

In this lab, we'll investigate the Ethernet protocol and the ARP protocol. Before beginning this lab, you'll probably want to read RFC 826 (<ftp://ftp.rfc-editor.org/in-notes/std/std37.txt>) which 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.

1. Capturing and analyzing Ethernet frames

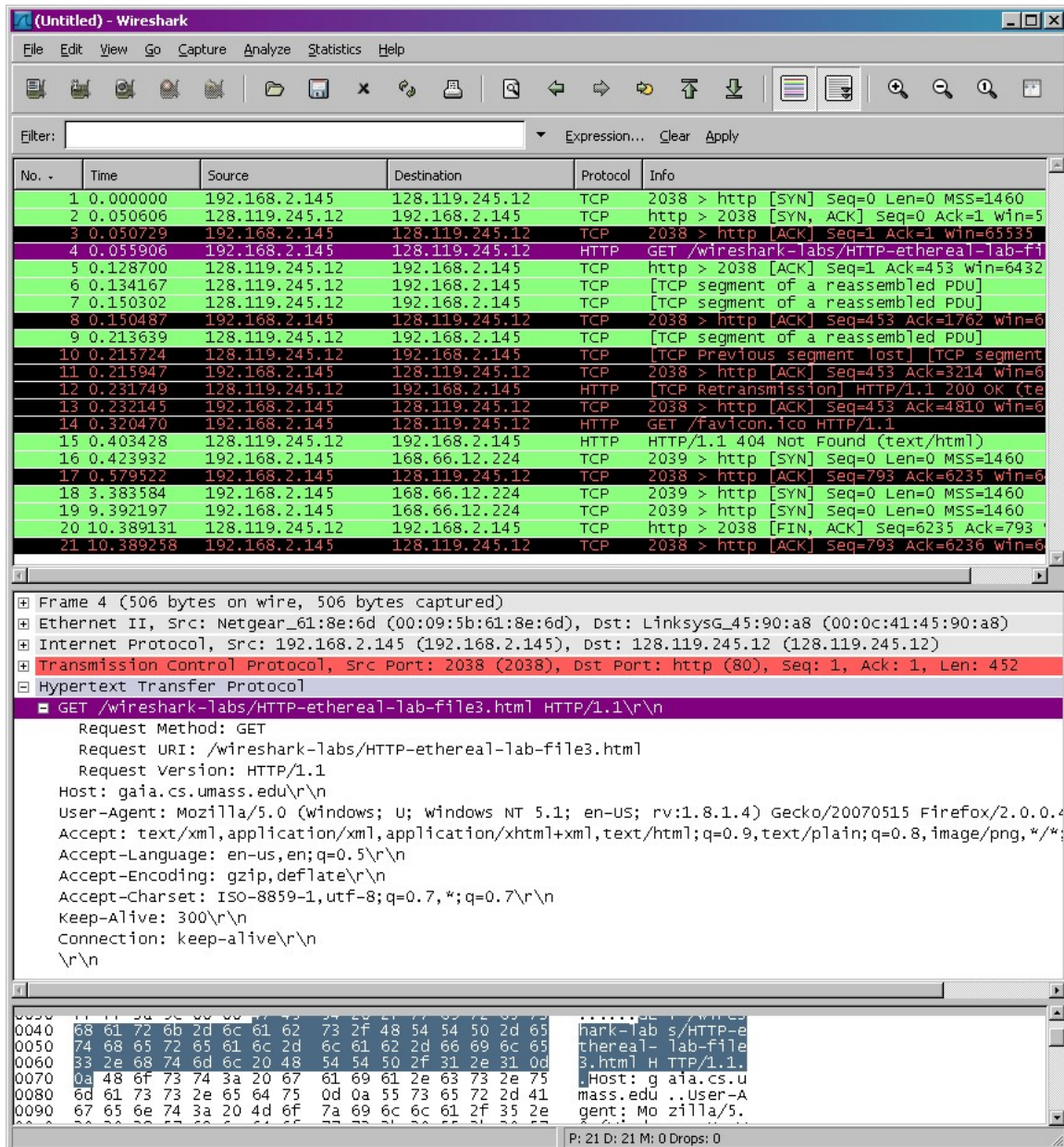
Let's begin by capturing a set of Ethernet frames to study. Do the following¹:

- First, make sure your browser's cache is empty. To do this under Mozilla Firefox V3, select *Tools->Clear Recent History* and check the box for Cache. For Internet Explorer, select *Tools->Internet Options->Delete Files*. Start up the Wireshark packet sniffer
- Enter the following URL into your browser
<http://gaia.cs.umass.edu/wireshark-labs/HTTP-ethereal-lab-file3.html>
Your browser should display the rather lengthy US Bill of Rights.

