



University
of Windsor

Course Name

Networking and Data Security (COMP-8677)

Document Type

Lab 3 Work & Lab 2(Question 4)

Professor

Dr. Shaoquan Jiang

Team - Members

Student ID

Harbhajan Singh

110100089

Ques 4:

FROM LAB2 ASSIGNMENT (Lab 2.4 Question)

Write a Python TCP server program that will accept an unlimited number of connections, one at a time, just as in the sample program in the lecture. Upon receiving a TCP connections request, it should reply with the client's IP address and port number. Then, it waits for commands from the client. Valid commands are "TIME", and "EXIT". To the TIME command, the server should return the current time (see the example below how to obtain a time string). To an invalid command (e.g., HELLO), it returns string "Invalid command!". If the client closes the connection or does not send a command in 15 seconds (see below how to set a timeout for a socket), the server closes the current connection and waits for another connection. To the EXIT command, your server closes all open sockets (including the welcome socket). A sample client program is attached to the assignment. You can use it to test your server. You can modify the client with your own sequence of commands.

Submission requirement:

- Your server programs
- The sequence of commands from the client sides (no need to include your modified client program)
- Screen shot on the running result at the client side (which should include the printout of all the server messages).
- Screen shot on the packet list (no packet details required) between client and server. If you run client server on the same VM, you may need to run Wireshark on the loopback interface.
- All the above are put in a single assignment solution file.

Answer: Coding solution of Question 4 -

Server.py
<pre>""" Student Name - Harbhajan Singh Student ID - 110100089 Course Name - Networking and Data Security Lecture Day - Monday Lab Number - 2 Question - 4 """ ##### CODE ##### #!/usr/bin/env python3 # -*- coding: utf-8 -*- """ Created on Wed Sep 20 12:19:49 2023 @author: bhajji """ #Importing important packages import socket import time</pre>

```

# Function to handle client connections
def handle_client_commands(client_socket, client_address):
    try:
        # Send a welcome message to the client upon connection
        client_socket.send(f"MESSAGE!!! ----> We are now connected to
{client_address}\n".encode())

        # Set a timeout for the client socket
        client_socket.settimeout(15)

        while True:
            # Receive and decode commands from the client
            received_command = client_socket.recv(1024).decode().strip()

            if not received_command:
                break

            if received_command == "TIME":
                # Get and send the current time to the client
                current_time = time.ctime()
                client_socket.send(current_time.encode())
            elif received_command == "EXIT":
                break
            else:
                # Send a warning for invalid commands
                client_socket.send("WARNING!!! ----> Please enter a valid command!\n".encode())

        except socket.timeout:
            print(f"MESSAGE!!! ----> Client at {client_address} timed out...!")
        except Exception as e:
            print(f"ERROR!!! ----> An error occurred ----> {e}")
        finally:
            # Close the client socket
            client_socket.close()

# Function to run the server
def run_server():
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind(('0.0.0.0', 8080))
    server_socket.listen(5) # Allow up to 5 clients to wait in the queue

    print("MESSAGE!!! ----> Server started & listening on port 8080.\n")

    try:
        while True:
            client_socket, client_address = server_socket.accept()
            print(f"MESSAGE!!! ----> The Connection from client {client_address} is accepted...!")
            handle_client_commands(client_socket, client_address)
        except KeyboardInterrupt:
            print("MESSAGE!!! ----> The Server connection is shutting down...!")

```

```
finally:
    server_socket.close()

if __name__ == "__main__":
    run_server()
```

client.py

```
"""
Student Name   - Harbhajan Singh
Student ID     - 110100089
Course Name    - Networking and Data Security
Lecture Day    - Monday
Lab Number     - 2
Question      - 4

##### CODE #####
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""

Created on Wed Sep 20 12:19:49 2023
@author: bhajji
"""

#Importing important packages
import socket

# Function to run client
def run_client():
    # Create a TCP socket
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    # Define the server address and port
    server_address = ('localhost', 8080)

    try:
        # Connect to the server
        client_socket.connect(server_address)

        # Receive and print the initial connection message from the server
        initial_message = client_socket.recv(1024).decode()
        print("MESSAGE!!! ----> Server Message:", initial_message)

        while True:
            # Prompt the user for a command (TIME or EXIT)
            user_command = input("MESSAGE!!! ----> Please enter a command (TIME or EXIT): ").strip()

            # Send the user's command to the server
            client_socket.send(user_command.encode())
```

```

# Check if the user wants to exit
if user_command == "EXIT":
    break

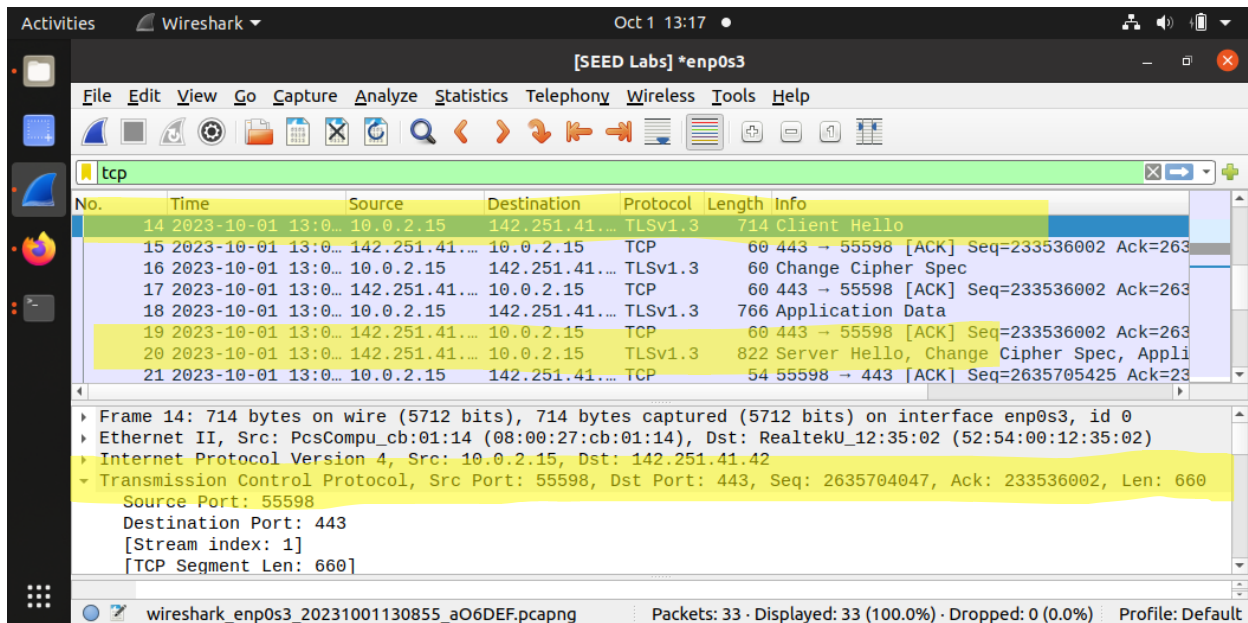
# Receive and print the server's response
response = client_socket.recv(1024).decode()
print("MESSAGE!!! ----> Server Response:", response)

# Adding a pause to keep the console open before exiting
input("MESSAGE!!! ----> Press Enter to exit from the console...")

finally:
    # Close the socket when done
    client_socket.close()

if __name__ == '__main__':
    run_client()

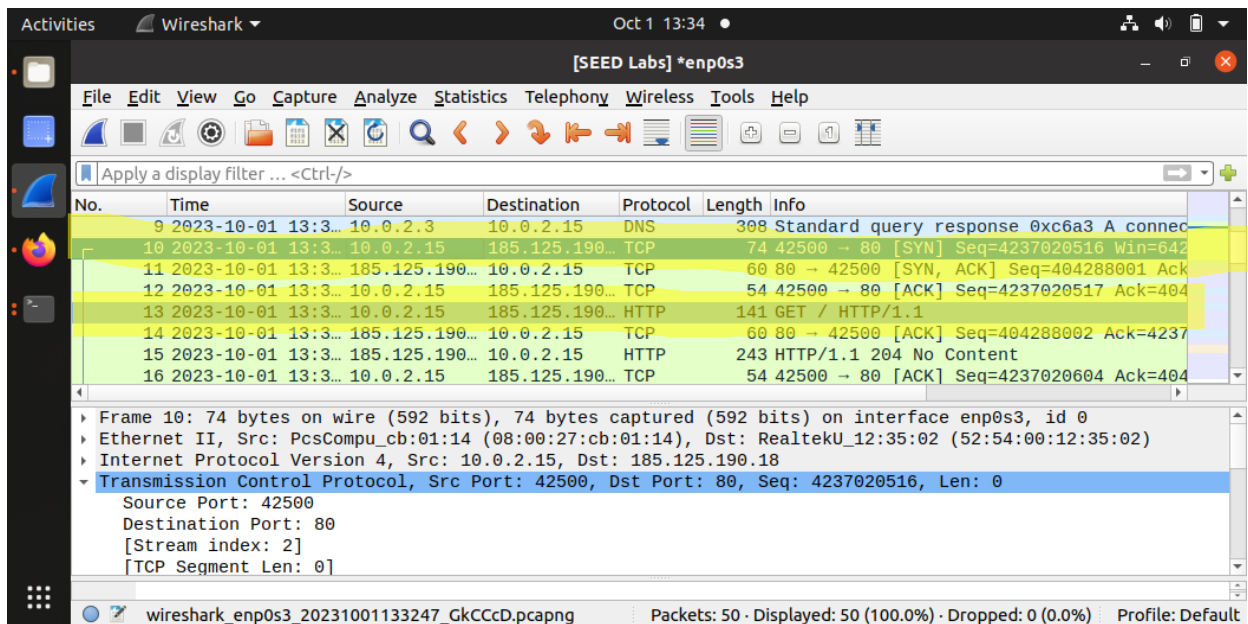
```



