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.

Prefer using VM1 - UBUNTU – 22.04

Part 1

Objective 1.1 - Downloading Wireshark and Navigation Overview

1. Download and install Wireshark:

   For Ubuntu 22.04 refer - How to Install and Configure Wireshark on Ubuntu 22.04 (linuxhint.com)

   For Windows and MAC:

   - 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

Note: after installing you might need to reboot once.

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):

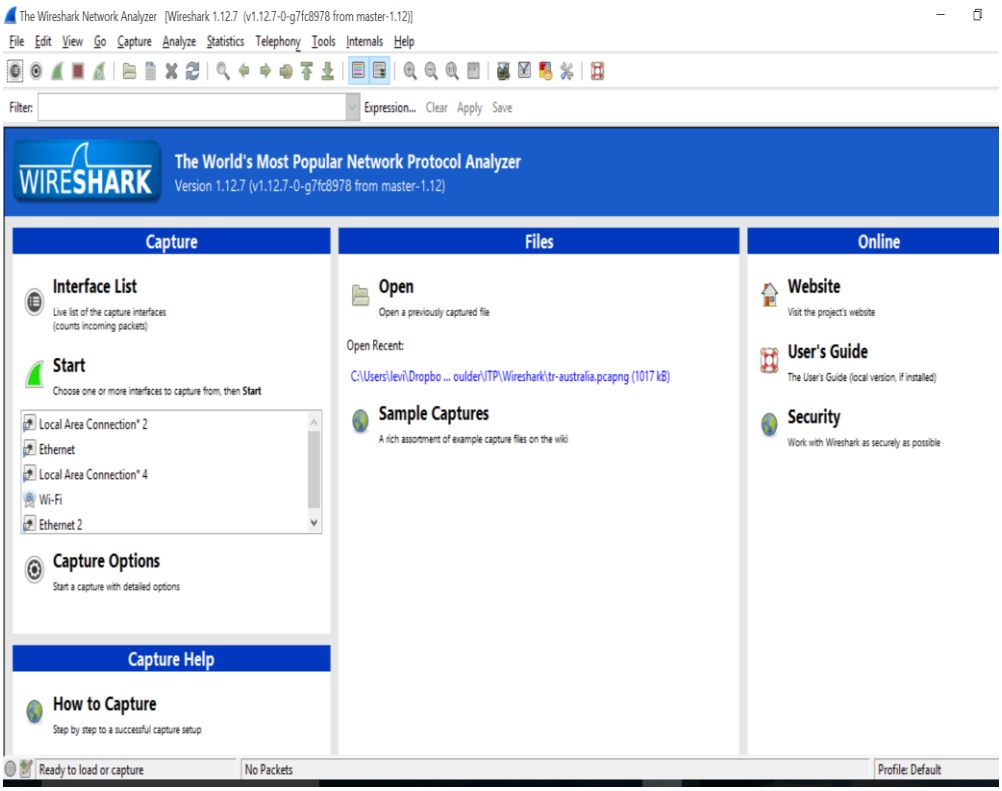


Figure 1: Initial Wireshark Screen

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.

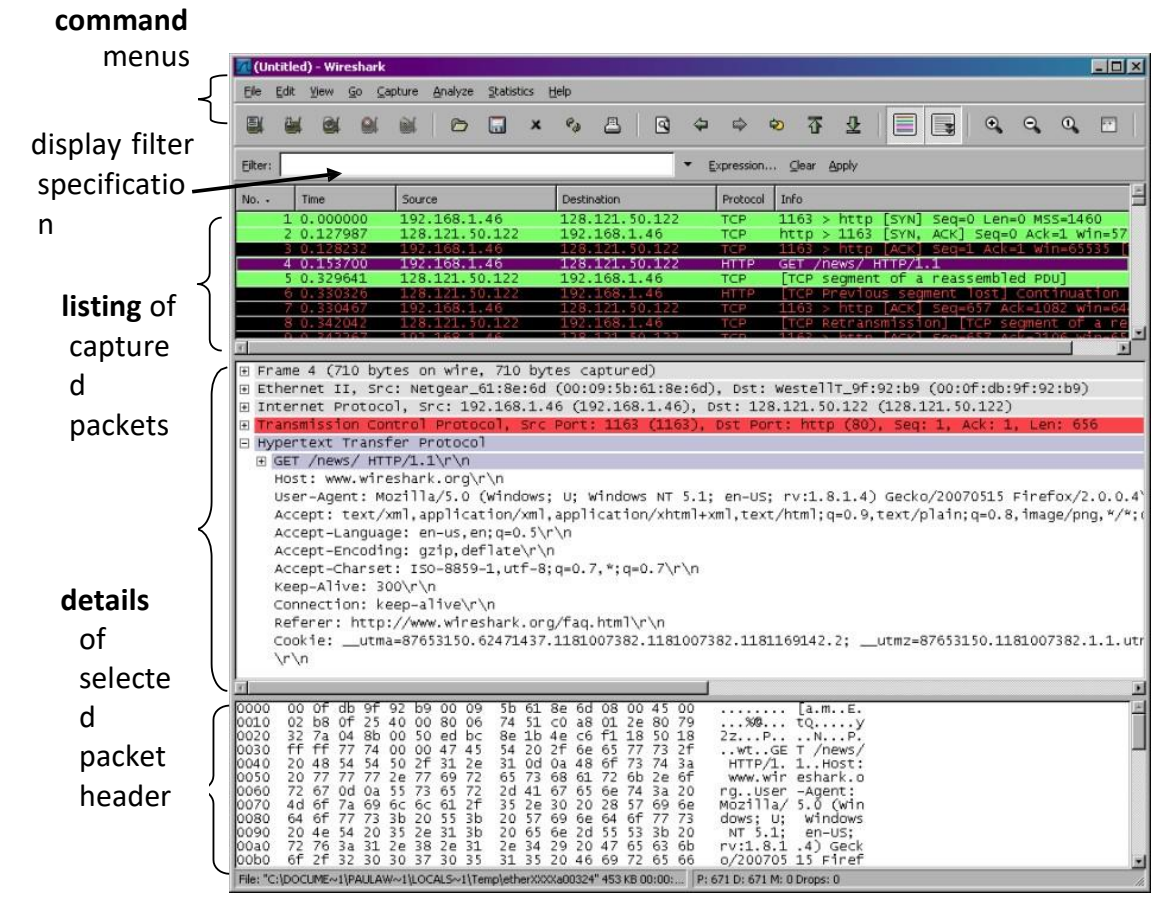


Figure 2: Wireshark Graphical User Interface, during packet capture and analysis

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

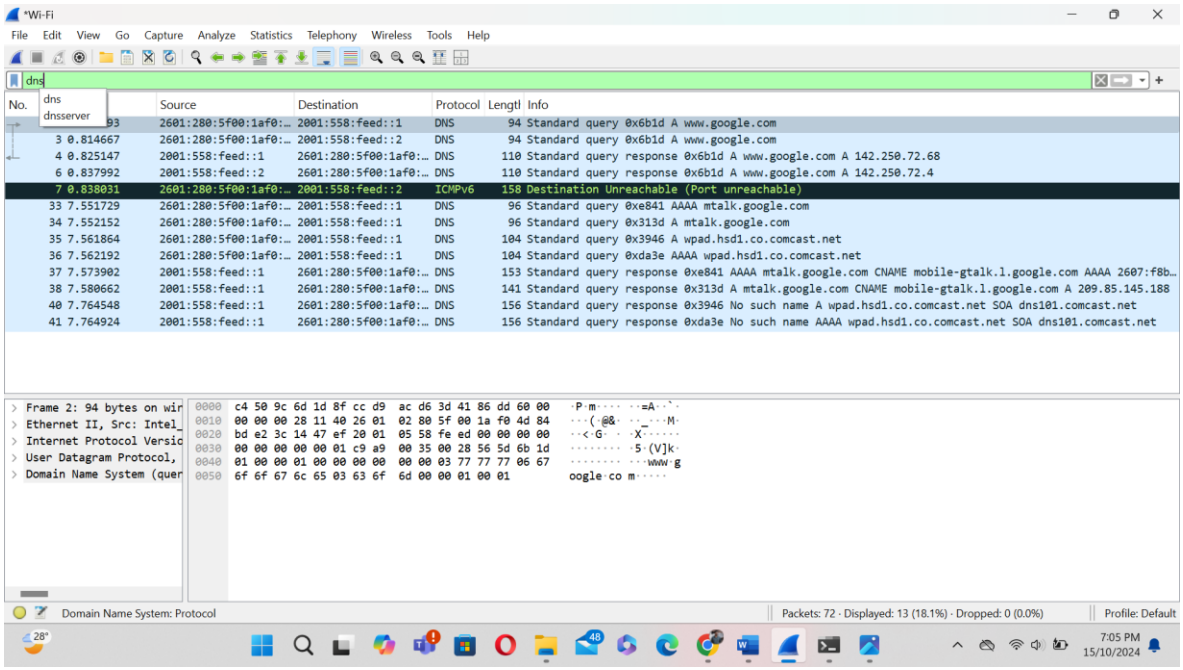
allows you to save captured packet data or open a file containing previously captured packet data and exit the Wireshark application. The Capture menu allows you to begin packet capture.

- 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**]



I had pinged www.google.com it shows a DNS reply from the server that shows the ip address 142.250.72.68.

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

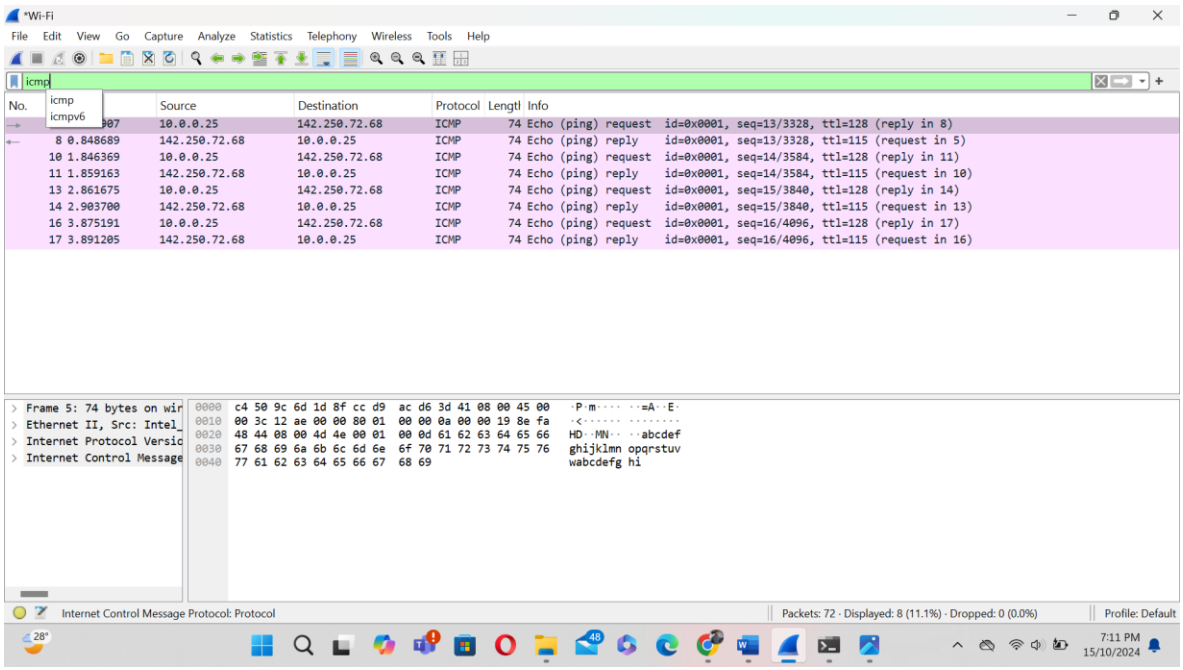
When I pinged the domain name the system resolves the domain into an IP address using DNS (Domain Name System). The DNS lookup converts the

URL into its corresponding IP address before the actual ping packets can be sent to the server.

