

ELEC3506/9506 Communication Networks – Lab 01

Instructions

This lab instruction comprises two phases, where each contains multiple questions that students need to answer them during the lab session to comprehend the main goal of the phase.

The Final Lab Report you need to submit to the Canvas should be within maximum of 7 pages, stating your overall learning experience from the lab experiments and the main results you have obtained. In the first phase, students get familiar with the basics of the Wireshark, to capture the data packets and analyze the packets' transmissions under different protocols. In the second phase, the role and importance of two well-known protocols, namely Ethernet and Address Resolution Protocol (ARP), are investigated.

A template has been provided for each Lab Report that you should follow it to prepare the Final Lab Report. You can find the template in the Canvas. Before starting the lab, students should follow the instructions provided for the lab session. The instructions are summarized as follows:

- You need to follow up the lab instruction, provided in the Canvas, regarding forming groups and submitting the Final Lab Reports.
- Complete the lab according to the instructions in the following pages.
- You should observe lab rules and behave properly in the lab.
- Relevant materials: HTTP - page 861; Ethernet - page 395; ARP - page 612.

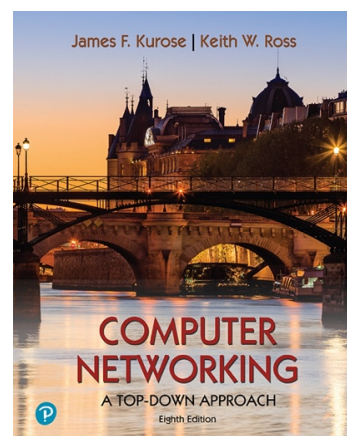
Notation: Students can find the corresponding concepts in the main reference provided for this course:

Phase 1: Getting Started

Supplement to *Computer Networking: A Top-Down Approach*, 8th 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

© 2005-2020, J.F Kurose and K.W. Ross, All Rights Reserved



One’s understanding of network protocols can often be greatly deepened by “seeing protocols in action” and by “playing around with protocols” – observing the sequence of messages exchanged between two protocol entities, delving down into the details of protocol operation, and causing protocols to perform certain actions and then observing these actions and their consequences. This can be done in simulated scenarios or in a “real” network environment such as the Internet. In the Wireshark labs you’ll be doing in this course, you’ll be running various network applications in different scenarios using your own computer. You’ll observe the network protocols in your computer “in action,” interacting and exchanging messages with protocol entities executing elsewhere in the Internet. Thus, you and your computer will be an integral part of these “live” labs. You’ll observe, and you’ll learn, by doing.

In this first Wireshark lab, you’ll get acquainted with Wireshark, and make some simple packet captures and observations.

The basic tool for observing the messages exchanged between executing protocol entities is called a **packet sniffer**. As the name suggests, a packet sniffer captures (“sniffs”) messages being sent/received from/by your computer; it will also typically store and/or display the contents of the various protocol fields in these captured messages. A packet sniffer itself is passive. It observes messages being sent and received by applications and protocols running on your computer, but never sends packets itself. Similarly, received packets are never explicitly addressed to the packet sniffer. Instead, a packet sniffer receives a *copy* of packets that are sent/received from/by application and protocols executing on your machine.

Figure 1 shows the structure of a packet sniffer. At the right of Figure 1 are the protocols (in this case, Internet protocols) and applications (such as a web browser or email client) that normally run on your computer. The packet sniffer, shown within the dashed rectangle in Figure 1 is an addition to the usual software in your computer, and consists of two parts. The **packet capture library** receives a copy of every link-layer frame that is sent from or received by your computer over a given interface (link layer, such as

Ethernet or WiFi). Recall from the discussion from section 1.5 in the text (Figure 1.24¹) that messages exchanged by higher layer protocols such as HTTP, FTP, TCP, UDP, DNS, or IP all are eventually encapsulated in link-layer frames that are transmitted over physical media such as an Ethernet cable or an 802.11 WiFi radio. Capturing all link-layer frames thus gives you all messages sent/received across the monitored link from/by all protocols and applications executing in your computer.

