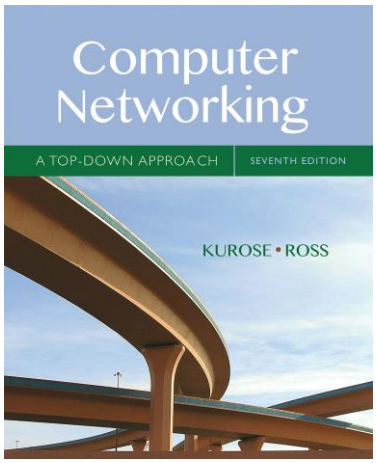
Name: _____Timur Guner_____

Wireshark Lab: Ethernet and ARP v7.0

Supplement to *Computer Networking: A Top-Down Approach, 7th ed.*, J.F. Kurose and K.W. Ross

"Tell me and I forget. Show me and I remember. Involve me and I understand." Chinese proverb

© 2005-2016 J.F Kurose and K.W. Ross, All Rights Reserved



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.

1. Capturing and analyzing Ethernet frames

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

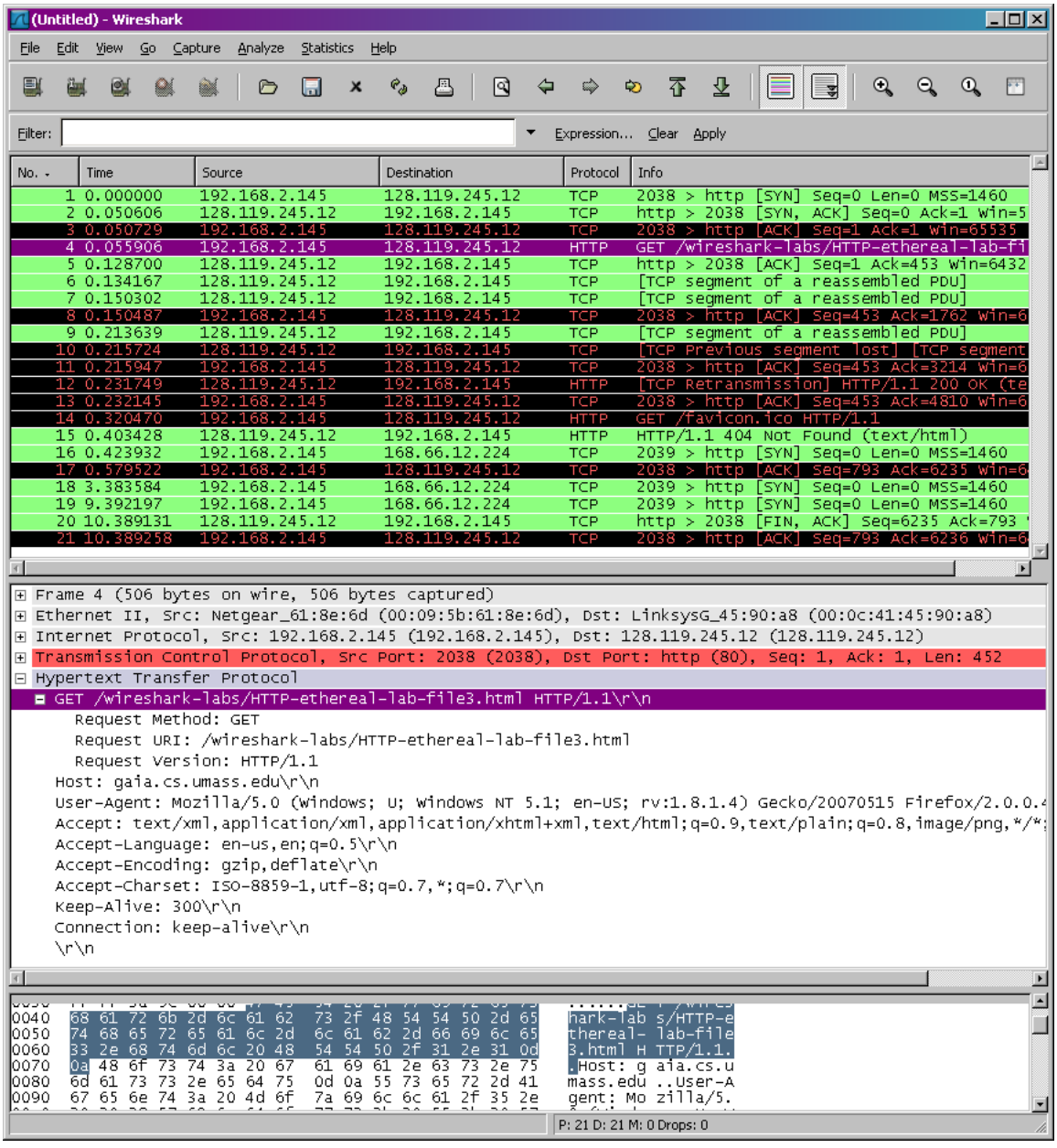
- First, make sure your browser's cache is empty. To do this under Mozilla Firefox V3, select *Tools->Clear Recent History* and check the box for Cache. For Internet Explorer, select *Tools->Internet Options->Delete Files.* Start up the Wireshark packet sniffer
- Enter the following URL into your browser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

