Course Name
Networking and Data Security (COMP-8677)

Document Type
Lab 3 Work + Lab 2.4 Question

Professor
Dr. Shaoquan Jiang

Team - Members	Student ID
Manjinder Singh	110097177

FROM PREVIOUS ASSIGNEMNT(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 connection 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 program**
- **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.**

0. Submit the coding solution for Q4 in Lab Assignment 2.

Answer –

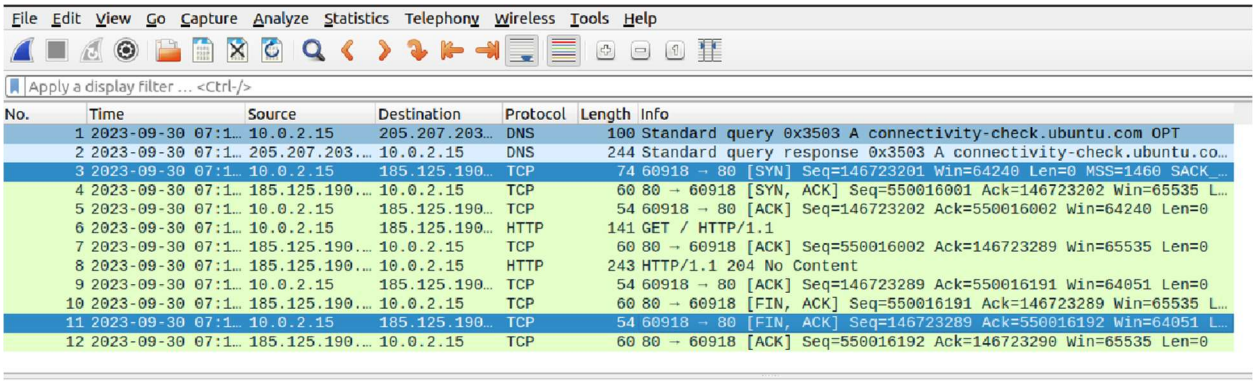
Server Program

```
"""
Name - Manjinder Singh
Student ID - 110097177
Subject - Networking and Data Security
Class Day - Monday
Lab - 2; Question - 4
"""

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

"""
Created on Mon Sep 18 18:00:07 2023
@author: Manjinder Singh
"""

# Package import for socket and time library.
import socket
```

```
import time

def clientCmds(clientSock, clientAddr):
    try:
        clientSock.send(f"We are now connected to {clientAddr}\n".encode())

        clientSock.settimeout(15)

        while True:
            recCmd = clientSock.recv(1024).decode().strip()

            if not recCmd:
                break

            if recCmd == "TIME":
                currTime = time.ctime()
                clientSock.send(currTime.encode())
            elif recCmd == "EXIT":
                break
            else:
                clientSock.send("Invalid command entered!\n".encode())

    except socket.timeout:
        print(f"Client at {clientAddr} timed out.")
    except Exception as e:
        print(f"ISSUE ENCOUNTERED: An error occurred: {e}")
    finally:
        clientSock.close()

def serverFunctionality():
    serverSock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    serverSock.bind(('0.0.0.0', 8080))
    serverSock.listen(5)  # Allow up to 5 clients to wait in the queue

    print("Server started & listening on port 8080.\n")

    try:
        while True:
            clientSock, clientAddr = serverSock.accept()
            print(f"Connection is accepted from {clientAddr}")
            clientCmds(clientSock, clientAddr)
    except KeyboardInterrupt:
        print("Server connection is shutting down...")
    finally:
        serverSock.close()
```

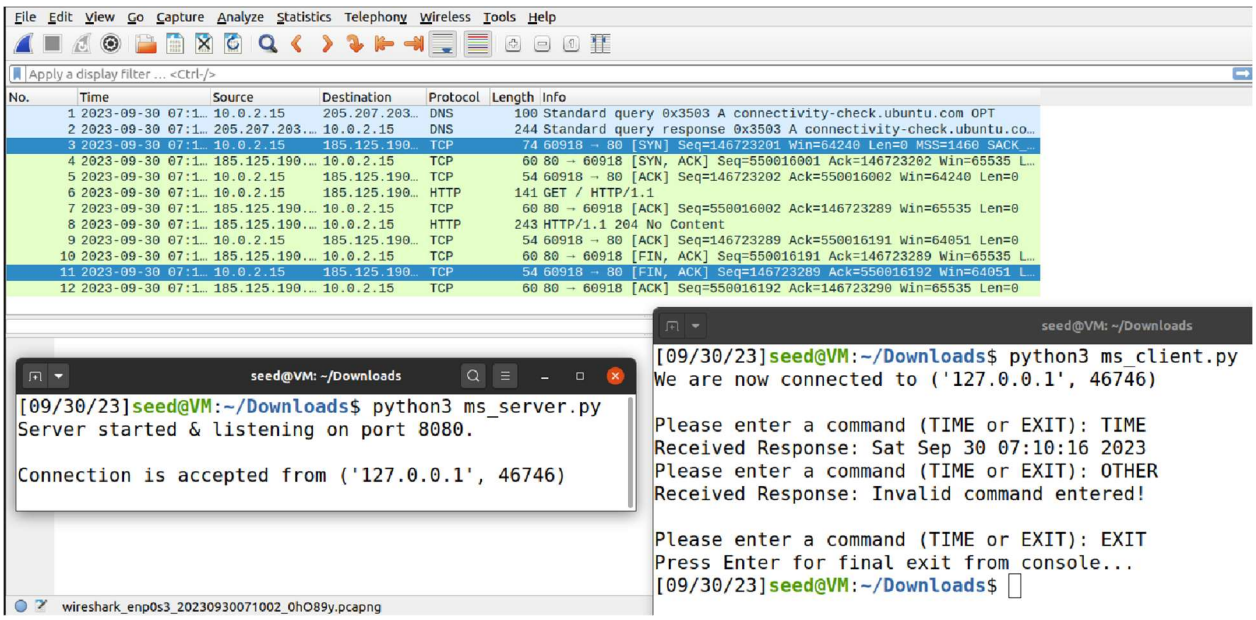
```
if __name__ == "__main__":
    serverFunctionality()
```

