Lab 9.2.1.6 Using Wireshark to Observe the TCP 3-Way Handshake

1. Topology



1. Objectives

Part 1: Prepare Wireshark to Capture Packets

Part 2: Capture, Locate, and Examine Packets

1. Background / Scenario

In this lab, you will use Wireshark to capture and examine packets generated between the PC browser using the **HyperText Transfer Protocol (HTTP)** and a web server, such as [**www.google.com**](http://www.google.com). When an application, such as HTTP or File Transfer Protocol (FTP) first starts on a host, **TCP** uses the **three-way handshake** to establish a reliable TCP session between the two hosts. For example, when a PC uses a web browser to surf the Internet, a three-way handshake is initiated, and a session is established between the PC host and web server. A PC can have multiple, simultaneous, active TCP sessions with various web sites.

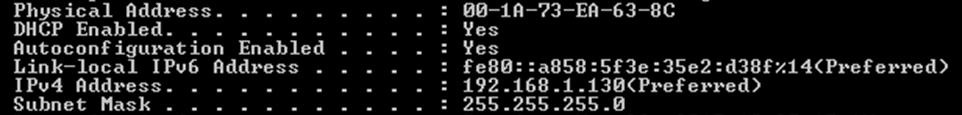
1. Prepare Wireshark to Capture Packets

In Part 1, you will start the Wireshark program and select the appropriate interface to begin capturing packets.

* 1. Retrieve the PC interface addresses.

For this lab, you need to retrieve your PC’s IP address and its network interface card (NIC) physical address, also called the MAC address.

* + 1. Open a command prompt window, type **ipconfig /all**, and press Enter.



* + 1. Write down the IP and MAC addresses associated with the selected Ethernet adapter. That is the source address to look for when examining captured packets.

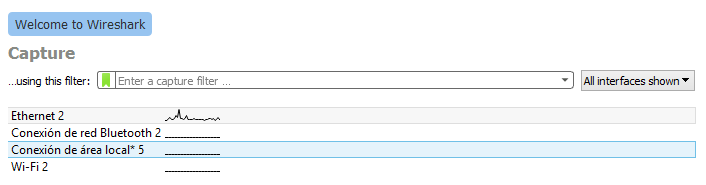
The PC host IP address: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The PC host MAC address: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Default gateway IP address: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

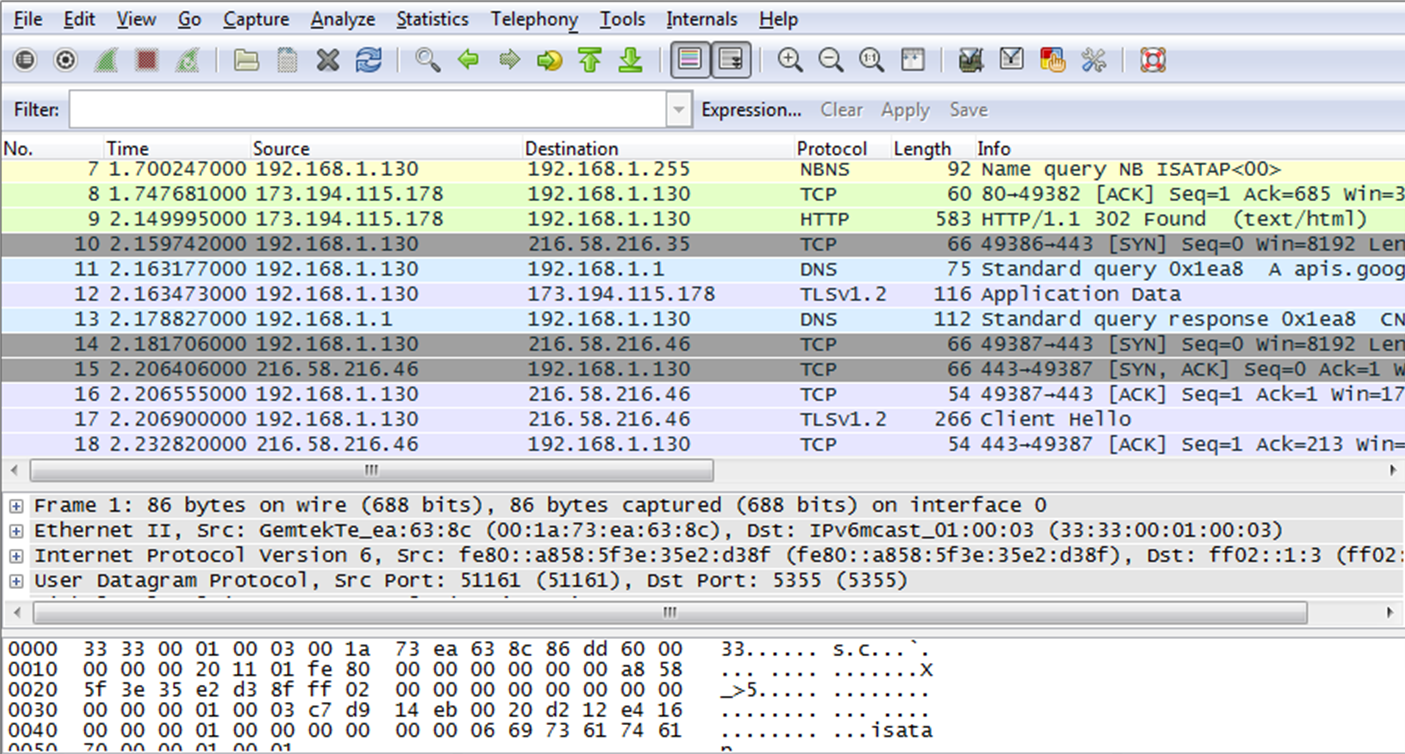
DNS server IP address: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* 1. Start Wireshark and select the appropriate interface.
     1. Start **Wireshark**.
     2. In the **Wireshark: Capture Interfaces** window, click to the interface that is connected to your LAN.