---

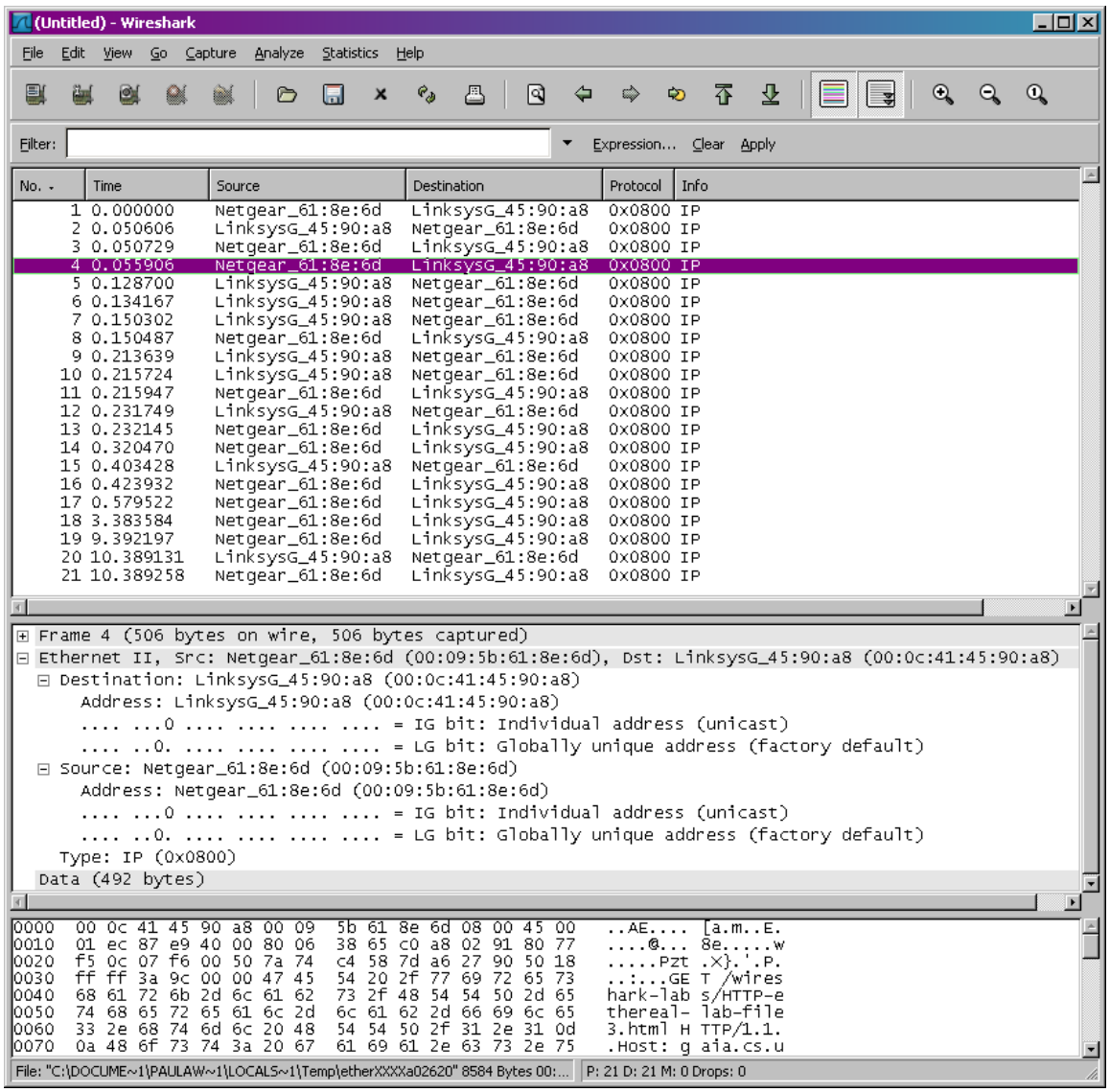
[1] References to figures and sections are for the 7th edition of our text, *Computer Networks, A Top-down Approach, 7th ed.*, J.F. Kurose and K.W. Ross, Addison-Wesley/Pearson, 2016.

[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.

- 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)



- 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:

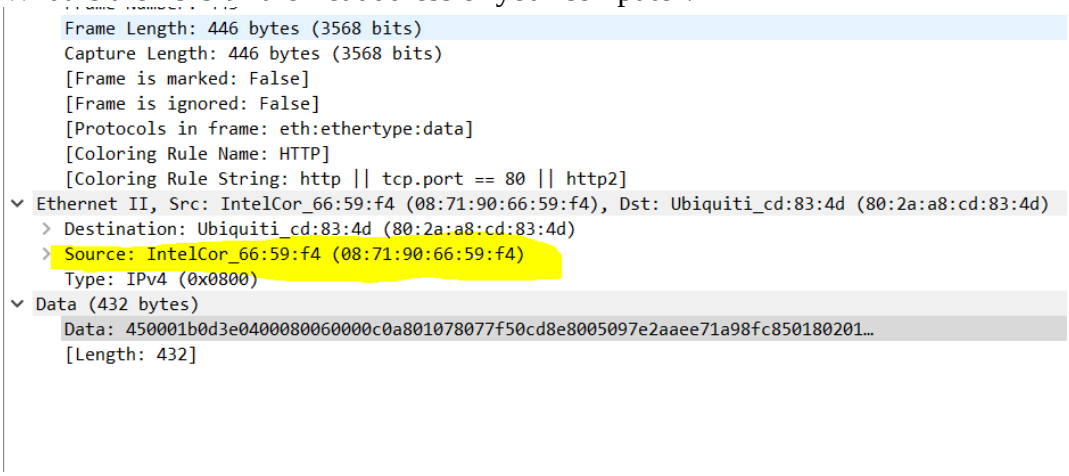


In order to answer the following questions, you'll need to look into the packet details and packet contents windows (the middle and lower display windows in Wireshark).

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.

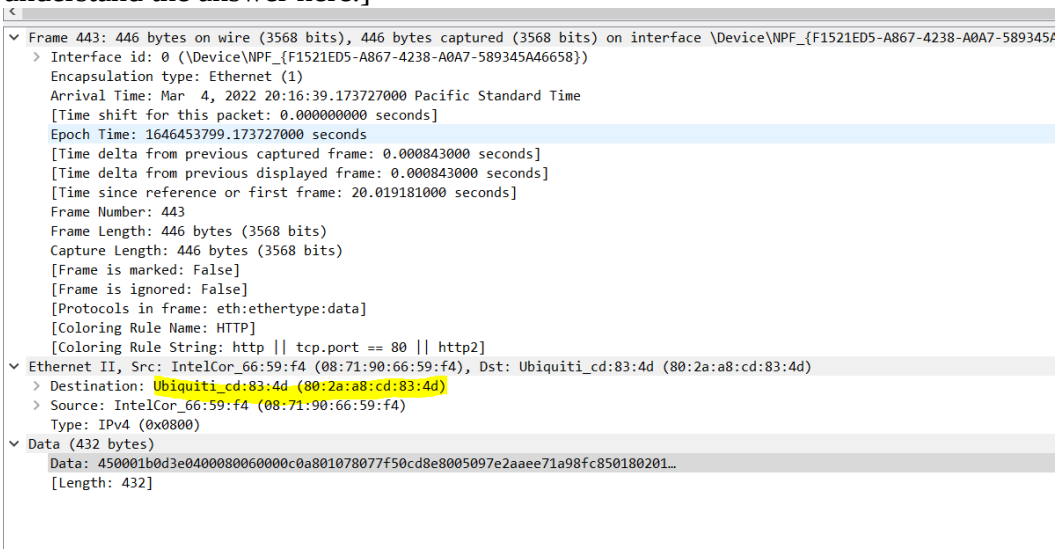
Answer the following questions, based on the contents of the Ethernet frame containing the HTTP GET message. Whenever possible, when answering a question you should include a screenshot of the packet(s) within the trace that you used to answer the question asked. Make sure to include in the screenshot ALL and ONLY the minimum amount of packet detail that you need to answer the question.

