

## Wireshark Lab for ECE374

Posted: 03/27/15

Due: 04/03/15

### Ethernet and ARP

#### 1. Capturing and analyzing Ethernet frames

Let's begin by capturing a set of Ethernet frames to study. Do the following<sup>1</sup>:

- First, make sure your browser's cache is empty. To do this under Mozilla Firefox V3, select *Tools->Clear Private Data* 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.
- 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)

---

<sup>1</sup> 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.

The screenshot shows the Wireshark interface with a list of 21 captured packets. The selected packet (No. 4) is a GET request for a file named 'file3.html'. The detailed view shows the following information:

```

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,*/*;\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
  
```

At the bottom of the interface, a hex dump of the packet data is visible, showing the raw bytes of the request.

- 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 and select OK.

You should now see a Wireshark window that looks like:

The screenshot shows the Wireshark interface with a capture of 21 IP packets. Packet 4 is selected. The packet details pane shows the following information:

- Frame 4 (506 bytes on wire, 506 bytes captured)
- Ethernet II, Src: Netgear\_61:8e:6d (00:09:5b:61:8e:6d), Dst: LinksysG\_45:90:a8 (00:0c:41:45:90:a8)
  - Destination: LinksysG\_45:90:a8 (00:0c:41:45:90:a8)
    - Address: LinksysG\_45:90:a8 (00:0c:41:45:90:a8)
      - .... 0 = IG bit: Individual address (unicast)
      - .... 0. = LG bit: Globally unique address (factory default)
    - Source: Netgear\_61:8e:6d (00:09:5b:61:8e:6d)
      - Address: Netgear\_61:8e:6d (00:09:5b:61:8e:6d)
        - .... 0 = IG bit: Individual address (unicast)
        - .... 0. = LG bit: Globally unique address (factory default)
  - Type: IP (0x0800)
  - Data (492 bytes)

The packet bytes pane shows the following hex and ASCII data:

```

0000 00 0c 41 45 90 a8 00 09 5b 61 8e 6d 08 00 45 00  ..AE.... [a.m..E.
0010 01 ec 87 e9 40 00 80 06 38 65 c0 a8 02 91 80 77  ....@... 8e....w
0020 f5 0c 07 f6 00 50 7a 74 c4 58 7d a6 27 90 50 18  ....Pzt.X}.P.
0030 ff ff 3a 9c 00 00 47 45 54 20 2f 77 69 72 65 73  .....GE T/wires
0040 68 61 72 6b 2d 6c 61 62 73 2f 48 54 54 50 2d 65  hark-lab s/HTTP-e
0050 74 68 65 72 65 61 6c 2d 6c 61 62 2d 66 69 6c 65  thereal- lab-file
0060 33 2e 68 74 6d 6c 20 48 54 54 50 2f 31 2e 31 0d  3.html H TTP/1.1.
0070 0a 48 6f 73 74 3a 20 67 61 69 61 2e 63 73 2e 75  .Host: g aia.cs.u
  
```

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). 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 hand in a printout of 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?
3. Give the hexadecimal value for the two-byte Frame type field. What do the bit(s) whose value is 1 mean within the flag field?
4. How many bytes from the very start of the Ethernet frame does the ASCII “G” in “GET” appear in the Ethernet frame?
5. What is the hexadecimal value of the CRC field in this Ethernet frame?

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

6. 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?
7. What is the destination address in the Ethernet frame? Is this the Ethernet address of your computer?
8. Give the hexadecimal value for the two-byte Frame type field. What do the bit(s) whose value is 1 mean within the flag field?
9. 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?
10. What is the hexadecimal value of the CRC field in this Ethernet frame?

## 2. The Address Resolution Protocol

### 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.** The executable for the `arp` command can be in various places. Popular locations are `/sbin/arp` (for linux), `/usr/sbin/arp`, or `/usr/etc/arp` (for some Unix variants).

The *arp* command with no arguments will display the contents of the ARP cache on your computer. (For some OS you have to use *arp -a*). Run the *arp* command.

11. Write down the contents of your computer's ARP cache. What is the meaning of each column value? (Submitting a screenshot instead of writing the results down is absolutely okay.)

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.** 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 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 footnote 1.

Note: For some OS (e.g., MacOS) you have to use *arp -d -a* as root. E.g., *sudo arp -a -d*.

### Observing ARP in action

Do the following<sup>2</sup>:

- 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-ethereal-lab-file3.html>
- 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:

---

<sup>2</sup> 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).

ethernet-ethereal-trace-1 - Wireshark

File Edit View Go Capture Analyze Statistics Help

Filter: Expression... Clear Apply

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  ..... ..
  
```

File: "C:\Documents and Settings\Paula Wing\My Documents\Wireshark\traces - ethereal\... P: 17 D: 17 M: 0

In the example above, the first two frames in the trace contain ARP messages (as does the 6<sup>th</sup> 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 do the bit(s) whose value is 1 mean within the flag field?
- Download the ARP specification from <ftp://ftp.rfc-editor.org/innotes/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>.
  - How many bytes from the very beginning of the Ethernet frame does the ARP *opcode* field begin?
  - What is the value of the *opcode* field within the ARP-payload part of the Ethernet frame in which an ARP request is made?
  - 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?
15. 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?
  16. What are the hexadecimal values for the source and destination addresses in the Ethernet frame containing the ARP reply message?
  17. 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?