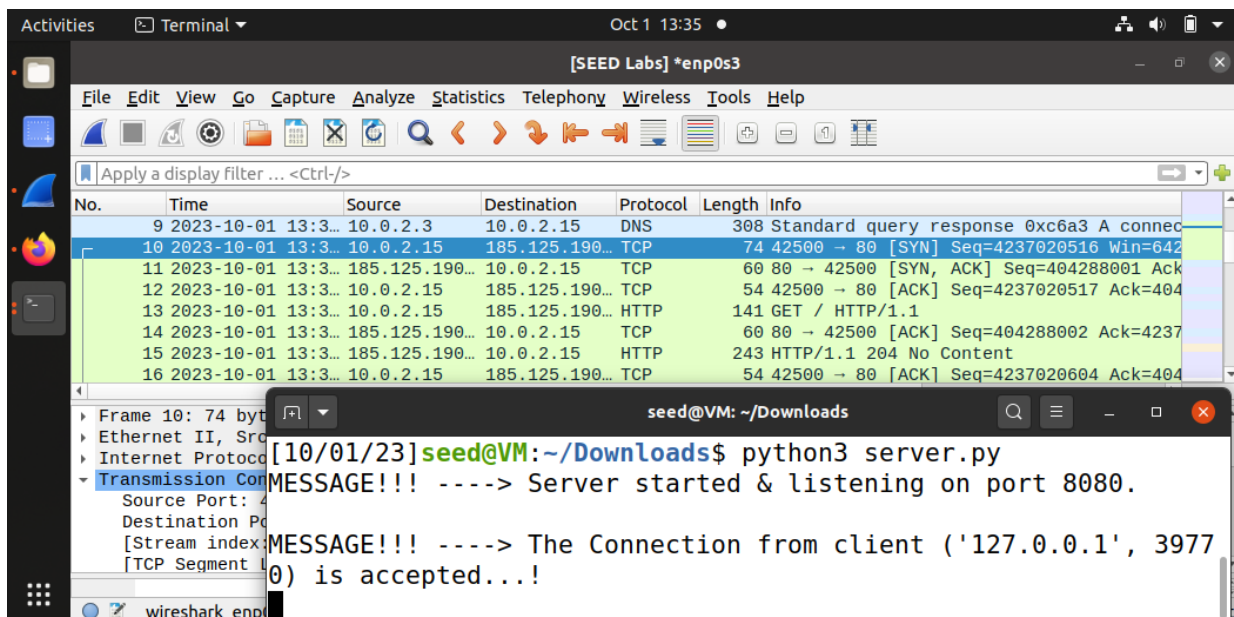
Screenshot 0

Command performed for Server-Client program:

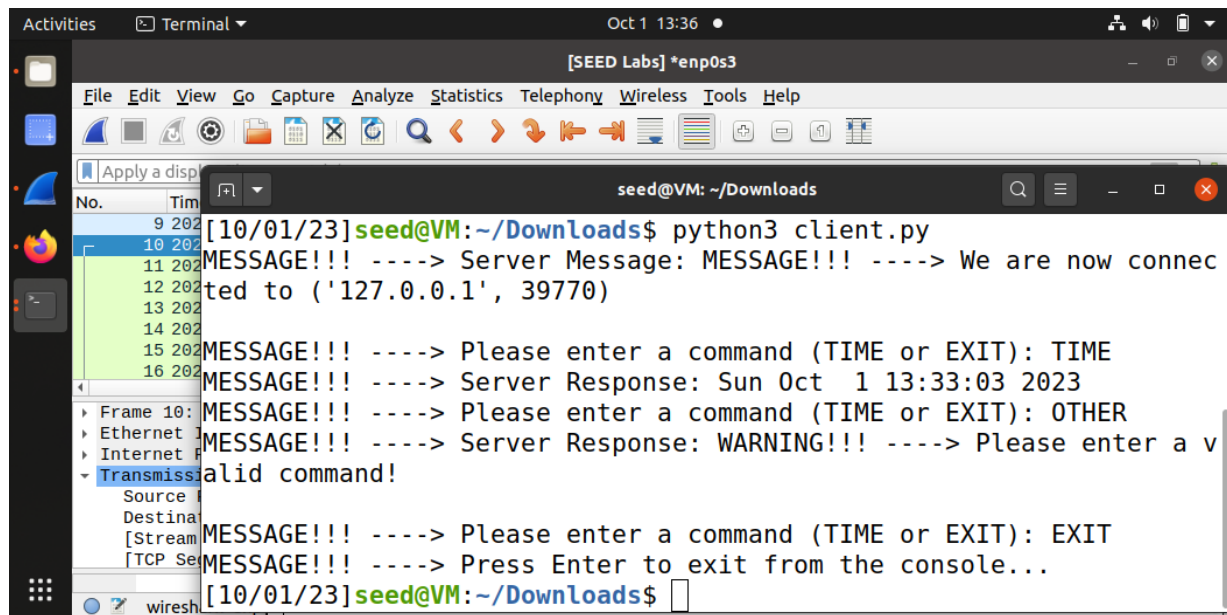
1. python3 server.py
2. python3 client.py
3. TIME
4. OTHER
5. EXIT
6. ENTER



Screenshot 1



Screenshot 2



```
[10/01/23]seed@VM:~/Downloads$ python3 client.py
MESSAGE!!! ----> Server Message: MESSAGE!!! ----> We are now connected to ('127.0.0.1', 39770)
MESSAGE!!! ----> Please enter a command (TIME or EXIT): TIME
MESSAGE!!! ----> Server Response: Sun Oct  1 13:33:03 2023
MESSAGE!!! ----> Please enter a command (TIME or EXIT): OTHER
MESSAGE!!! ----> Server Response: WARNING!!! ----> Please enter a valid command!
MESSAGE!!! ----> Please enter a command (TIME or EXIT): EXIT
MESSAGE!!! ----> Press Enter to exit from the console...
[10/01/23]seed@VM:~/Downloads$
```

Screenshot 3

LAB 3

1. In this problem, you will get familiar with ip format. Start the Wireshark and run ping

www.mit.edu

and then stop Wireshark. Ping www.mit.edu is to send an icmp packet. Check the first echo request packet in the Wireshark window and answer the following questions.

