Lab Instructions

- Please leave your bags on the Iron shelf near the SP16 entrance.
- Login to the Ubuntu OS on your machine. The login credentials are as follows:
  - Username: user
  - Password: 123456
- Mark your attendance in the attendance sheet before leaving the lab.
- Handle the lab resources with utmost care.
- Please go through the following exercises in today's lab.
- It is recommended that you complete all the following exercises during the lab slot itself.
- If you face any difficulties, please feel free to seek help online or from your peers or TAs.
- After finishing all exercises, please carry your solutions with you (via email/pen drive) for future reference, and delete the files from the desktop.

Introduction

In this lab, we'll investigate the Ethernet protocol and the ARP protocol.

Part-1: Capturing and analyzing Ethernet frames

Let's begin by capturing a set of Ethernet frames to study. To do this, of course, you'll need access to a wired Ethernet connection for your system.
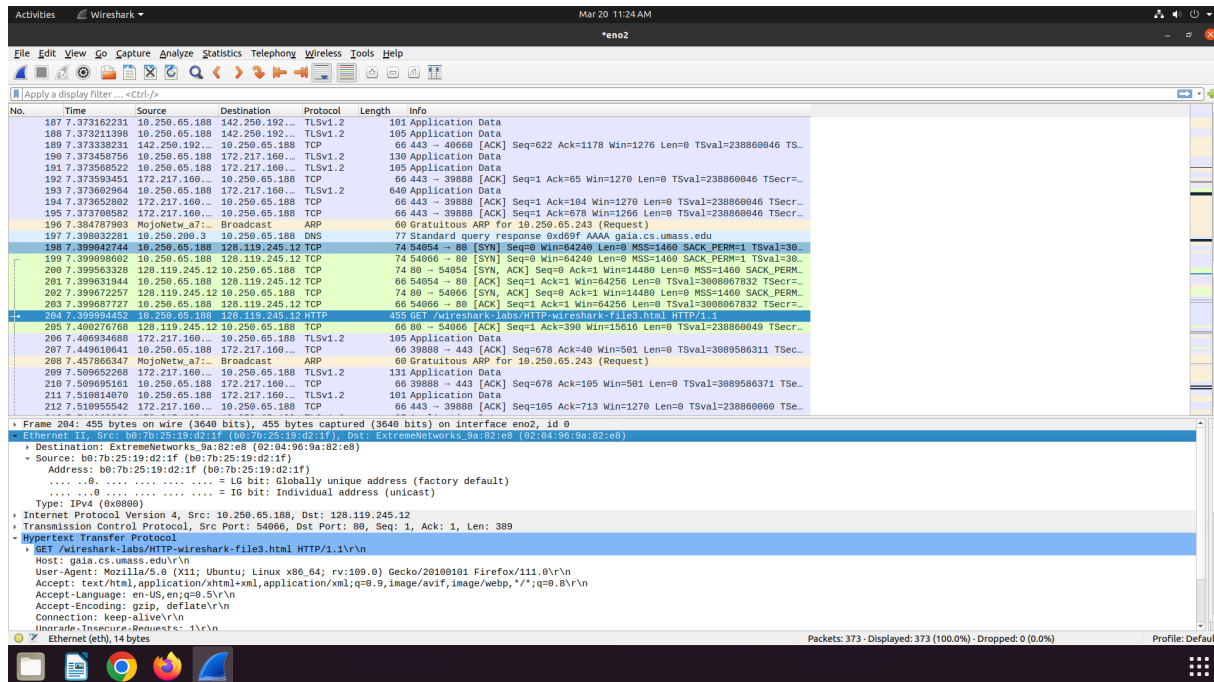
Do the following:

- First, make sure your browser's cache of previously downloaded documents is empty.
- Start up Wireshark and enter the following URL into your browser: http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file3.html. Your browser should display the rather lengthy US Bill of Rights.
- Stop Wireshark packet capture.

First, find the packet number (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 204 in the screen shot below contains the HTTP GET message)

Let's start by looking at 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. Your display should look similar to that shown in the Figure below.