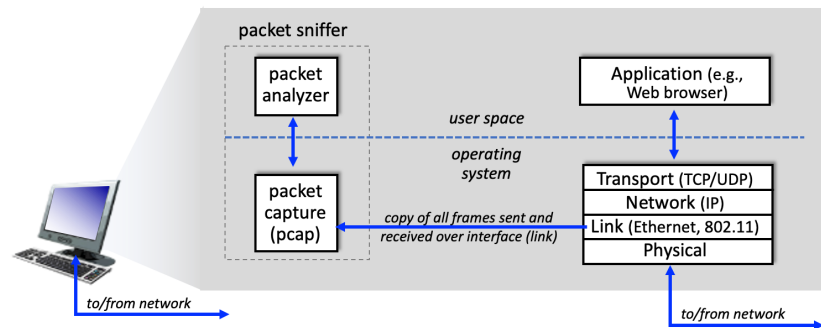


Figure 1: packet sniffer structure

The second component of a packet sniffer is the **packet analyzer**, which displays the contents of all fields within a protocol message. In order to do so, the packet analyzer must “understand” the structure of all messages exchanged by protocols. For example, suppose we are interested in displaying the various fields in messages exchanged by the HTTP protocol in Figure 1. The packet analyzer understands the format of Ethernet frames, and so can identify the IP datagram within an Ethernet frame. It also understands the IP datagram format, so that it can extract the TCP segment within the IP datagram. Finally, it understands the TCP segment structure, so it can extract the HTTP message contained in the TCP segment. Finally, it understands the HTTP protocol and so, for example, knows that the first bytes of an HTTP message will contain the string “GET,” “POST,” or “HEAD,” as shown in Figure 2.8 in the text.

We will be using the Wireshark packet sniffer [<http://www.wireshark.org/>] for these labs, allowing us to display the contents of messages being sent/received from/by protocols at different levels of the protocol stack. (Technically speaking, Wireshark is a packet analyzer that uses a packet capture library in your computer. Also, technically speaking, Wireshark captures link-layer frames as shown in Figure 1, but uses the generic term “packet” to refer to link-layer frames, network-layer datagrams, transport-layer segments, and application-layer messages, so we’ll use the less-precise “packet” term here to go along with Wireshark convention). Wireshark is a free network protocol analyzer that runs on Windows, Mac, and Linux/Unix computers. It’s an ideal packet analyzer for our labs – it is stable, has a large user base and well-documented support that includes a user-guide (http://www.wireshark.org/docs/wsug_html_chunked/), man pages

¹ References to figures and sections are for the 8th edition of our text, *Computer Networks, A Top-down Approach*, 8th ed., J.F. Kurose and K.W. Ross, Addison-Wesley/Pearson, 2020.

(<http://www.wireshark.org/docs/man-pages/>), and a detailed FAQ (<http://www.wireshark.org/faq.html>), rich functionality that includes the capability to analyze hundreds of protocols, and a well-designed user interface. It operates in computers using Ethernet, serial (PPP), 802.11 (WiFi) wireless LANs, and many other link-layer technologies.

Getting Wireshark

In order to run Wireshark, you'll need to have access to a computer that supports both Wireshark and the *libpcap* or *WinPCap* packet capture library. The *libpcap* software will be installed for you, if it is not installed within your operating system, when you install Wireshark. See <http://www.wireshark.org/download.html> for a list of supported operating systems and download sites.

Download and install the Wireshark software:

- Go to <http://www.wireshark.org/download.html> and download and install the Wireshark binary for your computer.

The Wireshark FAQ has a number of helpful hints and interesting tidbits of information, particularly if you have trouble installing or running Wireshark.

Running Wireshark

When you run the Wireshark program, you'll get a startup screen that looks something like the screen below. Different versions of Wireshark will have different startup screens – so don't panic if yours doesn't look exactly like the screen below! The Wireshark documentation states “As Wireshark runs on many different platforms with many different window managers, different styles applied and there are different versions of the underlying GUI toolkit used, your screen might look different from the provided screenshots. But as there are no real differences in functionality these screenshots should still be well understandable.” Well said.