1. What is the 48-bit Ethernet address of your computer?



08:71:90:66:59:f4

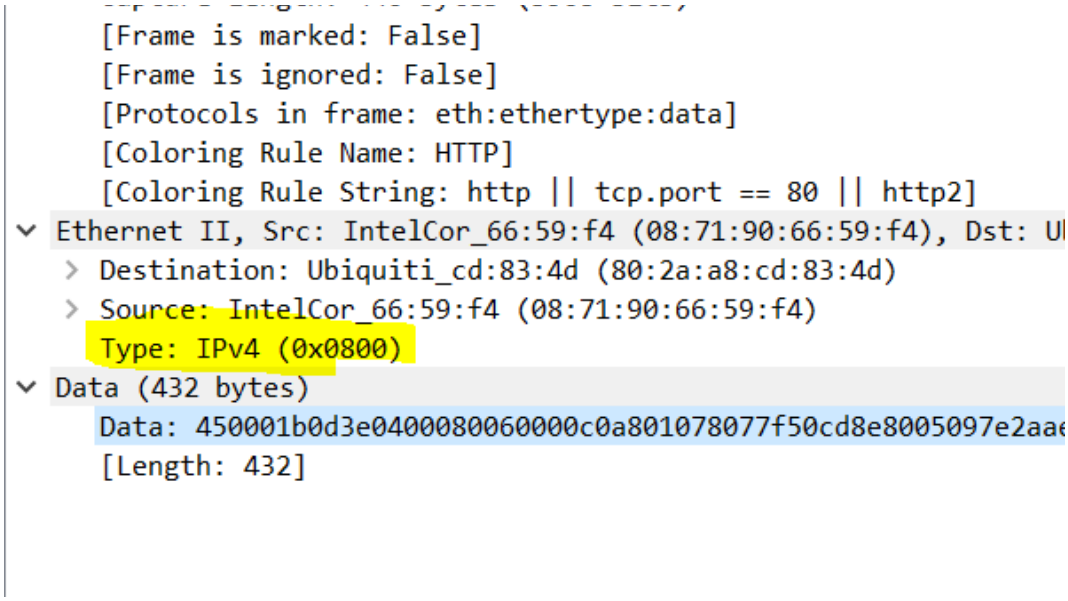
2. What is the 48-bit destination address in the Ethernet frame? Is this the Ethernet address of gaia.cs.umass.edu? (Hint: the answer is *no*). What device has this as its Ethernet address? [Note: this is an important question, and one that students sometimes get wrong. Re-read pages 468-469 in the text and make sure you understand the answer here.]



80:2a:a8:cd:83:4d

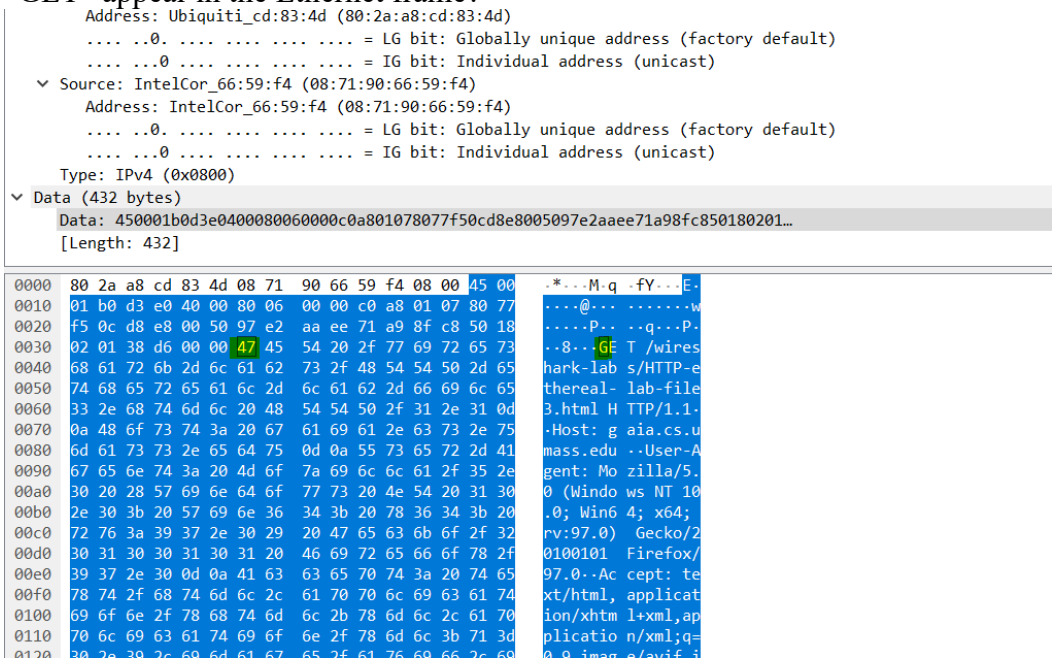
It is the ethernet address of my router Ubiquiti

