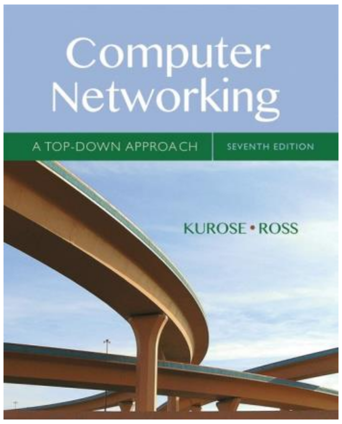
CS 447: Network and Data Communication

Wireshark Lab #03: Ethernet & ARP



Note: Make sure you produce your answers and any packet prints in PDF. Moodle will only accept PDF files.

In this lab, we'll investigate the Ethernet protocol and the ARP protocol. Before beginning this lab, you'll probably want to review sections 6.4.1 (Link-layer addressing and ARP) and 6.4.2 (Ethernet) in the text[1]. RFC 826 (ftp://ftp.rfc-editor.org/in-notes/std/std37.txt) contains the gory details of the ARP protocol, which is used by an IP device to determine the IP address of a remote interface whose Ethernet address is known.

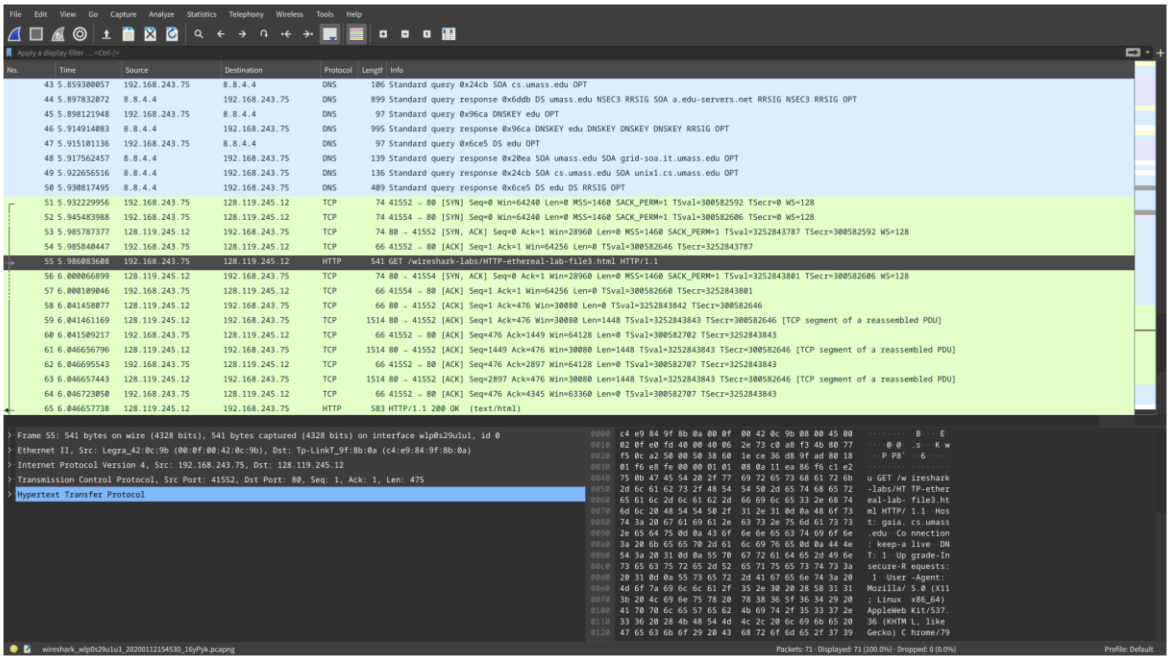
Capturing and Analyzing Ethernet Frames

Let's begin by capturing a set of Ethernet frames to study. Do the following[2]:

- First, make sure your browser's cache is empty.
- Start up the Wireshark packet sniffer
- Enter the following URL into your browser http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-lab-file3.html. Your browser should display the rather lengthy US Bill of Rights.
- Stop Wireshark packet capture. First, find the packet numbers (the leftmost column in the upper Wireshark window) of the HTTP GET message that was sent from your computer to gaia.cs.umass.edu, as well as the beginning of the HTTP response message sent to your computer by gaia.cs.umass.edu. You should see a screen that looks something like this (where packet 55 in the screenshot below contains the HTTP GET message).

