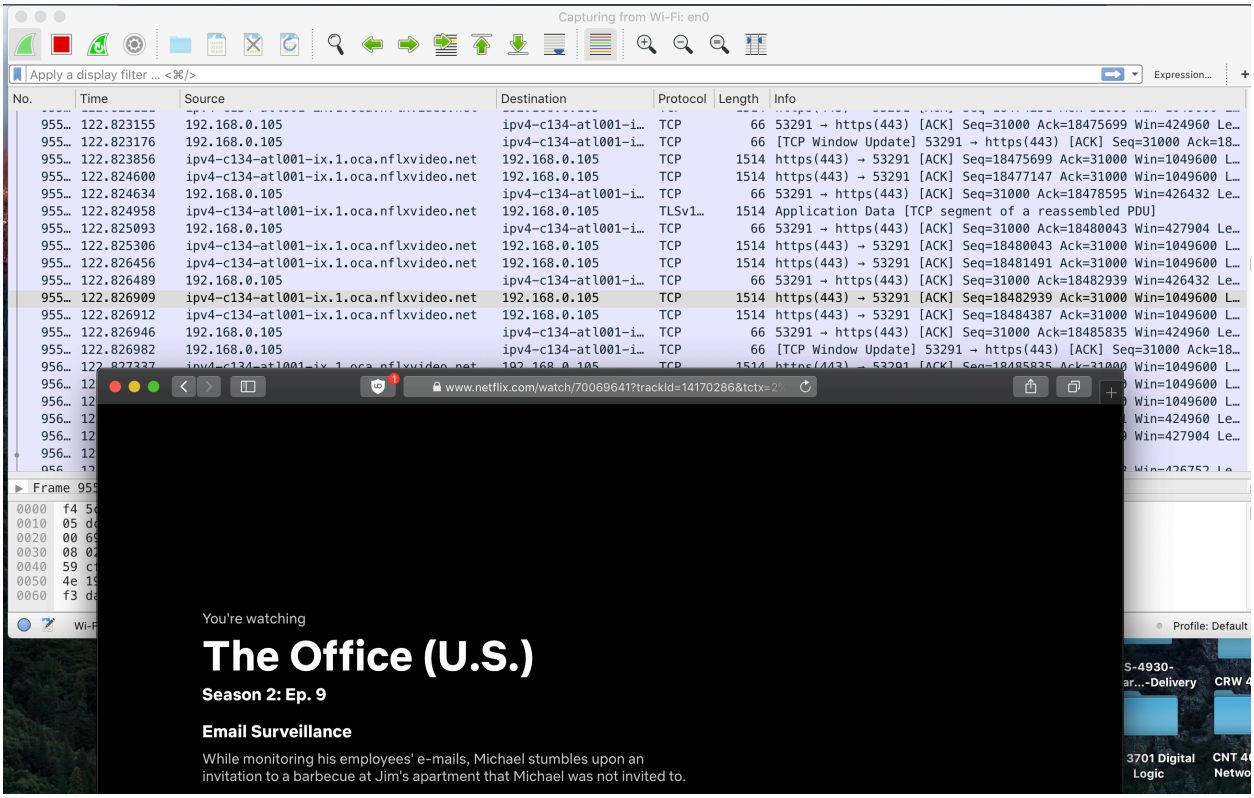
Screenshot of answer (Skype uses UDP):



Suggested Question 2:

