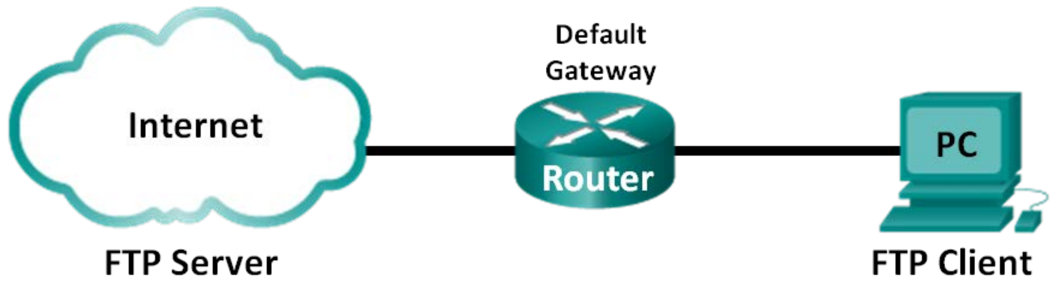
Lab - Using Wireshark to Examine TCP and UDP Captures

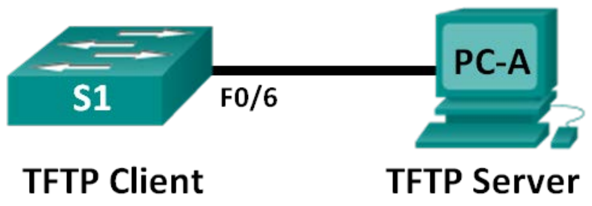
Topology – Part 1 (FTP)

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 by Using a Wireshark FTP Session Capture

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

Background / Scenario

Two protocols in the TCP/IP ttransport layer are 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 the open source tool Wireshark 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 to download a file. In Part 2 of this lab, you will use Wireshark to capture and analyze UDP header fields for TFTP file transfers between the host computer and S1.

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 the 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.

Required Resources – Part 1 (FTP)

1 PC (Windows 7, 8, or 10 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, 8, or 10 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.

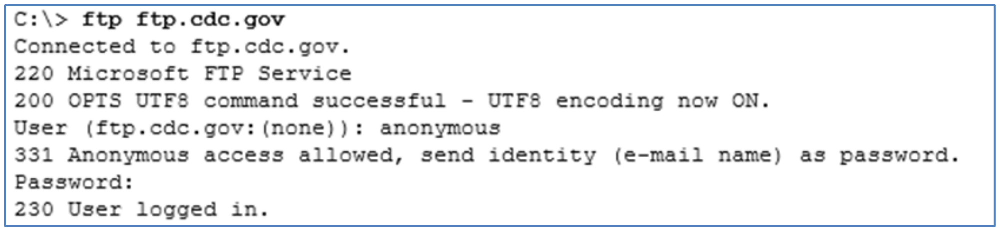
a. Close all unnecessary network traffic, such as the web browser, to limit the amount traffic during the Wireshark capture.

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:\> ftp ftp.cdc.gov
Connected to ftp.cdc.gov.
220 Microsoft FTP Service
200 OPTS UTF8 command successful - UTF8 encoding now ON.
User (ftp.cdc.gov:(none)): anonymous
331 Anonymous access allowed, send identity (e-mail name) as password.
Password:
230 User logged in.
```

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

```
ftp> ls
200 PORT command successful.
150 Opening ASCII mode data connection.
.change.dir
.message
pub
Readme
Siteinfo
up.htm
w3c
welcome.msg
226 Transfer complete.
ftp: 75 bytes received in 0.00Seconds 75000.00Kbytes/sec.
```

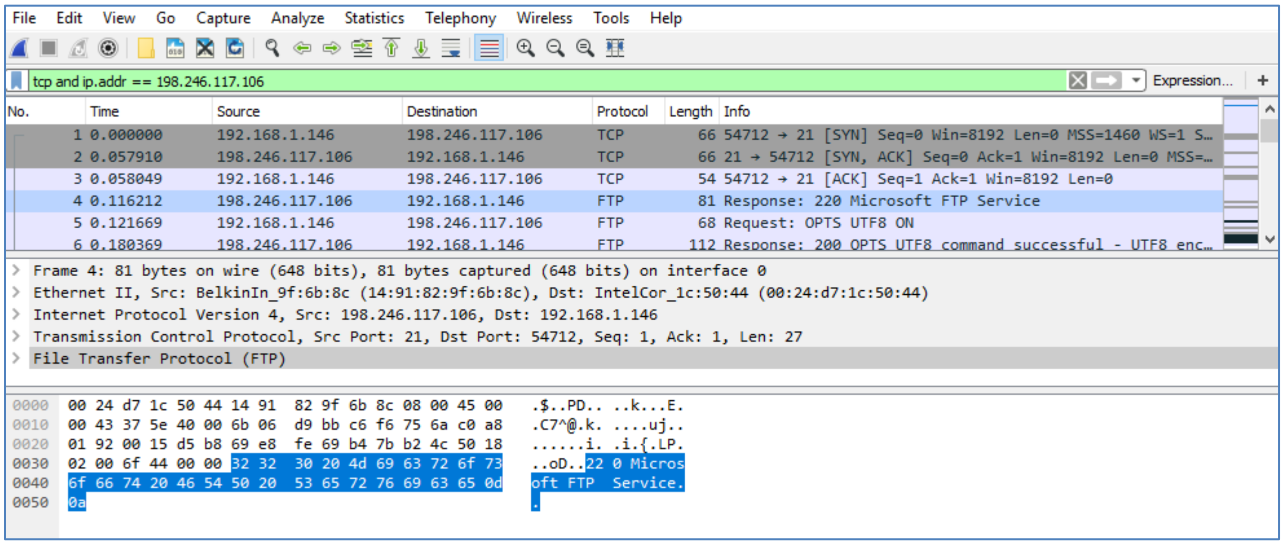
d. Enter the command **get Readme** to download the file. When the download is complete, enter the command **quit** to exit.