6. Filter the capture to only display the Ping traffic
    a. Were the Pings successful?

        Yes, the ping was successful.

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



The ping will run 4 times by default and the ICMP echo requests and ICMP echo replies and since both are there the ping was successful.

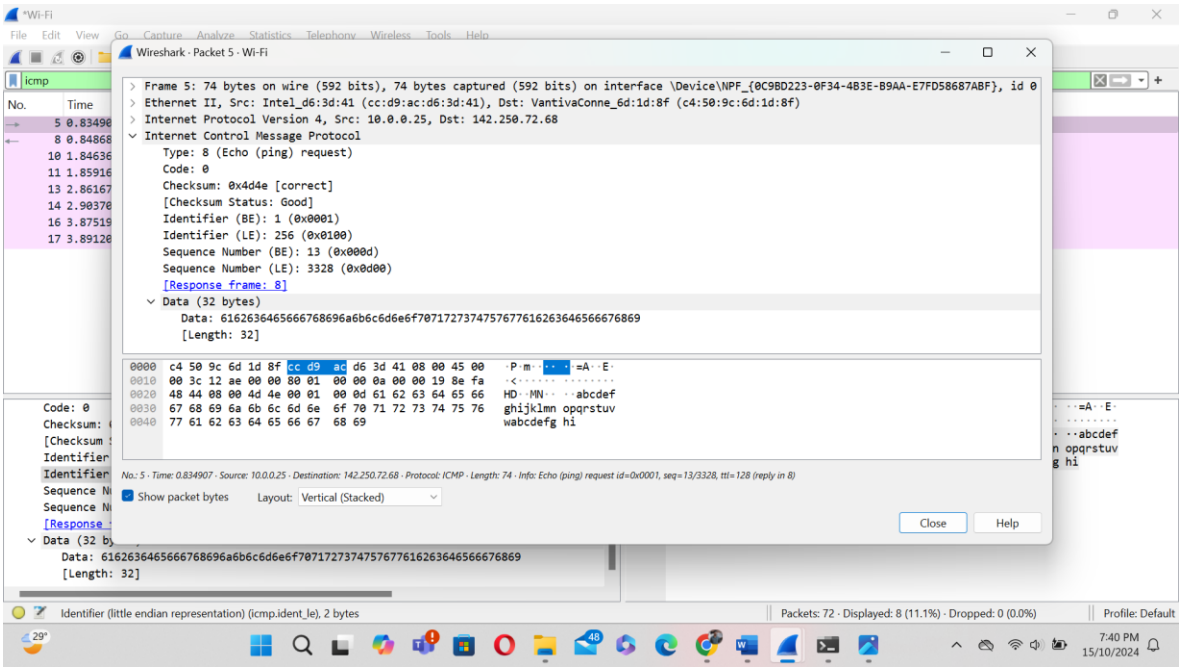
7. What is the IP address of your host? (show within Wireshark) [**5 points**]

From the above screenshot the source IP address is 10.0.0.25.

8. What is the IP address of the destination host?
    a. Show within Wireshark, and explain how this address was selected? [**10 points**]

From the above screenshot, the IP address of www.google.com which is the destination IP address is 142.250.72.68.

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 Type:8

ICMP Code:0

Other Fields are checksum:0x4d4e

Identifier (BE): 1(0x0001)

Identifier (LE): 256(0x0100)

Sequence number:13(0x000d)

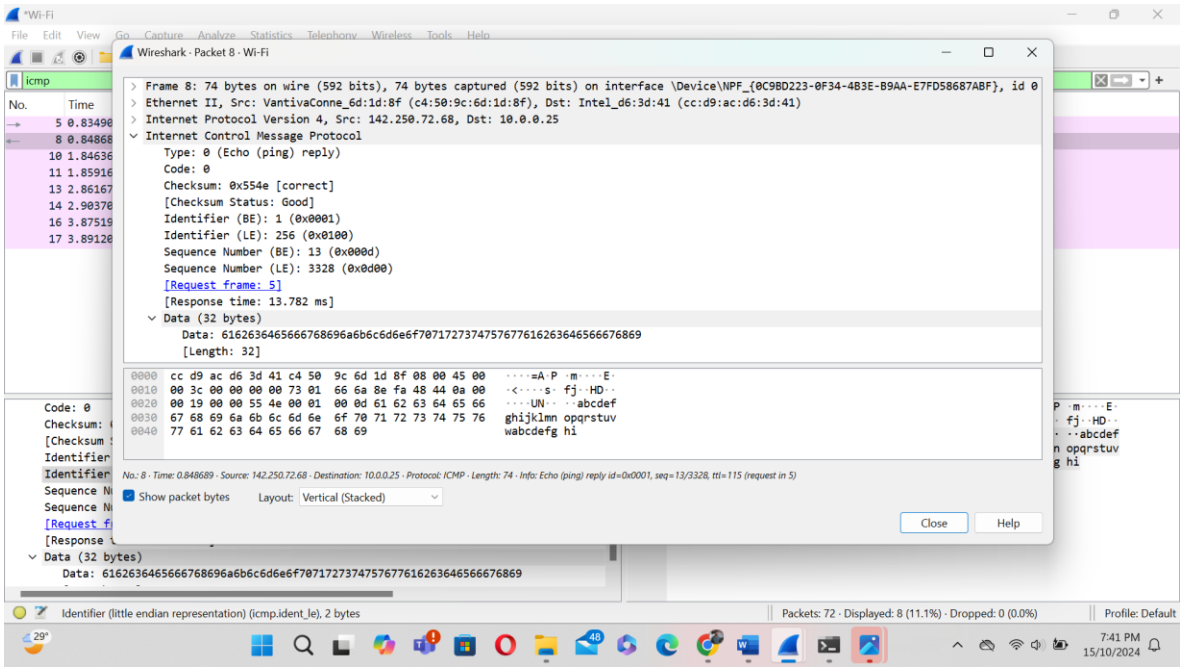
Sequence number:3328(0x0d00)

Sizes of Fields are Checksum: 2 bytes

Identifier: 2 bytes

Sequence Number: 2 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 Type:0

ICMP Code:0

Other Fields are checksum:0x554e

Identifier (BE) :1(0x0001)

Identifier (LE) :256(0x0100)

Sequence number (BE):13(0x000d)

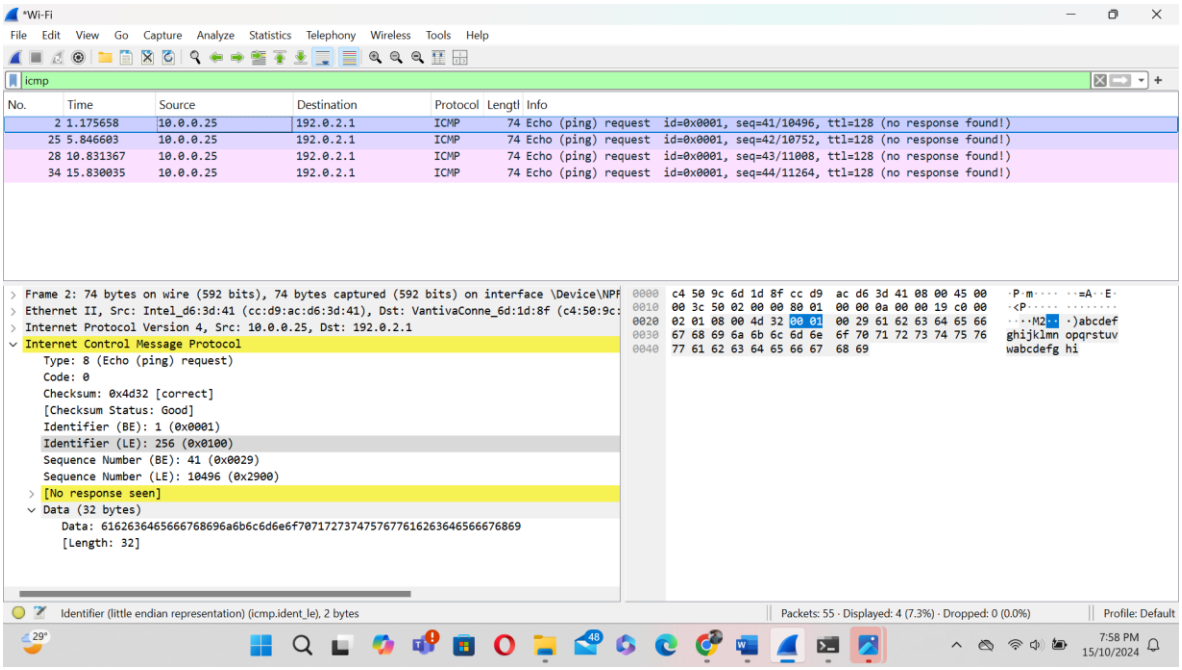
Sequence number (LE):3328(0x0d00)

Sizes of Fields are Checksum: 2 bytes

Identifier: 2 bytes

Sequence Number: 2 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**]



ICMP Type:8

ICMP Code:0

Other Fields are checksum:0x4d32

Identifier (BE) :1(0x0001)

Identifier (LE) :256(0x0100)

Sequence number (BE)41(0x0029)

Sequence number (LE):10496(0x2900)

Sizes of Fields are Checksum: 2 bytes

Identifier: 2 bytes

Sequence Number: 2 bytes.

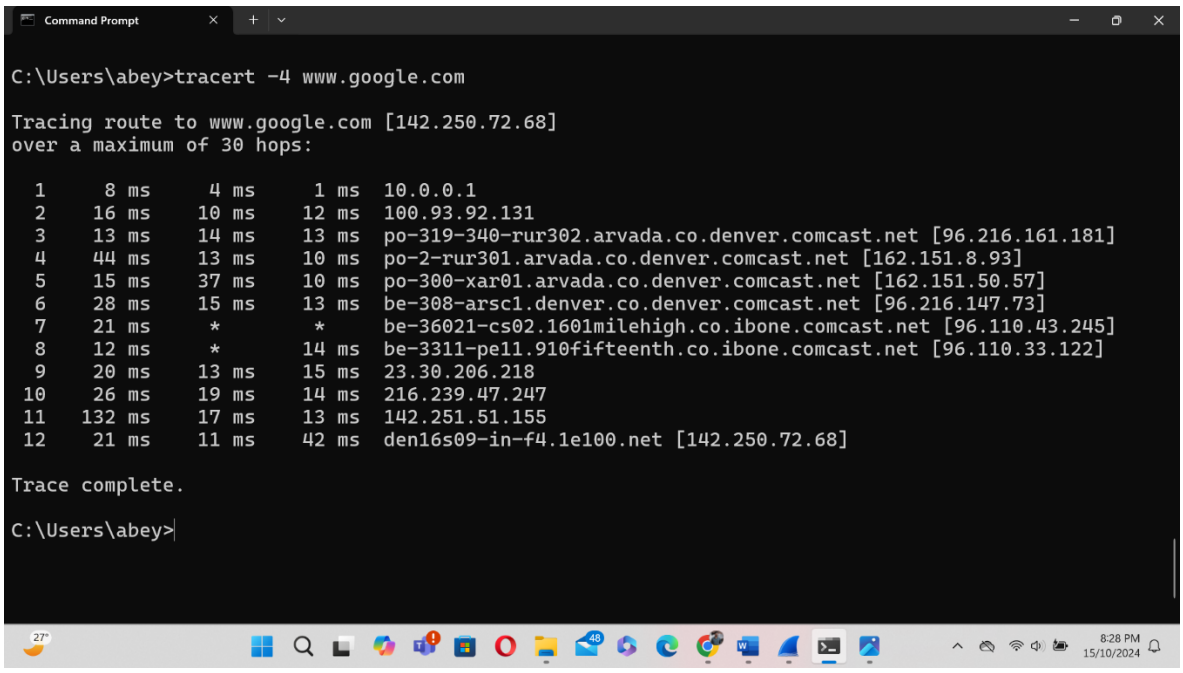
12. Do you see both ICMP Echo Request and Echo Reply messages? [**5 points**]

In this scenario, we can see only see ICMP Echo request messages in Wireshark since there was a request time out message during ping and therefore there was no echo reply (server did not reply).

