Project – Protocol Analysis (DNS, TCP, HTTP)

Objective

In this exercise we analyze the layered structure of network protocols using a web browsing example. We examine the header structure of the PDUs at the data link, IP, transport, and application layers. In particular we observe how addresses and port numbers work together to enable end-to-end applications.

Protocols Examined

- Ethernet and IP addressing
- DNS Query and Response
- TCP three-way handshake, sequence and ACK numbering
- HTTP GET and Response messages

Prerequisite

The students should know how to capture packets using WireShark or Ethereal.

Procedure

1. Start the protocol analyzer..
2. Start a web browser and enter the URL of a website of your choice but do *not* press ENTER. **Using this website "http://www.testingmcafeesites.com/testcat_ac.html"**
3. Start packet capture.
4. Access the website by pressing ENTER in the browser web page. 5. Once page is loaded, stop capture. Save the capture file.
6. Save the displayed web page for later reference.

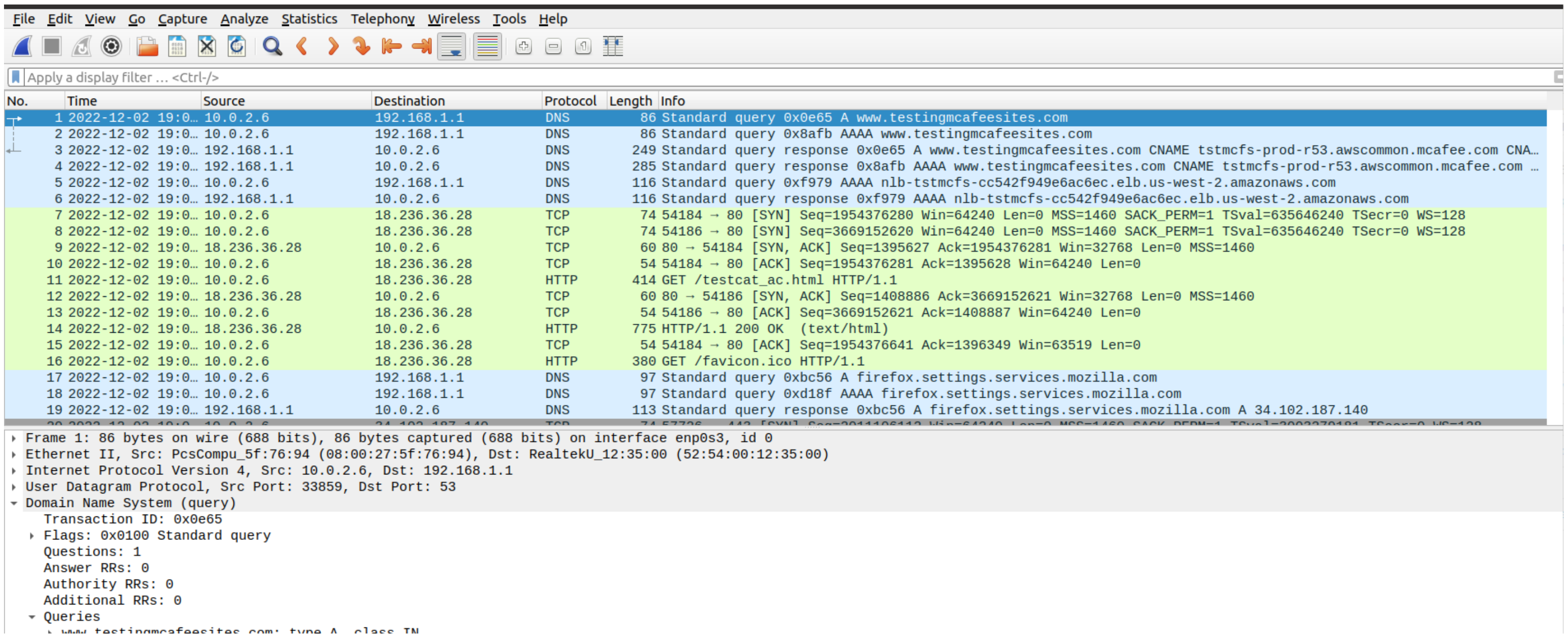
Protocol Analysis Questions

To answer the following questions, start Ethereal and open the packet capture file created above.

1. Protocols Captured - Gianna Galard

• Examine the protocol column in the top pane of the window. Confirm that you have captured DNS, TCP, and HTTP packets.

- **Confirmed**


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

```
No.   Time              Source         Destination    Protocol Length Info
    1 2022-12-02 19:0… 10.0.2.6       192.168.1.1    DNS          86 Standard query 0x0e65 A www.testingmcafeesites.com
    2 2022-12-02 19:0… 10.0.2.6       192.168.1.1    DNS          86 Standard query 0x8afb AAAA www.testingmcafeesites.com
    3 2022-12-02 19:0… 192.168.1.1    10.0.2.6       DNS         249 Standard query response 0x0e65 A www.testingmcafeesites.com CNAME tstmcfs-prod-r53.awscommon.mcafee.com CNA…
    4 2022-12-02 19:0… 192.168.1.1    10.0.2.6       DNS         285 Standard query response 0x8afb AAAA www.testingmcafeesites.com CNAME tstmcfs-prod-r53.awscommon.mcafee.com …
    5 2022-12-02 19:0… 10.0.2.6       192.168.1.1    DNS         116 Standard query 0xf979 AAAA nlb-tstmcfs-cc542f949e6ac6ec.elb.us-west-2.amazonaws.com
    6 2022-12-02 19:0… 192.168.1.1    10.0.2.6       DNS         116 Standard query response 0xf979 AAAA nlb-tstmcfs-cc542f949e6ac6ec.elb.us-west-2.amazonaws.com
    7 2022-12-02 19:0… 10.0.2.6       18.236.36.28   TCP          74 54184 → 80 [SYN] Seq=1954376280 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=635646240 TSecr=0 WS=128
    8 2022-12-02 19:0… 10.0.2.6       18.236.36.28   TCP          74 54186 → 80 [SYN] Seq=3669152620 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=635646240 TSecr=0 WS=128
    9 2022-12-02 19:0… 18.236.36.28   10.0.2.6       TCP          60 80 → 54184 [SYN, ACK] Seq=1395627 Ack=1954376281 Win=32768 Len=0 MSS=1460
   10 2022-12-02 19:0… 10.0.2.6       18.236.36.28   TCP          54 54184 → 80 [ACK] Seq=1954376281 Ack=1395628 Win=64240 Len=0
   11 2022-12-02 19:0… 10.0.2.6       18.236.36.28   HTTP        414 GET /testcat_ac.html HTTP/1.1
   12 2022-12-02 19:0… 18.236.36.28   10.0.2.6       TCP          60 80 → 54186 [SYN, ACK] Seq=1408886 Ack=3669152621 Win=32768 Len=0 MSS=1460
   13 2022-12-02 19:0… 10.0.2.6       18.236.36.28   TCP          54 54186 → 80 [ACK] Seq=3669152621 Ack=1408887 Win=64240 Len=0
   14 2022-12-02 19:0… 18.236.36.28   10.0.2.6       HTTP        775 HTTP/1.1 200 OK  (text/html)
   15 2022-12-02 19:0… 10.0.2.6       18.236.36.28   TCP          54 54184 → 80 [ACK] Seq=1954376641 Ack=1396349 Win=63519 Len=0
   16 2022-12-02 19:0… 10.0.2.6       18.236.36.28   HTTP        380 GET /favicon.ico HTTP/1.1
   17 2022-12-02 19:0… 10.0.2.6       192.168.1.1    DNS          97 Standard query 0xbc56 A firefox.settings.services.mozilla.com
   18 2022-12-02 19:0… 10.0.2.6       192.168.1.1    DNS          97 Standard query 0xd18f AAAA firefox.settings.services.mozilla.com
   19 2022-12-02 19:0… 192.168.1.1    10.0.2.6       DNS         113 Standard query response 0xbc56 A firefox.settings.services.mozilla.com A 34.102.187.140
```

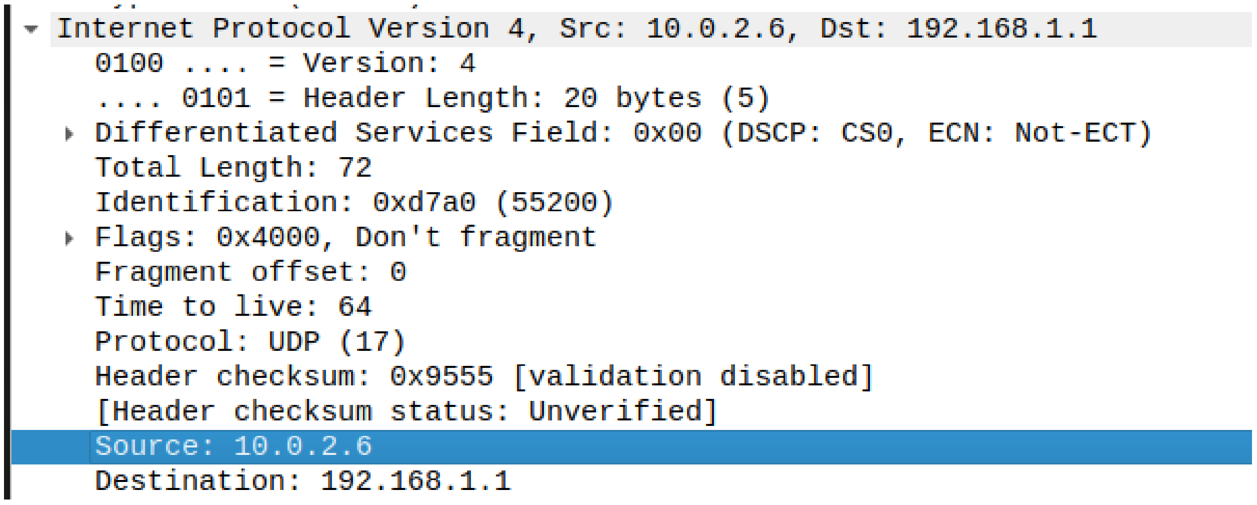
```
› Frame 1: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface enp0s3, id 0
› Ethernet II, Src: PcsCompu_5f:76:94 (08:00:27:5f:76:94), Dst: RealtekU_12:35:00 (52:54:00:12:35:00)
› Internet Protocol Version 4, Src: 10.0.2.6, Dst: 192.168.1.1
› User Datagram Protocol, Src Port: 33859, Dst Port: 53
⌄ Domain Name System (query)
    Transaction ID: 0x0e65
  › Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  ⌄ Queries
```

2. Ethernet frame, IP packet, and UDP datagram - Gianna Galard

• Examine the frame for the first DNS packet sent by the client.