```
ftp> get Readme
200 PORT command successful.
150 Opening ASCII mode data connection.
226 Transfer complete.
ftp: 1428 bytes received in 0.08Seconds 18.08Kbytes/sec.
```

Step 3: Stop the Wireshark capture.

Step 4: View the Wireshark main window.

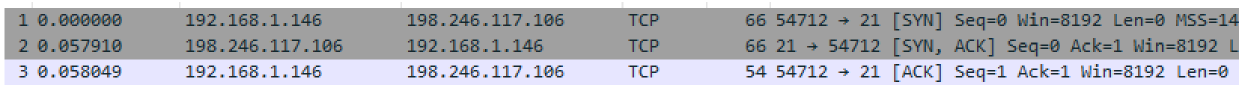
Wireshark captured many packets during the FTP session to ftp.cdc.gov. To limit the amount of data for analysis, type **tcp and ip.addr == 198.246.117.106** in the **Filter: entry** area and press **Enter**. The IP address, 198.246.117.106, is the address for ftp.cdc.gov at this time.


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Expression...

```
tcp and ip.addr == 198.246.117.106

No.  Time      Source           Destination      Protocol Length Info
   1 0.000000  192.168.1.146    198.246.117.106  TCP      66 54712 → 21 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=1 S…
   2 0.057910  198.246.117.106  192.168.1.146    TCP      66 21 → 54712 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=…
   3 0.058049  192.168.1.146    198.246.117.106  TCP      54 54712 → 21 [ACK] Seq=1 Ack=1 Win=8192 Len=0
   4 0.116212  198.246.117.106  192.168.1.146    FTP      81 Response: 220 Microsoft FTP Service
   5 0.121669  192.168.1.146    198.246.117.106  FTP      68 Request: OPTS UTF8 ON
   6 0.180369  198.246.117.106  192.168.1.146    FTP     112 Response: 200 OPTS UTF8 command successful - UTF8 enc…

> Frame 4: 81 bytes on wire (648 bits), 81 bytes captured (648 bits) on interface 0
> Ethernet II, Src: BelkinIn_9f:6b:8c (14:91:82:9f:6b:8c), Dst: IntelCor_1c:50:44 (00:24:d7:1c:50:44)
> Internet Protocol Version 4, Src: 198.246.117.106, Dst: 192.168.1.146
> Transmission Control Protocol, Src Port: 21, Dst Port: 54712, Seq: 1, Ack: 1, Len: 27
> File Transfer Protocol (FTP)

0000  00 24 d7 1c 50 44 14 91  82 9f 6b 8c 08 00 45 00   .$..PD.. ..k...E.
0010  00 43 37 5e 40 00 6b 06  d9 bb c6 f6 75 6a c0 a8   .C7^@.k. ....uj..
0020  01 92 00 15 d5 b8 69 e8  fe 69 b4 7b b2 4c 50 18   ......i. .i.{.LP.
0030  02 00 6f 44 00 00 32 32  30 20 4d 69 63 72 6f 73   ..oD..22 0 Micros
0040  6f 66 74 20 46 54 50 20  53 65 72 76 69 63 65 0d   oft FTP  Service.
0050  0a                                                 .
```

Step 5: Analyze the TCP fields.

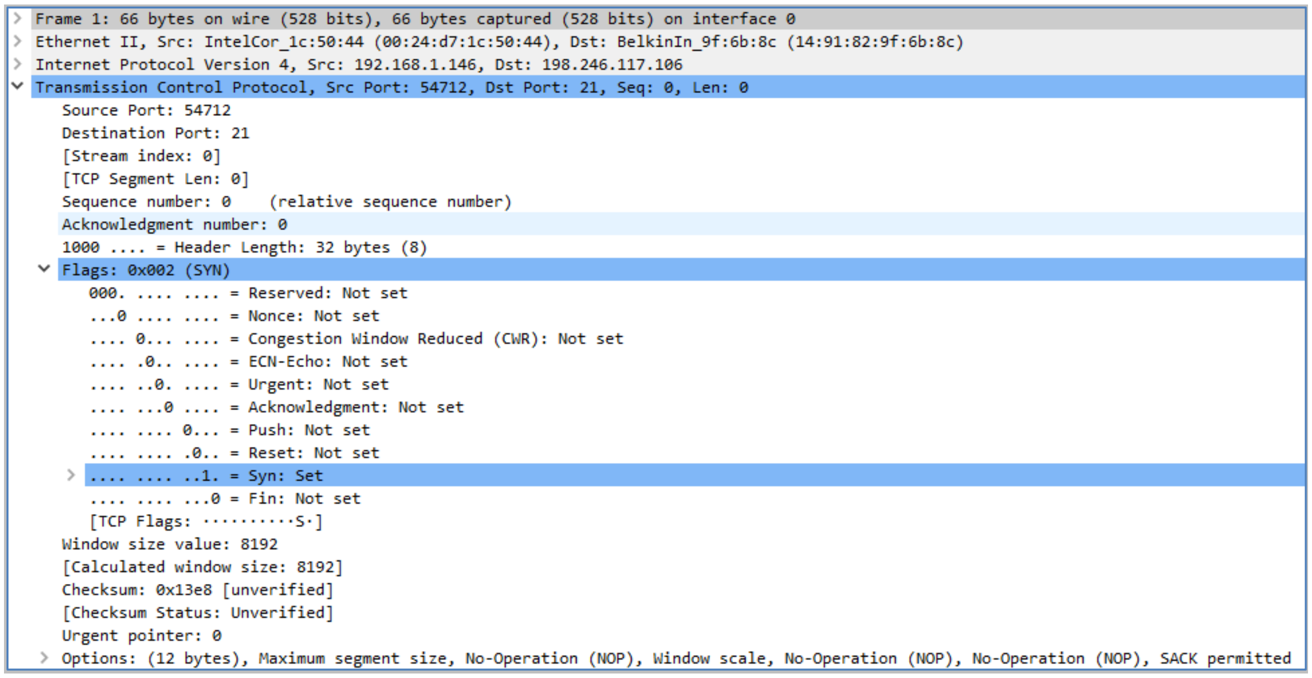
After the TCP filter has been applied, the first three frames in the packet list pane (top section) display the transport layer protocol TCP creating a reliable session. The sequence of [SYN], [SYN, ACK], and [ACK] illustrates the three-way handshake.



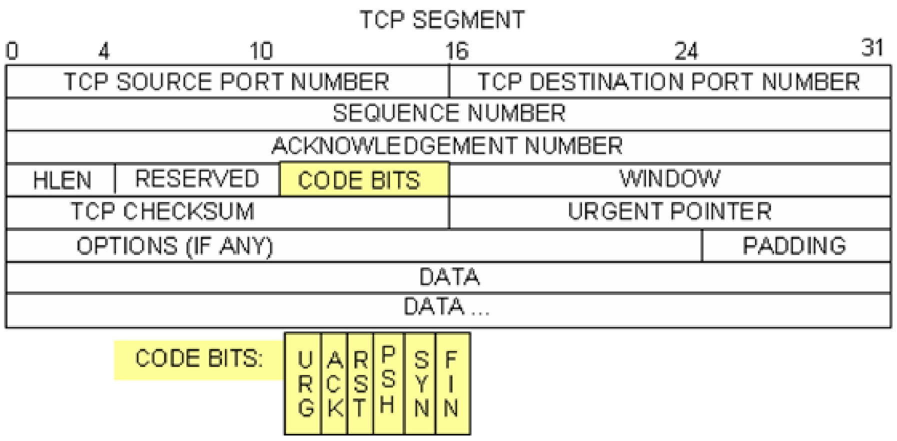
```
1 0.000000    192.168.1.146      198.246.117.106    TCP      66 54712 → 21 [SYN] Seq=0 Win=8192 Len=0 MSS=14
2 0.057910    198.246.117.106    192.168.1.146      TCP      66 21 → 54712 [SYN, ACK] Seq=0 Ack=1 Win=8192 L
3 0.058049    192.168.1.146      198.246.117.106    TCP      54 54712 → 21 [ACK] Seq=1 Ack=1 Win=8192 Len=0
```

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.



