# CSCI 5010 – Fundamentals of Data Communications

Lab 6
Wireshark

University of Colorado Boulder
Department of Computer Science
Network Engineering

Professor Levi Perigo, Ph.D.

## Objectives

- Learn the basic operations of Wireshark
- Learn how to capture and analyze ICMP traffic
- Demonstrate best practices for analyzer placement
- Differentiate Wireshark captures between switches and routers
- Display which NICs on analyzer are capturing traffic
- Display IPv4 and/or IPv6 addresses on NICs
- Learn how to perform continuous captures for HTTP requests
- Explain and display different ways to demonstrate top talkers on the network
- Learn how to create coloring rules in Wireshark
- Learn how to create graphs for visual representation
- Learn how to capture and analyze application specific traffic – DHCP, HTTP

## Summary

One's understanding of network protocols can often be greatly deepened by "seeing protocols in action" and by "playing around with protocols" – observing the sequence of messages exchanged between two protocol entities, delving down into the details of protocol operation, and causing protocols to perform certain actions and then observing these actions and their consequences. This can be done in simulated scenarios or in a "real" network environment such as the Internet. In the Wireshark lab you'll be doing in this course, you'll be running various network applications in different scenarios using your own computer or in a virtual machine environment. You'll observe the network protocols "in action," interacting and exchanging messages with protocol entities executing elsewhere in the Internet. Thus, you and your computer will be an integral part of these "live" labs. You'll observe, and you'll learn, by doing.

In this Wireshark lab, you'll get acquainted with Wireshark, and make some simple packet captures and observations.

The basic tool for observing the messages exchanged between executing protocol entities is called a packet sniffer. As the name suggests, a packet sniffer captures ("sniffs") messages being sent/received from/by your computer; it will also typically store and/or display the contents of the various protocol fields in these captured messages. A packet sniffer itself is passive. It observes messages being sent and received by applications and protocols running on your computer, but never sends packets itself. Similarly, received packets are never explicitly addressed to the packet sniffer. Instead, a packet sniffer receives a copy of packets that are sent/received from/by application and protocols executing on your machine. The packet capture library receives a copy of every link-layer frame that is sent from or received by your computer.

The second component of a packet sniffer is the packet analyzer, which displays the contents of all fields within a protocol message. In order to do so, the packet analyzer must "understand" the structure of all messages exchanged by protocols. For example, suppose we are interested in displaying the various fields in messages exchanged by the HTTP protocol. The packet analyzer understands the format of Ethernet frames, and so can identify the IP datagram within an Ethernet frame. It also understands the IP datagram format, so that it can extract the TCP segment within the IP datagram. Finally, it understands the TCP segment structure, so it can extract the HTTP message contained in the TCP segment. Finally, it understands the HTTP protocol and so, for example, knows that the first bytes of an HTTP message will contain the string "GET," "POST," or "HEAD."

We will be using the Wireshark packet sniffer [http://www.wireshark.org/] for this lab, allowing us to display the contents of messages being sent/received from/by protocols at different levels of the protocol stack. (Technically speaking, Wireshark is a packet analyzer that uses a packet capture library in your computer). Wireshark is a free network protocol analyzer that runs on Windows, Linux/Unix, and Mac computers.

# Part 1

## Objective 1.1 - Downloading Wireshark and Navigation Overview

1. Download and install Wireshark:
   - Go to http://www.wireshark.org/download.html, download and install Wireshark.
   - The Wireshark FAQ has many helpful hints and interesting tidbits of information, particularly if you have trouble installing or running Wireshark.
   - Helpful install video: https://www.youtube.com/watch?v=flDzURAm8wQ
2. Wireshark Navigation

   Helpful navigation video: https://www.youtube.com/watch?v=PYrCS21sPbA

## Objective 1.2 - Running Wireshark

When you run the Wireshark program, you'll get a startup screen, as shown below (note: this screenshot may be an older version of Wireshark):

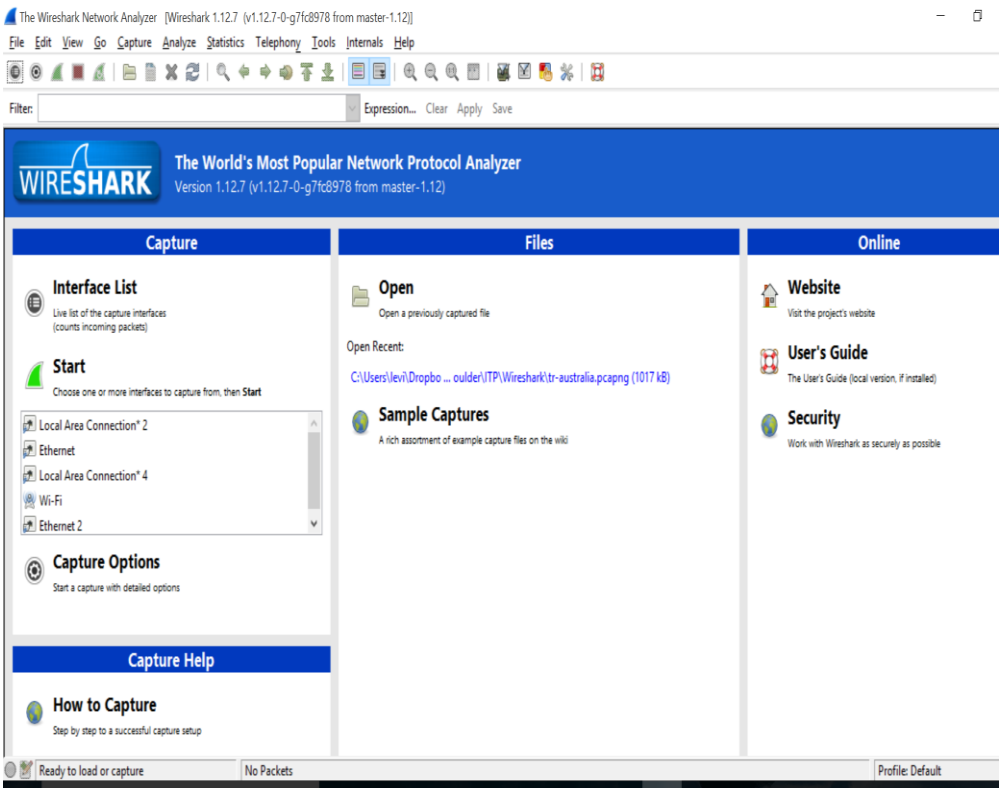



**Figure 1:** Initial Wireshark Screen

Take a look at the upper left-hand side of the screen – you'll see an "Interface list". This is the list of network interfaces on your computer. Once you choose an interface, Wireshark will capture all packets on that interface. In the example above, there is an Ethernet interface (Gigabit network connection) and a wireless interface ("Wi-Fi").

If you click on one of these interfaces to start packet capture (i.e., for Wireshark to begin capturing all packets being sent to/from that interface), a screen like the one below will be displayed, showing information about the packets being captured. Once you start packet capture, you can stop it by using the Capture pull down menu and selecting Stop.

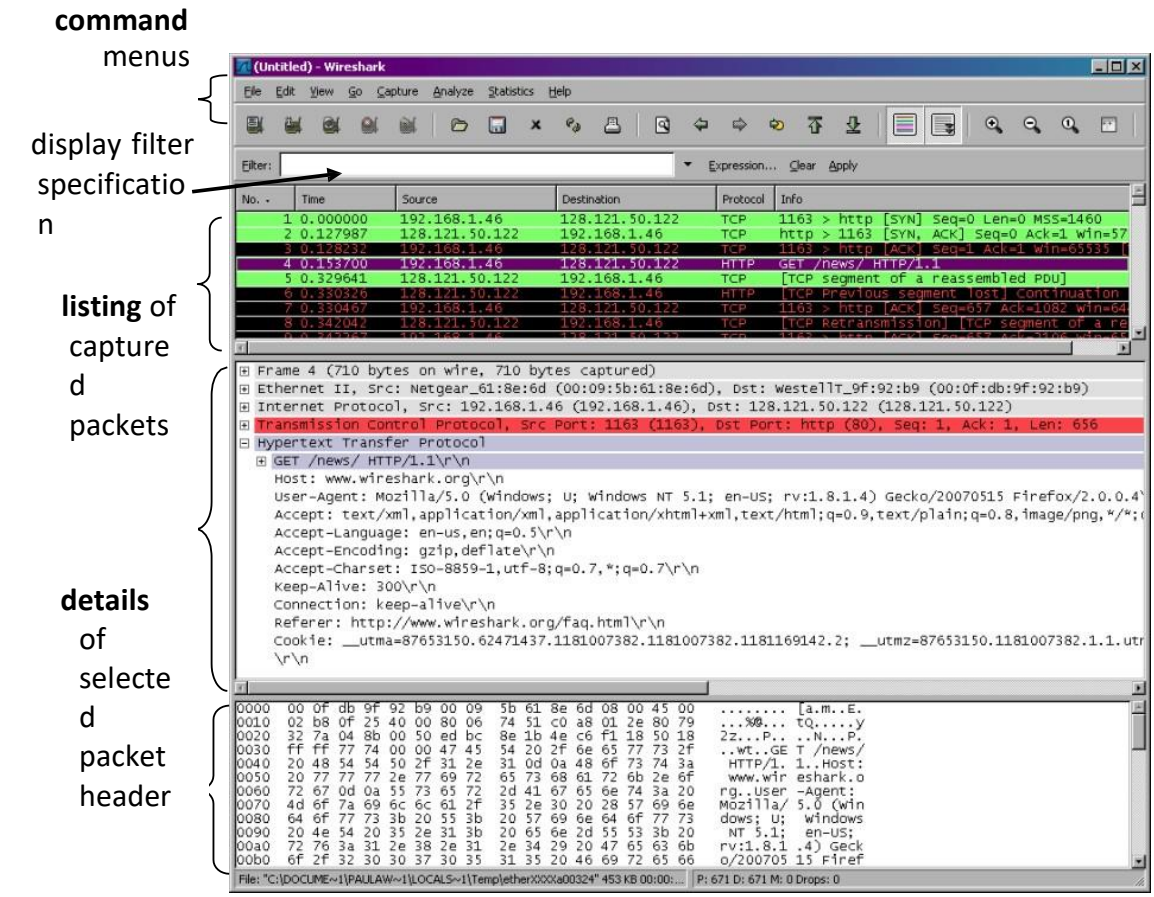



**Figure 2:** Wireshark Graphical User Interface, during packet capture and analysis

The Wireshark interface has five major components:

- The **command menus** are standard pulldown menus located at the top of the window. Of interest to us now are the File and Capture menus. The File menu

allows you to save captured packet data or open a file containing previously captured packet data and exit the Wireshark application. The Capture menu allows you to begin packet capture.

- The **packet-listing window** displays a one-line summary for each packet captured, including the packet number (assigned by Wireshark; this is *not* a packet number contained in any protocol's header), the time at which the packet was captured, the packet's source and destination addresses, the protocol type, and protocol-specific information contained in the packet. The packet listing can be sorted according to any of these categories by clicking on a column name. The protocol type field lists the highest-level protocol that sent or received this packet, i.e., the protocol that is the source or ultimate sink for this packet.
- The **packet-header details window** provides details about the packet selected (highlighted) in the packet-listing window. (To select a packet in the packet-listing window, place the cursor over the packet's one-line summary in the packet-listing window and click with the left mouse button.). These details include information about the Ethernet frame (assuming the packet was sent/received over an Ethernet interface) and IP datagram that contains this packet. The amount of Ethernet and IP-layer detail displayed can be expanded or minimized by clicking on the plus minus boxes to the left of the Ethernet frame or IP datagram line in the packet details window. If the packet has been carried over TCP or UDP, TCP or UDP details will also be displayed, which can similarly be expanded or minimized. Finally, details about the highest-level protocol that sent or received this packet are also provided.
- The **packet-contents window** displays the entire contents of the captured frame, in both ASCII and hexadecimal format.
- Towards the top of the Wireshark graphical user interface, is the **packet display filter field,** into which a protocol name or other information can be entered in order to filter the information displayed in the packet-listing window (and hence the packet-header and packet-contents windows). In the example below, we'll use the packet-display filter field to have Wireshark hide (not display) packets except those that correspond to HTTP messages.

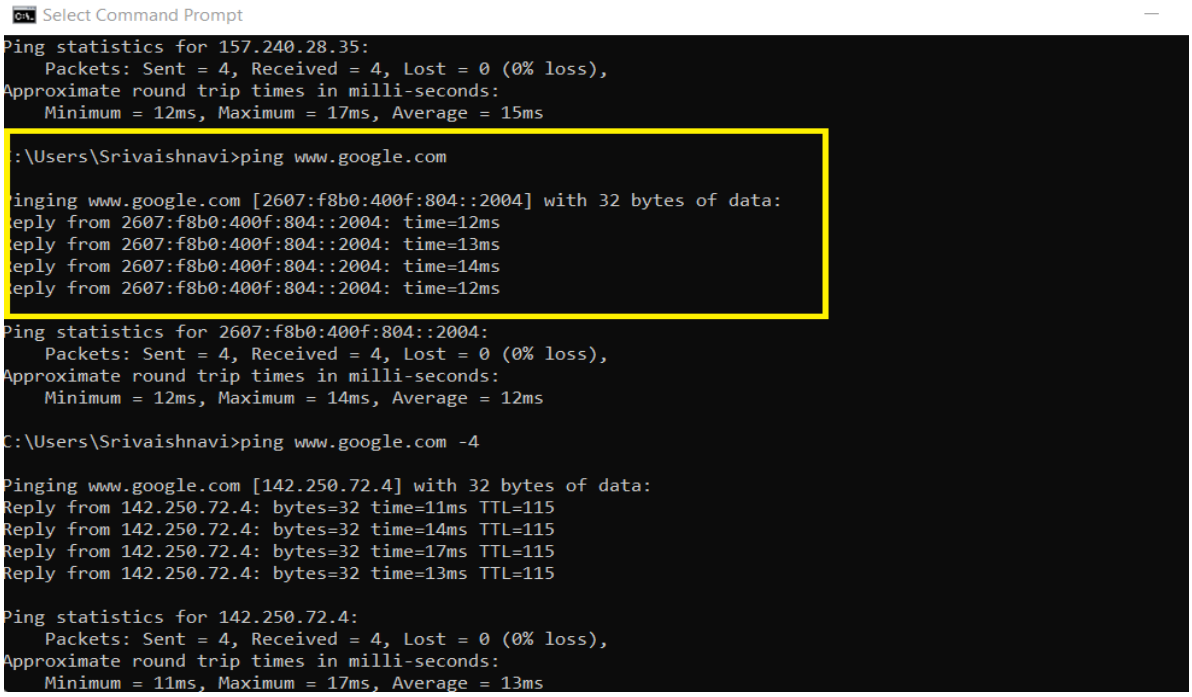
# Part 2

## Objective 2.1 – ICMP

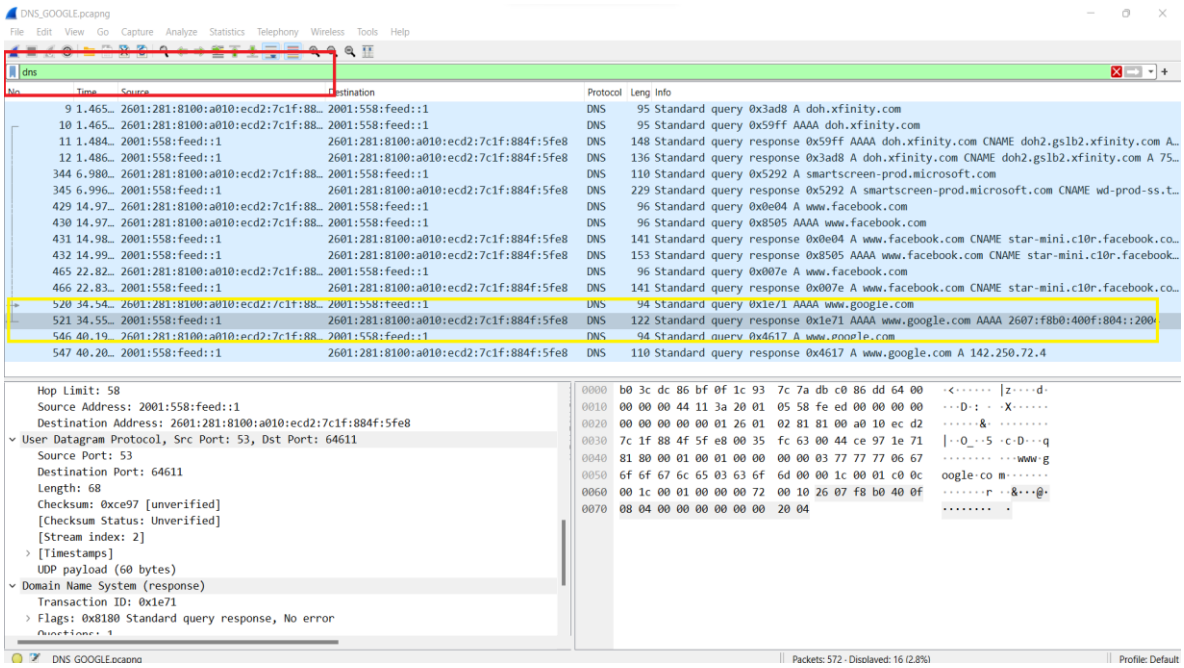
1. Open command prompt/terminal (depending on the operating system)
2. Start Wireshark and begin capture
3. Ping any "hostname" (where the "hostname" is a URL, example: ping www.google.com)
4. When the Ping finishes, stop the capture. [Press Ctrl + C in MAC to stop the ping]