3. Give the hexadecimal value for the two-byte Frame type field. What upper layer protocol does this correspond to?



0x0800 and is IPv4

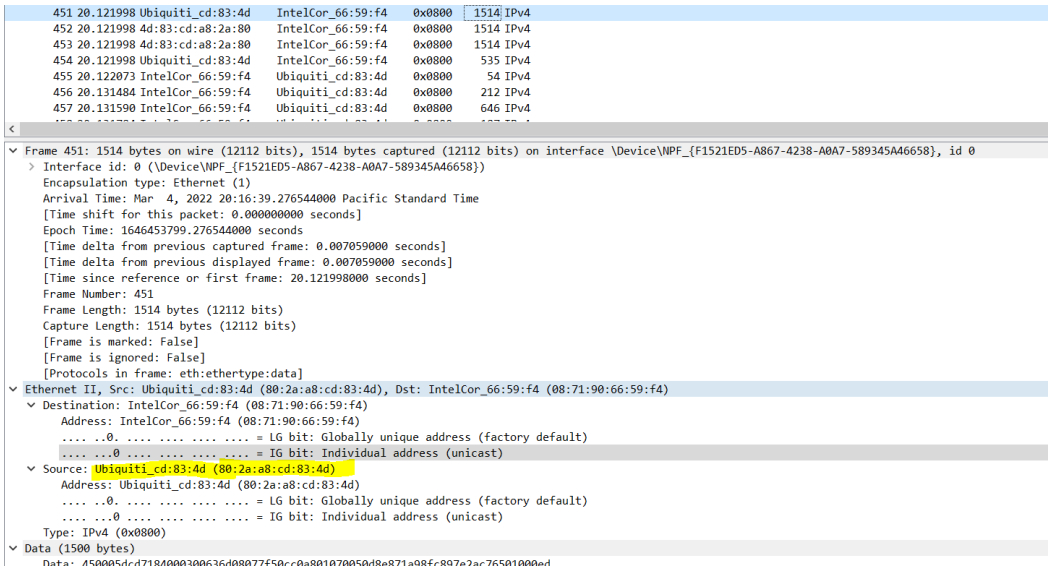
4. How many bytes from the very start of the Ethernet frame does the ASCII "G" in "GET" appear in the Ethernet frame?



Based on counting from the beginning of starting 80 is 0th Byte then 47 (G) is the 54th Byte

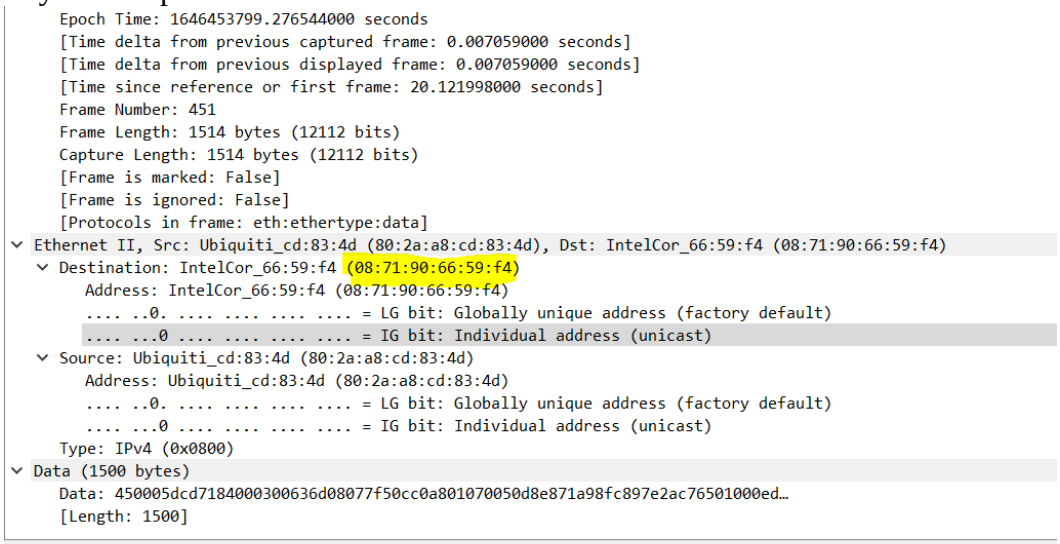
Next, answer the following questions, based on the contents of the Ethernet frame containing the first byte of the HTTP response message.

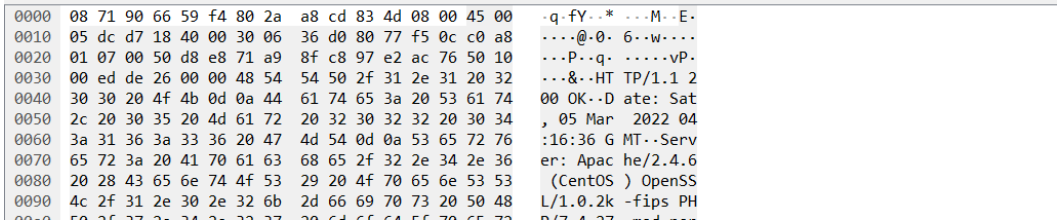
5. What is the value of the Ethernet source address? Is this the address of your computer, or of gaia.cs.umass.edu (Hint: the answer is *no*). What device has this as its Ethernet address?



80:2a:a8:cd:83:4d and this is my router Ubiquiti

6. What is the destination address in the Ethernet frame? Is this the Ethernet address of your computer?





08:71:90:66:59:f4 which is my computer

7. Give the hexadecimal value for the two-byte Frame type field. What upper layer protocol does this correspond to?

```
[Time delta from previous displayed frame: 0.007059000 seconds]
[Time since reference or first frame: 20.121998000 seconds]
Frame Number: 451
Frame Length: 1514 bytes (12112 bits)
Capture Length: 1514 bytes (12112 bits)
[Frame is marked: False]
[Frame is ignored: False]
[Protocols in frame: eth:ethertype:data]
Ethernet II, Src: Ubiquiti_cd:83:4d (80:2a:a8:cd:83:4d), Dst: IntelCor_66:59:f4 (08:71:90:66:59:
  Destination: IntelCor_66:59:f4 (08:71:90:66:59:f4)
    Address: IntelCor_66:59:f4 (08:71:90:66:59:f4)
    .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
    .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
  Source: Ubiquiti_cd:83:4d (80:2a:a8:cd:83:4d)
    Address: Ubiquiti_cd:83:4d (80:2a:a8:cd:83:4d)
    .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
    .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
  Type: IPv4 (0x0800)
Data (1500 bytes)
  Data: 450005dcd7184000300636d08077f50cc0a801070050d8e871a98fc897e2ac76501000ed…
  [Length: 1500]

0000  08 71 90 66 59 f4 80 2a  a8 cd 83 4d 08 00 45 00   ·q·fY··* ···M··E·
```