1. Capture, Locate, and Examine Packets
   1. Capture the data.
      1. Navigate to **www.google.com.** Minimize the browser and return to Wireshark. Stop the data capture.

The capture window is now active. Locate the **Source**, **Destination**, and **Protocol** columns.



* 1. Locate appropriate packets for the web session.

If the computer was recently started and there has been no activity in accessing the Internet, you can see the entire process in the captured output, including the **Address Resolution Protocol (ARP)**, **Domain Name System (DNS)**, and the **TCP three-way handshake**. If the PC already had an ARP entry for the default gateway; therefore, it started with the DNS query to resolve www.google.com.

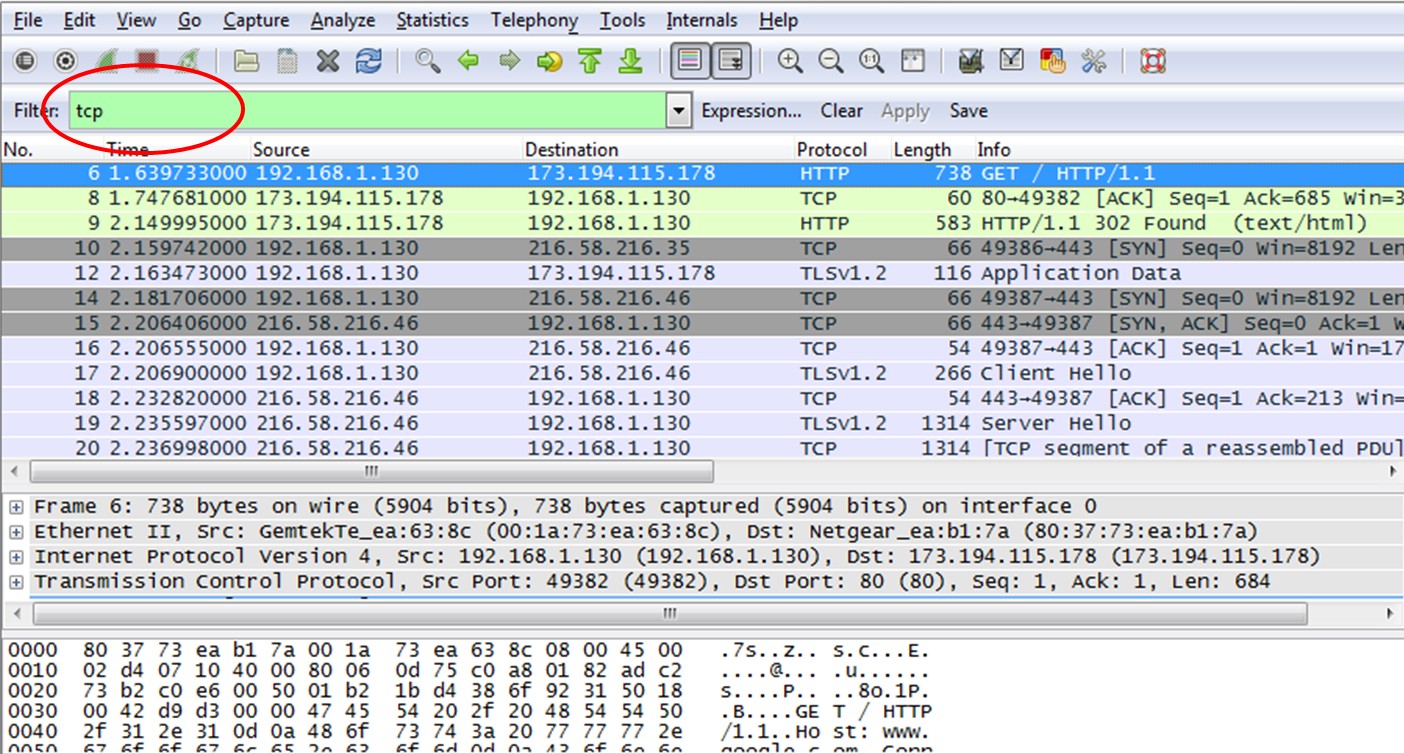
* + 1. **Frame 11** shows the DNS query from the PC to the DNS server, which is attempting to resolve the domain name **www.google.com** to the IP address of the web server. The PC must have the IP address before it can send the first packet to the web server.

What is the IP address of the **DNS server** that the computer queried? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