[1] References to figures and sections are for the 7th edition of text, *Computer Networks, A Top-down Approach, 7th ed.*, J.F. Kurose and K.W. Ross, Addison-Wesley/Pearson, 2016

[2] If you are unable to run Wireshark live on a computer, you can download the zip file http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.zip and extract the file *ethernet--ethereal-trace-1*. The traces in this zip file were collected by Wireshark running on one of the author's computers, while performing the steps indicated in the Wireshark lab. Once you have downloaded the trace, you can load it into Wireshark and view the trace using the *File* pull down menu, choosing *Open*, and then selecting the ethernet-ethereal-trace-1 trace file. You can then use this trace file to answer the questions below.



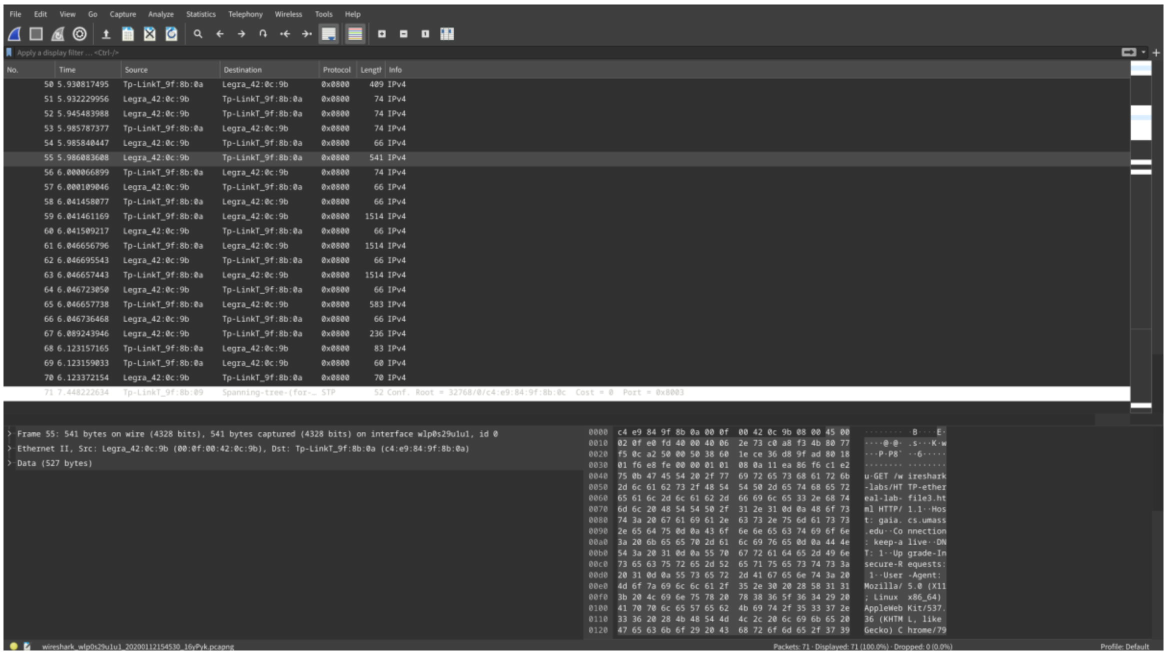
```
No.   Time         Source           Destination      Protocol Length Info
   43 5.859300057  192.168.243.75   8.8.4.4          DNS       106 Standard query 0x24cb SOA cs.umass.edu OPT
   44 5.897832072  8.8.4.4          192.168.243.75   DNS       899 Standard query response 0x6ddb DS umass.edu NSEC3 RRSIG SOA a.edu-servers.net RRSIG NSEC3 RRSIG OPT
   45 5.898121948  192.168.243.75   8.8.4.4          DNS        97 Standard query 0x96ca DNSKEY edu OPT
   46 5.914914083  8.8.4.4          192.168.243.75   DNS       995 Standard query response 0x96ca DNSKEY edu DNSKEY DNSKEY DNSKEY RRSIG OPT
   47 5.915101136  192.168.243.75   8.8.4.4          DNS        97 Standard query 0x6ce5 DS edu OPT
   48 5.917562457  8.8.4.4          192.168.243.75   DNS       139 Standard query response 0x20ea SOA umass.edu SOA grid-soa.it.umass.edu OPT
   49 5.922656516  8.8.4.4          192.168.243.75   DNS       136 Standard query response 0x24cb SOA cs.umass.edu SOA unix1.cs.umass.edu OPT
   50 5.930817495  8.8.4.4          192.168.243.75   DNS       409 Standard query response 0x6ce5 DS edu DS RRSIG OPT
   51 5.932229956  192.168.243.75   128.119.245.12   TCP        74 41552 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=300582592 TSecr=0 WS=128
   52 5.945483988  192.168.243.75   128.119.245.12   TCP        74 41554 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=300582606 TSecr=0 WS=128
   53 5.985787377  128.119.245.12   192.168.243.75   TCP        74 80 → 41552 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=3252843787 TSecr=300582592 WS=128
   54 5.985840447  192.168.243.75   128.119.245.12   TCP        66 41552 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=300582646 TSecr=3252843787
   55 5.986083608  192.168.243.75   128.119.245.12   HTTP      541 GET /wireshark-labs/HTTP-ethereal-lab-file3.html HTTP/1.1
   56 6.000066899  128.119.245.12   192.168.243.75   TCP        74 80 → 41554 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=3252843801 TSecr=300582606 WS=128
   57 6.000109046  192.168.243.75   128.119.245.12   TCP        66 41554 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=300582660 TSecr=3252843801