0x0800 and it is IPv4

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?

```
    .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
  Type: IPv4 (0x0800)
Data (1500 bytes)
  Data: 450005dcd7184000300636d08077f50cc0a801070050d8e871a98fc897e2ac76501000ed…
  [Length: 1500]

0000  08 71 90 66 59 f4 80 2a  a8 cd 83 4d 08 00 45 00   ·q·fY··* ···M··E·
0010  05 dc d7 18 40 00 30 06  36 d0 80 77 f5 0c c0 a8   ····@·0· 6··w····
0020  01 07 00 50 d8 e8 71 a9  8f c8 97 e2 ac 76 50 10   ···P··q· ·····vP·
0030  00 ed de 26 00 00 48 54  54 50 2f 31 2e 31 20 32   ···&··HT TP/1.1 2
0040  30 30 20 4f 4b 0d 0a 44  61 74 65 3a 20 53 61 74   00 OK··D ate: Sat
0050  2c 20 30 35 20 4d 61 72  20 32 30 32 32 20 30 34   , 05 Mar  2022 04
0060  3a 31 36 3a 33 36 20 47  4d 54 0d 0a 53 65 72 76   :16:36 G MT··Serv
0070  65 72 3a 20 41 70 61 63  68 65 2f 32 2e 34 2e 36   er: Apac he/2.4.6
0080  20 28 43 65 6e 74 4f 53  29 20 4f 70 65 6e 53 53    (CentOS ) OpenSS
0090  4c 2f 31 2e 30 2e 32 6b  2d 66 69 70 73 20 50 48   L/1.0.2k -fips PH
00a0  50 2f 37 2e 34 2e 32 37  20 6d 6f 64 5f 70 65 72   P/7.4.27  mod_per
00b0  6c 2f 32 2e 30 2e 31 31  20 50 65 72 6c 2f 76 35   l/2.0.11  Perl/v5
00c0  2e 31 36 2e 33 0d 0a 4c  61 73 74 2d 4d 6f 64 69   .16.3··L ast-Modi
```

Doing the same method as question 4, we count from beginning and 4f (O) is 67th Byte

2. 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

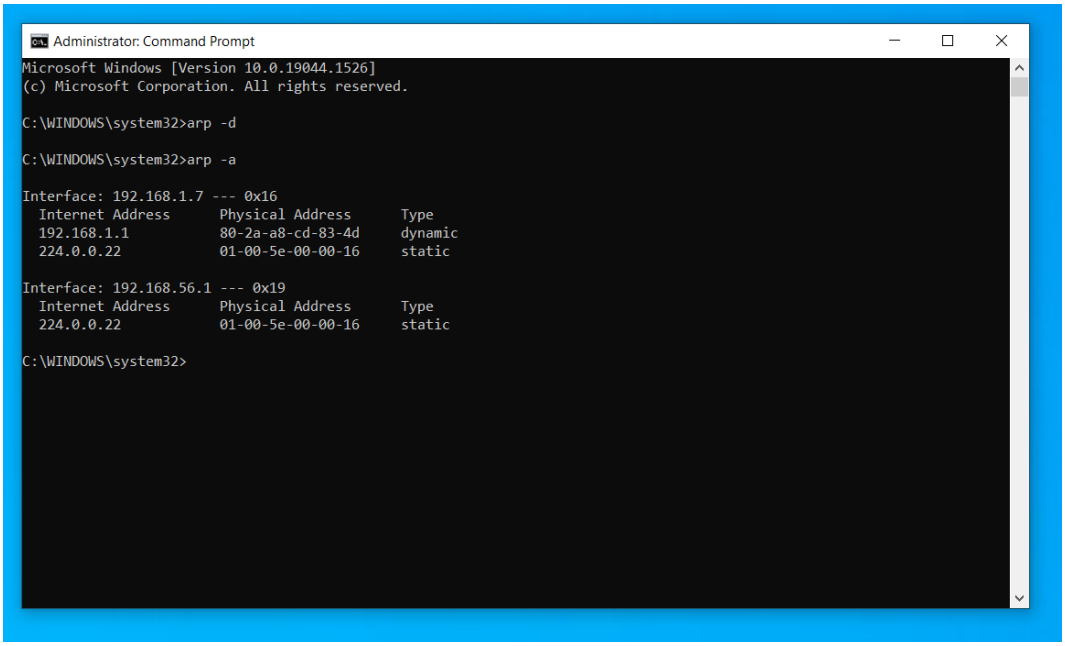
Recall that the ARP protocol typically maintains a cache of IP-to-Ethernet address translation pairs on your comnputer The *arp* command (in both MSDOS and Linux/Unix) is used to view and manipulate the contents of this cache. Since the *arp* command and the ARP protocol have the same name, it's understandably 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 -a* command will display the contents of the ARP cache on your computer. Run the *arp -a* command.

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



Internet Address: IP address
Physical Address: the MAC address
Type: The protocol type

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.