Client and Server Code is uploaded to One Drive Folder for Reference(Only Accounts with Uwindsor can access) :
[Networking and Data Security - Lab 3 - Submitted to Doc]



Screenshot: 1 (Packet List)

Sequence of Commands executed on the Client Side
python3 msClient.py TIME OTHER EXIT (ENTER)



Screenshot: 2 (Packet List, Running result at the client side, and client connection with server)

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** - **headerlegnth**.

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.

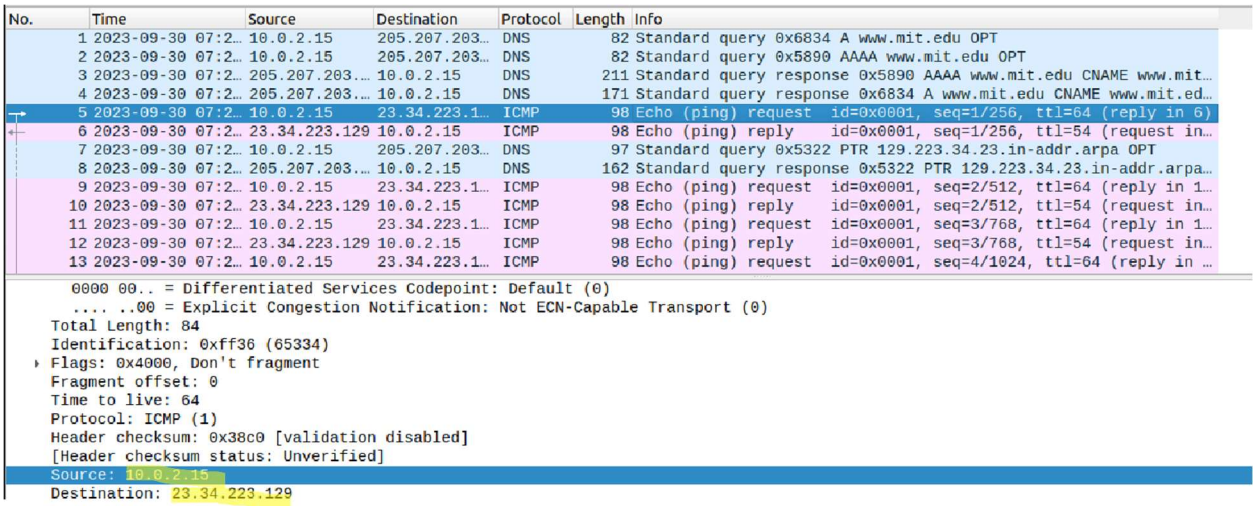
Answer –

```
seed@VM:~$ ping www.mit.edu
PING e9566.dscb.akamaiedge.net (104.93.164.136) 56(84) bytes of data.
64 bytes from a104-93-164-136.deploy.static.akamaitechnologies.com (104.93.164.136): icmp_seq=1
 ttl=57 time=12.5 ms
64 bytes from a104-93-164-136.deploy.static.akamaitechnologies.com (104.93.164.136): icmp_seq=2
 ttl=57 time=93.4 ms
64 bytes from a104-93-164-136.deploy.static.akamaitechnologies.com (104.93.164.136): icmp_seq=3
 ttl=57 time=21.9 ms
64 bytes from a104-93-164-136.deploy.static.akamaitechnologies.com (104.93.164.136): icmp_seq=4
 ttl=57 time=19.2 ms
64 bytes from a104-93-164-136.deploy.static.akamaitechnologies.com (104.93.164.136): icmp_seq=5
 ttl=57 time=18.7 ms
```