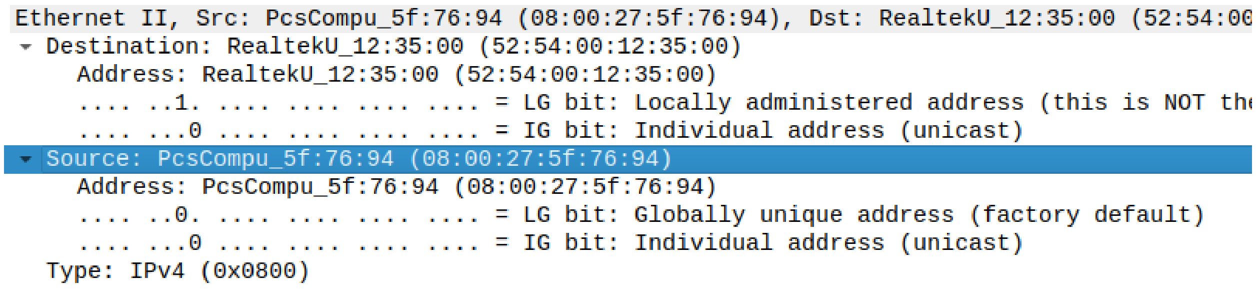
a. Identify the Ethernet and IP address of the client.

- **IP:**



```
▾ Internet Protocol Version 4, Src: 10.0.2.6, Dst: 192.168.1.1
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 72
    Identification: 0xd7a0 (55200)
  ▸ Flags: 0x4000, Don't fragment
    Fragment offset: 0
    Time to live: 64
    Protocol: UDP (17)
    Header checksum: 0x9555 [validation disabled]
    [Header checksum status: Unverified]
    Source: 10.0.2.6
    Destination: 192.168.1.1
```

- **Ethernet:**

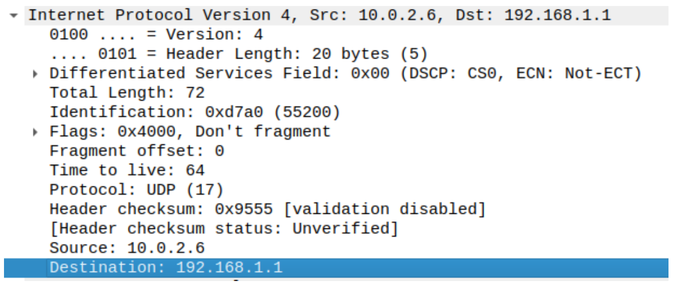


```
Ethernet II, Src: PcsCompu_5f:76:94 (08:00:27:5f:76:94), Dst: RealtekU_12:35:00 (52:54:0
▾ Destination: RealtekU_12:35:00 (52:54:00:12:35:00)
    Address: RealtekU_12:35:00 (52:54:00:12:35:00)
    .... ..1. .... .... .... .... = LG bit: Locally administered address (this is NOT th
    .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
▾ Source: PcsCompu_5f:76:94 (08:00:27:5f:76:94)
    Address: PcsCompu_5f:76:94 (08:00:27:5f:76:94)
    .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
    .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
  Type: IPv4 (0x0800)
```

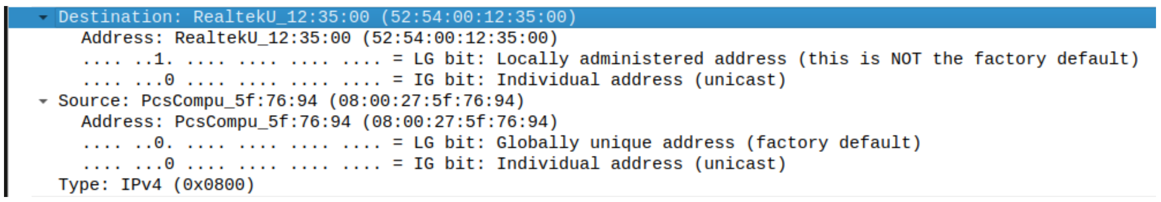
b. What is the content of the type field in the Ethernet frame?

- **Type: IPv4 (0x0800)**

c. What are the destination Ethernet and IP addresses and to which machines do these addresses correspond? Explain how this depends on how your machine is connected to the Internet.



```
▾ Internet Protocol Version 4, Src: 10.0.2.6, Dst: 192.168.1.1
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 72
    Identification: 0xd7a0 (55200)
  ▸ Flags: 0x4000, Don't fragment
    Fragment offset: 0
    Time to live: 64
    Protocol: UDP (17)
    Header checksum: 0x9555 [validation disabled]
    [Header checksum status: Unverified]
    Source: 10.0.2.6
    Destination: 192.168.1.1
```



```
▾ Destination: RealtekU_12:35:00 (52:54:00:12:35:00)
    Address: RealtekU_12:35:00 (52:54:00:12:35:00)
    .... ..1. .... .... .... .... = LG bit: Locally administered address (this is NOT the factory default)
    .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
▾ Source: PcsCompu_5f:76:94 (08:00:27:5f:76:94)
    Address: PcsCompu_5f:76:94 (08:00:27:5f:76:94)
    .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
    .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
  Type: IPv4 (0x0800)
```

- **The destination IP and Ethernet refer to the website that we are sending the packets to, and they are the unique address allocation of the devices, which is important since it's the only way a hardware can connect to a network.**

• Examine the IP header for the first DNS packet sent by the client.

a. What is the header length? What is the total packet length?

- **86**

b. Identify the protocol type field. What is the number and type of the protocol in the payload?

- **UPD (17)**

• Examine the UDP header of the first DNS packet sent by the client.

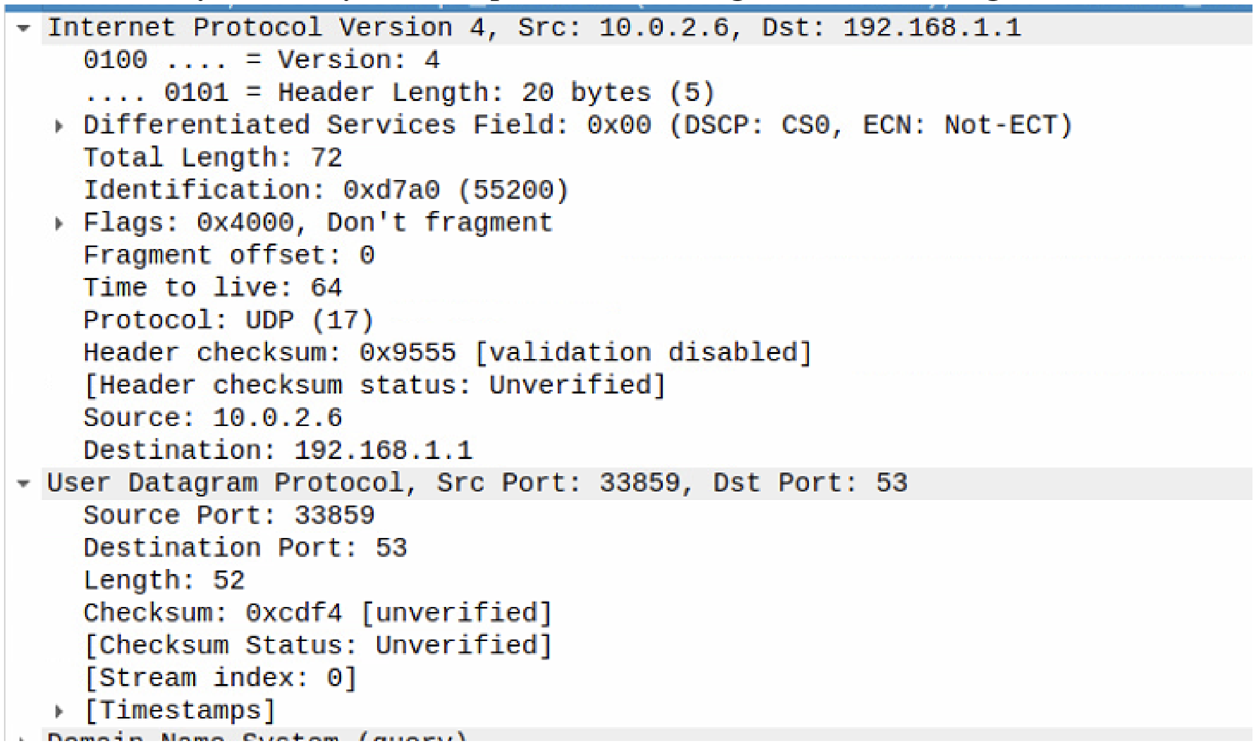
a. Identify the client ephemeral port number and the server well-known port number. What type of application layer protocol is in the payload?

- **Client port num: 33859, Server port num: 53, it's using TCP, UDP**

b. Confirm that the length field in the UDP header is consistent with the IP header length information.

