Alexandria Deleon 6226100

Abstract

This lab was about using Wireshark to look at network traffic which included pinging certain hosts and analyzing the ethernet addresses of certain fields.

Introduction

For this lab, I set to use Wireshark to ping the hosts Ocelot.aul.fiu.edu, fiu.edu, and google.com and then take snippets of the results. I also set out to use Wireshark to analyze the ethernet addresses and type/length code of fields in a pcap file using the help of the table provided. I assume that I might have issues getting started as I am not familiar with Wireshark however, I believe I will be able to complete the lab successfully.

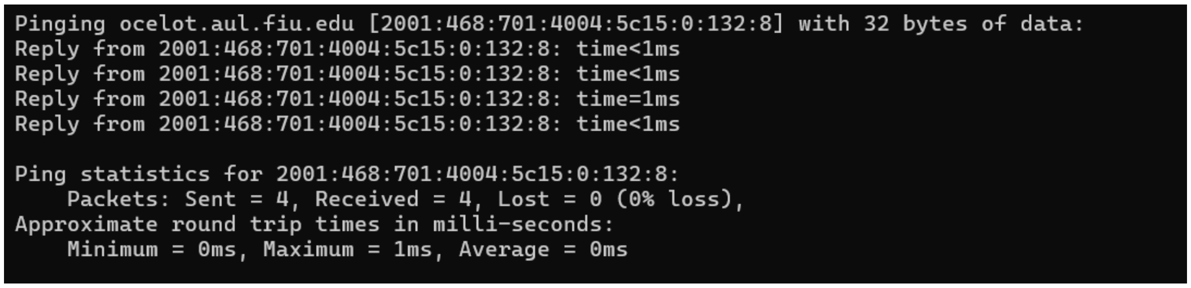
Details

To complete the first part of the lab I will access a lab machine, open Wireshark, and begin a capture. Next, I will use the command prompt and ping command to ping one of the hosts. I will then stop the capture and filter the field results by either using the icmp or icmpv6 filter. This will allow me to easily see the results of the host I pinged. I will then screenshot both the results of me pinging the host in the command prompt and the results of the pinged host in Wireshark. I will then repeat this process for the next two hosts. After this is completed, I will move on to the second part of the lab. Within the lab machine I will download the lab2.pcap file and open it in Wireshark. I will then use the information in the file and the table to answer the questions provided, which you can see the results of in the Questions section.

Results

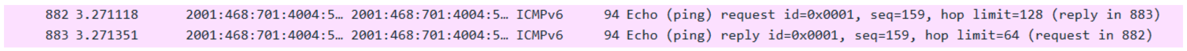
When I followed the procedures mentioned above, I was able to successfully complete the lab with few hiccups. At first, it was difficult to determine if the host was successfully pinged since I wasn't sure what to look for. However, after looking closely and using outside sources, I was able to realize that the source ip address shown in the capture is the same as the one shown in the command prompt and the reply and response indicators in the capture also indicate that the host was successfully pinged. The pictures below show the results of what was shown in both the command prompt and Wireshark after I completed this sections' procedures.

<u>Pinging host ocelot.aul.fiu.edu</u>



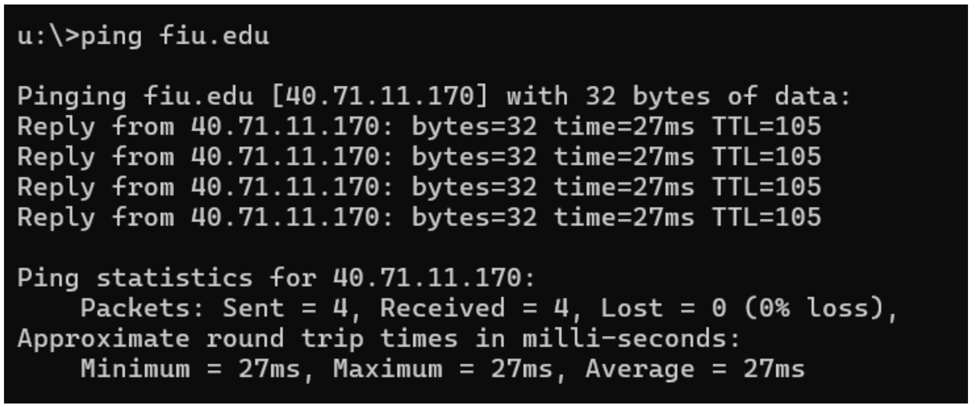
```
Pinging ocelot.aul.fiu.edu [2001:468:701:4004:5c15:0:132:8] with 32 bytes of data:
Reply from 2001:468:701:4004:5c15:0:132:8: time<1ms
Reply from 2001:468:701:4004:5c15:0:132:8: time<1ms
Reply from 2001:468:701:4004:5c15:0:132:8: time=1ms
Reply from 2001:468:701:4004:5c15:0:132:8: time<1ms

Ping statistics for 2001:468:701:4004:5c15:0:132:8:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms
```