¹ 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.

Lab-2 Assignment

- 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 4 in the screen shot below contains the HTTP GET message)



The screenshot shows the Wireshark interface with a packet capture of an HTTP transaction. The packet list on the left shows 21 packets. Packet 4 is selected, showing an HTTP GET request to /wireshark-labs/HTTP-ethereal-lab-file3.html. The packet details pane on the right shows the structure of the packet, including Ethernet II, Internet Protocol, Transmission Control Protocol, and Hypertext Transfer Protocol. The packet bytes pane at the bottom shows the raw data of the packet.

No.	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.2.145	128.119.245.12	TCP	2038 > http [SYN] Seq=0 Len=0 MSS=1460
2	0.050606	128.119.245.12	192.168.2.145	TCP	http > 2038 [SYN, ACK] Seq=0 Ack=1 win=5
3	0.050729	192.168.2.145	128.119.245.12	TCP	2038 > http [ACK] Seq=1 Ack=1 win=65535
4	0.055906	192.168.2.145	128.119.245.12	HTTP	GET /wireshark-labs/HTTP-ethereal-lab-file3.html HTTP/1.1
5	0.128700	128.119.245.12	192.168.2.145	TCP	http > 2038 [ACK] Seq=1 Ack=453 win=6432
6	0.134167	128.119.245.12	192.168.2.145	TCP	[TCP segment of a reassembled PDU]
7	0.150302	128.119.245.12	192.168.2.145	TCP	[TCP segment of a reassembled PDU]
8	0.190487	192.168.2.145	128.119.245.12	TCP	2038 > http [ACK] Seq=453 Ack=1762 win=6
9	0.213639	128.119.245.12	192.168.2.145	TCP	[TCP segment of a reassembled PDU]
10	0.215724	128.119.245.12	192.168.2.145	TCP	[TCP Previous segment lost] [TCP segment of a reassembled PDU]
11	0.215947	192.168.2.145	128.119.245.12	TCP	2038 > http [ACK] Seq=453 Ack=3214 win=6
12	0.231749	128.119.245.12	192.168.2.145	HTTP	[TCP Retransmission] HTTP/1.1 200 OK (text/html)
13	0.232145	192.168.2.145	128.119.245.12	TCP	2038 > http [ACK] Seq=453 Ack=4810 win=6
14	0.320470	192.168.2.145	128.119.245.12	HTTP	GET /favicon.ico HTTP/1.1
15	0.403428	128.119.245.12	192.168.2.145	HTTP	HTTP/1.1 404 Not Found (text/html)
16	0.423932	192.168.2.145	168.66.12.224	TCP	2039 > http [SYN] Seq=0 Len=0 MSS=1460
17	0.579522	192.168.2.145	128.119.245.12	TCP	2038 > http [ACK] Seq=793 Ack=6235 win=6
18	3.383584	192.168.2.145	168.66.12.224	TCP	2039 > http [SYN] Seq=0 Len=0 MSS=1460
19	9.392197	192.168.2.145	168.66.12.224	TCP	2039 > http [SYN] Seq=0 Len=0 MSS=1460
20	10.389131	128.119.245.12	192.168.2.145	TCP	http > 2038 [FIN, ACK] Seq=6235 Ack=793
21	10.389258	192.168.2.145	128.119.245.12	TCP	2038 > http [ACK] Seq=793 Ack=6236 win=6

Frame 4 (506 bytes on wire (506 bytes captured) on interface 0: Ethernet II, Src: Netgear_61:8e:6d (00:09:5b:61:8e:6d), Dst: LinksysG_45:90:a8 (00:0c:41:45:90:a8)

Internet Protocol, Src: 192.168.2.145 (192.168.2.145), Dst: 128.119.245.12 (128.119.245.12)

Transmission Control Protocol, Src Port: 2038 (2038), Dst Port: http (80), Seq: 1, Ack: 1, Len: 452

Hypertext Transfer Protocol

GET /wireshark-labs/HTTP-ethereal-lab-file3.html HTTP/1.1\r\n

Request Method: GET

Request URI: /wireshark-labs/HTTP-ethereal-lab-file3.html

Request Version: HTTP/1.1

Host: gaia.cs.umass.edu\r\n

User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.4) Gecko/20070515 Firefox/2.0.0.4\r\n

Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5\r\n

Accept-Language: en-us,en;q=0.5\r\n

Accept-Encoding: gzip,deflate\r\n

Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7\r\n

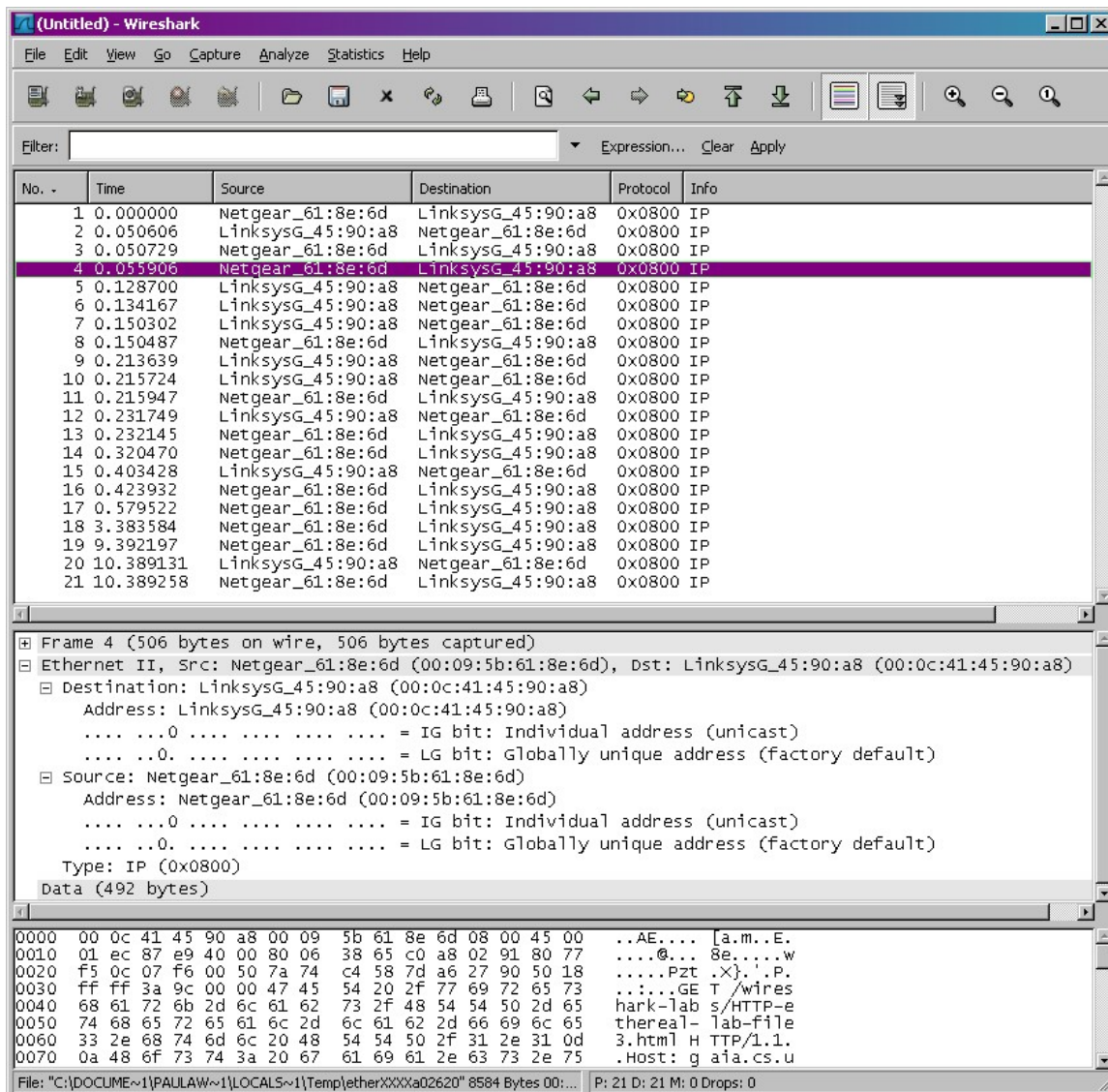
Keep-Alive: 300\r\n

Connection: keep-alive\r\n

\r\n

Lab-2 Assignment

- 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 IP box and select *OK*. You should now see an Wireshark window that looks like:



In order to answer the following questions, you'll need to look into the packet details and packet contents windows (the middle and lower display windows in Wireshark).



Lab-2 Assignment

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. 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 include in your assignment submission, the packet(s) within the trace that you used to answer the question asked. Annotate the printout 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?