Objective 2.2 – ICMP and Traceroute

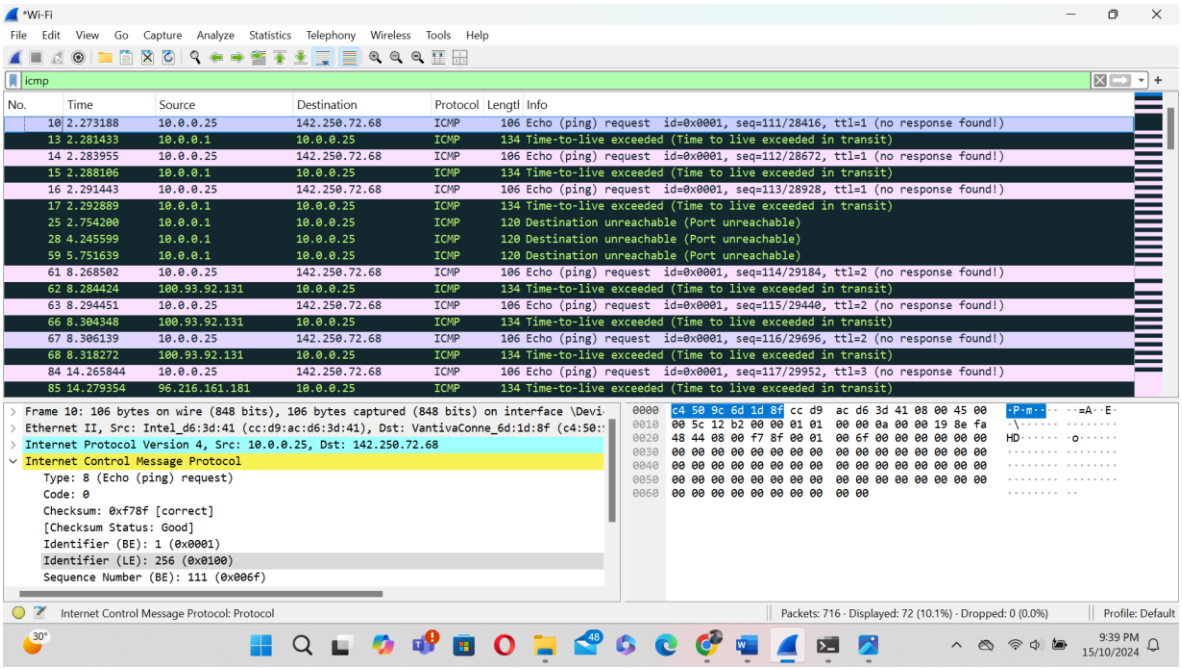
1. Open command prompt/terminal (depending on the operating system)
2. Start Wireshark and begin capture

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 last line says the trace complete which confirms that the packets successfully reached their target. The traceroute reached the final destination 142.250.72.68.

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**]

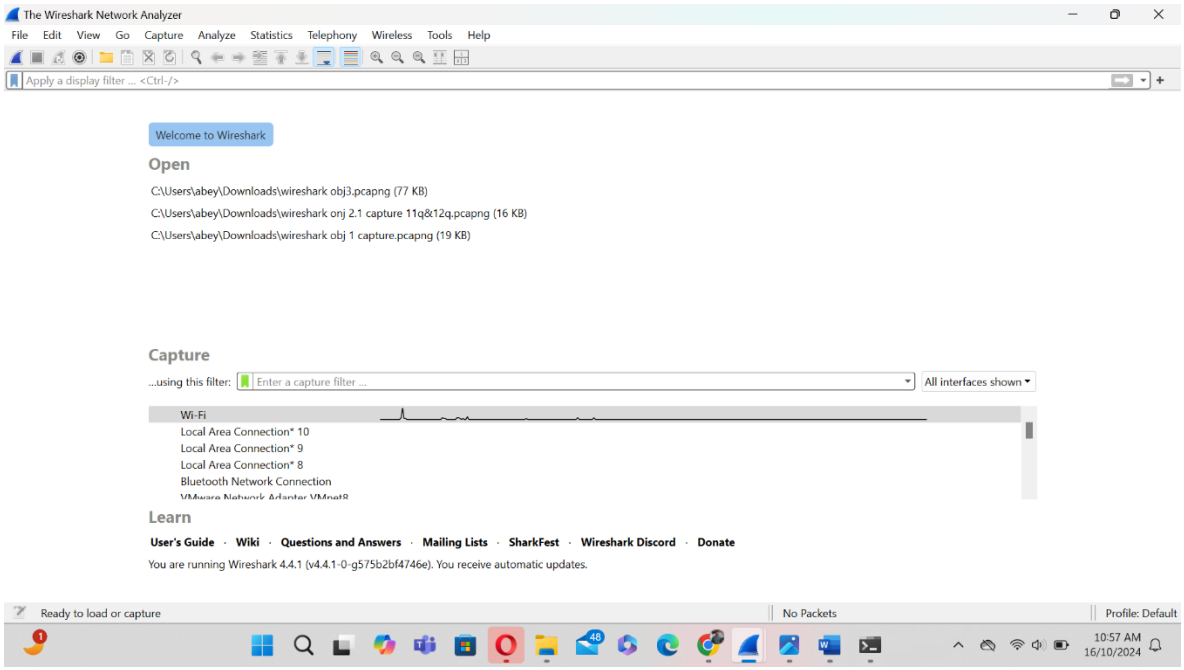


In the previous objective there was only ping request, only ICMP echo request (type 8) and ICMP Echo reply messages are exchanged. In Traceroute you can see ICMP echo requests but the ICMP Time to live (TTL) exceeded responses from routers. It is essential for mapping the path to the target host. Traceroute sends out packets with increasing TTL values starting with 1. Each router along the path decreases the TTL by 1. When the TTL reaches zero, the router sends back an ICMP time exceeded message. This takes place in the capture of every hop until the destination is reached. Ping is a simple ICMP tool that checks the reachability of a host by sending a echo request and receiving an echo reply whereas traceroute not only checks reachability but also maps the path by incrementally adjusting the TTL and provides the IP address and response time of each router.

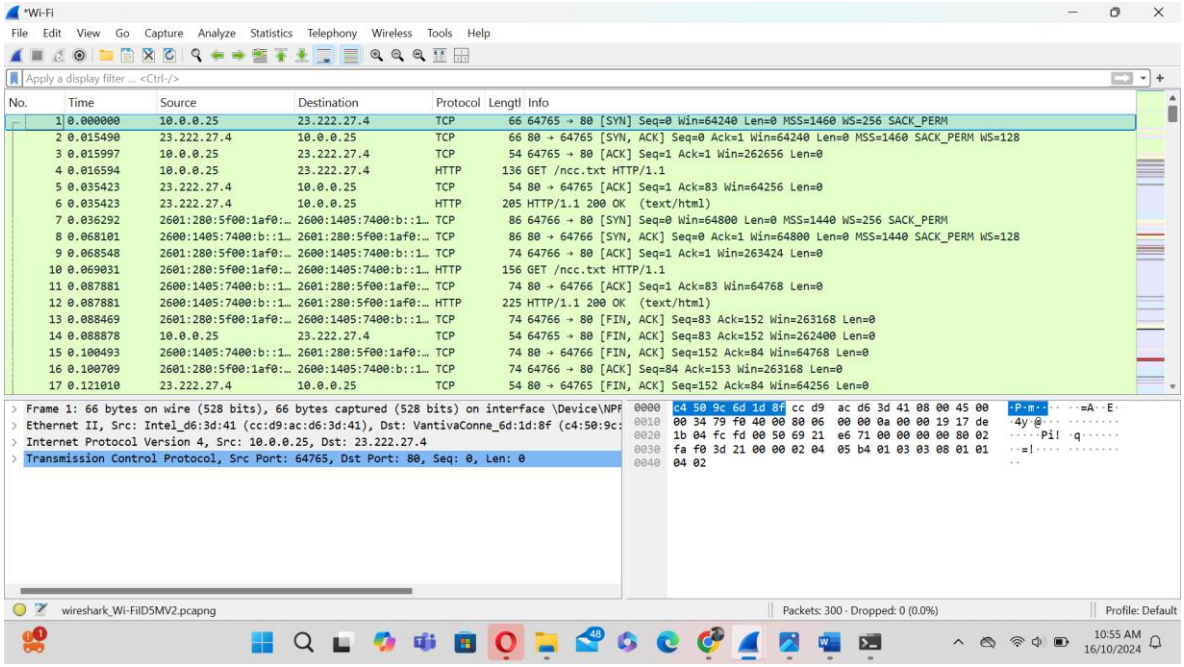
Part 3 – Wireshark NICs and IPv4/IPv6 addresses

Objective 3.1

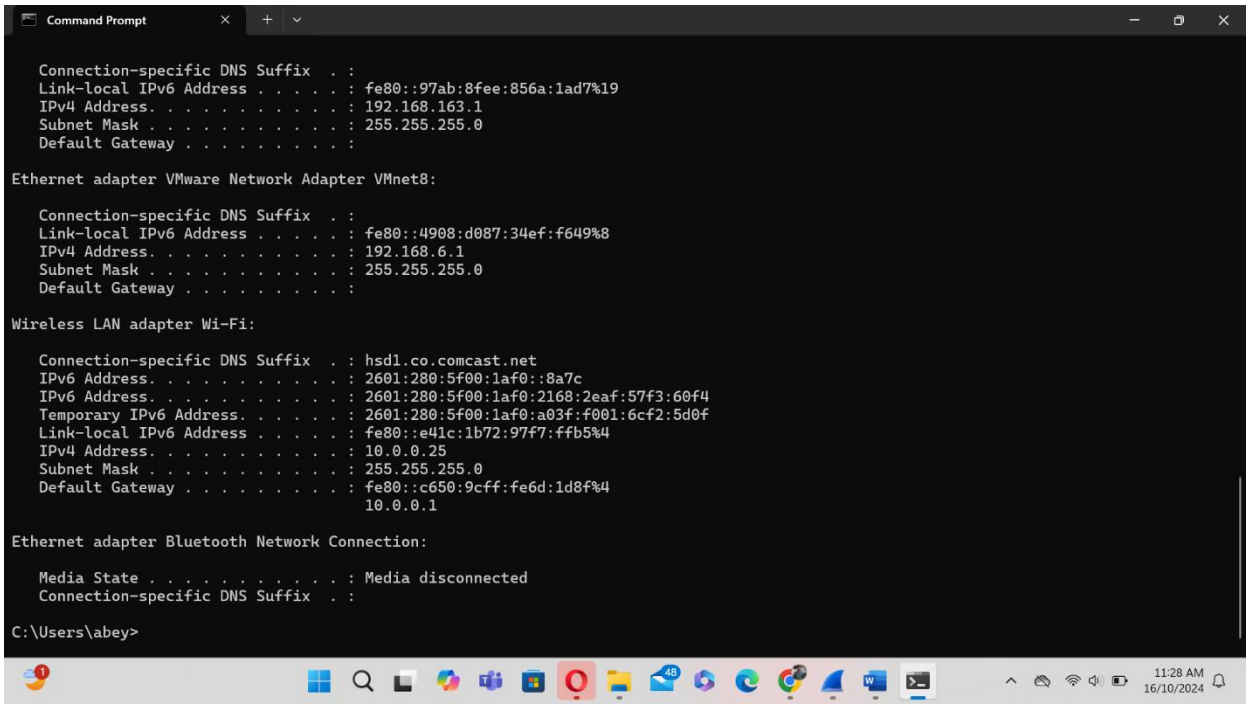
1. Provide a screenshot of the NICs that Wireshark has to choose from on the PC. **[10 points]**