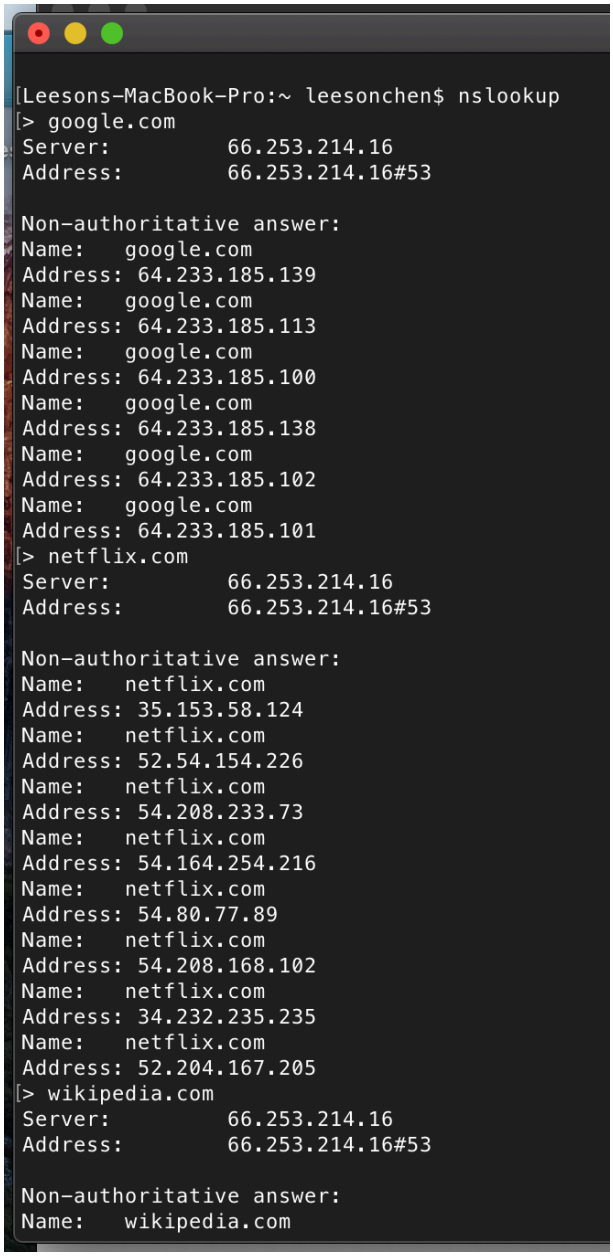
During class, we mentioned that video streaming services might prefer to use UDP over TCP due to the same reasons: packets can be dropped in favor of maintaining real-time display. What sort of protocol do you think Netflix uses? Use Wireshark to determine the answer.

Screenshot of Answer (Netflix uses TCP):



Suggested Question 3:
Query a DNS server for the IP resolutions of some common hostnames / websites. (Hint: use the **nslookup** command).

Screenshot of Answer:



Code

UDPclient.py:

```
from socket import *
serverName = 'storm.cise.ufl.edu'
#serverName = 'localhost'
serverPort = 2212
clientSocket = socket(AF_INET, SOCK_DGRAM)
message = raw_input('What\'s your name, and what city were you born in?\n> ')
clientSocket.sendto(message.encode(),(serverName, serverPort))
modifiedMessage, serverAddress = clientSocket.recvfrom(2048)
print(modifiedMessage.decode())
clientSocket.close()
```

UDPserver.py:

```
from socket import *
serverPort = 2212
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(('', serverPort))
print('The server is ready to receive')
while True:
    message, clientAddress = serverSocket.recvfrom(2048)
    #modifiedMessage = message.decode().upper()
    if (message): print ('Got a message from client\n')
    modifiedMessage = "Howdy there, " + message + "!"
    serverSocket.sendto(modifiedMessage.encode(), clientAddress)
```

TCPclient.py:

```
from socket import *
serverName = 'storm.cise.ufl.edu'
serverPort = 2212
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, serverPort))
sentence = raw_input('What\'s your name, and what city were you born in?\n> ')
clientSocket.send(sentence.encode())
modifiedSentence = clientSocket.recv(1024)
print('From Server: ', modifiedSentence.decode())
clientSocket.close()
```

TCPserver.py:

```
from socket import *
serverPort = 2212
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind(('', serverPort))
serverSocket.listen(1)
print('The server is ready to receive')
while True:
    connectionSocket, addr = serverSocket.accept()
    sentence = connectionSocket.recv(1024).decode()
    #capitalizedSentence = sentence.upper()
    #connectionSocket.send(capitalizedSentence.encode())
    if (sentence): print ('Got a sentence from client\n')
    returnsentence = "Hello, " + sentence + "!"
    connectionSocket.send(returnsentence.encode())
    connectionSocket.close()
```