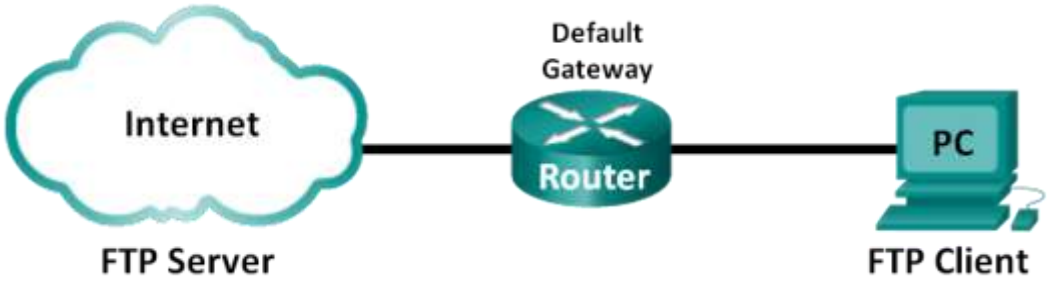
Lab - Using Wireshark to Examine FTP and TFTP Captures (Instructor Version)

Instructor Note: Red font color or Gray highlights indicate text that appears in the instructor copy only.

Topology – Part 1 (FTP)

Part 1 will highlight a TCP capture of an FTP session. This topology consists of a PC with Internet access.



Topology – Part 2 (TFTP)

Part 2 will highlight a UDP capture of a TFTP session. The PC must have both an Ethernet connection and a console connection to Switch S1.

Addressing Table (Part 2)

Device	Interface	IP Address	Subnet Mask	Default Gateway
S1	VLAN 1	192.168.1.1	255.255.255.0	N/A
PC-A	NIC	192.168.1.3	255.255.255.0	192.168.1.1

Objectives

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

Part 2: Identify UDP Header Fields and Operation Using a Wireshark TFTP Session Capture

Background / Scenario

The two protocols in the TCP/IP transport layer are the TCP, defined in RFC 761, and UDP, defined in RFC 768. 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.

Note: Understanding the parts of the TCP and UDP headers and operation are a critical skill for network engineers.

In Part 1 of this lab, you will use 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. In Part 2 of this lab, you will use Wireshark to capture and analyze UDP protocol header fields for TFTP file transfers between the host computer and Switch S1.

Instructor Note: If Wireshark version 1.8.3 or later has not been loaded on the PC, it may be downloaded from URL http://www.wireshark.org/download.html. For Part 2 of the lab, if tftpd32 version 4.0 or later has not been installed on the PC, it may be downloaded from URL http://tftpd32.jounin.net/tftpd32_download.html.

Note: The switch used is a Cisco Catalyst 2960s with Cisco IOS Release 15.0(2) (lanbasek9 image). Other switches and Cisco IOS versions can be used. Depending on the model and Cisco IOS version, the available commands and output produced might vary from what displays in the labs.

Note: Make sure that the switch has been erased and has no startup configurations. If you are unsure, contact your instructor.

Note: Part 1 assumes the PC has Internet access and cannot be performed using Netlab. Part 2 is Netlab compatible.

Instructor Note: Instructions for erasing the switch are provided in the Lab Manual.

Instructor Note: This lab may be performed in two sessions based on time and equipment availability. The sequence of Part 1 and Part 2 is not critical.

Instructor Note: Using a packet sniffer such as Wireshark may be considered a breach of the security policy of the school. It is recommended that permission is obtained before running Wireshark for this lab. If using a packet sniffer such as Wireshark is an issue, the instructor may wish to assign the lab as homework or perform a walk-through demonstration.

Required Resources – Part 1 (FTP)

1 PC (Windows 7, Vista, or XP with command prompt access, Internet access, and Wireshark installed)

Required Resources – Part 2 (TFTP)

- 1 Switch (Cisco 2960 with Cisco IOS Release 15.0(2) lanbasek9 image or comparable)
- 1 PC (Windows 7, Vista, or XP with Wireshark and a TFTP server, such as tftpd32 installed)
- Console cable to configure the Cisco IOS devices via the console port
- Ethernet cable as shown in the topology

Part 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.

Step 1: Start a Wireshark capture.

a. Close all unnecessary network traffic, such as the web browser, to limit the amount traffic during the Wireshark capture.

b. Start the Wireshark capture.

Step 2: Download the Readme file.

a. From the command prompt, enter **ftp ftp.cdc.gov**.

b. Log into the FTP site for Centers for Disease Control and Prevention (CDC) with user **anonymous** and no password.

