Networking with Linux Lab
Module 4: Analyzing Network traffic

Assignment 12: Monitoring the Network using WireShark

1. Explain how to create a .pcap file in ns3.

Hint: Utilize the EnablePcapAll() method and refer to the first.cc, second.cc, and third.cc files.

Aim: To capture and analyze network traffic in ns3 simulations

Theory:

Creating a .pcap file in NS-3 involves capturing network traffic generated during a simulation and saving it in the .pcap format, which is commonly used for packet capture. NS-3 provides a simple way to enable .pcap tracing for all nodes in a network using the EnablePcapAll() method. This method is typically called on the point-to-point devices or NetDevices.

Here's a step-by-step guide to creating a .pcap file in NS-3:

1. **Include Necessary Headers:** First, include the necessary headers in your simulation script. You'll typically need headers for the network devices, .pcap tracing, and any other modules you're using.
2. **Create the Network Topology:** Define the network topology you want to simulate. This includes creating nodes, setting up point-to-point or other types of connections between them, and configuring their attributes.
3. **Enable Pcap Tracing:** After setting up the network topology, enable pcap tracing for all the nodes in the network. This is done using the EnablePcapAll() method, which is typically called on the point-to-point devices or NetDevices.
4. **Run the Simulation:** Once you've set up the network and enabled pcap tracing, run the simulation script. During the simulation, NS-3 will capture the network traffic and save it to a .pcap file.
5. **Analyze the .pcap File:** After the simulation has completed, you can analyze the generated .pcap file using tools like Wireshark. This allows you to inspect the captured packets, analyze network behavior, and debug your simulation if necessary.

Code:

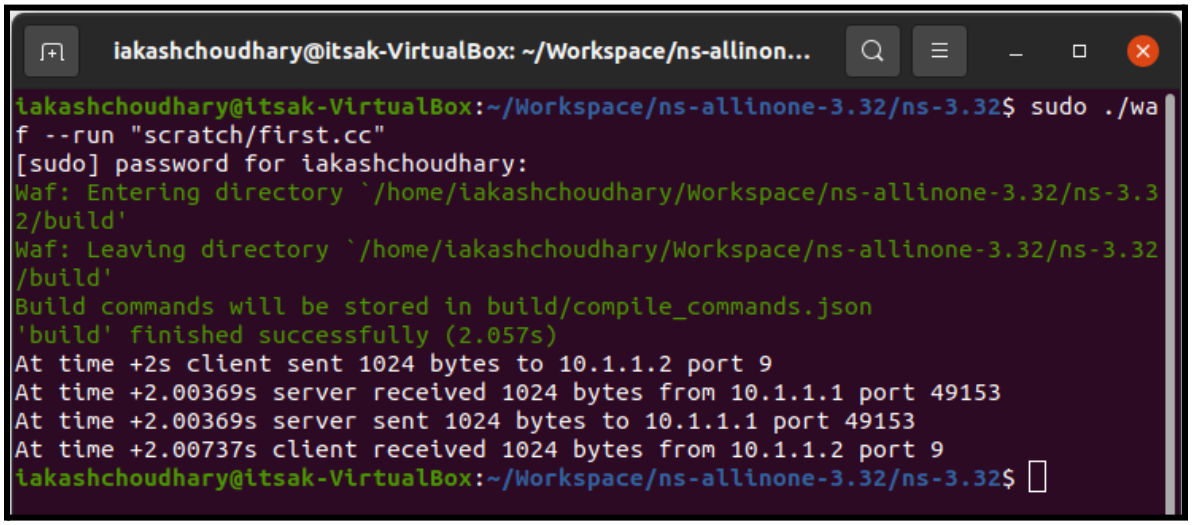
> first.cc

```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"
#include "ns3/mobility-helper.h"
#include "ns3/netanim-module.h"

// Default Network Topology
//
//       10.1.1.0
// n0 -------------- n1
//    point-to-point
//

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("FirstScriptExample");

int
main (int argc, char *argv[])
{
  CommandLine cmd (__FILE__);
  cmd.Parse (argc, argv);

  Time::SetResolution (Time::NS);
  LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
  LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);

  NodeContainer nodes;
  nodes.Create (2);

  PointToPointHelper pointToPoint;
  pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
  pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));

  NetDeviceContainer devices;
  devices = pointToPoint.Install (nodes);

  InternetStackHelper stack;
  stack.Install (nodes);

  Ipv4AddressHelper address;
```

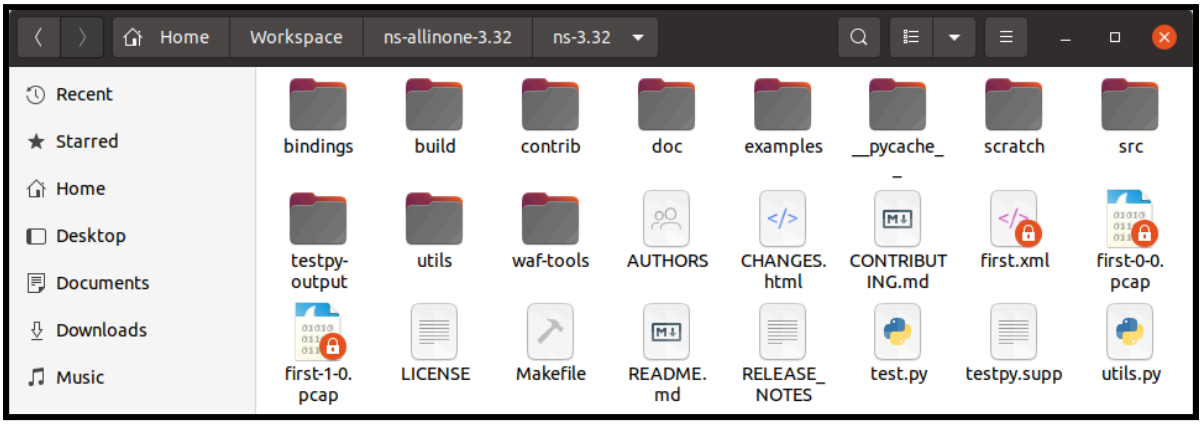
```
  address.SetBase ("10.1.1.0", "255.255.255.0");

  Ipv4InterfaceContainer interfaces = address.Assign (devices);

  UdpEchoServerHelper echoServer (9);

  ApplicationContainer serverApps = echoServer.Install (nodes.Get (1));
  serverApps.Start (Seconds (1.0));
  serverApps.Stop (Seconds (10.0));

  UdpEchoClientHelper echoClient (interfaces.GetAddress (1), 9);
  echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
  echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient.SetAttribute ("PacketSize", UintegerValue (1024));

  ApplicationContainer clientApps = echoClient.Install (nodes.Get (0));
  clientApps.Start (Seconds (2.0));
  clientApps.Stop (Seconds (10.0));

  // NetAnimation ---before simulator run
  MobilityHelper mobility;
  mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
  mobility.Install(nodes);

  AnimationInterface anim("first.xml");
  AnimationInterface::SetConstantPosition(nodes.Get(0), 10, 25);
  AnimationInterface::SetConstantPosition(nodes.Get(1), 40, 25);
  anim.EnablePacketMetadata(true);

  // Creating .pcap files for Wireshark-pcap: packet capture information
  pointToPoint.EnablePcapAll("first");

  Simulator::Run ();
  Simulator::Destroy ();
  return 0;
}
```

Command & Screenshot:

> **$ sudo ./waf --run "scratch/first.cc"**



> After **successfully executing** the above **command**, a **.pcap file** will be **created**.



Conclusion: Enabling .pcap functionality in ns3 facilitates packet capture and analysis for network simulations.

2. Analyze the network traffic captured in the first and third (.pcap) files.

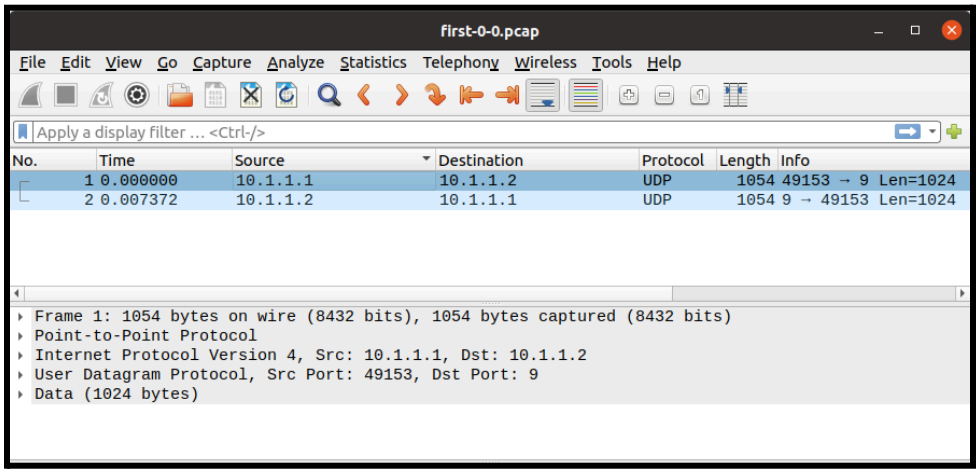
Aim: To understand and evaluate the communication patterns, potential security threats, and overall network performance

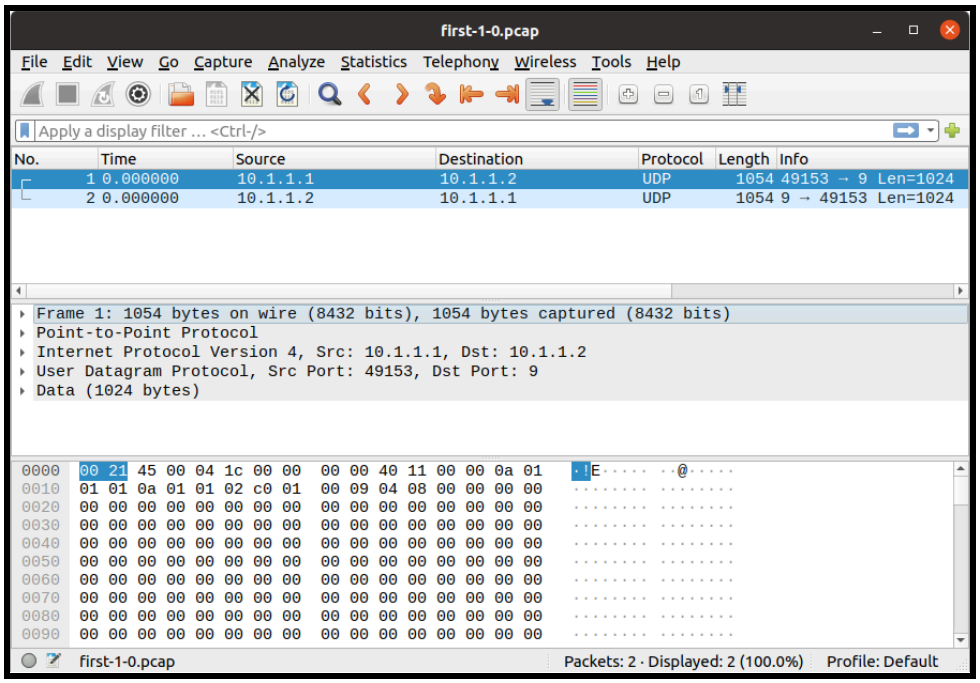
Theory:

Analyzing network traffic captured in .pcap files involves several key steps and theories:

1. **Packet Inspection:** Examine individual packets within the .pcap files to understand their structure, headers, and payload contents.
2. **Protocol Analysis:** Identify the protocols used in the communication (e.g., HTTP, TCP, UDP) to understand how data is being transmitted.
3. **Traffic Patterns:** Analyze the patterns of communication, such as frequency of requests, data volume transferred, and timing of interactions.
4. **Anomaly Detection:** Look for any irregularities or suspicious activity in the network traffic that could indicate security threats or performance issues.
5. **Traffic Flow Analysis:** Determine the flow of traffic between network nodes, including source and destination addresses, ports, and protocols.
6. **Performance Metrics:** Measure network performance metrics such as latency, throughput, and packet loss to assess the efficiency of communication.
7. **Interpretation:** Interpret the findings to draw conclusions about the behavior of the network, potential issues, and areas for optimization or security enhancement.

Screenshot:




first-1-0.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
1 0.000000 10.1.1.1 10.1.1.2 UDP 1054 49153 → 9 Len=1024
2 0.000000 10.1.1.2 10.1.1.1 UDP 1054 9 → 49153 Len=1024
Frame 1: 1054 bytes on wire (8432 bits), 1054 bytes captured (8432 bits)
Point-to-Point Protocol
Internet Protocol Version 4, Src: 10.1.1.1, Dst: 10.1.1.2
User Datagram Protocol, Src Port: 49153, Dst Port: 9
Data (1024 bytes)
0000 00 21 45 00 04 1c 00 00 00 00 40 11 00 00 0a 01
0010 01 01 0a 01 01 02 c0 01 00 09 04 08 00 00 00 00
0020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
first-1-0.pcap
Packets: 2 · Displayed: 2 (100.0%)
Profile: Default

2.1 Steps to capture live data packets.

Theory:

1. **Choose a Tool:** Select a packet capturing tool like Wireshark or tcpdump.
2. **Launch the Tool:** Open the chosen tool on your device.
3. **Select Interface:** Choose the network interface you want to monitor (e.g., Wi-Fi, Ethernet).
4. **Start Capturing:** Begin capturing packets by clicking the "Start" or "Capture" button.
5. **Analyze Data:** Review captured packets for relevant information like source, destination, and protocol.
6. **Filter Data (Optional):** Use filters to refine captured data based on specific criteria.
7. **Stop Capturing:** End the packet capture process when you've collected enough data.
8. **Save Data (Optional):** Save captured packets for further analysis or documentation.