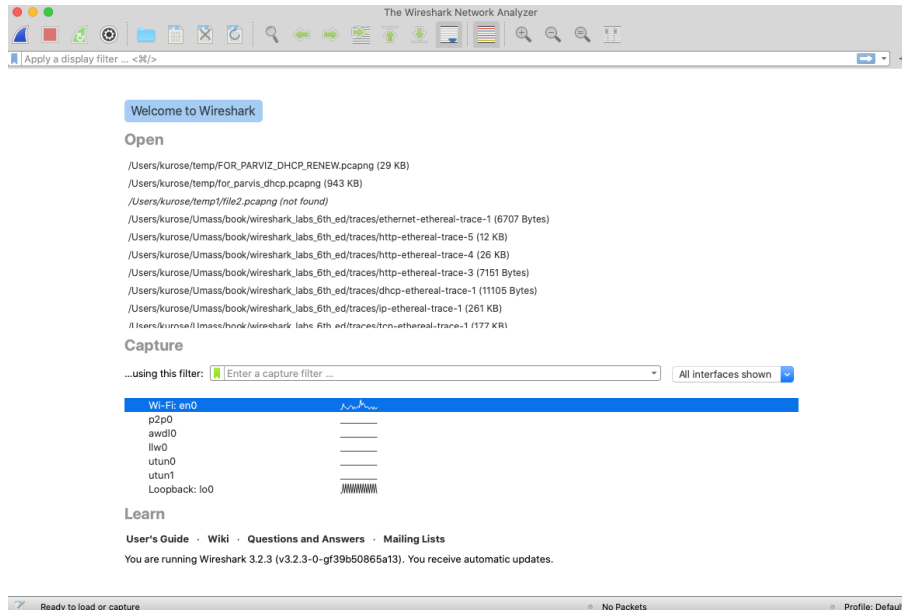


Figure 2: Initial Wireshark Screen

There's not much that's very interesting on this screen. But note that under the Capture section, there is a list of so-called interfaces. The Mac computer we're taking these screenshots from has just one interface – “Wi-Fi en0,” (shaded in blue in Figure 2) which is the interface for Wi-Fi access. All packets to/from this computer will pass through the Wi-Fi interface, so it's here where we'll want to capture packets. On a Mac, double click on this interface (or on another computer locate the interface on startup page through which you are getting Internet connectivity, e.g., mostly likely a WiFi or Ethernet interface, and select that interface).

Let's take Wireshark out for a spin! If you click on one of these interfaces to start packet capture (i.e., for Wireshark to begin capturing all packets being sent to/from that interface), a screen like the one below will be displayed, showing information about the packets being captured. Once you start packet capture, you can stop it by using the Capture pull down menu and selecting Stop (or by clicking on the red square button next to the Wireshark fin in Figure 2).²

² If you are unable to run Wireshark, you can still look at packet traces that were captured on one of the author's (Jim's) computer. Download the zip file <http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces-8E.zip> and extract the file *wireshark-intro-trace*. The traces in this zip file were collected by Wireshark running on one of the author's (Jim's) computers, while performing the steps indicated above. 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 *wireshark-intro-trace* trace file. The resulting display should look similar to Figures 3 and 5. (The Wireshark user interface displays just a bit differently on different operating systems, and in different versions of Wireshark).