   58 6.041458077  128.119.245.12   192.168.243.75   TCP        66 80 → 41552 [ACK] Seq=1 Ack=476 Win=30080 Len=0 TSval=3252843842 TSecr=300582646
   59 6.041461169  128.119.245.12   192.168.243.75   TCP      1514 80 → 41552 [ACK] Seq=1 Ack=476 Win=30080 Len=1448 TSval=3252843843 TSecr=300582646 [TCP segment of a reassembled PDU]
   60 6.041509217  192.168.243.75   128.119.245.12   TCP        66 41552 → 80 [ACK] Seq=476 Ack=1449 Win=64128 Len=0 TSval=300582702 TSecr=3252843843
   61 6.046656796  128.119.245.12   192.168.243.75   TCP      1514 80 → 41552 [ACK] Seq=1449 Ack=476 Win=30080 Len=1448 TSval=3252843843 TSecr=300582646 [TCP segment of a reassembled PDU]
   62 6.046695543  192.168.243.75   128.119.245.12   TCP        66 41552 → 80 [ACK] Seq=476 Ack=2897 Win=64128 Len=0 TSval=300582707 TSecr=3252843843
   63 6.046657443  128.119.245.12   192.168.243.75   TCP      1514 80 → 41552 [ACK] Seq=2897 Ack=476 Win=30080 Len=1448 TSval=3252843843 TSecr=300582646 [TCP segment of a reassembled PDU]
   64 6.046723050  192.168.243.75   128.119.245.12   TCP        66 41552 → 80 [ACK] Seq=476 Ack=4345 Win=63360 Len=0 TSval=300582707 TSecr=3252843843
   65 6.046657738  128.119.245.12   192.168.243.75   HTTP      583 HTTP/1.1 200 OK  (text/html)
```

```
> Frame 55: 541 bytes on wire (4328 bits), 541 bytes captured (4328 bits) on interface wlp0s29u1u1, id 0
> Ethernet II, Src: Legra_42:0c:9b (00:0f:00:42:0c:9b), Dst: Tp-LinkT_9f:8b:0a (c4:e9:84:9f:8b:0a)
> Internet Protocol Version 4, Src: 192.168.243.75, Dst: 128.119.245.12
> Transmission Control Protocol, Src Port: 41552, Dst Port: 80, Seq: 1, Ack: 1, Len: 475
> Hypertext Transfer Protocol
```

```
0000  c4 e9 84 9f 8b 0a 00 0f  00 42 0c 9b 08 00 45 00   ········ ·B····E·
0010  02 0f e0 fd 40 00 40 06  2e 73 c0 a8 f3 4b 80 77   ····@·@· .s···K·w
0020  f5 0c a2 50 00 50 38 60  1e ce 36 d8 9f ad 80 18   ···P·P8` ··6·····
0030  01 f6 e8 fe 00 00 01 01  08 0a 11 ea 86 f6 c1 e2   ········ ········
0040  75 0b 47 45 54 20 2f 77  69 72 65 73 68 61 72 6b   u·GET /w ireshark
0050  2d 6c 61 62 73 2f 48 54  54 50 2d 65 74 68 65 72   -labs/HT TP-ether
0060  65 61 6c 2d 6c 61 62 2d  66 69 6c 65 33 2e 68 74   eal-lab- file3.ht
0070  6d 6c 20 48 54 54 50 2f  31 2e 31 0d 0a 48 6f 73   ml HTTP/ 1.1··Hos
0080  74 3a 20 67 61 69 61 2e  63 73 2e 75 6d 61 73 73   t: gaia. cs.umass
0090  2e 65 64 75 0d 0a 43 6f  6e 6e 65 63 74 69 6f 6e   .edu··Co nnection
00a0  3a 20 6b 65 65 70 2d 61  6c 69 76 65 0d 0a 44 4e   : keep-a live··DN
00b0  54 3a 20 31 0d 0a 55 70  67 72 61 64 65 2d 49 6e   T: 1··Up grade-In
00c0  73 65 63 75 72 65 2d 52  65 71 75 65 73 74 73 3a   secure-R equests:
00d0  20 31 0d 0a 55 73 65 72  2d 41 67 65 6e 74 3a 20    1··User -Agent: 
00e0  4d 6f 7a 69 6c 6c 61 2f  35 2e 30 20 28 58 31 31   Mozilla/ 5.0 (X11
00f0  3b 20 4c 69 6e 75 78 20  78 38 36 5f 36 34 29 20   ; Linux  x86_64) 
0100  41 70 70 6c 65 57 65 62  4b 69 74 2f 35 33 37 2e   AppleWeb Kit/537.
0110  33 36 20 28 4b 48 54 4d  4c 2c 20 6c 69 6b 65 20   36 (KHTM L, like 
0120  47 65 63 6b 6f 29 20 43  68 72 6f 6d 65 2f 37 39   Gecko) C hrome/79
```