```
> Frame 1: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
> Ethernet II, Src: IntelCor_1c:50:44 (00:24:d7:1c:50:44), Dst: BelkinIn_9f:6b:8c (14:91:82:9f:6b:8c)
> Internet Protocol Version 4, Src: 192.168.1.146, Dst: 198.246.117.106
v Transmission Control Protocol, Src Port: 54712, Dst Port: 21, Seq: 0, Len: 0
    Source Port: 54712
    Destination Port: 21
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 0    (relative sequence number)
    Acknowledgment number: 0
    1000 .... = Header Length: 32 bytes (8)
  v 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
      [TCP Flags: ··········S·]
    Window size value: 8192
    [Calculated window size: 8192]
    Checksum: 0x13e8 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  > Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operation (NOP), No-Operation (NOP), SACK permitted
```



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.

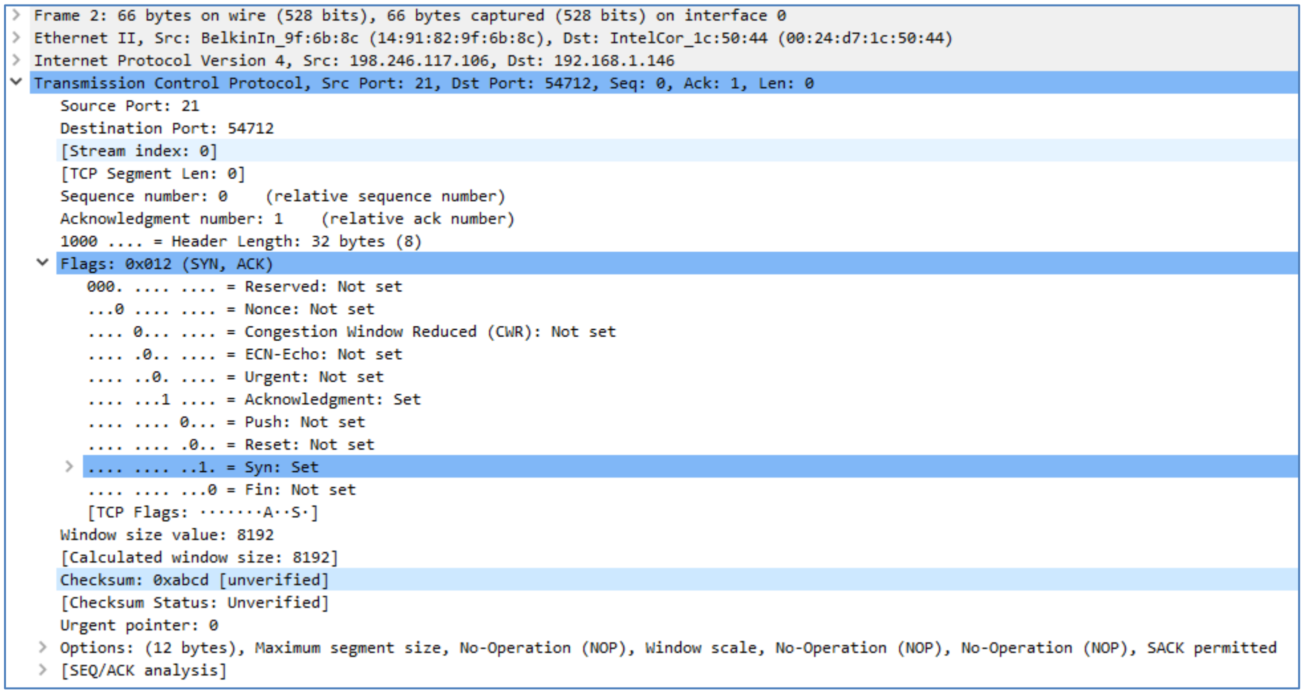
- The **Sequence Number** specifies the number of the last octet in a segment.
- The **Acknowledgment Number** specifies the next octet expected by the receiver.
- The **Code bits** have a special meaning in session management and in the treatment of segments. Among interesting values are:
  - ACK — Acknowledgment of a segment receipt.
  - SYN — Synchronize, only set when a new TCP session is negotiated during the TCP three-way handshake.
  - 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 acknowledgment.
- 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.

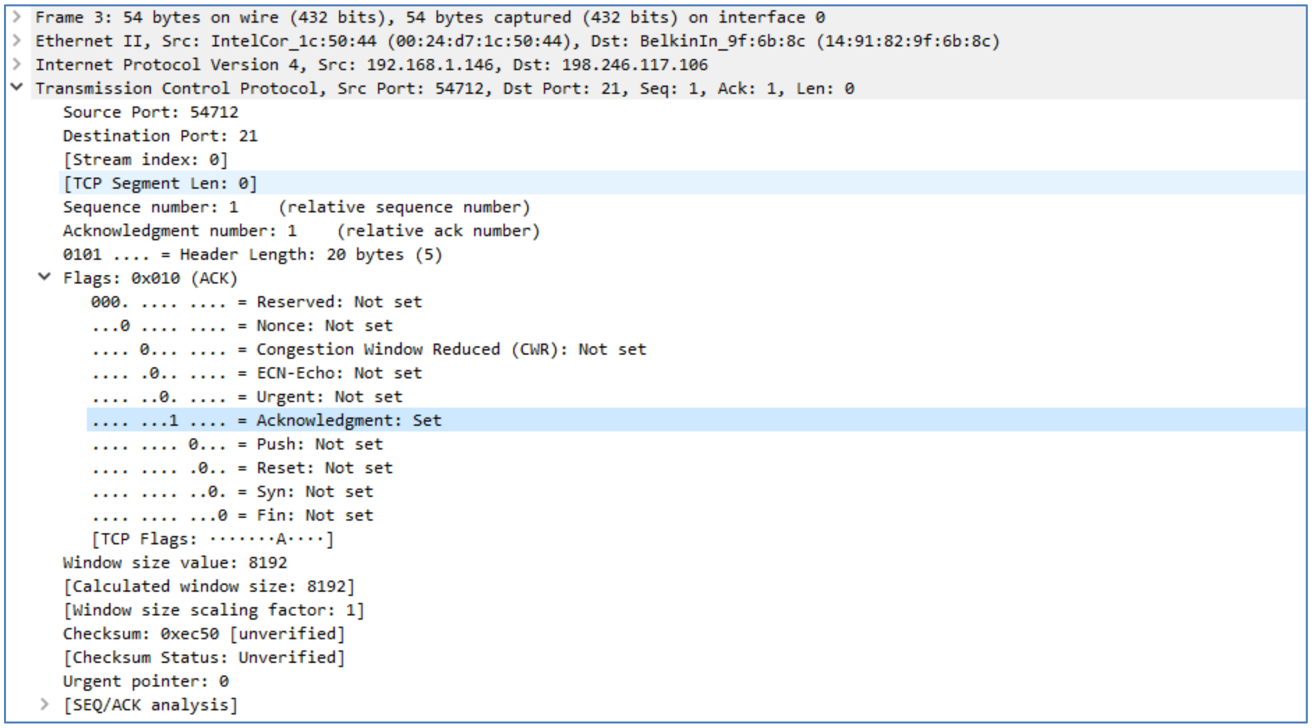