- **The total length for the IP is (72 - 20 = 52) 52, and the length of the UDP is 52. They will always match up after subtracting the IP header length.**

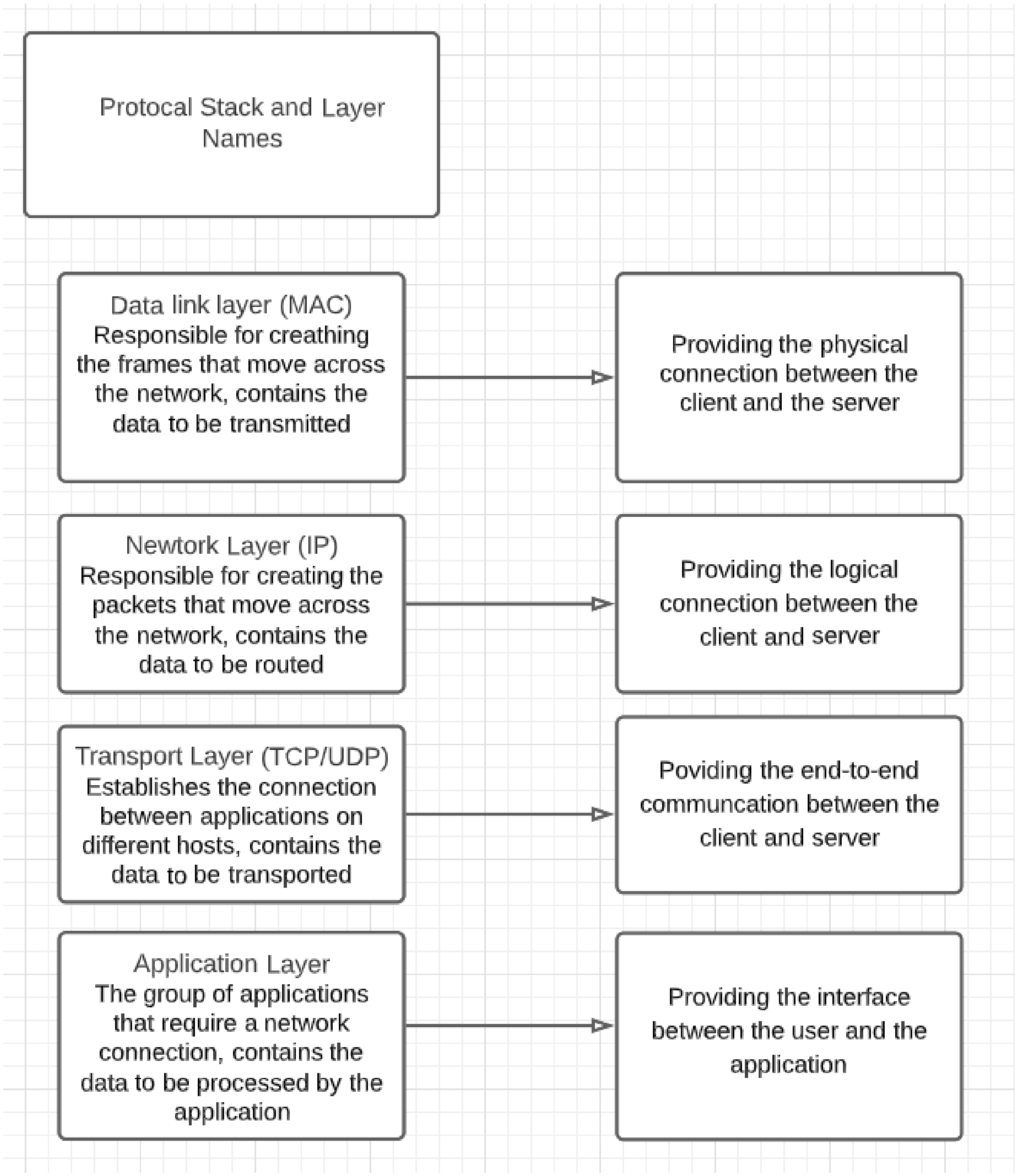
-



```
▾ Internet Protocol Version 4, Src: 10.0.2.6, Dst: 192.168.1.1
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 72
    Identification: 0xd7a0 (55200)
  ▸ Flags: 0x4000, Don't fragment
    Fragment offset: 0
    Time to live: 64
    Protocol: UDP (17)
    Header checksum: 0x9555 [validation disabled]
    [Header checksum status: Unverified]
    Source: 10.0.2.6
    Destination: 192.168.1.1
▾ User Datagram Protocol, Src Port: 33859, Dst Port: 53
    Source Port: 33859
    Destination Port: 53
    Length: 52
    Checksum: 0xcdf4 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 0]
  ▸ [Timestamps]
```

• Sketch the protocol stack from the data link layer up to the application layer at the client and server sides and explain how the contents in the various PDUs enable end-to-end communication between application layer processes.

-

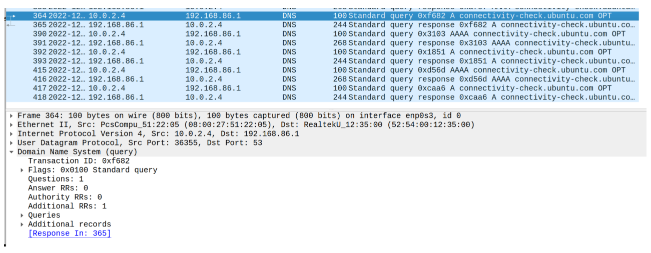

Protocal Stack and Layer Names
Data link layer (MAC) Responsible for creathing the frames that move across the network, contains the data to be transmitted
Providing the physical connection between the client and the server
Newtork Layer (IP) Responsible for creating the packets that move across the network, contains the data to be routed
Providing the logical connection between the client and server
Transport Layer (TCP/UDP) Establishes the connection between applications on different hosts, contains the data to be transported
Poviding the end-to-end communcation between the client and server
Application Layer The group of applications that require a network connection, contains the data to be processed by the application
Providing the interface between the user and the application

3. *DNS*

• Examine the DNS query message in the DNS packet sent by the client.

a. What field indicates whether the message is a query or a response?

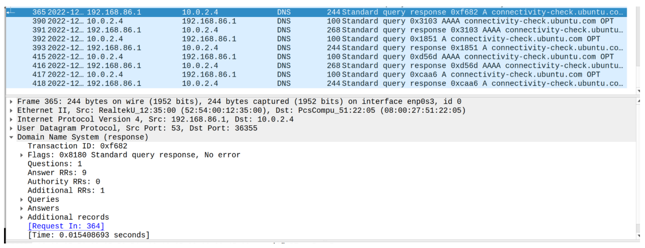
- The Flag/Parameters Field of a DNS header would identify if it is a query or a presonse
- Below shows a query



```
364 2022-12… 10.0.2.4       192.168.86.1   DNS   100 Standard query 0xf682 A connectivity-check.ubuntu.com OPT
365 2022-12… 192.168.86.1   10.0.2.4       DNS   244 Standard query response 0xf682 A connectivity-check.ubuntu.co…
390 2022-12… 10.0.2.4       192.168.86.1   DNS   100 Standard query 0x3103 AAAA connectivity-check.ubuntu.com OPT
391 2022-12… 192.168.86.1   10.0.2.4       DNS   268 Standard query response 0x3103 AAAA connectivity-check.ubuntu…
392 2022-12… 10.0.2.4       192.168.86.1   DNS   100 Standard query 0x1851 A connectivity-check.ubuntu.com OPT
393 2022-12… 192.168.86.1   10.0.2.4       DNS   244 Standard query response 0x1851 A connectivity-check.ubuntu.co…
415 2022-12… 10.0.2.4       192.168.86.1   DNS   100 Standard query 0xd56d AAAA connectivity-check.ubuntu.com OPT
416 2022-12… 192.168.86.1   10.0.2.4       DNS   268 Standard query response 0xd56d AAAA connectivity-check.ubuntu…
417 2022-12… 10.0.2.4       192.168.86.1   DNS   100 Standard query 0xcaa6 A connectivity-check.ubuntu.com OPT
418 2022-12… 192.168.86.1   10.0.2.4       DNS   244 Standard query response 0xcaa6 A connectivity-check.ubuntu.co…

▸ Frame 364: 100 bytes on wire (800 bits), 100 bytes captured (800 bits) on interface enp0s3, id 0
▸ Ethernet II, Src: PcsCompu_51:22:05 (08:00:27:51:22:05), Dst: RealtekU_12:35:00 (52:54:00:12:35:00)
▸ Internet Protocol Version 4, Src: 10.0.2.4, Dst: 192.168.86.1
▸ User Datagram Protocol, Src Port: 36355, Dst Port: 53
▾ Domain Name System (query)
    Transaction ID: 0xf682
  ▸ Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 1
  ▸ Queries
  ▸ Additional records
    [Response In: 365]
```

- Below shows a response



```
365 2022-12… 192.168.86.1   10.0.2.4       DNS   244 Standard query response 0xf682 A connectivity-check.ubuntu.co…
390 2022-12… 10.0.2.4       192.168.86.1   DNS   100 Standard query 0x3103 AAAA connectivity-check.ubuntu.com OPT
391 2022-12… 192.168.86.1   10.0.2.4       DNS   268 Standard query response 0x3103 AAAA connectivity-check.ubuntu…
392 2022-12… 10.0.2.4       192.168.86.1   DNS   100 Standard query 0x1851 A connectivity-check.ubuntu.com OPT
393 2022-12… 192.168.86.1   10.0.2.4       DNS   244 Standard query response 0x1851 A connectivity-check.ubuntu.co…
415 2022-12… 10.0.2.4       192.168.86.1   DNS   100 Standard query 0xd56d AAAA connectivity-check.ubuntu.com OPT
416 2022-12… 192.168.86.1   10.0.2.4       DNS   268 Standard query response 0xd56d AAAA connectivity-check.ubuntu…
417 2022-12… 10.0.2.4       192.168.86.1   DNS   100 Standard query 0xcaa6 A connectivity-check.ubuntu.com OPT
418 2022-12… 192.168.86.1   10.0.2.4       DNS   244 Standard query response 0xcaa6 A connectivity-check.ubuntu.co…

▸ Frame 365: 244 bytes on wire (1952 bits), 244 bytes captured (1952 bits) on interface enp0s3, id 0
▸ Ethernet II, Src: RealtekU_12:35:00 (52:54:00:12:35:00), Dst: PcsCompu_51:22:05 (08:00:27:51:22:05)
▸ Internet Protocol Version 4, Src: 192.168.86.1, Dst: 10.0.2.4
▸ User Datagram Protocol, Src Port: 53, Dst Port: 36355
▾ Domain Name System (response)
    Transaction ID: 0xf682
  ▸ Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 9
    Authority RRs: 0
    Additional RRs: 1
  ▸ Queries
  ▸ Answers
  ▸ Additional records
    [Request In: 364]
    [Time: 0.015408693 seconds]
```

b. What information is carried in the body of the query?