Lab-2 Assignment

2. The Address Resolution Protocol

In this section, we'll observe the ARP protocol in action. You are strongly recommended to read RFC 826 (<http://ftp.rfc-editor.org/in-notes/std/std37.txt>) 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).
- **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).

The Windows *arp* command with no arguments will display the contents of the ARP cache on your computer. Run the *arp* command.

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 *arp -d ** will clear your ARP cache. In order to run this command you'll need root privileges. If you don't have root privileges and



Lab-2 Assignment

can't run Wireshark on a Windows machine, you can skip the trace collection part of this lab and just use the trace discussed in the earlier footnote.



Lab-2 Assignment

Observing ARP in action

Do the following²:

- Clear your ARP cache, as described above.
- Next, make sure your browser's cache is empty. To do this under Mozilla Firefox V3, select *Tools->Clear Recent History* and check the box for Cache. For Internet Explorer, select *Tools->Internet Options->Delete Files*.
- 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 IP box and select *OK*. You should now see an Wireshark window that looks like:

² 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).

Lab-2 Assignment

No.	Time	Source	Destination	Protocol	Info
1	0.000000	AmbitMic_a9:3d:68	Broadcast	ARP	who has 192.168.1.1? Tell 192.168.1.105
2	0.001018	LinksysG_da:af:73	AmbitMic_a9:3d:68	ARP	192.168.1.1 is at 00:06:25:da:af:73
3	0.001028	AmbitMic_a9:3d:68	LinksysG_da:af:73	0x0800	IP
4	2.962850	AmbitMic_a9:3d:68	LinksysG_da:af:73	0x0800	IP
5	8.971488	AmbitMic_a9:3d:68	LinksysG_da:af:73	0x0800	IP
6	13.542974	Telebit_73:8d:ce	Broadcast	ARP	who has 192.168.1.117? Tell 192.168.1.104
7	17.444423	AmbitMic_a9:3d:68	LinksysG_da:af:73	0x0800	IP
8	17.465902	LinksysG_da:af:73	AmbitMic_a9:3d:68	0x0800	IP
9	17.465927	AmbitMic_a9:3d:68	LinksysG_da:af:73	0x0800	IP
10	17.466468	AmbitMic_a9:3d:68	LinksysG_da:af:73	0x0800	IP
11	17.494766	LinksysG_da:af:73	AmbitMic_a9:3d:68	0x0800	IP
12	17.498935	LinksysG_da:af:73	AmbitMic_a9:3d:68	0x0800	IP
13	17.500025	LinksysG_da:af:73	AmbitMic_a9:3d:68	0x0800	IP
14	17.500069	AmbitMic_a9:3d:68	LinksysG_da:af:73	0x0800	IP
15	17.527057	LinksysG_da:af:73	AmbitMic_a9:3d:68	0x0800	IP
16	17.527422	LinksysG_da:af:73	AmbitMic_a9:3d:68	0x0800	IP
17	17.527457	AmbitMic_a9:3d:68	LinksysG_da:af:73	0x0800	IP

Frame 1 (42 bytes on wire, 42 bytes captured)

Ethernet II, Src: AmbitMic_a9:3d:68 (00:d0:59:a9:3d:68), Dst: Broadcast (ff:ff:ff:ff:ff:ff)

Address Resolution Protocol (request)

Hardware type: Ethernet (0x0001)
 Protocol type: IP (0x0800)
 Hardware size: 6
 Protocol size: 4
 Opcode: request (0x0001)
 Sender MAC address: AmbitMic_a9:3d:68 (00:d0:59:a9:3d:68)
 Sender IP address: 192.168.1.105 (192.168.1.105)
 Target MAC address: 00:00:00:00:00:00 (00:00:00:00:00:00)
 Target IP address: 192.168.1.1 (192.168.1.1)

0000 ff ff ff ff ff ff 00 d0 59 a9 3d 68 08 06 00 01 Y.=h....
 0010 08 00 06 04 00 01 00 d0 59 a9 3d 68 c0 a8 01 69 Y.=h...i
 0020 00 00 00 00 00 00 c0 a8 01 01

In the example above, the first two frames in the trace contain ARP messages (as does the 6th message). The screen shot above corresponds to the trace referenced in footnote 1.

Answer the following questions:

- What are the hexadecimal values for the source and destination addresses in the Ethernet frame containing the ARP request message?
- Give the hexadecimal value for the two-byte Ethernet Frame type field. What upper layer protocol does this correspond to?
- 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>.



Lab-2 Assignment

- 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

EX-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?

EX-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.



Lab-2 Assignment