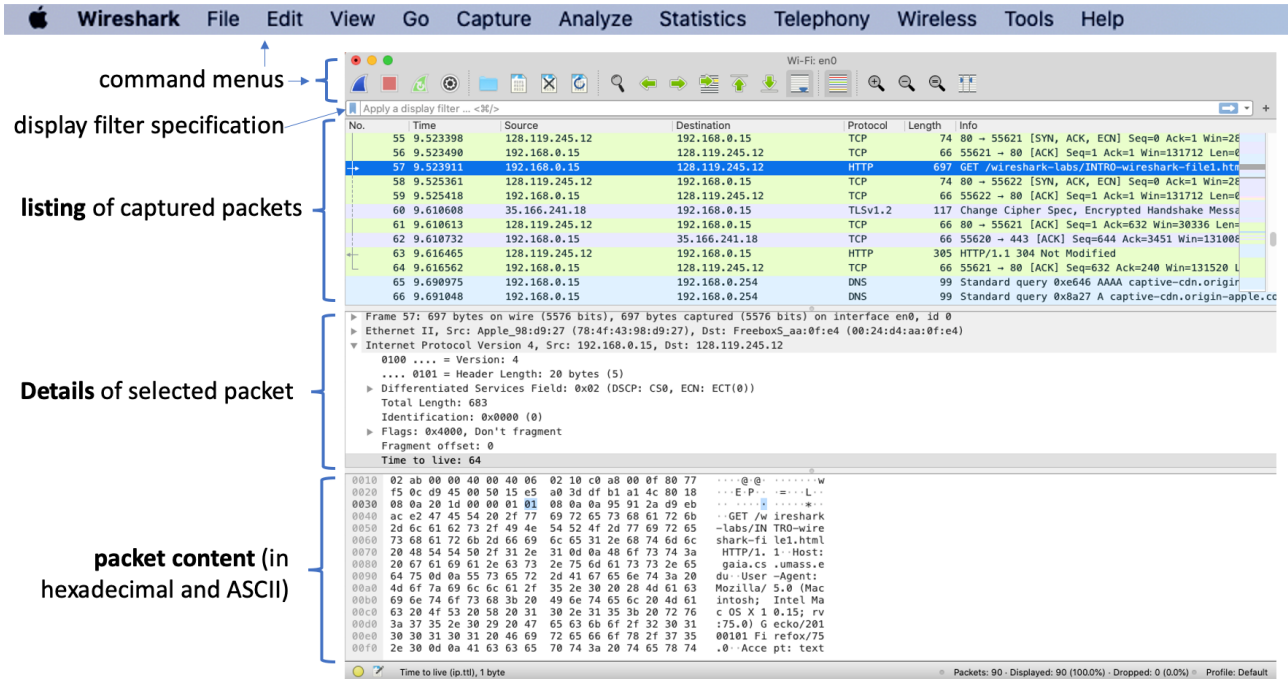


Figure 3: Wireshark window, during and after capture

This looks more interesting! The Wireshark interface has five major components:

- The **command menus** are standard pulldown menus located at the top of the Wireshark window (and on a Mac at the top of the screen as well; the screenshot in Figure 3 is from a Mac). Of interest to us now are the File and Capture menus. The File menu allows you to save captured packet data or open a file containing previously captured packet data and exit the Wireshark application. The Capture menu allows you to begin packet capture.
- The **packet-listing window** displays a one-line summary for each packet captured, including the packet number (assigned by Wireshark; note that this is *not* a packet number contained in any protocol's header), the time at which the packet was captured, the packet's source and destination addresses, the protocol type, and protocol-specific information contained in the packet. The packet listing can be sorted according to any of these categories by clicking on a column name. The protocol type field lists the highest-level protocol that sent or received this packet, i.e., the protocol that is the source or ultimate sink for this packet.
- The **packet-header details window** provides details about the packet selected (highlighted) in the packet-listing window. (To select a packet in the packet-listing window, place the cursor over the packet's one-line summary in the packet-listing window and click with the left mouse button.). These details include information about the Ethernet frame (assuming the packet was sent/received over an Ethernet interface) and IP datagram that contains this packet. The amount of Ethernet and IP-layer detail displayed can be expanded or

minimized by clicking on the plus/minus boxes or right/downward-pointing triangles to the left of the Ethernet frame or IP datagram line in the packet details window. If the packet has been carried over TCP or UDP, TCP or UDP details will also be displayed, which can similarly be expanded or minimized. Finally, details about the highest-level protocol that sent or received this packet are also provided.

- The **packet-contents window** displays the entire contents of the captured frame, in both ASCII and hexadecimal format.
- Towards the top of the Wireshark graphical user interface, is the **packet display filter field**, into which a protocol name or other information can be entered in order to filter the information displayed in the packet-listing window (and hence the packet-header and packet-contents windows). In the example below, we'll use the packet-display filter field to have Wireshark hide (not display) packets except those that correspond to HTTP messages.

Taking Wireshark for a Test Run

The best way to learn about any new piece of software is to try it out! We'll assume that your computer is connected to the Internet via a wired Ethernet interface or a wireless 802.11 WiFi interface. Do the following:

1. Start up your favorite web browser, which will display your selected homepage.
2. Start up the Wireshark software. You will initially see a window similar to that shown in Figure 2. Wireshark has not yet begun capturing packets.
3. To begin packet capture, select the Capture pull down menu and select *Interfaces*. This will cause the “Wireshark: Capture Interfaces” window to be displayed (on a PC) or you can choose Options on a Mac. You should see a list of interfaces, as shown in Figures 4a (Windows) and 4b (Mac).