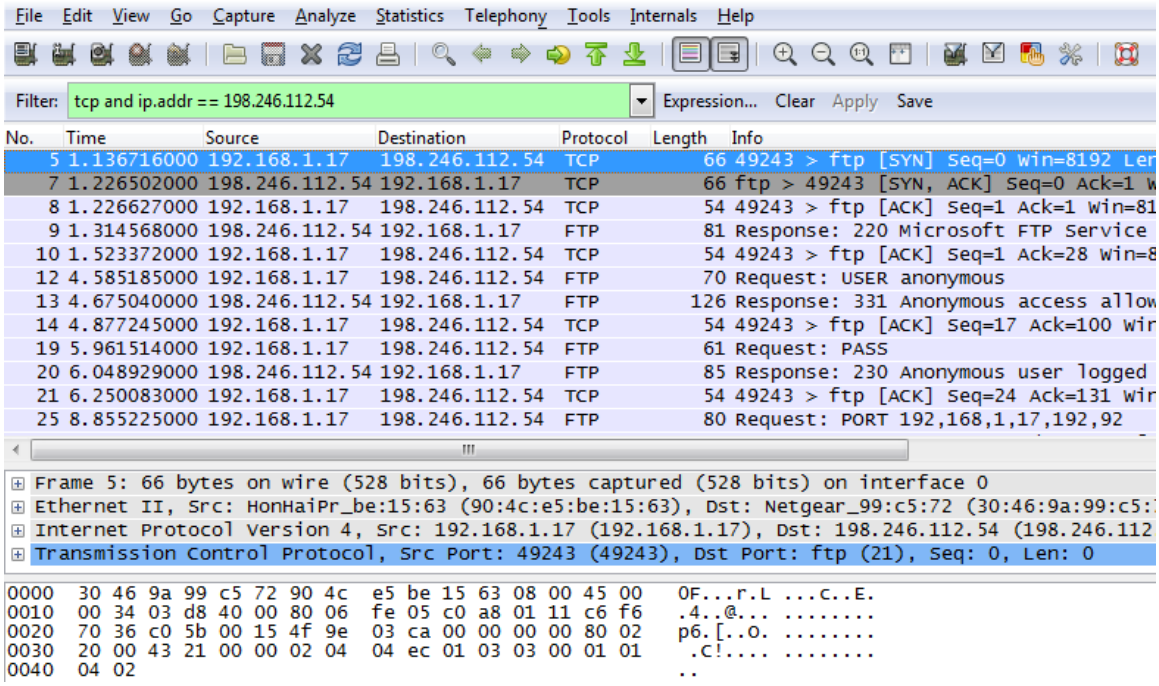
c. Locate and download the Readme file.

```
C:\Users\user1>ftp ftp.cdc.gov
Connected to ftp.cdc.gov.
220 Microsoft FTP Service
User (ftp.cdc.gov:(none)): anonymous
331 Anonymous access allowed, send identity (e-mail name) as password.
Password:
230 Anonymous user logged in.
ftp> ls
200 PORT command successful.
150 Opening ASCII mode data connection for file list.
aspnet_client
pub
Readme
Siteinfo
up.htm
w3c
web.config
welcome.msg
226 Transfer complete.
ftp: 76 bytes received in 0.00Seconds 19.00Kbytes/sec.
ftp> get Readme
200 PORT command successful.
150 Opening ASCII mode data connection for Readme(1428 bytes).
226 Transfer complete.
ftp: 1428 bytes received in 0.01Seconds 204.00Kbytes/sec.
ftp> quit
221
```

Step 3: Stop the Wireshark capture.

Step 4: 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 == 198.246.112.54** in the **Filter: entry** area and click **Apply**. The IP address, 198.246.112.54, is the address for ftp.cdc.gov.



Step 5: Analyze the TCP fields.

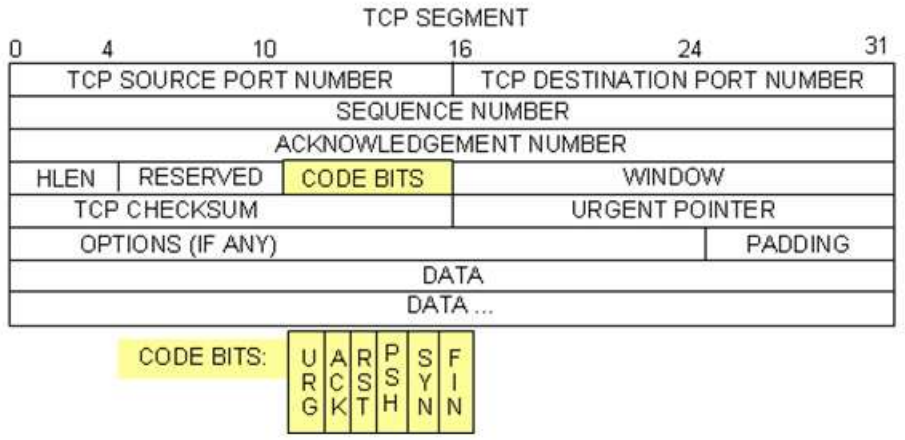
After the TCP filter has been applied, the first three frames in the packet list pane (top section) displays the transport layer protocol TCP creating a reliable session. The sequence of [SYN], [SYN, ACK], and [ACK] illustrates the three-way handshake.

```
    5 1.136716000  192.168.1.17    198.246.112.54   TCP          66 49243 > ftp [SYN] Seq=0 Win=8192 Le
    7 1.226502000  198.246.112.54  192.168.1.17     TCP          66 ftp > 49243 [SYN, ACK] Seq=0 Ack=1
    8 1.226627000  192.168.1.17    198.246.112.54   TCP          54 49243 > ftp [ACK] Seq=1 Ack=1 Win=8
```