A. Look at the ip header, what is the source and destination ip address?

B. What is the upper layer protocol in ip header?

C. what is the ip header length?

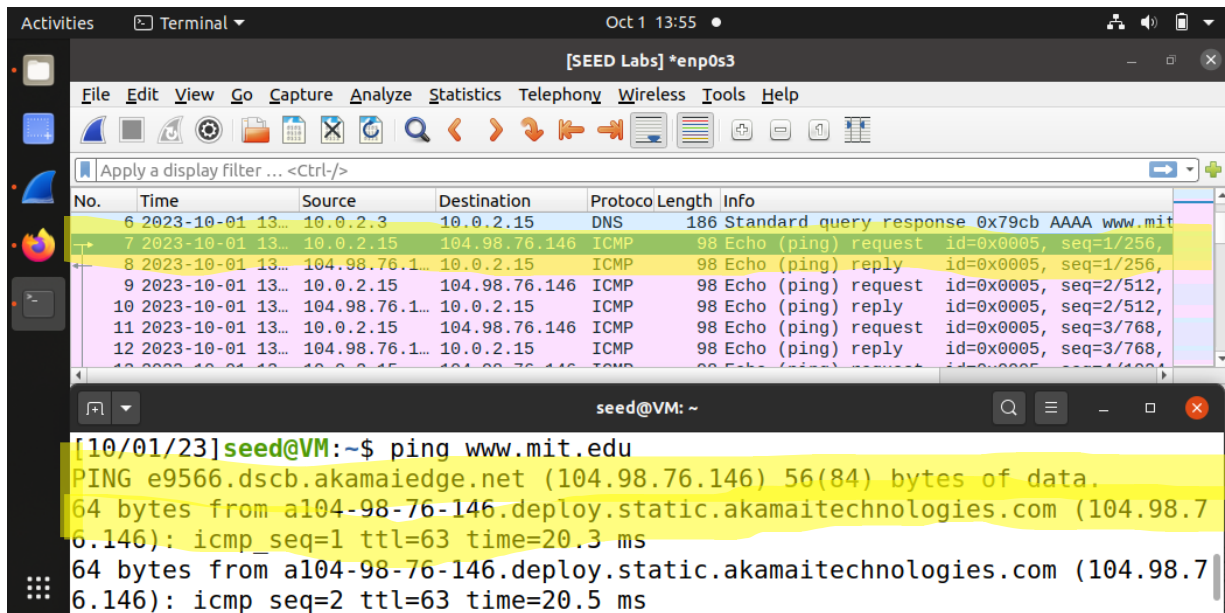
D. Calculate the payload length for ip packet. This is totallength - headerlength.

E. what is the TTL value and what is its meaning?

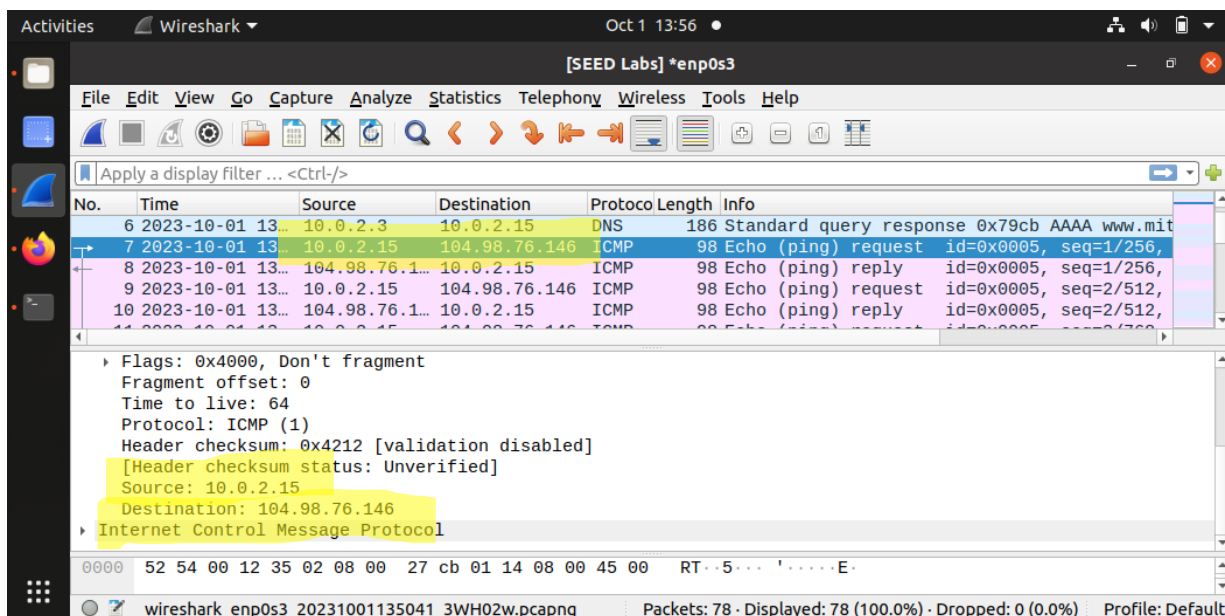
F. find out which field shows the ip header is in ipv4 or ipv6 format.

SOLUTION:

1A)