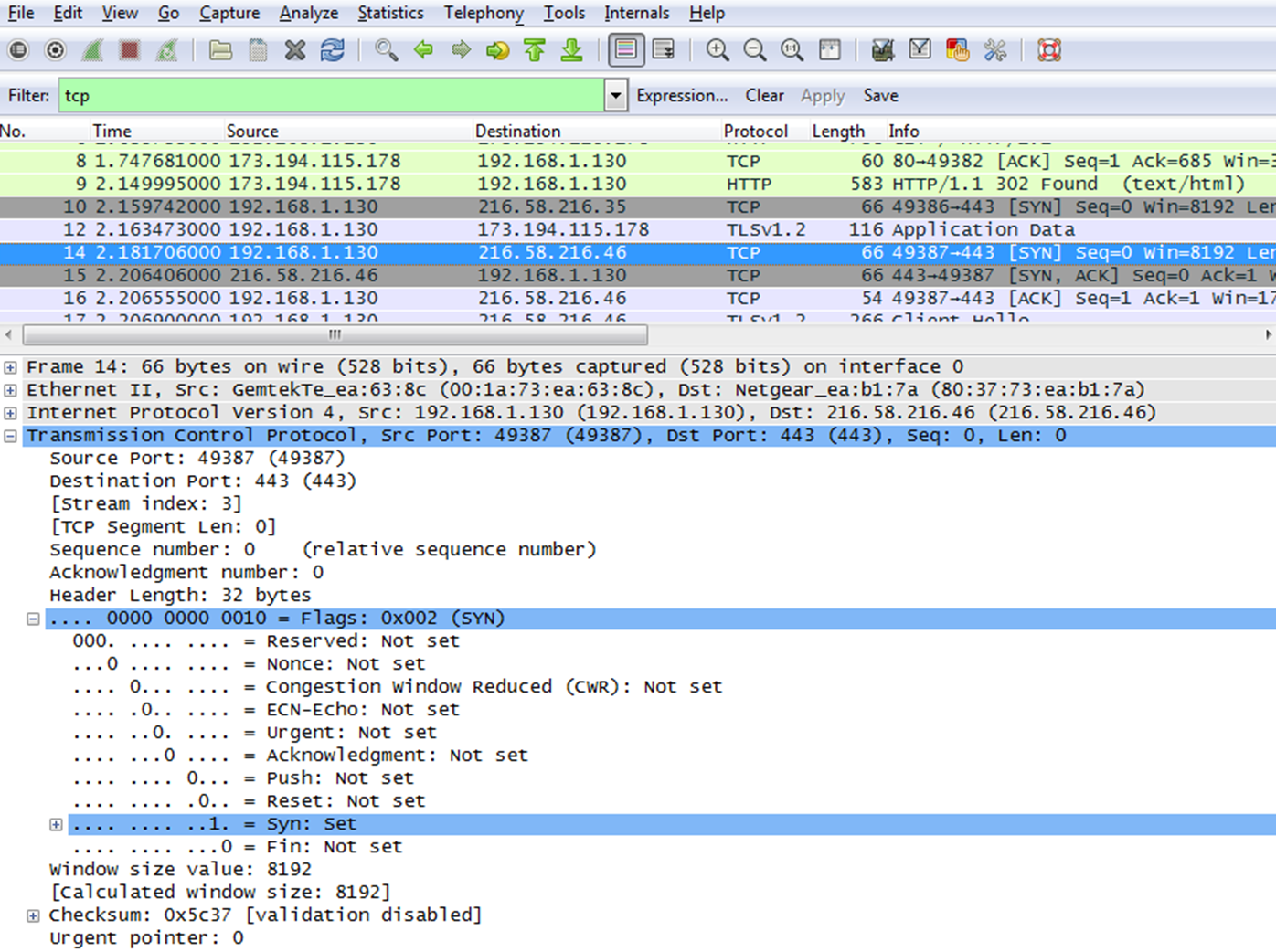
* + 1. **Frame 13** is the response from the DNS server. It contains the IP address of www.google.com.
    2. Find the appropriate packet for the start of your **three-way handshake**. In the example, **frame 14** is the start of the TCP three-way handshake.

What is the IP address of the **Google web server**? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* + 1. If you have many packets that are unrelated to the TCP connection, it may be necessary to use the Wireshark filter tool. Type **tcp** in the filter entry area within Wireshark and press **Enter**.



* 1. Examine the information within packets including IP addresses, TCP port numbers, and TCP control flags.
     1. In our example, frame 14 is the start of the three-way handshake between the PC and the Google web server. In the packet list pane (top section of the main window), select the frame. This highlights the line and displays the decoded information from that packet in the two lower panes. Examine the TCP information in the packet details pane (middle section of the main window).
     2. Click the **+** icon to the left of the Transmission Control Protocol in the packet details pane to expand the view of the TCP information.
     3. Click the **+** icon to the left of the Flags. Look at the source and destination ports and the flags that are set.



What is the TCP source port number? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

How would you classify the source port? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

What is the TCP destination port number? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

How would you classify the destination port? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Which flag (or flags) is set? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

What is the relative sequence number set to? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* + 1. To select the next frame in the three-way handshake, select **Go** on the Wireshark menu and select **Next Packet In Conversation**. In this example, this is frame 15. This is the Google web server reply to the initial request to start a session.



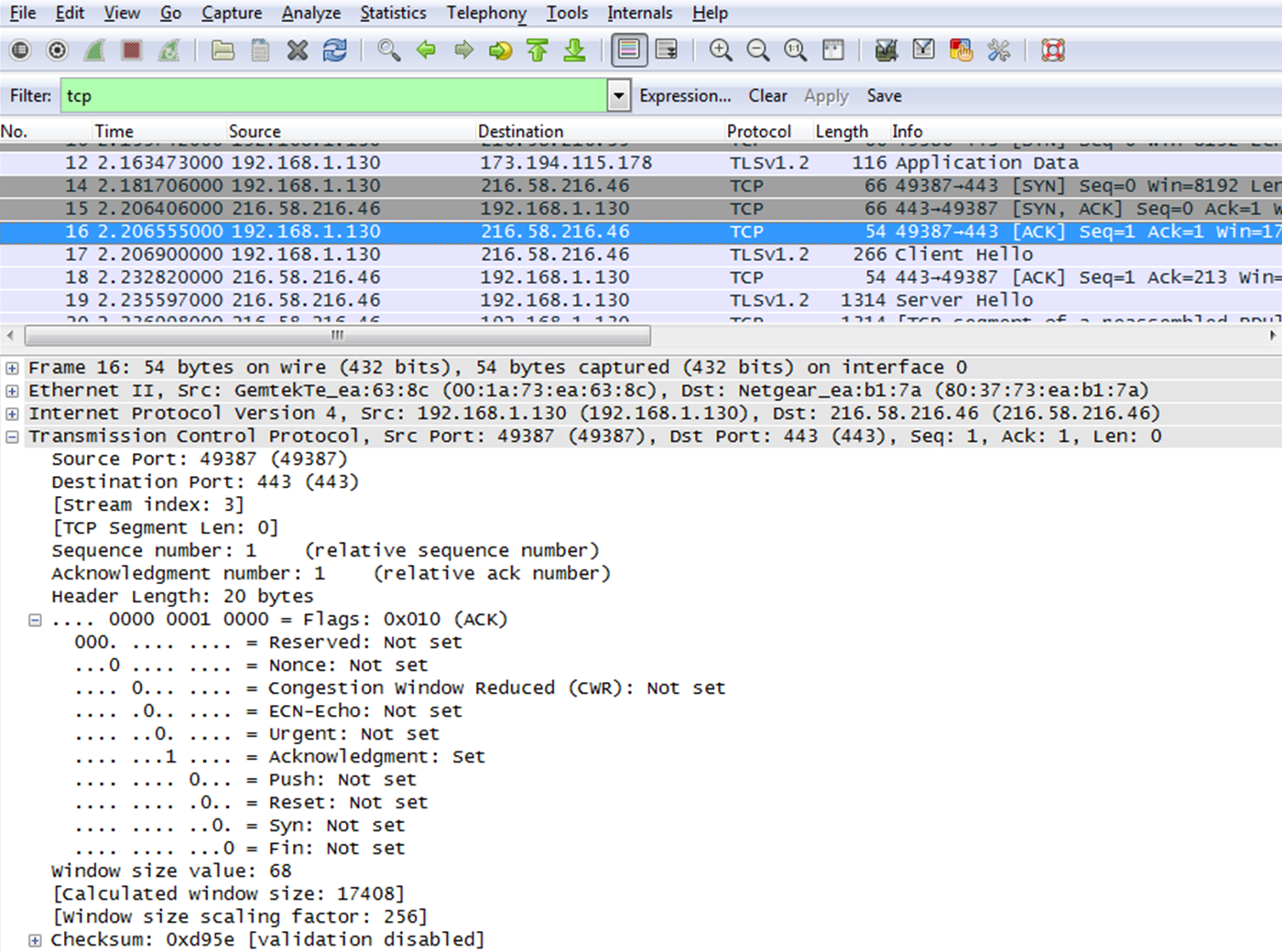
What are the values of the source and destination ports? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Which flags are set? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