2.2. Explain the three sections of Wireshark with examples.

Theory:

1. **Packet List Pane:** This section displays a list of captured packets in a tabular format. Each row represents a single packet, and columns provide details such as the packet number, time of capture, source and destination addresses, protocol type, and packet length. This pane gives you a quick overview of all captured network traffic.Example: In this pane, you might see rows representing different types of packets such as Ethernet frames, IP packets, TCP segments, or UDP datagrams. You can examine each packet to analyze its contents and understand the communication between network hosts.

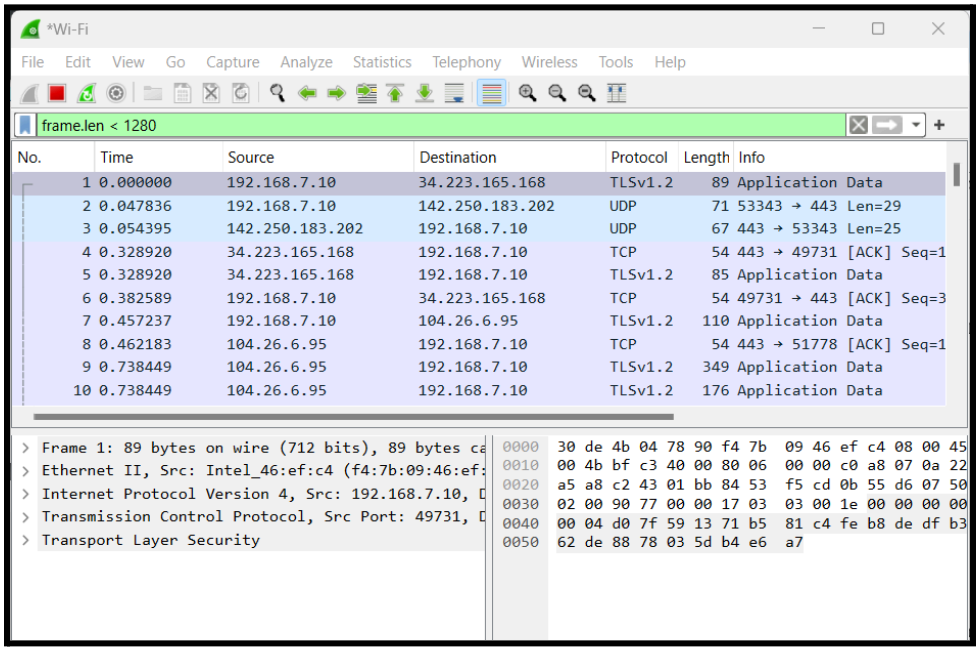
2. **Packet Details Pane:** When you select a packet from the Packet List Pane, its details are displayed in this section. It provides a hierarchical view of the packet contents, showing various protocol layers encapsulated within the packet. Each layer contains fields and their corresponding values, allowing you to inspect the headers and payloads of different network protocols.Example: If you select an HTTP packet, you'll see the Ethernet frame header, followed by the IP header, then the TCP header, and finally the HTTP header and body. You can analyze the HTTP request and response messages, including URLs, headers, and data payloads.

3. **Packet Bytes Pane:** This section provides a hexadecimal and ASCII representation of the selected packet's raw bytes. It allows you to view the packet's binary data in detail, which can be useful for analyzing packet structures, identifying patterns, or troubleshooting issues at the byte level.Example: If you're investigating a network protocol or trying to diagnose a network problem, you can use the Packet Bytes Pane to examine the raw packet contents. You might look for specific byte sequences, protocol signatures, or anomalies in the data.

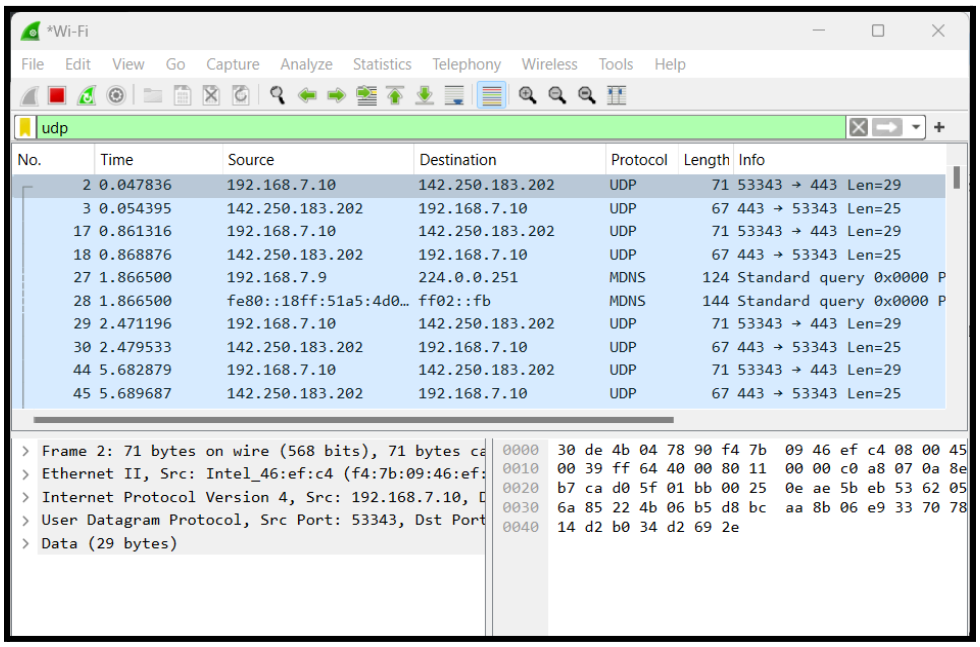
2.3. Perform at least four filter expressions in the Wireshark tool on any packet (live or offline).

Screenshot:

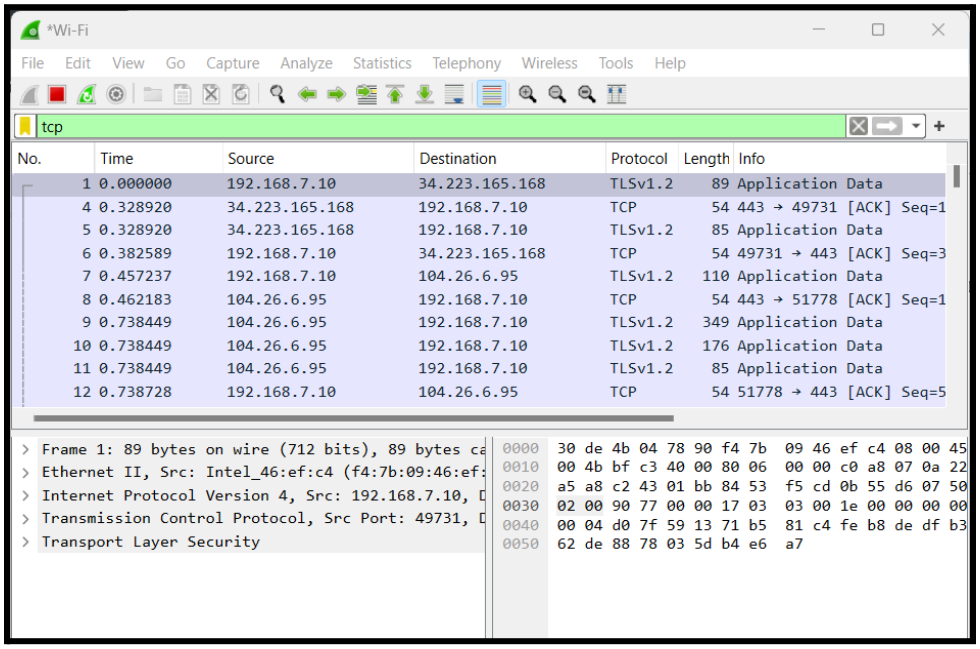
1. **frame.len < 1280**



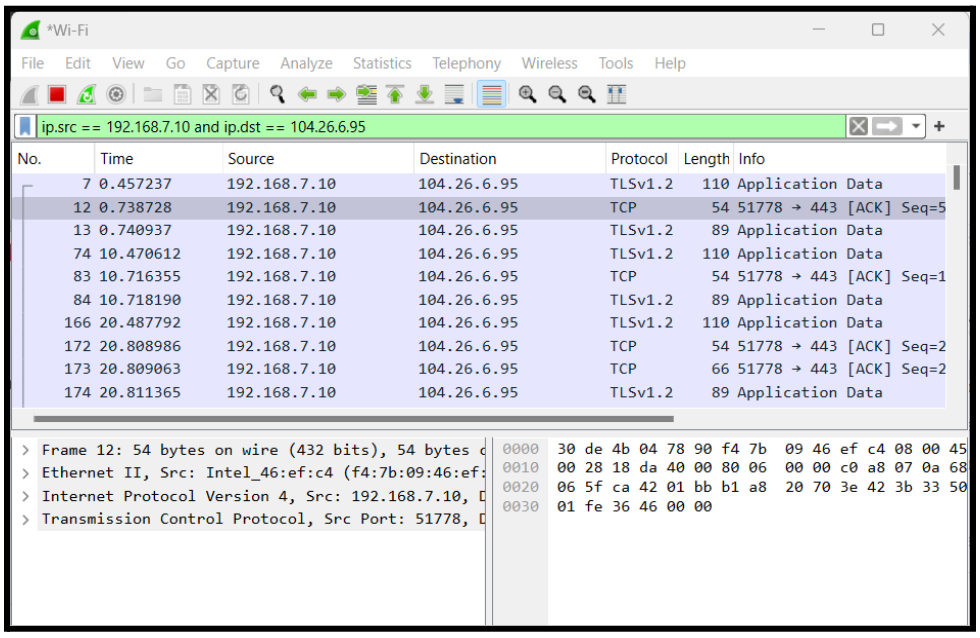
2. **udp**



3. **tcp**



4. **ip.src == 192.168.7.10 and ip.dst == 104.26.6.95**



2.4. Does Wireshark impact performance?

Theory:

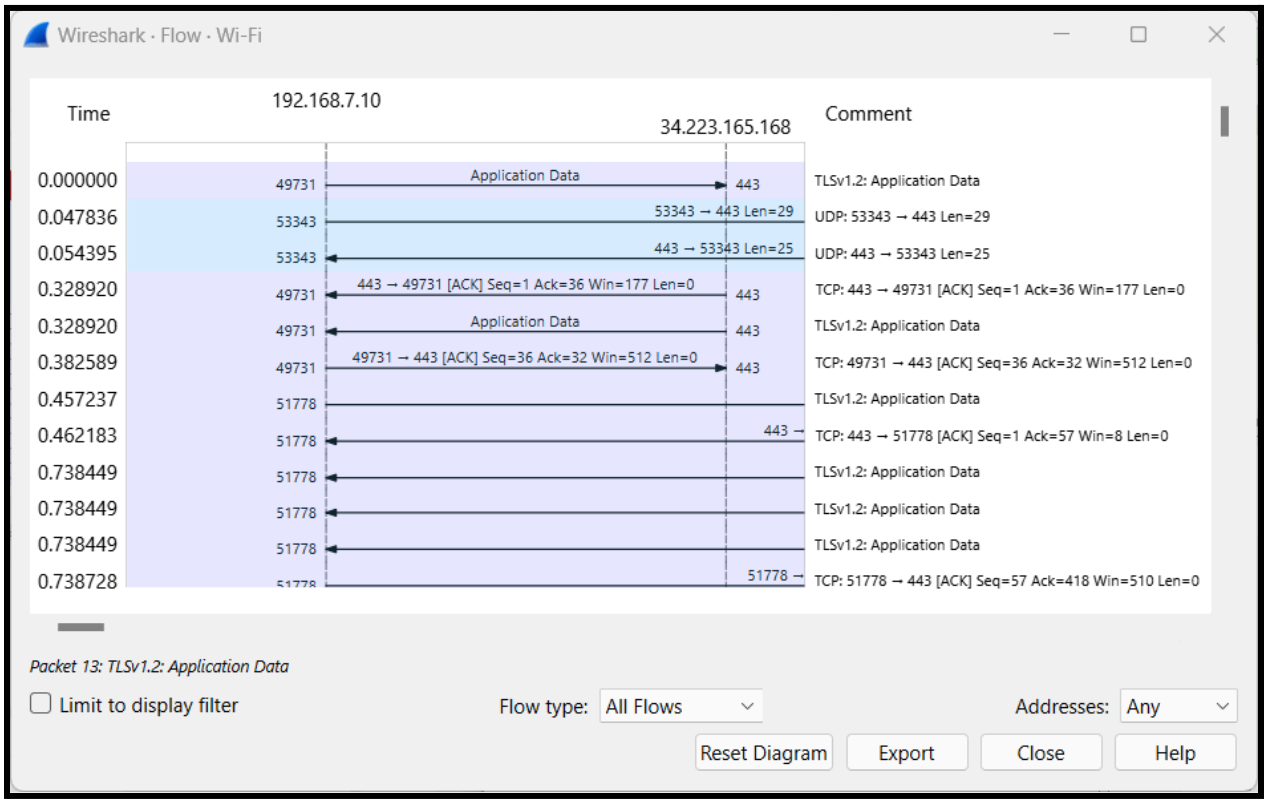
No, Wireshark doesn't impact performance. It typically has minimal impact on performance during normal usage. Its overhead is generally negligible unless you're capturing a large amount of network traffic or running it on a system with very limited resources. However, in most cases, the impact should be insignificant.