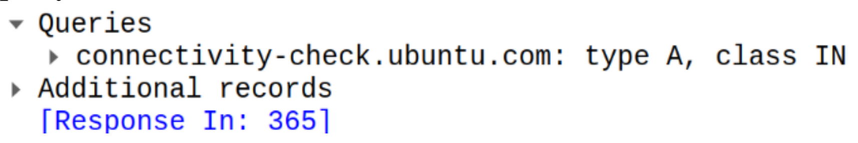
- **The DNS will carry answers to the questions asked by the DNS servers.**

c. What is the query transaction ID?

- **The transaction ID is a random number generated by the nameserver initiating the query.**

d. Identify the fields that carry the type and class of the query.

- **In WireShark, if you click on a DNS response, then Somain Name SYsteem, then Queries, you can see the type and class of the query**
-

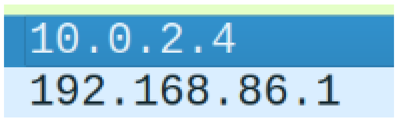


```
▾ Queries
  ▸ connectivity-check.ubuntu.com: type A, class IN
▸ Additional records
  [Response In: 365]
```

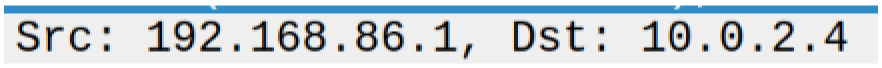
• Now consider the packet that carries the DNS response to the above query.

a. What should the Ethernet and IP addresses for this packet be? Verify that these addresses are as expected.

- **Our VM is located in 10.0.2.4 and the destinations address is 192.168.86.1**
-



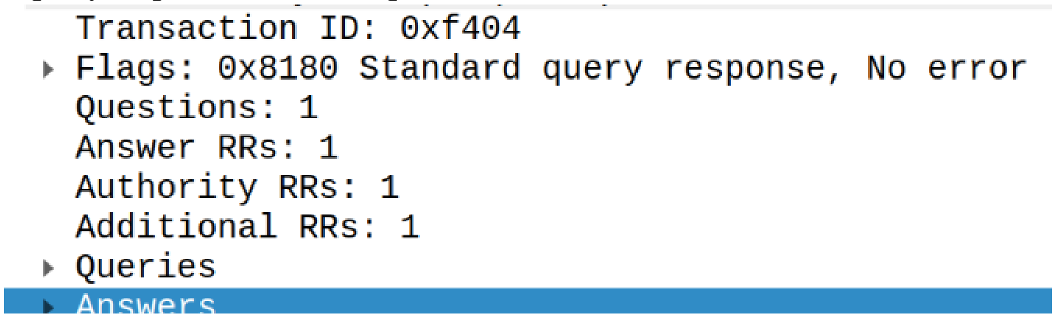
```
10.0.2.4
192.168.86.1
```

- 
```
Src: 192.168.86.1, Dst: 10.0.2.4
```

b. What is the size of the IP packet and UDP datagram that carry the response? Is it longer than the query?

- **260 bytes**

c. Confirm that the transaction ID in the response message is correct. d. How many answers are provided in the response message? Compare the answers and their time-to-live values.

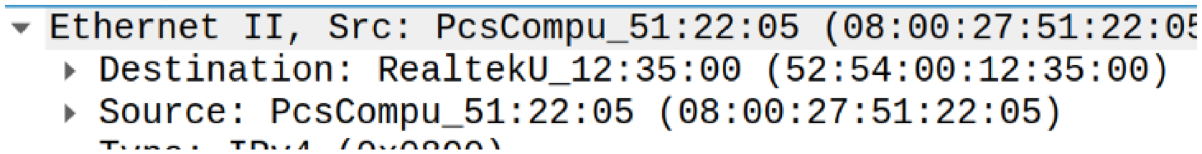
- **The below image shows a transaction ID of 0xf404 with an answer of 1 fron a query response that has 1 question**
- 
```
  Transaction ID: 0xf404
▸ Flags: 0x8180 Standard query response, No error
  Questions: 1
  Answer RRs: 1
  Authority RRs: 1
  Additional RRs: 1
▸ Queries
▸ Answers
```

4. TCP three-way handshake

• Identify the frame that carries the first TCP segment in the three-way handshake that sets up the connection between the http client and server.

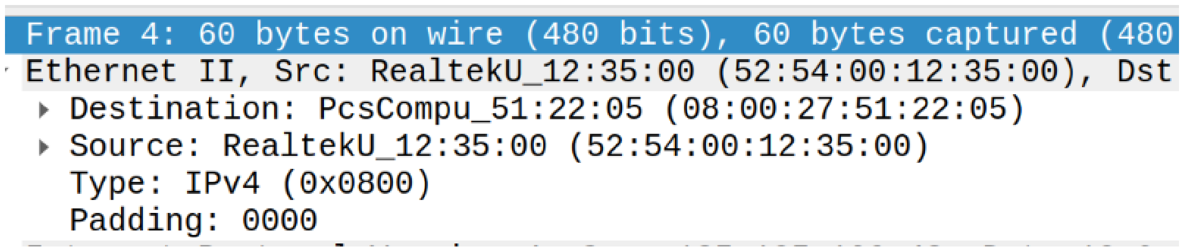
a. What source Ethernet and IP addresses do you expect in this segment? What protocol and type fields do you expect in the first segment? Confirm that these addresses are as expected.

- **I am expecting a request for a connection from my computer to the server. This is all expected under SYN of a TCP protocol. I confirmed this with the [SYN, ACK]**



```
▾ Ethernet II, Src: PcsCompu_51:22:05 (08:00:27:51:22:05
    ▸ Destination: RealtekU_12:35:00 (52:54:00:12:35:00)
    ▸ Source: PcsCompu_51:22:05 (08:00:27:51:22:05)
```

-



```
Frame 4: 60 bytes on wire (480 bits), 60 bytes captured (480
Ethernet II, Src: RealtekU_12:35:00 (52:54:00:12:35:00), Dst
  ▸ Destination: PcsCompu_51:22:05 (08:00:27:51:22:05)
  ▸ Source: RealtekU_12:35:00 (52:54:00:12:35:00)
    Type: IPv4 (0x0800)
    Padding: 0000
```

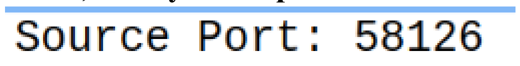
-

b. Explain the values in the destination Ethernet and IP addresses in the first segment? To what machine(s) do these addresses correspond?

- **The source is the IP of our computer while the destination is the servers of the URL. we will need to first request for a connection request so it will always be from our to their side.**

c. Identify the ephemeral port number used by the client and confirm that the well-known port number is the correct value for HTTP.

- **58126, is a dynamic port that isnt widely used**



```
Source Port: 58126
```

-

d. What is the length of the TCP segment?

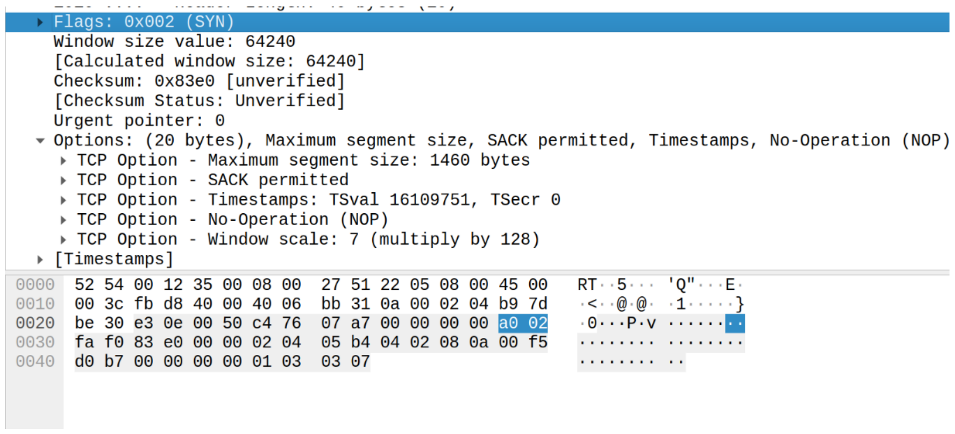
- **64240**

e. What is the initial sequence number for the segments from the client to the server? What is the initial window size? What is the maximum segment size?

- **The initial sequence number is 3296069543. The window size is 64240**

f. Find the hex character that contains the SYN flag bit.