5. Filter the capture to only display DNS traffic

    a. Provide a screenshot of the DNS reply from the server that shows the IP address of the URL. [**5 points**]




Capturing only "DNS" traffic and the www.google.com is resoluted to its IP address




**Screenshot_1**

b. Explain why DNS would be in this capture when you pinged? [**5 points**]

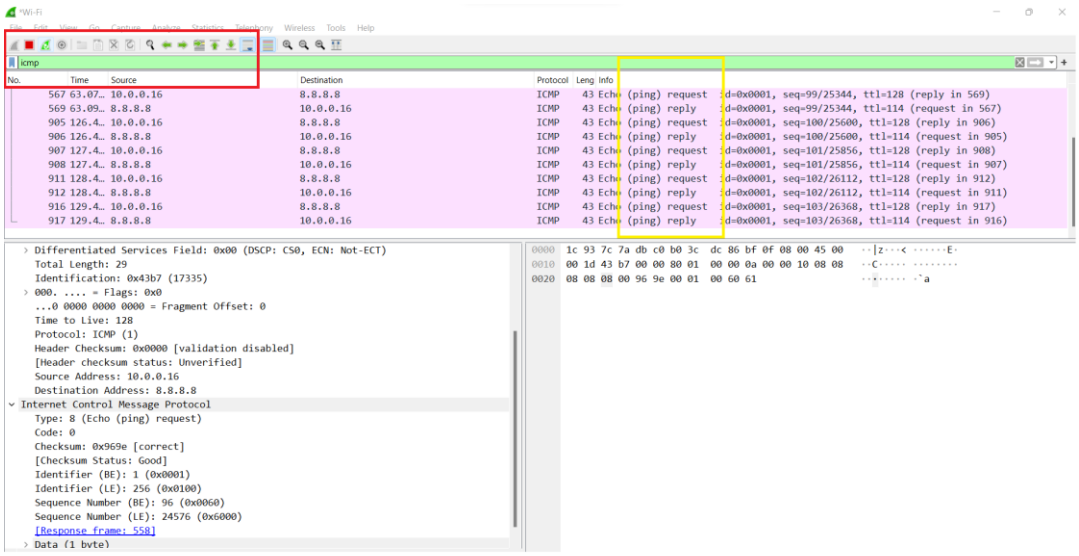
When www.google.com is pinged, the domain address resolution happens to convert the nomain to network understandable IP address. The DNS happens before the first ICMP Ping Request packet is sent. DNS will affect the first ping only because initially a DNS request will be required to resolve the domain name to an IP. The time it takes to resolve the domain will increase the first ping time, the second and subsequent pings will be unaffected

6. Filter the capture to only display the Ping traffic

   a. Were the Pings successful? b. Provide the filtered Wireshark screenshot, and explain how you know they were/were not successful? [**10 points**]

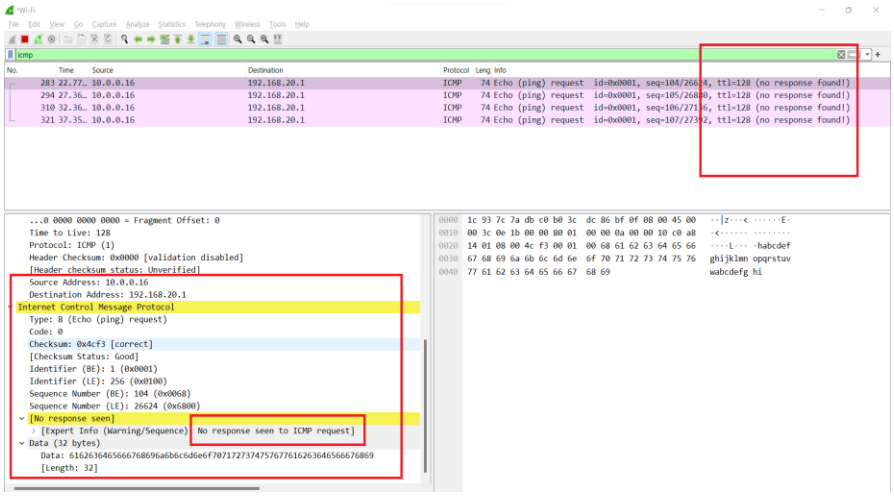
Pinged the 8.8.8.8 (The Google DNS server) passing "1"byte payload in ping request (ping screenshot is attached below) and captured the ping (aka ICMP cho traffic) by means of filtering using the "ICMP" in filter section of the wireshark application. **"The ping is successful".**

One way to check the ping request status is for every echo request, the response reply is successful with a derived TTL time. There is no "No Response found" " Request Timed out" error/warning sequence to the request in the "info" section. And the echo ping reply code is "0", Checksum status = "Good" hence the I know the ping is successful. Attached the *screenshot (3) for an unsuccessful ping request*.




**Screenshot-2**

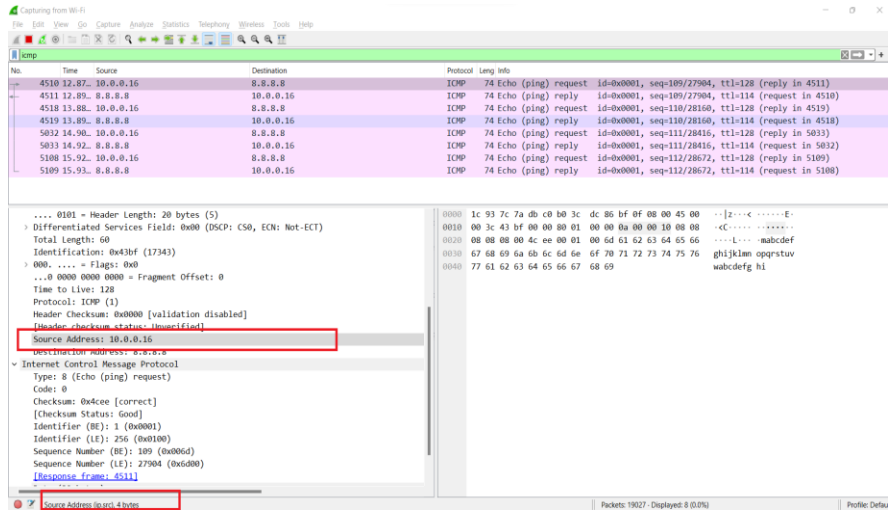
**the reference to unsuccessful ping:**

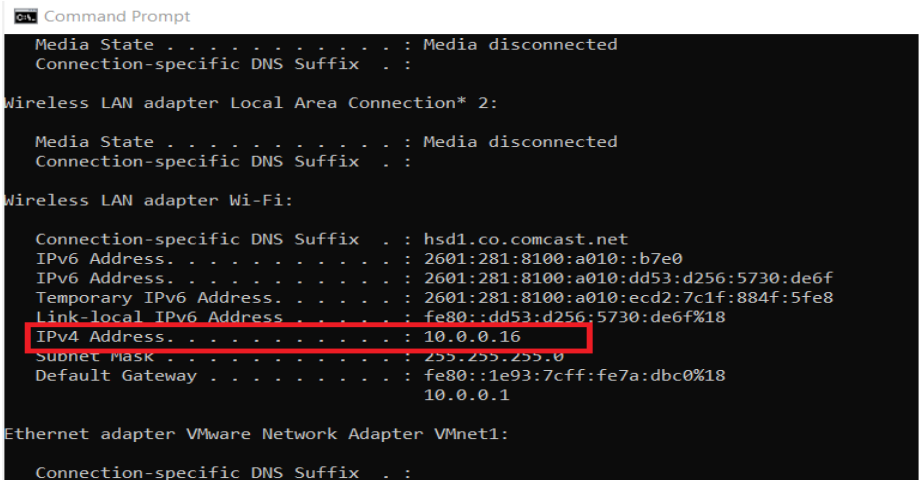



**Screenshot_3**

7. What is the IP address of your host? (Show within Wireshark) [**5 points**]

The IP Address of my host machine (aka source address) is :: 10.0.0.16

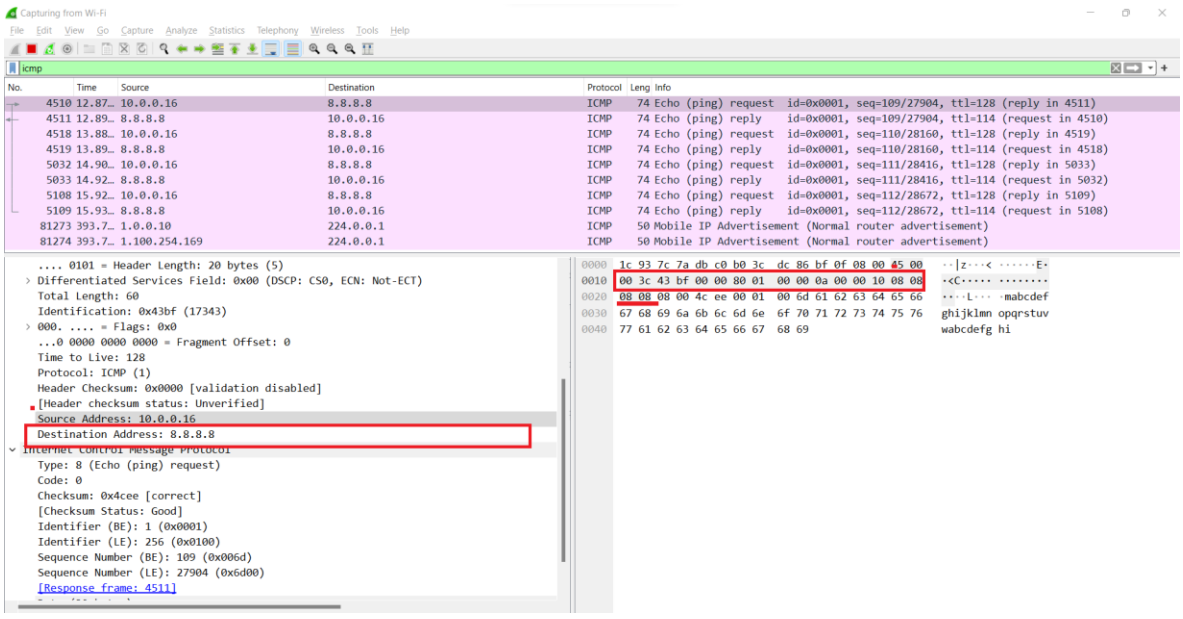


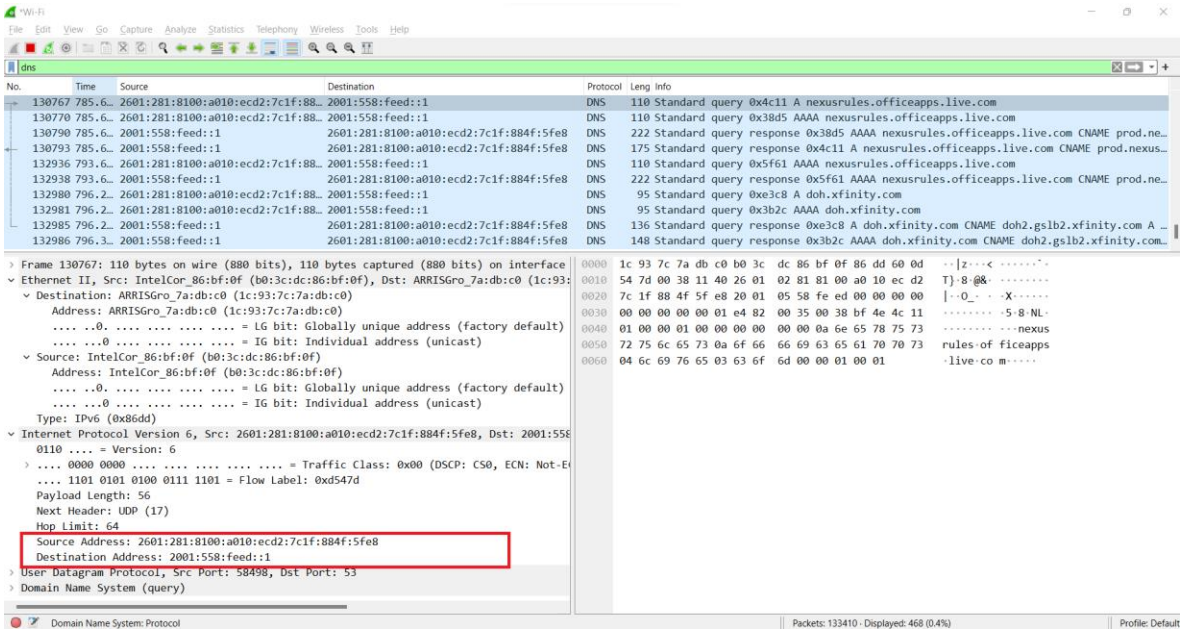



8. What is the IP address of the destination host?

    a. Show within Wireshark, and explain how this address was selected? [**10 points**]