Pinged host ocelot.aul.fiu.edu



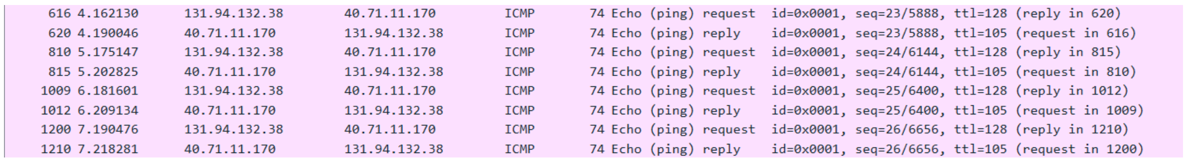
```
    882 3.271118      2001:468:701:4004:5… 2001:468:701:4004:5… ICMPv6     94 Echo (ping) request id=0x0001, seq=159, hop limit=128 (reply in 883)
    883 3.271351      2001:468:701:4004:5… 2001:468:701:4004:5… ICMPv6     94 Echo (ping) reply id=0x0001, seq=159, hop limit=64 (request in 882)
```

Pinging host ocelot.aul.fiu.edu



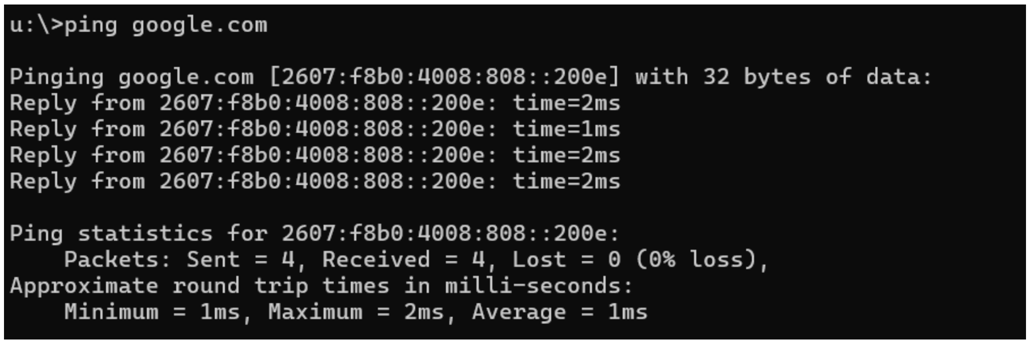
```
u:\>ping fiu.edu

Pinging fiu.edu [40.71.11.170] with 32 bytes of data:
Reply from 40.71.11.170: bytes=32 time=27ms TTL=105
Reply from 40.71.11.170: bytes=32 time=27ms TTL=105
Reply from 40.71.11.170: bytes=32 time=27ms TTL=105
Reply from 40.71.11.170: bytes=32 time=27ms TTL=105

Ping statistics for 40.71.11.170:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 27ms, Maximum = 27ms, Average = 27ms
```