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. Finally, 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 record. The expanded TCP datagram appears similar to the packet detail pane shown below.

```
⊞ Frame 5: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
⊞ Ethernet II, Src: HonHaiPr_be:15:63 (90:4c:e5:be:15:63), Dst: Netgear_99:c5:72 (30:46:9a:99:c5:72)
⊞ Internet Protocol Version 4, Src: 192.168.1.17 (192.168.1.17), Dst: 198.246.112.54 (198.246.112.54)
⊟ Transmission Control Protocol, Src Port: 49243 (49243), Dst Port: ftp (21), Seq: 0, Len: 0
    Source port: 49243 (49243)
    Destination port: ftp (21)
    [Stream index: 0]
    Sequence number: 0    (relative sequence number)
    Header length: 32 bytes
  ⊟ Flags: 0x002 (SYN)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
      .... 0... .... = Congestion Window Reduced (CWR): Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...0 .... = Acknowledgment: Not set
      .... .... 0... = Push: Not set
      .... .... .0.. = Reset: Not set
    ⊞ .... .... ..1. = Syn: Set
      .... .... ...0 = Fin: Not set
    Window size value: 8192
    [Calculated window size: 8192]
  ⊞ Checksum: 0x4321 [validation disabled]
  ⊞ Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operation (NOP), Nc
```



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, HTTP, and so on). The combination of the source IP address, source port, destination IP address, and destination port uniquely identifies the session to both 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:
  - ACK — Acknowledgement of a segment receipt.
  - SYN — Synchronize, only set when a new TCP session is negotiated during the TCP three-way handshake.
  - FIN — Finish, request to close the TCP session.
- The **Window size** is the value of the sliding window; 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).

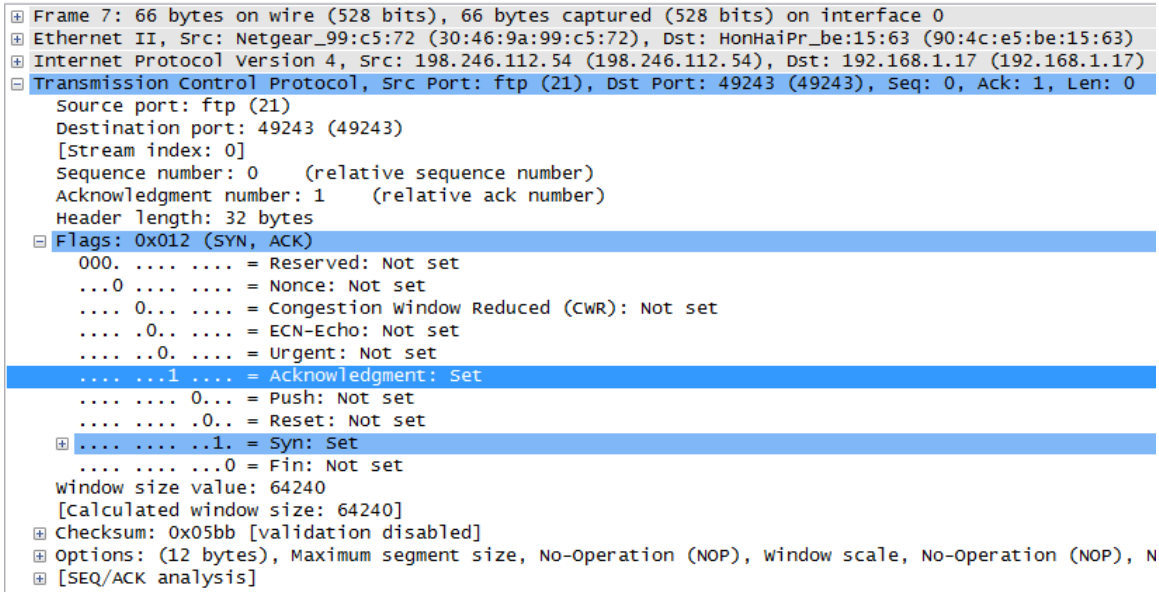
Using the Wireshark capture of the first TCP session startup (SYN bit set to 1), fill in information about the TCP header:

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

Source IP Address:	192.168.1.17*
Destination IP Address:	198.246.112.54
Source port number:	49243*
Destination port number:	21
Sequence number:	0 (relative)
Acknowledgement number:	Not Applicable for this capture
Header length:	32 bytes
Window size:	8192

*Student answers will vary.

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.