What is the relative sequence set to? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

What is the acknowledgement number set to? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* + 1. Finally, examine the third packet of the three-way handshake in the example. Click frame 16 in the top window to display the following information in this example:



Examine the third and final packet of the handshake.

Which flag (or flags) is set? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The relative sequence and acknowledgement numbers are set to 1 as a starting point. The TCP connection is established and communication between the source computer and the web server can begin.

Lab 9.2.3.5 Using Wireshark to Examine a UDP DNS Capture

1. Objectives

Part 1: Use Wireshark to Capture DNS Queries and Responses

Part 2: Analyze Captured DNS or UDP Packets

1. Background / Scenario

If you have ever used the Internet, you have used the **Domain Name System (DNS).** DNS is a distributed network of servers that translates user-friendly domain names like www.google.com to an IP address. When you type a website URL into your browser, your PC performs a DNS query to the DNS server’s IP address. Your PC’s DNS server query and the DNS server’s response make use of the User Datagram Protocol (UDP) as the transport layer protocol. UDP is connectionless and does not require a session setup as does TCP. DNS queries and responses are very small and do not require the overhead of TCP.

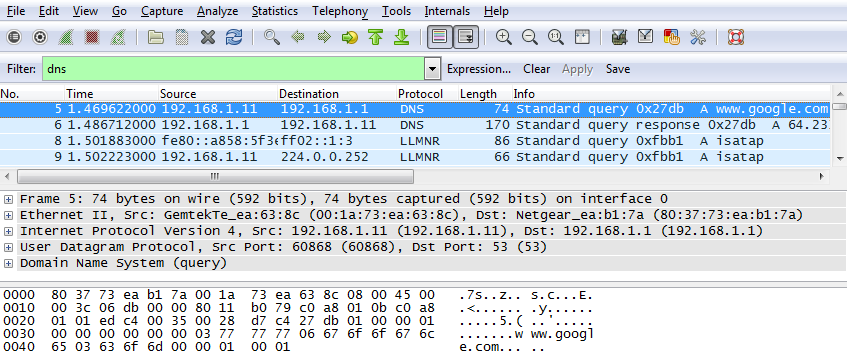
In this lab, you will communicate with a DNS server by sending a DNS query using the UDP transport protocol. You will use Wireshark to examine the DNS query and response exchanges with the same server.

1. Analyze Captured DNS or UDP Packets

In Part 1, you will examine the UDP packets that were generated when communicating with a DNS server for the IP addresses for www.google.com.

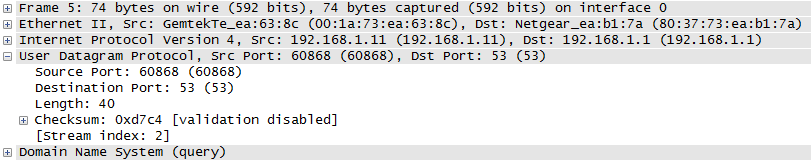
* 1. Filter DNS packets.
     1. In the Wireshark main window, type **dns** in the entry area of the **Filter** toolbar. Click **Apply** or press **Enter**.

**Note**: If you do not see any results after the DNS filter was applied, close the web browser. In the command prompt window, type **ipconfig /flushdns** to remove all previous DNS results. Restart the Wireshark capture and repeat the instructions in Part 2b –2e. If this does not resolve the issue, type **nslookup** [**www.google.com**](http://www.google.com) in the command prompt window as an alternative to the web browser.



* + 1. In the packet list pane (top section) of the main window, locate the packet that includes **Standard query** and **A www.google.com**. See frame 5 as an example.
  1. Examine a UDP segment using DNS query.

Examine the **UDP** by using a DNS query for **www.google.com** as captured by Wireshark. In this example, Wireshark capture **frame 5** in the packet list pane is selected for analysis. The protocols in this query are displayed in the packet details pane (middle section) of the main window. The protocol entries are highlighted in gray.