Pinged host ocelot.aul.fiu.edu



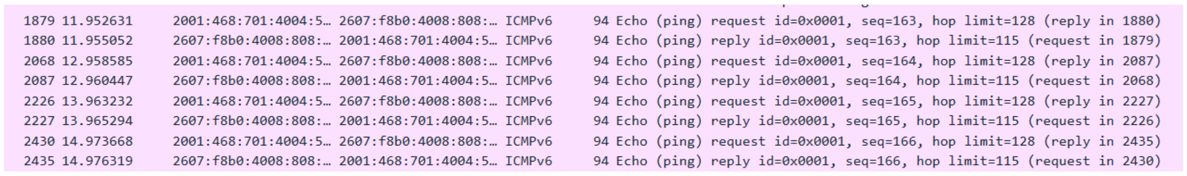
```
    616 4.162130      131.94.132.38       40.71.11.170        ICMP       74 Echo (ping) request  id=0x0001, seq=23/5888, ttl=128 (reply in 620)
    620 4.190046      40.71.11.170        131.94.132.38       ICMP       74 Echo (ping) reply    id=0x0001, seq=23/5888, ttl=105 (request in 616)
    810 5.175147      131.94.132.38       40.71.11.170        ICMP       74 Echo (ping) request  id=0x0001, seq=24/6144, ttl=128 (reply in 815)
    815 5.202825      40.71.11.170        131.94.132.38       ICMP       74 Echo (ping) reply    id=0x0001, seq=24/6144, ttl=105 (request in 810)
   1009 6.181601      131.94.132.38       40.71.11.170        ICMP       74 Echo (ping) request  id=0x0001, seq=25/6400, ttl=128 (reply in 1012)
   1012 6.209134      40.71.11.170        131.94.132.38       ICMP       74 Echo (ping) reply    id=0x0001, seq=25/6400, ttl=105 (request in 1009)
   1200 7.190476      131.94.132.38       40.71.11.170        ICMP       74 Echo (ping) request  id=0x0001, seq=26/6656, ttl=128 (reply in 1210)
   1210 7.218281      40.71.11.170        131.94.132.38       ICMP       74 Echo (ping) reply    id=0x0001, seq=26/6656, ttl=105 (request in 1200)
```

Pinging host google.com



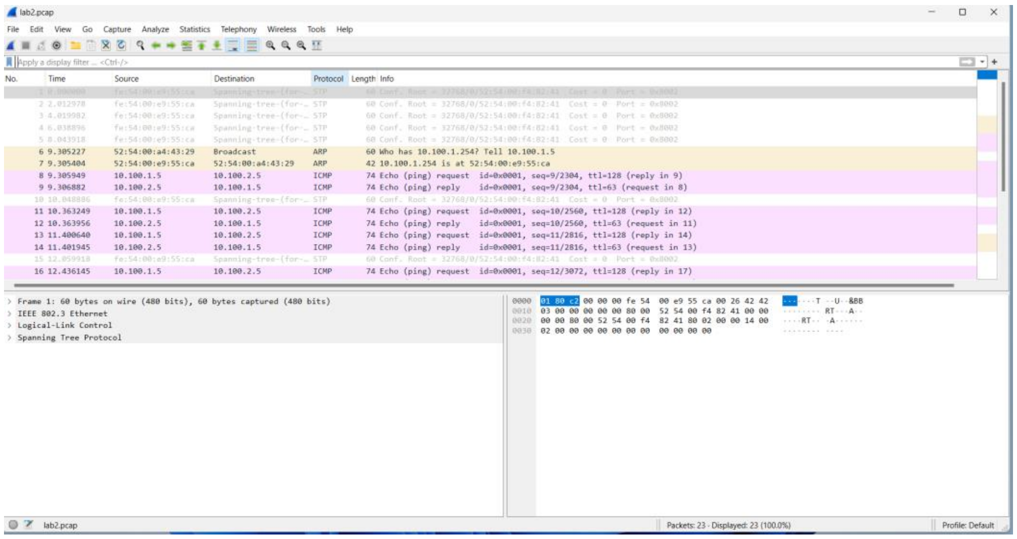
```
u:\>ping google.com

Pinging google.com [2607:f8b0:4008:808::200e] with 32 bytes of data:
Reply from 2607:f8b0:4008:808::200e: time=2ms
Reply from 2607:f8b0:4008:808::200e: time=1ms
Reply from 2607:f8b0:4008:808::200e: time=2ms
Reply from 2607:f8b0:4008:808::200e: time=2ms

Ping statistics for 2607:f8b0:4008:808::200e:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 2ms, Average = 1ms
```

Pinged host google.com



```
1879 11.952631    2001:468:701:4004:5… 2607:f8b0:4008:808:… ICMPv6    94 Echo (ping) request id=0x0001, seq=163, hop limit=128 (reply in 1880)
1880 11.955052    2607:f8b0:4008:808:… 2001:468:701:4004:5… ICMPv6    94 Echo (ping) reply id=0x0001, seq=163, hop limit=115 (request in 1879)
2068 12.958585    2001:468:701:4004:5… 2607:f8b0:4008:808:… ICMPv6    94 Echo (ping) request id=0x0001, seq=164, hop limit=128 (reply in 2087)
2087 12.960447    2607:f8b0:4008:808:… 2001:468:701:4004:5… ICMPv6    94 Echo (ping) reply id=0x0001, seq=164, hop limit=115 (request in 2068)
2226 13.963232    2001:468:701:4004:5… 2607:f8b0:4008:808:… ICMPv6    94 Echo (ping) request id=0x0001, seq=165, hop limit=128 (reply in 2227)
2227 13.965294    2607:f8b0:4008:808:… 2001:468:701:4004:5… ICMPv6    94 Echo (ping) reply id=0x0001, seq=165, hop limit=115 (request in 2226)
2430 14.973668    2001:468:701:4004:5… 2607:f8b0:4008:808:… ICMPv6    94 Echo (ping) request id=0x0001, seq=166, hop limit=128 (reply in 2435)
2435 14.976319    2607:f8b0:4008:808:… 2001:468:701:4004:5… ICMPv6    94 Echo (ping) reply id=0x0001, seq=166, hop limit=115 (request in 2430)
```