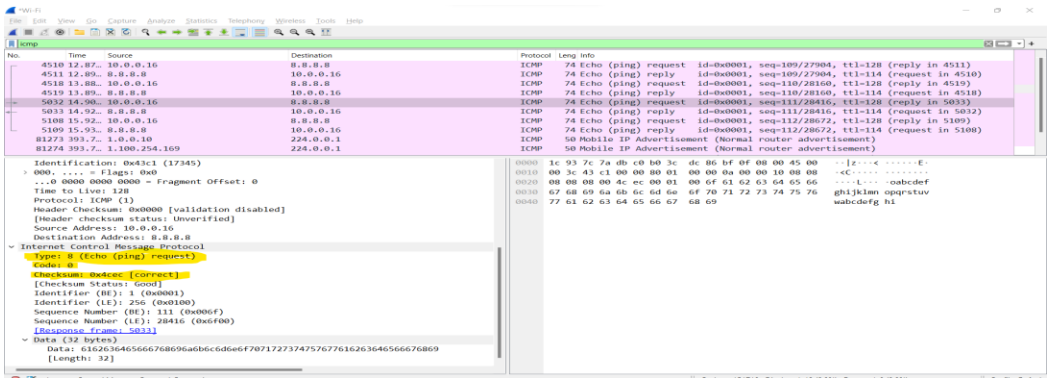
The IP Address of destination host can be found in the TCP/UDP Request, While ICMP packets do have a data section, their purpose is not to wrap and carry protocols like HTTP and DNS of the ping request or DNS request. Generally, the destination host address is the traffic that's going from your host to the public network, and depends on the network/website you access. Here, I pinged google/cisco/facebook so the destination address was selected to be the DNS address of www.google.com



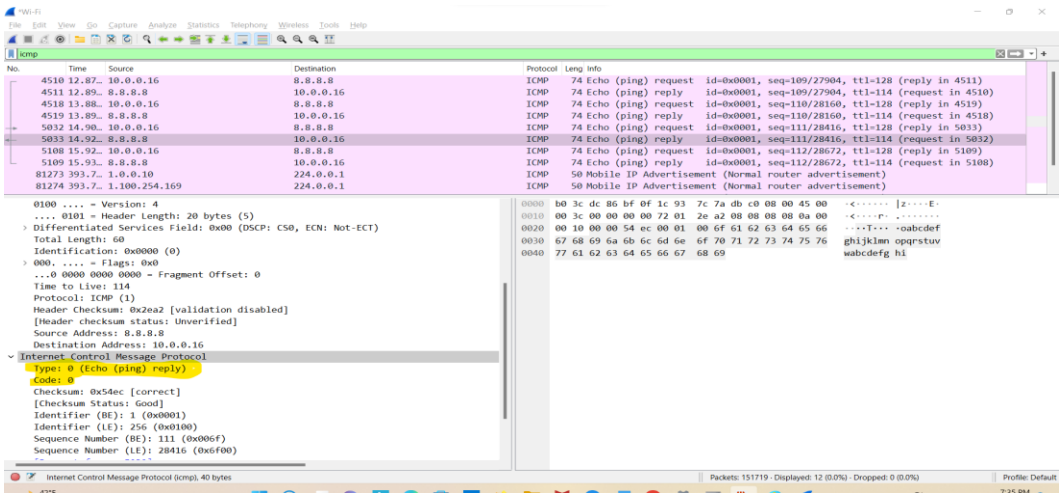
9. Examine one of the ping request packets sent by your host. What are the ICMP type and code numbers? What other fields does this ICMP packet have? How many bytes are the checksum, sequence number, and identifier fields? [**10 points**]

ICMP Packet type = 8 (Echo request) and the Code=0. The other fileds Checksum (0x4 cec) = 2 Bytes, Checksum status = [good], Identifier(BE) = 2 Bytes, Identifier (LE) = 2 Bytes, Sequence number (BE/LE)=4 Bytes. Total ICMP packet is 40 Bytes.



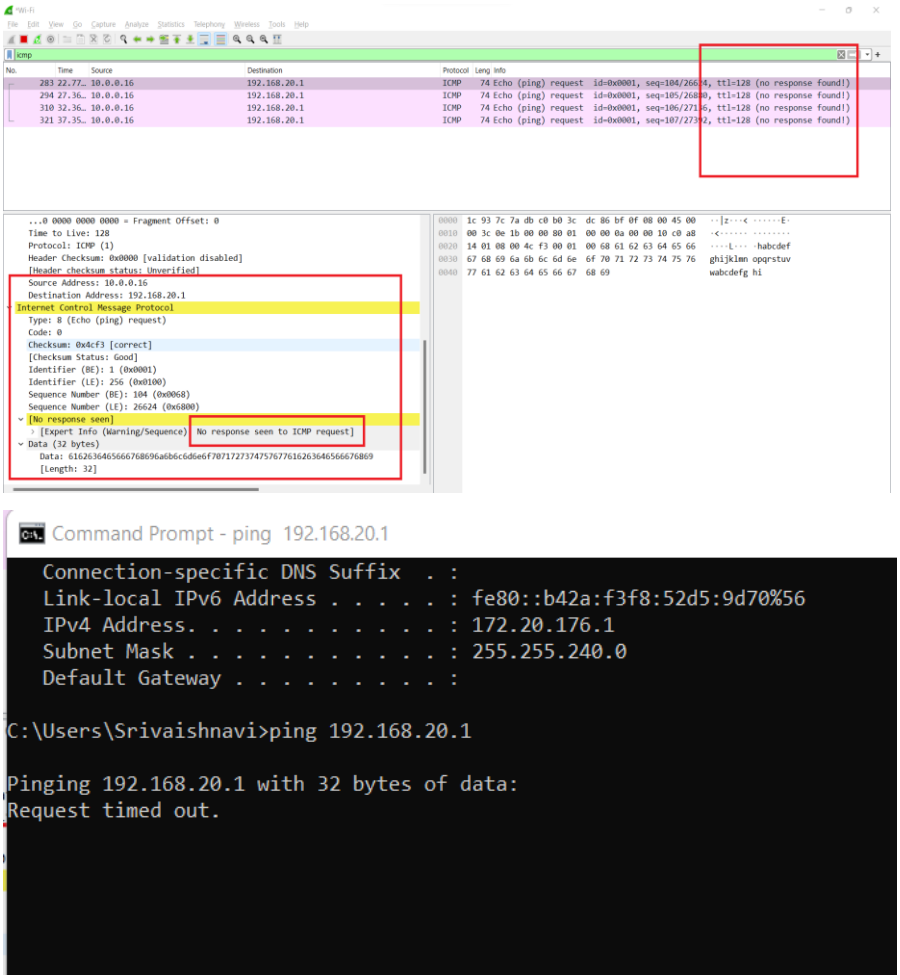
10. Examine the corresponding ping reply packet. What are the ICMP type and code numbers? What other fields does this ICMP packet have? How many bytes are the checksum, sequence number and identifier fields? [**10 points**]

ICMP Packet type = 0 (Echo reply) and the Code=0. The other fileds Checksum (0x54 ec) = 2 Bytes, Checksum status = [good], Identifier(BE) = 2 Bytes, Identifier (LE) = 2 Bytes, Sequence number (BE/LE)=4 Bytes. Total ICMP packet is 40 Bytes



11. Start a new Wireshark Capture. Ping a hostname or IP that gives you a “Request Timed Out” message. (e.g. You can try www.wellsfargo.com or any another website/IP of your choice.). Filter the ICMP traffic. Find the Type and Code of the packet in the above scenario. Paste the relevant screenshots. [**5 points**]

    The ICMP Type = 8 (Echo ping request). And the code = 0



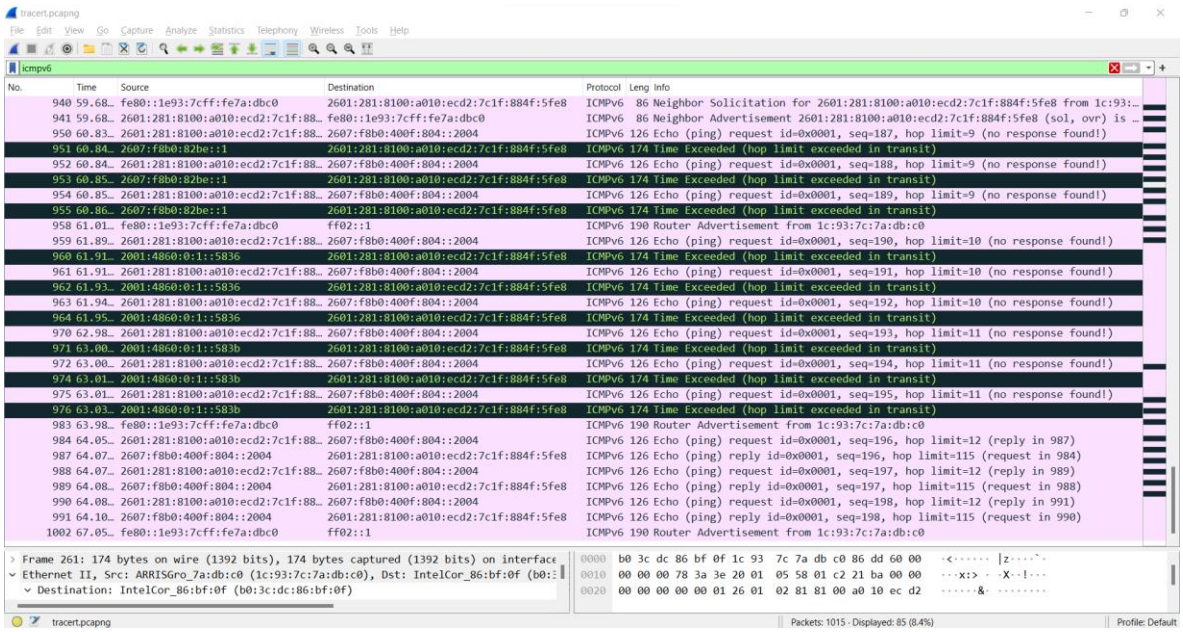

12. Do you see both ICMP Echo Request and Echo Reply messages? [**5 points**]

    Only ICMP Request packets are seen and there is no Reply message. The request message is attached with “No Response found” warning/sequence within the ICMP Packet

## Objective 2.2 – ICMP and Traceroute

1. Open command prompt/terminal (depending on the operating system)
2. Start Wireshark and begin capture
3. Traceroute to a “hostname” (where the “hostname” is a URL, example: tracert www.google.com)
4. When trace completes, stop capture.
5. Provide a screenshot of the trace. Was it successful? How do you know? [**5 points**]

   The tracert is successful after initial “Time exceeded” requests hit the ttl to 0. The ICMPv6 packet has successfully the advertised the router reply to source host with hot limit 255, payload length = 136.



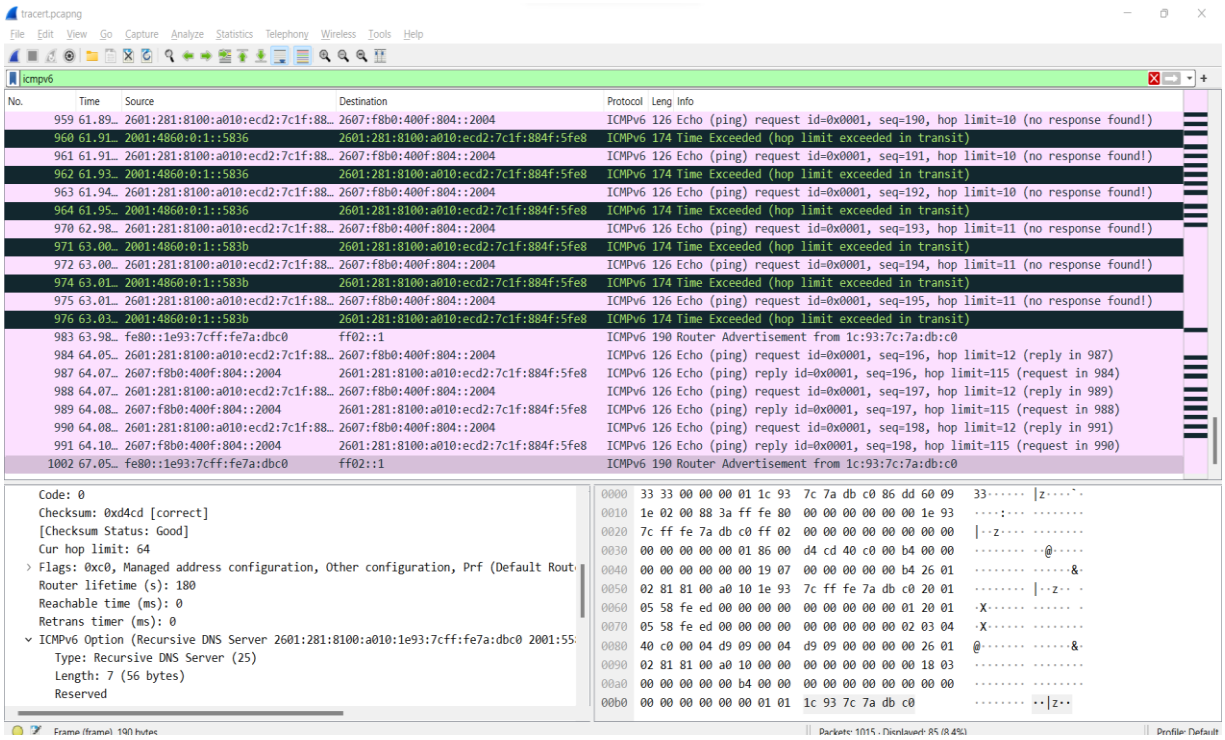

```
C:\Users\Srivaishnavi>tracert www.google.com

Tracing route to www.google.com [2607:f8b0:400f:804::2004]
over a maximum of 30 hops:

  1     4 ms     4 ms    15 ms  2601:281:8100:a010:1e93:7cff:fe7a:dbc0
  2    15 ms    10 ms    12 ms  2001:558:4070:1b::1
  3    13 ms    13 ms    10 ms  ae-252-1209-rur101.boulder.co.denver.comcast.net [2001:558:1c2:21ba::1]
  4    32 ms    14 ms    15 ms  ae-29-ar01.denver.co.denver.comcast.net [2001:558:1c0:12f::1]
  5     *        *        *     Request timed out.
  6     *        *        *     Request timed out.
  7     *        *        *     Request timed out.
  8     *       16 ms     *     as15133-5-c.nota.fl.ibone.comcast.net [2001:559::63a]
  9    12 ms    11 ms    10 ms  2607:f8b0:82be::1
 10    19 ms    21 ms    16 ms  2001:4860:0:1::5836
 11    18 ms    12 ms    14 ms  2001:4860:0:1::583b
 12    13 ms    10 ms    15 ms  den16s08-in-x04.1e100.net [2607:f8b0:400f:804::2004]

Trace complete.
```

6. Filter the Wireshark capture to only show the relevant trace route data. Examine the ICMP traffic in Wireshark. What is different in the capture from the trace when compared to the capture of the Ping in previous objective? Explain what is different between the Ping and the trace route, and how this relates to how trace route works [**15 points**]

The main difference I analyzed is - the Ping program in the source host sends a packet to the target IP address; if the target is live, the Ping program in the target host responds by sending a packet back to the source host with the ICMP packet is of Type 8 and Code 0 - a so-called ICMP "echo request" packet and Type 0, Code 0 for reply message.

Where as in Traceroute, the program sends the first packet with TTL=1, the second packet with TTL=2, and so on. Recalling what **Levi** said in the class that a router will decrement a packet's TTL value as the packet passes through the router. When a packet arrives at a router with TTL=1, the router sends an ICMP error packet back to the source.