- Under Transmission OCntrol Protocal, you can find out there the SYN flag but is and by clicking on that you can see where it is located on the chart



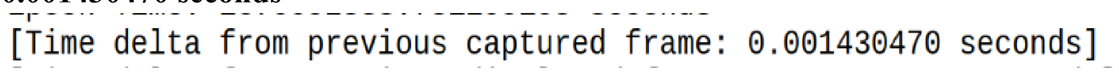
```
▸ Flags: 0x002 (SYN)
  Window size value: 64240
  [Calculated window size: 64240]
  Checksum: 0x83e0 [unverified]
  [Checksum Status: Unverified]
  Urgent pointer: 0
▾ Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP)
  ▸ TCP Option - Maximum segment size: 1460 bytes
  ▸ TCP Option - SACK permitted
  ▸ TCP Option - Timestamps: TSval 16109751, TSecr 0
  ▸ TCP Option - No-Operation (NOP)
  ▸ TCP Option - Window scale: 7 (multiply by 128)
▸ [Timestamps]
0000  52 54 00 12 35 00 08 00  27 51 22 05 08 00 45 00   RT··5··· 'Q"···E·
0010  00 3c fb d8 40 00 40 06  bb 31 0a 00 02 04 b9 7d   ·<··@·@· ·1·····}
0020  be 30 e3 0e 00 50 c4 76  07 a7 00 00 00 00 a0 02   ·0···P·v ······ ··
0030  fa f0 83 e0 00 00 02 04  05 b4 04 02 08 0a 00 f5   ········ ········
0040  d0 b7 00 00 00 00 01 03  03 07                     ········ ··
```

-

• Identify the frame that carries the second segment in the three-way handshake.

a. How much time elapsed between the capture of the first and second segments?

- **0.001430470 seconds**



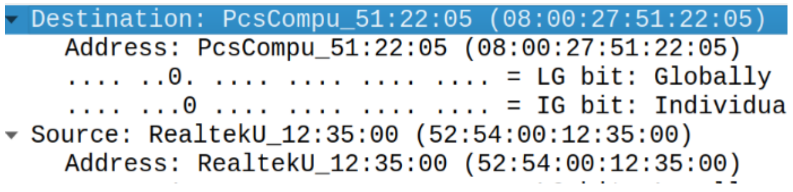
```
[Time delta from previous captured frame: 0.001430470 seconds]
```

-

b. Before examining the captured packets, specify the values for the following fields in this frame:

• Source and destination addresses and type field in Ethernet frame.

- **Source: 52:54:00:12:35:00**
- **Destination: 08:00:27:51:22:05**

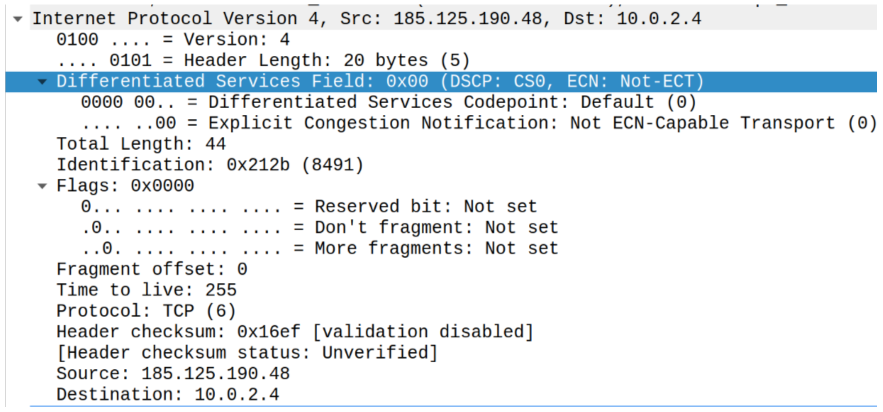


```
▾ Destination: PcsCompu_51:22:05 (08:00:27:51:22:05)
    Address: PcsCompu_51:22:05 (08:00:27:51:22:05)
    .... ..0. .... .... .... .... = LG bit: Globally
    .... ...0 .... .... .... .... = IG bit: Individua
▾ Source: RealtekU_12:35:00 (52:54:00:12:35:00)
    Address: RealtekU_12:35:00 (52:54:00:12:35:00)
```

-

• Source and destination IP addresses and port numbers in IP packet.

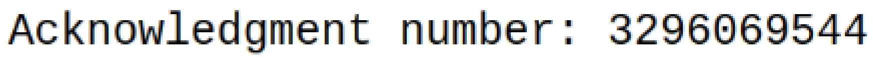
- **Source: 185.125.190.48**
- **Destination: 10.0.2.4**



```
▾ Internet Protocol Version 4, Src: 185.125.190.48, Dst: 10.0.2.4
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▾ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
      0000 00.. = Differentiated Services Codepoint: Default (0)
      .... ..00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
    Total Length: 44
    Identification: 0x212b (8491)
  ▾ Flags: 0x0000
      0... .... .... .... = Reserved bit: Not set
      .0.. .... .... .... = Don't fragment: Not set
      ..0. .... .... .... = More fragments: Not set
    Fragment offset: 0
    Time to live: 255
    Protocol: TCP (6)
    Header checksum: 0x16ef [validation disabled]
    [Header checksum status: Unverified]
    Source: 185.125.190.48
    Destination: 10.0.2.4
```

-

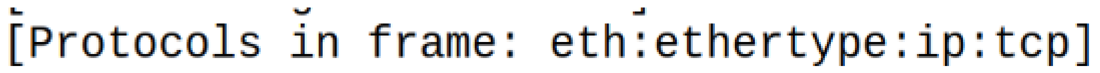
• Acknowledgement number in TCP segment.

- 
```
Acknowledgment number: 3296069544
```

• Values of flag bits.

- **0x012**

• Confirm that the frame contains the expected values.

- **Confirmed**
- 
```
[Protocols in frame: eth:ethertype:ip:tcp]
```

c. What is the length of the TCP segment?

- **32768**

d. What is the initial sequence number for the connection from the server to the client? What is the maximum segment size?

- **386658**

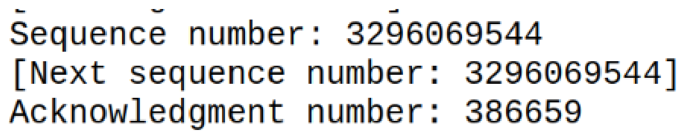
• Identify the frame that carries the last segment in the three-way handshake.

a. How much time elapsed between the capture of the second and last segment? Compare to the elapsed time between the first and second segments and explain the difference.

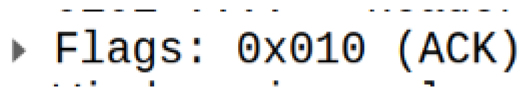
- **0.1523458 seconds between second and last segment**
- **0.1345039 seconds between first and second segment**
- **Length is longer fron first to second compared to second to last**

b. Specify the following values in the TCP segment:

• Acknowledgement and sequence numbers.

- 
```
Sequence number: 3296069544
[Next sequence number: 3296069544]
Acknowledgment number: 386659
```

• Flag bits and window size.

- **0x010**
- 
```
▸ Flags: 0x010 (ACK)
```

• Confirm that the segment contains the expected values

- **Confirmed**

c. What is the length of the third TCP segment?

- **64240**

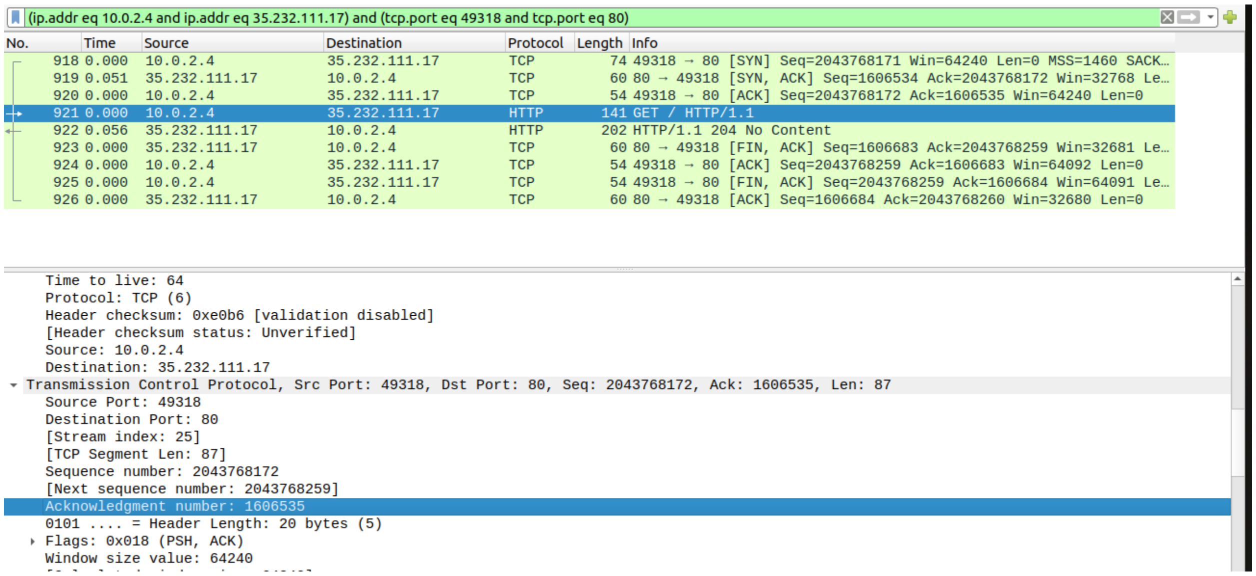
Link use for wire shark: https://en.wikipedia.org/wiki/Main_Page

5. HTTP GET - Unaiza Nizami

• Identify the frame that carries the HTTP "GET" message.

a. Confirm that the sequence and acknowledgement values in the TCP header are as expected.

As we can see below our acknowledgement value in the tcp is the same as the http get TCP header.