Screenshot 4

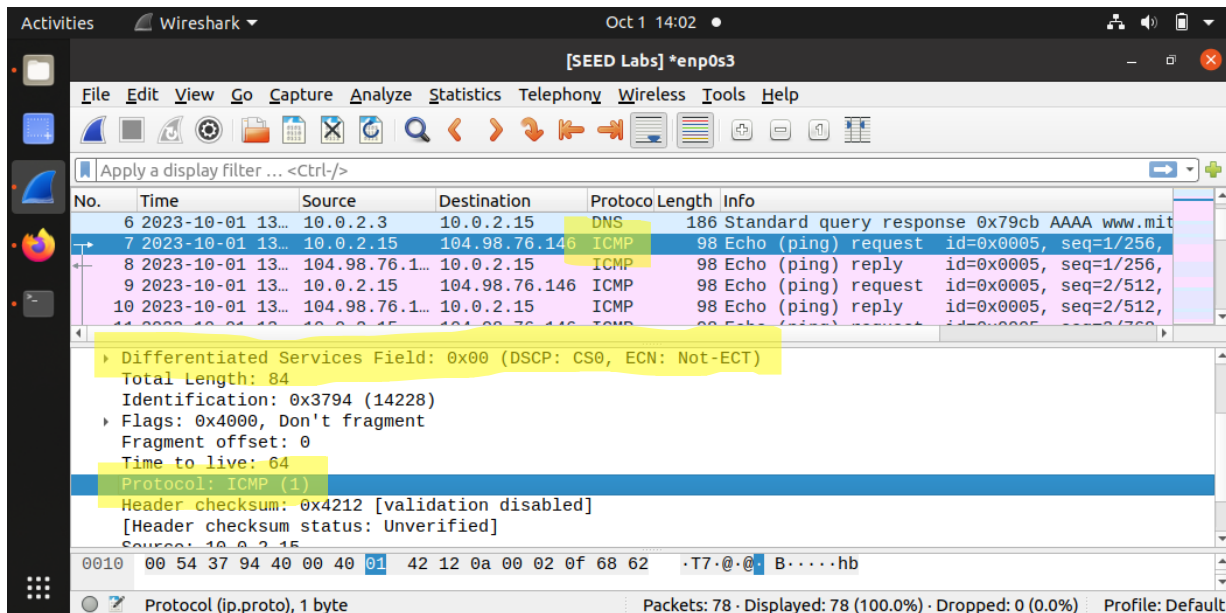


Screenshot 5

The source IP address in the IP header is **10.0.2.15**, and

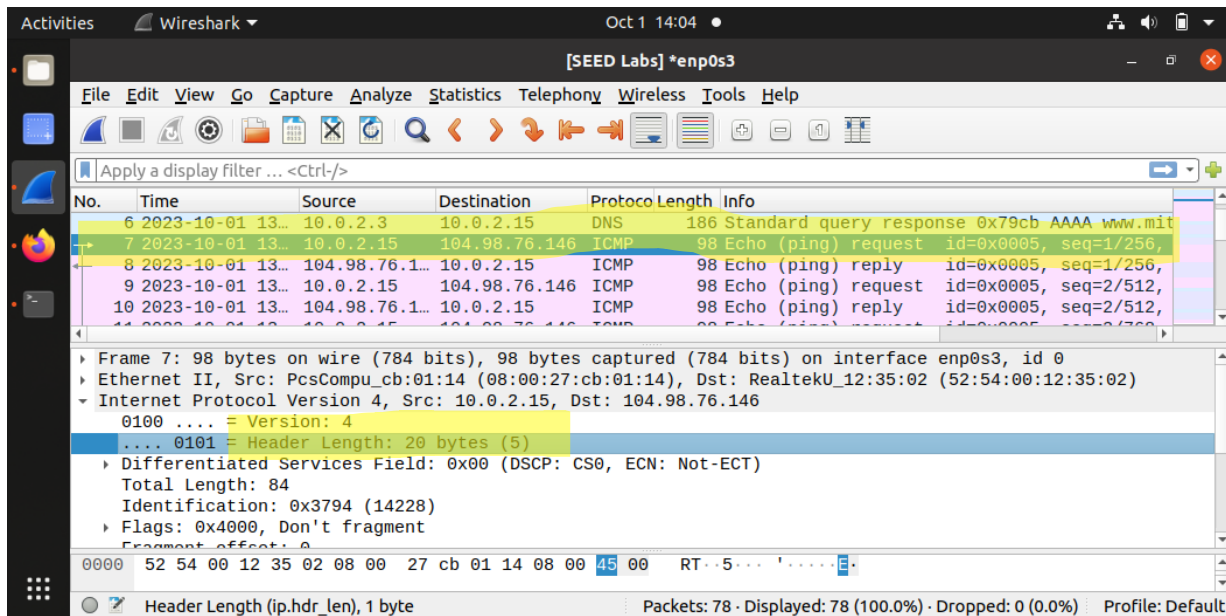
The destination IP address is **104.98.76.146**

1B) In the IP header, the protocol identified in the upper layer is ICMP (Internet Control Message Protocol), which is assigned a protocol number of 1.



Screenshot 6

1C) The length of the IP header is indicated in the "Header Length" field, and it is set to 20 bytes.