wireshark_wlp0s29u1u1_20200112154530_16yPyk.pcapng   Packets: 71 · Displayed: 71 (100.0%) · Dropped: 0 (0.0%)   Profile: Default

- Since this lab is about Ethernet and ARP, we're not interested in IP or higher-layer protocols. So let's change Wireshark's "listing of captured packets" window so that it shows information only about protocols below IP. To have Wireshark do this, select *Analyze->Enabled Protocols*. Then uncheck the IPv4 box (use the filter at the top) and select OK. You should now see a Wireshark window that looks like the following:



```
No.   Time         Source             Destination        Protocol Length Info
   50 5.930817495  Tp-LinkT_9f:8b:0a  Legra_42:0c:9b     0x0800    409 IPv4
   51 5.932229956  Legra_42:0c:9b     Tp-LinkT_9f:8b:0a  0x0800     74 IPv4
   52 5.945483988  Legra_42:0c:9b     Tp-LinkT_9f:8b:0a  0x0800     74 IPv4
   53 5.985787377  Tp-LinkT_9f:8b:0a  Legra_42:0c:9b     0x0800     74 IPv4
   54 5.985840447  Legra_42:0c:9b     Tp-LinkT_9f:8b:0a  0x0800     66 IPv4
   55 5.986083608  Legra_42:0c:9b     Tp-LinkT_9f:8b:0a  0x0800    541 IPv4
   56 6.000066899  Tp-LinkT_9f:8b:0a  Legra_42:0c:9b     0x0800     74 IPv4
   57 6.000109046  Legra_42:0c:9b     Tp-LinkT_9f:8b:0a  0x0800     66 IPv4
   58 6.041458077  Tp-LinkT_9f:8b:0a  Legra_42:0c:9b     0x0800     66 IPv4
   59 6.041461169  Tp-LinkT_9f:8b:0a  Legra_42:0c:9b     0x0800   1514 IPv4
   60 6.041509217  Legra_42:0c:9b     Tp-LinkT_9f:8b:0a  0x0800     66 IPv4
   61 6.046656796  Tp-LinkT_9f:8b:0a  Legra_42:0c:9b     0x0800   1514 IPv4
   62 6.046695543  Legra_42:0c:9b     Tp-LinkT_9f:8b:0a  0x0800     66 IPv4
   63 6.046657443  Tp-LinkT_9f:8b:0a  Legra_42:0c:9b     0x0800   1514 IPv4
   64 6.046723050  Legra_42:0c:9b     Tp-LinkT_9f:8b:0a  0x0800     66 IPv4
   65 6.046657738  Tp-LinkT_9f:8b:0a  Legra_42:0c:9b     0x0800    583 IPv4
   66 6.046736468  Legra_42:0c:9b     Tp-LinkT_9f:8b:0a  0x0800     66 IPv4
   67 6.089243946  Legra_42:0c:9b     Tp-LinkT_9f:8b:0a  0x0800    236 IPv4
   68 6.123157165  Tp-LinkT_9f:8b:0a  Legra_42:0c:9b     0x0800     83 IPv4
   69 6.123159033  Tp-LinkT_9f:8b:0a  Legra_42:0c:9b     0x0800     60 IPv4
   70 6.123372154  Legra_42:0c:9b     Tp-LinkT_9f:8b:0a  0x0800     70 IPv4
   71 7.448222634  Tp-LinkT_9f:8b:09  Spanning-tree-(for-… STP      52 Conf. Root = 32768/0/c4:e9:84:9f:8b:0c  Cost = 0  Port = 0x8003
```

```
> Frame 55: 541 bytes on wire (4328 bits), 541 bytes captured (4328 bits) on interface wlp0s29u1u1, id 0
> Ethernet II, Src: Legra_42:0c:9b (00:0f:00:42:0c:9b), Dst: Tp-LinkT_9f:8b:0a (c4:e9:84:9f:8b:0a)
> Data (527 bytes)
```

```
0000  c4 e9 84 9f 8b 0a 00 0f  00 42 0c 9b 08 00 45 00   ········ ·B····E·
0010  02 0f e0 fd 40 00 40 06  2e 73 c0 a8 f3 4b 80 77   ····@·@· .s···K·w
0020  f5 0c a2 50 00 50 38 60  1e ce 36 d8 9f ad 80 18   ···P·P8` ··6·····
0030  01 f6 e8 fe 00 00 01 01  08 0a 11 ea 86 f6 c1 e2   ········ ········
0040  75 0b 47 45 54 20 2f 77  69 72 65 73 68 61 72 6b   u·GET /w ireshark
0050  2d 6c 61 62 73 2f 48 54  54 50 2d 65 74 68 65 72   -labs/HT TP-ether
0060  65 61 6c 2d 6c 61 62 2d  66 69 6c 65 33 2e 68 74   eal-lab- file3.ht
0070  6d 6c 20 48 54 54 50 2f  31 2e 31 0d 0a 48 6f 73   ml HTTP/ 1.1··Hos
0080  74 3a 20 67 61 69 61 2e  63 73 2e 75 6d 61 73 73   t: gaia. cs.umass
0090  2e 65 64 75 0d 0a 43 6f  6e 6e 65 63 74 69 6f 6e   .edu··Co nnection
00a0  3a 20 6b 65 65 70 2d 61  6c 69 76 65 0d 0a 44 4e   : keep-a live··DN
00b0  54 3a 20 31 0d 0a 55 70  67 72 61 64 65 2d 49 6e   T: 1··Up grade-In
00c0  73 65 63 75 72 65 2d 52  65 71 75 65 73 74 73 3a   secure-R equests:
00d0  20 31 0d 0a 55 73 65 72  2d 41 67 65 6e 74 3a 20    1··User -Agent: 
00e0  4d 6f 7a 69 6c 6c 61 2f  35 2e 30 20 28 58 31 31   Mozilla/ 5.0 (X11
00f0  3b 20 4c 69 6e 75 78 20  78 38 36 5f 36 34 29 20   ; Linux  x86_64) 
0100  41 70 70 6c 65 57 65 62  4b 69 74 2f 35 33 37 2e   AppleWeb Kit/537.
0110  33 36 20 28 4b 48 54 4d  4c 2c 20 6c 69 6b 65 20   36 (KHTM L, like 
0120  47 65 63 6b 6f 29 20 43  68 72 6f 6d 65 2f 37 39   Gecko) C hrome/79
```

wireshark_wlp0s29u1u1_20200112154530_16yPyk.pcapng   Packets: 71 · Displayed: 71 (100.0%) · Dropped: 0 (0.0%)   Profile: Default

In order to answer the following questions, you'll need to look into the packet details and packet content windows (lower left and right panels on the screenshots. You could have a different arrangement).

Select the Ethernet frame containing the HTTP GET message. (Recall that the HTTP GET message is carried inside of a TCP segment, which is carried inside of an IP datagram, which is carried inside of an Ethernet frame; reread section 1.5.2 in the text if you find this encapsulation a bit confusing). Expand the Ethernet II information in the packet details window. Note that the contents of the Ethernet frame (header as well as payload) are displayed in the packet contents window.