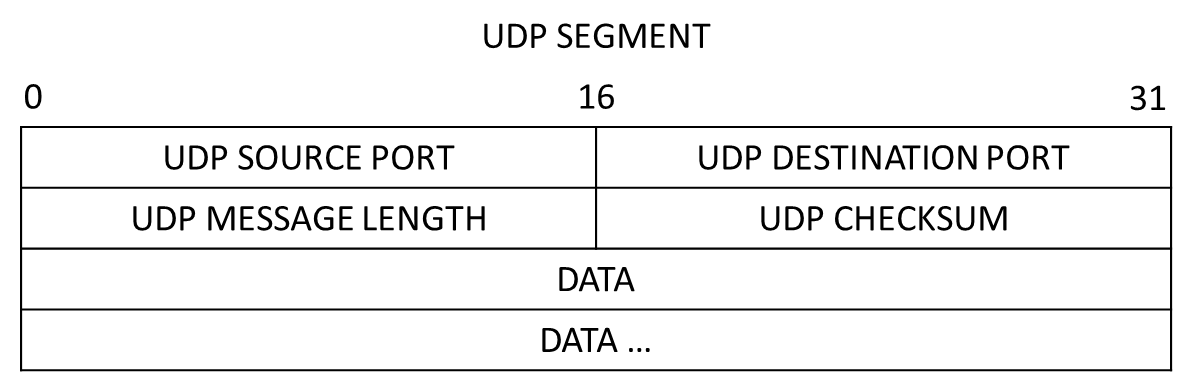
* + 1. In the first line in the packet details pane, **frame 5** had **74 bytes** of data on the wire. This is the number of bytes to send a DNS query to a named server requesting the IP addresses of www.google.com.
    2. The **Ethernet II** line displays the source and destination MAC addresses. The source MAC address is from your local PC because your local PC originated the DNS query. The destination MAC address is from the **default gateway** because this is the last stop before this query exits the local network.

Is the source MAC address the same as the one recorded from Part 1 for the local PC? \_\_\_\_\_\_\_\_\_\_\_\_\_\_

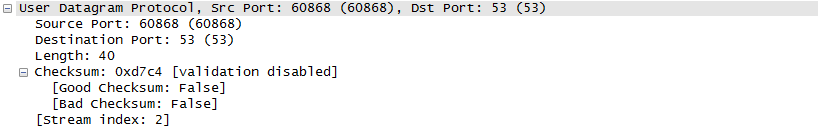
* + 1. In the Internet Protocol Version 4 line, the IP packet Wireshark capture indicates that the source IP address of this DNS query is **192.168.1.11** and the destination IP address is **192.168.1.1**. In this example, the destination address is the default gateway. The router is the default gateway in this network.

The IP packet and header encapsulates the UDP segment. The UDP segment contains the DNS query as the data.

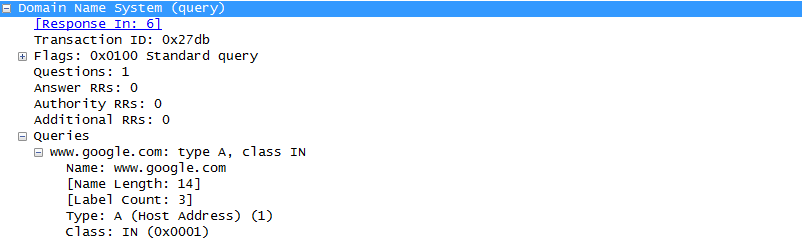
* + 1. A UDP header only has four fields: source port, destination port, length, and checksum. Each field in a UDP header is only 16 bits as depicted below.



Expand the User Datagram Protocol in the packet details pane by clicking the plus (+) sign. Notice that there are only four fields. The source port number in this example is 60868. The source port was randomly generated by the local PC using port numbers that are not reserved. The destination port is 53. Port 53 is a well-known port reserved for use with DNS. DNS servers listen on port 53 for DNS queries from clients.



In this example, the length of the UDP segment is 40 bytes. Out of 40 bytes, 8 bytes are used as the header. The other 32 bytes are used by DNS query data. The 32 bytes of DNS query data is highlighted in the following illustration in the packet bytes pane (lower section) of the Wireshark main window.



The checksum is used to determine the integrity of the packet after it has traversed the Internet.

The UDP header has low overhead because UDP does not have fields that are associated with the three-way handshake in TCP. Any data transfer reliability issues that occur must be handled by the application layer.

Record your Wireshark results in the table below:

|  |  |
| --- | --- |
| Frame size |  |
| Source MAC address |  |
| Destination MAC address |  |
| Source IP address |  |
| Destination IP address |  |
| Source port |  |
| Destination port |  |

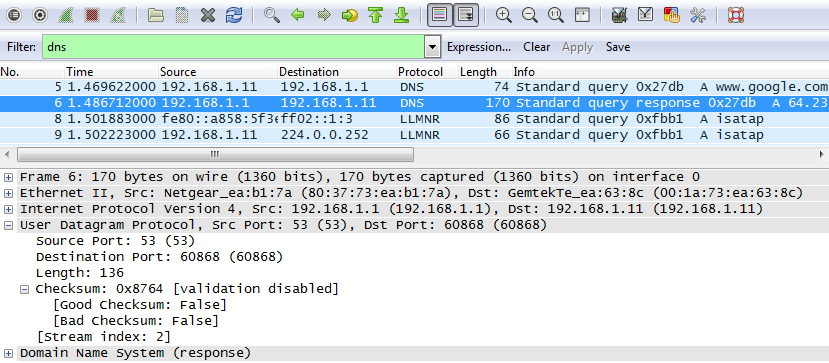
Is the source IP address the same as the local PC’s IP address you recorded in Part 1? \_\_\_\_\_\_\_\_\_\_\_\_\_

Is the destination IP address the same as the default gateway noted in Part 1? \_\_\_\_\_\_\_\_\_\_\_\_\_

* 1. Examine a UDP using DNS response.

In this step, you will examine the DNS response packet and verify that the DNS response packet also uses the UDP.

* + 1. In this example, frame 6 is the corresponding DNS response packet. Notice the number of bytes on the wire is 170. It is a larger packet compared to the DNS query packet.



* + 1. In the Ethernet II frame for the DNS response, what device is the source MAC address and what device is the destination MAC address?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* + 1. Notice the source and destination IP addresses in the IP packet. What is the destination IP address? What is the source IP address?