No.	Time	Source	Destination	Protocol	Length	Info
1	2023-09-30 07:2…	10.0.2.15	205.207.203…	DNS	82	Standard query 0x6834 A www.mit.edu OPT
2	2023-09-30 07:2…	10.0.2.15	205.207.203…	DNS	82	Standard query 0x5890 AAAA www.mit.edu OPT
3	2023-09-30 07:2…	205.207.203.…	10.0.2.15	DNS	211	Standard query response 0x5890 AAAA www.mit.edu CNAME www.mit…
4	2023-09-30 07:2…	205.207.203.…	10.0.2.15	DNS	171	Standard query response 0x6834 A www.mit.edu CNAME www.mit.ed…
5	2023-09-30 07:2…	10.0.2.15	23.34.223.1…	ICMP	98	Echo (ping) request id=0x0001, seq=1/256, ttl=64 (reply in 6)
6	2023-09-30 07:2…	23.34.223.129	10.0.2.15	ICMP	98	Echo (ping) reply id=0x0001, seq=1/256, ttl=54 (request in…
7	2023-09-30 07:2…	10.0.2.15	205.207.203…	DNS	97	Standard query 0x5322 PTR 129.223.34.23.in-addr.arpa OPT
8	2023-09-30 07:2…	205.207.203.…	10.0.2.15	DNS	162	Standard query response 0x5322 PTR 129.223.34.23.in-addr.arpa…
9	2023-09-30 07:2…	10.0.2.15	23.34.223.1…	ICMP	98	Echo (ping) request id=0x0001, seq=2/512, ttl=64 (reply in 1…
10	2023-09-30 07:2…	23.34.223.129	10.0.2.15	ICMP	98	Echo (ping) reply id=0x0001, seq=2/512, ttl=54 (request in…
11	2023-09-30 07:2…	10.0.2.15	23.34.223.1…	ICMP	98	Echo (ping) request id=0x0001, seq=3/768, ttl=64 (reply in 1…
12	2023-09-30 07:2…	23.34.223.129	10.0.2.15	ICMP	98	Echo (ping) reply id=0x0001, seq=3/768, ttl=54 (request in…
13	2023-09-30 07:2…	10.0.2.15	23.34.223.1…	ICMP	98	Echo (ping) request id=0x0001, seq=4/1024, ttl=64 (reply in …

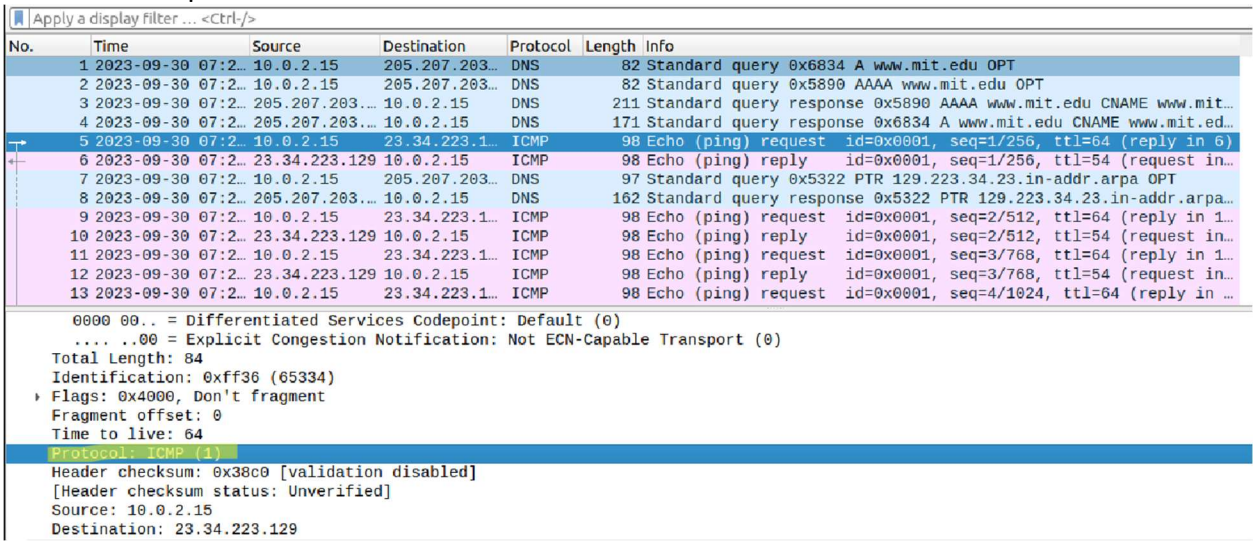
Screenshot: 3

(a) The source IP address in the IP header is 10.0.2.15, and the destination IP address is 23.34.223.129.



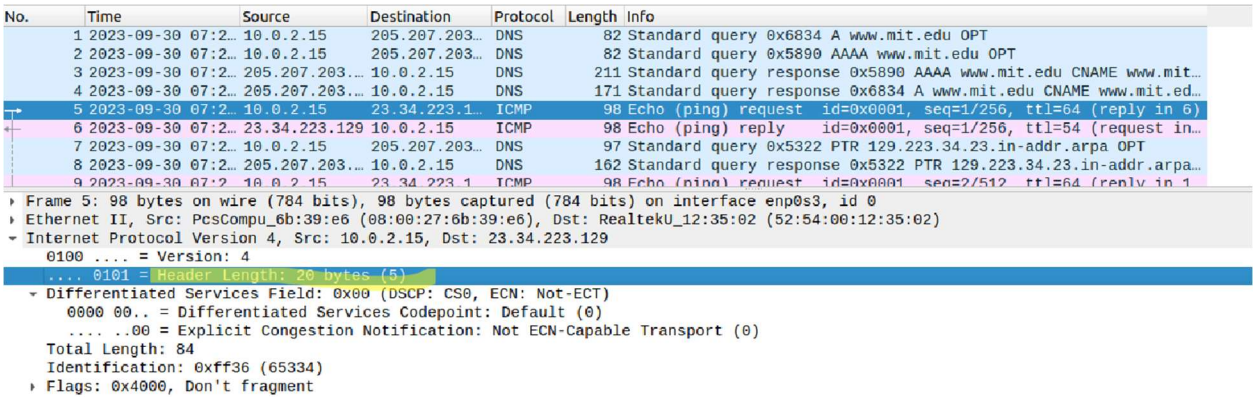
Screenshot: 4

(b) The upper-layer protocol in the IP header is ICMP (Internet Control Message Protocol), with a protocol number of 1.