2.5. Flowgraph.

Theory:

In Wireshark, a flowgraph is a graphical representation of network traffic patterns over time. It displays the flow of packets between hosts or network devices, helping users visualize communication patterns and identify anomalies. Flowgraphs in Wireshark typically show packet interactions, such as connections, data transfers, and protocol exchanges, which can be useful for troubleshooting network issues, analyzing performance, or investigating security incidents.

Screenshot:

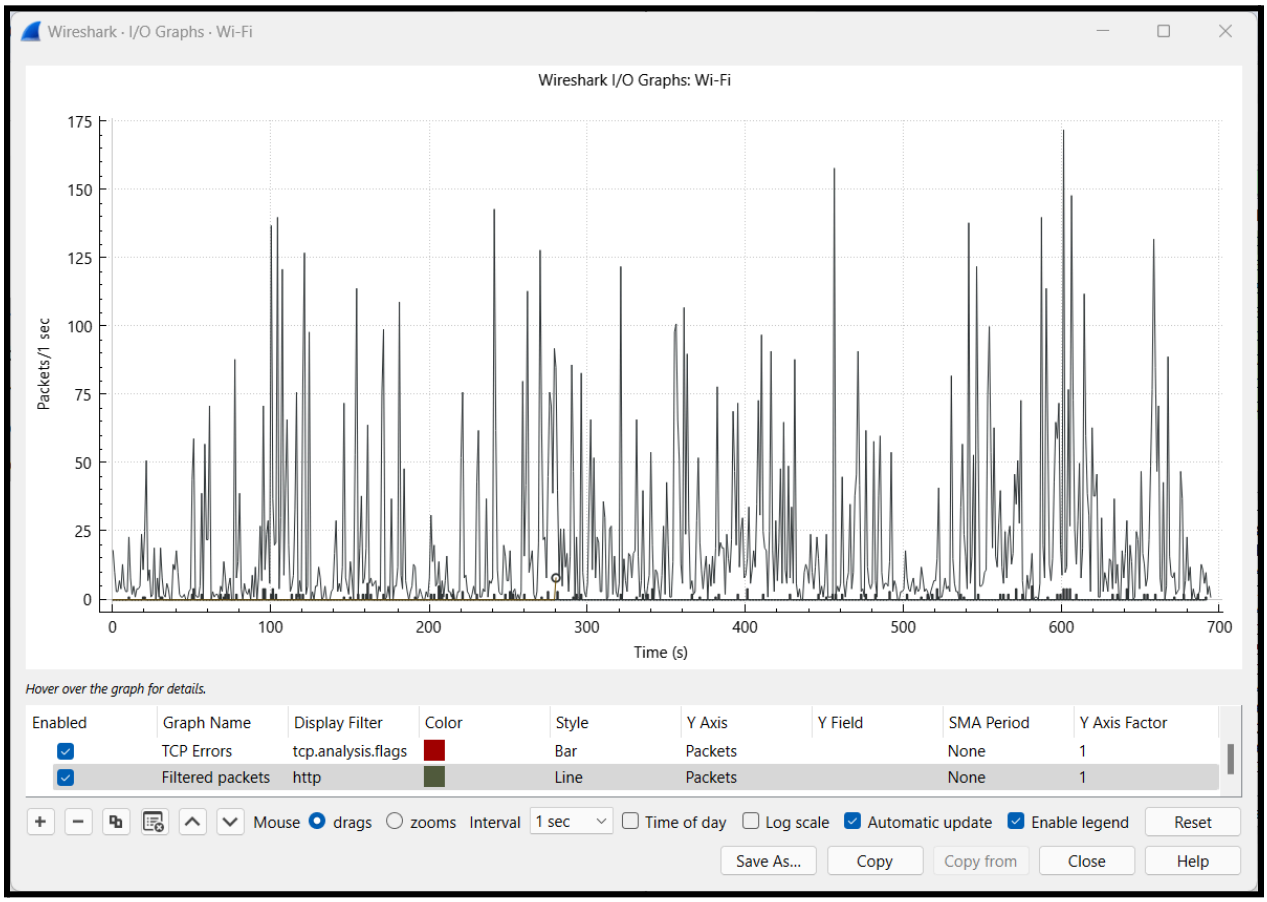


2.6. I/O Graph.

Theory:

An I/O Graph in Wireshark is a graphical representation of input and output (I/O) traffic over time. It provides a visual depiction of network activity, making it easier to identify patterns, trends, and anomalies. It typically plots parameters like packet count, packet size, throughput, or any other metrics captured by Wireshark over a specified time range. This visual representation aids in network analysis and troubleshooting by allowing users to quickly identify network issues or unusual behavior.

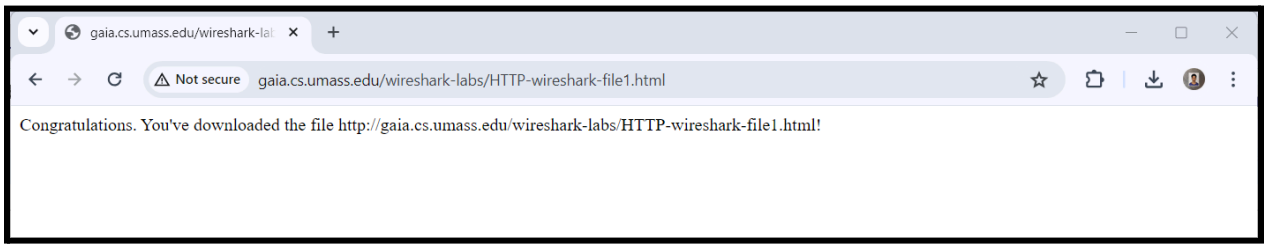
Screenshot:



Conclusion: Analyzing network traffic captured in the provided .pcap files reveals patterns of communication and potential security threats.

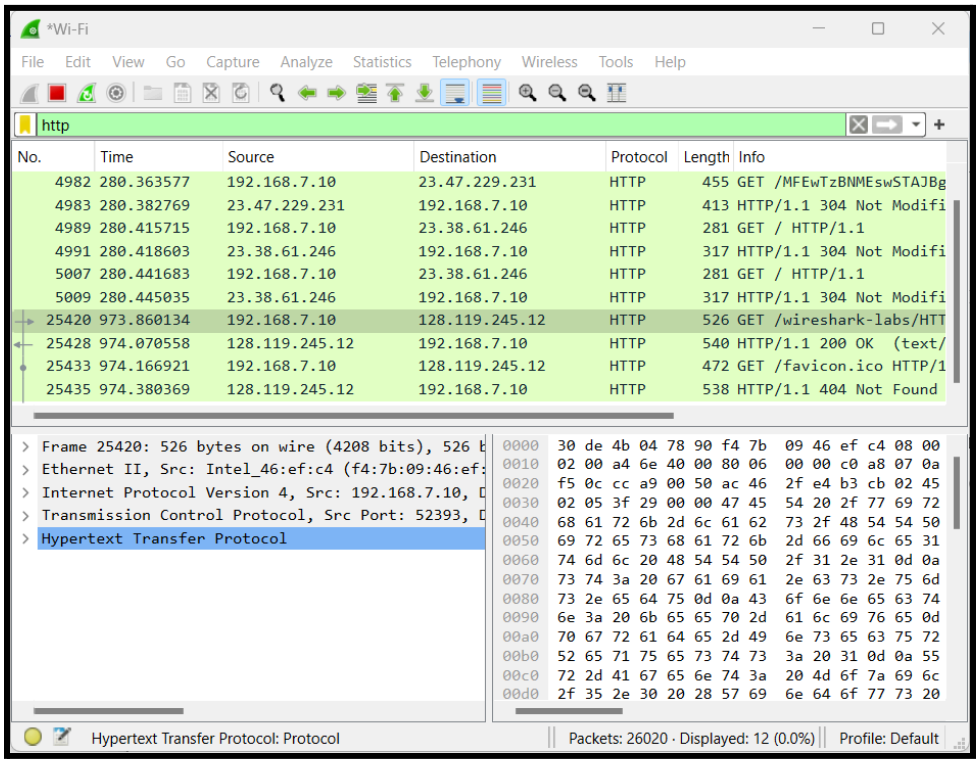
3. Use Wireshark to capture packets by browsing http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file1.html to answer the following questions:

Screenshot:



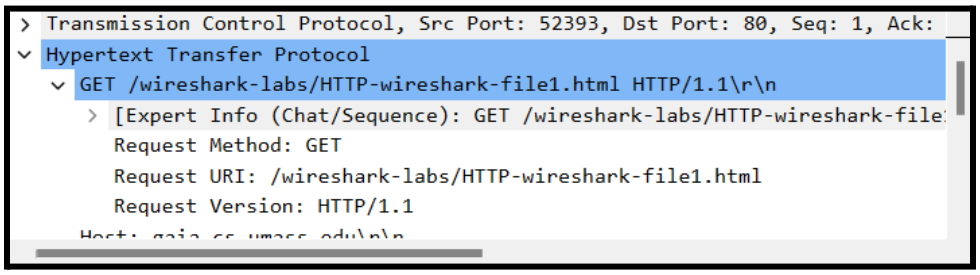
a. Filter the packets using "http."

Screenshot:



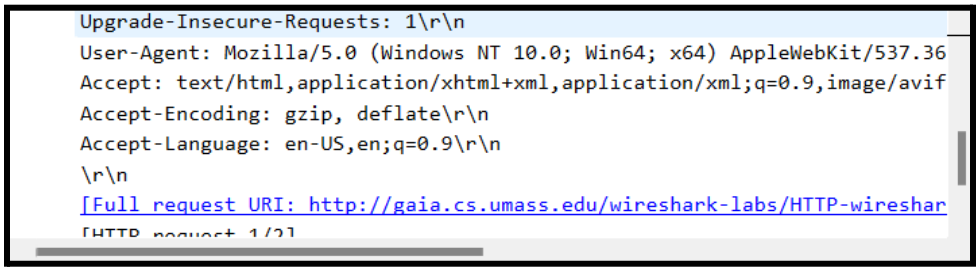
b. What version of HTTP is the server running?

Screenshot:



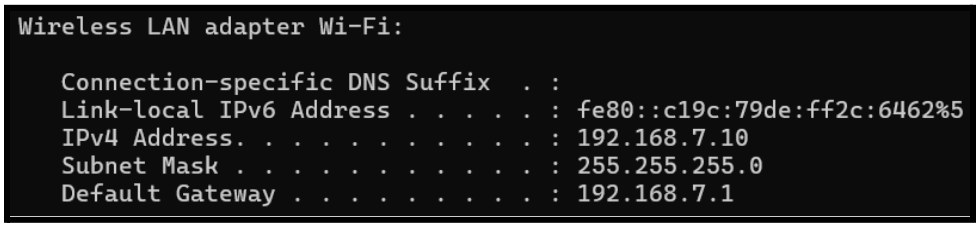
c. What languages (if any) does your browser indicate that it can accept to the server?

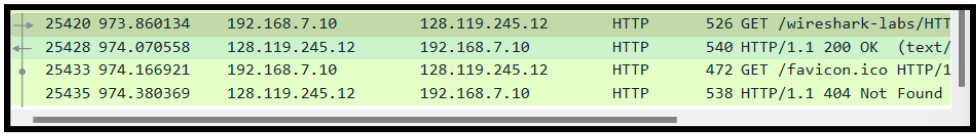
Screenshot:



d. What is the IP address of your computer? What is the IP address of the server?

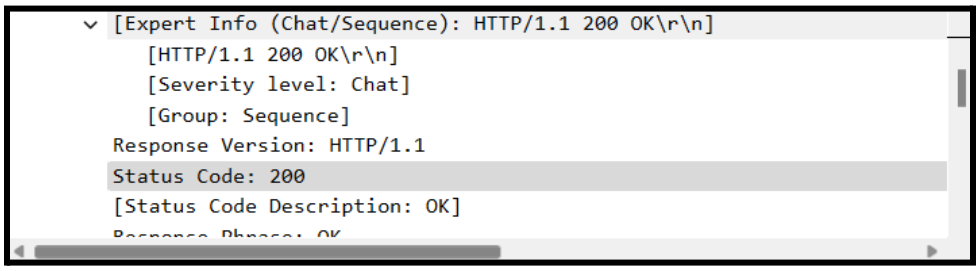
Screenshot:





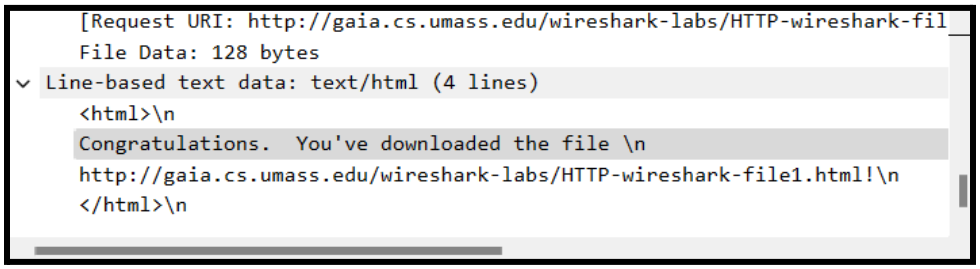
e. What is the status code returned from the server to your browser?

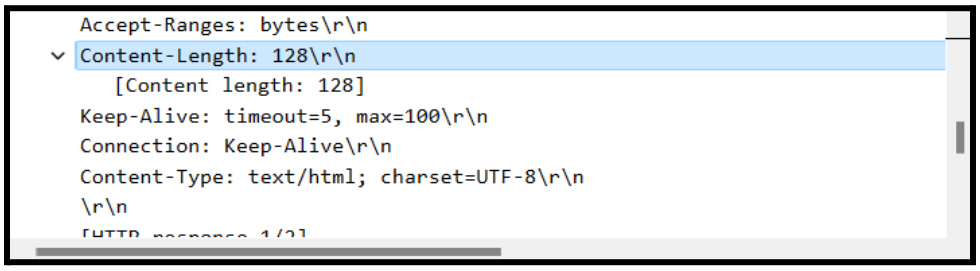
Screenshot:



f. Is the server returning any content? If so, how many bytes of content are being returned to your browser?