Afterwords, I began the second part of the lab and used the pcap file to answer the questions. Though I mainly completed this without issue, there were certain questions where I had to use outside sources for help. Below is a picture of what was shown when I opened the file in Wireshark.


lab2.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>
Packets: 23 · Displayed: 23 (100.0%)
Profile: Default

```
No.  Time        Source              Destination           Protocol Length Info
   1 0.000000    fe:54:00:e9:55:ca   Spanning-tree-(for-…  STP      60 Conf. Root = 32768/0/52:54:00:f4:82:41  Cost = 0  Port = 0x8002
   2 2.012978    fe:54:00:e9:55:ca   Spanning-tree-(for-…  STP      60 Conf. Root = 32768/0/52:54:00:f4:82:41  Cost = 0  Port = 0x8002
   3 4.019982    fe:54:00:e9:55:ca   Spanning-tree-(for-…  STP      60 Conf. Root = 32768/0/52:54:00:f4:82:41  Cost = 0  Port = 0x8002
   4 6.038896    fe:54:00:e9:55:ca   Spanning-tree-(for-…  STP      60 Conf. Root = 32768/0/52:54:00:f4:82:41  Cost = 0  Port = 0x8002
   5 8.043918    fe:54:00:e9:55:ca   Spanning-tree-(for-…  STP      60 Conf. Root = 32768/0/52:54:00:f4:82:41  Cost = 0  Port = 0x8002
   6 9.305227    52:54:00:a4:43:29   Broadcast             ARP      60 Who has 10.100.1.254? Tell 10.100.1.5
   7 9.305404    52:54:00:e9:55:ca   52:54:00:a4:43:29     ARP      42 10.100.1.254 is at 52:54:00:e9:55:ca
   8 9.305949    10.100.1.5          10.100.2.5            ICMP     74 Echo (ping) request  id=0x0001, seq=9/2304, ttl=128 (reply in 9)
   9 9.306882    10.100.2.5          10.100.1.5            ICMP     74 Echo (ping) reply    id=0x0001, seq=9/2304, ttl=63 (request in 8)
  10 10.048886   fe:54:00:e9:55:ca   Spanning-tree-(for-…  STP      60 Conf. Root = 32768/0/52:54:00:f4:82:41  Cost = 0  Port = 0x8002
  11 10.363249   10.100.1.5          10.100.2.5            ICMP     74 Echo (ping) request  id=0x0001, seq=10/2560, ttl=128 (reply in 12)
  12 10.363956   10.100.2.5          10.100.1.5            ICMP     74 Echo (ping) reply    id=0x0001, seq=10/2560, ttl=63 (request in 11)
  13 11.400640   10.100.1.5          10.100.2.5            ICMP     74 Echo (ping) request  id=0x0001, seq=11/2816, ttl=128 (reply in 14)
  14 11.401945   10.100.2.5          10.100.1.5            ICMP     74 Echo (ping) reply    id=0x0001, seq=11/2816, ttl=63 (request in 13)
  15 12.059918   fe:54:00:e9:55:ca   Spanning-tree-(for-…  STP      60 Conf. Root = 32768/0/52:54:00:f4:82:41  Cost = 0  Port = 0x8002
  16 12.436145   10.100.1.5          10.100.2.5            ICMP     74 Echo (ping) request  id=0x0001, seq=12/3072, ttl=128 (reply in 17)

> Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)
> IEEE 802.3 Ethernet
> Logical-Link Control
> Spanning Tree Protocol

0000  01 80 c2 00 00 00 fe 54  00 e9 55 ca 00 26 42 42   ·······T ··U··&BB
0010  03 00 00 00 00 00 80 00  52 54 00 f4 82 41 00 00   ········ RT···A··
0020  00 00 80 00 52 54 00 f4  82 41 80 02 00 00 14 00   ····RT·· ·A······
0030  02 00 00 00 00 00 00 00  00 00 00 00               ········ ····
```