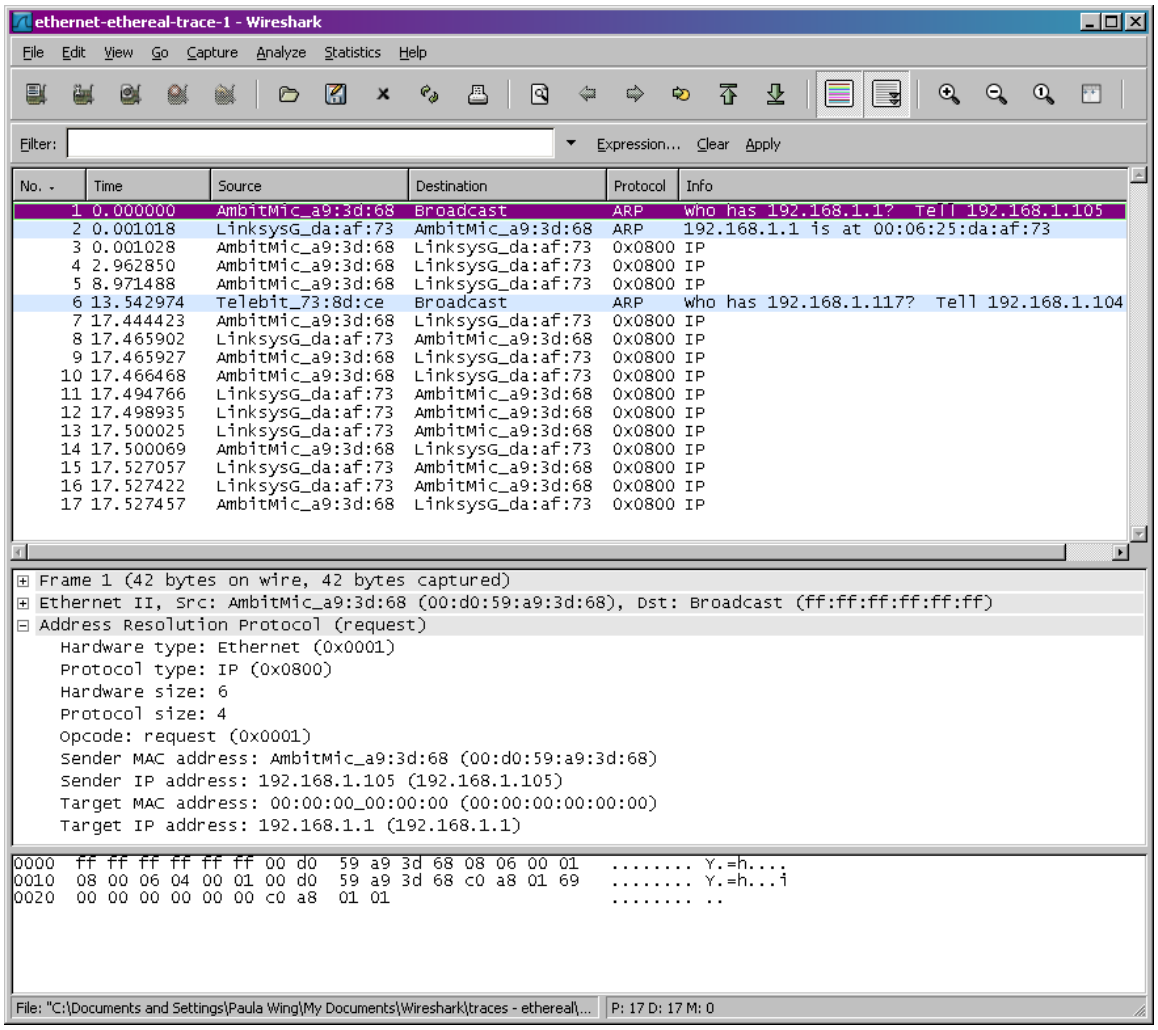
- **MS-DOS.** The MS-DOS *arp –d* * command will clear your ARP cache. The *–d* flag indicates a deletion operation, and the * is the wildcard that says to delete all table entries.
- **Linux/Unix/MacOS.** The *arp –d -a* will clear your ARP cache. In order to run this command you'll need root privileges. 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 the earlier footnote.

Observing ARP in action

Do the following[3]:

- 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:

[3] 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).

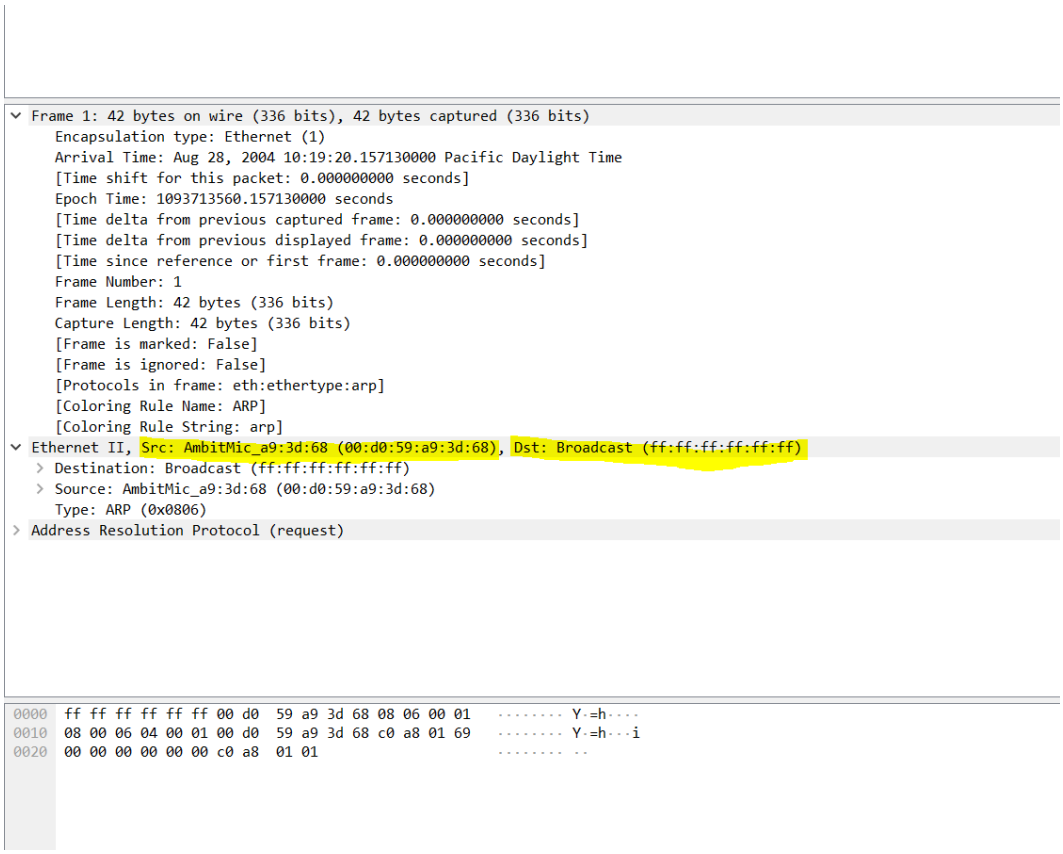


In the example above, the first two frames in the trace contain ARP messages (as does the 6th message). The screen shot above corresponds to the trace referenced in footnote 1.