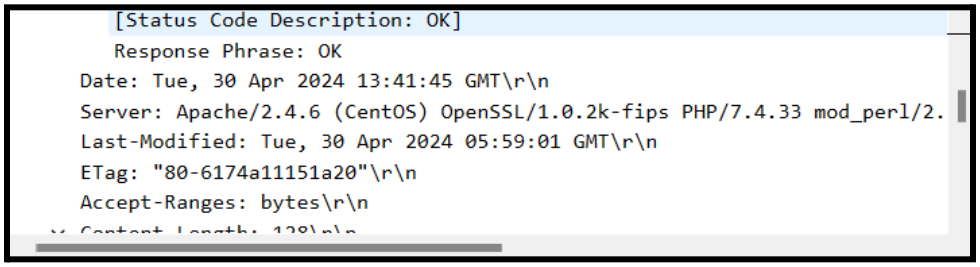
Screenshot:





g. When was the HTML file that you are retrieving last modified at the server?

Screenshot:

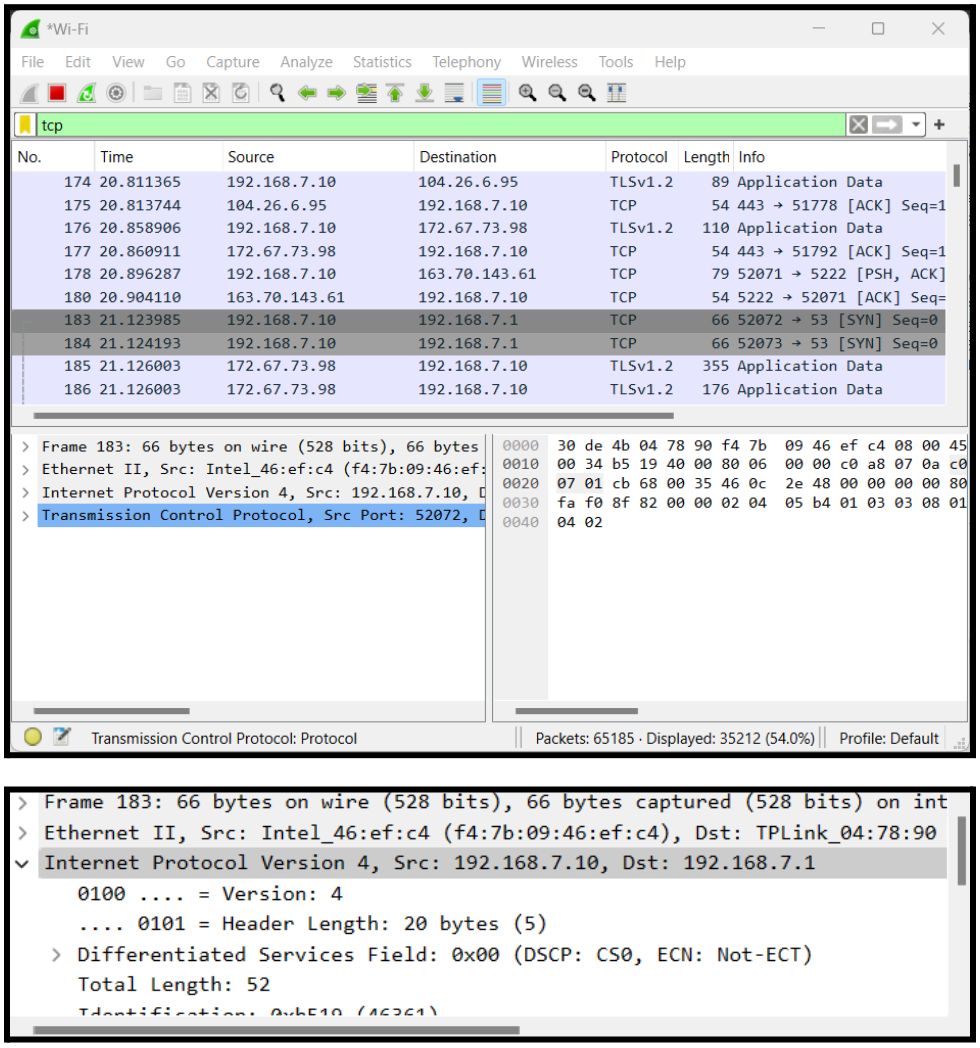


4. Use Wireshark to capture packets by browsing your college website and answer the following questions:

Screenshot:

a. Choose any packet. What are the IP addresses of the source and destination?

Screenshot:



b. What is the value in the Time-to-Live (TTL) field in this IPv4 datagram's header? What does it mean?

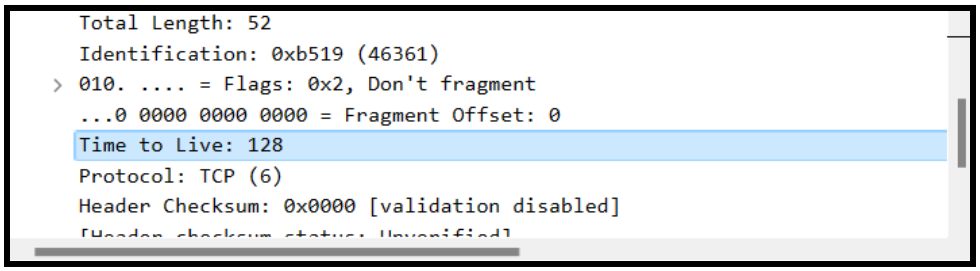
Theory:

The TTL field serves multiple purposes:

1. **Preventing loops:** It ensures that packets don't endlessly circulate in a network due to routing errors or other issues.

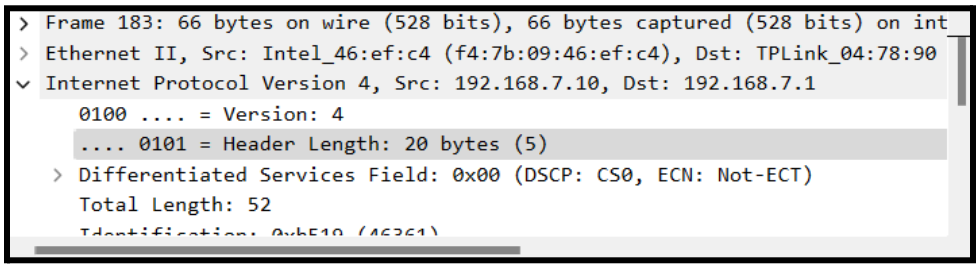
2. **Determining packet freshness:** It provides a means to gauge the "freshness" or "age" of a packet, as it decreases with each hop. This is particularly useful for routing protocols to prevent the propagation of outdated routing information.

Screenshot:



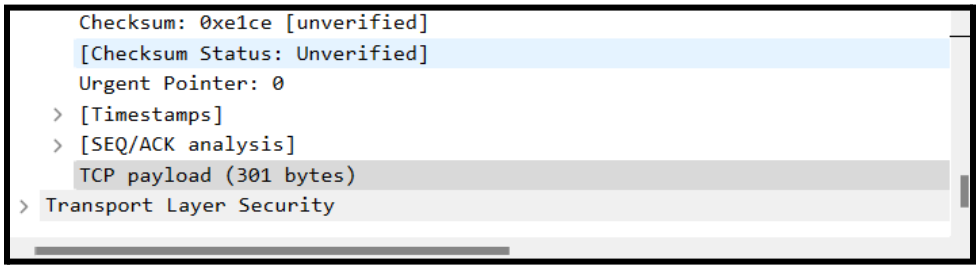
c. How many bytes are in the IP header?

Screenshot:



d. How many bytes are in the payload of the IP datagram?

Screenshot:



e. Explain how you determined the number of payload bytes.

Theory:

The TCP header typically consists of 20 bytes, and the payload size can be calculated by subtracting the header length from the total length of the packet.

If the total length of the packet is 301 bytes and the TCP header length is 20 bytes, then the payload size would indeed be 281 bytes.

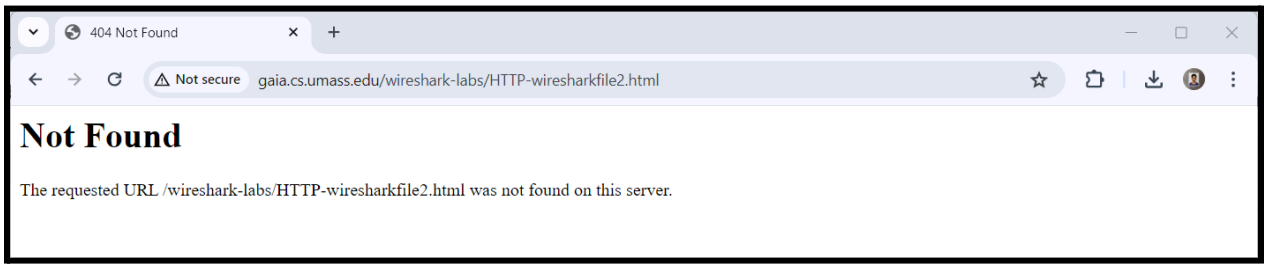
So, Payload size = Total length of packet - TCP header length

Payload size = 301 bytes - 20 bytes

Payload size = 281 bytes

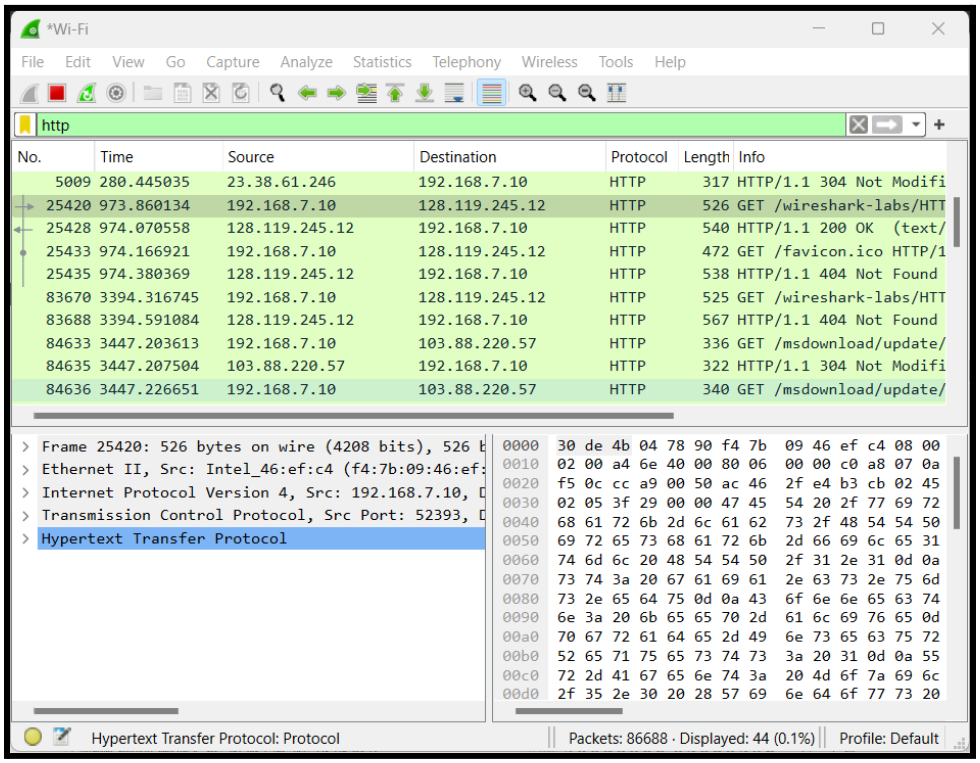
5. Use Wireshark to capture packets by browsing http://gaia.cs.umass.edu/wireshark-labs/HTTP-wiresharkfile2.html and answer the following questions:

Screenshot:



a. Filter the packets using "http."

Screenshot:



b. Inspect the contents of the first HTTP GET request from your browser to the server. Do you see an "IF-MODIFIED-SINCE" line in the HTTP GET?

Theory:

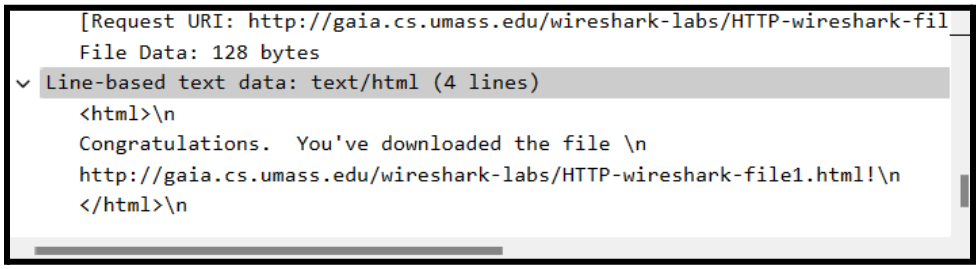
No, there is no IF-MODIFIED-SINCE line in the GET message.

c. Inspect the contents of the server response. Did the server explicitly return the contents of the file? How can you tell?

Theory:

The server did explicitly return the contents of the file. Wireshark includes a section titled "Line-Based Text Data" which shows what the server sent back to my browser which is specifically what the website showed when I brought it up on my browser.