From this figure we see that for each TTL value, the source program sends three probe packets. Traceroute displays the RTTs for each of the probe packets, as well as the IP address (and possibly the name) of the router that returned the ICMP TTL-exceeded message

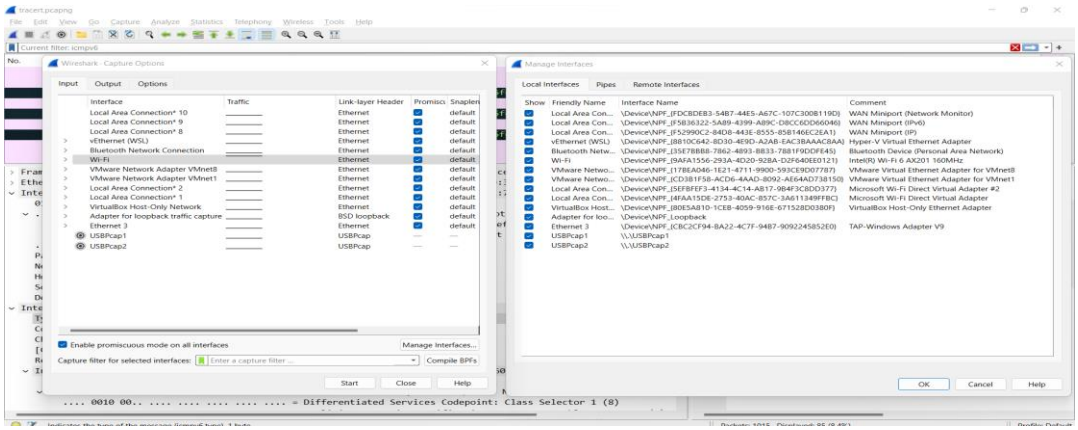


## Part 3 – Wireshark NICs and IPv4/IPv6 addresses

### Objective 3.1

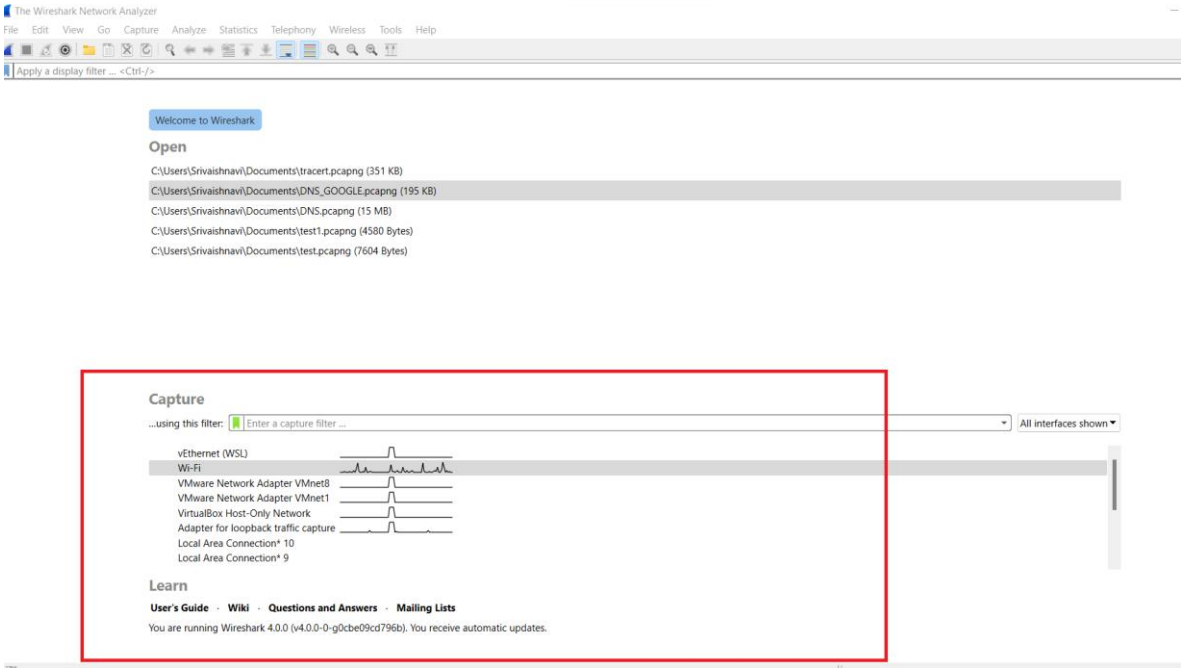
1. Provide a screenshot of the NICs that Wireshark has to choose from on the PC. **[10 points]**

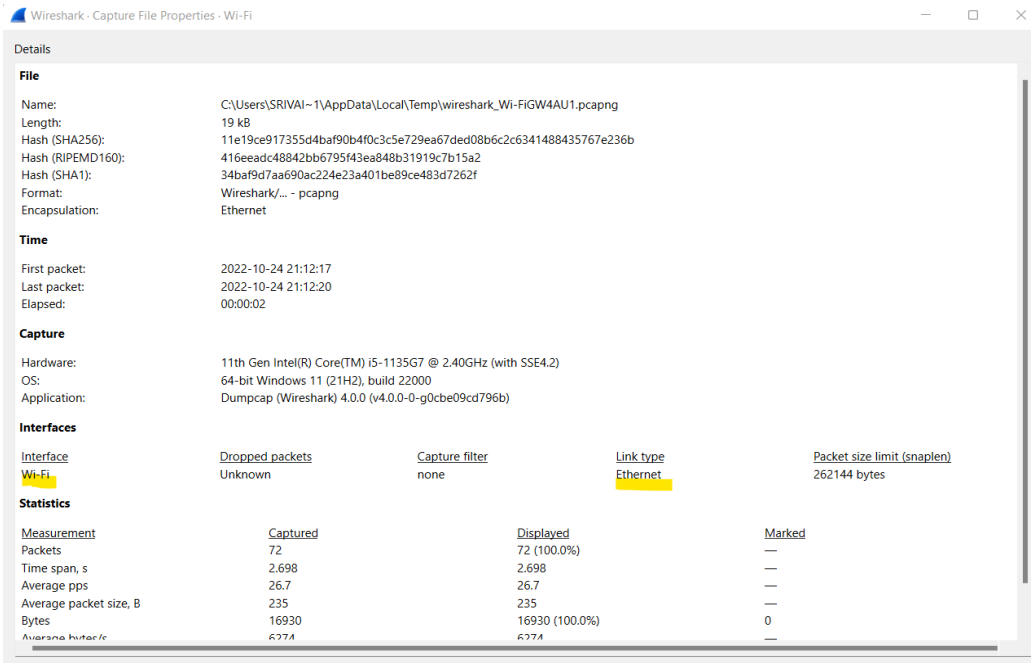
   The "Manage Interfaces" shows the NICs that Wireshark chooses interfaces

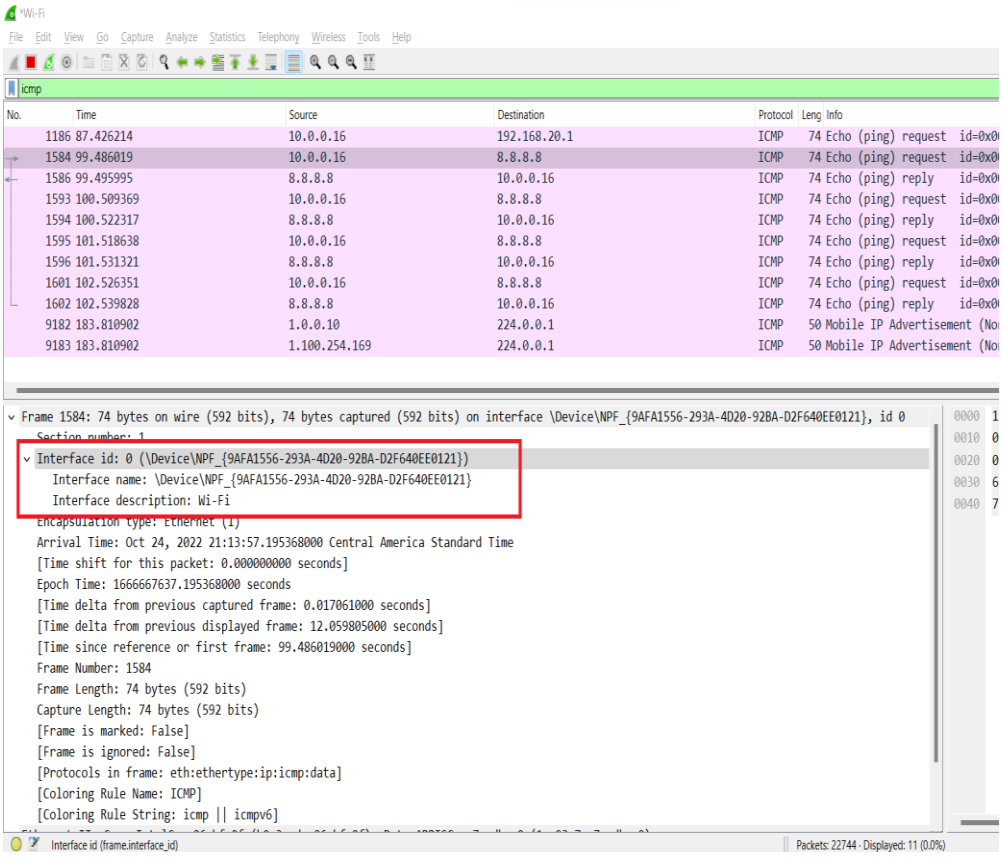


2. Which interface is currently capturing traffic? How do you know? (Provide a screenshot) **[10 points]**

   After choosing the respective interface from the home screen > statistics > Capture File Properties will show the interface on which the capture is happening or simply one can check in the TCP/UDP/ICMP/Protocol "Frame" to check the Interface ID and Name

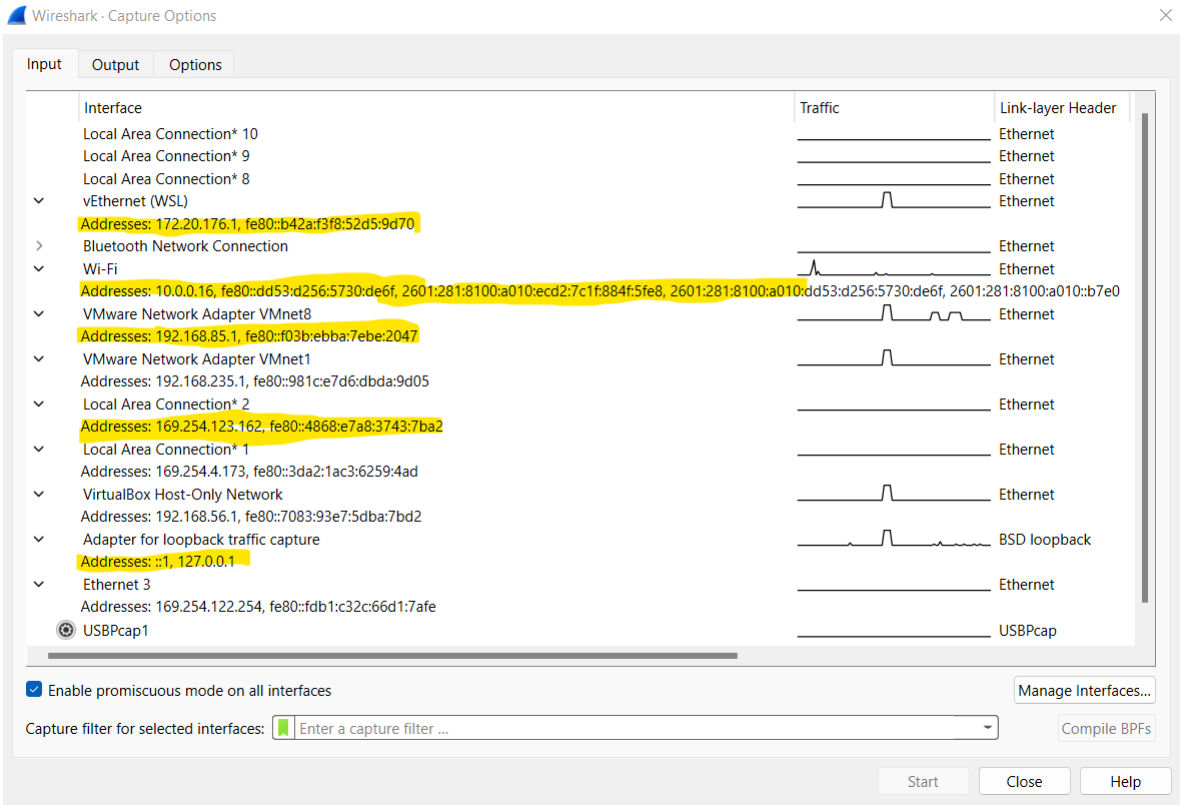



Wireshark · Capture File Properties · Wi-Fi
Details
File
Name: C:\Users\SRIVAI~1\AppData\Local\Temp\wireshark_Wi-FiGW4AU1.pcapng
Length: 19 kB
Hash (SHA256): 11e19ce917355d4baf90b4f0c3c5e729ea67ded08b6c2c63414888435767e236b
Hash (RIPEMD160): 416eeadc48842bb6795f43ea848b31919c7b15a2
Hash (SHA1): 34baf9d7aa690ac224e23a401be89ce483d7262f
Format: Wireshark/... - pcapng
Encapsulation: Ethernet
Time
First packet: 2022-10-24 21:12:17
Last packet: 2022-10-24 21:12:20
Elapsed: 00:00:02
Capture
Hardware: 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz (with SSE4.2)
OS: 64-bit Windows 11 (21H2), build 22000
Application: Dumpcap (Wireshark) 4.0.0 (v4.0.0-0-g0cbe09cd796b)
Interfaces
Interface Dropped packets Capture filter Link type Packet size limit (snaplen)
Wi-Fi Unknown none Ethernet 262144 bytes
Statistics
Measurement Captured Displayed Marked
Packets 72 72 (100.0%) —
Time span, s 2.698 2.698 —
Average pps 26.7 26.7 —
Average packet size, B 235 235 —
Bytes 16930 16930 (100.0%) 0


*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
icmp
No. Time Source Destination Protocol Leng Info
1186 87.426214 10.0.0.16 192.168.20.1 ICMP 74 Echo (ping) request
1584 99.486019 10.0.0.16 8.8.8.8 ICMP 74 Echo (ping) request
1586 99.495995 8.8.8.8 10.0.0.16 ICMP 74 Echo (ping) reply
1593 100.509369 10.0.0.16 8.8.8.8 ICMP 74 Echo (ping) request
1594 100.522317 8.8.8.8 10.0.0.16 ICMP 74 Echo (ping) reply
1595 101.518638 10.0.0.16 8.8.8.8 ICMP 74 Echo (ping) request
1596 101.531321 8.8.8.8 10.0.0.16 ICMP 74 Echo (ping) reply
1601 102.526351 10.0.0.16 8.8.8.8 ICMP 74 Echo (ping) request
1602 102.539828 8.8.8.8 10.0.0.16 ICMP 74 Echo (ping) reply
9182 183.810902 1.0.0.10 224.0.0.1 ICMP 50 Mobile IP Advertisement
9183 183.810902 1.100.254.169 224.0.0.1 ICMP 50 Mobile IP Advertisement
Frame 1584: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface \Device\NPF_{9AFA1556-293A-4D20-92BA-D2F640EE0121}, id 0
Interface id: 0 (\Device\NPF_{9AFA1556-293A-4D20-92BA-D2F640EE0121})
Interface name: \Device\NPF_{9AFA1556-293A-4D20-92BA-D2F640EE0121}
Interface description: Wi-Fi
Encapsulation type: Ethernet (1)
Arrival Time: Oct 24, 2022 21:13:57.195368000 Central America Standard Time
[Time shift for this packet: 0.000000000 seconds]
Epoch Time: 1666667637.195368000 seconds
[Time delta from previous captured frame: 0.017061000 seconds]
[Time delta from previous displayed frame: 12.059805000 seconds]
[Time since reference or first frame: 99.486019000 seconds]
Frame Number: 1584
Frame Length: 74 bytes (592 bits)
Capture Length: 74 bytes (592 bits)
[Frame is marked: False]
[Frame is ignored: False]
[Protocols in frame: eth:ethertype:ip:icmp:data]
[Coloring Rule Name: ICMP]
[Coloring Rule String: icmp || icmpv6]
Interface id (frame.interface_id)
Packets: 22744 · Displayed: 11 (0.0%)

3. What is the easiest way to determine the IPv4/IPv6 address of the NICs before a capture is started? Provide a screenshot where some NICs show IPv6 addresses and some show IPv4 addresses. **[10 points]**

Hovering over an interface will show any associated IPv4 and IPv6 addresses and its capture filter. Or one can open up the "Manage Interfaces" to check the Ipv6 and Ipv4 address associated with the interface.