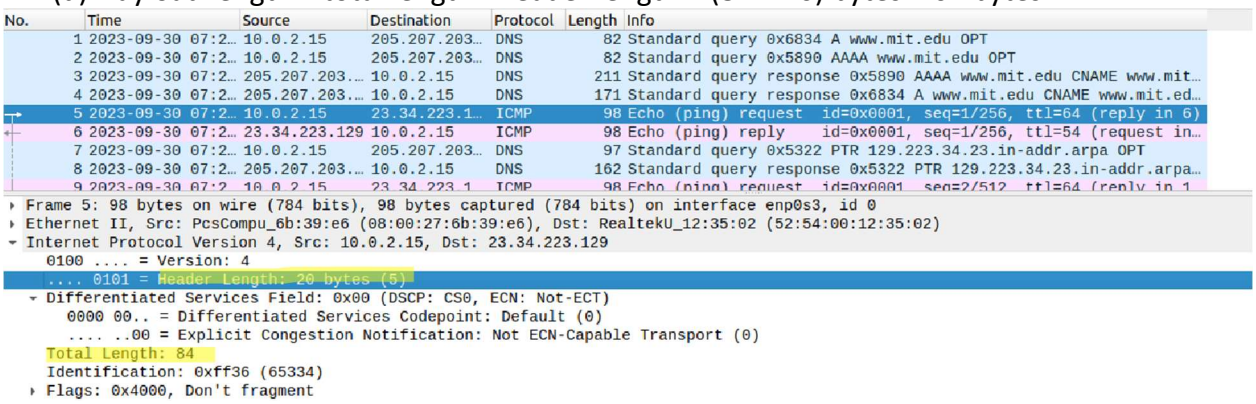
Screenshot: 5

(c) The IP header length is specified in the "Header Length" field, which is 20 bytes