Answer the following questions, based on the contents of the Ethernet frame containing the HTTP GET message. Whenever possible, when answering a question, you should hand in a printout of the packet(s) within the trace that you used to answer the question asked. Annotate the printout[3] to explain your answer. To print a packet, use *File → Print*, choose *Selected packet only*, choose *Packet summary line*, and select the minimum amount of packet detail that you need to answer the question.

1. What is the 48-bit Ethernet address of your computer?
2. What is the 48-bit destination address in the Ethernet frame? Is this the Ethernet address of gaia.cs.umass.edu? (Hint: the answer is no). What device has this as its Ethernet address? [Note: this is an important question, and one that students sometimes get wrong. Re-read pages 468-469 in the text and make sure you understand the answer here.]
3. Give the hexadecimal value for the two-byte Frame type field. What upper layer protocol does this correspond to?
4. How many bytes from the very start of the Ethernet frame does the ASCII "G" in "GET" appear in the Ethernet frame?

Next, answer the following questions, based on the contents of the Ethernet frame containing the first byte of the HTTP response message.

5. What is the value of the Ethernet source address? Is this the address of your computer, or of gaia.cs.umass.edu (Hint: the answer is no). What device has this as its Ethernet address?
6. What is the destination address in the Ethernet frame? Is this the Ethernet address of your computer?
7. Give the hexadecimal value for the two-byte Frame type field. What upper layer protocol does this correspond to?
8. How many bytes from the very start of the Ethernet frame does the ASCII "O" in "OK" (i.e., the HTTP response code) appear in the Ethernet frame?

The Address Resolution Protocol

In this section, we'll observe the ARP protocol in action. We strongly recommend that you re-read section 6.4.1 in the text before proceeding.

ARP Caching

Recall that the ARP protocol typically maintains a cache of IP-to-Ethernet address translation pairs on your computer. The `arp` command (in both MSDOS and Linux/Unix) is used to view and manipulate the contents of this cache. Since the `arp` command and the ARP protocol have the same name, it's understandably easy to confuse them. But keep in mind that they are different - the `arp` command is used to view and manipulate the ARP cache contents, while the ARP protocol defines the format and meaning of the messages sent and received, and defines the actions taken on message transmission and receipt.

Let's take a look at the contents of the ARP cache on your computer:

- **MS-DOS**. The `arp` command is in *c:\windows\system32*, so type either "`arp`" or "`c:\windows\system32\arp`" in the MS-DOS command line (without quotation marks).

[3] What is meant by "annotate"? Please highlight and annotate where you've found answers and add an explaination of what you've found in what you've highlighted.

- **Linux/Unix/MacOS**. The executable for the `arp` command can be in various places. Popular locations are */sbin/arp* (for linux) and */usr/etc/arp* (for some Unix variants).

If on Windows try `arp -a`. If on Linux try `arp -n`. This will display the current ARP cache on your computer.

9. Write down the contents of your computer's ARP cache. What is the meaning of each column value?

In order to observe your computer sending and receiving ARP messages, we'll need to clear the ARP cache, since otherwise your computer is likely to find a needed IP-Ethernet address translation pair in its cache and consequently not need to send out an ARP message.

- **MS-DOS**. The MS-DOS "`arp -d *`" command will clear your ARP cache. The `-d` flag indicates a deletion operation, and the * is the wildcard that says to delete all table entries.