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



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]**

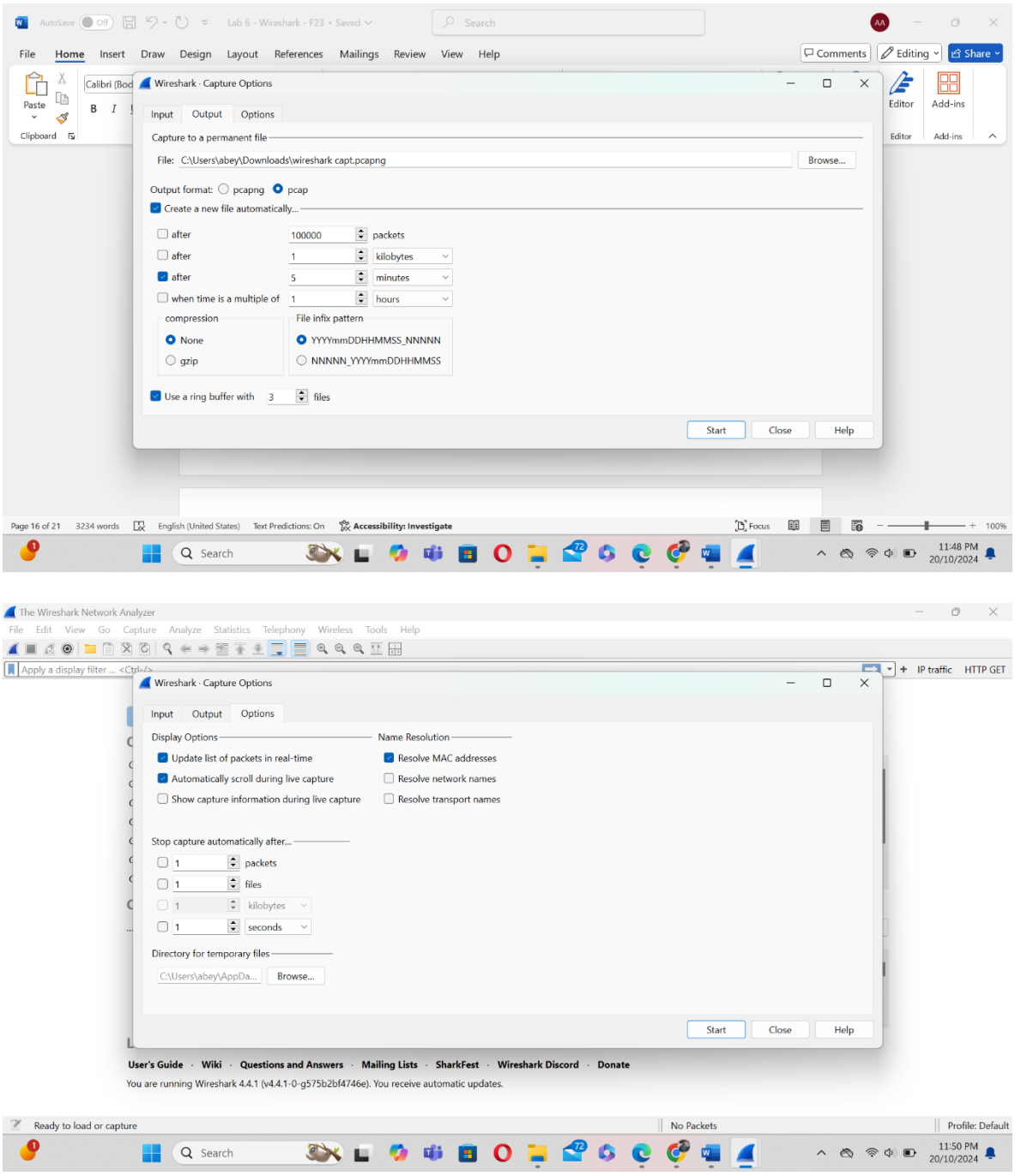


In my windows I ran the command ipconfig will display the IP addresses for each NIC. IPv4 address is 10.0.0.25 and IPv6 address is 2601

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]**



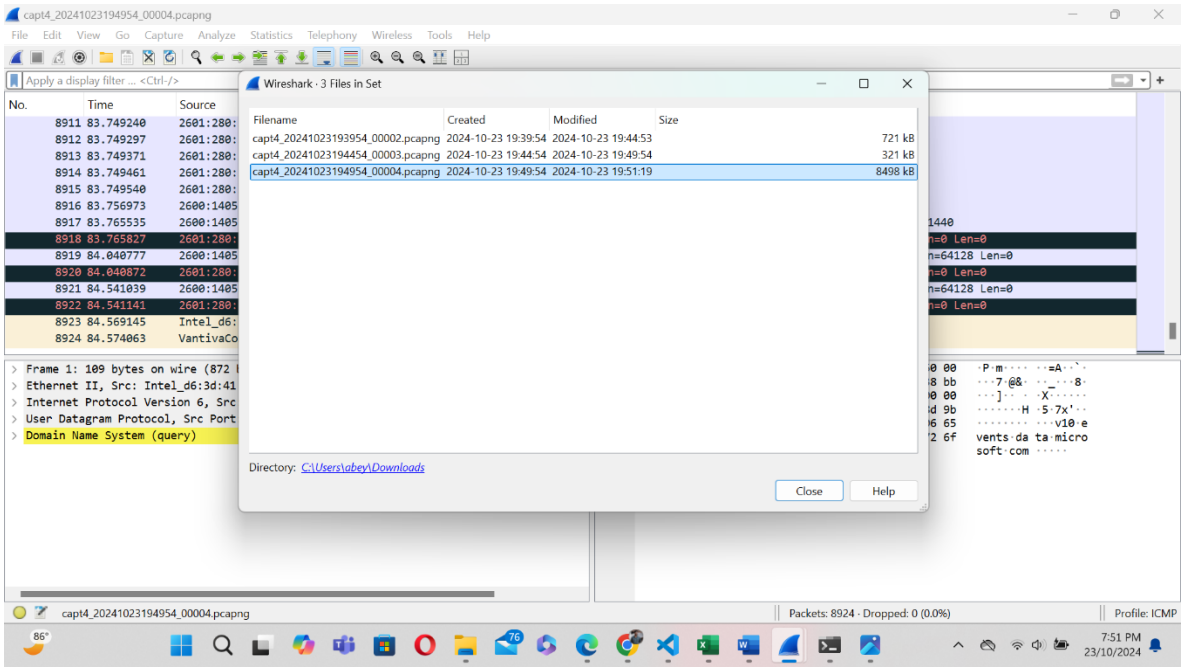
2. Browse ten different websites, during this 15-minute continual capture time window.
3. What are THREE reasons why you would want to create multiple files? **[15 points]**

   It's easier to navigate multiple smaller files and segmentation of data especially if we want to check the logs at a specific period.

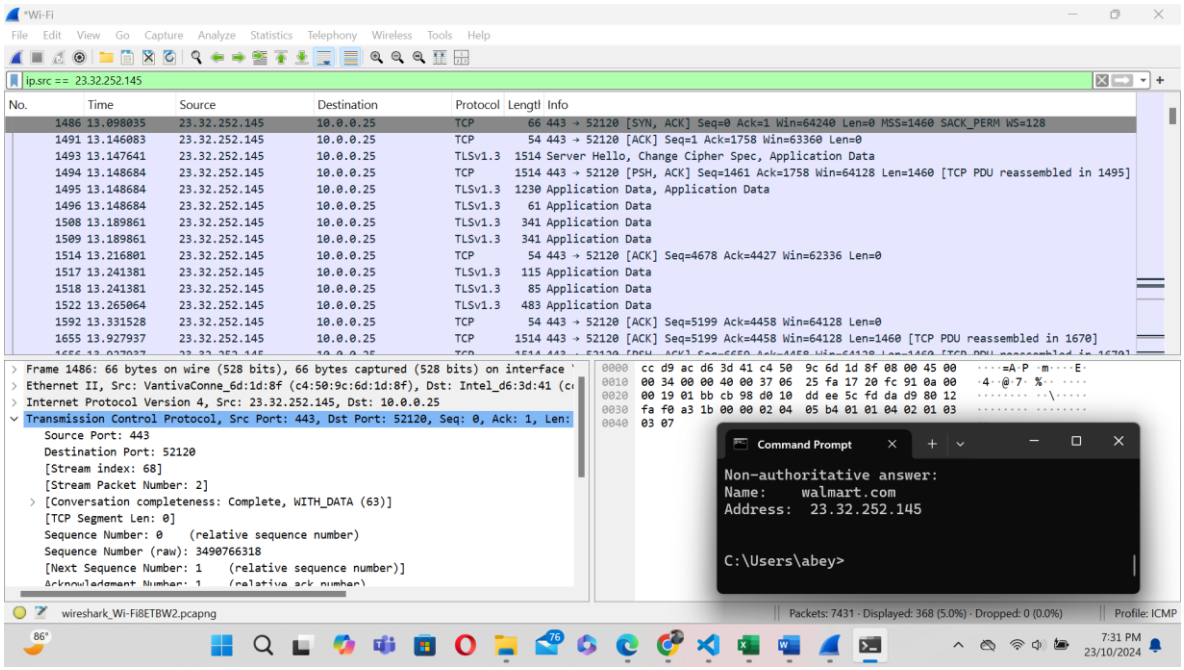
Splitting larger files to smaller files prevents the creation of large files that are difficult to analyze.

Capturing logs in one file for a long period of time can cause issues if it is a large file, it has the potential to crash and lose data. Having multiple files can ensure the data is preserved.