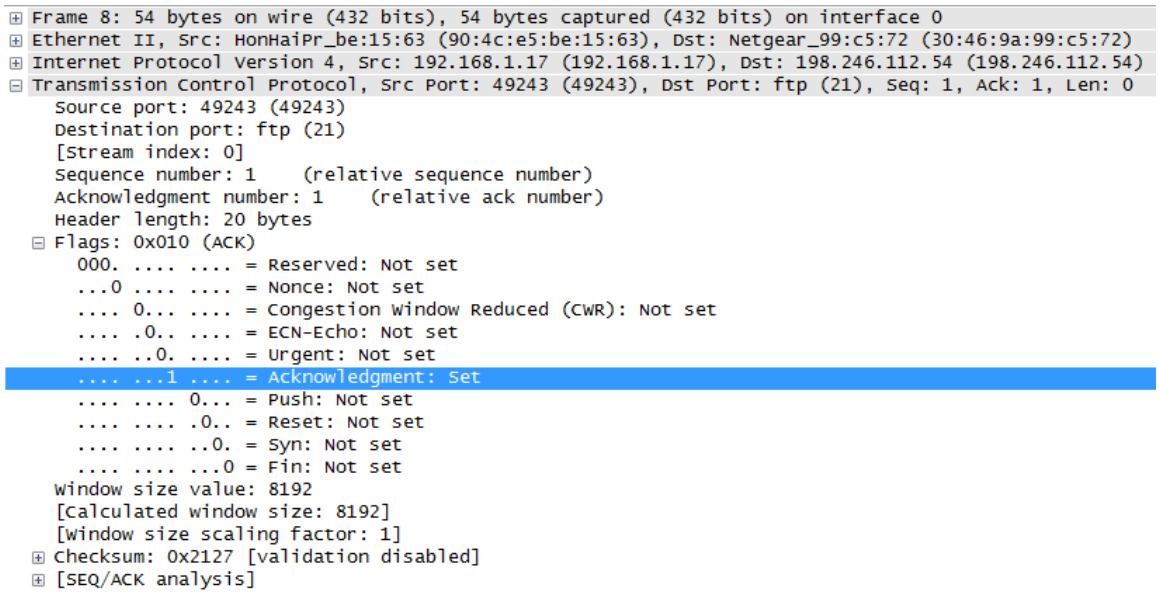


Fill in the following information regarding the SYN-ACK message.

Source IP address:	198.246.112.54
Destination IP address:	192.168.1.17*
Source port number:	21
Destination port number:	49243*
Sequence number:	0 (relative)
Acknowledgement number:	1
Header length:	32 bytes
Window size:	64240

*Student answers will vary.

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:	192.168.1.17*
Destination IP address:	198.246.112.54
Source port number:	49243*
Destination port number:	21
Sequence number:	1
Acknowledgement number:	1
Header length:	20
Window size:	8192*

*Student answers will vary.

How many other TCP datagrams contained a SYN bit?

_____

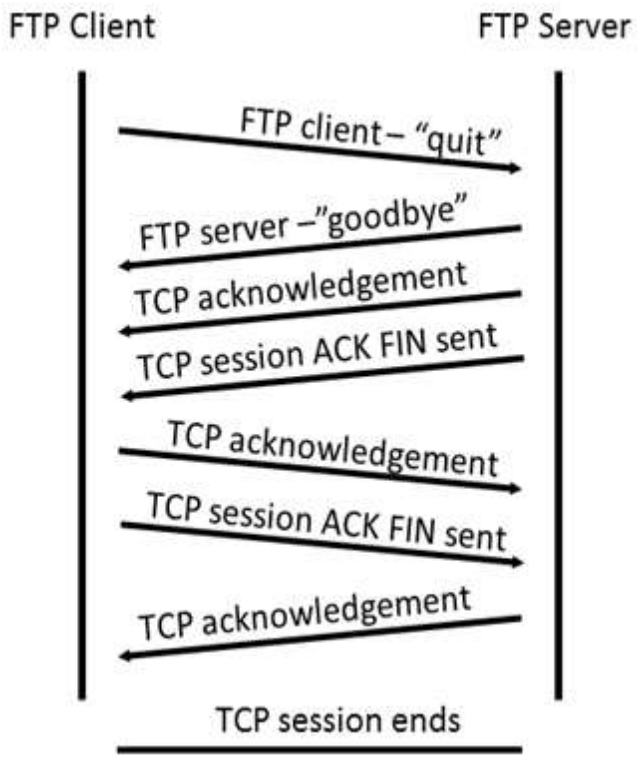
One. The first packet send by the host at the beginning of a TCP session.

After a TCP session is established, FTP traffic can occur between the PC and FTP server. The FTP client and server communicate between 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.