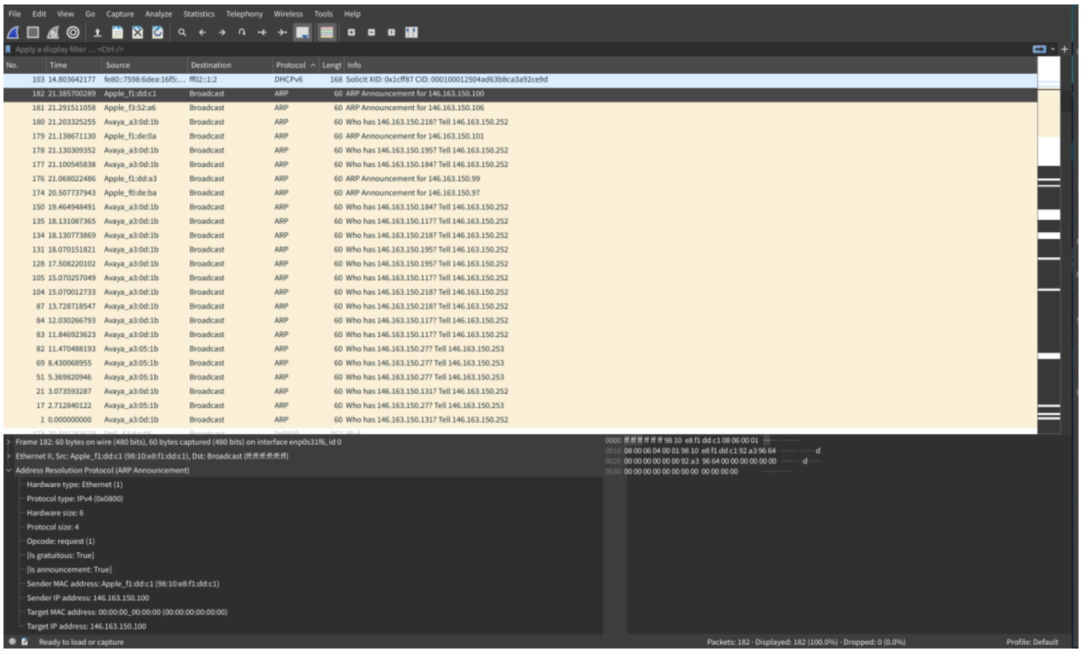
- **Linux/Unix/MacOS**. The "`ip -s -s neigh flush all`" on Linux will clear your ARP cache. You may have to elevate yourself to root to run this command.

Observing ARP in Action

Do the following[4]:

- Clear your ARP cache, as described above.
- Next, make sure your browser's cache is empty.
- Start up the Wireshark packet sniffer
- Enter the following URL into your browser http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-lab-file3.html. Your browser should again display the rather lengthy US Bill of Rights.
- Stop Wireshark packet capture. Again, we're not interested in IP or higher-layer protocols, so change Wireshark's "listing of captured packets" window so that it shows information only about protocols below IP. To have Wireshark do this, select *Analyze->Enabled Protocols*. Then uncheck the IPv4 box and select OK. You should now see a Wireshark window that looks like the following:

[4] The *ethernet-ethereal-trace-1* trace file in http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.zip was created using the steps below (in particular after the ARP cache had been flushed).


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

No.	Time	Source	Destination	Protocol	Lengt	Info
103	14.803642177	fe80::7598:6dea:16f5:…	ff02::1:2	DHCPv6	168	Solicit XID: 0x1cff87 CID: 000100012504ad63b8ca3a92ce9d
182	21.385700289	Apple_f1:dd:c1	Broadcast	ARP	60	ARP Announcement for 146.163.150.100
181	21.291511058	Apple_f3:52:a6	Broadcast	ARP	60	ARP Announcement for 146.163.150.106
180	21.203325255	Avaya_a3:0d:1b	Broadcast	ARP	60	Who has 146.163.150.218? Tell 146.163.150.252
179	21.138671130	Apple_f1:de:0a	Broadcast	ARP	60	ARP Announcement for 146.163.150.101
178	21.130309352	Avaya_a3:0d:1b	Broadcast	ARP	60	Who has 146.163.150.195? Tell 146.163.150.252
177	21.100545838	Avaya_a3:0d:1b	Broadcast	ARP	60	Who has 146.163.150.184? Tell 146.163.150.252
176	21.068022486	Apple_f1:dd:a3	Broadcast	ARP	60	ARP Announcement for 146.163.150.99
174	20.507737943	Apple_f0:de:ba	Broadcast	ARP	60	ARP Announcement for 146.163.150.97
150	19.464948491	Avaya_a3:0d:1b	Broadcast	ARP	60	Who has 146.163.150.184? Tell 146.163.150.252