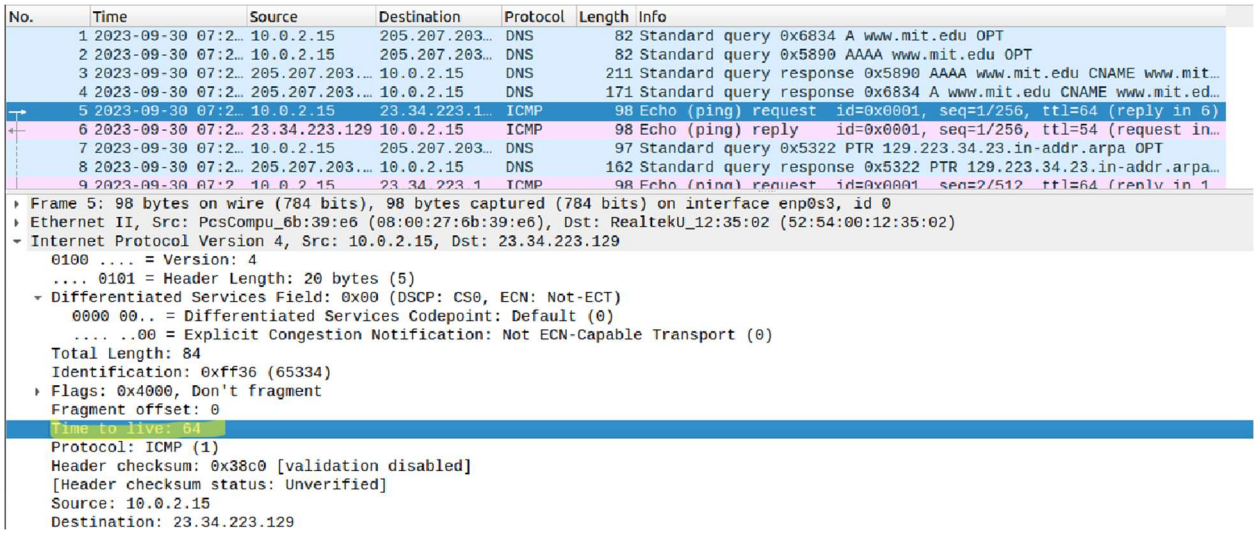
Screenshot: 6

(d) Payload length = total length - header length = (84 – 20) bytes = 64 bytes



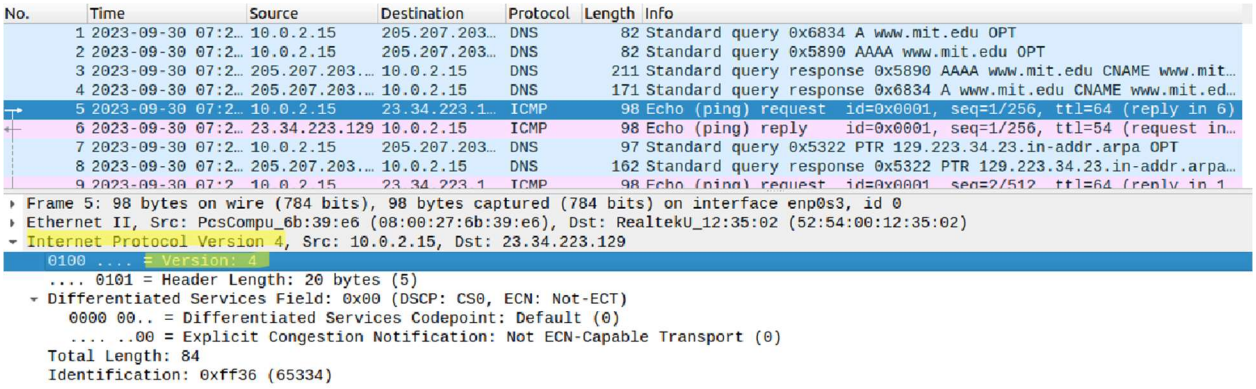
Screenshot: 7

(e) The Time to Live (TTL) value in the IP header is 64. The TTL field represents the maximum number of hops (routers) the packet can pass through before it is discarded. Each router decrements the TTL value by 1, and when it reaches 0, the packet is discarded to prevent it from looping indefinitely in the network.



Screenshot: 8

(f) The field that shows the IP header format (IPv4 or IPv6) is the "Version" field, which is the first 4 bits of the IP header. In this case, the "Version" field is set to 4, which indicates that the IP header is in IPv4 format.



Screenshot: 9

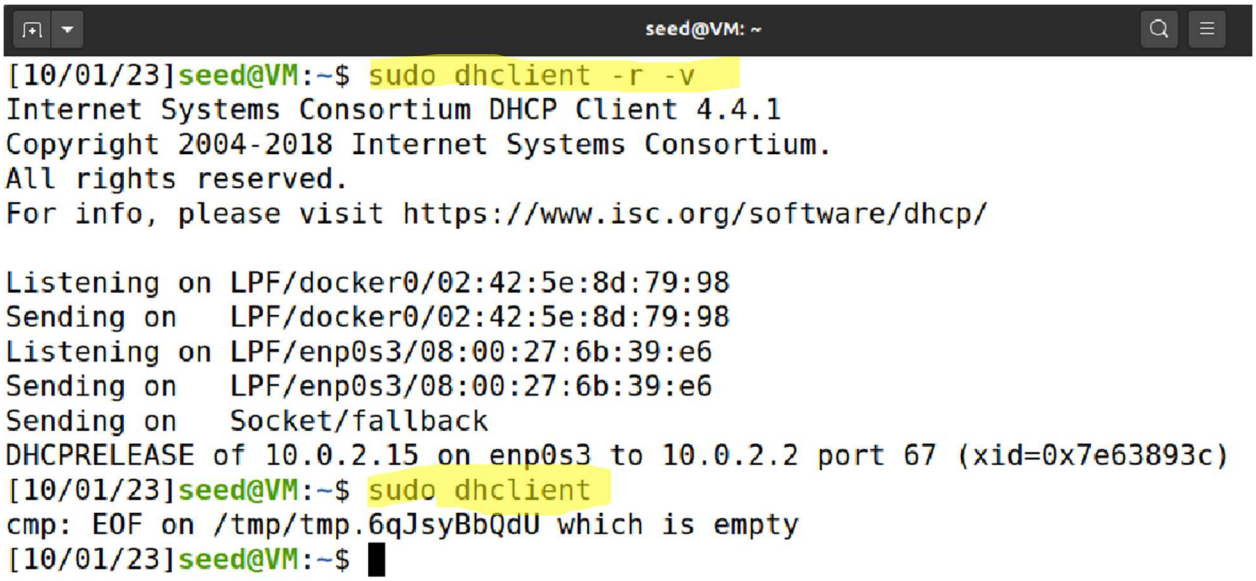
2. Start Wireshark on your VM. Next, run command **sudo dhclient -r -v** and then **sudo dhclient** and finally stop Wireshark. Command **sudo dhclient –r –v** will release your current ip address. Then **sudo dhclient** will execute the DHCP protocol. Use packets in Wireshark from executing DHCP to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