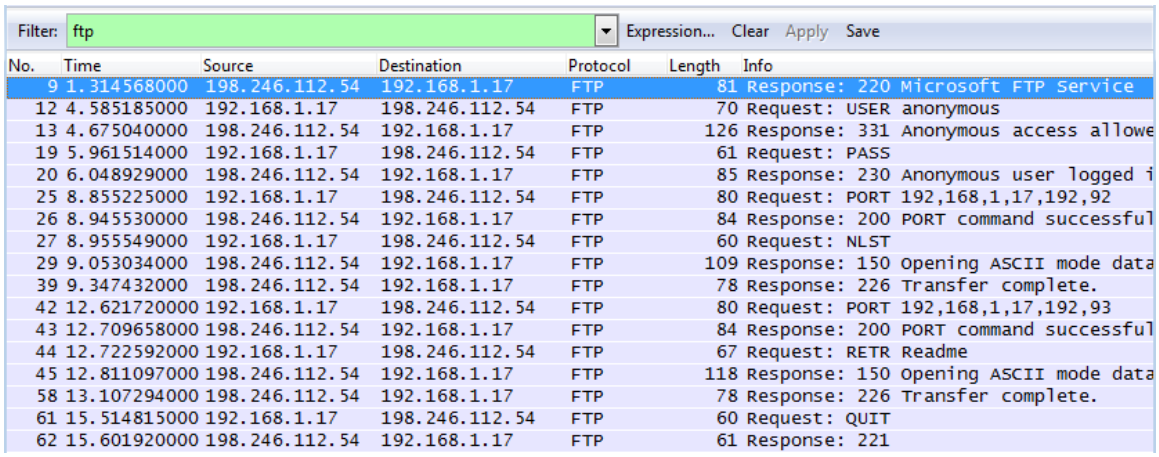
```
 9 1.314568000  198.246.112.54  192.168.1.17     FTP          81 Response: 220 Microsoft FTP Service
10 1.523372000  192.168.1.17    198.246.112.54   TCP          54 49243 > ftp [ACK] Seq=1 Ack=28 Win=
12 4.585185000  192.168.1.17    198.246.112.54   FTP          70 Request: USER anonymous
13 4.675040000  198.246.112.54  192.168.1.17     FTP         126 Response: 331 Anonymous access allo
```

```
⊞ Frame 9: 81 bytes on wire (648 bits), 81 bytes captured (648 bits) on interface 0
⊞ Ethernet II, Src: Netgear_99:c5:72 (30:46:9a:99:c5:72), Dst: HonHaiPr_be:15:63 (90:4c:e5:be:15:63)
⊞ Internet Protocol Version 4, Src: 198.246.112.54 (198.246.112.54), Dst: 192.168.1.17 (192.168.1.17)
⊞ Transmission Control Protocol, Src Port: ftp (21), Dst Port: 49243 (49243), Seq: 1, Ack: 1, Len: 27
⊟ File Transfer Protocol (FTP)
  ⊟ 220 Microsoft FTP Service\r\n
      Response code: Service ready for new user (220)
      Response arg: Microsoft FTP Service
```

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, 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.



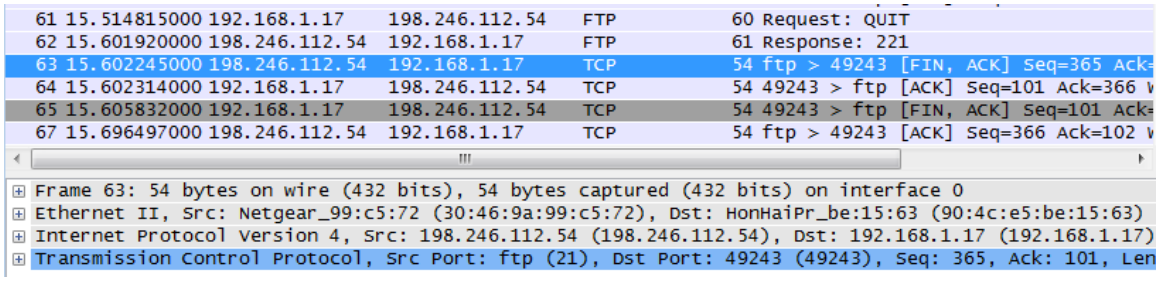
By applying an **ftp** filter, the entire sequence of the FTP traffic can be examined in Wireshark. Notice the sequence of the events during this FTP session. The username anonymous was used to retrieve the Readme file. After the file transfer completed, the user ended the FTP session.



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 63. The PC sends an ACK to acknowledge the receipt of the FIN to terminate the session from the server to the client in frame 64.

In frame 65, 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 67. Now the TCP session terminated between the FTP server and PC.