Screenshot:



d. Now inspect the contents of the second HTTP GET request from your browser to the server. Do you see an "IF-MODIFIED-SINCE:" line in the HTTP GET? If so, what information follows the "IF-MODIFIED-SINCE:" header?

Theory:

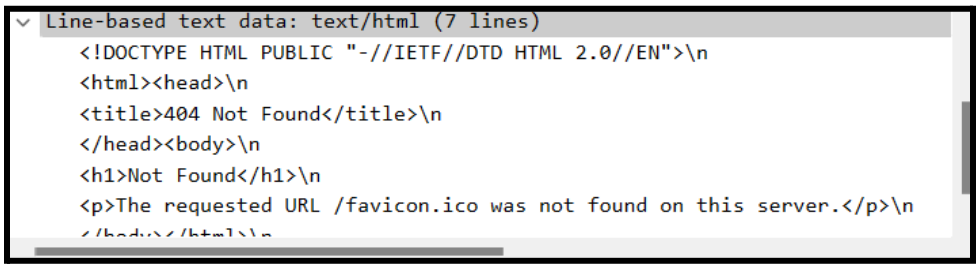
No, there is no IF-MODIFIED-SINCE line in the GET message.

e. What is the HTTP status code and phrase returned from the server in response to this second HTTP GET? Did the server explicitly return the contents of the file? Explain.

Theory:

The HTTP status code is “404: Not Modified”. The server did not return the contents of the file because the browser simply retrieved the contents from its cache. Had the file been modified since it was last accessed, it would have returned the contents of the file, instead it simply told my browser to retrieve the old file from its cached memory.

Screenshot:

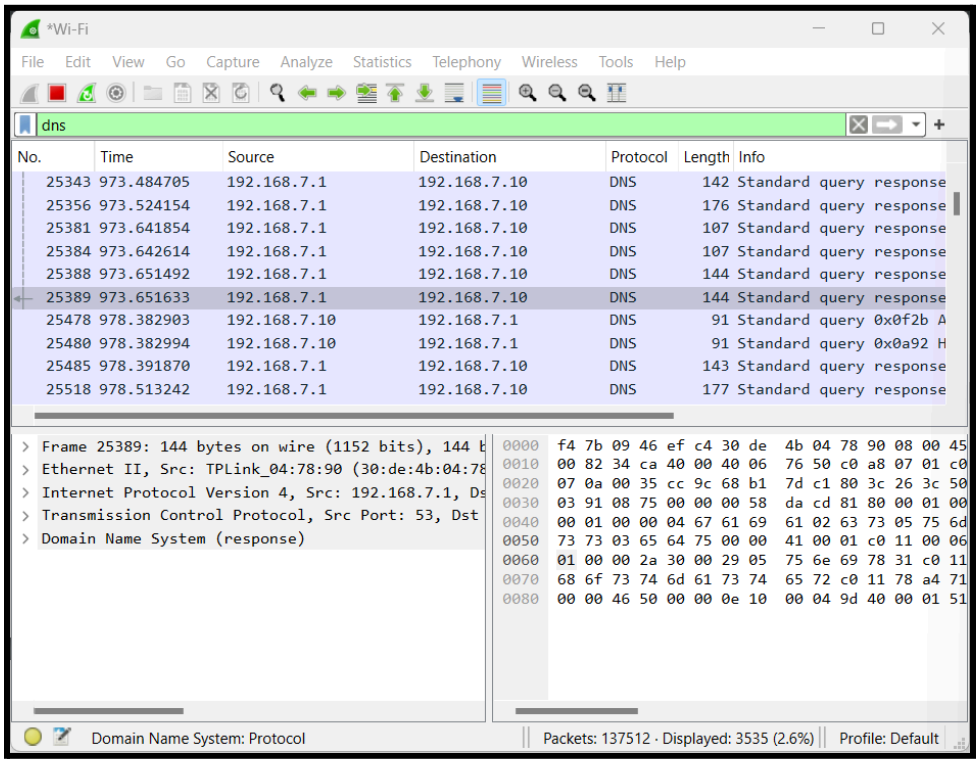


6. Use Wireshark to capture packets by browsing your college website and answer the following questions:

Screenshot:

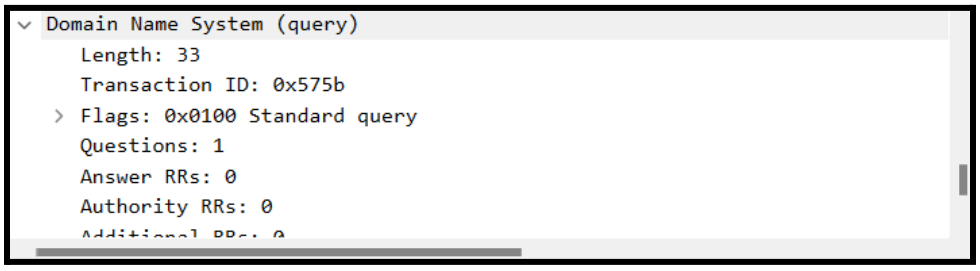
a. Filter the packets using "DNS."

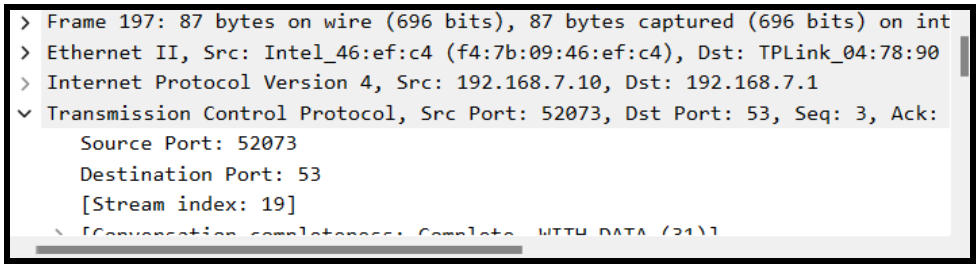
Screenshot:



b. Locate the first DNS query message. What is the packet number in the trace for the DNS query message? Is this query message sent over UDP or TCP?

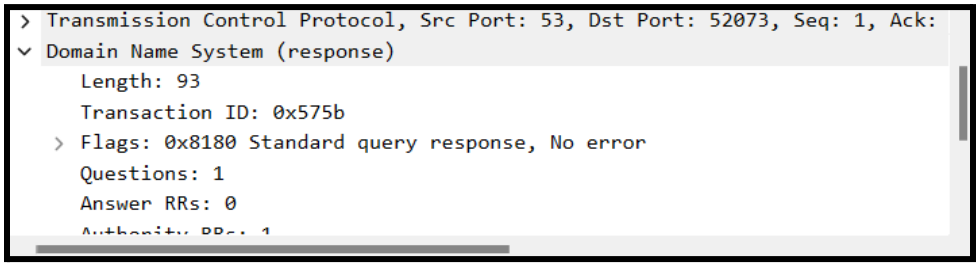
Screenshot:





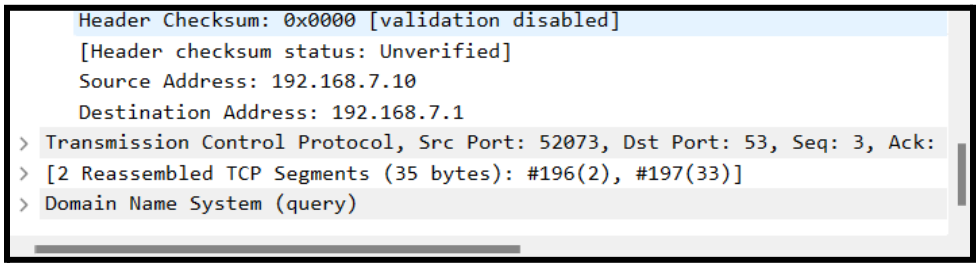
c. Now locate the corresponding DNS response to the initial DNS query. What is the packet number in the trace for the DNS response message? Is this response message received via UDP or TCP?

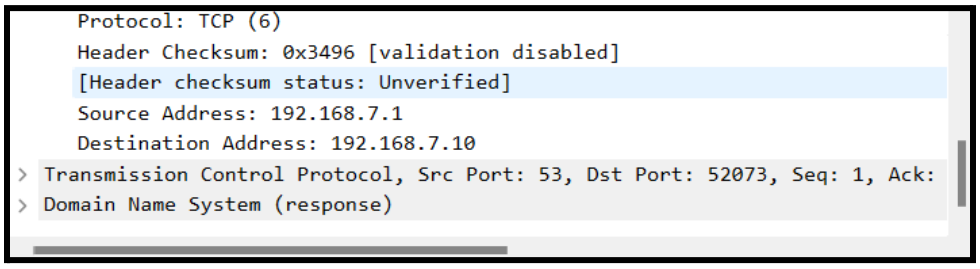
Screenshot:



d. What is the destination port for the DNS query message? What is the source port of the DNS response message?

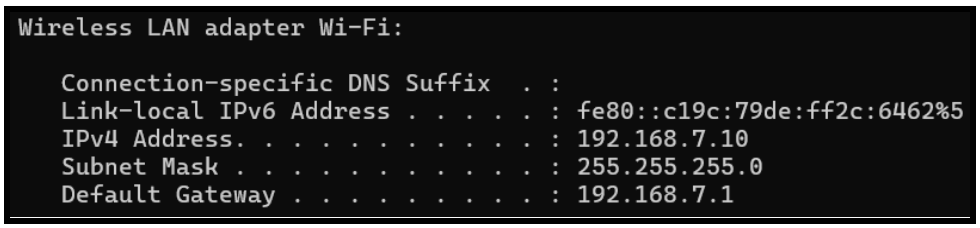
Screenshot:





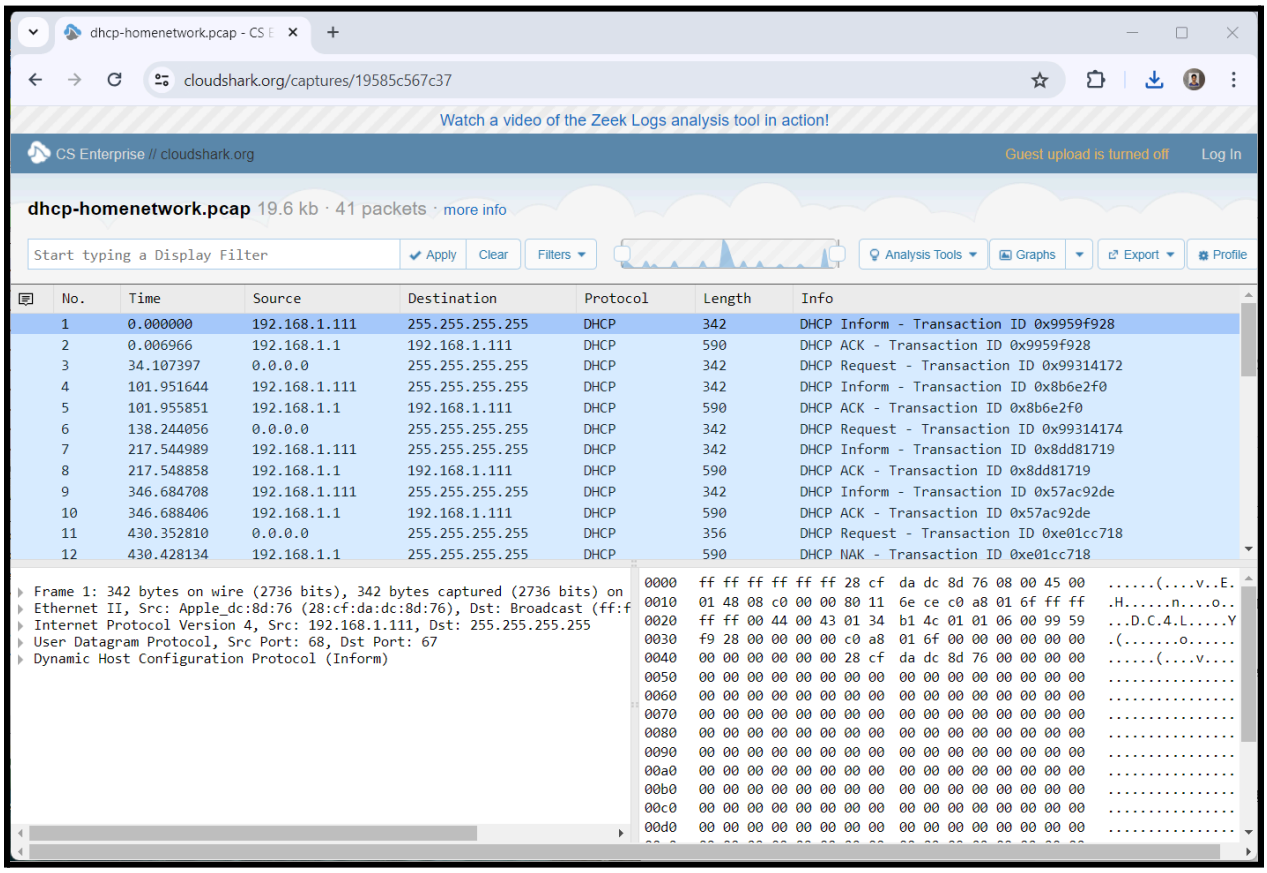
e. To what IP address is the DNS query message sent?