135	18.131087365	Avaya_a3:0d:1b	Broadcast	ARP	60	Who has 146.163.150.117? Tell 146.163.150.252
134	18.130773869	Avaya_a3:0d:1b	Broadcast	ARP	60	Who has 146.163.150.218? Tell 146.163.150.252
131	18.070151821	Avaya_a3:0d:1b	Broadcast	ARP	60	Who has 146.163.150.195? Tell 146.163.150.252
128	17.508220102	Avaya_a3:0d:1b	Broadcast	ARP	60	Who has 146.163.150.195? Tell 146.163.150.252
105	15.070257049	Avaya_a3:0d:1b	Broadcast	ARP	60	Who has 146.163.150.117? Tell 146.163.150.252
104	15.070012733	Avaya_a3:0d:1b	Broadcast	ARP	60	Who has 146.163.150.218? Tell 146.163.150.252
87	13.728718547	Avaya_a3:0d:1b	Broadcast	ARP	60	Who has 146.163.150.218? Tell 146.163.150.252
84	12.030266793	Avaya_a3:0d:1b	Broadcast	ARP	60	Who has 146.163.150.117? Tell 146.163.150.252
83	11.846923623	Avaya_a3:0d:1b	Broadcast	ARP	60	Who has 146.163.150.117? Tell 146.163.150.252
82	11.470488193	Avaya_a3:05:1b	Broadcast	ARP	60	Who has 146.163.150.27? Tell 146.163.150.253
69	8.430068955	Avaya_a3:05:1b	Broadcast	ARP	60	Who has 146.163.150.27? Tell 146.163.150.253
51	5.369820946	Avaya_a3:05:1b	Broadcast	ARP	60	Who has 146.163.150.27? Tell 146.163.150.253
21	3.073593287	Avaya_a3:0d:1b	Broadcast	ARP	60	Who has 146.163.150.131? Tell 146.163.150.252
17	2.712840122	Avaya_a3:05:1b	Broadcast	ARP	60	Who has 146.163.150.27? Tell 146.163.150.253
1	0.000000000	Avaya_a3:0d:1b	Broadcast	ARP	60	Who has 146.163.150.131? Tell 146.163.150.252

```
> Frame 182: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface enp0s31f6, id 0
> Ethernet II, Src: Apple_f1:dd:c1 (98:10:e8:f1:dd:c1), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
v Address Resolution Protocol (ARP Announcement)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    [Is gratuitous: True]
    [Is announcement: True]
    Sender MAC address: Apple_f1:dd:c1 (98:10:e8:f1:dd:c1)
    Sender IP address: 146.163.150.100
    Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Target IP address: 146.163.150.100
```

```
0000  ff ff ff ff ff ff 98 10  e8 f1 dd c1 08 06 00 01   ················
0010  08 00 06 04 00 01 98 10  e8 f1 dd c1 92 a3 96 64   ···············d
0020  00 00 00 00 00 00 92 a3  96 64 00 00 00 00 00 00   ·········d······
0030  00 00 00 00 00 00 00 00  00 00 00 00               ············
```