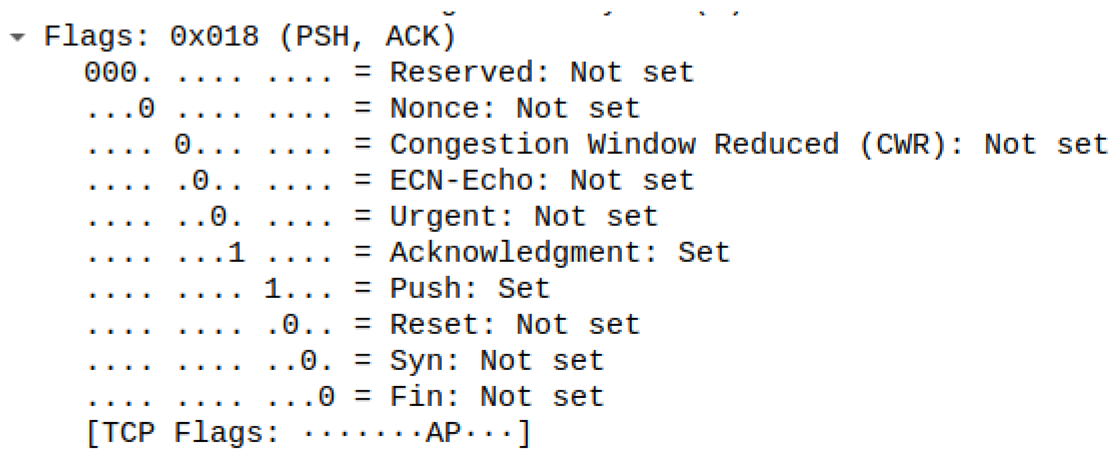
```
(ip.addr eq 10.0.2.4 and ip.addr eq 35.232.111.17) and (tcp.port eq 49318 and tcp.port eq 80)

No.  Time   Source          Destination     Protocol Length Info
918  0.000  10.0.2.4        35.232.111.17   TCP      74  49318 → 80 [SYN] Seq=2043768171 Win=64240 Len=0 MSS=1460 SACK…
919  0.051  35.232.111.17   10.0.2.4        TCP      60  80 → 49318 [SYN, ACK] Seq=1606534 Ack=2043768172 Win=32768 Le…
920  0.000  10.0.2.4        35.232.111.17   TCP      54  49318 → 80 [ACK] Seq=2043768172 Ack=1606535 Win=64240 Len=0
921  0.000  10.0.2.4        35.232.111.17   HTTP     141 GET / HTTP/1.1
922  0.056  35.232.111.17   10.0.2.4        HTTP     202 HTTP/1.1 204 No Content
923  0.000  35.232.111.17   10.0.2.4        TCP      60  80 → 49318 [FIN, ACK] Seq=1606683 Ack=2043768259 Win=32681 Le…
924  0.000  10.0.2.4        35.232.111.17   TCP      54  49318 → 80 [ACK] Seq=2043768259 Ack=1606683 Win=64092 Len=0
925  0.000  10.0.2.4        35.232.111.17   TCP      54  49318 → 80 [FIN, ACK] Seq=2043768259 Ack=1606684 Win=64091 Le…
926  0.000  35.232.111.17   10.0.2.4        TCP      60  80 → 49318 [ACK] Seq=1606684 Ack=2043768260 Win=32680 Len=0
```

```
    Time to live: 64
    Protocol: TCP (6)
    Header checksum: 0xe0b6 [validation disabled]
    [Header checksum status: Unverified]
    Source: 10.0.2.4
    Destination: 35.232.111.17
▾ Transmission Control Protocol, Src Port: 49318, Dst Port: 80, Seq: 2043768172, Ack: 1606535, Len: 87
    Source Port: 49318
    Destination Port: 80
    [Stream index: 25]
    [TCP Segment Len: 87]
    Sequence number: 2043768172
    [Next sequence number: 2043768259]
    Acknowledgment number: 1606535
    0101 .... = Header Length: 20 bytes (5)
  ▸ Flags: 0x018 (PSH, ACK)
    Window size value: 64240
```

b. Examine the flag bits in the TCP header. Can you explain why the two flag bits are set?

In the TCP header the flags are used to indicate a particular state of connection or to provide some information like how to handle a particular connection or trouble shooting. In the flags a value of 1 means it is set.



```
▾ Flags: 0x018 (PSH, ACK)
    000. .... .... = Reserved: Not set
    ...0 .... .... = Nonce: Not set
    .... 0... .... = Congestion Window Reduced (CWR): Not set
    .... .0.. .... = ECN-Echo: Not set
    .... ..0. .... = Urgent: Not set
    .... ...1 .... = Acknowledgment: Set
    .... .... 1... = Push: Set
    .... .... .0.. = Reset: Not set
    .... .... ..0. = Syn: Not set
    .... .... ...0 = Fin: Not set
    [TCP Flags: ·······AP···]
```

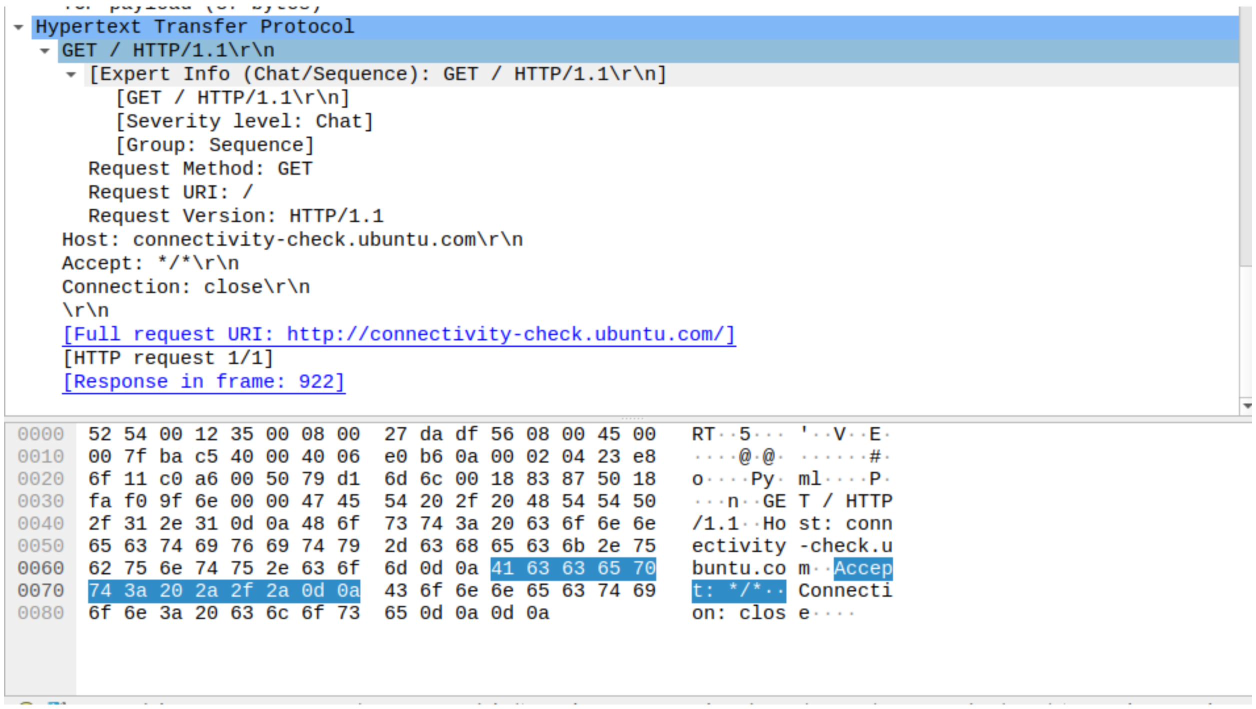
In this TCP header there are two flags set the PSH and the ACK. The ACK is the Acknowledgement flag that is used to a acknowledge the successful receipt of the packets. The PSH is the push flag that ensures that the data is given priority and is being processed at the sending or the receiving end.

c. What are the lengths of the TCP segment and payload?

TCP segment len: 87
TCP Payload: 87 (bytes)

• Now consider the contents of the "GET" message.

a. Scroll down the third pane in the Ethereal window and compare the decoded text with the contents of the HTTP message in the second window.

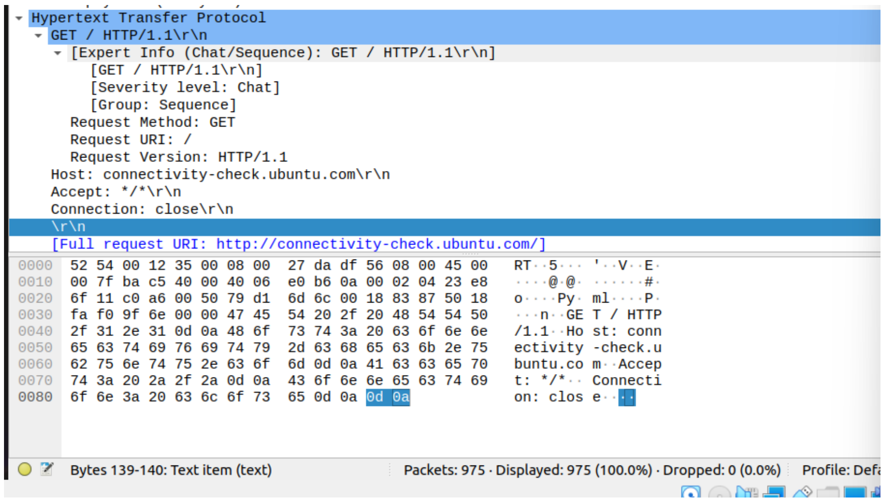


```
▾ Hypertext Transfer Protocol
  ▾ GET / HTTP/1.1\r\n
    ▾ [Expert Info (Chat/Sequence): GET / HTTP/1.1\r\n]
        [GET / HTTP/1.1\r\n]
        [Severity level: Chat]
        [Group: Sequence]
      Request Method: GET
      Request URI: /
      Request Version: HTTP/1.1
    Host: connectivity-check.ubuntu.com\r\n
    Accept: */*\r\n
    Connection: close\r\n
    \r\n
    [Full request URI: http://connectivity-check.ubuntu.com/]
    [HTTP request 1/1]
    [Response in frame: 922]
```

```
0000  52 54 00 12 35 00 08 00  27 da df 56 08 00 45 00   RT··5···  '··V··E·
0010  00 7f ba c5 40 00 40 06  e0 b6 0a 00 02 04 23 e8   ····@·@·  ······#·
0020  6f 11 c0 a6 00 50 79 d1  6d 6c 00 18 83 87 50 18   o····Py·  ml····P·
0030  fa f0 9f 6e 00 00 47 45  54 20 2f 20 48 54 54 50   ···n··GE T / HTTP
0040  2f 31 2e 31 0d 0a 48 6f  73 74 3a 20 63 6f 6e 6e   /1.1··Ho st: conn
0050  65 63 74 69 76 69 74 79  2d 63 68 65 63 6b 2e 75   ectivity -check.u
0060  62 75 6e 74 75 2e 63 6f  6d 0d 0a 41 63 63 65 70   buntu.co m··Accep
0070  74 3a 20 2a 2f 2a 0d 0a  43 6f 6e 6e 65 63 74 69   t: */*·· Connecti
0080  6f 6e 3a 20 63 6c 6f 73  65 0d 0a 0d 0a            on: clos e····
```

b. Count the number of octets in the message and verify that this number is consistent with the length information in the TCP header.

