Name: ________________________________ Lab Time: ____________________

Cisco 1– Lab 5: TCP and UDP Packets in the Transport Layer

Background

The two protocols in the TCP/IP Transport Layer are the transmission control protocol (TCP), and user datagram protocol (UDP). Both protocols support upper-layer protocol communication. For example, TCP is used to provide Transport Layer support for the HTTP and FTP protocols, among others. UDP provides Transport Layer support for domain name services (DNS) and trivial file transfer protocol (TFTP), among others.

The ability to understand the parts of the TCP and UDP headers and the protocol operation are a critical skill for network engineers.

Task 1: Identify UDP header fields and operation using a Wireshark DNS session capture.

The UDP protocol is referred to as a non-reliable protocol. This doesn't mean it is a bad protocol; it just means that UDP doesn't attempt to verify if a sent packets arrived correctly, in the correct order, or even if they arrived at all. No delivery verification as attempted (although, error checking can be done to ensure a packet is received without corruption).

In this task, we will capture packets involved in a DNS lookup. DNS communication utilizes UDP rather than TCP.

Step 1: Capture a DNS session.

1. Open a command prompt. We will use this prompt to issue commands to generate network traffic. Click the **"Start" button > All Programs > Accessories** and then **Right –Click Command Prompt** and select "**Run as Administrator**". Leave this window open.
2. Start a Wireshark capture on the Intel interface on your computer. If you have trouble with this step, refer back to your previous labs or see your instructor.
3. In your Administrative Command prompt, use the ipconfig and ping commands as shown below to obtain an address from DNS and then send test messages:

```
C:\> ipconfig /flushdns
C:\> ping www.google.com
```

4. When finished, stop your Wireshark capture.

Step 2: Filter your Wireshark Display

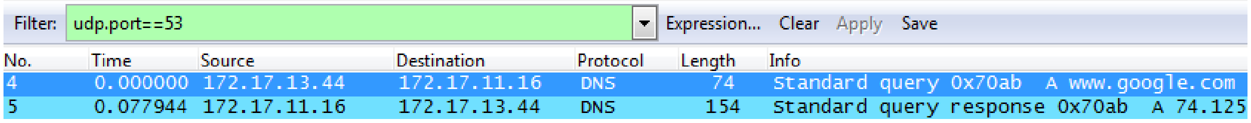
If you have more than just DNS packets (and you probably will), you may create a display filter to show only packets containing DNS.

5. In the filter field at the above the packet list pane, enter **udp.port==53** and click “Apply”.

Why did we specify UDP port 53 to filter our captured packets in order to see only DNS traffic?

Step 3: Analyze the UDP fields.

Look at your Wireshark capture. Your capture should be somewhat similar to the following capture:

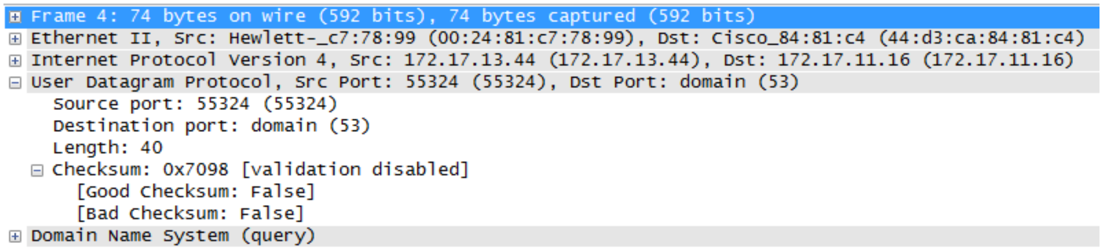


```
No.  Time      Source        Destination   Protocol Length Info
4    0.000000  172.17.13.44  172.17.11.16  DNS          74 Standard query 0x70ab  A www.google.com
5    0.077944  172.17.11.16  172.17.13.44  DNS         154 Standard query response 0x70ab  A 74.125
```

Figure 1: Packets filtered for DNS

Recall that in Wireshark, detailed UDP information is available in the Packet Detail Pane.