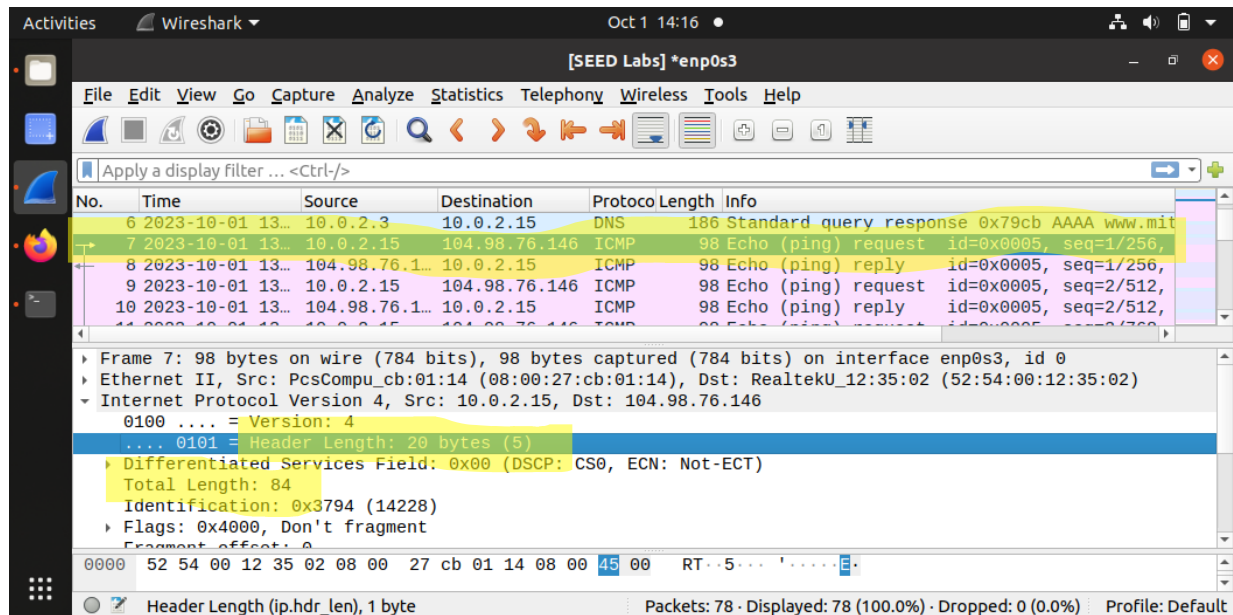
Screenshot 7

1D) Payload length = total length - header length

Now, total length = 84 bytes

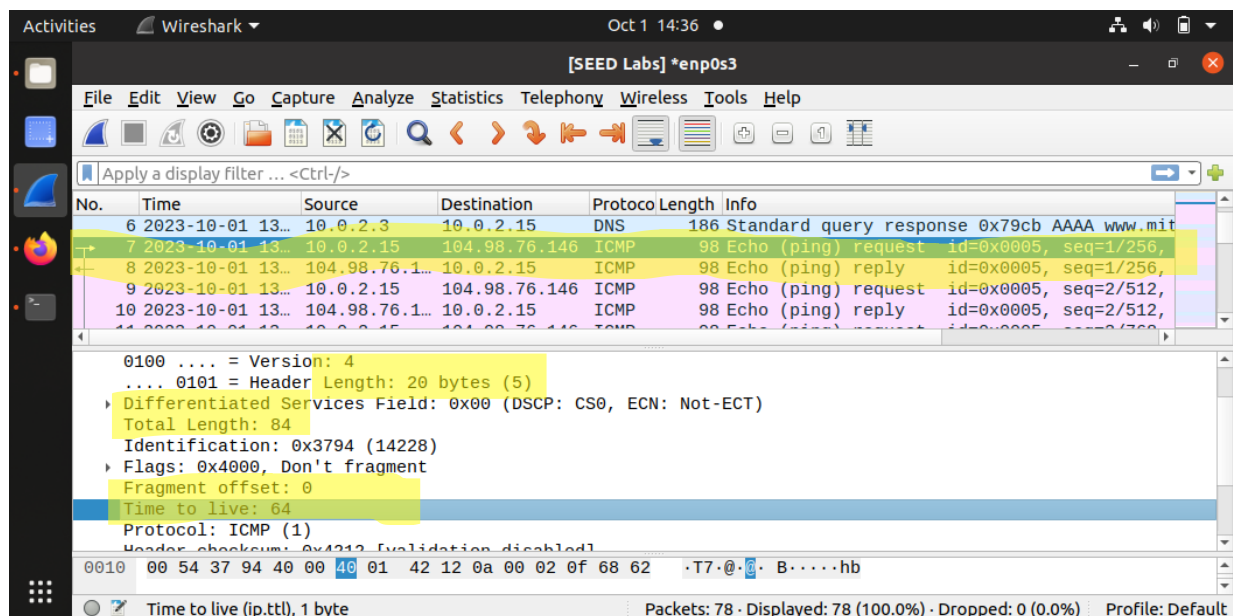
header length = 20 bytes

Then, **Payload length** = (84 – 20) bytes = **64 bytes**



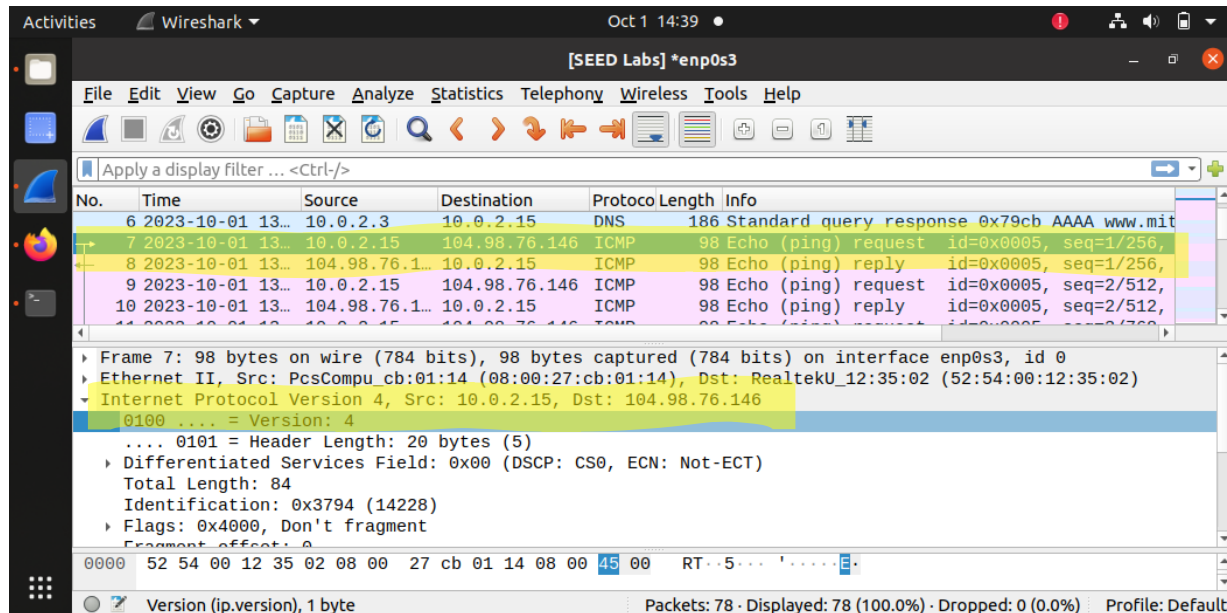
Screenshot 8

1E) The Time to Live (TTL) value found in the IP header is typically set to 64. This TTL value serves as a representation of the maximum number of hops or routers that a packet is allowed to traverse through before it is no longer valid. Essentially, with each router the packet encounters on its journey, the TTL value decreases by 1. Once the TTL value reaches 0, the packet is intentionally discarded. This mechanism is in place to prevent packets from endlessly circulating within the network, ensuring that they have a finite lifespan as they travel from source to destination.



Screenshot 9

1F) The indicator for distinguishing between IPv4 and IPv6 within the IP header format is the "Version" field. This field comprises the first 4 bits of the IP header and, in the context mentioned, it is configured with a value of 4. This value signifies that the IP header conforms to the IPv4 format.

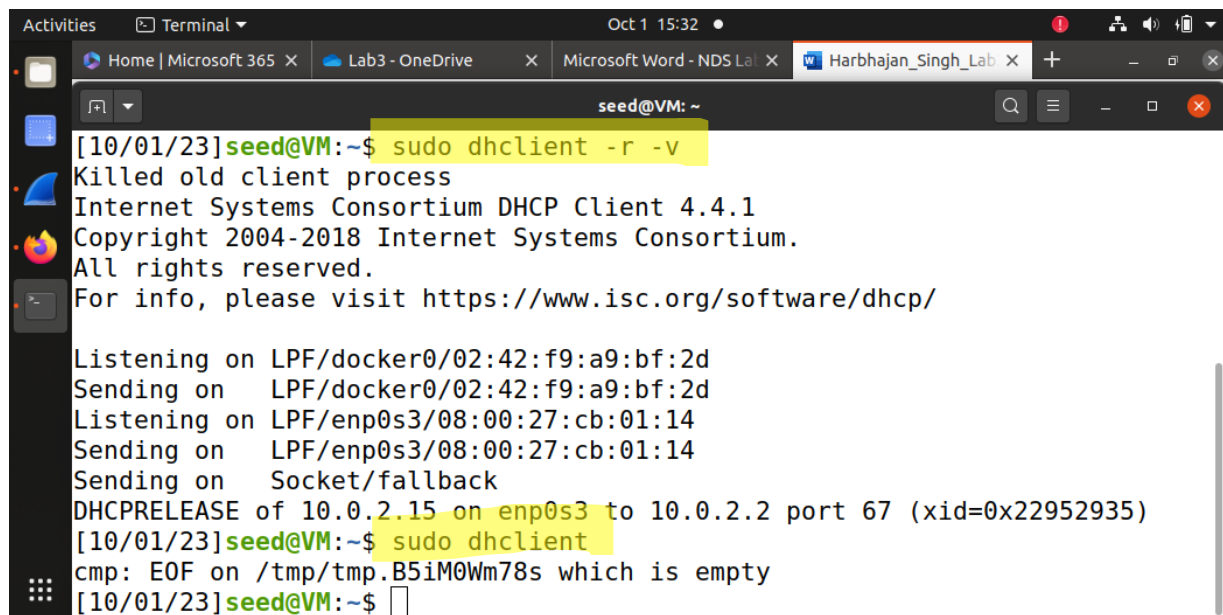


Screenshot 10

QUESTION 2: Start Wireshark on your VM. Next, run command `sudo dhclient -r -v` and then `sudo dhclient` and finally stop Wireshark. Command `sudo dhclient -r -v` will release your current ip address. Then `sudo dhclient` will execute the DHCP protocol. Use packets in Wireshark from executing DHCP to answer the following questions.

- A) Confirm that the transport layer protocol of DHCP protocol is UDP. To do this, check a packet with DHCP protocol data and look at the transport layer header. Think about why it is not TCP (recall that TCP needs to establish a connection before exchanging messages).

Answer 2A:



```
[10/01/23]seed@VM:~$ sudo dhclient -r -v
Killed old client process
Internet Systems Consortium DHCP Client 4.4.1
Copyright 2004-2018 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/

Listening on LPF/docker0/02:42:f9:a9:bf:2d
Sending on   LPF/docker0/02:42:f9:a9:bf:2d
Listening on LPF/enp0s3/08:00:27:cb:01:14
Sending on   LPF/enp0s3/08:00:27:cb:01:14
Sending on   Socket/fallback
DHCPRELEASE of 10.0.2.15 on enp0s3 to 10.0.2.2 port 67 (xid=0x22952935)
[10/01/23]seed@VM:~$ sudo dhclient
cmp: EOF on /tmp/tmp.B5iM0Wm78s which is empty
[10/01/23]seed@VM:~$
```