Answer the following questions:

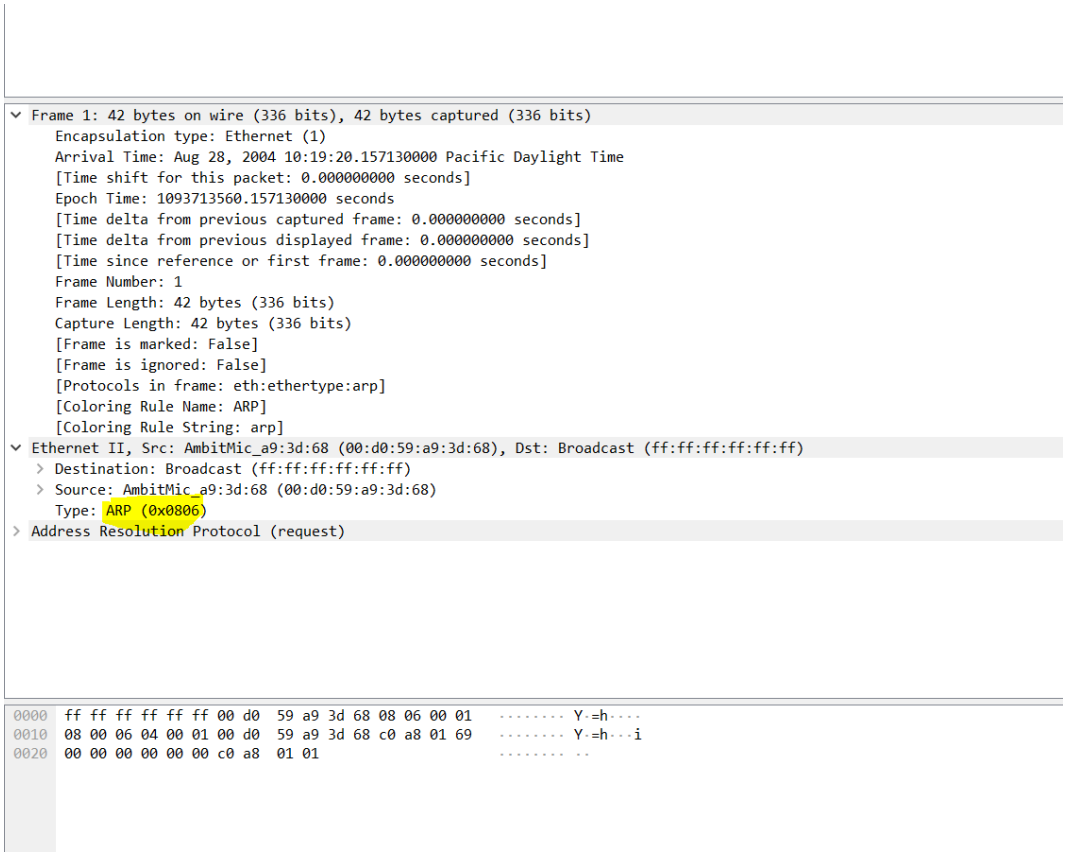
Note that from here on I am using the provided source file from gaia

10. What are the hexadecimal values for the source and destination addresses in the Ethernet frame containing the ARP request message?



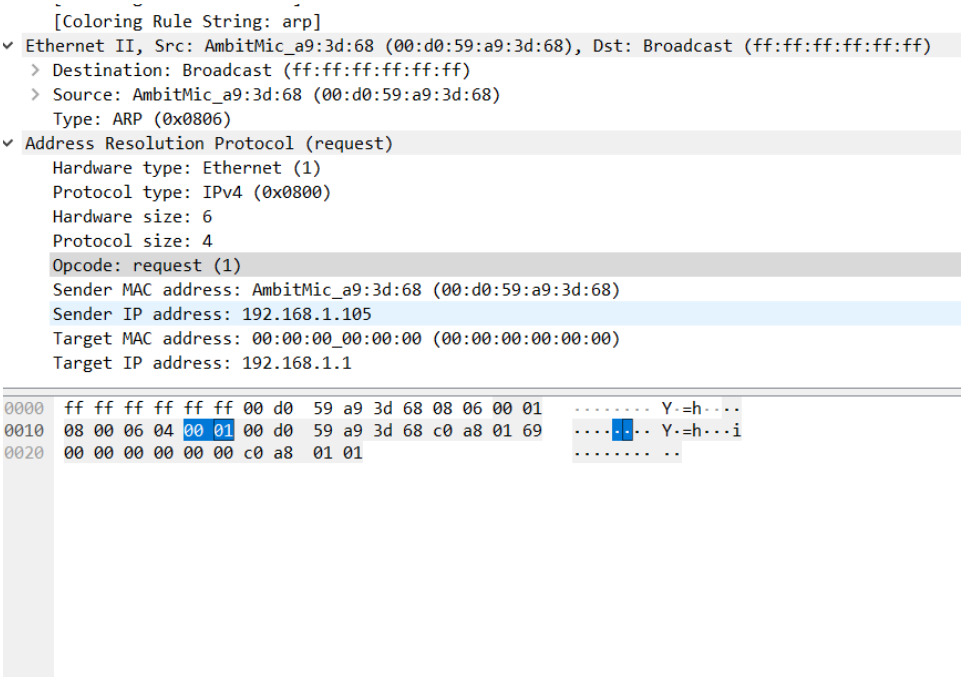
Source: 00:d0:59:a9:3d:68
Destination: ff:ff:ff:ff:ff:ff

11. Give the hexadecimal value for the two-byte Ethernet Frame type field. What upper layer protocol does this correspond to?



Hex is 0x0806 which is ARP

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?