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 –



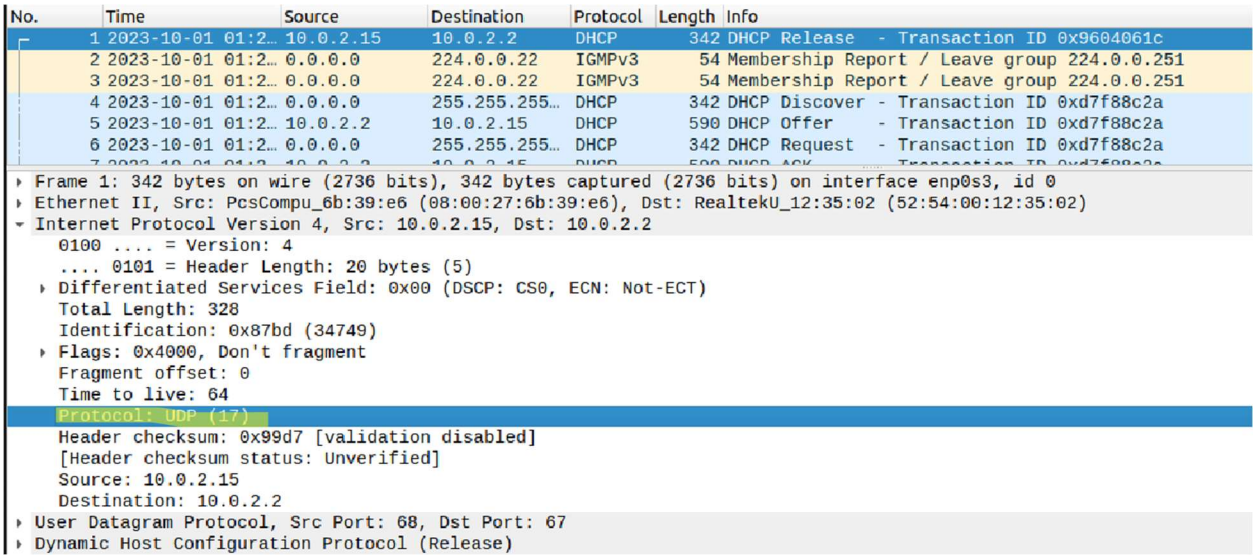
Screenshot: 10

(a) Refer Packet Number 1 from Screenshot 11

Well, the transport layer protocol for the DHCP (Dynamic Host Configuration Protocol) is UDP (User Datagram Protocol) and it can be confirmed from packet number 1.

In the packet details:

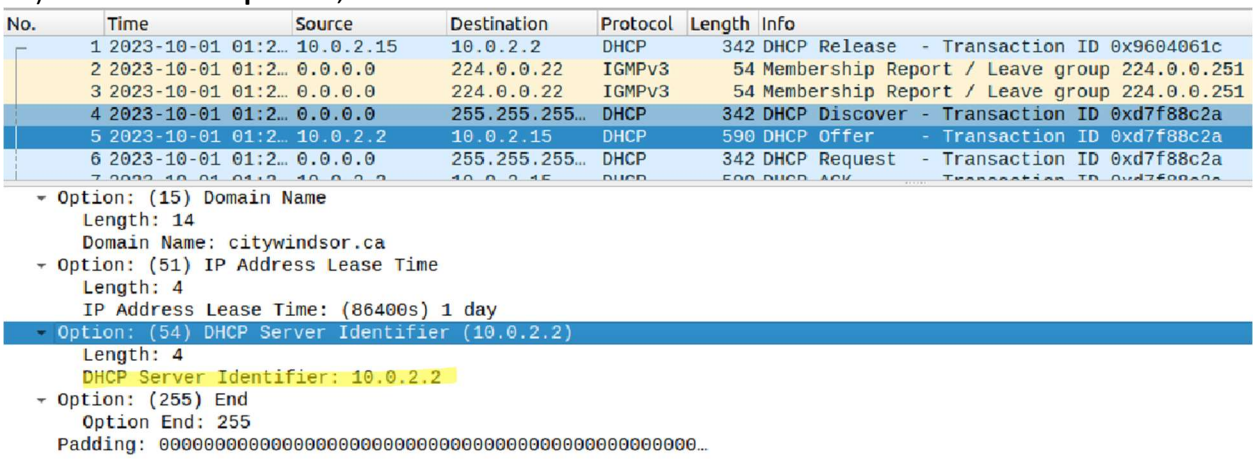
The "Protocol" field in the IP header indicates that the transport layer protocol being used is UDP, with a protocol number of 17.
Also, In the UDP section, we observed that the source port is 68, and the destination port is 67. Port 68 is typically used by DHCP clients to send requests, while port 67 is used by DHCP servers to receive these requests.
In addition to this, the reason DHCP uses UDP instead of TCP is because UDP is
➔ connectionless and
➔ lightweight transport protocol.
DHCP mainly involves the exchange of short, stateless messages between the client and server for the purpose of configuring network parameters. As DHCP messages are simple and don't require the overhead of establishing a connection and maintaining state, so as a result UDP is a more efficient choice for this protocol.



Screenshot: 11

(b)

DHCP Server IP: The DHCP server's IP address is provided in the "Server Identifier" option (Option 54). In '**DHCP offer packet'**, the DHCP Server Identifier is **10.0.2.2.**



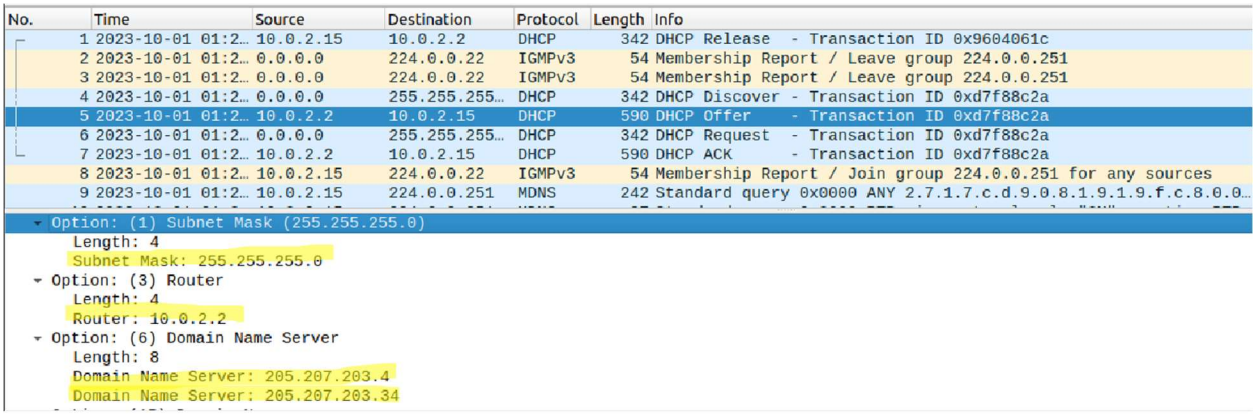
Screenshot: 12

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'**, there are two DNS server IP addresses:
DNS Server 1: 205.207.203.4