Conclusion

This lab was on analyzing network traffic using Wireshark. To do this, certain hosts were pinged and analyzed in the program as well as a pcap file. While doing the lab, I was a bit confused at first on how to determine if the host were pinged but after doing some research, I soon understood what to look for. From following the procedures and completing the lab, I was able to garner a deeper understanding of how a network system works and how tools like Wireshark can be utilized. Specifically, I was able to learn more about what ethernet and ip addresses are used for and the significance of the ethernet type/length code.

Questions

Question 1 (2 points)

Identify a frame with a destination broadcast address in the lab2.pcap file. What is the source ethernet address of this packet?

- The source ethernet address is 52:54:00:a4:43:29.

Question 2 (2 points)

Identify a frame with a destination broadcast address in the lab2.pcap file. Use table above to identify the interface that transmitted this packet that was captured.

- The interface was Win10 (ws1) Ethernet.

Question 3 (2 points)

In the frame following the frame from question 2, what were the source and destination ethernet addresses.

- The source address was 52:54:00:e9:55:ca and the destination address was 52:54:00:a4:43:29.

Question 4 (3 points)

Identify a frame with a destination broadcast address in the lab2.pcap file. What is the ethernet TYPE or LENGTH code? How do you know it is a TYPE or LENGTH? Yes, wireshark will tell you, but how does wireshark know? What is the implication of it it being type or length?

- The destination broadcast address has the ethernet TYPE code of 0x0806. Wireshark knows it’s a TYPE because the value of the field is greater than 1500 (in decimal). This field implies that the frame is an ethernet II frame.

Question 5 (3 points)

Identify a frame with a source IP address of 10.100.2.5 in the lab2.pcap file. What is the source and destination Ethernet address of this frame? Which system and interface (from table above) sent this frame when it was captured? Was this system/interface the same as the system/interface that made the ping reply (Linux eth0)?

- The source ethernet address is 52:54:00:e9:55:ca and the destination ethernet address is 52:54:00:a4:43:29. The system and interface that sent this frame is monitor eth1 which is not the same as the interface that made the ping reply.

Question 6 (2 points)

Identify a frame with a destination ethernet address of 01:80:c2:00:00:00. (The top frame summary might identify this as "Spanning Tree for Bridges". What type of destination address is this? How do you know what type it is?

- The type of destination address is a multicast address. I know what type it is because it is shown in wireshark but also because its within the range of the IEEE.

Question 7 (2 points)

Identify a frame with a destination ethernet address of 01:80:c2:00:00:00. What is the ethernet TYPE or LENGTH code. How do you know it is a TYPE or LENGTH? Yes, wireshark will tell you, but how does wireshark know? What is the implication of it it being type or length?

- The ethernet LENGTH code is 38. Wireshark knows it's a LENGTH because the value of the field is less than 1500 (in decimal). The length field implies that the frame is an ethernet frame.

Question 8 (2 points)

Include in your report the ascii export all expanded output of the frame from Question 1.

```
No.     Time           Source                Destination           Protocol Length Info

      6 9.305227       52:54:00:a4:43:29     Broadcast             ARP      60     Who has
10.100.1.254? Tell 10.100.1.5



Frame 6: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)

Ethernet II, Src: 52:54:00:a4:43:29 (52:54:00:a4:43:29), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
```

```
    Destination: Broadcast (ff:ff:ff:ff:ff:ff)

        Address: Broadcast (ff:ff:ff:ff:ff:ff)

        .... ..1. .... .... .... .... = LG bit: Locally administered address (this is NOT the factory
default)

        .... ...1 .... .... .... .... = IG bit: Group address (multicast/broadcast)

    Source: 52:54:00:a4:43:29 (52:54:00:a4:43:29)

        Address: 52:54:00:a4:43:29 (52:54:00:a4:43:29)

        .... ..1. .... .... .... .... = LG bit: Locally administered address (this is NOT the factory
default)

        .... ...0 .... .... .... .... = IG bit: Individual address (unicast)

    Type: ARP (0x0806)

    Padding: 000000000000000000000000000000000000

Address Resolution Protocol (request)



0000  ff ff ff ff ff ff 52 54 00 a4 43 29 08 06 00 01   ......RT..C)....

0010  08 00 06 04 00 01 52 54 00 a4 43 29 0a 64 01 05   ......RT..C).d..

0020  00 00 00 00 00 00 0a 64 01 fe 00 00 00 00 00 00   .......d........

0030  00 00 00 00 00 00 00 00 00 00 00 00               ............
```