Screenshot:



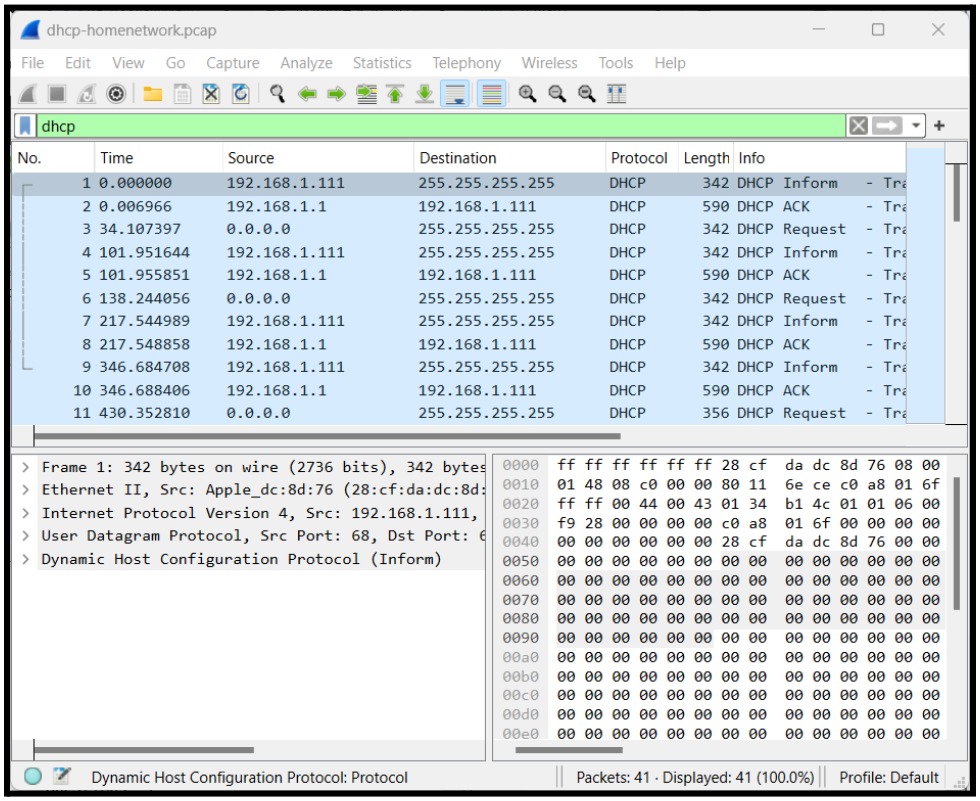
7. Download the dhcp.pcap file from https://www.cloudshark.org/captures/19585c567c37 (latest) and use Wireshark to answer the following questions:

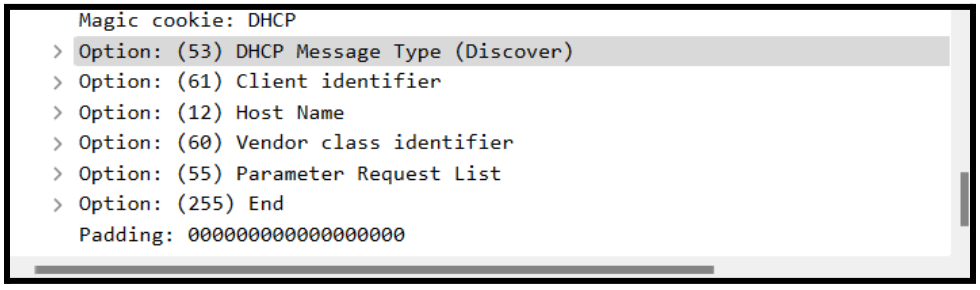
Screenshot:



a. Filter the packets using "dhcp." Let's start by looking at the DHCP Discover message. Locate the IP datagram containing the first Discover message in your trace.

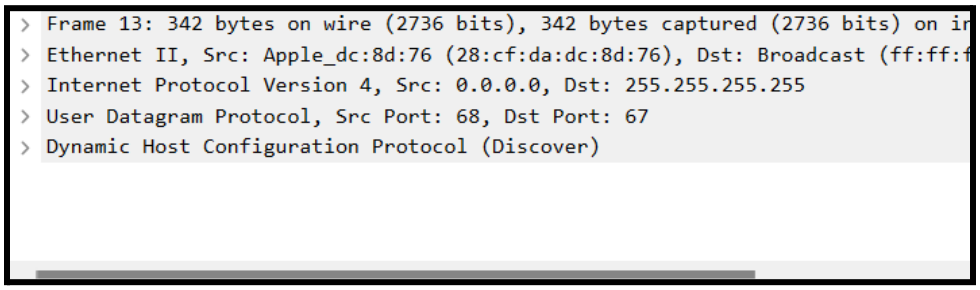
Screenshot:





b. Is this DHCP Discover message sent out using UDP or TCP as the underlying transport protocol?

Screenshot:

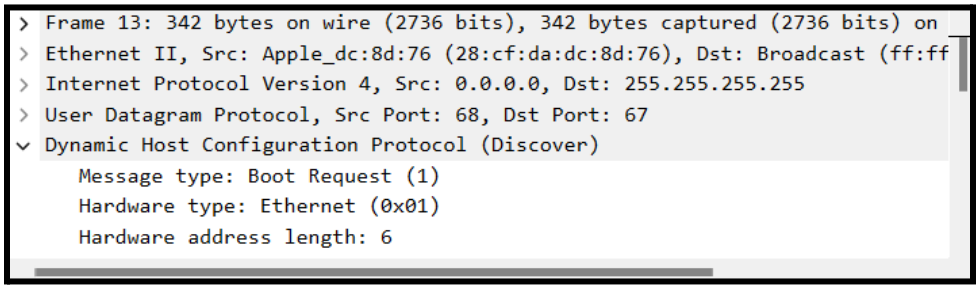


c. What are the source IP address and destination IP address used in the IP datagram containing the Discover message? Is there anything special about this address? Explain.

Theory:

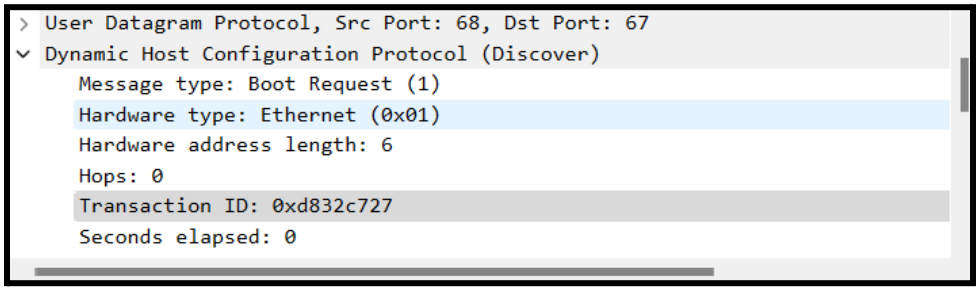
The source IP address used in the IP datagram containing the Discover message is 0.0.0.0 because the host doesn't have an IP address assigned to it yet. The DHCP client broadcasts a DHCPDISCOVER message on the network subnet using the destination address 255.255.255.255.

Screenshot:



d. What is the value in the transaction ID field of this DHCP Discover message?

Screenshot:

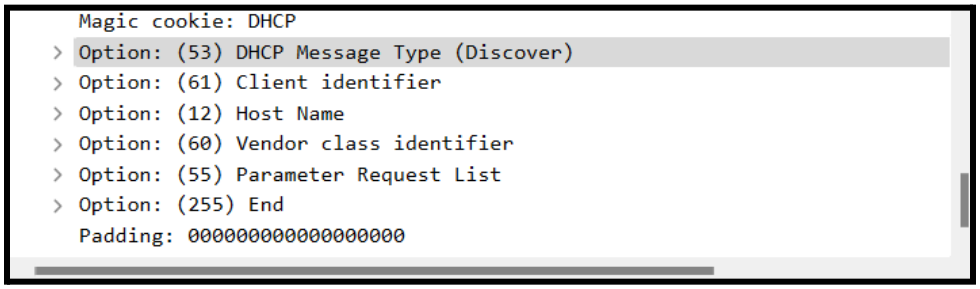


e. Now inspect the options field in the DHCP Discover message. What are five pieces of information (beyond an IP address) that the client is suggesting or requesting to receive from the DHCP server as part of this DHCP transaction?

Theory:

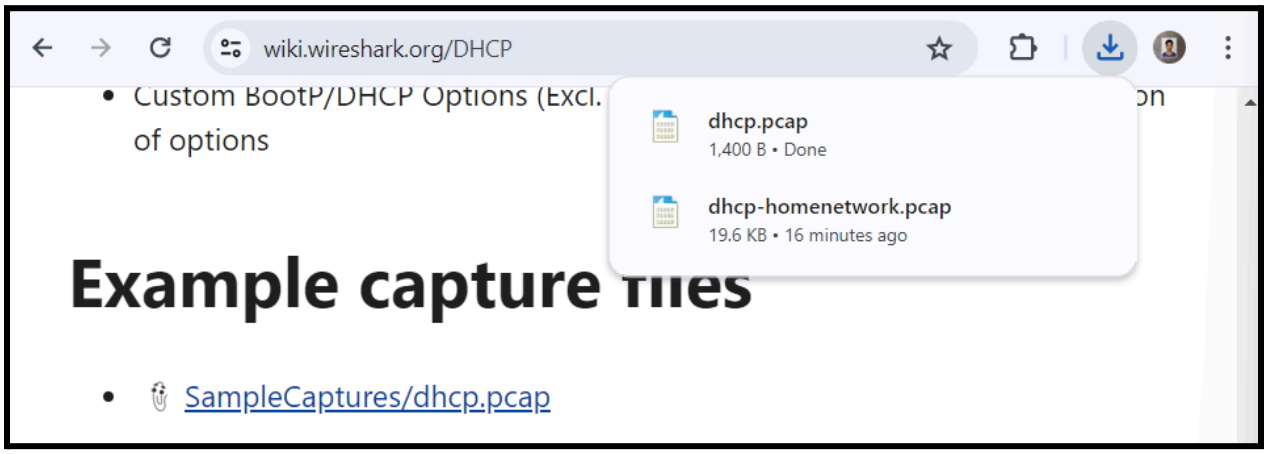
The DHCP options field that the client is asking the DHCP server to provide as part of the DHCP transaction will be: Parameter Request List, Maximum DHCP Message Size, Client Identifier, IP Address Lease Time, Host Name.

Screenshot:



8. Download the dhcp.pcap file from https://wiki.wireshark.org/DHCP and use Wireshark to answer the following questions:

Screenshot:

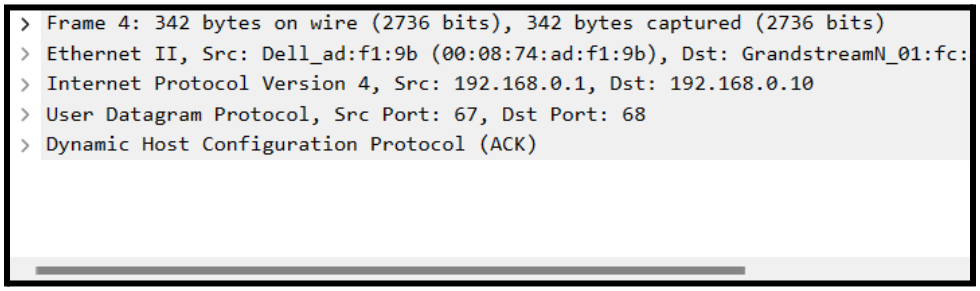


a. What is the source IP address in the IP datagram containing this ACK message? Is there anything special about this address? Explain.

Theory:

The source IP address in the IP datagram containing this ACK message is 192.168.0.1. This address belongs to the private IP address range reserved for internal networks, as defined in RFC 1918. It's commonly used in home or office networks for devices to communicate internally while being hidden behind a router or firewall. These addresses are not routable over the public internet, which helps improve network security by keeping internal network devices hidden from external threats.

Screenshot:

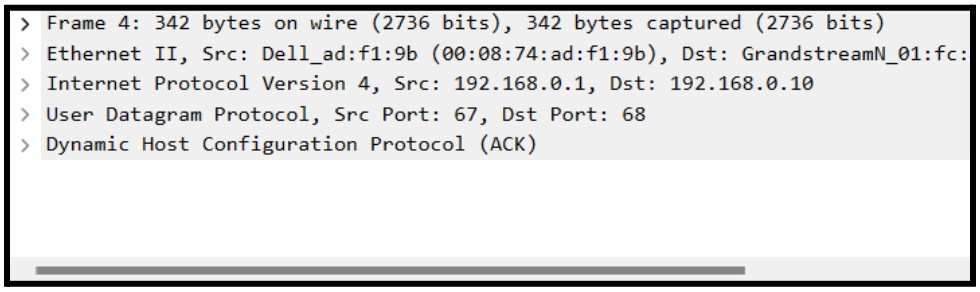


b. What is the destination IP address used in the datagram containing this ACK message? Is there anything special about this address? Explain.

Theory:

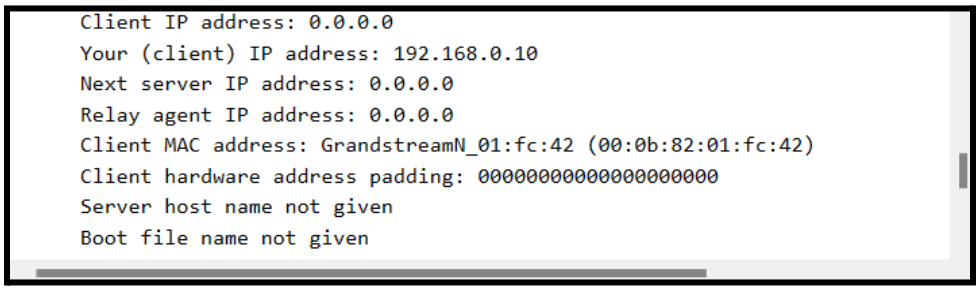
The destination IP address in the IP datagram containing this ACK message is 192.168.0.10. This address belongs to the same local network (192.168.0.0/24) as the source address (192.168.0.1), indicating that the communication is happening within the same local network. This is often referred to as local or internal communication. It typically doesn't traverse through routers or the internet, making it faster and more secure compared to communication across different networks.