```
> Frame 2: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
> Ethernet II, Src: BelkinIn_9f:6b:8c (14:91:82:9f:6b:8c), Dst: IntelCor_1c:50:44 (00:24:d7:1c:50:44)
> Internet Protocol Version 4, Src: 198.246.117.106, Dst: 192.168.1.146
v Transmission Control Protocol, Src Port: 21, Dst Port: 54712, Seq: 0, Ack: 1, Len: 0
    Source Port: 21
    Destination Port: 54712
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 0    (relative sequence number)
    Acknowledgment number: 1    (relative ack number)
    1000 .... = Header Length: 32 bytes (8)
  v Flags: 0x012 (SYN, ACK)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
      .... 0... .... = Congestion Window Reduced (CWR): Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...1 .... = Acknowledgment: Set
      .... .... 0... = Push: Not set
      .... .... .0.. = Reset: Not set
    > .... .... ..1. = Syn: Set
      .... .... ...0 = Fin: Not set
      [TCP Flags: ·······A··S·]
    Window size value: 8192
    [Calculated window size: 8192]
    Checksum: 0xabcd [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  > Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operation (NOP), No-Operation (NOP), SACK permitted
  > [SEQ/ACK analysis]
```

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

Source IP address	
Destination IP address	
Source port number	
Destination port number	
Sequence number	
Acknowledgment number	
Header length	
Window size	

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