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]**

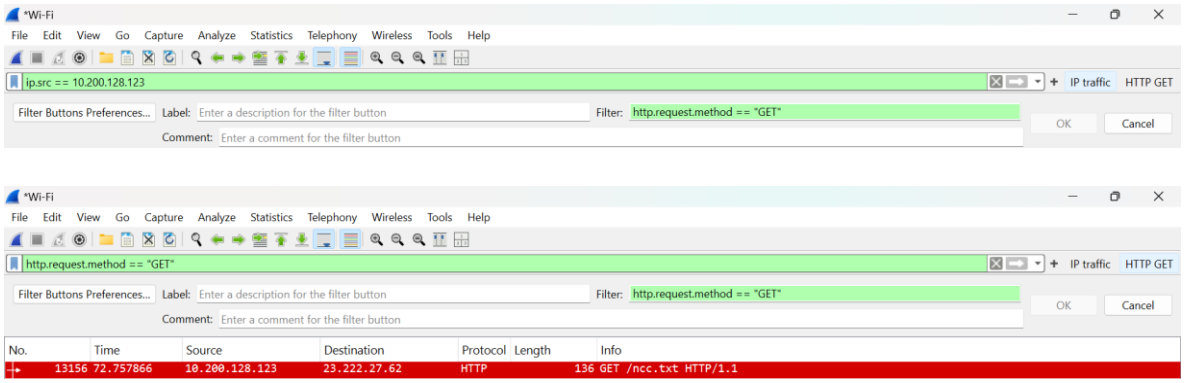


5. How do you see which websites you browsed during the capture? (Hint: Statistics > HTTP) **[5 points]**



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]**



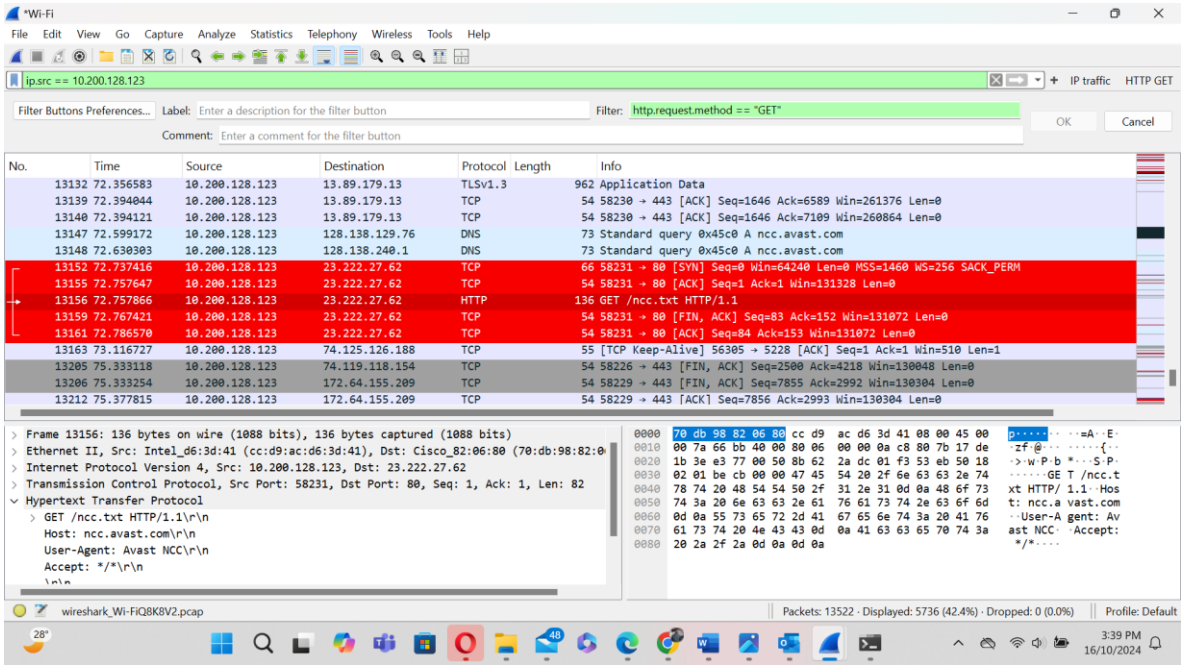
Objective 4.3

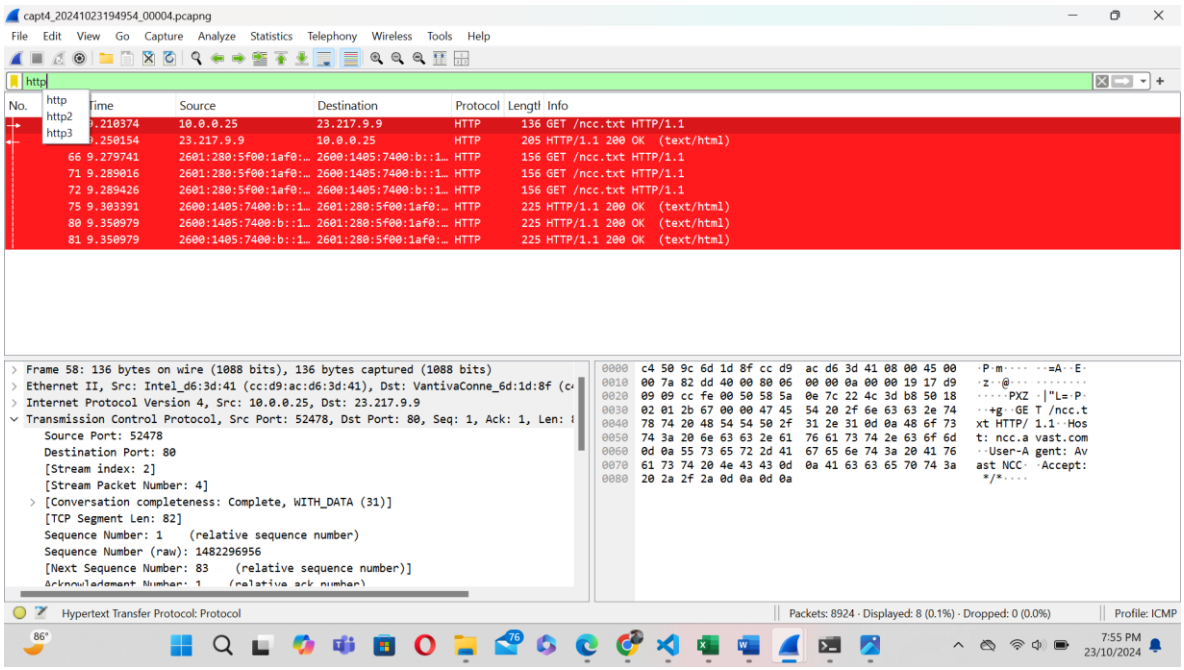
1. Create a coloring rule for HTTP traffic. **[5 points]**

   Navigate to View-> Coloring Rules from the top menu.

Click on HTTP and change the background color to red and white lettering and click Ok to save the rule.

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]**



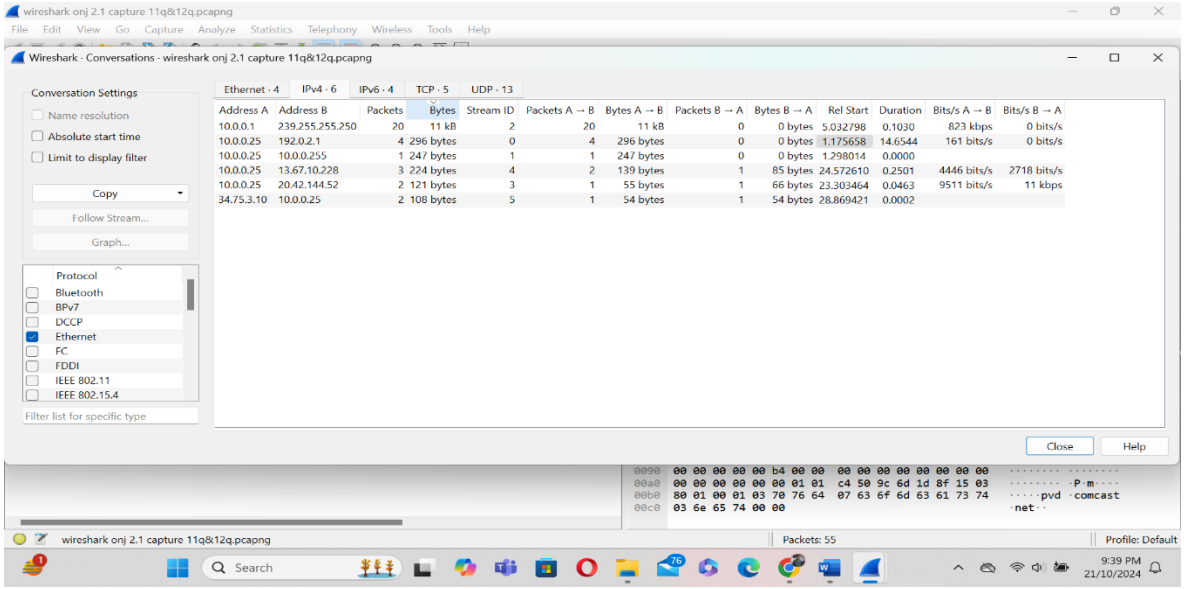


Part 5 - Top Talkers, Profiles, and Graphs

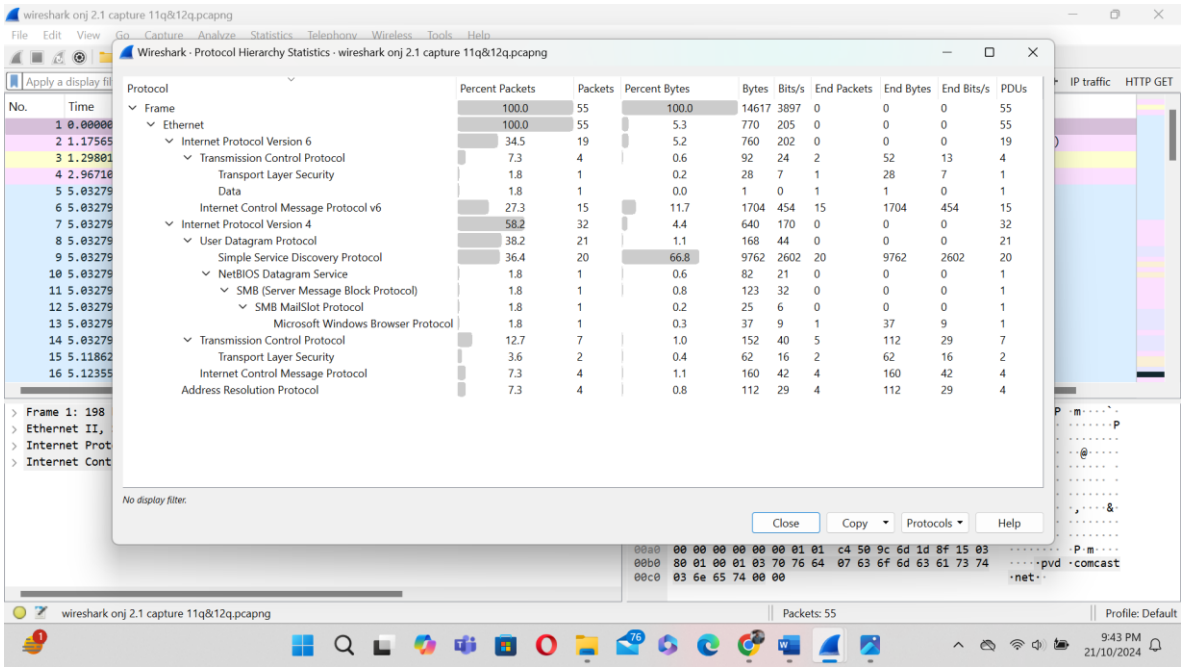
Objective 5.1

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

   Top Talkers in the network were determined by going to statistics and click on conversations.



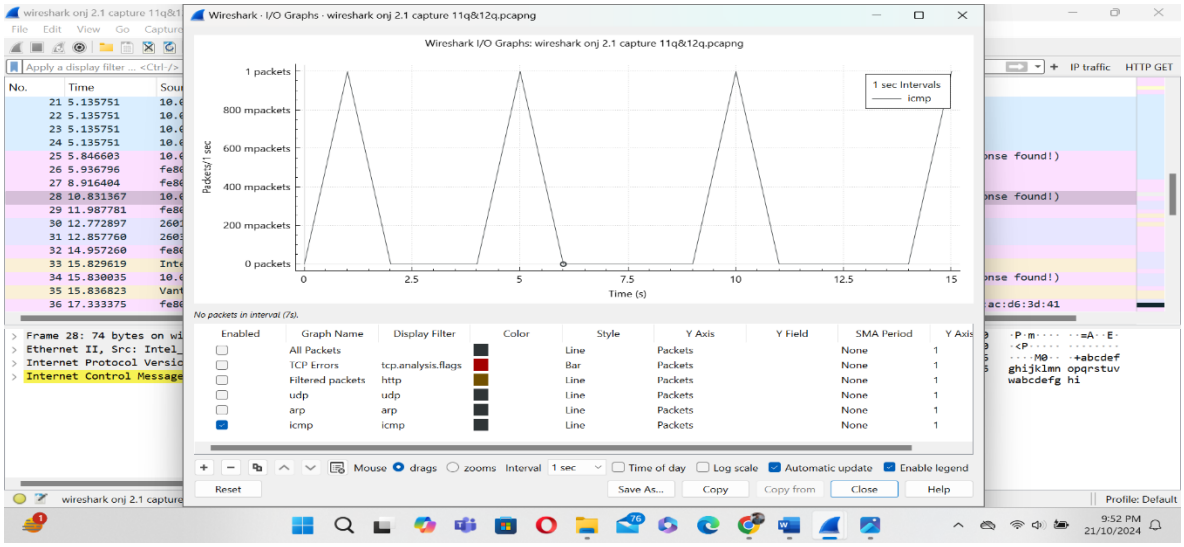
2. What are two ways you can determine what device/IP address is transmitting the most traffic on the network? Provide a screenshot of one of those ways. (Hint: Protocol Hierarchy; Conversations; Endpoints) **[10 points]**