Yes, the number is consistent with the length of the TCP header. If we count the octets it goes from 0 - 140 bytes.



```
▾ Hypertext Transfer Protocol
  ▾ GET / HTTP/1.1\r\n
    ▾ [Expert Info (Chat/Sequence): GET / HTTP/1.1\r\n]
        [GET / HTTP/1.1\r\n]
        [Severity level: Chat]
        [Group: Sequence]
      Request Method: GET
      Request URI: /
      Request Version: HTTP/1.1
    Host: connectivity-check.ubuntu.com\r\n
    Accept: */*\r\n
    Connection: close\r\n
    \r\n
    [Full request URI: http://connectivity-check.ubuntu.com/]
```

```
0000  52 54 00 12 35 00 08 00  27 da df 56 08 00 45 00   RT··5···  '··V··E·
0010  00 7f ba c5 40 00 40 06  e0 b6 0a 00 02 04 23 e8   ····@·@·  ······#·
0020  6f 11 c0 a6 00 50 79 d1  6d 6c 00 18 83 87 50 18   o····Py·  ml····P·
0030  fa f0 9f 6e 00 00 47 45  54 20 2f 20 48 54 54 50   ···n··GE T / HTTP
0040  2f 31 2e 31 0d 0a 48 6f  73 74 3a 20 63 6f 6e 6e   /1.1··Ho st: conn
0050  65 63 74 69 76 69 74 79  2d 63 68 65 63 6b 2e 75   ectivity -check.u
0060  62 75 6e 74 75 2e 63 6f  6d 0d 0a 41 63 63 65 70   buntu.co m··Accep
0070  74 3a 20 2a 2f 2a 0d 0a  43 6f 6e 6e 65 63 74 69   t: */*·· Connecti
0080  6f 6e 3a 20 63 6c 6f 73  65 0d 0a 0d 0a            on: clos e····
```

Bytes 139-140: Text item (text)   Packets: 975 · Displayed: 975 (100.0%) · Dropped: 0 (0.0%)   Profile: Defa

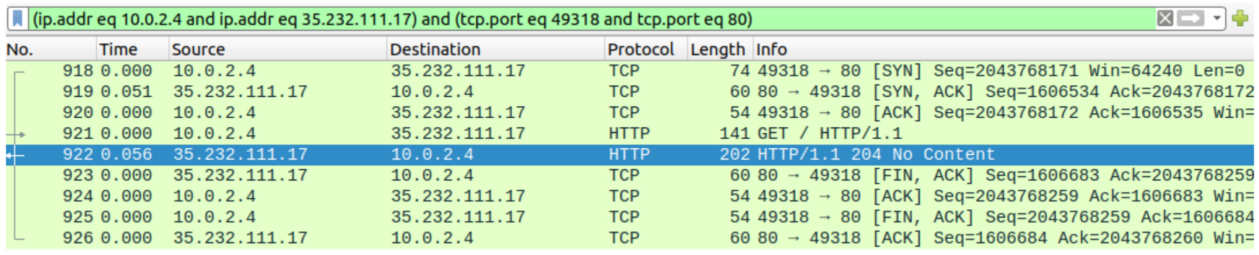
c. What is the next sequence number that is expected in the next segment from the server?

[Next sequence number: 2043768259]

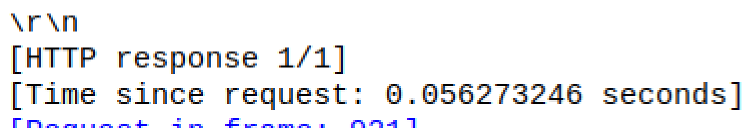
6. HTTP Response - Unaiza Nizami

• How much time elapses between the capture of the GET message and the capture of the corresponding Response message?

The time elapse of the get and the response message is 0.056 which we can see from wireshark table and when we click on the 202 HTTP/1.1 and scroll down to the end we can see the response time.



```
(ip.addr eq 10.0.2.4 and ip.addr eq 35.232.111.17) and (tcp.port eq 49318 and tcp.port eq 80)
No.   Time  Source         Destination    Protocol Length Info
  918 0.000 10.0.2.4       35.232.111.17  TCP        74 49318 → 80 [SYN] Seq=2043768171 Win=64240 Len=0
  919 0.051 35.232.111.17  10.0.2.4       TCP        60 80 → 49318 [SYN, ACK] Seq=1606534 Ack=2043768172
  920 0.000 10.0.2.4       35.232.111.17  TCP        54 49318 → 80 [ACK] Seq=2043768172 Ack=1606535 Win=
  921 0.000 10.0.2.4       35.232.111.17  HTTP      141 GET / HTTP/1.1
  922 0.056 35.232.111.17  10.0.2.4       HTTP      202 HTTP/1.1 204 No Content
  923 0.000 35.232.111.17  10.0.2.4       TCP        60 80 → 49318 [FIN, ACK] Seq=1606683 Ack=2043768259
  924 0.000 10.0.2.4       35.232.111.17  TCP        54 49318 → 80 [ACK] Seq=2043768259 Ack=1606683 Win=
  925 0.000 10.0.2.4       35.232.111.17  TCP        54 49318 → 80 [FIN, ACK] Seq=2043768259 Ack=1606684
  926 0.000 35.232.111.17  10.0.2.4       TCP        60 80 → 49318 [ACK] Seq=1606684 Ack=2043768260 Win=
```

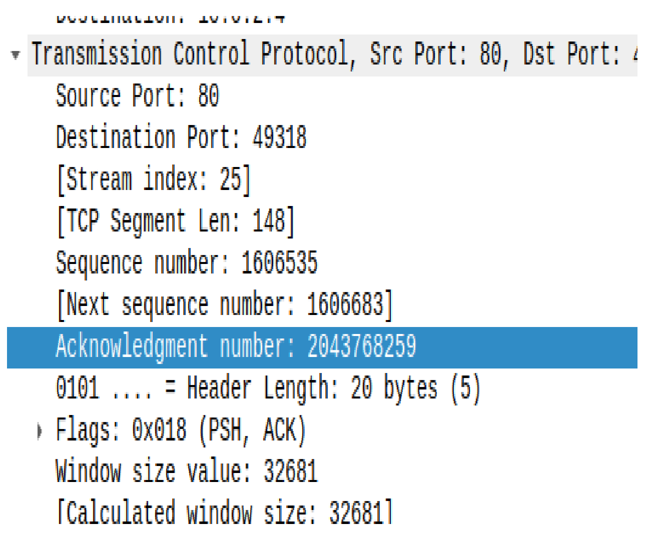


```
\r\n
[HTTP response 1/1]
[Time since request: 0.056273246 seconds]
[Request in frame: 921]
```

• Determine whether the server responds with an HTTP response message or simply with a TCP ACK segment. Verify that the sequence number in the segment from the server is as expected.

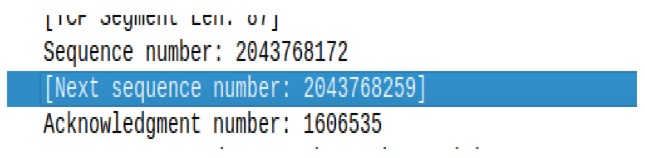
We get an HTTP response message for the HTTP get. We can see the acknowledgment number is the same as the next sequence number of the get response as shown below.