```
> Frame 3: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0
> Ethernet II, Src: IntelCor_1c:50:44 (00:24:d7:1c:50:44), Dst: BelkinIn_9f:6b:8c (14:91:82:9f:6b:8c)
> Internet Protocol Version 4, Src: 192.168.1.146, Dst: 198.246.117.106
v Transmission Control Protocol, Src Port: 54712, Dst Port: 21, Seq: 1, Ack: 1, Len: 0
    Source Port: 54712
    Destination Port: 21
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 1    (relative sequence number)
    Acknowledgment number: 1    (relative ack number)
    0101 .... = Header Length: 20 bytes (5)
  v Flags: 0x010 (ACK)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
      .... 0... .... = Congestion Window Reduced (CWR): Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...1 .... = Acknowledgment: Set
      .... .... 0... = Push: Not set
      .... .... .0.. = Reset: Not set
      .... .... ..0. = Syn: Not set
      .... .... ...0 = Fin: Not set
      [TCP Flags: ·······A····]
    Window size value: 8192
    [Calculated window size: 8192]
    [Window size scaling factor: 1]
    Checksum: 0xec50 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  > [SEQ/ACK analysis]
```

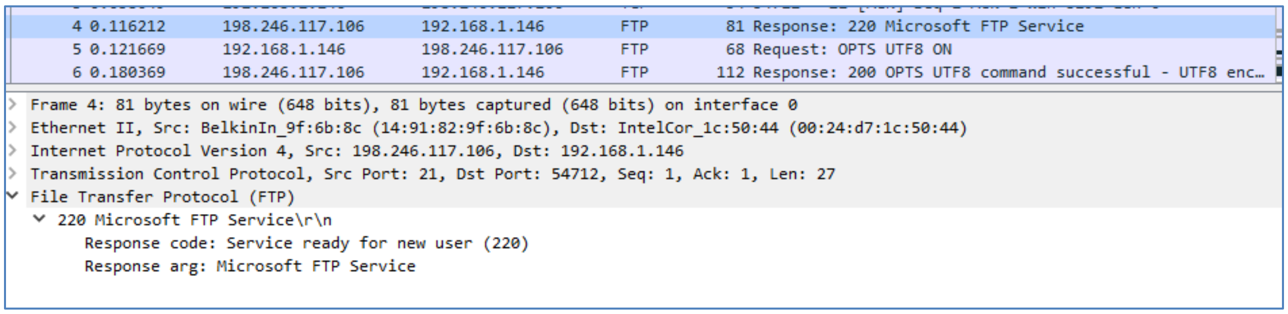
Fill in the following information regarding the ACK message.

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?

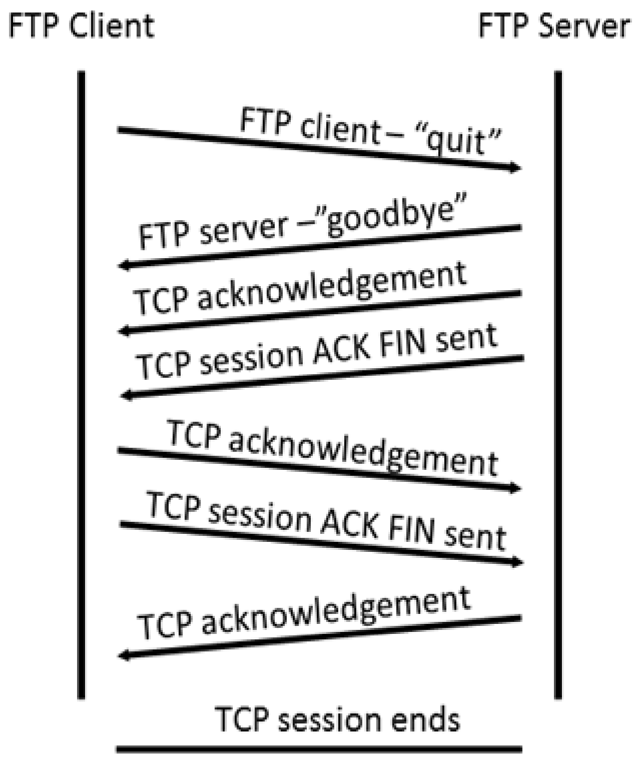
After a TCP session is established, FTP traffic can occur between the PC and FTP server. The FTP client and server communicate with each other, unaware that TCP has control and management over the session.

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.

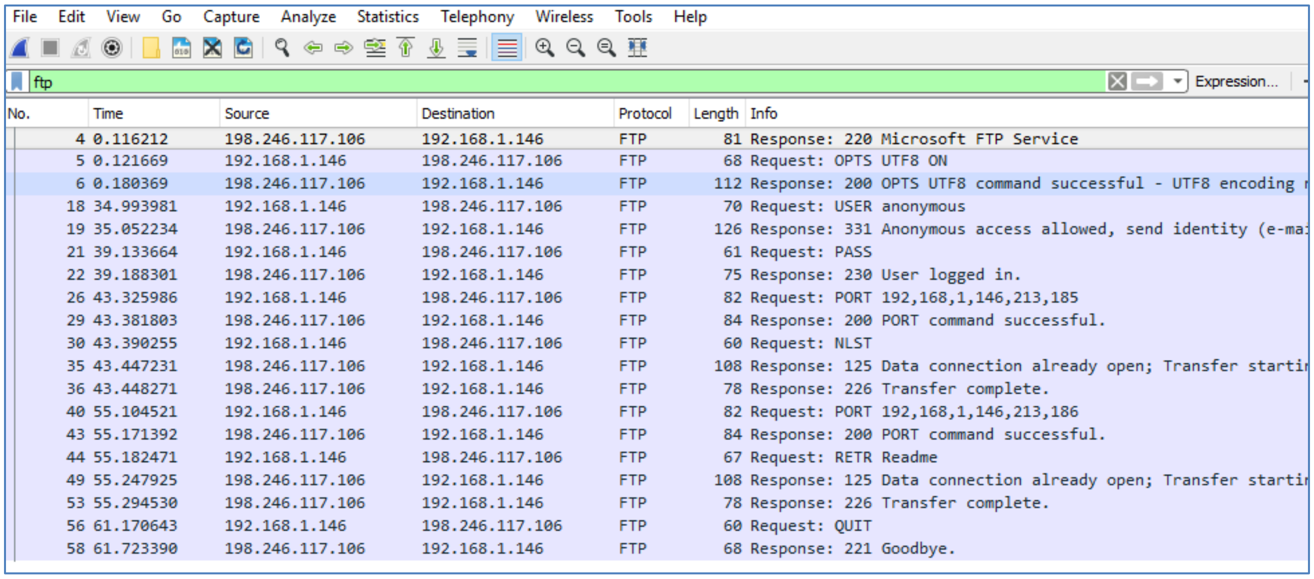


```
4 0.116212   198.246.117.106   192.168.1.146     FTP    81 Response: 220 Microsoft FTP Service
5 0.121669   192.168.1.146     198.246.117.106   FTP    68 Request: OPTS UTF8 ON
6 0.180369   198.246.117.106   192.168.1.146     FTP   112 Response: 200 OPTS UTF8 command successful - UTF8 enc…

> Frame 4: 81 bytes on wire (648 bits), 81 bytes captured (648 bits) on interface 0
> Ethernet II, Src: BelkinIn_9f:6b:8c (14:91:82:9f:6b:8c), Dst: IntelCor_1c:50:44 (00:24:d7:1c:50:44)
> Internet Protocol Version 4, Src: 198.246.117.106, Dst: 192.168.1.146
> Transmission Control Protocol, Src Port: 21, Dst Port: 54712, Seq: 1, Ack: 1, Len: 27
v File Transfer Protocol (FTP)
  v 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 (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.