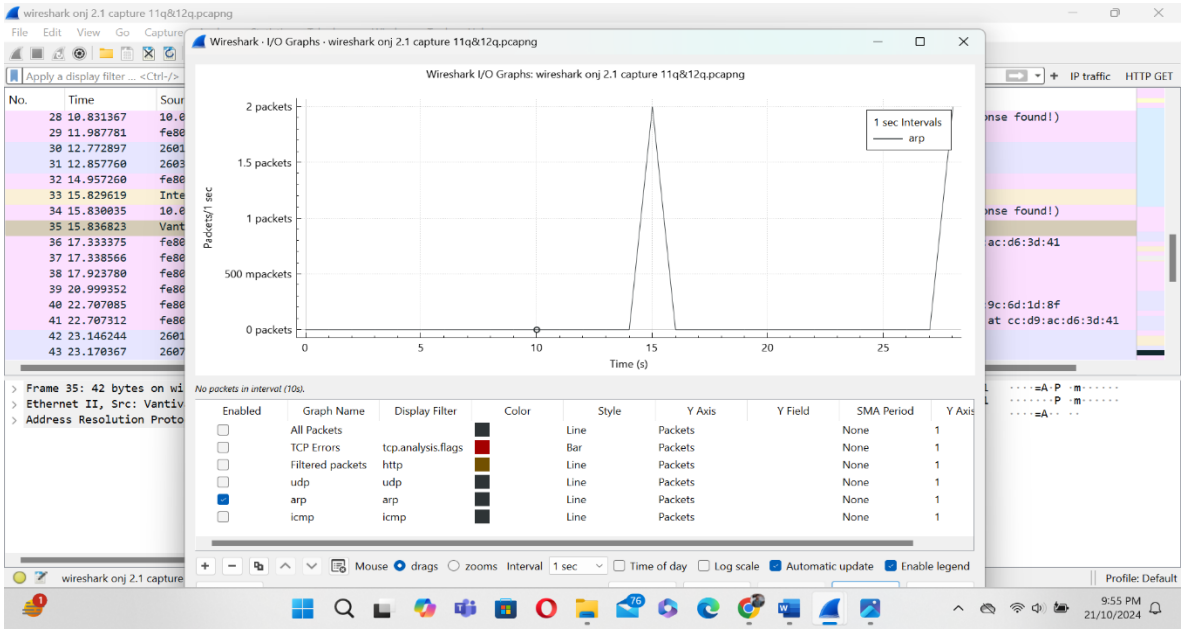


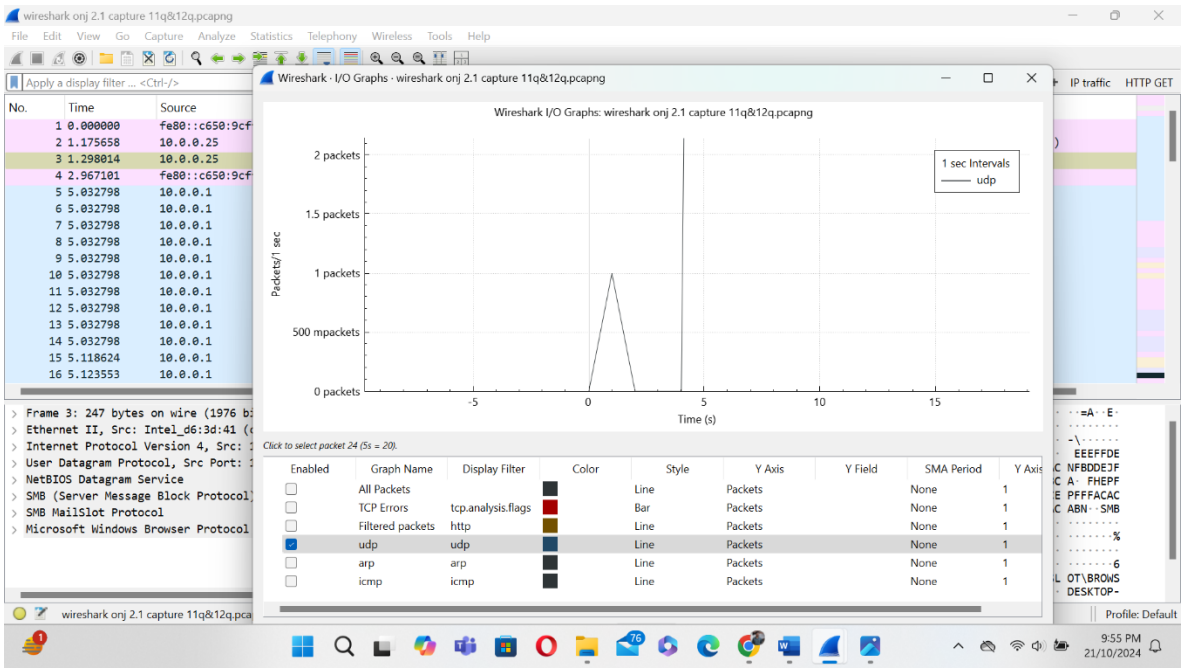
Two ways to determine the top talkers are through conversations or through protocol hierarchy. The above screenshot is done using protocol hierarchy.

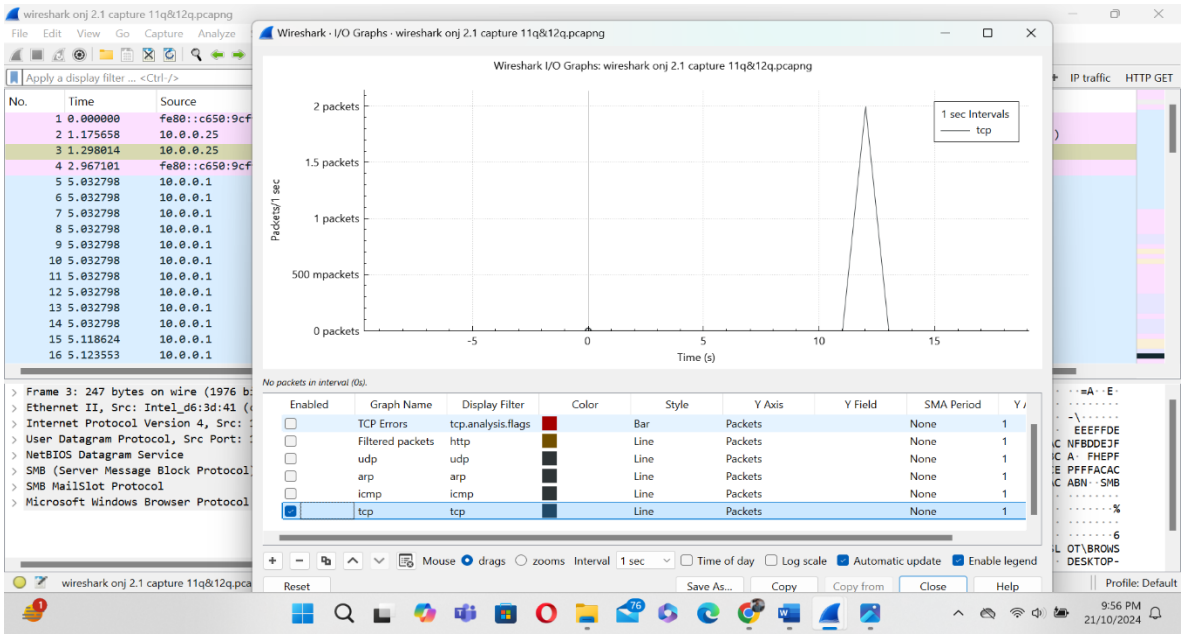
Objective 5.2

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



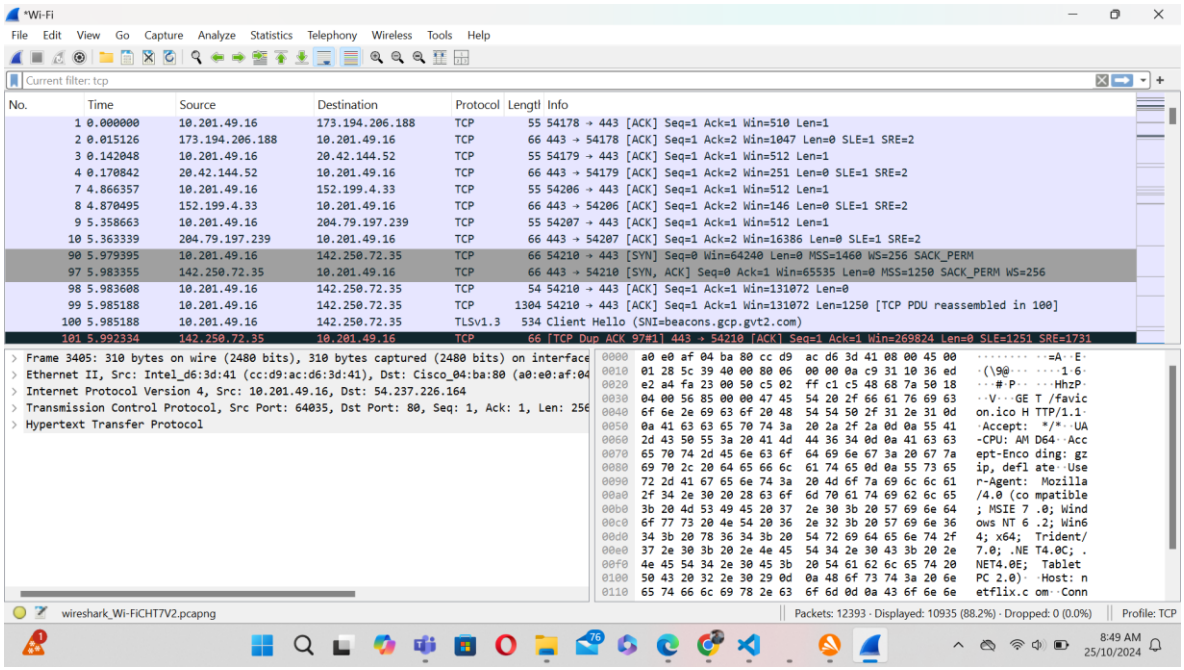

wireshark onj 2.1 capture 11q&12q.pcapng
Wireshark · I/O Graphs · wireshark onj 2.1 capture 11q&12q.pcapng
Wireshark I/O Graphs: wireshark onj 2.1 capture 11q&12q.pcapng
2 packets
1.5 packets
1 packets
500 mpackets
0 packets
Packets/1 sec
Time (s)
1 sec Intervals
arp
No packets in interval (10s).
Enabled
Graph Name
Display Filter
Color
Style
Y Axis
Y Field
SMA Period
All Packets
TCP Errors
tcp.analysis.flags
Filtered packets
http
udp
arp
icmp
Mouse
drags
zooms
Interval
1 sec
Time of day
Log scale
Automatic update
Enable legend
Profile: Default
9:55 PM
21/10/2024