Source IP address: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Destination IP address: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

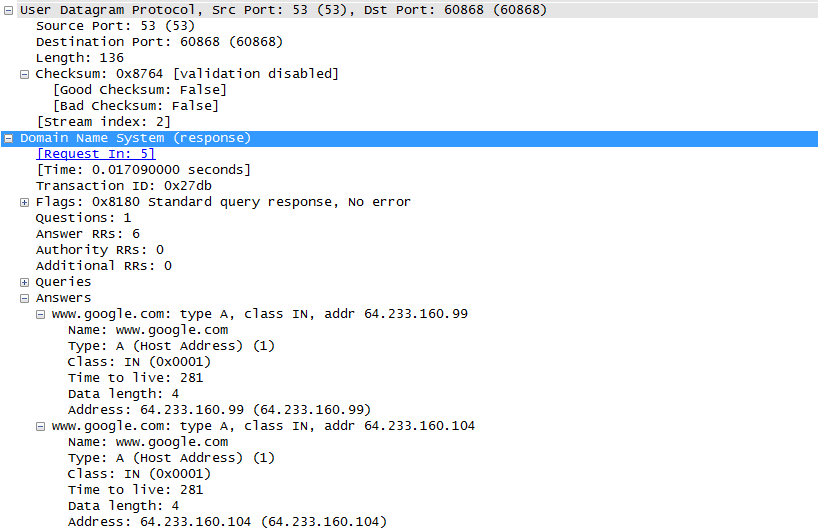
What happened to the roles of source and destination for the local host and default gateway?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* + 1. In the UDP segment, the role of the port numbers has also reversed. The destination port number is 60868. Port number 60868 is the same port that was generated by the local PC when the DNS query was sent to the DNS server. Your local PC listens for a DNS response on this port.

The source port number is 53. The DNS server listens for a DNS query on port 53 and then sends a DNS response with a source port number of 53 back to the originator of the DNS query.

When the DNS response is expanded, notice the resolved IP addresses for www.google.com in the **Answers** section.



1. Reflection

What are the benefits of using UDP instead of TCP as a transport protocol for DNS?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Lab 9.2.4.3 Using Wireshark to Examine TCP Captures



1. Objectives

Part 1: Identify TCP Header Fields and Operation Using a Wireshark FTP Session Capture

1. Background / Scenario

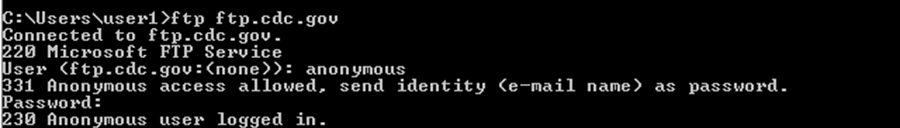
Two protocols in the TCP/IP **transport layer** are **TCP** and **UDP**. Both protocols support upper-layer protocol communication. For example, TCP is used to provide transport layer support for the HyperText Transfer Protocol (HTTP) and FTP protocols, among others. UDP provides transport layer support for the Domain Name System (DNS) and TFTP, among others.

In this lab, you will use the Wireshark open source tool to capture and analyze TCP protocol header fields for FTP file transfers between the host computer and an anonymous FTP server. The Windows command line utility is used to connect to an anonymous FTP server and download a file.

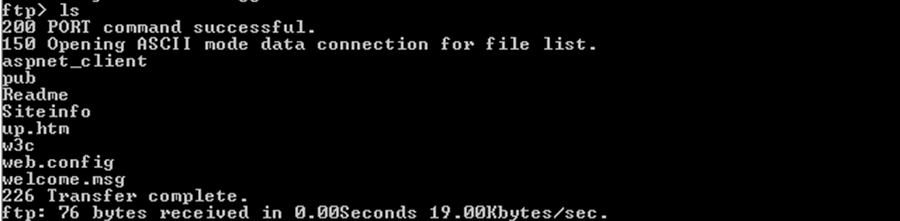
1. Identify TCP Header Fields and Operation Using a Wireshark FTP Session Capture

In Part 1, you use Wireshark to capture an FTP session and inspect TCP header fields.

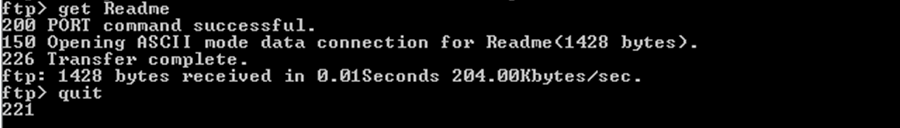
* 1. Start a Wireshark capture.
     1. Start the Wireshark capture.
  2. Download the Readme file.
     1. From the command prompt, enter **ftp ftp.cdc.gov**.
     2. Log into the FTP site for Centers for Disease Control and Prevention (CDC) with user **anonymous** and no password.



* + 1. Locate and download the Readme file by entering the **ls** command to list the files.

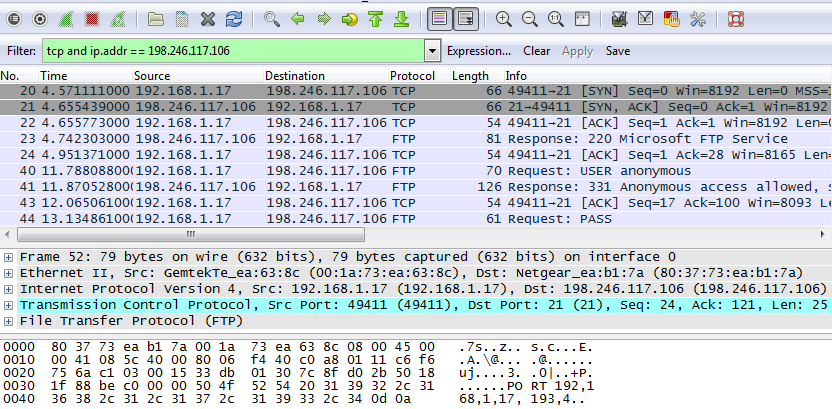


* + 1. Enter the command **get Readme** to download the file. When the download is complete, enter the command **quit** to exit.