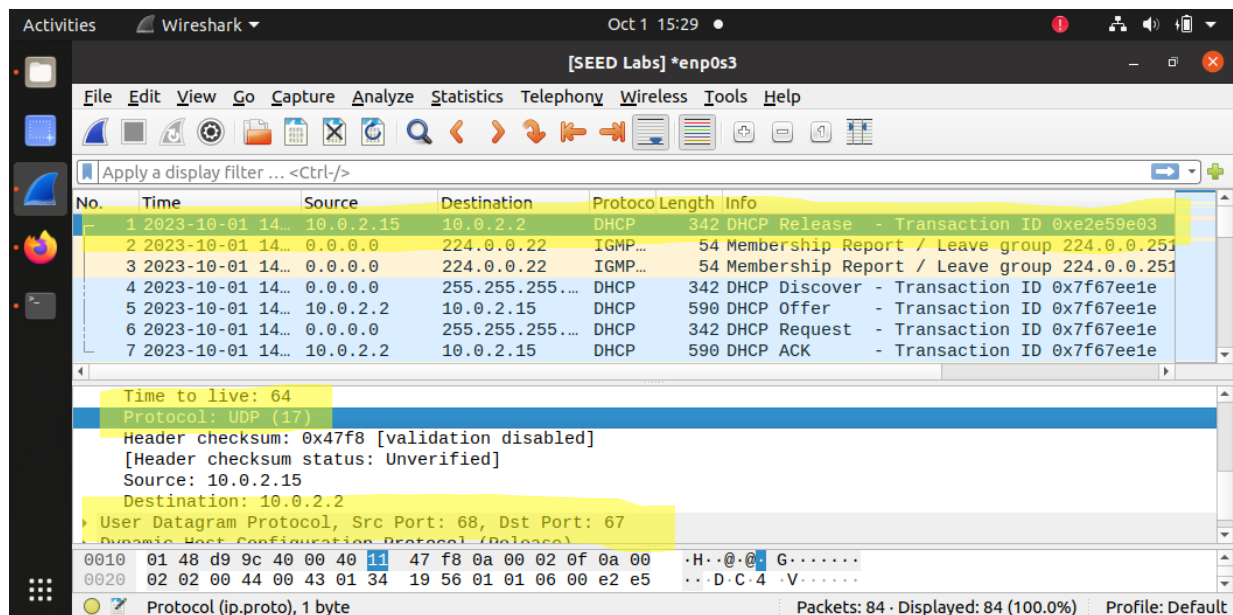
Screenshot 11

In the provided screenshot, we can determine that the transport layer protocol for DHCP (Dynamic Host Configuration Protocol) is UDP (User Datagram Protocol), as evident from packet number 1.

Upon examining the packet details:

The "Protocol" field within the IP header indicates the use of UDP, identified by a protocol number of 17. Further analysis in the UDP section reveals that the source port is 68, while the destination port is 67. Typically, port 68 is employed by DHCP clients to transmit requests, while port 67 is designated for DHCP servers to receive these requests.

The choice of UDP over TCP in DHCP is deliberate. UDP is characterized by its connectionless and lightweight nature, making it a suitable transport protocol for DHCP. DHCP primarily involves the exchange of concise, stateless messages between the client and server, aimed at configuring network parameters. Given the simplicity of DHCP messages, they do not require the overhead associated with establishing and maintaining a connection. Consequently, UDP proves to be a more efficient choice for this particular protocol.



Screenshot 12

B) In addition to offer the ip address to your computer, DHCP can in fact provide you more useful configuration. Check DHCP offer packet to find out the following information.

DHCP server IP: you need this to extend your time to use the current IP address.

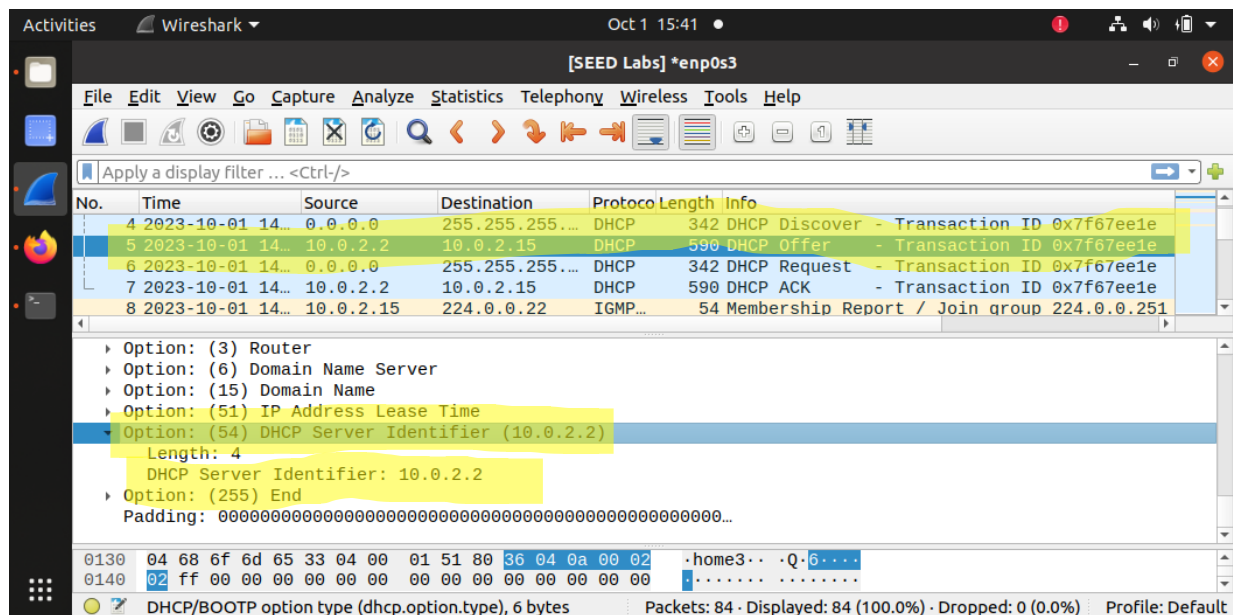
Subnet mask: this tells you the subnet type.

Router IP: That is the ip address your outgoing packet will first go to.

DNS IP: this is the ip address of the DNS server that you will request to resolve your DNS query. That is, this is your **local** DNS server.

ANSWER 2B:

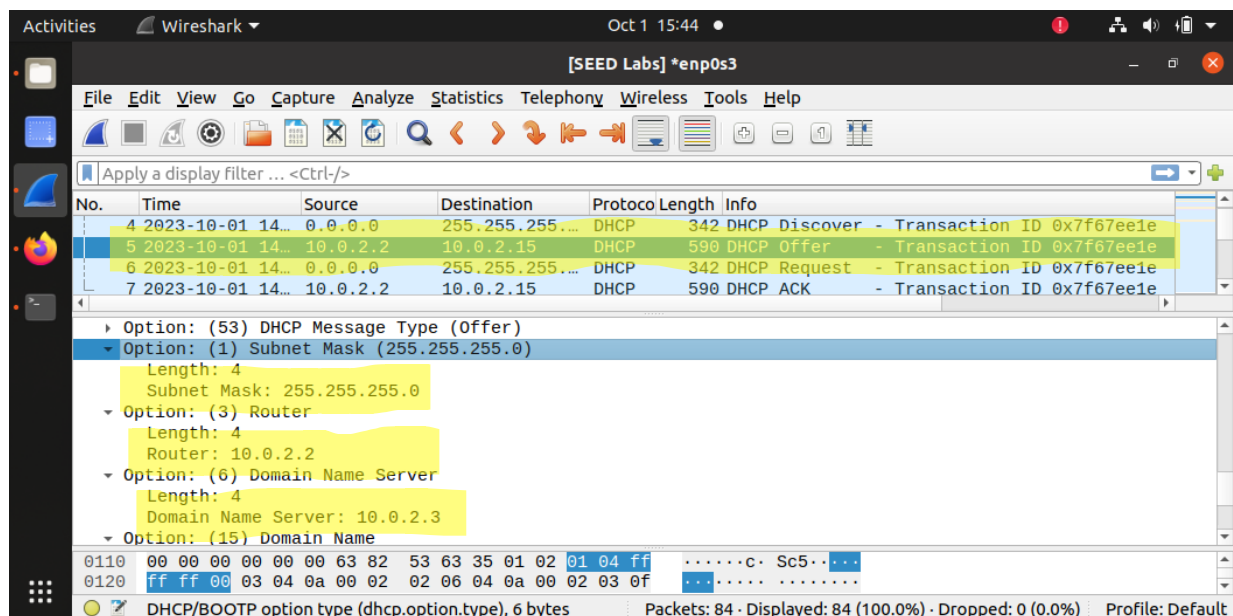
DHCP Server IP: The IP address of the DHCP server can be found in the "DHCP Server Identifier" option, specifically in Option 54. In the 'DHCP offer packet,' the **DHCP Server Identifier** is indicated as **10.0.2.2**.