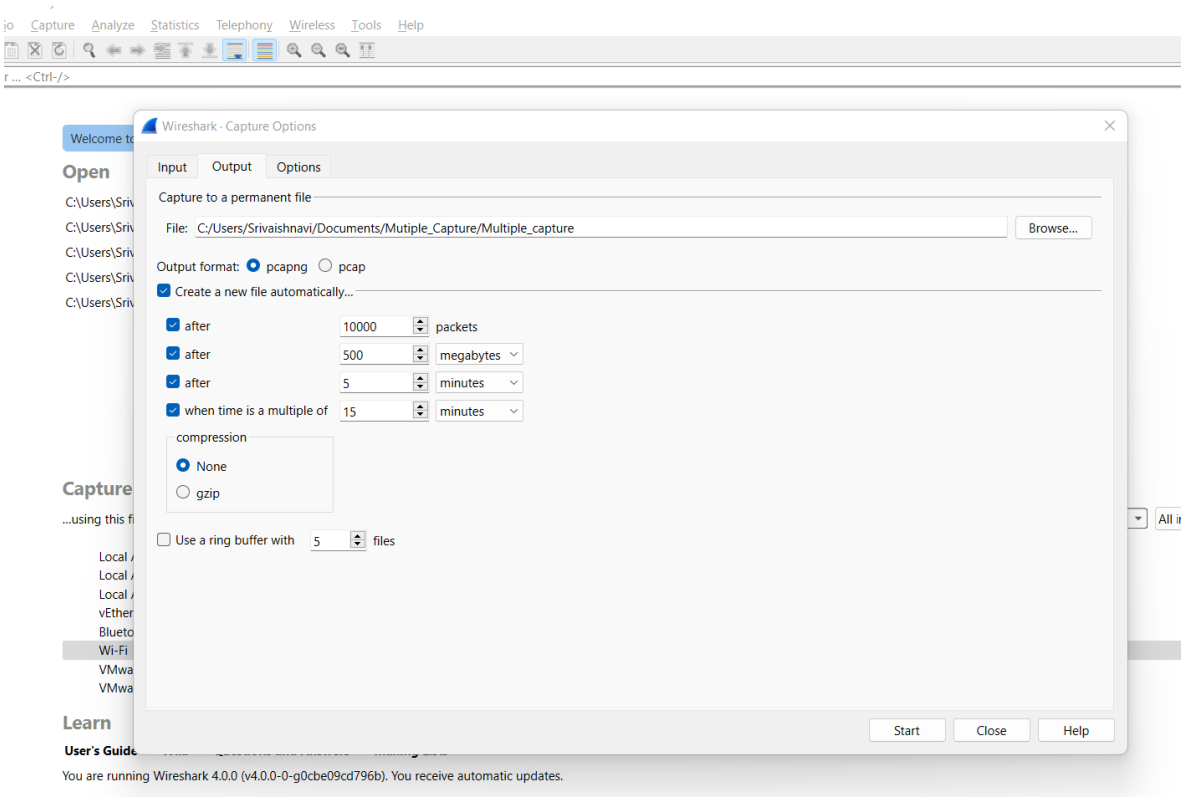



## Part 4 – Continuous Captures, Filtering, and Analysis

### Objective 4.1

1. Initiate a Wireshark capture that uses multiple files, where it creates a new file every 5 minutes, for a total of 15 minutes. (Provide a screenshot of the Capture Options you selected). Remember where you save this file, as we will use it in the future. Also, try to use the wireless NIC if possible, as there will be more traffic in the capture to analyze. **[15 points]**

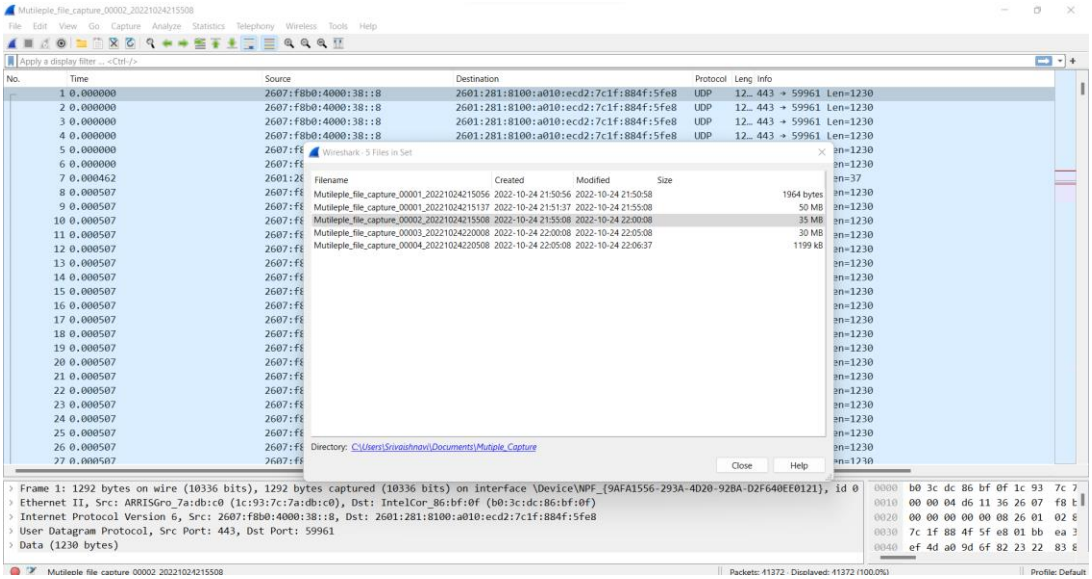
Choose wireless (Wifi interface) > Capture options > Set the time/bytes > capture




2. Browse ten different websites, during this 15-minute continual capture time window.
3. What are THREE reasons why you would want to create multiple files? **[15 points]**
   1. packet capture across multiple files is handy for systems limited in storage/memory/diskspace preventing one large file from using the memory
   2. Working with large files (of hundreds of MB's) can be slow so to do long captures from a high network traffic using "Multiple files" is the best option as it spreads the captured data over smaller files which makes it easy to analyze the captured traffic
   3. It's easy to compare two or more packets, their latency, loss and converge when captured over multiple files.

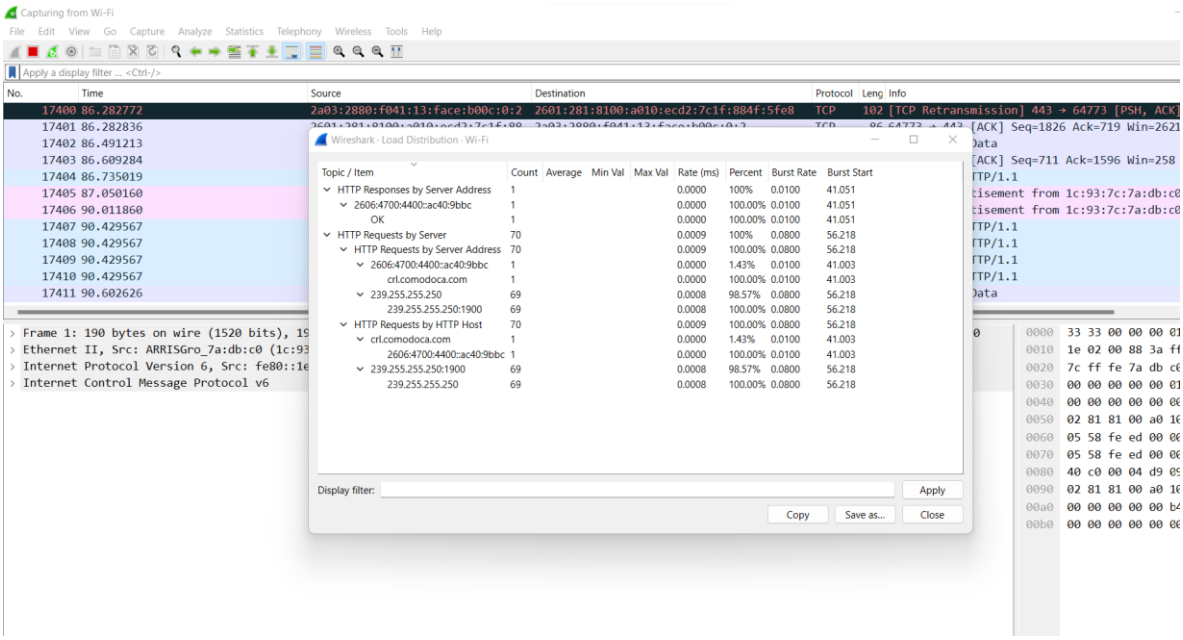
4. How do you view the three files captured within Wireshark, and move between them, after they have been completed and saved? (Hint: File > File Set) **[10 points]**

**File > File Set > List Files > Next File/Previous File**



5. How do you see which websites you browsed during the capture? (Hint: Statistics > HTTP) **[5 points]**

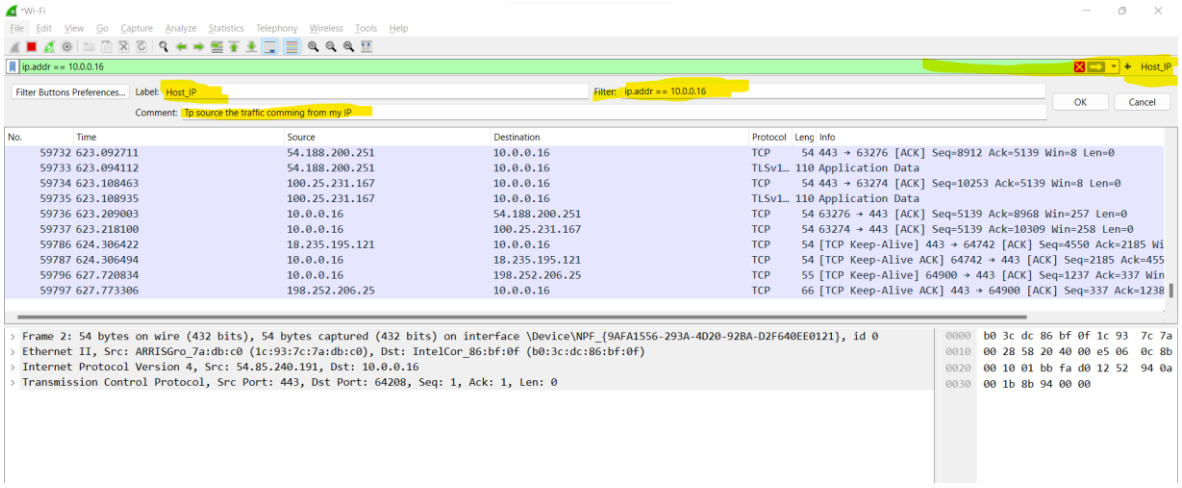
**Statistics > HTTP > Load Distribution > HTTP Requests by Host**



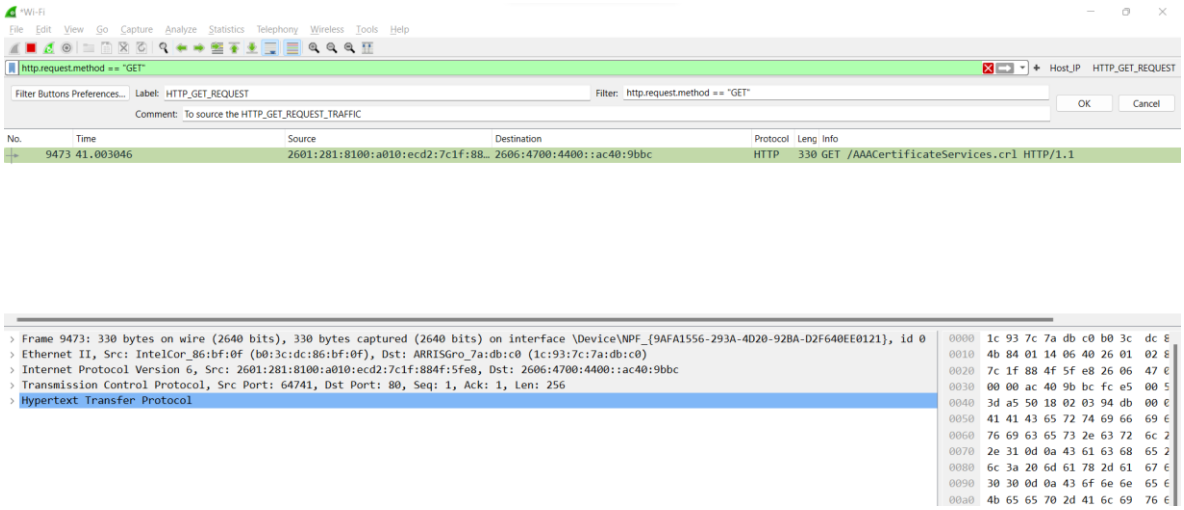
## Objective 4.2

1. Create two Display filter buttons. One for traffic sourced from your machine's IP address and one that only displays HTTP GET requests. (Hint: HTTP contains)
2. Provide a screenshot of the buttons you created, and the corresponding filtered capture. **[10 points]**