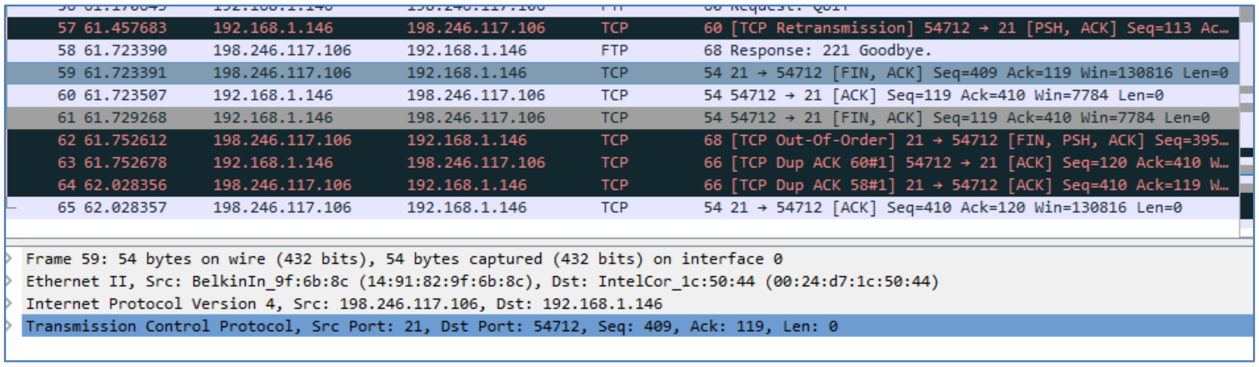

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Expression...

```
ftp

No.   Time        Source            Destination       Protocol  Length Info
    4 0.116212    198.246.117.106   192.168.1.146     FTP        81 Response: 220 Microsoft FTP Service
    5 0.121669    192.168.1.146     198.246.117.106   FTP        68 Request: OPTS UTF8 ON
    6 0.180369    198.246.117.106   192.168.1.146     FTP       112 Response: 200 OPTS UTF8 command successful - UTF8 encoding r
   18 34.993981   192.168.1.146     198.246.117.106   FTP        70 Request: USER anonymous
   19 35.052234   198.246.117.106   192.168.1.146     FTP       126 Response: 331 Anonymous access allowed, send identity (e-ma
   21 39.133664   192.168.1.146     198.246.117.106   FTP        61 Request: PASS
   22 39.188301   198.246.117.106   192.168.1.146     FTP        75 Response: 230 User logged in.
   26 43.325986   192.168.1.146     198.246.117.106   FTP        82 Request: PORT 192,168,1,146,213,185
   29 43.381803   198.246.117.106   192.168.1.146     FTP        84 Response: 200 PORT command successful.
   30 43.390255   192.168.1.146     198.246.117.106   FTP        60 Request: NLST
   35 43.447231   198.246.117.106   192.168.1.146     FTP       108 Response: 125 Data connection already open; Transfer starti
   36 43.448271   198.246.117.106   192.168.1.146     FTP        78 Response: 226 Transfer complete.
   40 55.104521   192.168.1.146     198.246.117.106   FTP        82 Request: PORT 192,168,1,146,213,186
   43 55.171392   198.246.117.106   192.168.1.146     FTP        84 Response: 200 PORT command successful.
   44 55.182471   192.168.1.146     198.246.117.106   FTP        67 Request: RETR Readme
   49 55.247925   198.246.117.106   192.168.1.146     FTP       108 Response: 125 Data connection already open; Transfer starti
   53 55.294530   198.246.117.106   192.168.1.146     FTP        78 Response: 226 Transfer complete.
   56 61.170643   192.168.1.146     198.246.117.106   FTP        60 Request: QUIT
   58 61.723390   198.246.117.106   192.168.1.146     FTP        68 Response: 221 Goodbye.
```

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 59 The PC sends an ACK to acknowledge the receipt of the FIN to terminate the session from the server to the client in frame 60.

In frame 61, 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 65. Now the TCP session terminated between the FTP server and PC.

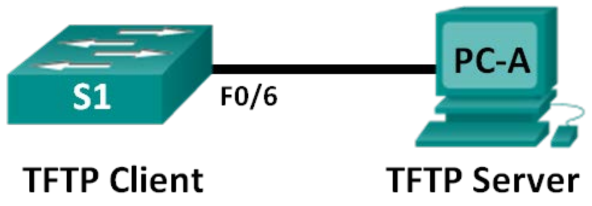


```
   57 61.457683   192.168.1.146     198.246.117.106   TCP     60 [TCP Retransmission] 54712 → 21 [PSH, ACK] Seq=113 Ac…
   58 61.723390   198.246.117.106   192.168.1.146     FTP     68 Response: 221 Goodbye.
   59 61.723391   198.246.117.106   192.168.1.146     TCP     54 21 → 54712 [FIN, ACK] Seq=409 Ack=119 Win=130816 Len=0
   60 61.723507   192.168.1.146     198.246.117.106   TCP     54 54712 → 21 [ACK] Seq=119 Ack=410 Win=7784 Len=0
   61 61.729268   192.168.1.146     198.246.117.106   TCP     54 54712 → 21 [FIN, ACK] Seq=119 Ack=410 Win=7784 Len=0
   62 61.752612   198.246.117.106   192.168.1.146     TCP     68 [TCP Out-Of-Order] 21 → 54712 [FIN, PSH, ACK] Seq=395…
   63 61.752678   192.168.1.146     198.246.117.106   TCP     66 [TCP Dup ACK 60#1] 54712 → 21 [ACK] Seq=120 Ack=410 W…
   64 62.028356   198.246.117.106   192.168.1.146     TCP     66 [TCP Dup ACK 58#1] 21 → 54712 [ACK] Seq=410 Ack=119 W…
   65 62.028357   198.246.117.106   192.168.1.146     TCP     54 21 → 54712 [ACK] Seq=410 Ack=120 Win=130816 Len=0

> Frame 59: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0
> Ethernet II, Src: BelkinIn_9f:6b:8c (14:91:82:9f:6b:8c), Dst: IntelCor_1c:50:44 (00:24:d7:1c:50:44)
> Internet Protocol Version 4, Src: 198.246.117.106, Dst: 192.168.1.146
> Transmission Control Protocol, Src Port: 21, Dst Port: 54712, Seq: 409, Ack: 119, Len: 0
```