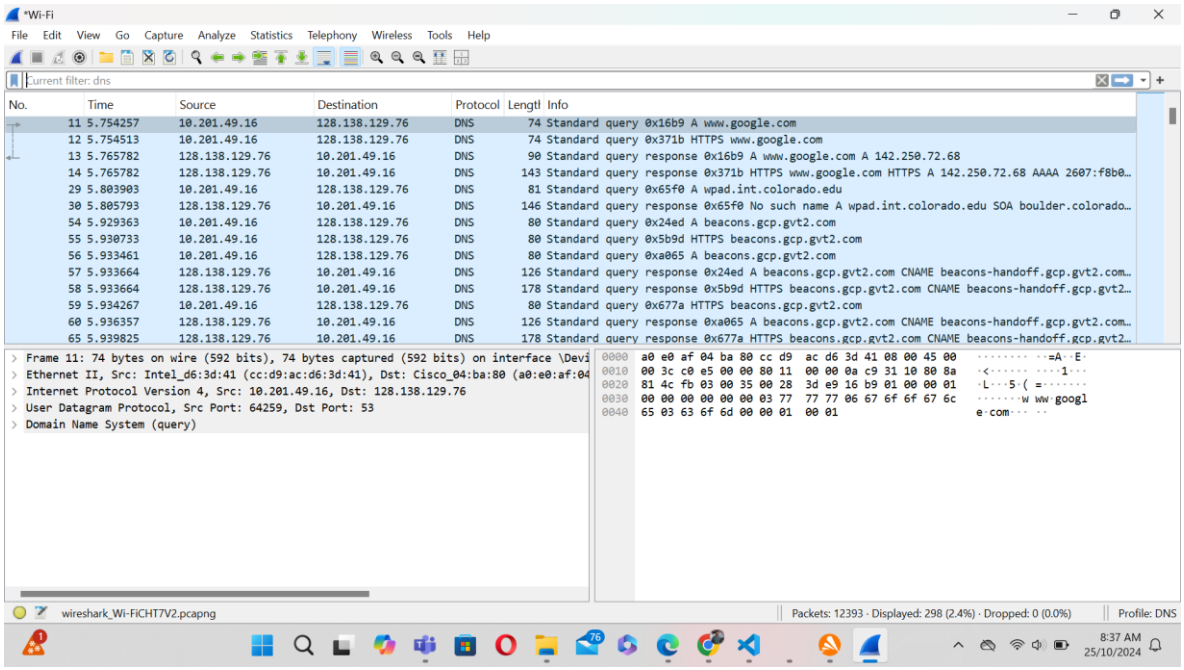

wireshark onj 2.1 capture 11q&12q.pcapng
Wireshark · I/O Graphs · wireshark onj 2.1 capture 11q&12q.pcapng
Wireshark I/O Graphs: wireshark onj 2.1 capture 11q&12q.pcapng
2 packets
1.5 packets
1 packets
500 mpackets
0 packets
Packets/1 sec
Time (s)
1 sec Intervals
udp
Click to select packet 24 (5s = 20).
Enabled
Graph Name
Display Filter
Color
Style
Y Axis
Y Field
SMA Period
All Packets
TCP Errors
tcp.analysis.flags
Filtered packets
http
udp
arp
icmp
Profile: Default
9:55 PM
21/10/2024

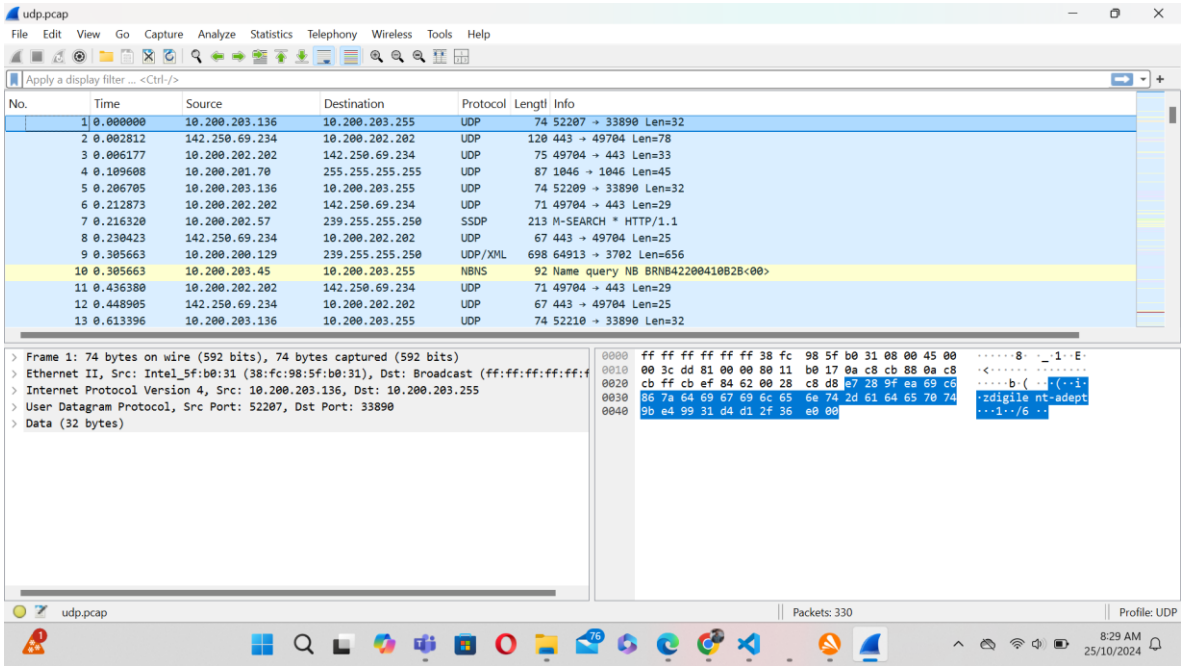


Objective 5.3

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]**

   TCP Profile: Useful for analyzing transmission control protocol traffic, including connection establishment, termination and packet retransmission. The profile is ideal for troubleshooting issues related to reliable data transmission.

UDP Profile: Since UDP is connectionless and suitable for real-time applications such as monitoring packet loss in VoIP calls or detect incorrect configuration in services like DHCP.

DNS Profile: Specifically for analyzing DNS traffic, which includes queries, responses. The profile is used for diagnosing DNS resolution issues, such as misconfigured DNS servers, slow responses.

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:

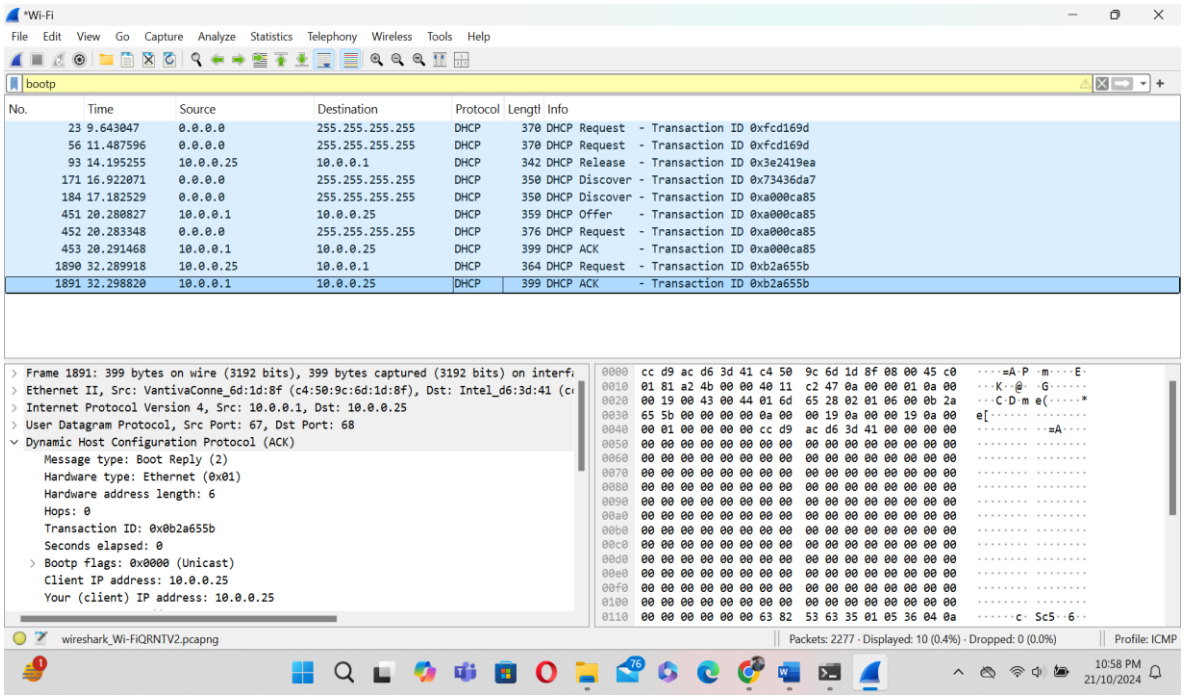
    i. DHCP server address [**2 points**]

    DHCP server address is 10.0.0.25.

    ii. The IP address your machine was offered and accepted [**2 points**]

    The IP address the machine was offered and accepted is 10.0.0.25. Since I'm running the DHCP server in the same machine as the DHCP client it is giving the same IP address.

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



DHCP does DORA process.

Discover: The client sends a DHCP Discover packet to find available DHCP servers.

Offer: The DHCP server responds with an offer packet, offering an IP address to the client.

Request: The client responds with a Request packet, requesting the IP address from the DHCP server.

Acknowledgment: The DHCP server sends an ACK packet to confirm that the IP address has been leased to the client.

Part 7 – Web traffic (HTTP) and TCP Connection

Objective 7.1

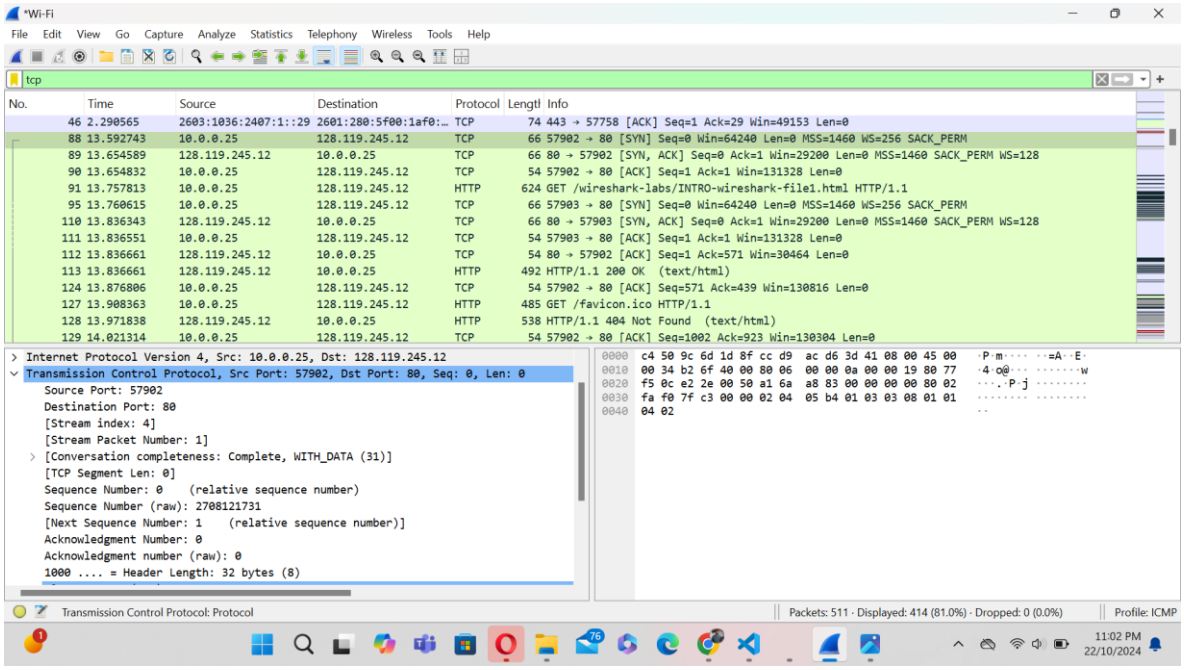
1. Start Wireshark and select the appropriate interface to begin capturing packets.
2. Go to http://gaia.cs.umass.edu/wireshark-labs/INTRO-wireshark-file1.html in your browser and capture the web session on Wireshark.
3. How is the TCP connection established? Explain the process. What is it called? Locate it in your Wireshark capture. Paste relevant screenshots. **[5 points]**

The TCP connection establishment process is known as three-way handshake. It provides reliable connection between the client and server. The steps are :

SYN: The client sends a SYN (synchronize) packet to the server, indicating the desire to open a connection and specifying the initial sequence number (Seq = X).

SYN-ACK: The server responds with a SYN-ACK (synchronize-acknowledge) packet, acknowledging its own sequence number(ISN) and acknowledgement number (ISN+1).

ACK: The clients send an ACK packet to the server, confirming the connection (ACK= ISN + 1)



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]**

The destination port number is 80, which is the default port for HTTP. It is classified as well-known port which permanent and assigned for a application protocol.

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

The SYN flag is set in the first packet, it implies that this is packet is part of the initial connection request, starting the TCP three-way handshake.