**Ip.addr == 10.0.0.16 (my host IP)**

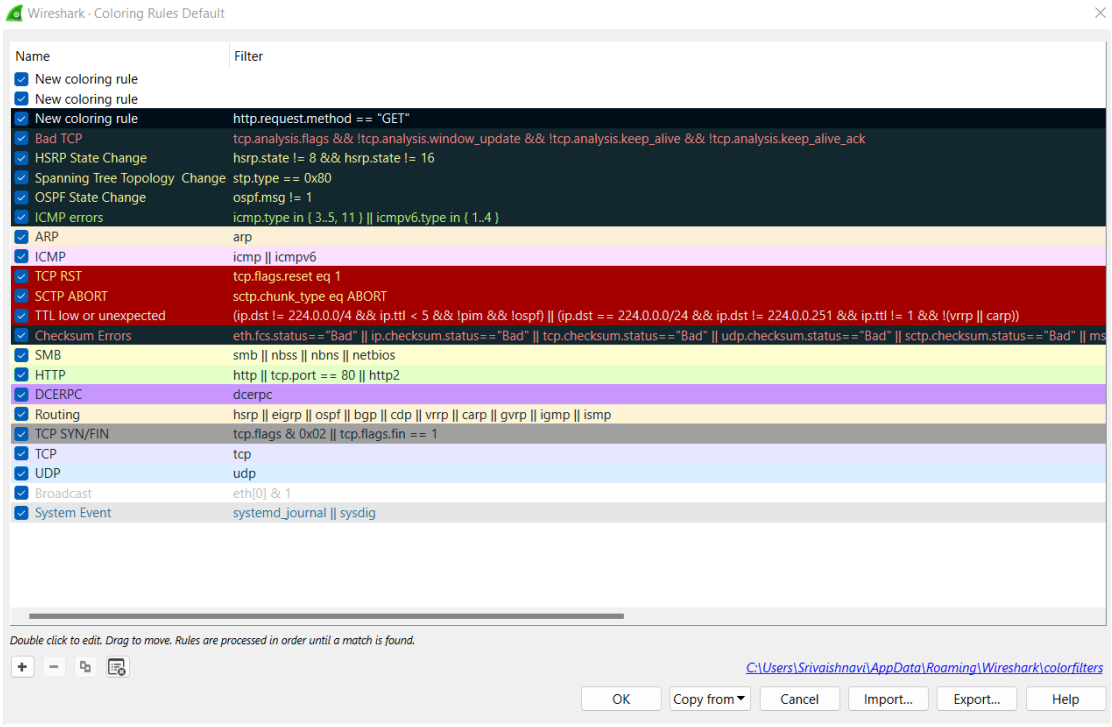



**http.request.method == "GET"**

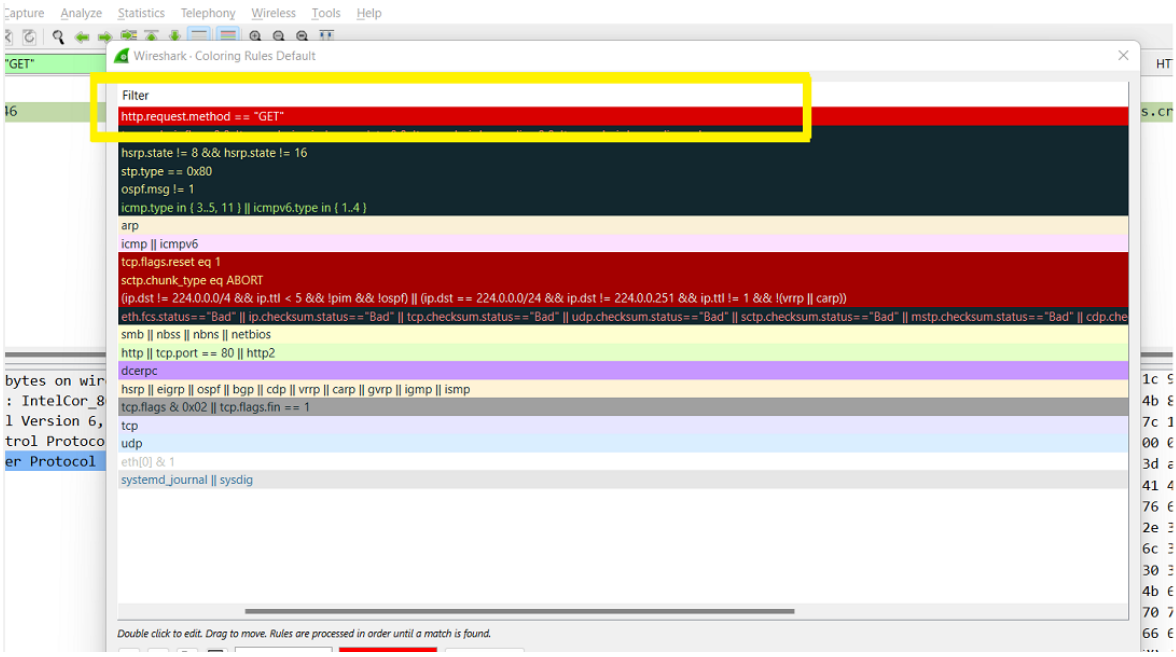


## Objective 4.3

1. Create a coloring rule for HTTP traffic. **[5 points]**




2. Provide a screenshot of your capture from above, showing where you changed the color of HTTP GET requests to Red background with White lettering. (Hint: Did you remember to move your color rule to the top?) **[10 points]**

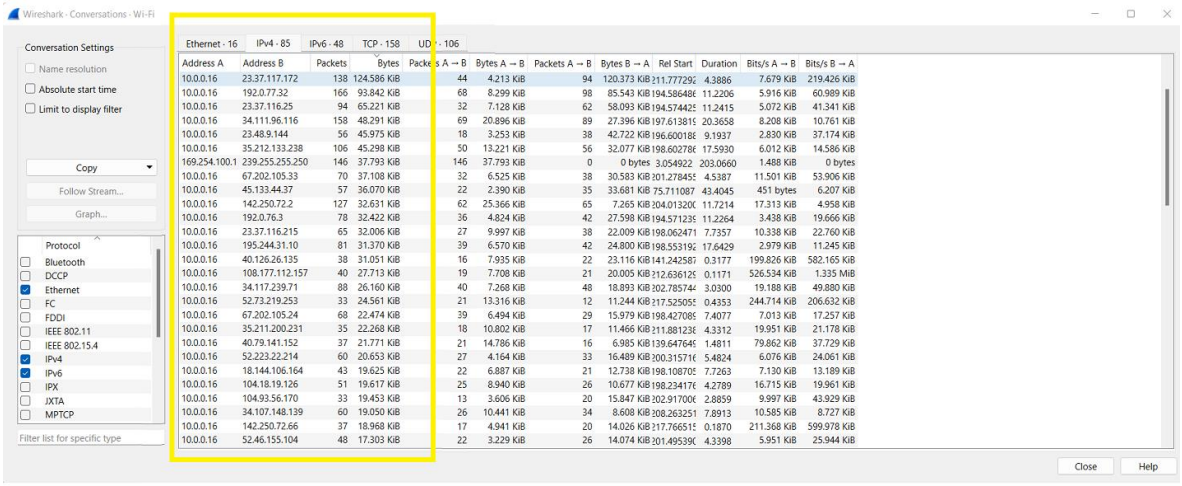


# Part 5 - Top Talkers, Profiles, and Graphs

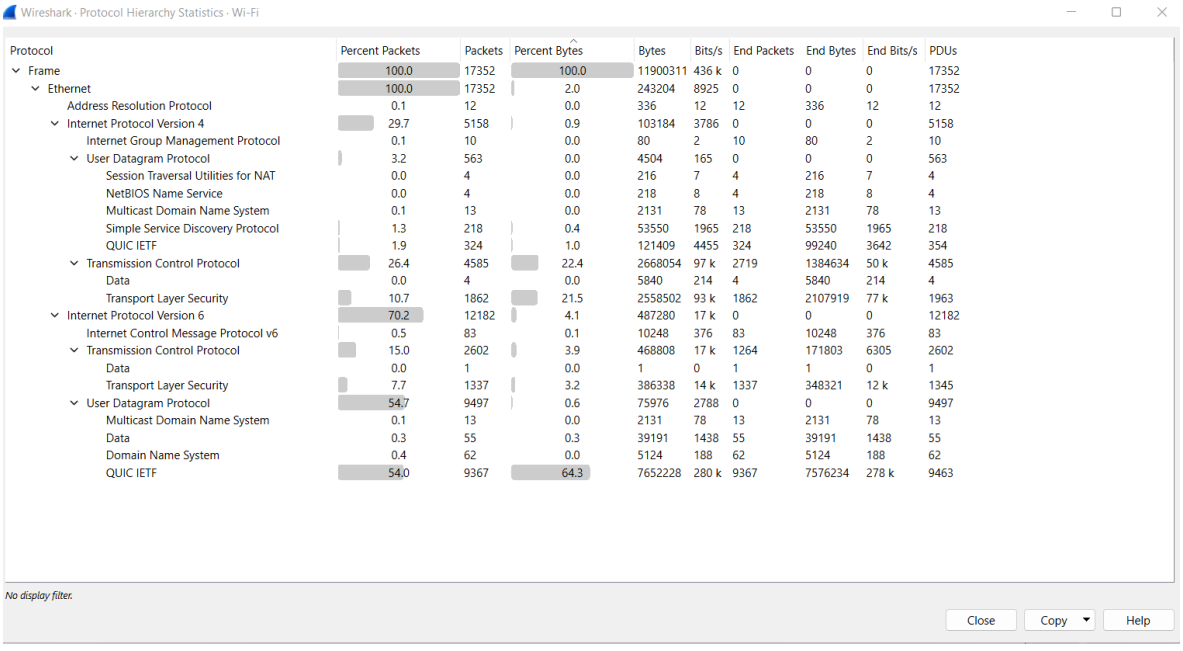
## Objective 5.1

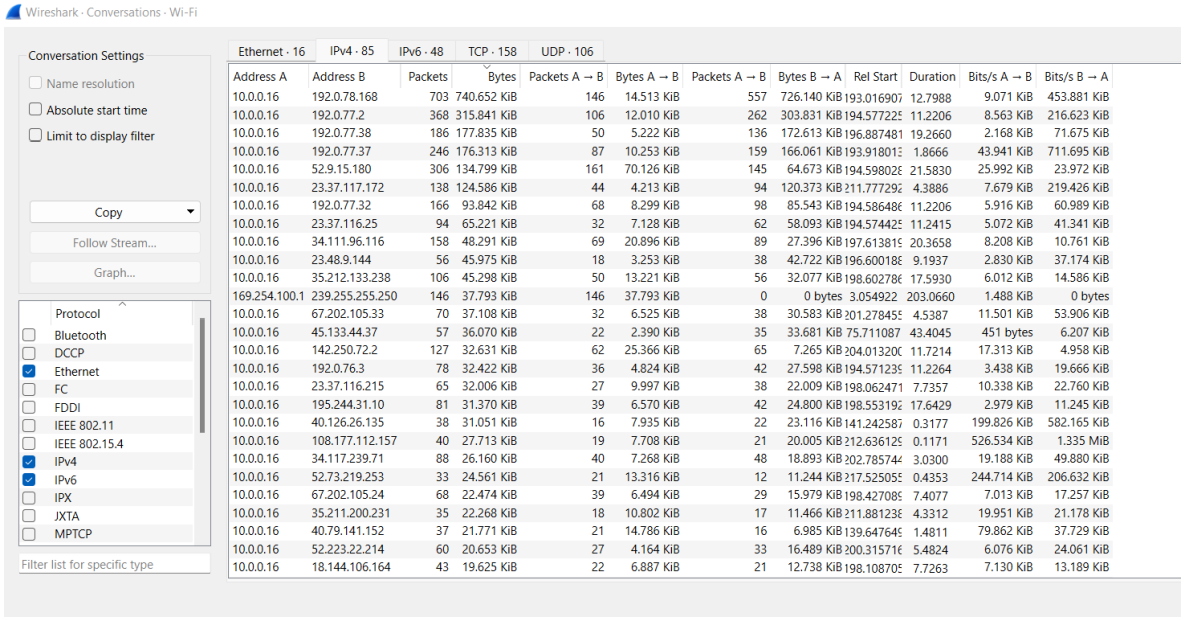
1. Determine Top Talkers on the network. **[10 points]**

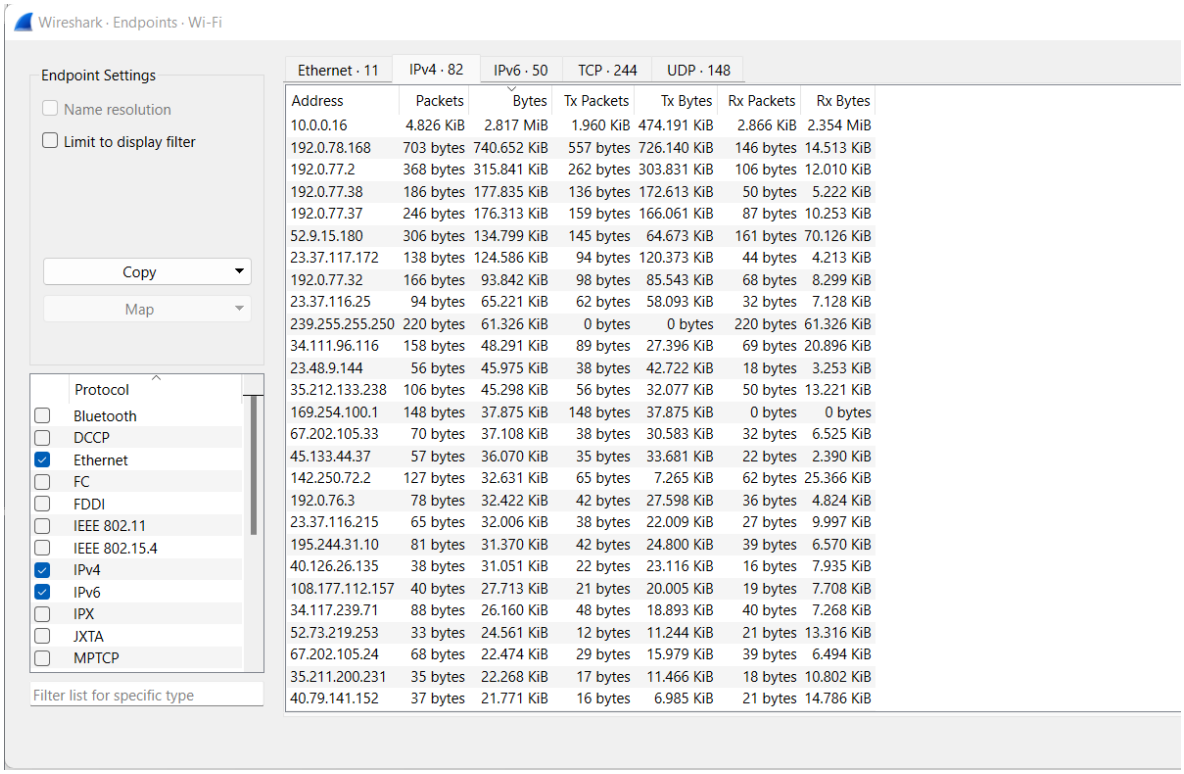
   **Statistic > Coversation > Choose the protocol > Bytes (Sort desc) > Top Talkers**




2. What are two ways you can determine what device/IP address is transmitting the most traffic on the network? Provide a screenshot of one of those ways. (Hint: Protocol Hierarchy; Conversations; Endpoints) **[10 points]**