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 the 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 S1.

b. 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
```

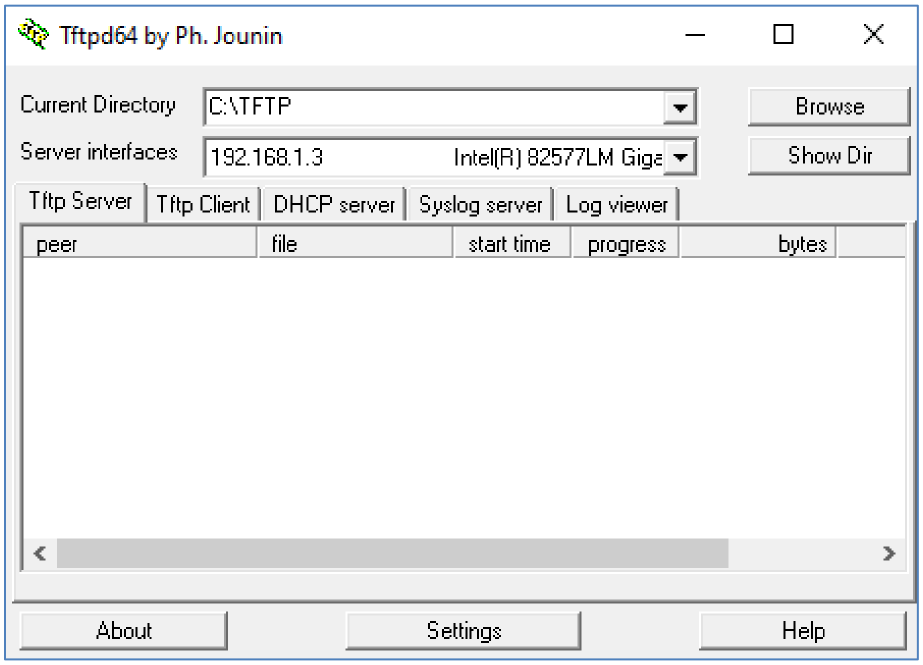
Step 2: Prepare the TFTP server on the PC.

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

b. Start **tftpd32** or **Tftpd64**on the PC.

c. Click **Browse** and change the current directory to **C:\TFTP**.

The TFTP server should look like this:



Current Directory: C:\TFTP

Server interfaces: 192.168.1.3 Intel(R) 82577LM Giga

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

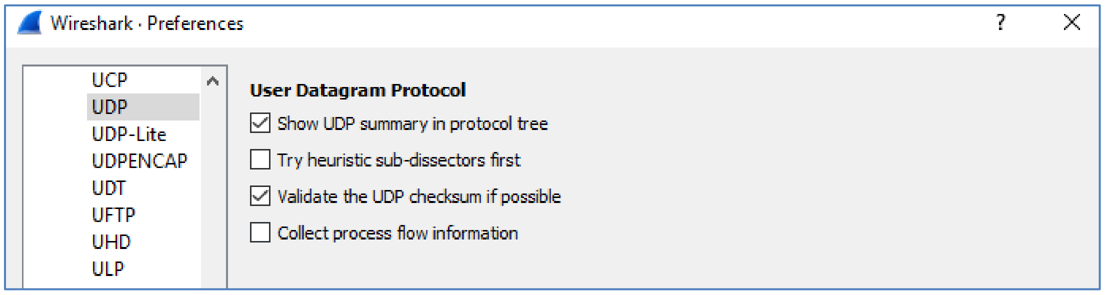
d. Test the ability to copy a file using TFTP from the switch to the PC. Troubleshoot as necessary.