*eno2

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter … <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
187	7.373162231	10.250.65.188	142.250.192…	TLSv1.2	101	Application Data
188	7.373211398	10.250.65.188	142.250.192…	TLSv1.2	105	Application Data
189	7.373338231	142.250.192…	10.250.65.188	TCP	66	443 → 40660 [ACK] Seq=622 Ack=1178 Win=1276 Len=0 TSval=238860046 TS…
190	7.373458756	10.250.65.188	172.217.160…	TLSv1.2	130	Application Data
191	7.373568522	10.250.65.188	172.217.160…	TLSv1.2	105	Application Data
192	7.373593451	172.217.160…	10.250.65.188	TCP	66	443 → 39888 [ACK] Seq=1 Ack=65 Win=1270 Len=0 TSval=238860046 TSecr=…
193	7.373602964	10.250.65.188	172.217.160…	TLSv1.2	640	Application Data
194	7.373652802	172.217.160…	10.250.65.188	TCP	66	443 → 39888 [ACK] Seq=1 Ack=104 Win=1270 Len=0 TSval=238860046 TSecr…
195	7.373708582	172.217.160…	10.250.65.188	TCP	66	443 → 39888 [ACK] Seq=1 Ack=678 Win=1266 Len=0 TSval=238860046 TSecr…
196	7.384787903	MojoNetw_a7:…	Broadcast	ARP	60	Gratuitous ARP for 10.250.65.243 (Request)
197	7.398032281	10.250.200.3	10.250.65.188	DNS	77	Standard query response 0xd69f AAAA gaia.cs.umass.edu
198	7.399042744	10.250.65.188	128.119.245.12	TCP	74	54054 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=238…
199	7.399098602	10.250.65.188	128.119.245.12	TCP	74	54066 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=30…
200	7.399563328	128.119.245.12	10.250.65.188	TCP	74	80 → 54054 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460 SACK_PERM…
201	7.399631944	10.250.65.188	128.119.245.12	TCP	66	54054 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3008067832 TSecr=…
202	7.399672257	128.119.245.12	10.250.65.188	TCP	74	80 → 54066 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460 SACK_PERM…
203	7.399687727	10.250.65.188	128.119.245.12	TCP	66	54066 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3008067832 TSecr=…
204	7.399994452	10.250.65.188	128.119.245.12	HTTP	455	GET /wireshark-labs/HTTP-wireshark-file3.html HTTP/1.1
205	7.400276768	128.119.245.12	10.250.65.188	TCP	66	80 → 54066 [ACK] Seq=1 Ack=390 Win=15616 Len=0 TSval=238860049 TSecr…
206	7.406934688	172.217.160…	10.250.65.188	TLSv1.2	105	Application Data
207	7.449610641	10.250.65.188	172.217.160…	TCP	66	39888 → 443 [ACK] Seq=678 Ack=40 Win=501 Len=0 TSval=3089586311 TSec…
208	7.457866347	MojoNetw_a7:…	Broadcast	ARP	60	Gratuitous ARP for 10.250.65.243 (Request)
209	7.509652268	172.217.160…	10.250.65.188	TLSv1.2	131	Application Data
210	7.509695161	10.250.65.188	172.217.160…	TCP	66	39888 → 443 [ACK] Seq=678 Ack=105 Win=501 Len=0 TSval=3089586371 TSe…
211	7.510814070	10.250.65.188	172.217.160…	TLSv1.2	101	Application Data
212	7.510955542	172.217.160…	10.250.65.188	TCP	66	443 → 39888 [ACK] Seq=1 Ack=713 Win=1270 Len=0 TSval=238860060 TSe…

```
▸ Frame 204: 455 bytes on wire (3640 bits), 455 bytes captured (3640 bits) on interface eno2, id 0
▾ Ethernet II, Src: b0:7b:25:19:d2:1f (b0:7b:25:19:d2:1f), Dst: ExtremeNetworks_9a:82:e8 (02:04:96:9a:82:e8)
   ▸ Destination: ExtremeNetworks_9a:82:e8 (02:04:96:9a:82:e8)
   ▾ Source: b0:7b:25:19:d2:1f (b0:7b:25:19:d2:1f)
      Address: b0:7b:25:19:d2:1f (b0:7b:25:19:d2:1f)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
   Type: IPv4 (0x0800)
▸ Internet Protocol Version 4, Src: 10.250.65.188, Dst: 128.119.245.12
▸ Transmission Control Protocol, Src Port: 54066, Dst Port: 80, Seq: 1, Ack: 1, Len: 389
▾ Hypertext Transfer Protocol
   ▸ GET /wireshark-labs/HTTP-wireshark-file3.html HTTP/1.1\r\n
     Host: gaia.cs.umass.edu\r\n
     User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/111.0\r\n
     Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8\r\n
     Accept-Language: en-US,en;q=0.5\r\n
     Accept-Encoding: gzip, deflate\r\n
     Connection: keep-alive\r\n
     Upgrade-Insecure-Requests: 1\r\n
```