Question 9 (2 points)

Include in your report the ascii export all expanded output of the frame from Question 5.

```
No.     Time           Source                Destination           Protocol Length Info

      9 9.306882       10.100.2.5            10.100.1.5            ICMP     74     Echo (ping) reply
id=0x0001, seq=9/2304, ttl=63 (request in 8)
```

```
Frame 9: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)

Ethernet II, Src: 52:54:00:e9:55:ca (52:54:00:e9:55:ca), Dst: 52:54:00:a4:43:29
(52:54:00:a4:43:29)

    Destination: 52:54:00:a4:43:29 (52:54:00:a4:43:29)

        Address: 52:54:00:a4:43:29 (52:54:00:a4:43:29)

        .... ..1. .... .... .... .... = LG bit: Locally administered address (this is NOT the factory
default)

        .... ...0 .... .... .... .... = IG bit: Individual address (unicast)

    Source: 52:54:00:e9:55:ca (52:54:00:e9:55:ca)

        Address: 52:54:00:e9:55:ca (52:54:00:e9:55:ca)

        .... ..1. .... .... .... .... = LG bit: Locally administered address (this is NOT the factory
default)

        .... ...0 .... .... .... .... = IG bit: Individual address (unicast)

    Type: IPv4 (0x0800)

Internet Protocol Version 4, Src: 10.100.2.5, Dst: 10.100.1.5

Internet Control Message Protocol



0000  52 54 00 a4 43 29 52 54 00 e9 55 ca 08 00 45 00   RT..C)RT..U...E.

0010  00 3c 67 2d 00 00 3f 01 fc c2 0a 64 02 05 0a 64   .<g-..?....d...d

0020  01 05 00 00 55 52 00 01 00 09 61 62 63 64 65 66   ....UR....abcdef

0030  67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76   ghijklmnopqrstuv

0040  77 61 62 63 64 65 66 67 68 69                     wabcdefghi
```

Question 10 (2 points)

Include in your report the ascii export all expanded output of the frame from Question 6.

```
No.     Time           Source                Destination           Protocol Length Info

     10 10.048886      fe:54:00:e9:55:ca     Spanning-tree-(for-bridges)_00 STP      60
Conf. Root = 32768/0/52:54:00:f4:82:41  Cost = 0  Port = 0x8002



Frame 10: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)

IEEE 802.3 Ethernet

    Destination: Spanning-tree-(for-bridges)_00 (01:80:c2:00:00:00)

        Address: Spanning-tree-(for-bridges)_00 (01:80:c2:00:00:00)

        .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)

        .... ...1 .... .... .... .... = IG bit: Group address (multicast/broadcast)

    Source: fe:54:00:e9:55:ca (fe:54:00:e9:55:ca)

        Address: fe:54:00:e9:55:ca (fe:54:00:e9:55:ca)

        .... ..1. .... .... .... .... = LG bit: Locally administered address (this is NOT the factory
default)

        .... ...0 .... .... .... .... = IG bit: Individual address (unicast)

    Length: 38

    Padding: 0000000000000000

Logical-Link Control

Spanning Tree Protocol



0000  01 80 c2 00 00 00 fe 54 00 e9 55 ca 00 26 42 42   .......T..U..&BB

0010  03 00 00 00 00 00 80 00 52 54 00 f4 82 41 00 00   ........RT...A..

0020  00 00 80 00 52 54 00 f4 82 41 80 02 00 00 14 00   ....RT...A......
```

```
0030  02 00 00 00 00 00 00 00 00 00 00 00            ...........
```

Resources

The resources I used to complete the lab are the “Ethernet” web page from the Wireshark Wiki, which gives information on ethernet addresses and how to do certain things concerning them in Wireshark, the “Ethernet Frame Format” web page from GeeksforGeeks, which gives an overview on what the ethernet frame format is and provides insight into what the ethernet type/length field is, and the “Wireshark User’s Guide”, which gives detailed information on Wireshark and how to use it, which really helped in figuring out where to get started for the lab. All these sources are linked below.

Ethernet - Wireshark Wiki

Ethernet Frame Format - GeeksforGeeks

Wireshark User’s Guide