DNS Server 2: 205.207.203.34



Screenshot: 13

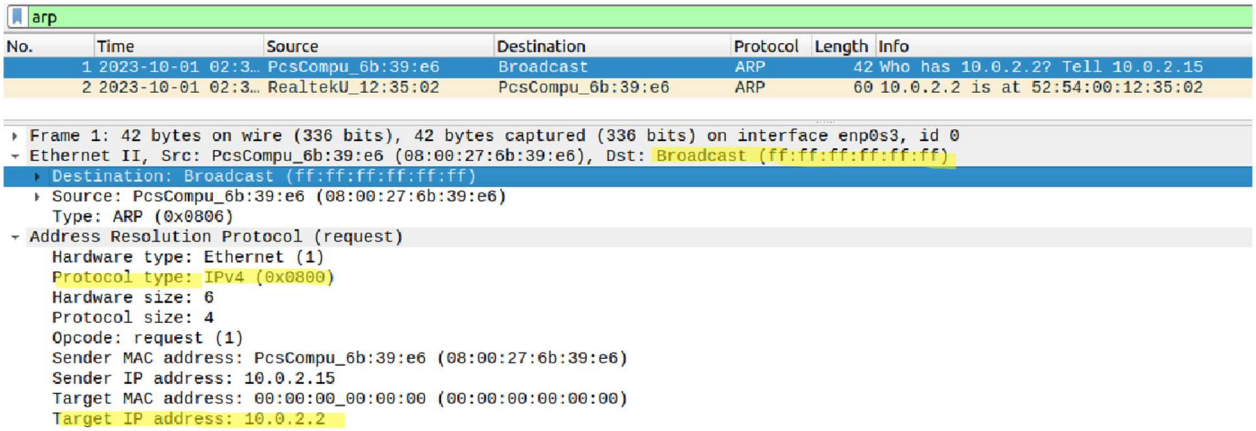
3. In this exercise, you will look in the arp protocol execution. First, run **arp** to find out the list of records in the arp table. Next, start your wireshark and run **sudo arp -d *routerIP*** to delete the record of *routerIP*. Here routerIP is the **Router IP** obtained in the previous DHCP experiment.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

b. look at the response packet for the ARP query. What is the ip address of the sender? What is its MAC address?

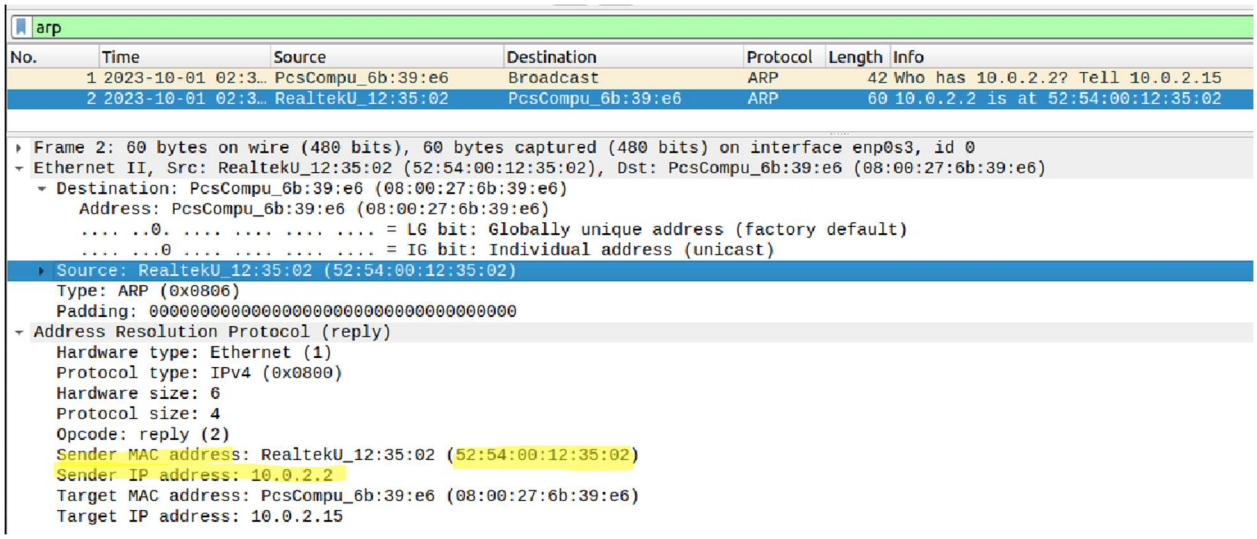
Answer –

(a) Upper layer protocol: IPv4 (0x0800)
Broadcast MAC: ff:ff:ff:ff:ff:ff
Target IP: 10.0.2.2



Screenshot: 14

(b) **IP Address of the Sender:** The sender's IP address is 10.0.2.2.
MAC Address of the Sender: The sender's MAC address is 52:54:00:12:35:02



Screenshot: 15

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.