* 1. Stop the Wireshark capture.
  2. View the Wireshark main window.

Wireshark captured many packets during the FTP session to ftp.cdc.gov. To limit the amount of data for analysis, type **tcp and ip.addr == 198.246.117.106** in the **Filter: entry** area and click **Apply**. The IP address, 198.246.117.106, is the address for [ftp.cdc.gov](ftp://ftp.cdc.gov) at this time.



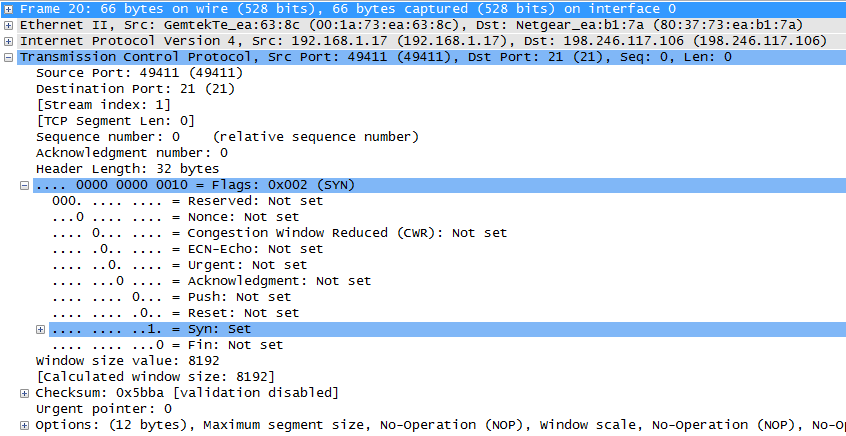
* 1. Analyze the TCP fields.

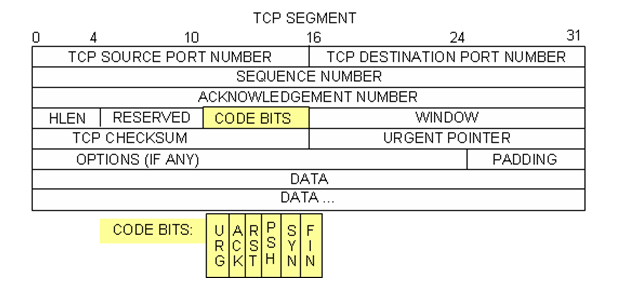
After the TCP filter has been applied, the first three frames in the packet list pane (top section) display the transport layer protocol TCP creating a reliable session. The sequence of **[SYN]**, **[SYN, ACK]**, and **[ACK]** illustrates the three-way handshake.



TCP is routinely used during a session to control datagram delivery, verify datagram arrival, and manage window size. For each data exchange between the FTP client and FTP server, a new TCP session is started. At the conclusion of the data transfer, the TCP session is closed. When the FTP session is finished, TCP performs an orderly shutdown and termination.

In Wireshark, detailed TCP information is available in the packet details pane (middle section). Highlight the first TCP datagram from the host computer, and expand the TCP datagram. The expanded TCP datagram appears similar to the packet detail pane shown below.





The image above is a TCP datagram diagram. An explanation of each field is provided for reference:

* The **TCP source port number** belongs to the TCP session host that opened a connection. The value is normally a random value above 1,023.
* The **TCP destination port number** is used to identify the upper layer protocol or application on the remote site. The values in the range 0–1,023 represent the “well-known ports” and are associated with popular services and applications (as described in RFC 1700), such as Telnet, FTP, and HTTP. The combination of the source IP address, source port, destination IP address, and destination port uniquely identifies the session to the sender and receiver.

**Note**: In the Wireshark capture below, the destination port is 21, which is FTP. FTP servers listen on port 21 for FTP client connections.

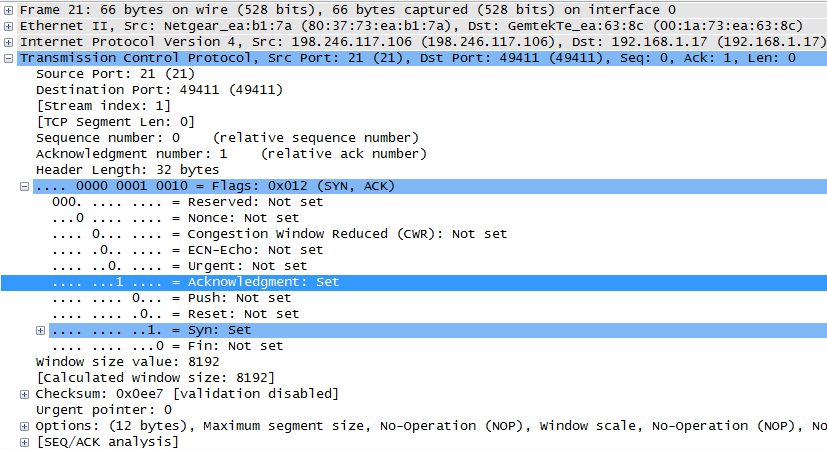
* The **Sequence number** specifies the number of the last octet in a segment.
* The **Acknowledgment number** specifies the next octet expected by the receiver.
* The **Code bits** have a special meaning in session management and in the treatment of segments. Among interesting values are:
  1. ACK — Acknowledgement of a segment receipt.
  2. SYN — Synchronize, only set when a new TCP session is negotiated during the TCP three-way handshake.
  3. FIN — Finish, the request to close the TCP session.
* The **Window size** is the value of the sliding window. It determines how many octets can be sent before waiting for an acknowledgement.
* The **Urgent pointer** is only used with an Urgent (URG) flag when the sender needs to send urgent data to the receiver.
* The **Options** has only one option currently, and it is defined as the maximum TCP segment size (optional value).

Using the Wireshark capture of the first TCP session startup (SYN bit set to 1), fill in information about the TCP header.