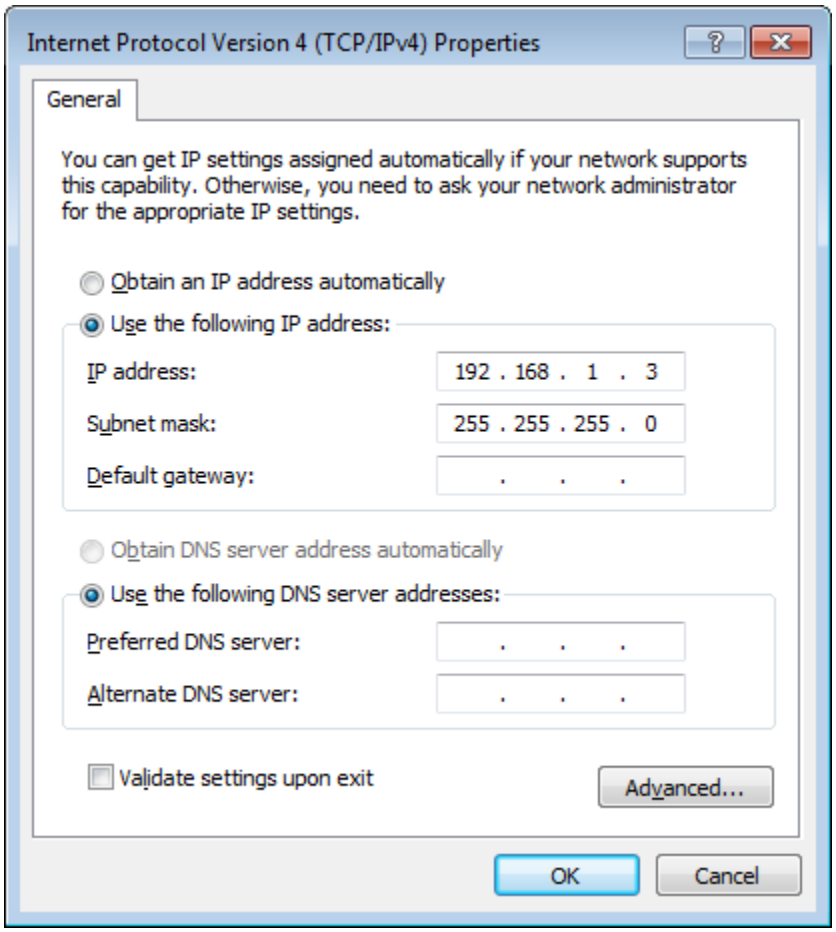
Part 2: Identify UDP Header Fields and Operation Using a Wireshark TFTP Session Capture

In Part 2, you use Wireshark to capture a TFTP session and inspect UDP header fields.

Step 1: Set up this physical topology and prepare for TFTP capture.

a. Establish a console and Ethernet connection between PC-A and Switch S1.

b. If not already done, manually configure the IP address on the PC to 192.168.1.3. It is not required to set the default gateway.



c. Configure the switch. Assign an IP address of 192.168.1.1 to VLAN 1. Verify connectivity with the PC by pinging 192.168.1.3. Troubleshoot as necessary.

```
Switch> enable
Switch# conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Switch(config)# host S1
S1(config)# interface vlan 1
S1(config-if)# ip address 192.168.1.1 255.255.255.0
S1(config-if)# no shut
*Mar  1 00:37:50.166: %LINK-3-UPDOWN: Interface Vlan1, changed state to up
*Mar  1 00:37:50.175: %LINEPROTO-5-UPDOWN: Line protocol on Interface Vlan1,
changed state to up
S1(config-if)# end
S1# ping 192.168.1.3
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.1.3, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/203/1007 ms
```

d. Save the running configuration to NVRAM:

```
S1# copy run start
```

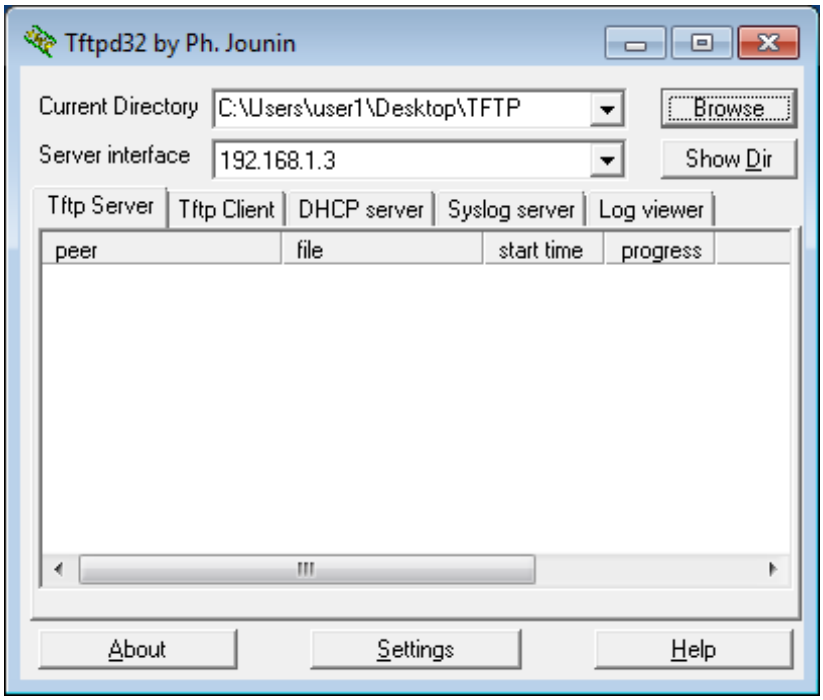
Step 2: Prepare the TFTP server on the PC.

a. If it does not already exist, create a folder on the PC desktop called **TFTP**. The files from the switch will be copied to this location.

b. Start **tftpd32** on the PC.

c. Click **Browse** and change the current directory to **C:\Users\user1\Desktop\TFTP** by replacing user1 with your username.