Screenshot:



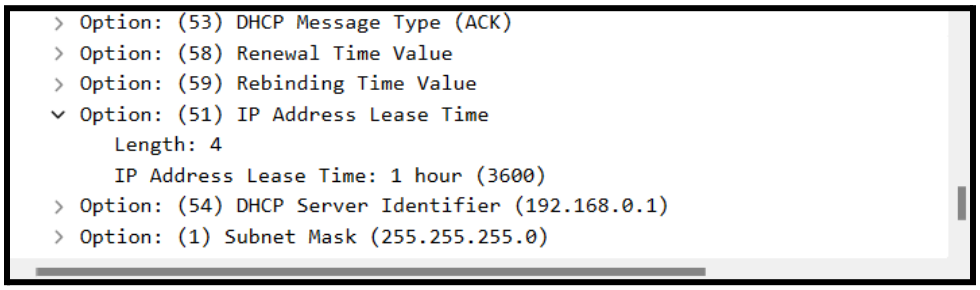
c. What is the name of the field in the DHCP ACK message (as indicated in the Wireshark window) that contains the assigned client IP address?

Screenshot:



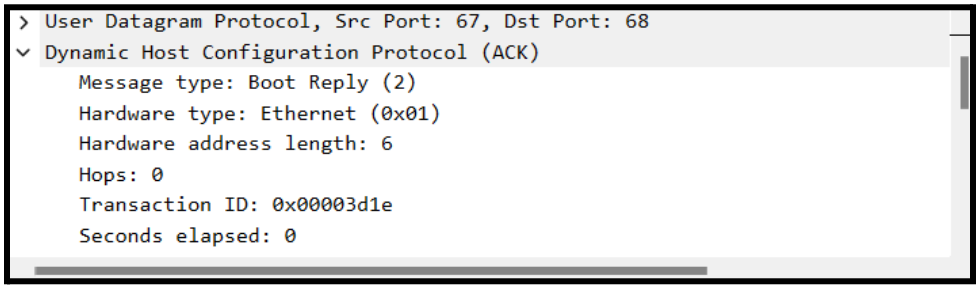
d. For how long a time (the so-called "lease time") has the DHCP server assigned this IP address to the client?

Screenshot:



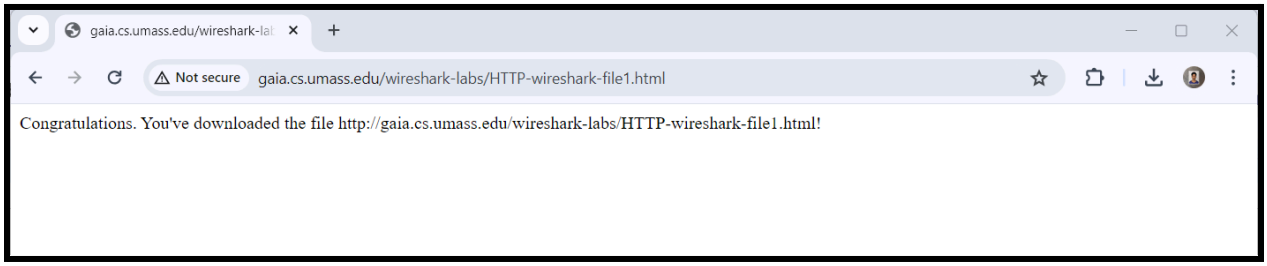
e. What is the IP address (returned by the DHCP server to the DHCP client in this DHCP ACK message) of the first hop router on the default path from the client to the rest of the Internet?

Screenshot:



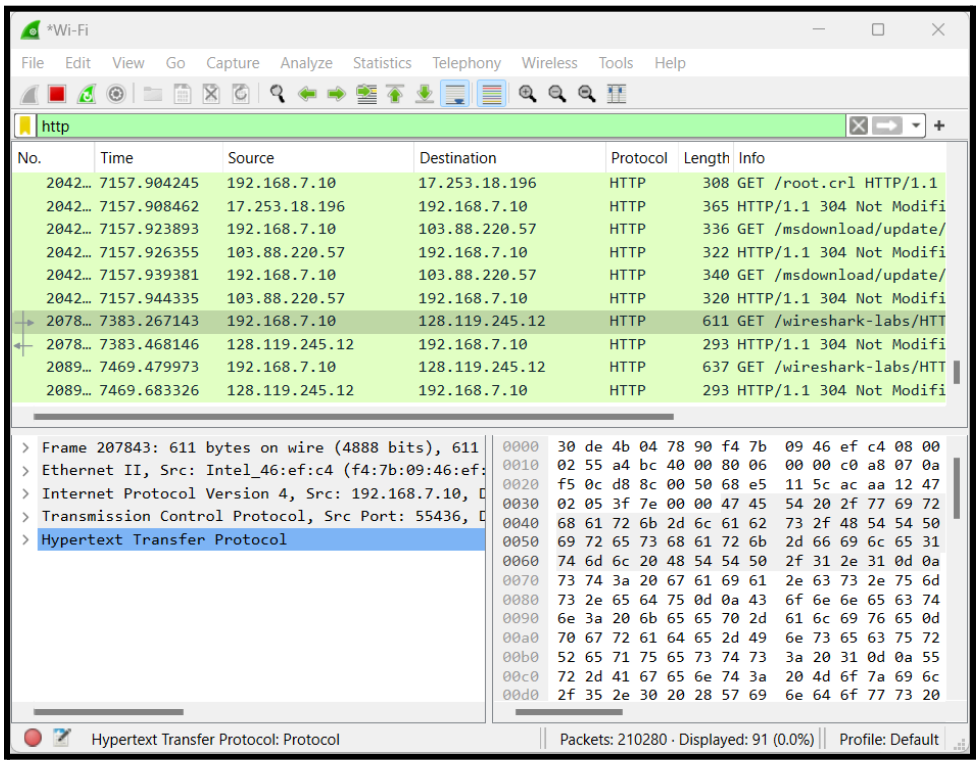
9. Use Wireshark to capture packets by browsing any website and answer the following questions based on the contents of the Ethernet frame containing the HTTP GET message from http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file1.html:

Screenshot:



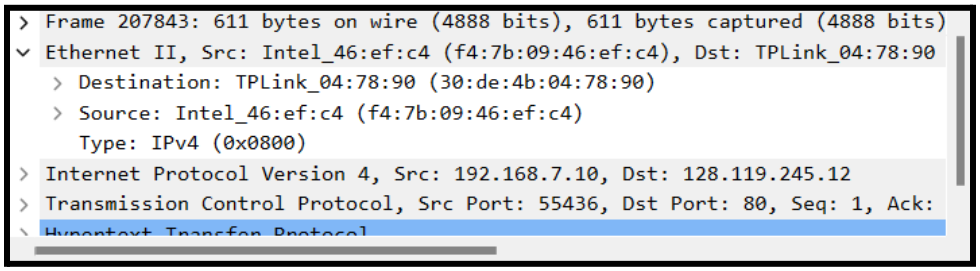
a. Filter the packets using "http." Answer the following questions, based on the contents of the Ethernet frame containing the HTTP GET message.

Screenshot:



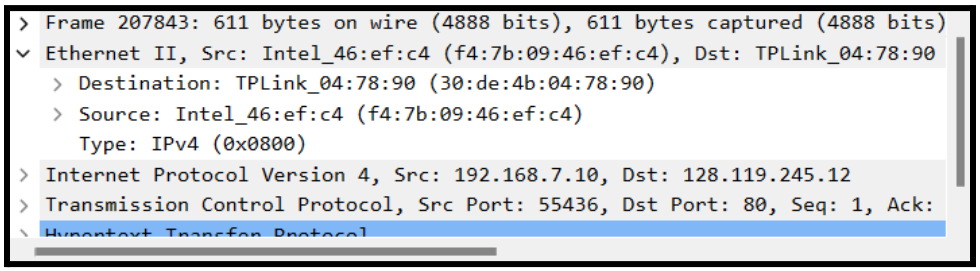
b. What is the 48-bit Ethernet address of your computer?

Screenshot:



c. What is the 48-bit destination address in the Ethernet frame?

Screenshot:



d. Is this the Ethernet address of gaia.cs.umass.edu? What device has this as its Ethernet address?

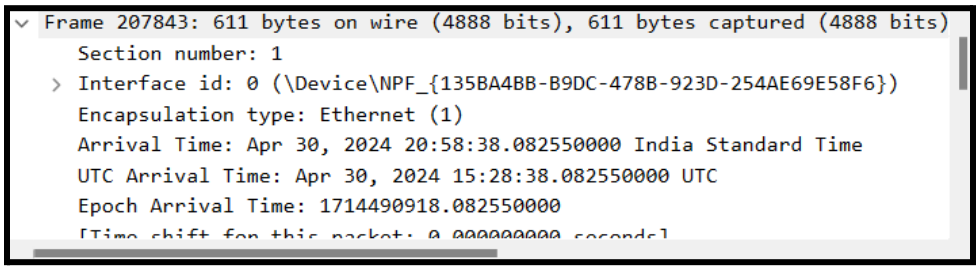
Theory:

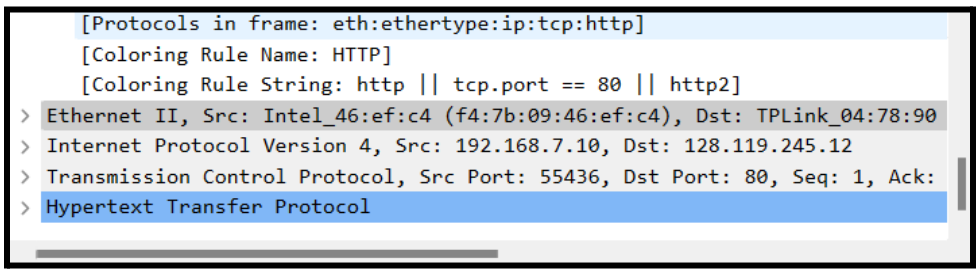
No, the Ethernet addresses provided are not directly linked to a hostname like "gaia.cs.umass.edu."

The Ethernet address "Intel_46:ef:c4 (f4:7b:09:46:ef:c4)" belongs to a device manufactured by Intel, and the Ethernet address "TPLink_04:78:90 (30:de:4b:04:78:90)" belongs to a device manufactured by TP-Link. Without additional context, it's not possible to determine which specific devices these addresses correspond to, as they could be network interface cards, routers, switches, or any other network-enabled devices.

e. Provide the hexadecimal value for the two-byte Frame type field. What upper layer protocol does this correspond to?

Screenshot:





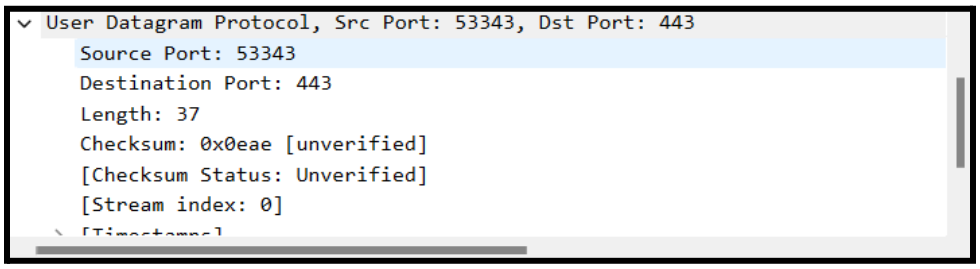
10. Use Wireshark to capture packets by browsing any website and answer the following questions:

a. Select any UDP packet. Find out how many fields are there in the UDP header and name these fields.

Theory:

The UDP header contains 4 fields: source port, destination port, length, and checksum.

Screenshot:



b. Determine the length of each of the UDP header fields.

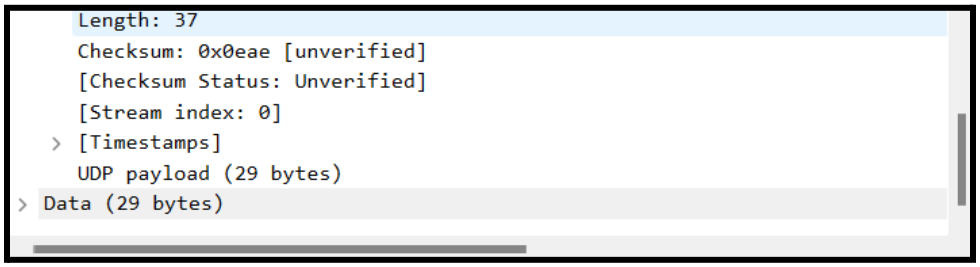
Theory:

From the packet content field, determine the length (in bytes) of each of the UDP header fields.

Each of the UDP header fields is 2 bytes long.

c. What is the value in the length field indicated? What is the length of the UDP payload?

Screenshot:



d. What is the maximum number of bytes that can be included in a UDP payload?

Theory:

The maximum number of bytes that can be included in a UDP payload is (2^16 – 1) bytes plus the header bytes. This gives 65535 bytes – 29 bytes = 65506 bytes.