6. Highlight the DNS datagram that contains the **request for the address of www.google.com** from your computer (listed as “Standard query” in the packet list pane).
7. In the packet detail pane of Wireshark expand the UDP header by clicking on the protocol expand box. It may be necessary to adjust the middle window. The expanded UDP datagram should look similar to the following:



```
⊞ Frame 4: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)
⊞ Ethernet II, Src: Hewlett-_c7:78:99 (00:24:81:c7:78:99), Dst: Cisco_84:81:c4 (44:d3:ca:84:81:c4)
⊞ Internet Protocol Version 4, Src: 172.17.13.44 (172.17.13.44), Dst: 172.17.11.16 (172.17.11.16)
⊟ User Datagram Protocol, Src Port: 55324 (55324), Dst Port: domain (53)
    Source port: 55324 (55324)
    Destination port: domain (53)
    Length: 40
  ⊟ Checksum: 0x7098 [validation disabled]
      [Good Checksum: False]
      [Bad Checksum: False]
⊞ Domain Name System (query)
```

Figure 2: Details of UDP header shown

Notice that each UDP datagram identifies the UDP source port and UDP destination port.

8. Using your Wireshark capture of the DNS Address Query packet (a request from your computer), fill in information about the packet: Port numbers come from the UDP header; IP addresses come from the IP header:

Source IP Address:	
Destination IP Address:	
Source port number:	
Destination port number:	

9. Highlight the captured DNS datagram that contains the **response with the address of www.google.com** from the DNS server ("Standard query response…"). Fill in information about the UDP header:

Source IP Address:	
Destination IP Address:	
Source port number:	
Destination port number:	

10. Compare the source port in the first packet to the destination port in the second packet. How are these source and destination ports related? Explain why that is?

Task 2: Identify TCP header fields and operation using a Wireshark FTP session capture.

TCP sessions are well controlled and managed by information exchanged in the TCP header fields. In this task, a FTP session will be made to the classroom FTP server. Note: these steps assume you are in one of the network labs on NWTC's campus.

Step 1: Capture an FTP session.

1. If necessary, open a command line window.
2. Start a new Wireshark capture on the **Intel** network card.
3. Back in the command window, start an FTP connection to the classroom FTP server. Type the command:

```
C:\> ftp 172.17.11.16
```

4. When prompted for a user id, type **anonymous**. When prompted for a password, press **ENTER**.
5. After you are connected, terminate the FTP sessions in each command line window with the FTP `quit` command:

```
ftp> quit
```

6. Stop the Wireshark capture. You may close the command prompt if you wish.

While we didn't do anything "useful" with the FTP server, we have generated network traffic we can now analyze to see how TCP operates.

Step 2: Filter Wireshark to show only packets with your IP address

In this step you will configure a Display filter in Wireshark to show only packets coming from or go to your station's IP address. As the process is similar to that performed earlier in this lab, fewer details will be provided.

7. Enter a packet filter rule of "**ip.addr==*youripaddress***" where *youripaddress* is the IP address assigned to your machine; for example **ip.addr==172.17.13.153.** Apply your filter.

You might still have other packets (coming from or going to your machine) so we will also filter the packets in an additional way.