The TFTP server should look like this:



Notice that in Current Directory, it lists the user and the Server (PC-A) interface as the IP address of **192.168.1.3**.

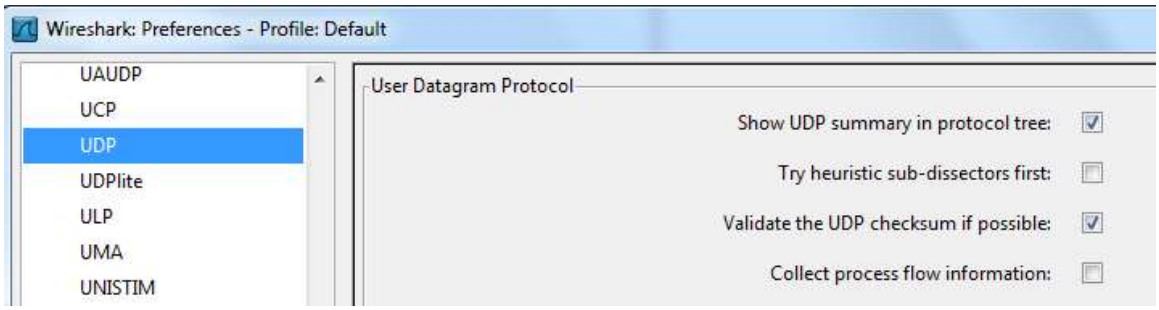
d. Test the ability to copy a file using TFTP from the switch to the PC. Troubleshoot as necessary.

```
S1# copy start tftp
Address or name of remote host []? 192.168.1.3
Destination filename [s1-confg]?
!!
1638 bytes copied in 0.026 secs (63000 bytes/sec)
```

If you see that the file has copied (as in the above output), then you are ready to go on to the next step. If not, then troubleshoot. If you get the `%Error opening tftp (Permission denied)` error, first check to make sure your firewall is not blocking TFTP, and that you are copying to a location where your username has adequate permission, such as the desktop.

Step 3: Capture a TFTP session in Wireshark

a. Open Wireshark. From the **Edit** menu, choose **Preferences** and click the (+) sign to expand **Protocols**. Scroll down and select **UDP**. Click the **Validate the UDP checksum if possible** check box and click **Apply**. Then click **OK**.

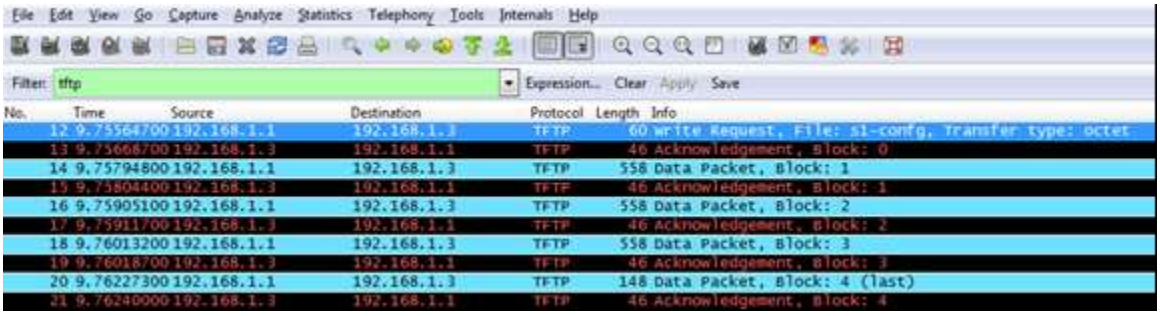


Instructor Note: This is a change from previous versions of this lab because the technology has changed. Search for "checksum offloading in Wireshark".

b. Start a Wireshark capture.

c. Run the **`copy start tftp`** command on the switch.

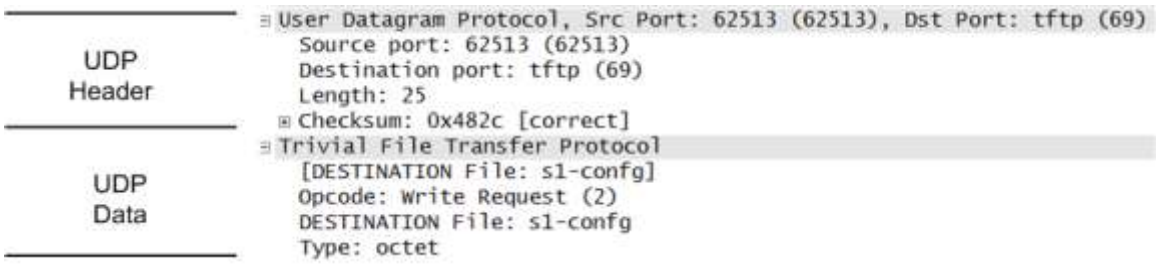
d. Stop the Wireshark capture.