Subnet Mask: The subnet mask is provided in the "Subnet Mask" option (Option 1). In 'DHCP Offer packet', the **Subnet Mask** is **255.255.255.0**.

Router IP (Gateway): The router's (or gateway's) IP address is provided in the "Router" option (Option 3). In 'DHCP Offer packet', the **Router IP** is **10.0.2.2**

DNS IP: The DNS server's IP addresses are provided in the "Domain Name Server" option (Option 6). In 'DHCP Offer packet', the **DNS server IP** addresses is **10.0.2.3**



Question 3. In this exercise, you will look in the arp protocol execution. First, run arp to find out the list of records in the arp table. Next, start your wireshark and run `sudo arp -d routerIP` to delete the record of routerIP. Here routerIP is the Router IP obtained in the previous DHCP experiment. Then, you should see your VM is now starting to run arp.

- A.** Find our arp broadcast from your VM. What is the upper layer protocol in the link layer header? What is the broadcast MAC address? What is the ip address for which your broadcast message is intended to find out the MAC address?

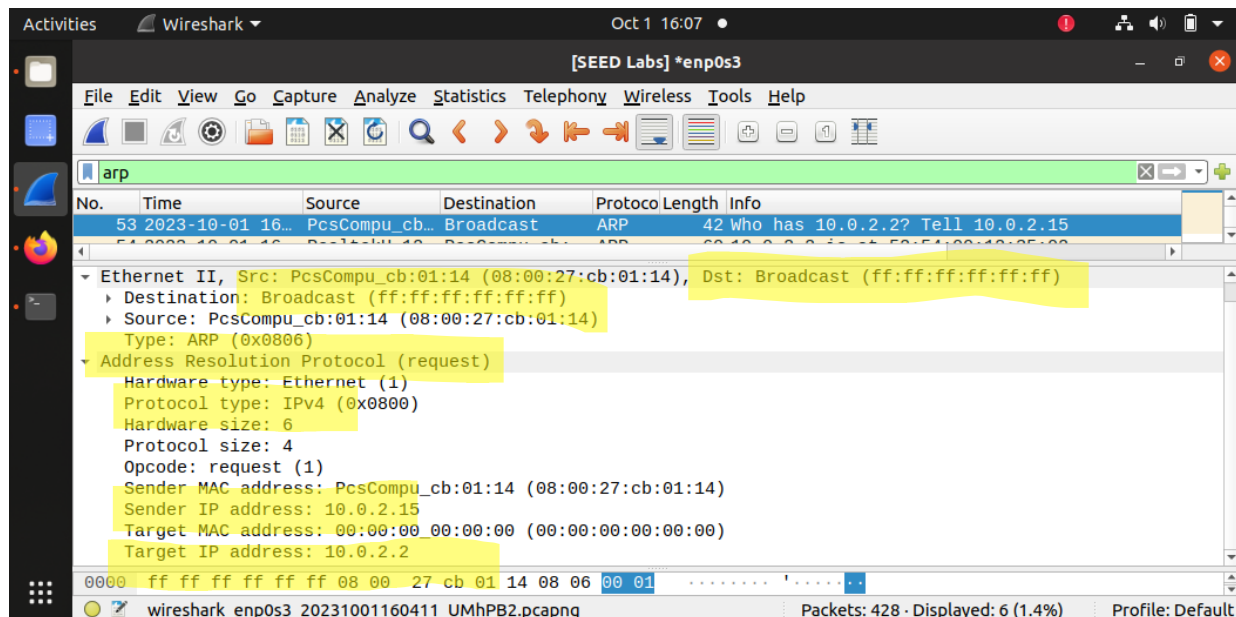
Answer 3A:

Upper layer protocol: IPv4 (0x0800)

Broadcast MAC: ff:ff:ff:ff:ff:ff

Target IP: 10.0.2.2

Sender IP: 10.0.2.15

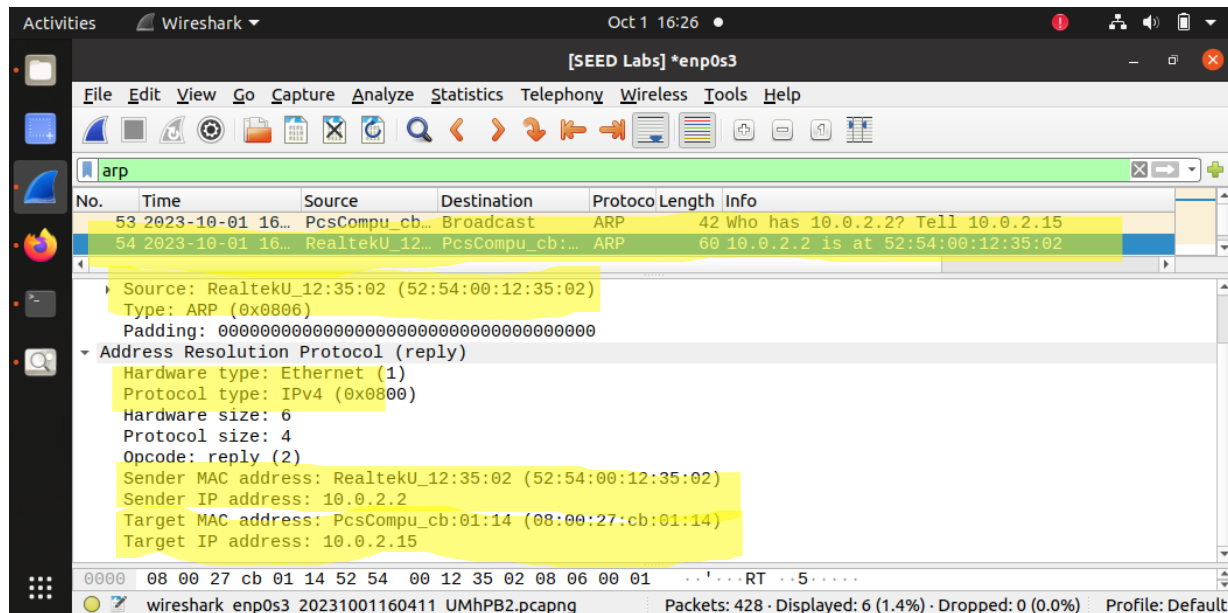


Screenshot 15

ANSWER 3B:

MAC Address of the Sender: The sender's MAC address is 52:54:00:12:35:02

IP Address of the Sender: The sender's IP address is 10.0.2.2.