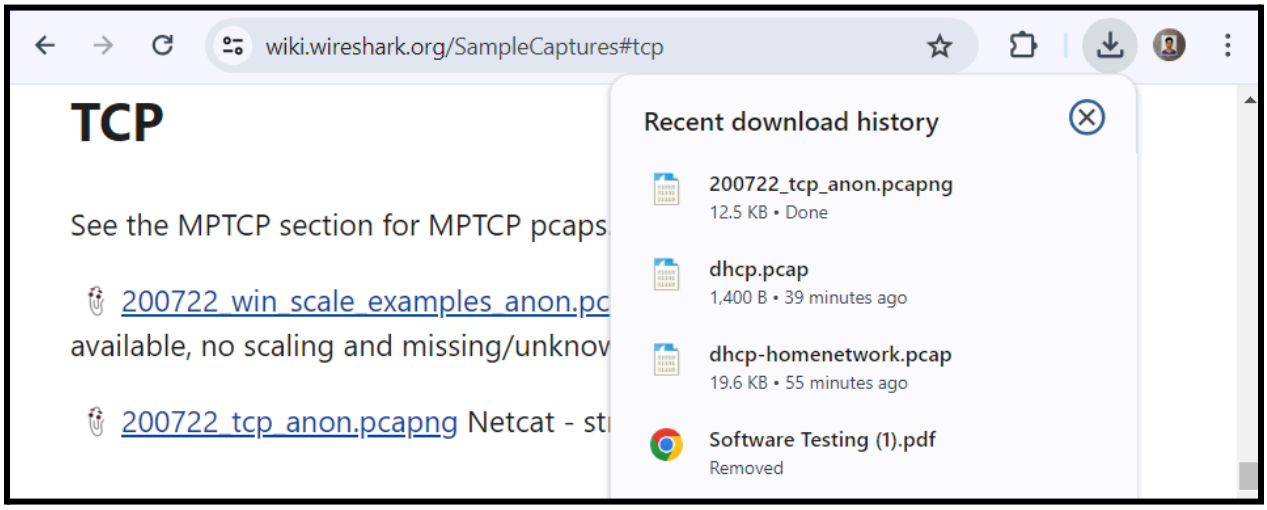
e. What are the largest possible source and destination port numbers?

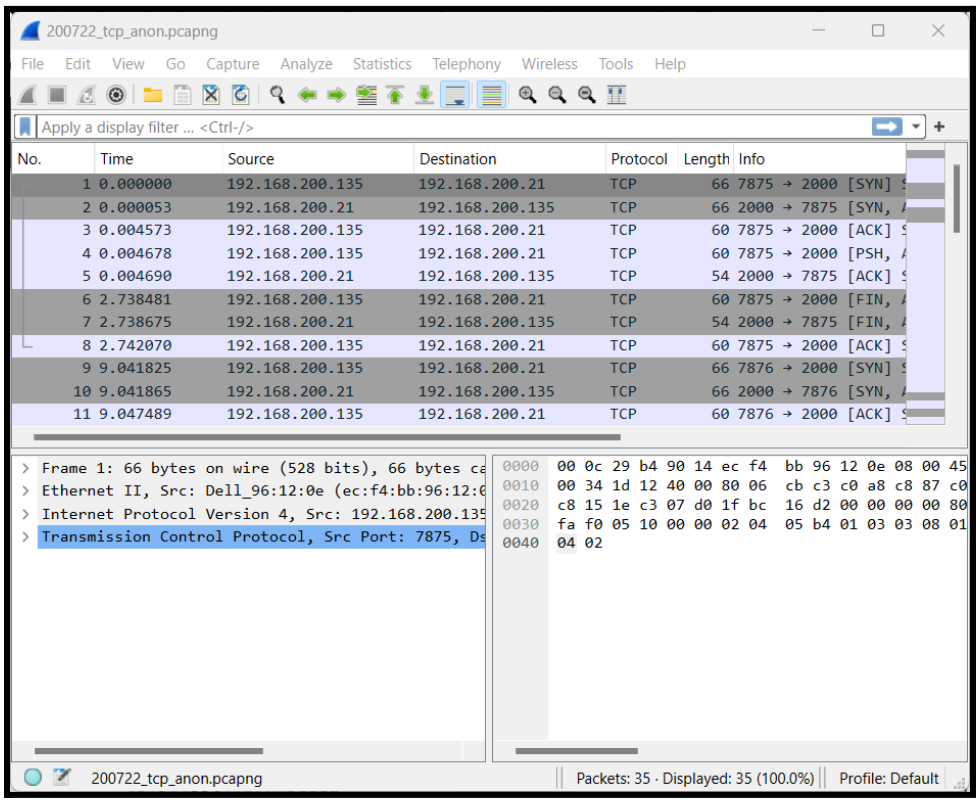
Theory:

The largest possible source and destination port numbers are both 65535 (2^16 - 1), as the port numbers range from 0 to 65535.

11. From https://wiki.wireshark.org/SampleCaptures#tcp, download the 200722_tcp_anon.pcap ng file. Use Wireshark to answer the following questions:

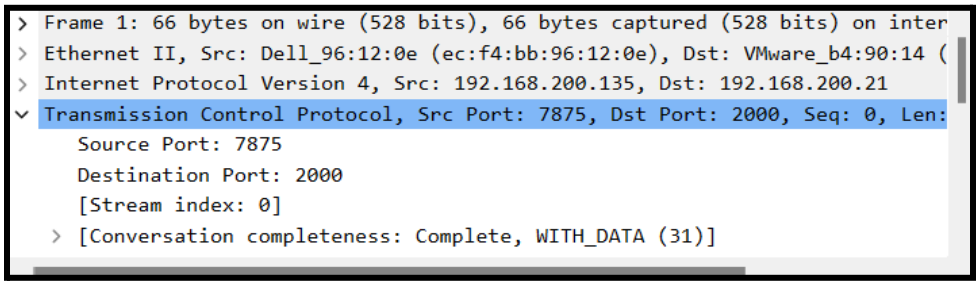
Screenshot:





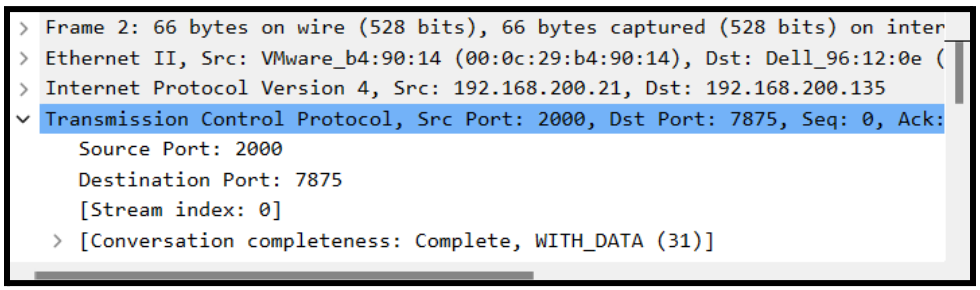
a. What is the IP address and TCP port number used by the client?

Screenshot:



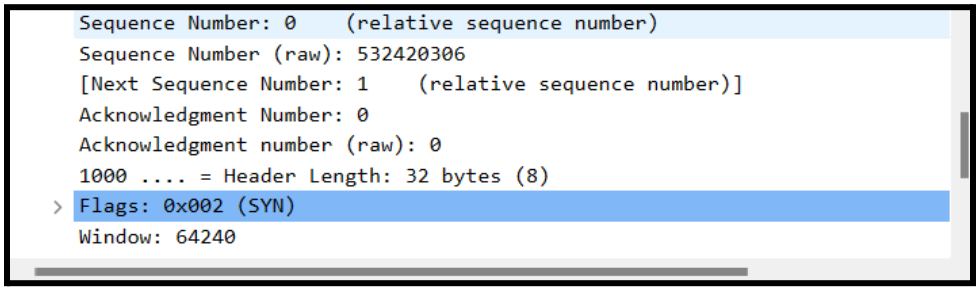
b. What is the IP address and TCP port number used by the server?

Screenshot:



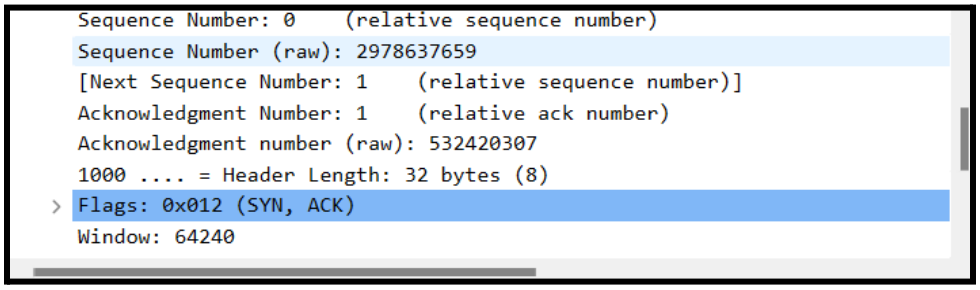
c. What is the sequence number of the TCP SYN segment that is used to initiate the TCP connection between the client computer and the server? What identifies the segment as a SYN segment?

Screenshot:



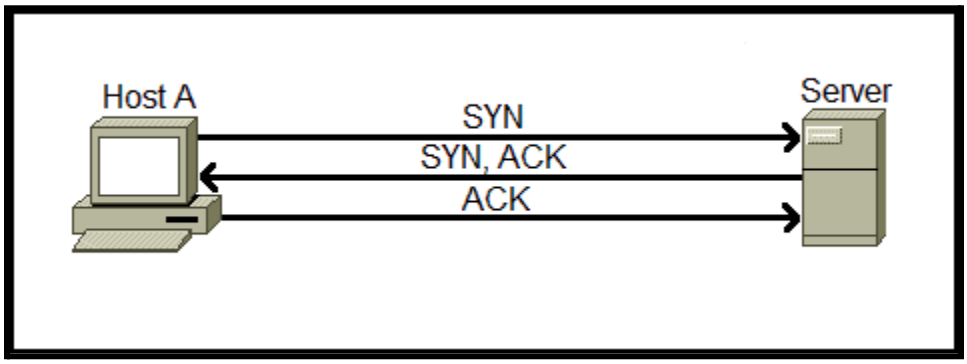
d. What is the sequence number of the TCP SYNACK segment sent by the server to the client computer in reply to the SYN? What is the value of the acknowledgment field in the SYNACK segment?

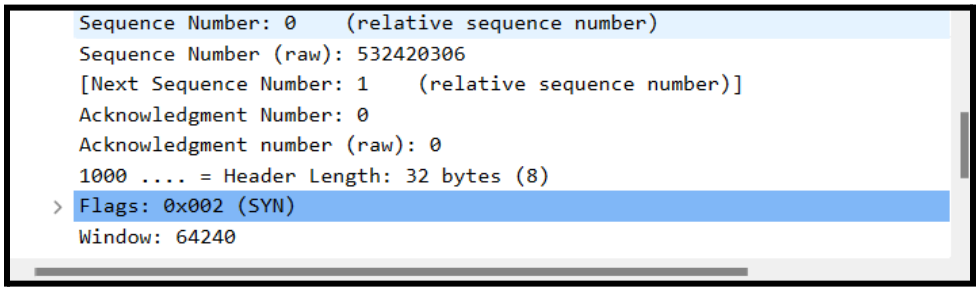
Screenshot:

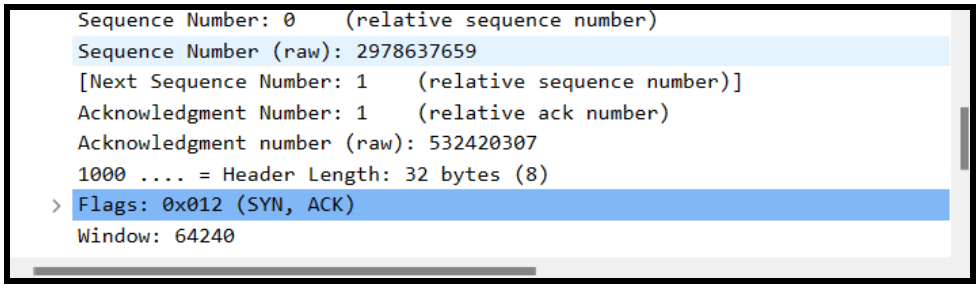


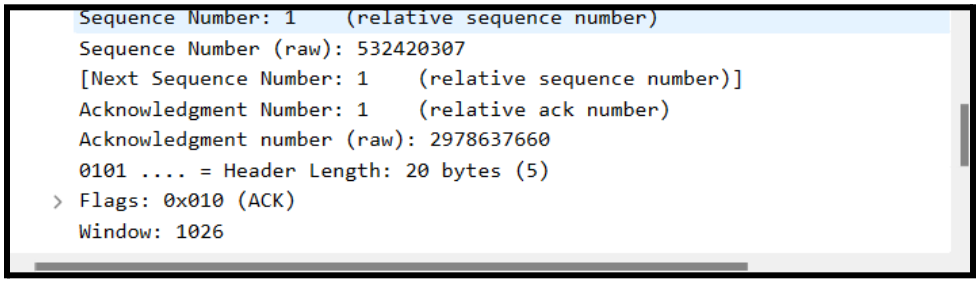
e. How did the server determine that value? What identifies the segment as a SYNACK segment?

Screenshot:






Sequence Number: 0 (relative sequence number)
Sequence Number (raw): 2978637659
[Next Sequence Number: 1 (relative sequence number)]
Acknowledgment Number: 1 (relative ack number)
Acknowledgment number (raw): 532420307
1000 .... = Header Length: 32 bytes (8)
Flags: 0x012 (SYN, ACK)
Window: 64240


Sequence Number: 1 (relative sequence number)
Sequence Number (raw): 532420307
[Next Sequence Number: 1 (relative sequence number)]
Acknowledgment Number: 1 (relative ack number)
Acknowledgment number (raw): 2978637660
0101 .... = Header Length: 20 bytes (5)
Flags: 0x010 (ACK)
Window: 1026