```
S1# copy start tftp
Address or name of remote host []? 192.168.1.3
Destination filename [s1-confg]?
!!
1083 bytes copied in 0.84 secs
```

If you see that the file has been copied, then you are ready to go on to the next step. If the file has not been copied, troubleshoot as needed. If you get the `%Error opening tftp (Permission denied)` error, determine whether your firewall is blocking TFTP and whether you are copying the file 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 **OK**.



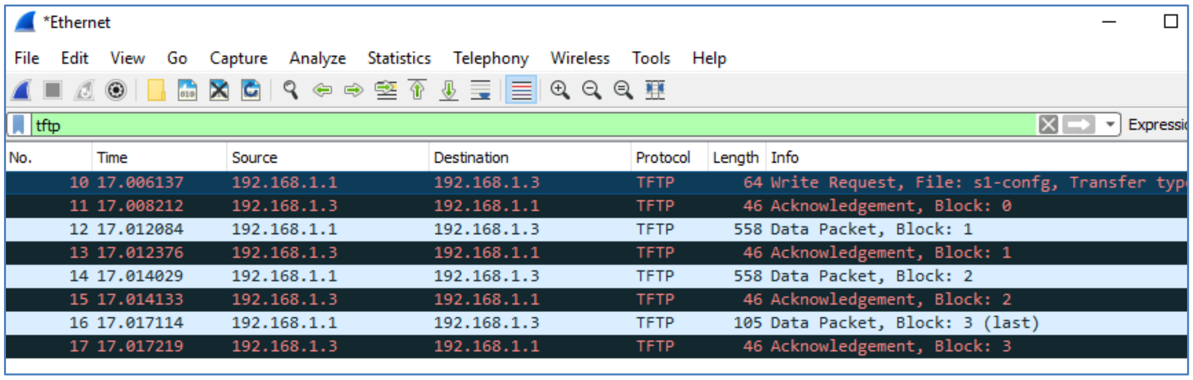
User Datagram Protocol

- [✓] Show UDP summary in protocol tree
- [ ] Try heuristic sub-dissectors first
- [✓] Validate the UDP checksum if possible
- [ ] Collect process flow information

b. Start a Wireshark capture.

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

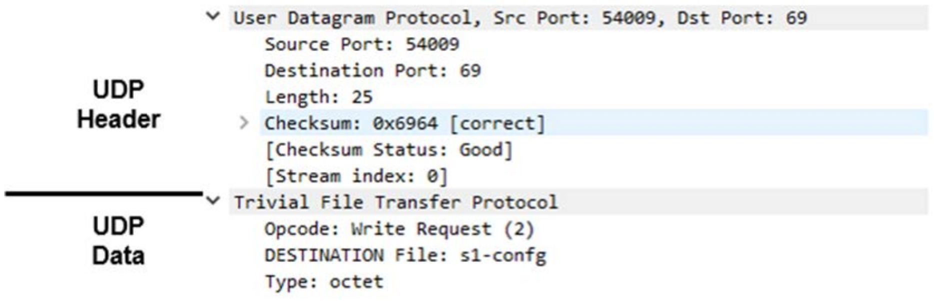
d. Stop the Wireshark capture.


*Ethernet
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tftp
Expressi

```
No.  Time       Source        Destination   Protocol  Length  Info
  10 17.006137  192.168.1.1   192.168.1.3   TFTP          64  Write Request, File: s1-confg, Transfer typ
  11 17.008212  192.168.1.3   192.168.1.1   TFTP          46  Acknowledgement, Block: 0
  12 17.012084  192.168.1.1   192.168.1.3   TFTP         558  Data Packet, Block: 1
  13 17.012376  192.168.1.3   192.168.1.1   TFTP          46  Acknowledgement, Block: 1
  14 17.014029  192.168.1.1   192.168.1.3   TFTP         558  Data Packet, Block: 2
  15 17.014133  192.168.1.3   192.168.1.1   TFTP          46  Acknowledgement, Block: 2
  16 17.017114  192.168.1.1   192.168.1.3   TFTP         105  Data Packet, Block: 3 (last)
  17 17.017219  192.168.1.3   192.168.1.1   TFTP          46  Acknowledgement, Block: 3
```

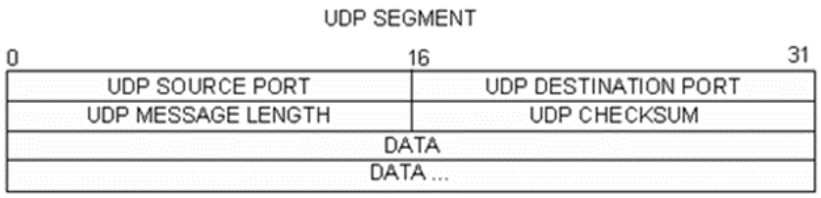
e. Set the filter to **tftp**. Your output should look similar to the output shown above. This TFTP transfer is used to analyze transport layer UDP operations.

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.



```
User Datagram Protocol, Src Port: 54009, Dst Port: 69
    Source Port: 54009
    Destination Port: 69
    Length: 25
  > Checksum: 0x6964 [correct]
    [Checksum Status: Good]
    [Stream index: 0]
Trivial File Transfer Protocol
    Opcode: Write Request (2)
    DESTINATION File: s1-confg
    Type: octet
```

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.



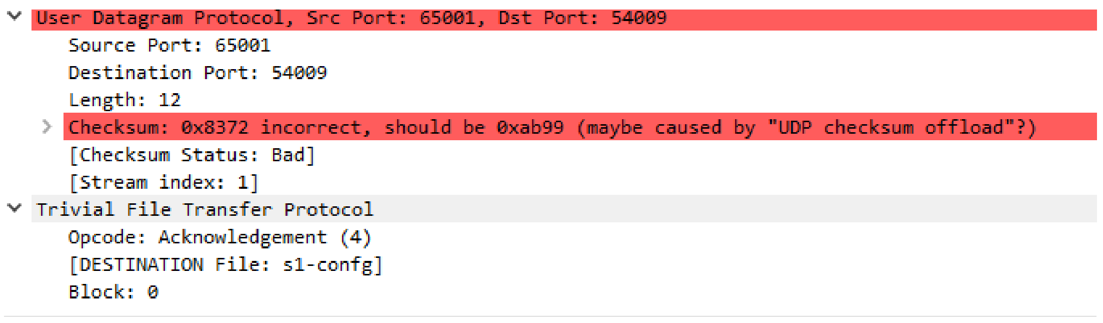
Using the Wireshark capture of the first UDP datagram, fill in information about the UDP header. The checksum value is a hexadecimal (base 16) value, denoted by the preceding 0x code:

Source IP address	
Destination IP address	
Source port number	
Destination port number	
UDP message length	
UDP checksum	

How does UDP verify datagram integrity?

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

Source IP address	
Destination IP address	
Source port number	
Destination port number	
UDP message length	
UDP checksum	



```
User Datagram Protocol, Src Port: 65001, Dst Port: 54009
    Source Port: 65001
    Destination Port: 54009
    Length: 12
  > Checksum: 0x8372 incorrect, should be 0xab99 (maybe caused by "UDP checksum offload"?)
    [Checksum Status: Bad]
    [Stream index: 1]
Trivial File Transfer Protocol
    Opcode: Acknowledgement (4)
    [DESTINATION File: s1-confg]
    Block: 0
```

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 from UDP?

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 more complicated, but the data field can be examined, and the configuration's 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 S1.
3) Remove the manual IP address from the PC and restore internet connectivity.