From the PC to CDC server (only the SYN bit is set to 1):

|  |  |
| --- | --- |
| Source IP address |  |
| Destination IP address |  |
| Source port number |  |
| Destination port number |  |
| Sequence number |  |
| Acknowledgement number |  |
| Header length |  |
| Window size |  |

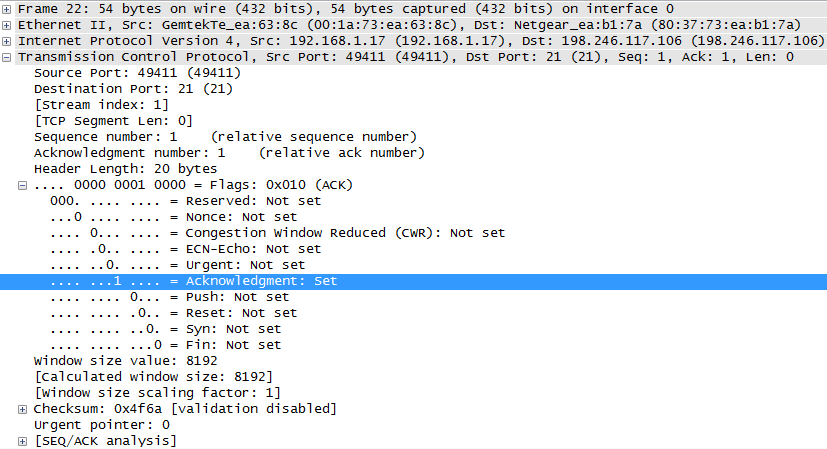
In the **second** Wireshark filtered capture, the CDC FTP server acknowledges the request from the PC. Note the values of the SYN and ACK bits.



Fill in the following information regarding the SYN-ACK message.

|  |  |
| --- | --- |
| Source IP address |  |
| Destination IP address |  |
| Source port number |  |
| Destination port number |  |
| Sequence number |  |
| Acknowledgement number |  |
| Header length |  |
| Window size |  |

In the final stage of the negotiation to establish communications, the PC sends an acknowledgement message to the server. Notice only the ACK bit is set to 1, and the Sequence number has been incremented to 1.



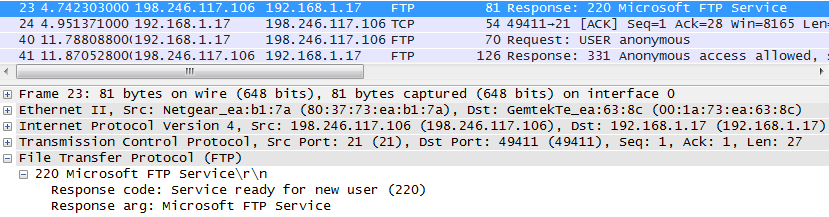
Fill in the following information regarding the ACK message.

|  |  |
| --- | --- |
| Source IP address |  |
| Destination IP address |  |
| Source port number |  |
| Destination port number |  |
| Sequence number |  |
| Acknowledgement number |  |
| Header length |  |
| Window size |  |

How many other TCP datagrams contained a SYN bit?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

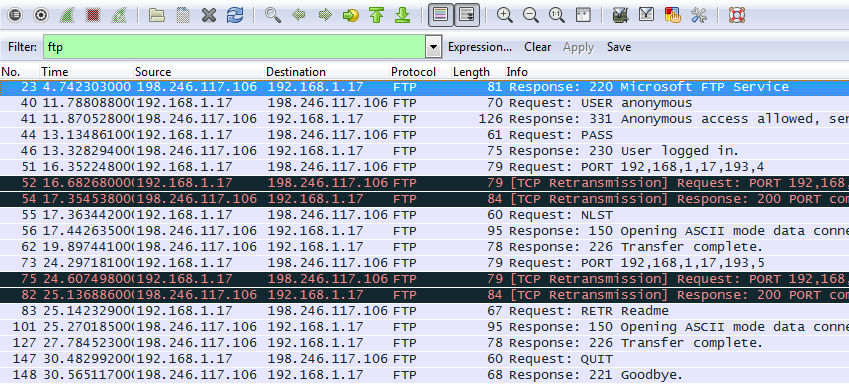
After a TCP session is established, FTP traffic can occur between the PC and FTP server. The FTP client and server communicate with each other, unaware that TCP has control and management over the session. When the FTP server sends a *Response: 220* to the FTP client, the TCP session on the FTP client sends an acknowledgment to the TCP session on the server. This sequence is visible in the Wireshark capture below.



When the FTP session has finished, the FTP client sends a command to **“quit”**. The FTP server acknowledges the FTP termination with a ***Response: 221 Goodbye***. At this time, the FTP server TCP session sends a TCP datagram to the FTP client, announcing the termination of the TCP session. The FTP client TCP session acknowledges receipt of the termination datagram, then sends its own TCP session termination. When the originator of the TCP termination (the FTP server) receives a duplicate termination, an ACK datagram is sent to acknowledge the termination and the TCP session is closed. This sequence is visible in the diagram and capture below.



By applying an **ftp** filter, the entire sequence of the FTP traffic can be examined in Wireshark. Notice the sequence of the events during this FTP session. The username **anonymous** was used to retrieve the Readme file. After the file transfer completed, the user ended the FTP session.



Apply the TCP filter again in Wireshark to examine the termination of the TCP session. **Four packets** are transmitted for the termination of the TCP session. Because TCP connection is full-duplex, each direction must terminate independently. Examine the source and destination addresses.

In this example, the FTP server has no more data to send in the stream. It sends a segment with the FIN flag set in frame 149. The PC sends an ACK to acknowledge the receipt of the FIN to terminate the session from the server to the client in frame 150.

In frame 151, the PC sends a FIN to the FTP server to terminate the TCP session. The FTP server responds with an ACK to acknowledge the FIN from the PC in frame 152. Now the TCP session terminated between the FTP server and PC.