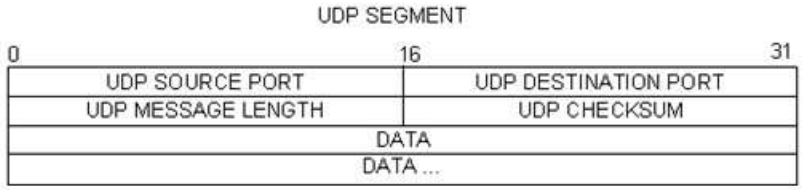
e. Set the filter to **tftp**. Your output should look similar to the output shown above. This TFTP transfer is used to analyze transport layer UDP operations.

Instructor Note: If students point out UDP acknowledgements, explain that the UDP header does not contain an acknowledgement field. Instead, it is the responsibility of the upper-layer protocol, in this case TFTP, to manage data transfer and receipt information. This will be shown during the UDP datagram examination.

In Wireshark, detailed UDP information is available in the Wireshark packet details pane. Highlight the first UDP datagram from the host computer, and move the mouse pointer to the packet details pane. It may be necessary to adjust the packet details pane and expand the UDP record by clicking the protocol expand box. The expanded UDP datagram should look similar to the diagram below.



The figure below is a UDP datagram diagram. Header information is sparse, compared to the TCP datagram. Similar to TCP, each UDP datagram is identified by the UDP source port and UDP destination port.



Using the Wireshark capture of the first UDP datagram, fill in information about the UDP header. The checksum value is a hexadecimal (base 16) value, denoted by the preceding 0x code:

Source IP Address:	192.168.1.1
Destination IP Address:	192.168.1.3
Source Port Number:	62513*
Destination Port Number:	69
UDP Message Length:	25 bytes*
UDP Checksum:	0x482c [correct]*

*Student answer will vary.

How does UDP verify datagram integrity?

_____

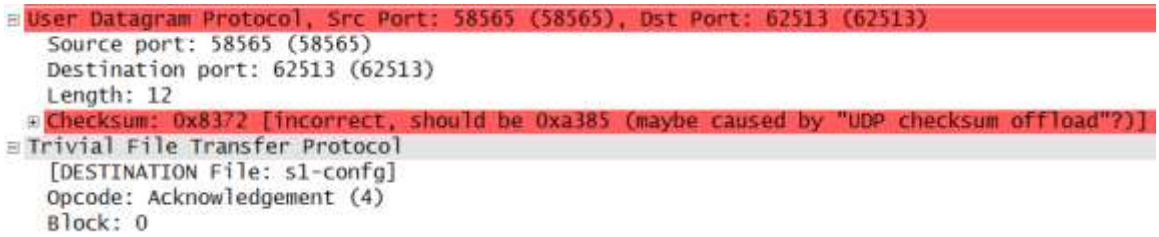
_____

A checksum is sent in the UDP datagram, and the datagram checksum value is recomputed upon receipt. If the computed checksum is identical to the sent checksum, then the UDP datagram is assumed to be complete.

Examine the first frame returned from tftpd server. Fill in the information about the UDP header:

Source IP Address:	192.168.1.3
Destination IP Address:	192.168.1.1
Source Port Number:	58565*
Destination Port Number:	62513*
UDP Message Length:	12 bytes*
UDP Checksum:	Checksum: 0x8372 [incorrect, should be 0xa385 (maybe caused by "UDP checksum offload"?)]*

*Student answer will vary.



Notice that the return UDP datagram has a different UDP source port, but this source port is used for the remainder of the TFTP transfer. Because there is no reliable connection, only the original source port used to begin the TFTP session is used to maintain the TFTP transfer.

Also notice that the UDP Checksum is incorrect. This is most likely caused by UDP checksum offload. You can learn more about why this happens by searching for “UDP checksum offload”.

Reflection

This lab provided the opportunity to analyze TCP and UDP protocol operations from captured FTP and TFTP sessions. How does TCP manage communication differently than UDP?

_____

_____

TCP manages communication much differently than UDP because reliability and guaranteed delivery requires additional control over the communication channel. UDP has less overhead and control, and the upper-layer protocol must provide some type of acknowledgement control. Both protocols, however, transport data between clients and servers using Application Layer protocols and are appropriate for the upper-layer protocol each supports.

Challenge

Because neither FTP nor TFTP are secure protocols, all transferred data is sent in clear text. This includes any user IDs, passwords, or clear-text file contents. Analyzing the upper-layer FTP session will quickly identify the user ID, password, and configuration file passwords. Upper-layer TFTP data examination is a bit more complicated, but the data field can be examined and the configuration user ID and password information extracted.

Cleanup

Unless directed otherwise by your instructor:

1) Remove the files that were copied to your PC.

2) Erase the configurations on switch **S1**.
3) Remove the manual IP address from the PC and restore Internet connectivity.

Device Configs

Switch S1

```
S1#show run
Building configuration...
!
hostname S1
!
!
interface Vlan1
 ip address 192.168.1.1 255.255.255.0
!
!
end
```