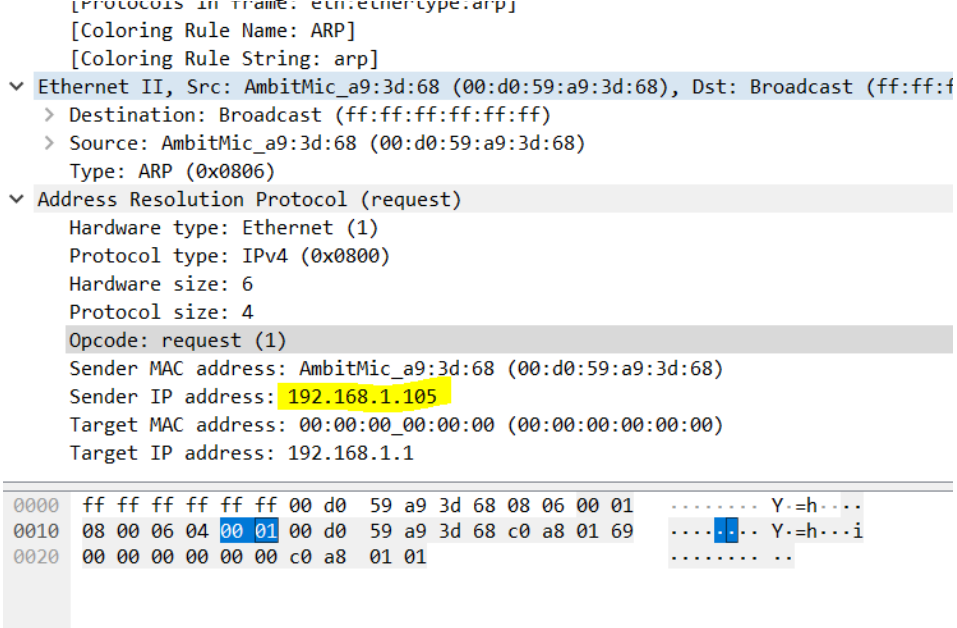


It starts at byte 20 where its 00

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?

In the screenshot in 12a, it is 0x0001

c) Does the ARP message contain the IP address of the sender?



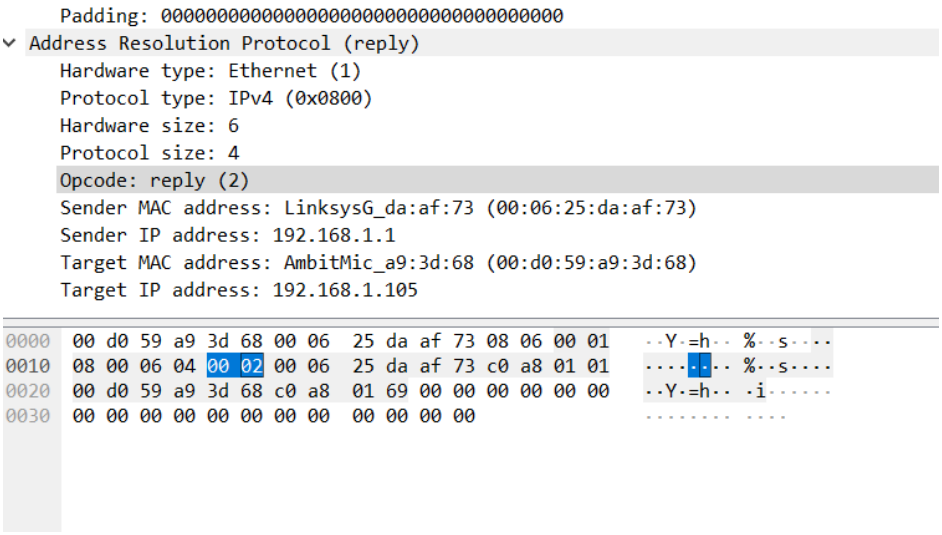
The IP address is 192.168.1.105

d) Where in the ARP request does the "question" appear – the Ethernet address of the machine whose corresponding IP address is being queried?

In the screenshot in 12c, target MAC address field is 00:00:00:00:00:00. Once the MAC address is resolved, this would be

populated with the corresponding complete MAC address of the server or router.

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?

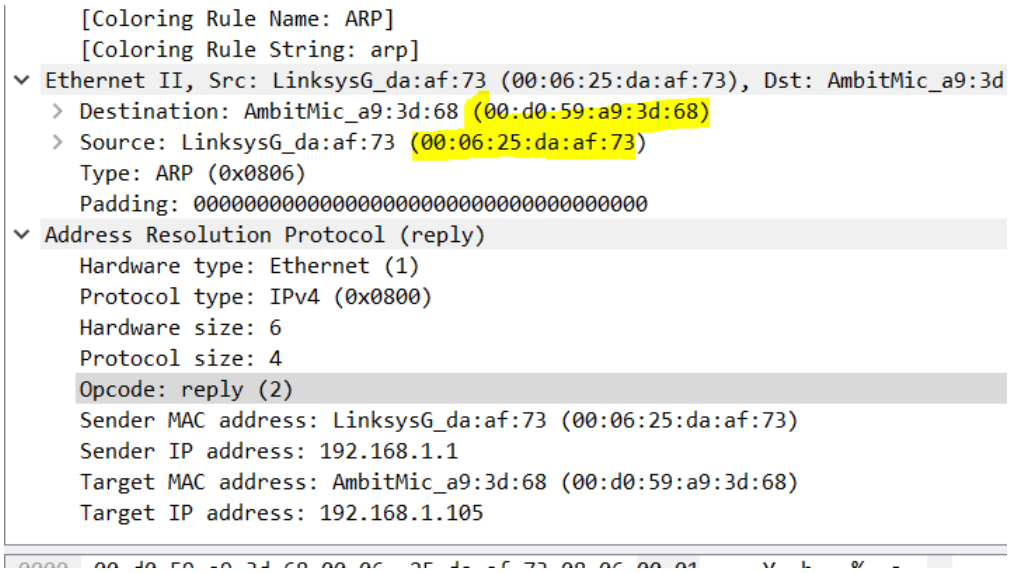


    It is also the 20th byte like in question 12a

    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?
    In the screenshot above, it shows it is 0x0002

    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?
    The previously "blank" target MAC Address field, which now would be the Sender MAC Address since this is a reply to the broadcast request, which is the server's MAC address, and is 00:06:25:da:af:73. This is shown in screenshot in 13a.

14. What are the hexadecimal values for the source and destination addresses in the Ethernet frame containing the ARP reply message?



Source: 00:06:25:da:af:73
Destination: 00:d0:59:a9:3d:68

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?
These IP addresses are in the same subnet that the router has mapped already in the ARP table. It does not need to be rediscovered and chronicled.

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.