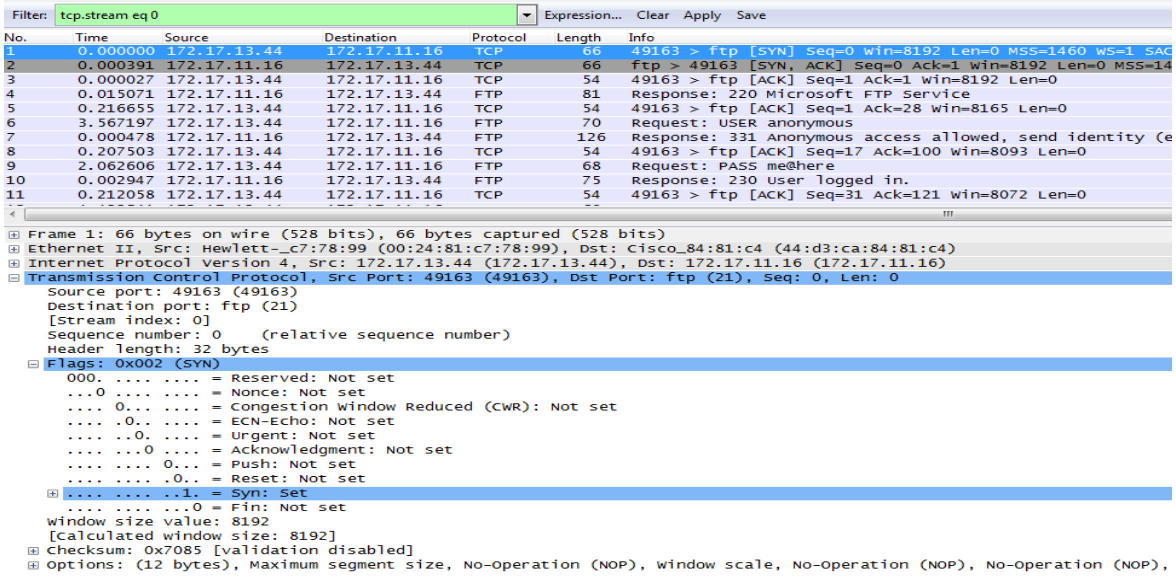
8. Right click on the first packet from your IP address to the IP address 172.17.11.16 and then select **Follow TCP Stream**. A new window will open showing an interpreted version of the conversation. Go ahead and close this "Follow TCP Stream" window. Wireshark will now show only those packets involved in the "conversation". Notice that your filter string will become **tcp.stream eq 0**. Your actual stream number may be different; this is ok.

Step 3: Analyze the TCP fields.

TCP is routinely used during a session to control datagram delivery, verify datagram arrival, and manage window size. Every time an FTP client contacts an FTP server, a new TCP session is started. At the conclusion of the data transfer, when the FTP session is finished, TCP performs an orderly shutdown and termination.

9. If necessary, switch to the Wireshark window.
10. Highlight the first TCP datagram from your computer; this packet should have the ftp port (21) as the destination port and have ONLY the Synchronization (SYN) bit set.

11. In the packet details pane, expand the TCP header by clicking on the protocol expand box. It may be necessary to adjust the size of the details pane to see everything. The expanded TCP datagram should look similar to the following figure:



Filter: tcp.stream eq 0

```
No.      Time     Source          Destination     Protocol Length Info
1        0.000000 172.17.13.44    172.17.11.16    TCP        66   49163 > ftp [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=1 SAC
2        0.000391 172.17.11.16    172.17.13.44    TCP        66   ftp > 49163 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=14
3        0.000027 172.17.13.44    172.17.11.16    TCP        54   49163 > ftp [ACK] Seq=1 Ack=1 Win=8192 Len=0
4        0.015071 172.17.11.16    172.17.13.44    FTP        81   Response: 220 Microsoft FTP Service
5        0.216655 172.17.13.44    172.17.11.16    TCP        54   49163 > ftp [ACK] Seq=1 Ack=28 Win=8165 Len=0
6        3.567197 172.17.13.44    172.17.11.16    FTP        70   Request: USER anonymous
7        0.000478 172.17.11.16    172.17.13.44    FTP       126   Response: 331 Anonymous access allowed, send identity (e
8        0.207503 172.17.13.44    172.17.11.16    TCP        54   49163 > ftp [ACK] Seq=17 Ack=100 Win=8093 Len=0
9        2.062606 172.17.13.44    172.17.11.16    FTP        68   Request: PASS me@here
10       0.002947 172.17.11.16    172.17.13.44    FTP        75   Response: 230 User logged in.
11       0.212058 172.17.13.44    172.17.11.16    TCP        54   49163 > ftp [ACK] Seq=31 Ack=121 Win=8072 Len=0
```

```
⊞ Frame 1: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)
⊞ Ethernet II, Src: Hewlett-_c7:78:99 (00:24:81:c7:78:99), Dst: Cisco_84:81:c4 (44:d3:ca:84:81:c4)
⊞ Internet Protocol Version 4, Src: 172.17.13.44 (172.17.13.44), Dst: 172.17.11.16 (172.17.11.16)
⊟ Transmission Control Protocol, Src Port: 49163 (49163), Dst Port: ftp (21), Seq: 0, Len: 0
    Source port: 49163 (49163)
    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: 0x7085 [validation disabled]
  ⊞ Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operation (NOP), No-Operation (NOP),
```

Figure 3: Details of TCP header

How is the first datagram in a TCP session identified (e.g., what flags are set to a value of 1)?