HTTP response



```
Destination: 10.0.2.4
▾ Transmission Control Protocol, Src Port: 80, Dst Port: 4
    Source Port: 80
    Destination Port: 49318
    [Stream index: 25]
    [TCP Segment Len: 148]
    Sequence number: 1606535
    [Next sequence number: 1606683]
    Acknowledgment number: 2043768259
    0101 .... = Header Length: 20 bytes (5)
  ▸ Flags: 0x018 (PSH, ACK)
    Window size value: 32681
    [Calculated window size: 32681]
```

HTTP get



```
[TCP Segment Len: 87]
Sequence number: 2043768172
[Next sequence number: 2043768259]
Acknowledgment number: 1606535
```

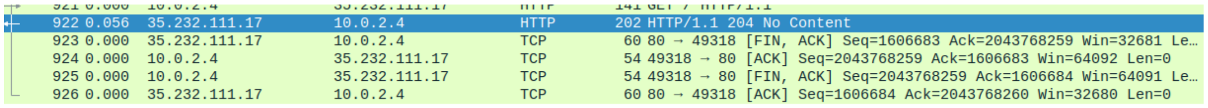
• Now consider the segment that contains the HTTP response message.

a. What is the length of the payload in the TCP segment?
TCP segment len: 148

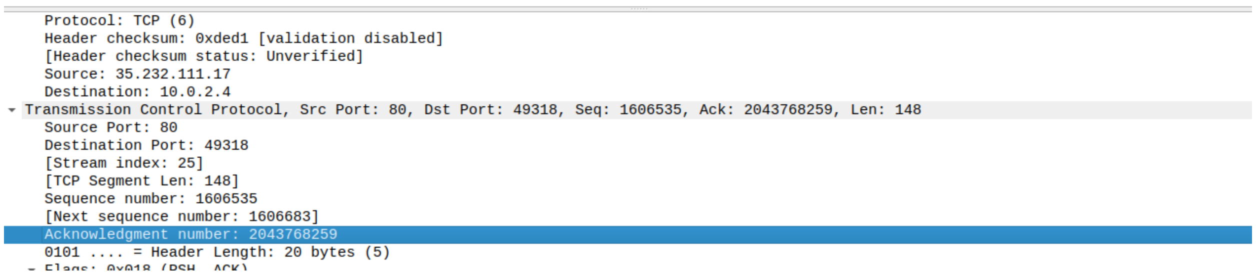
b. Examine whether any of the flags are set and explain why they are set.

There are two flags set are the ACK and PSH flag. The acknowledgement flag is set when the machine is sending the packet acknowledging that the data have been received from the other end and the push flag ensures that the data is given priority and is being processed at the sending or the receiving end. They are set because they notify or acknowledge the client or the server that the data have been received or being sent

c. What acknowledgement number is expected in the next segment from the client?
The acknowledgment number is 2043768259 which tell the client the next expected segment number is 2043768259.



```
921 0.000  10.0.2.4       35.232.111.17  HTTP    141 GET / HTTP/1.1
922 0.056  35.232.111.17  10.0.2.4       HTTP    202 HTTP/1.1 204 No Content
923 0.000  35.232.111.17  10.0.2.4       TCP      60 80 → 49318 [FIN, ACK] Seq=1606683 Ack=2043768259 Win=32681 Le…
924 0.000  10.0.2.4       35.232.111.17  TCP      54 49318 → 80 [ACK] Seq=2043768259 Ack=1606683 Win=64092 Len=0
925 0.000  10.0.2.4       35.232.111.17  TCP      54 49318 → 80 [FIN, ACK] Seq=2043768259 Ack=1606684 Win=64091 Le…
926 0.000  35.232.111.17  10.0.2.4       TCP      60 80 → 49318 [ACK] Seq=1606684 Ack=2043768260 Win=32680 Len=0
```

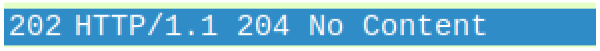


```
    Protocol: TCP (6)
    Header checksum: 0xded1 [validation disabled]
    [Header checksum status: Unverified]
    Source: 35.232.111.17
    Destination: 10.0.2.4
▾ Transmission Control Protocol, Src Port: 80, Dst Port: 49318, Seq: 1606535, Ack: 2043768259, Len: 148
    Source Port: 80
    Destination Port: 49318
    [Stream index: 25]
    [TCP Segment Len: 148]
    Sequence number: 1606535
    [Next sequence number: 1606683]
    Acknowledgment number: 2043768259
    0101 .... = Header Length: 20 bytes (5)
  ▾ Flags: 0x018 (PSH, ACK)
```

• Now consider the HTTP response message.

a. What is the result code in the response message?

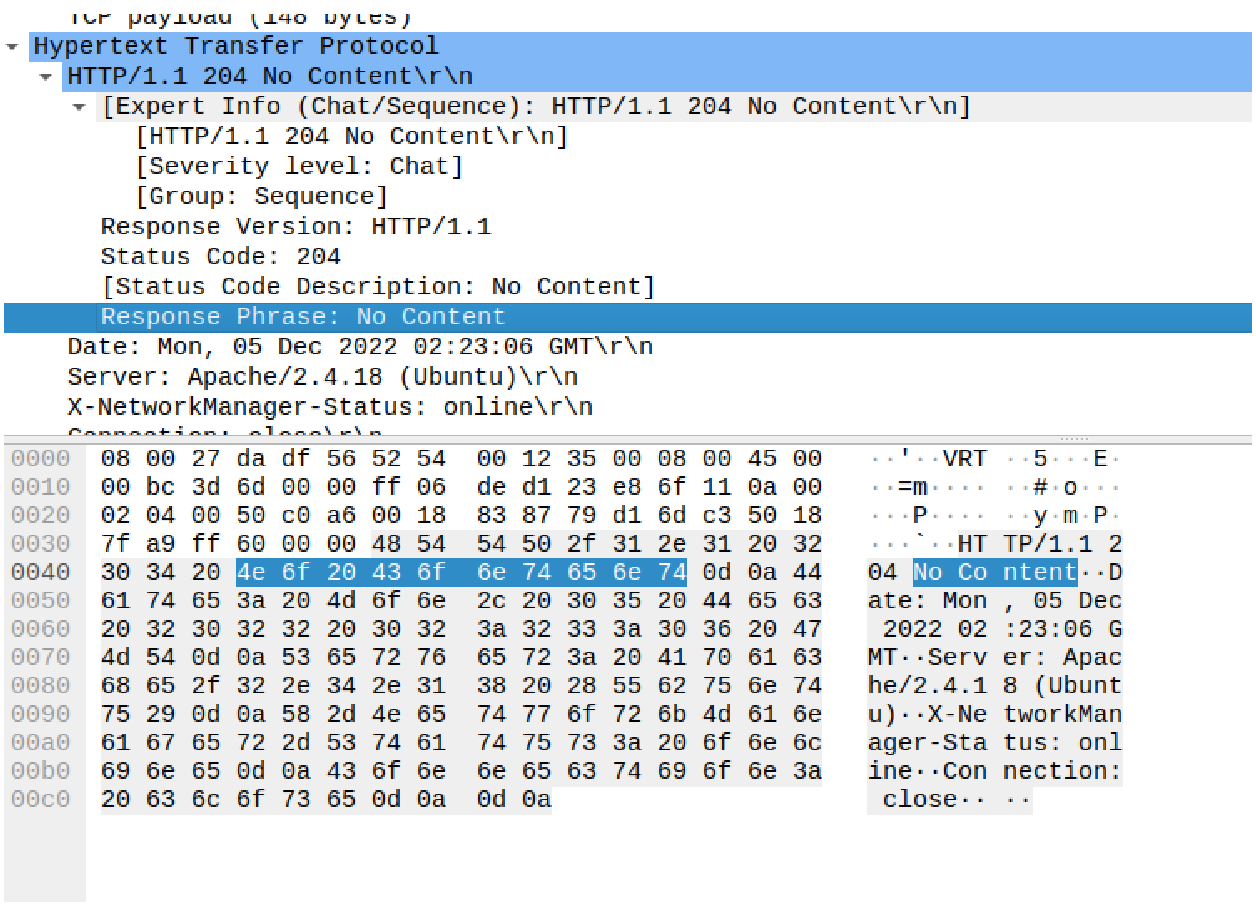
The result code of the response message is **202 HTTP/1.1 204 No Content** with status code 204



```
202 HTTP/1.1 204 No Content
```

b. Highlight the “data” section of the HTTP response message. Scroll down the third pane in the Ethereal window and compare the decoded text with the contents of the web page that was displayed on your screen.

Since this response message had no content but the request is succeeded and the client does not need to navigate to a different site..



```
    TCP payload (148 bytes)
▾ Hypertext Transfer Protocol
  ▾ HTTP/1.1 204 No Content\r\n
    ▾ [Expert Info (Chat/Sequence): HTTP/1.1 204 No Content\r\n]
        [HTTP/1.1 204 No Content\r\n]
        [Severity level: Chat]
        [Group: Sequence]
      Response Version: HTTP/1.1
      Status Code: 204
      [Status Code Description: No Content]
      Response Phrase: No Content
    Date: Mon, 05 Dec 2022 02:23:06 GMT\r\n
    Server: Apache/2.4.18 (Ubuntu)\r\n
    X-NetworkManager-Status: online\r\n
```

```
0000  08 00 27 da df 56 52 54  00 12 35 00 08 00 45 00   ··'··VRT ··5···E·
0010  00 bc 3d 6d 00 00 ff 06  de d1 23 e8 6f 11 0a 00   ··=m···· ··#·o···
0020  02 04 00 50 c0 a6 00 18  83 87 79 d1 6d c3 50 18   ···P···· ··y·m·P·
0030  7f a9 ff 60 00 00 48 54  54 50 2f 31 2e 31 20 32   ···`··HT TP/1.1 2
0040  30 34 20 4e 6f 20 43 6f  6e 74 65 6e 74 0d 0a 44   04 No Co ntent··D
0050  61 74 65 3a 20 4d 6f 6e  2c 20 30 35 20 44 65 63   ate: Mon , 05 Dec
0060  20 32 30 32 32 20 30 32  3a 32 33 3a 30 36 20 47    2022 02 :23:06 G
0070  4d 54 0d 0a 53 65 72 76  65 72 3a 20 41 70 61 63   MT··Serv er: Apac
0080  68 65 2f 32 2e 34 2e 31  38 20 28 55 62 75 6e 74   he/2.4.1 8 (Ubunt
0090  75 29 0d 0a 58 2d 4e 65  74 77 6f 72 6b 4d 61 6e   u)··X-Ne tworkMan
00a0  61 67 65 72 2d 53 74 61  74 75 73 3a 20 6f 6e 6c   ager-Sta tus: onl
00b0  69 6e 65 0d 0a 43 6f 6e  6e 65 63 74 69 6f 6e 3a   ine··Con nection:
00c0  20 63 6c 6f 73 65 0d 0a  0d 0a                      close·· ··
```