Answer the following questions:

10. What are the hexadecimal values for the source and destination addresses in the Ethernet frame containing the ARP request message?
11. Give the hexadecimal value for the two-byte Ethernet Frame type field. What upper layer protocol does this correspond to?
12. Download the ARP specification from ftp://ftp.rfc-editor.org/in-notes/std/std37.txt. A readable, detailed discussion of ARP is also at http://www.erg.abdn.ac.uk/users/gorry/course/inet-pages/arp.html.
    a. How many bytes from the very beginning of the Ethernet frame does the ARP opcode field begin?
    b. What is the value of the opcode field within the ARP-payload part of the Ethernet frame in which an ARP request is made?
    c. Does the ARP message contain the IP address of the sender?
    d. Where in the ARP request does the “question” appear – the Ethernet address of the machine whose corresponding IP address is being queried?
13. Now find the ARP reply that was sent in response to the ARP request.
    a. How many bytes from the very beginning of the Ethernet frame does the ARP *opcode* field begin?
    b. What is the value of the *opcode* field within the ARP-payload part of the Ethernet frame in which an ARP response is made?
    c. Where in the ARP message does the “answer” to the earlier ARP request appear – the IP address of the machine having the Ethernet address whose corresponding IP address is being queried?
14. What are the hexadecimal values for the source and destination addresses in the Ethernet frame containing the ARP reply message?
15. Open the ethernet-ethereal-trace-1 trace file in http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.zip. The first and second ARP packets in this trace correspond to an ARP request sent by the computer running Wireshark, and the ARP reply sent to the computer running Wireshark by the computer with the ARP-requested Ethernet address. But there is yet another computer on

this network, as indicated by packet 6 – another ARP request. Why is there no ARP reply (sent in response to the ARP request in packet 6) in the packet trace?

Extra Credit

1. The arp command:

   ```
   arp -s InetAddr EtherAddr
   ```

   allows you to manually add an entry to the ARP cache that resolves the IP address `InetAddr` to the physical address `EtherAddr`. What would happen if, when you manually added an entry, you entered the correct IP address, but the wrong Ethernet address for that remote interface?

2. What is the default amount of time that an entry remains in your ARP cache before being removed. You can determine this empirically (by monitoring the cache contents) or by looking this up in your operation system documentation. Indicate how/where you determined this value.