If necessary, refer back to the lecture notes and/or the book for an explanation of each field.

12. What are the source and destination IP address for the first packet from your computer to the FTP Server (only the SYN bit is set to 1); note that IP addresses will come from the packet summary or the IP header:

Source IP Address:	
Destination IP Address:	
TCP Sequence Number	
TCP Acknowledgement Number	NOT PRESENT IN THIS PACKET

13. Fill in information about the second packet from the FTP Server back to your computer (only the SYN and ACK bits are set to 1):

Source IP Address:	
Destination IP Address:	
TCP Sequence Number	
TCP Acknowledgement Number	

14. Fill in information about the third packet from your computer to the FTP Server (only the ACK bit is set to 1):

Source IP Address:	
Destination IP Address:	
TCP Sequence Number	
TCP Acknowledgement Number	

At this point the 3-way handshake has occurred between the two machines, the sequence numbers on each side have been synchronized, and the actual conversation is ready to start. ***Do not close your Wireshark session!***

Task 3: TCP Sequence/Acknowledgment Number Analysis

Background

We will now use Wireshark to examine the relationship between Sequence numbers and Acknowledgement numbers.

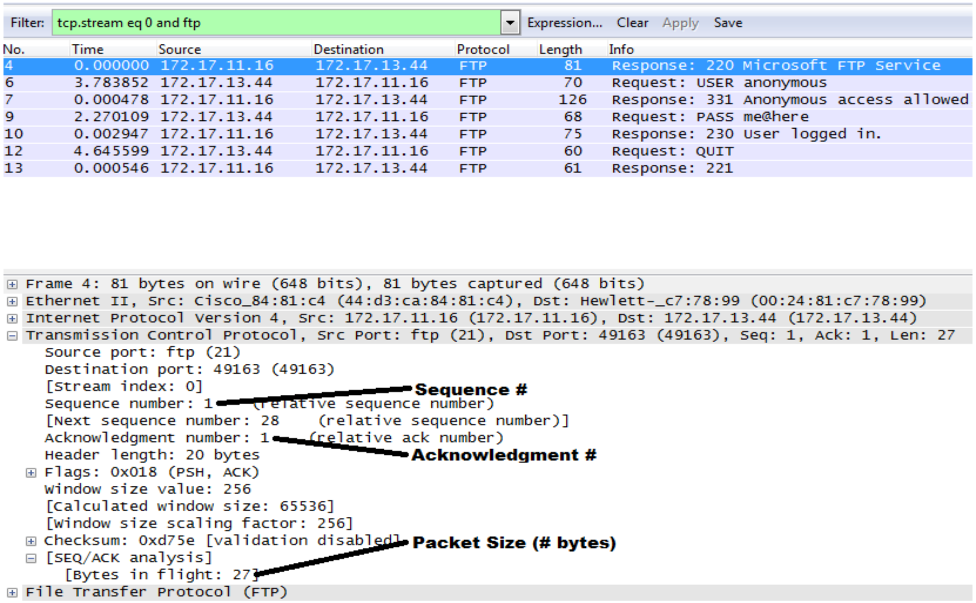
Looking at Seq, Ack, and bytes in flight

We have already looked at the 3-way handshake; we don't need to consider those packets in our conversation.