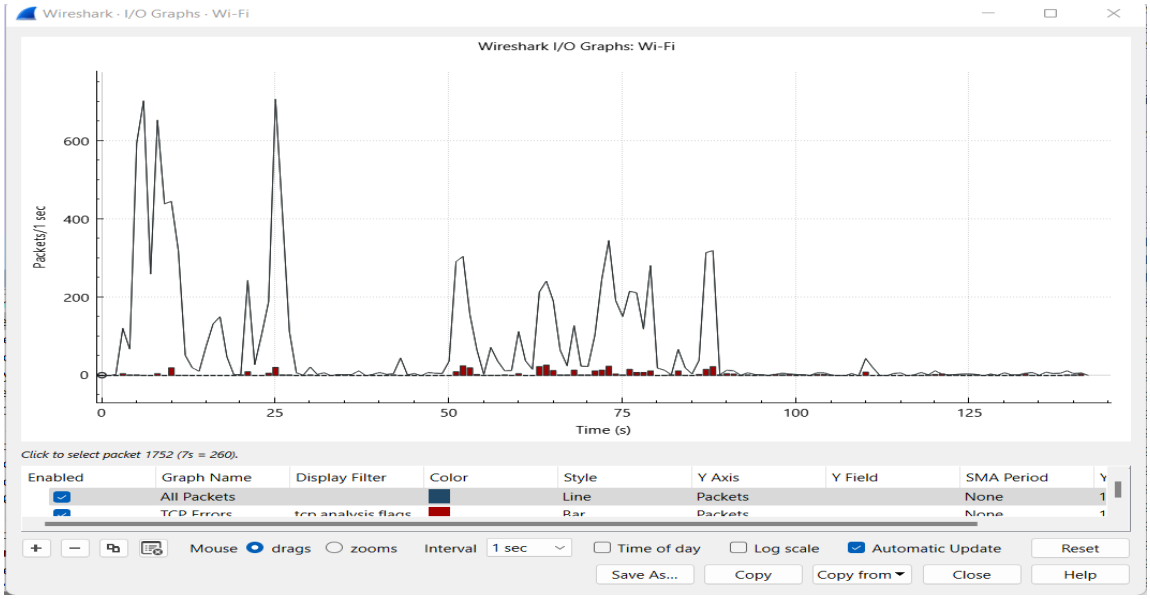



Wireshark · Conversations · Wi-Fi
Conversation Settings
Name resolution
Absolute start time
Limit to display filter
Copy
Follow Stream...
Graph...
Protocol
Bluetooth
DCCP
Ethernet
FC
FDDI
IEEE 802.11
IEEE 802.15.4
IPv4
IPv6
IPX
JXTA
MPTCP
Filter list for specific type
Ethernet · 16
IPv4 · 85
IPv6 · 48
TCP · 158
UDP · 106
Address A
Address B
Packets
Bytes
Packets A → B
Bytes A → B
Packets A → B
Bytes B → A
Rel Start
Duration
Bits/s A → B
Bits/s B → A
10.0.0.16 192.0.78.168 703 740.652 KiB 146 14.513 KiB 557 726.140 KiB 193.016907 12.7988 9.071 KiB 453.881 KiB
10.0.0.16 192.0.77.2 368 315.841 KiB 106 12.010 KiB 262 303.831 KiB 194.577225 11.2206 8.563 KiB 216.623 KiB
10.0.0.16 192.0.77.38 186 177.835 KiB 50 5.222 KiB 136 172.613 KiB 196.887481 19.2660 2.168 KiB 71.675 KiB
10.0.0.16 192.0.77.37 246 176.313 KiB 87 10.253 KiB 159 166.061 KiB 193.918013 1.8666 43.941 KiB 711.695 KiB
10.0.0.16 52.9.15.180 306 134.799 KiB 161 70.126 KiB 145 64.673 KiB 194.598028 21.5830 25.992 KiB 23.972 KiB
10.0.0.16 23.37.117.172 138 124.586 KiB 44 4.213 KiB 94 120.373 KiB 211.777292 4.3886 7.679 KiB 219.426 KiB
10.0.0.16 192.0.77.32 166 93.842 KiB 68 8.299 KiB 98 85.543 KiB 194.586486 11.2206 5.916 KiB 60.989 KiB
10.0.0.16 23.37.116.25 94 65.221 KiB 32 7.128 KiB 62 58.093 KiB 194.574425 11.2415 5.072 KiB 41.341 KiB
10.0.0.16 34.111.96.116 158 48.291 KiB 69 20.896 KiB 89 27.396 KiB 197.613815 20.3658 8.208 KiB 10.761 KiB
10.0.0.16 23.48.9.144 56 45.975 KiB 18 3.253 KiB 38 42.722 KiB 196.600186 9.1937 2.830 KiB 37.174 KiB
10.0.0.16 35.212.133.238 106 45.298 KiB 50 13.221 KiB 56 32.077 KiB 198.602786 17.5930 6.012 KiB 14.586 KiB
169.254.100.1 239.255.255.250 146 37.793 KiB 146 37.793 KiB 0 0 bytes 3.054922 203.0660 1.488 KiB 0 bytes
10.0.0.16 67.202.105.33 70 37.108 KiB 32 6.525 KiB 38 30.583 KiB 201.278455 4.5387 11.501 KiB 53.906 KiB
10.0.0.16 45.133.44.37 57 36.070 KiB 22 2.390 KiB 35 33.681 KiB 75.711087 43.4045 451 bytes 6.207 KiB
10.0.0.16 142.250.72.2 127 32.631 KiB 62 25.366 KiB 65 7.265 KiB 204.013200 11.7214 17.313 KiB 4.958 KiB
10.0.0.16 192.0.76.3 78 32.422 KiB 36 4.824 KiB 42 27.598 KiB 194.571235 11.2264 3.438 KiB 19.666 KiB
10.0.0.16 23.37.116.215 65 32.006 KiB 27 9.997 KiB 38 22.009 KiB 198.062471 7.7357 10.338 KiB 22.760 KiB
10.0.0.16 195.244.31.10 81 31.370 KiB 39 6.570 KiB 42 24.800 KiB 198.553192 17.6429 2.979 KiB 11.245 KiB
10.0.0.16 40.126.26.135 38 31.051 KiB 16 7.935 KiB 22 23.116 KiB 141.242587 0.3177 199.826 KiB 582.165 KiB
10.0.0.16 108.177.112.157 40 27.713 KiB 19 7.708 KiB 21 20.005 KiB 212.636125 0.1171 526.534 KiB 1.335 MiB
10.0.0.16 34.117.239.71 88 26.160 KiB 40 7.268 KiB 48 18.893 KiB 202.785744 3.0300 19.188 KiB 49.880 KiB
10.0.0.16 52.73.219.253 33 24.561 KiB 21 13.316 KiB 12 11.244 KiB 217.525055 0.4353 244.714 KiB 206.632 KiB
10.0.0.16 67.202.105.24 68 22.474 KiB 39 6.494 KiB 29 15.979 KiB 198.427085 7.4077 7.013 KiB 17.257 KiB
10.0.0.16 35.211.200.231 35 22.268 KiB 18 10.802 KiB 17 11.466 KiB 211.881238 4.3312 19.951 KiB 21.178 KiB
10.0.0.16 40.79.141.152 37 21.771 KiB 21 14.786 KiB 16 6.985 KiB 139.647645 1.4811 79.862 KiB 37.729 KiB
10.0.0.16 52.223.22.214 60 20.653 KiB 27 4.164 KiB 33 16.489 KiB 200.315716 5.4824 6.076 KiB 24.061 KiB
10.0.0.16 18.144.106.164 43 19.625 KiB 22 6.887 KiB 21 12.738 KiB 198.108705 7.7263 7.130 KiB 13.189 KiB


Wireshark · Endpoints · Wi-Fi
Endpoint Settings
Name resolution
Limit to display filter
Copy
Map
Protocol
Bluetooth
DCCP
Ethernet
FC
FDDI
IEEE 802.11
IEEE 802.15.4
IPv4
IPv6
IPX
JXTA
MPTCP
Filter list for specific type
Ethernet · 11
IPv4 · 82
IPv6 · 50
TCP · 244
UDP · 148
Address
Packets
Bytes
Tx Packets
Tx Bytes
Rx Packets
Rx Bytes
10.0.0.16 4.826 KiB 2.817 MiB 1.960 KiB 474.191 KiB 2.866 KiB 2.354 MiB
192.0.78.168 703 bytes 740.652 KiB 557 bytes 726.140 KiB 146 bytes 14.513 KiB
192.0.77.2 368 bytes 315.841 KiB 262 bytes 303.831 KiB 106 bytes 12.010 KiB
192.0.77.38 186 bytes 177.835 KiB 136 bytes 172.613 KiB 50 bytes 5.222 KiB
192.0.77.37 246 bytes 176.313 KiB 159 bytes 166.061 KiB 87 bytes 10.253 KiB
52.9.15.180 306 bytes 134.799 KiB 145 bytes 64.673 KiB 161 bytes 70.126 KiB
23.37.117.172 138 bytes 124.586 KiB 94 bytes 120.373 KiB 44 bytes 4.213 KiB
192.0.77.32 166 bytes 93.842 KiB 98 bytes 85.543 KiB 68 bytes 8.299 KiB
23.37.116.25 94 bytes 65.221 KiB 62 bytes 58.093 KiB 32 bytes 7.128 KiB
239.255.255.250 220 bytes 61.326 KiB 0 bytes 0 bytes 220 bytes 61.326 KiB
34.111.96.116 158 bytes 48.291 KiB 89 bytes 27.396 KiB 69 bytes 20.896 KiB
23.48.9.144 56 bytes 45.975 KiB 38 bytes 42.722 KiB 18 bytes 3.253 KiB
35.212.133.238 106 bytes 45.298 KiB 56 bytes 32.077 KiB 50 bytes 13.221 KiB
169.254.100.1 148 bytes 37.875 KiB 148 bytes 37.875 KiB 0 bytes 0 bytes
67.202.105.33 70 bytes 37.108 KiB 38 bytes 30.583 KiB 32 bytes 6.525 KiB
45.133.44.37 57 bytes 36.070 KiB 35 bytes 33.681 KiB 22 bytes 2.390 KiB
142.250.72.2 127 bytes 32.631 KiB 65 bytes 7.265 KiB 62 bytes 25.366 KiB
192.0.76.3 78 bytes 32.422 KiB 42 bytes 27.598 KiB 36 bytes 4.824 KiB
23.37.116.215 65 bytes 32.006 KiB 38 bytes 22.009 KiB 27 bytes 9.997 KiB
195.244.31.10 81 bytes 31.370 KiB 42 bytes 24.800 KiB 39 bytes 6.570 KiB
40.126.26.135 38 bytes 31.051 KiB 22 bytes 23.116 KiB 16 bytes 7.935 KiB
108.177.112.157 40 bytes 27.713 KiB 21 bytes 20.005 KiB 19 bytes 7.708 KiB
34.117.239.71 88 bytes 26.160 KiB 48 bytes 18.893 KiB 40 bytes 7.268 KiB
52.73.219.253 33 bytes 24.561 KiB 12 bytes 11.244 KiB 21 bytes 13.316 KiB
67.202.105.24 68 bytes 22.474 KiB 29 bytes 15.979 KiB 39 bytes 6.494 KiB
35.211.200.231 35 bytes 22.268 KiB 17 bytes 11.466 KiB 18 bytes 10.802 KiB
40.79.141.152 37 bytes 21.771 KiB 16 bytes 6.985 KiB 21 bytes 14.786 KiB

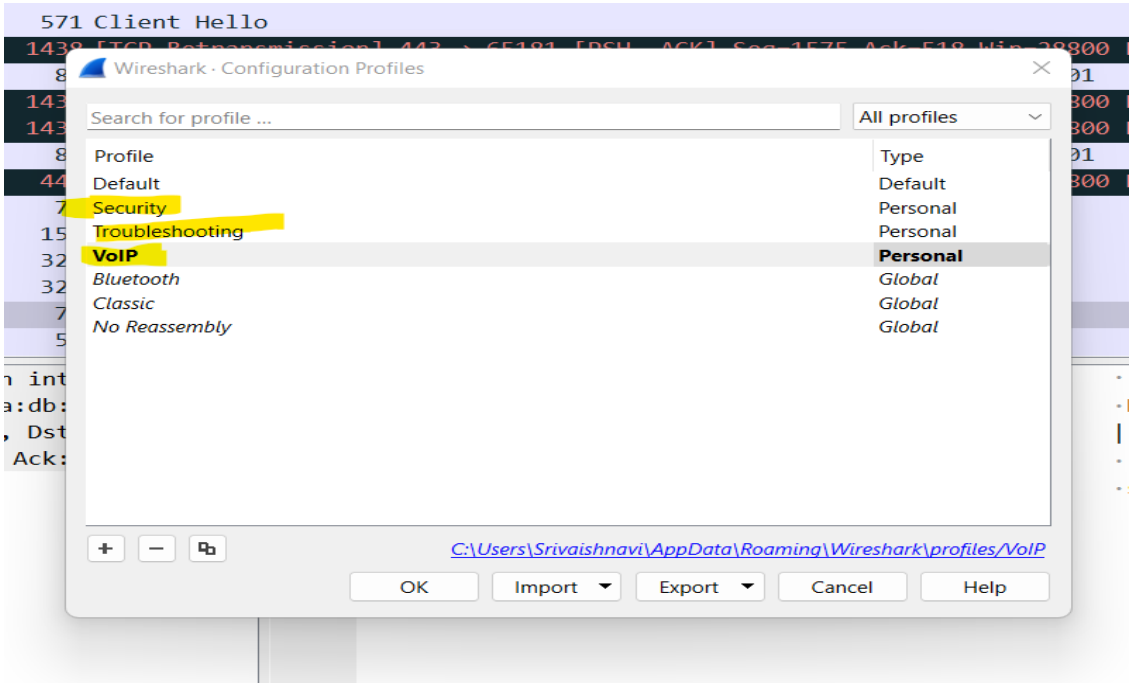
## Objective 5.2

1. Create a graph that displays the top 4 protocols from the capture.
2. Provide screenshot of the graph. **[10 points]**



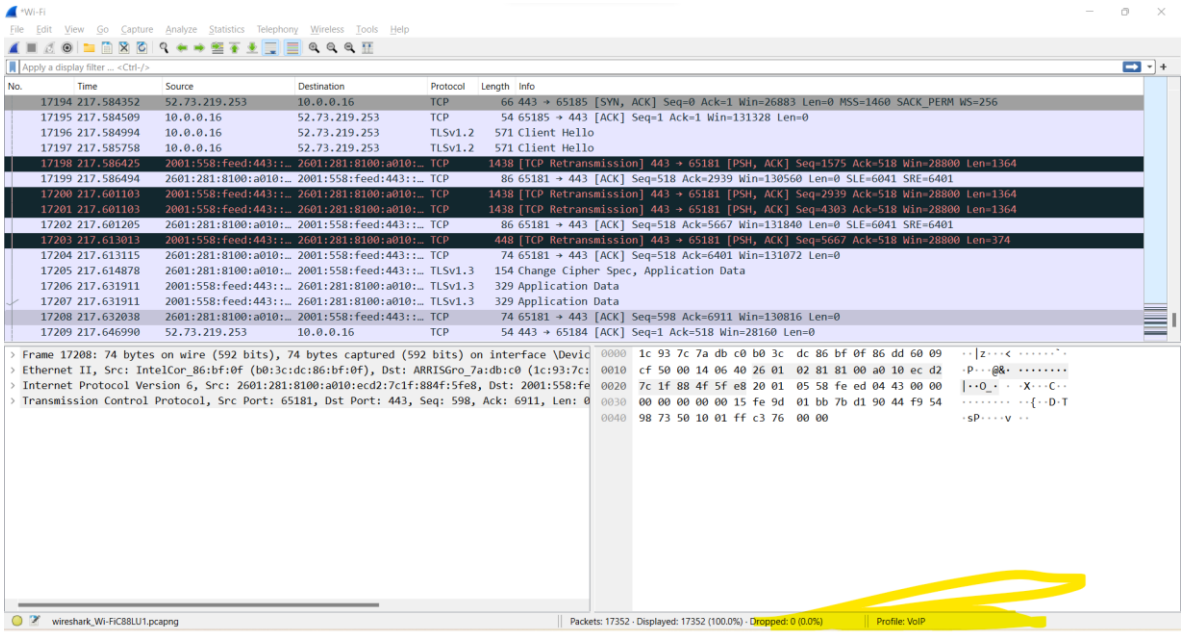

## Objective 5.3

1. Create three profiles that you will use for future analysis objectives. For example, Security, Troubleshooting, VoIP, etc. **[10 points]**



2. Explain what you would use each profile for, what you changed, and provide a screenshot of one of them. **[10 points]**

   **I would use Security profile to capture security/ firewall related data, VoIP profile will be used to capture the real time traffic and Troubleshooting will be used to troubleshoot the data. I added the profiles in Edit > configurations and changed / turned off "Auto Update", "Time Display Formatt", I turned off "Packet Bytes" and "Details"**




## Part 6 - DHCP Release and Renew

### Objective 6.1

1. Start Wireshark and begin capture
2. Release the DHCP IP address your machine has obtained
3. Renew the DHCP IP address (for your machine to obtain a new address)
4. After your machine receives an IP address from the DHCP server, stop the capture