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

The relative sequence number in the first packet is 0, which marks the start of the sequence for this TCP connection.

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

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

The control flags set in this packet are SYN and ACK, these flags together imply that the server is responding to the client's SYN request, providing its own sequence number and acknowledging the client with Sequence number +1.

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

The relative sequence number in this packet is 0. This is the server's Initial Sequence Number (ISN). The relative acknowledgment number is 1. This means that the server is acknowledging the client ISN by adding 1 which was 0 initially and expects the client to have a sequence number 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]**

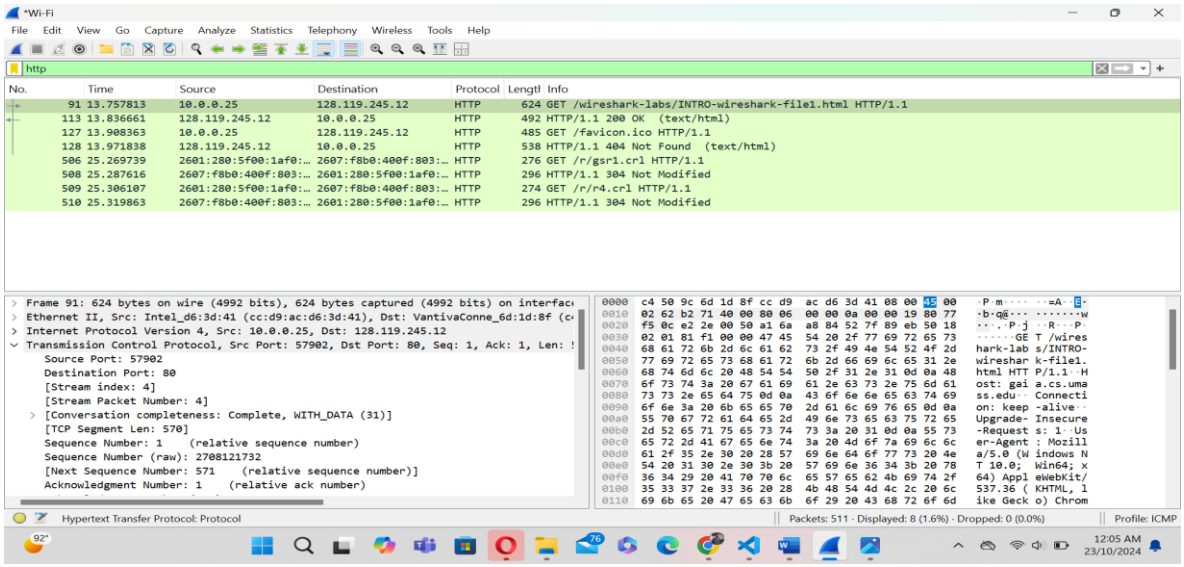
The ACK flag is set, it implies that client is acknowledging the servers SYN-ACK packet. This means that the client has received the servers sequence number and acknowledges it therefore completing the three-way handshake and the connection is established.

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

The relative sequence number in this packet is 1. This indicates that the client has incremented its sequence number by 1 after the initial SYN packet and is now ready to start sending data.

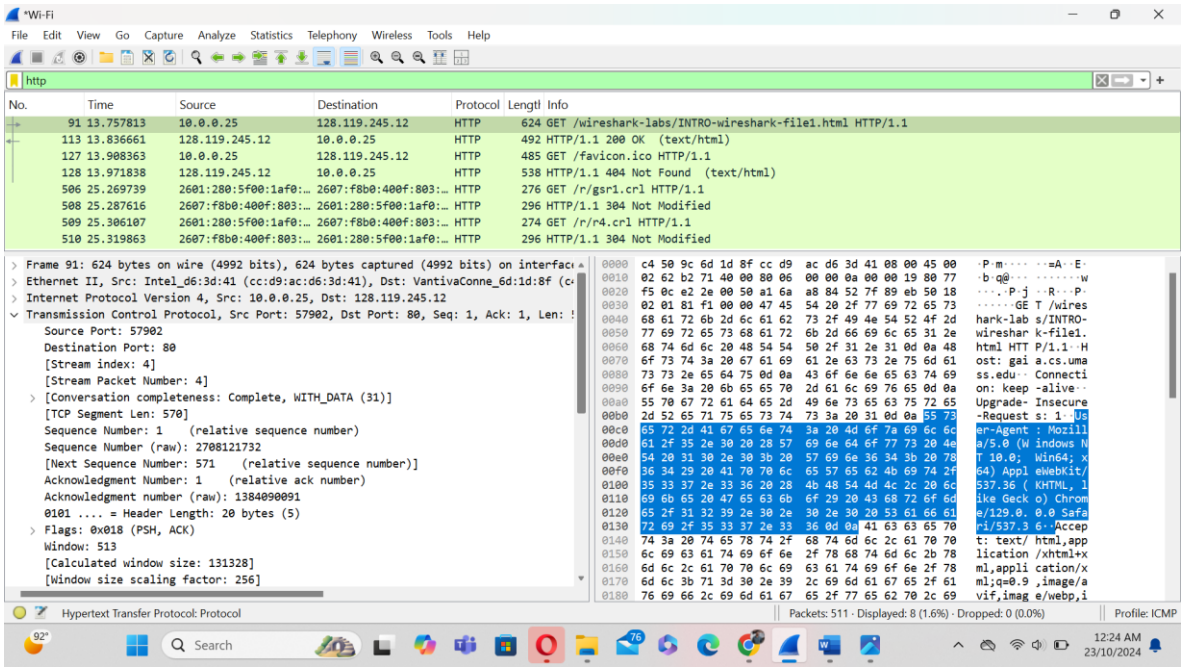
Relative Acknowledgement Number is 1. This implies that the client has received the servers initial sequence number and is acknowledging it by sending Ack =1.

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



8. 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]**

The Destination IP address is 128.119.245.12 and the destination port number is 80. ACK flag is set, it implies that client is acknowledging the server's previous message and is ready for communication. After acknowledging the handshake with the ACK flag, the client is now making an HTTP GET request to retrieve a specific resource.



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]**

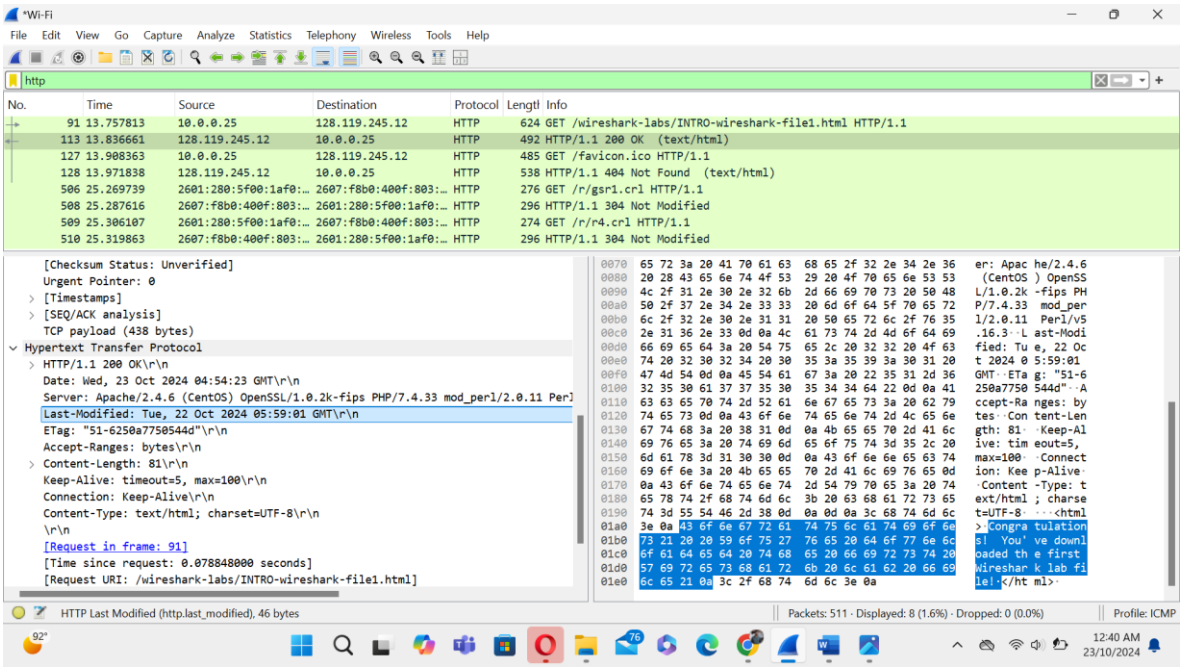
The client is running HTTP/1.1 as seen in the request packet and HTTP server version is also running HTTP/1.1.

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

The status code returned from the server is 200 OK.

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

The If- Modified-Since field in the HTTP request indicates that the client is requesting a file that may have been modified since Mon, 21 Oct 2024 05.59:01 GMT. The actual Last-Modified date from the server is Tue, 22 Oct 2024 05:59:01 GMT.

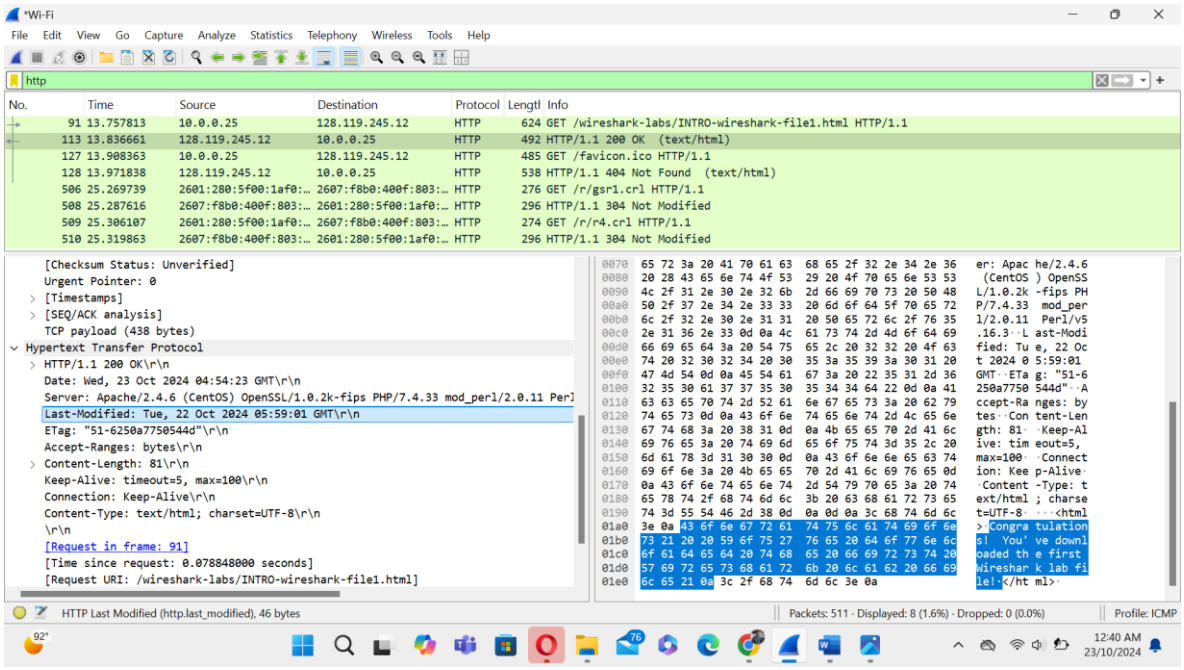


iv. How many bytes of content are being returned to your browser? **[1 point]**

492 bytes of content were returned by the server to the client based on the response packet.

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

Yes, you can see the text displayed on the browser in the Wireshark packets if it is transmitted in an unencrypted form. In the case of HTTP, which operates under plaintext and does not use encryption. So, we can see both client and the response returned by the server, including any HTML content. HTTP traffic is not encrypted, so it is visible to see the actual HTML text, headers when we capture. The TCP payload highlighted in blue in that the content is visible.



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]