○ ⦿ Ethernet (eth), 14 bytes   Packets: 373 · Displayed: 373 (100.0%) · Dropped: 0 (0.0%)   Profile: Default

---

Answer the questions below:

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. What is the hexadecimal value for the two-byte Frame type field in the Ethernet frame carrying the HTTP GET request? 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? Do not count any preamble bits in your count, i.e., assume that the Ethernet frame begins with the Ethernet frame's destination address.
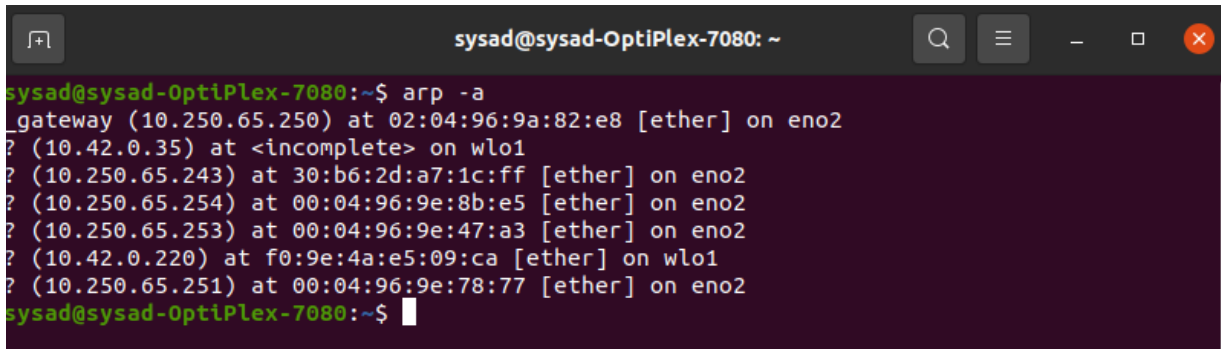
Next, answer the following questions, based on the contents of the Ethernet frame containing the first byte of the HTTP response message.

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? Do not count any preamble

bits in your count, i.e., assume that the Ethernet frame begins with the Ethernet frame's destination address.

9. How many Ethernet frames (each containing an IP datagram, each containing a TCP segment) carry data that is part of the complete HTTP "OK 200 ..." reply message?

Part-2: The Address Resolution Protocol

In this section, we'll observe the ARP protocol in action.

Recall that the ARP protocol typically maintains a cache of IP-to-Ethernet address translation pairs on your computer. The *arp* command (in both DOS, MacOS and Linux) 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 ARP message transmission and receipt.

Let's take a look at the contents of the ARP cache on your computer. In DOS, MacOS, and Linux, the "arp -a" command will display the contents of the ARP cache on your computer. So at the terminal, type "arp -a". The results of entering this command is shown in the Figure below.

1. How many entries are stored in your ARP cache?
2. What is contained in each displayed entry of the ARP cache?

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. The "*arp –d -a*" command will clear your ARP cache using the command line. In order to run this command on a Mac or Linux machine you'll need root privileges or use *sudo*. 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 use the trace file *ethernet-wireshark-trace1* discussed earlier.

Observing ARP in action

Do the following:

- Clear your ARP cache, as described above and make sure your browser's cache is cleared of previously downloaded documents.
- 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 let's just look at ARP packets. Your display should look similar to that shown in the Figure below (note we have entered "arp" into the display filter window at the top of the Wireshark screen).
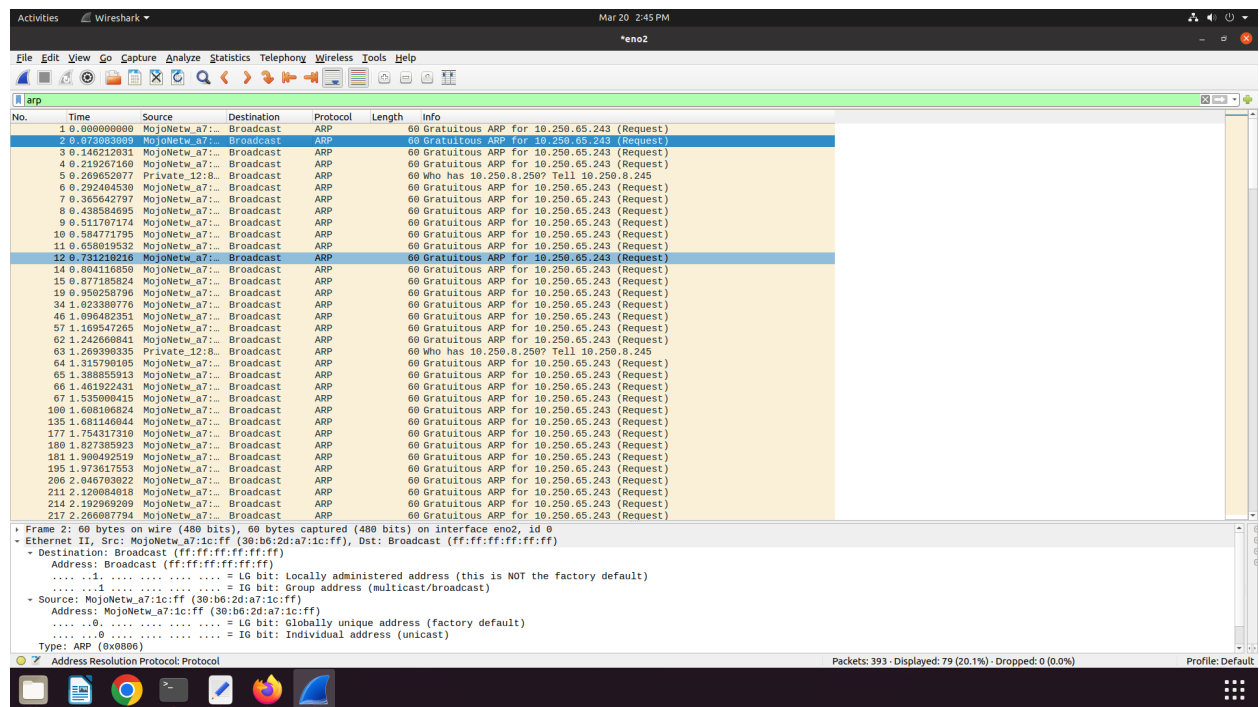
3. What is the hexadecimal value of the source address in the Ethernet frame containing the ARP request message sent out by your computer?
4. What is the hexadecimal value of the destination addresses in the Ethernet frame containing the ARP request message sent out by your computer? And what device(if any) corresponds to that address (e.g,, client, server, router, switch or otherwise...)?
5. What is the hexadecimal value for the two-byte Ethernet Frame *type* field? What upper layer protocol does this correspond to?

Now let's dig even a bit deeper into the ARP messages themselves. To answer this question, you will need to dig into ARP. The original RFC (https://datatracker.ietf.org/doc/html/rfc826) that defines ARP is a little hard to read. The Wikipedia entry for ARP is pretty good: https://en.wikipedia.org/wiki/Address_Resolution_Protocol

Answer the following question about the ARP request message sent by your computer.

6. How many bytes from the very beginning of the Ethernet frame does the ARP *opcode* field begin?
7. What is the value of the *opcode* field within the ARP request message sent by your computer?
8. Does the ARP request message contain the IP address of the sender? If the answer is yes, what is that value?
9. What is the IP address of the device whose corresponding Ethernet address is being requested in the ARP request message sent by your computer?

Now find the ARP reply message that was sent in response to the ARP request from your computer.

10. What is the value of the *opcode* field within the ARP reply message received by your computer?
11. *Finally (!),* let's look at the **answer** to the ARP request message! What is the Ethernet address corresponding to the IP address that was specified in the ARP request message sent by your computer?
12. We've looked at the ARP request message sent by your computer running Wireshark, and the ARP reply message sent in response. But there are other devices in this network that are also sending ARP request messages that you can find in the trace. Why are there no ARP replies in your trace that are sent in response to these other ARP request messages?

Submission Details
- Write your answers in a single doc/tex file, and submit its PDF named after your IIT Dharwad roll number, which contains all answers (with screenshots, if necessary).