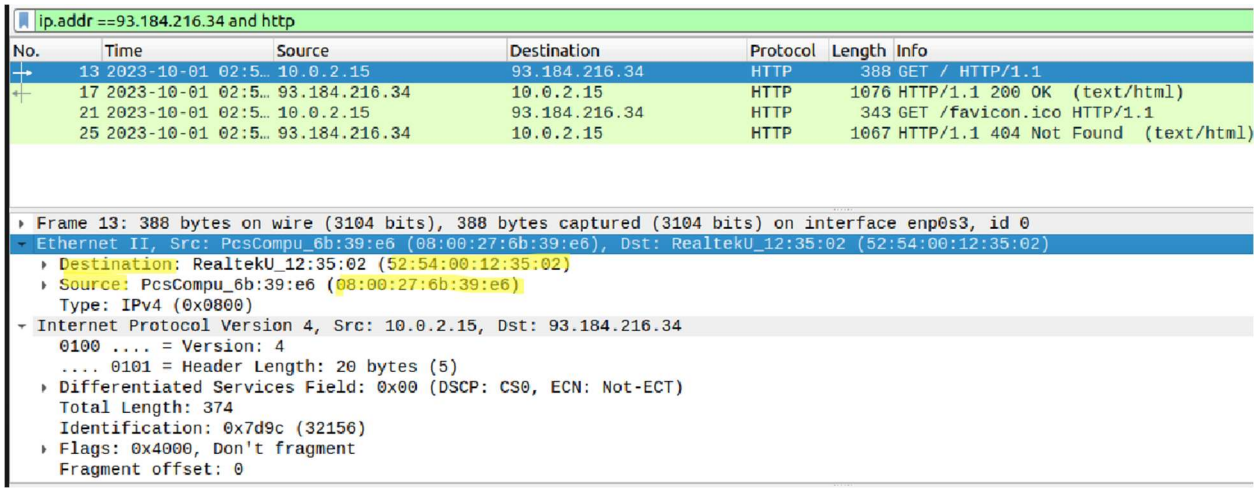
b. Does the destination MAC in **a** belong to 93.184.216.34? To find out your answer, run command **arp** to check the arp table of your VM. Is the destination MAC in **a** listed here? If yes, confirm that this MAC does not belong to 93.184.216.34 and instead belong to your router.

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 –

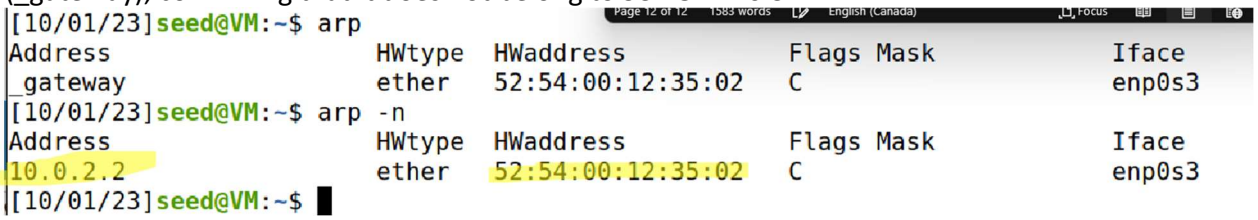
(a)

Source MAC: 08:00:27:6b:39:e6 (Your VM)
Destination MAC: 52:54:00:12:35:02 (RealtekU_12:35:02, the target system)



Screenshot: 16

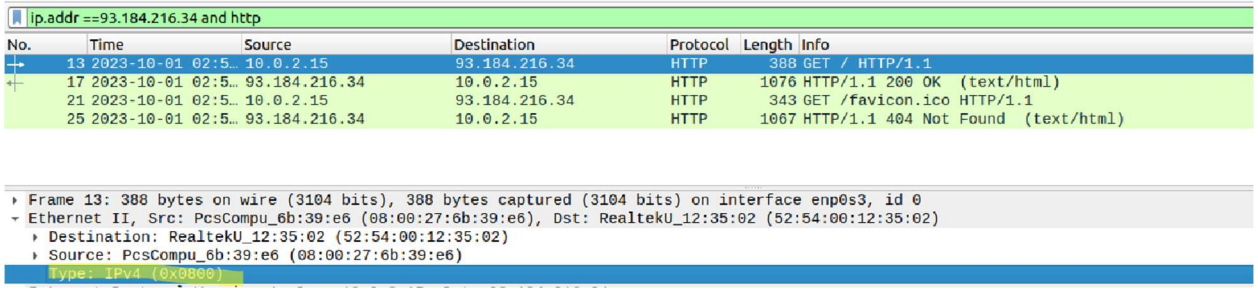
(b) Yes, the destination MAC address in packet Number 13 (52:54:00:12:35:02) – With Reference to Screenshot 16 - is listed in the ARP table output. However, it is associated with the IP address 10.0.2.2, not 93.184.216.34. This MAC address belongs to our router (_gateway), confirming that it does not belong to 93.184.216.34.



Screenshot: 17

(c) 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.

Source IP : 10.0.2.15 , **Destination IP:** 93.184.216.34



Screenshot: 18