15. Change your display filter so that it not only specifies the tcp.stream but also includes the requirement that the packet have FTP information. Your filter should read

tcp.stream eq 0 and ftp

Recall that the actual **tcp.stream** value you have in your capture *may* be different than the one shown in my example. After adding the ftp requirement, your screen should appear similar to the following:



Filter: tcp.stream eq 0 and ftp

```
No.   Time      Source        Destination   Protocol  Length  Info
4     0.000000  172.17.11.16  172.17.13.44  FTP           81  Response: 220 Microsoft FTP Service
6     3.783852  172.17.13.44  172.17.11.16  FTP           70  Request: USER anonymous
7     0.000478  172.17.11.16  172.17.13.44  FTP          126  Response: 331 Anonymous access allowed
9     2.270109  172.17.13.44  172.17.11.16  FTP           68  Request: PASS me@here
10    0.002947  172.17.11.16  172.17.13.44  FTP           75  Response: 230 User logged in.
12    4.645599  172.17.13.44  172.17.11.16  FTP           60  Request: QUIT
13    0.000546  172.17.11.16  172.17.13.44  FTP           61  Response: 221
```

```
⊞ Frame 4: 81 bytes on wire (648 bits), 81 bytes captured (648 bits)
⊞ Ethernet II, Src: Cisco_84:81:c4 (44:d3:ca:84:81:c4), Dst: Hewlett-_c7:78:99 (00:24:81:c7:78:99)
⊞ Internet Protocol Version 4, Src: 172.17.11.16 (172.17.11.16), Dst: 172.17.13.44 (172.17.13.44)
⊟ Transmission Control Protocol, Src Port: ftp (21), Dst Port: 49163 (49163), Seq: 1, Ack: 1, Len: 27
    Source port: ftp (21)
    Destination port: 49163 (49163)
    [Stream index: 0]
    Sequence number: 1    (relative sequence number)
    [Next sequence number: 28    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    Header length: 20 bytes
  ⊞ Flags: 0x018 (PSH, ACK)
    Window size value: 256
    [Calculated window size: 65536]
    [Window size scaling factor: 256]
  ⊞ Checksum: 0xd75e [validation disabled]
  ⊟ [SEQ/ACK analysis]
      [Bytes in flight: 27]
⊞ File Transfer Protocol (FTP)
```

Figure 4: Sequence, Acknowledgement, and Size information

In Figure 4, the first packet after the handshake is selected and the details of the TCP header as well as the [SEQ/ACK analysis] details shown inside the header have been expanded. Inside the details of the TCP header, Wireshark shows the Sequence number, acknowledgement number, and number of bytes in this packet.

16. In ***your packet capture*** (from the FTP session you captured and filtered above), select the first packet shown. This is actually the first packet *after* the 3-way handshake. If necessary, expand out the Transmission Control Protocol header and the [SEQ/ACK analysis] details inside the TCP header. Fill in the following table for this packet:

Source IP Address:	
Destination IP Address:	
TCP Sequence Number	
TCP Acknowledgement Number	
Packet Size (# bytes in flight)	

17. What will the sequence number of the *next* packet sent from this source machine in this conversation be? Hint: add the number of bytes to the sequence number.

18. Select the next packet in the conversation. This *should* have the source and destination IP addresses switched; that is, the packet is moving the "other direction". Again, fill in the following table:

Source IP Address:	
Destination IP Address:	
TCP Sequence Number	
TCP Acknowledgement Number	
Packet Size (# bytes in flight)	

19. How is the Acknowledgement number sent from machine B to machine A (in the table from Step 18) related to the sequence number sent from machine A to machine B (in the table from Step 16)? In other words, what does the number in an acknowledgement represent?

20. Select the next packet in the conversation. This *should* have the same source and destination IP addresses as those from the first packet after the handshake (those addresses seen in the table in Step 16). Once more, fill in the following table:

Source IP Address:	
Destination IP Address:	
TCP Sequence Number	
TCP Acknowledgement Number	
Packet Size (# bytes in flight)	

21. How does the Sequence number in this third table relate to your predicted "next sequence" number you calculated in Step 17? In other words, was your prediction correct?

22. View the details of several other packets in the conversation. Keep track of the direction the packet is moving (e.g., watch the source & destination addresses) and the Sequence & Acknowledgement numbers to ensure your reasoning about the relationship of the Acknowledgement and Sequence numbers is correct.
23. When finished, close your Wireshark capture.

Task 4: Reflection.

This lab provided students with the opportunity to analyze TCP and UDP protocol operations from captured FTP and DNS sessions. TCP manages communication much differently from UDP, but 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.

Cleanup

Unless directed otherwise by the instructor, turn off power to the host computers. Remove anything that was brought into the lab, and leave the room ready for the next class.