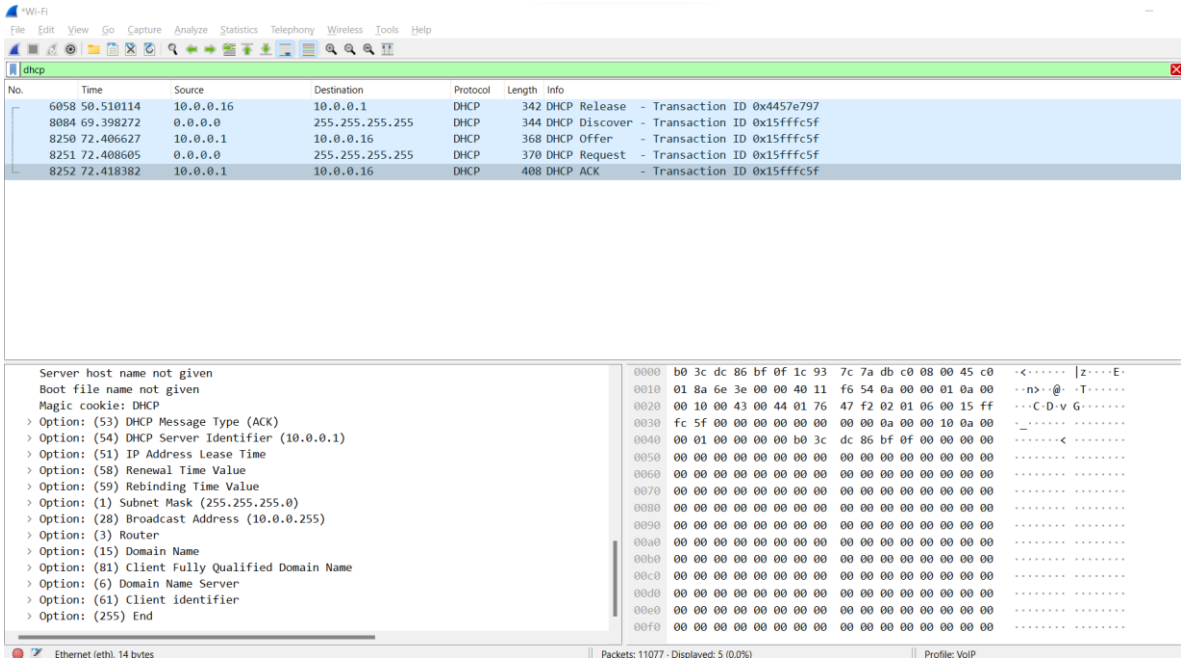
5. Filter the capture to only show the DHCP traffic. From the capture indicate the following:

   i. DHCP server address [**2 points**] :: **10.0.0.1**

   ii. The IP address your machine was offered, accepted [**2 points**] : **10.0.0.16**

   iii. Explain the DHCP process, include a screenshot [**10 points**]

   When a device wants access to a network it sends a request for an IP address to a DHCP server. The server responds be delivering an IP address to the device, then monitors the use of the address and takes it back after a specified time or when the device shuts down. The IP address is then returned to the pool of addresses managed by the DHCP server to be reassigned to another device as it seeks access to the network. It follows the Discover to discover the pool of address, and offer it to the requesting network and once the offer request is Acknowledged (ACK), the dhcp process ifs finished.




## Part 7 – Web traffic (HTTP) and TCP Connection

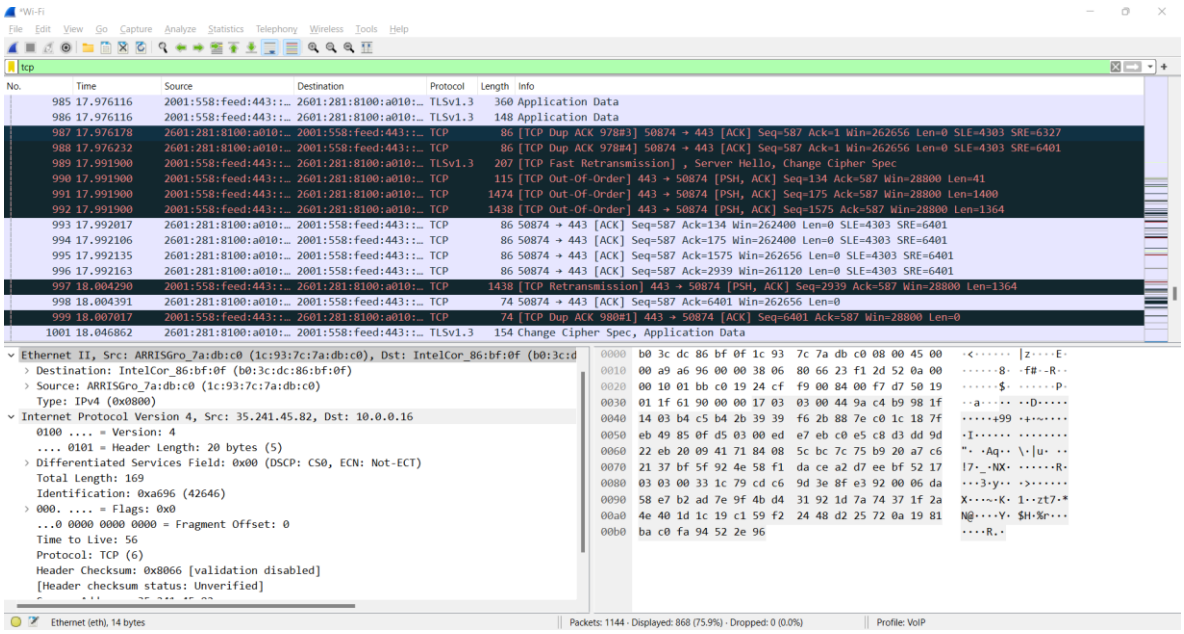
### Objective 7.1

1. Start Wireshark and select the appropriate interface to begin capturing packets.

2. Go to http://gaia.cs.umass.edu/wireshark-labs/INTRO-wireshark-file1.html in your browser and capture the web session on Wireshark.

3. How is the TCP connection established? Explain the process. What is it called? Locate it in your Wireshark capture. Paste relevant screenshots. **[5 points]**

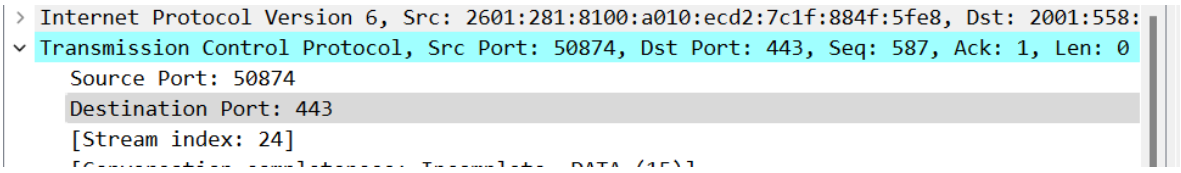
TCP Connection is established in a FULL-DUPLEX MODE and both sides SYNCHRONIZE (SYN), ACKNOWLEDGE (ACK), each other. The exchange of these flags is performed in three steps – **SYN, SYN+ACK,and ACK** . And the process is called **THREE-WAY HAND SHAKE**




4. Inspect information within the first packet of the TCP connection process.

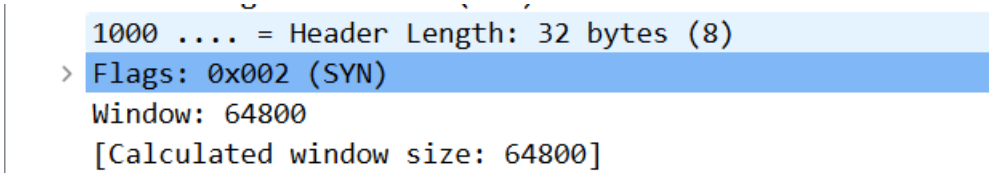
i. What is the destination port number? How would you classify it? **[1 point]**

**Dest Port :: 443. Its globally used for HTTPS services and classified as Standard Port for HTTPS (encrypted) Traffic.**



ii. Which control flag (or flags) is set? What does it imply? **[1 point]**

**Flag [SYN] is set. It implies that the packet is requesting a connection**



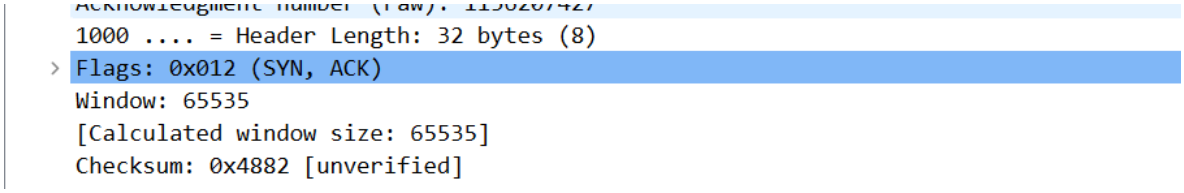

**iv.** What is the relative sequence number set to? **[1 point]**

**The relative sequence number is set to SEQ = 0**

5. Inspect the next packet in the TCP connection process.

i. Which control flag (or flags) is set? What do they imply? **[1 point]**

**Flag [SYN+ACK] is set, it implies, ACK is made for the respective request**



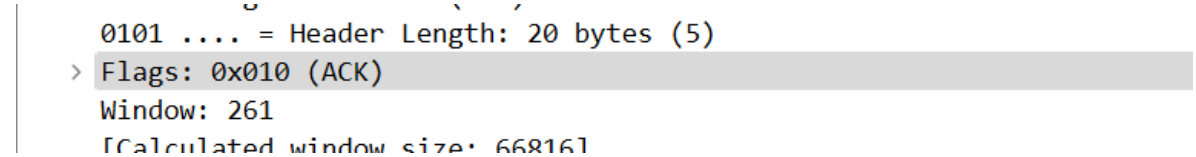

ii. What is the relative sequence number and relative acknowledgement number set to? **[1 point]**

**The relative sequence number is set to Seq=0, ACK =1**

6. Finally, inspect the third packet of the connection process.

i. Which control flag (or flags) is set? What do they imply? **[1 point]**

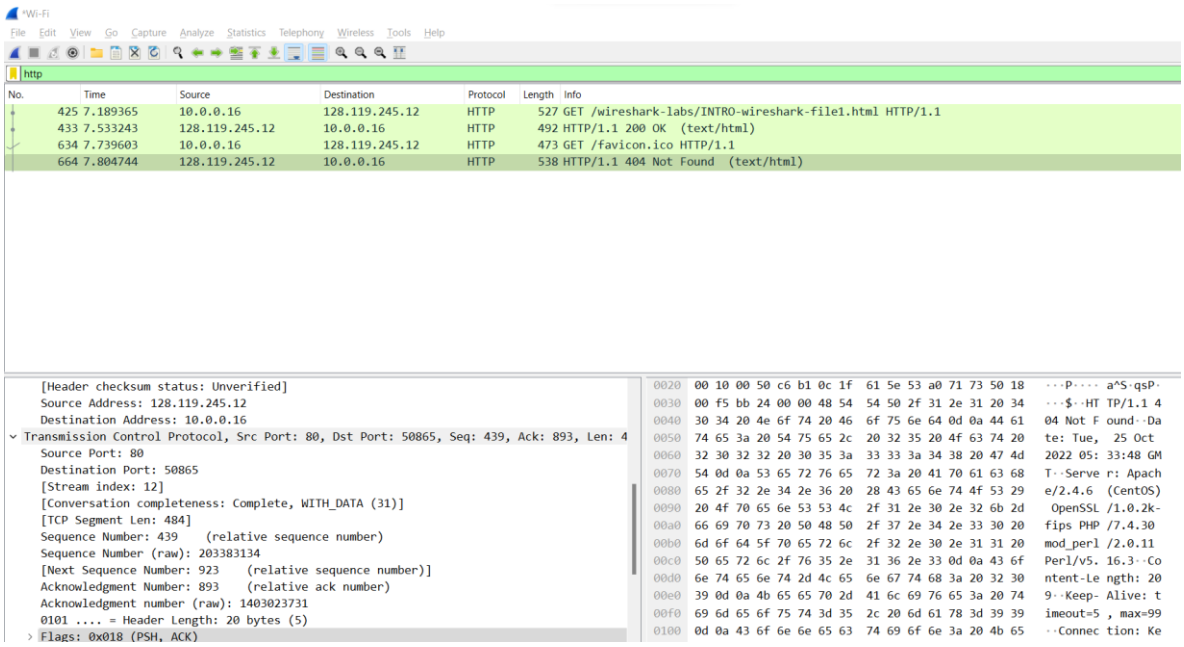
**Flag [ACK] is set, it implies, ACK and Connection established**



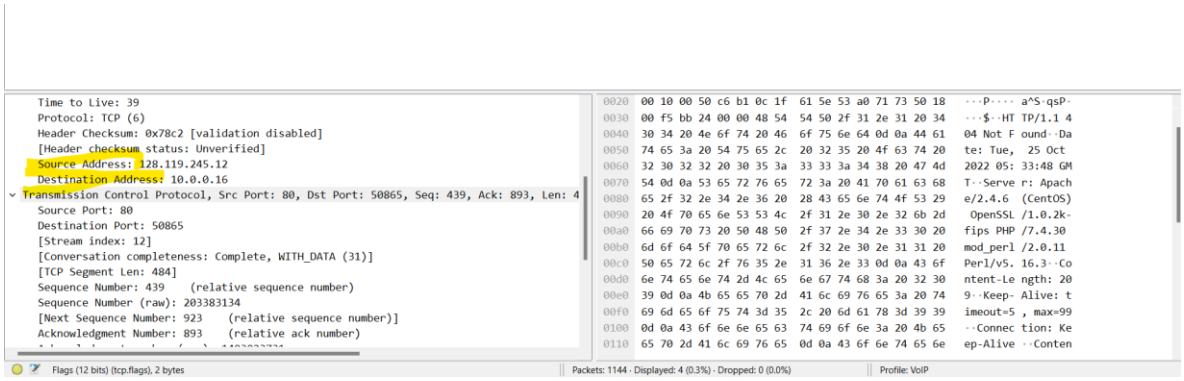

ii. What is the relative sequence number and relative acknowledgement number set to? What do they imply? **[1 point]**

**The relative sequence number is** SEQ = 1

6. In your Wireshark Capture display only the HTTP (Web) traffic. (Paste screenshot). [**5 points**]




**7.** Examine the HTTP request packet. What is the destination IP address and destination port number? Which TCP control flag (or flags) is set, and what do they mean? Paste relevant screenshots. **[5 points]**




9. Examine the HTTP packets and answer the following questions -

i. What HTTP version is running on the client? What version of HTTP is the server running? **[2 points]**

**iii.** What is the status code returned from the server to your browser? **[1 point]**

iv.

iii. When was the HTML file that you are retrieving last modified at the server? **[1 point]**

iv. How many bytes of content are being returned to your browser? **[1 point]**

10. Can you see the text displayed on the browser in your Wireshark packets as well? Why/why not? Paste relevant screenshots. [**5 points**]

## Part 8 – Parsing .pcap using Python [Extra Credit]

### Objective 8.1

1. Start a new capture in Wireshark using the capture filter of 'icmp'. Open the command prompt/terminal and execute these commands -

   ping -4 google.com

   ping wellsfargo.com

   [Use -c 4 option if pinging from MAC.]
2. Stop the capture and save the file as .pcap.
3. Write a script using Python that parses the saved .pcap file and prints out only the source and destination IPs of each packet of the file sequentially. You can use the Python library pcapfile for this purpose [https://pythonhosted.org/pypcapfile/installing.html]. [**20 points**]

## Total Score = _____/291 [+20 Extra Credit]