Screenshot 16

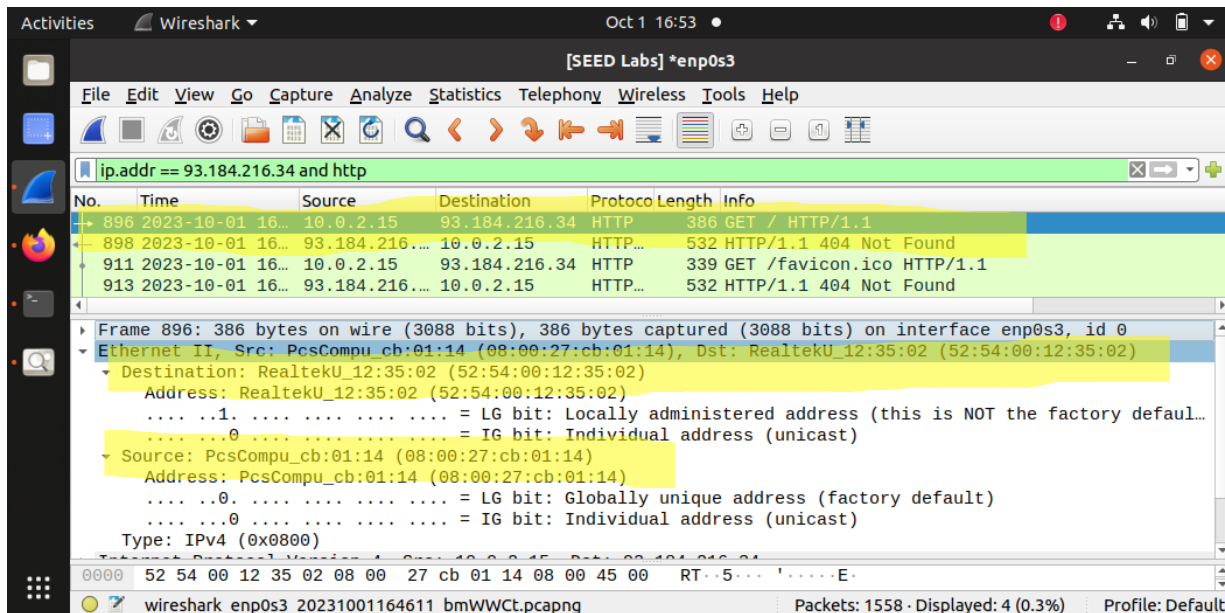
QUESTION 4: Run Wireshark and access www.example.com and stop Wireshark. Answer the following questions.

- A. Check the HTTP request packet to 93.184.216.34 (ip of www.example.com). What are the source MAC and destination MAC? You need to check the link layer header in the packet. The source MAC is the MAC of your VM.**

ANSWER 4A:

Destination MAC: 52:54:00:12:35:02 (RealtekU_12:35:02, the target system)

Source MAC: 08:00:27:cb:01:14 (MAC of My VM)

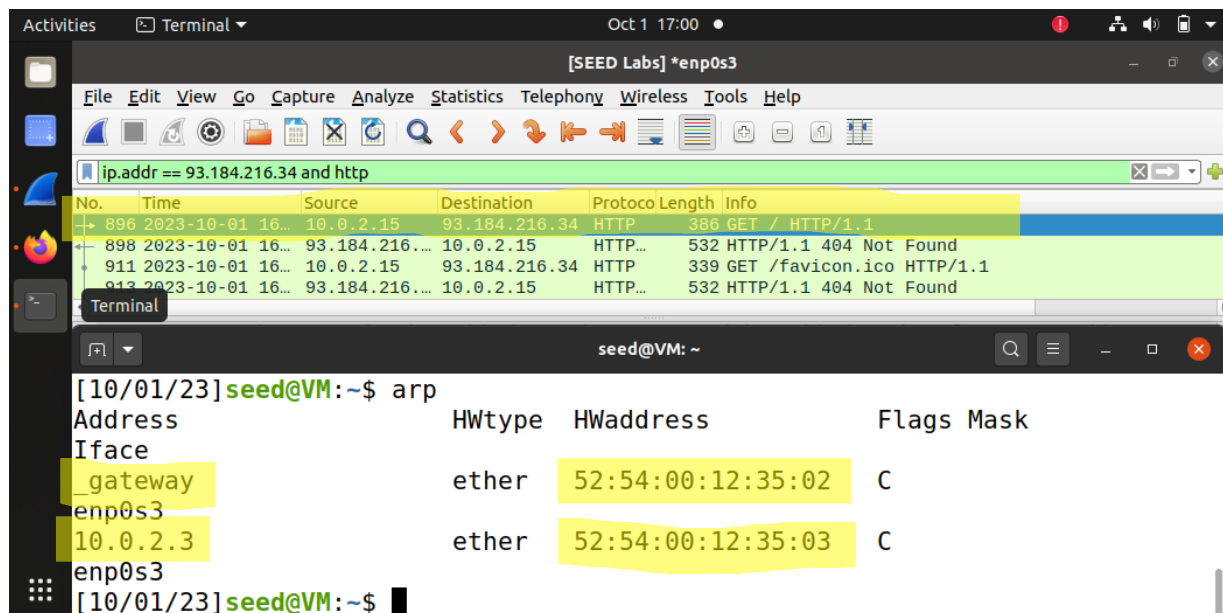


Screenshot 17

B. Does the destination MAC in a belong to 93.184.216.34? To find out your answer, run command arp to check the arp table of your VM. Is the destination MAC in a listed here? If yes, confirm that this MAC does not belong to 93.184.216.34 and instead belong to your router.

ANSWER 4B:

Yes, when we look at packet Number 896, we can see that the destination MAC address is 52:54:00:12:35:02, as shown in Screenshot 17. However, it's essential to note that this MAC address is found in the ARP table output. Interestingly, it's associated with the IP address '_gateway,' rather than 93.184.216.34. This MAC address is actually linked to our router, which is '_gateway.' So, it's clear that it doesn't correspond to the IP address 93.184.216.34.

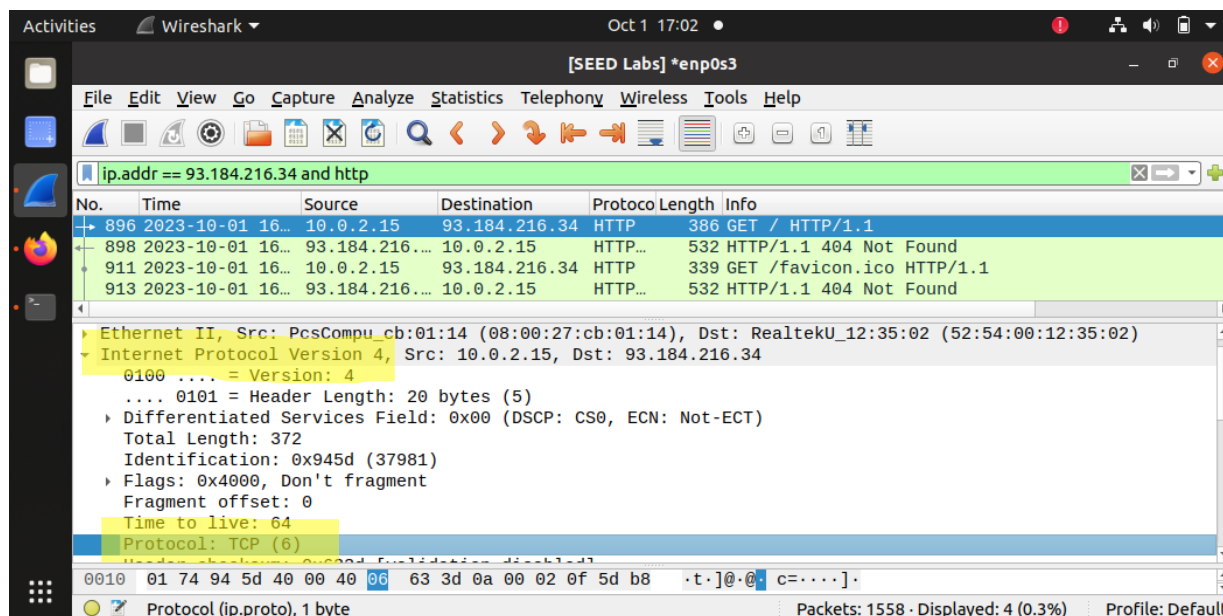


Screenshot 18

C. In the upper protocol field of link layer header of your HTTP request packet, what is the value? What protocol does it represent?

ANSWER 4C:

The upper protocol field in the link layer header of the HTTP request packet has a value of IPv4 (0x0800). This value represents the Internet Protocol version 4 (IPv4) at the network layer. It represents TCP (Transmission Control Protocol).



Screenshot 19

References

- 1. Week 3 – Class 3 Notes**
- 2. Week 3 – Class 3 Instruction Document**