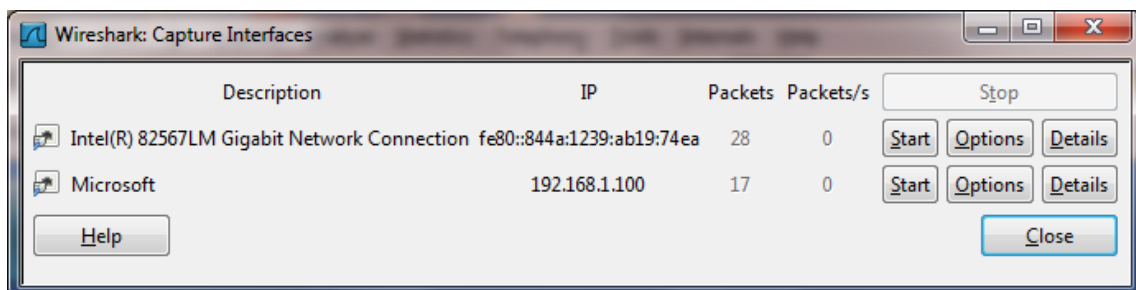


Figure 4a: Wireshark Capture interface window, on a Windows computer

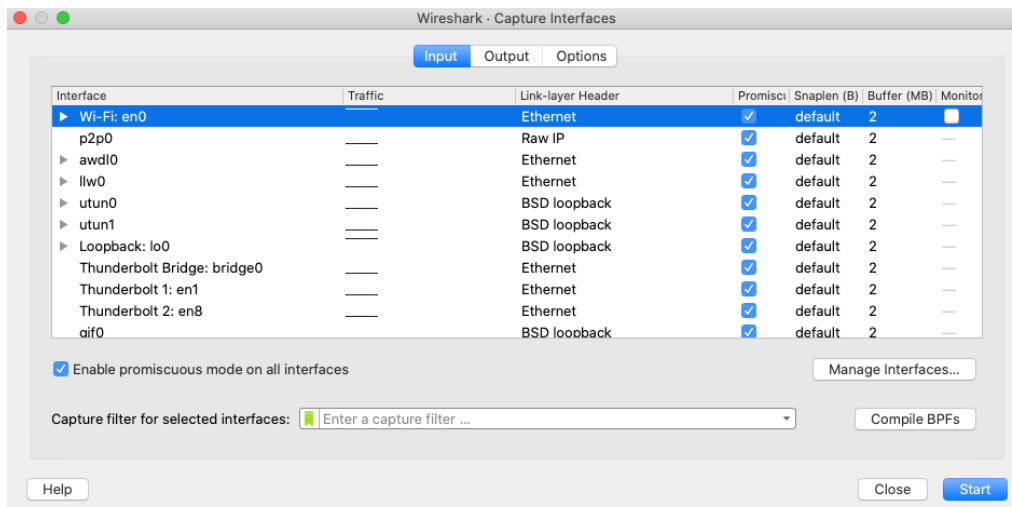


Figure 4b: Wireshark Capture interface window, on a Mac computer

4. You'll see a list of the interfaces on your computer as well as a count of the packets that have been observed on that interface so far. On a Windows machine, click on *Start* for the interface on which you want to begin packet capture (in the case in Figure 4a, the Gigabit network Connection). On a Windows machine, select the interface and click *Start* on the bottom of the window). Packet capture will now begin - Wireshark is now capturing all packets being sent/received from/by your computer!
5. Once you begin packet capture, a window similar to that shown in Figure 3 will appear. This window shows the packets being captured. By selecting *Capture* pulldown menu and selecting *Stop*, or by click on the red Stop square, you can stop packet capture. But don't stop packet capture yet. Let's capture some interesting packets first. To do so, we'll need to generate some network traffic. Let's do so using a web browser, which will use the HTTP protocol that we will study in detail in class to download content from a website.
6. While Wireshark is running, enter the URL:
<http://gaia.cs.umass.edu/wireshark-labs/INTRO-wireshark-file1.html>
and have that page displayed in your browser. In order to display this page, your browser will contact the HTTP server at gaia.cs.umass.edu and exchange HTTP messages with the server in order to download this page, as discussed in section 2.2 of the text. The Ethernet or WiFi frames containing these HTTP messages (as well as all other frames passing through your Ethernet or WiFi adapter) will be captured by Wireshark.
7. After your browser has displayed the `INTRO-wireshark-file1.html` page (it is a simple one line of congratulations), stop Wireshark packet capture by selecting stop in the Wireshark capture window. The main Wireshark window should now look similar to Figure 3. You now have live packet data that contains all protocol

messages exchanged between your computer and other network entities! The HTTP message exchanges with the `gaia.cs.umass.edu` web server should appear somewhere in the listing of packets captured. But there will be many other types of packets displayed as well (see, e.g., the many different protocol types shown in the *Protocol* column in Figure 3). Even though the only action you took was to download a web page, there were evidently many other protocols running on your computer that are unseen by the user. We'll learn much more about these protocols as we progress through the text! For now, you should just be aware that there is often much more going on than “meet's the eye”!

8. Type in “http” (without the quotes, and *in lower case* – all protocol names are in lower case in Wireshark) into the display filter specification window at the top of the main Wireshark window. Then select *Apply* (to the right of where you entered “http”) or just hit return. This will cause only HTTP message to be displayed in the packet-listing window. Figure 5 below shows a screenshot after the http filter has been applied to the packet capture window shown earlier in Figure 3. Note also that in the Selected packet details window, we've chosen to show detailed content for the Hypertext Transfer Protocol application message that was found within the TCP segment, that was inside the IPv4 datagram that was inside the Ethernet II (WiFi) frame. Focusing on content at a specific message, segment, datagram and frame level lets us focus on just what we want to look at (in this case HTTP messages).

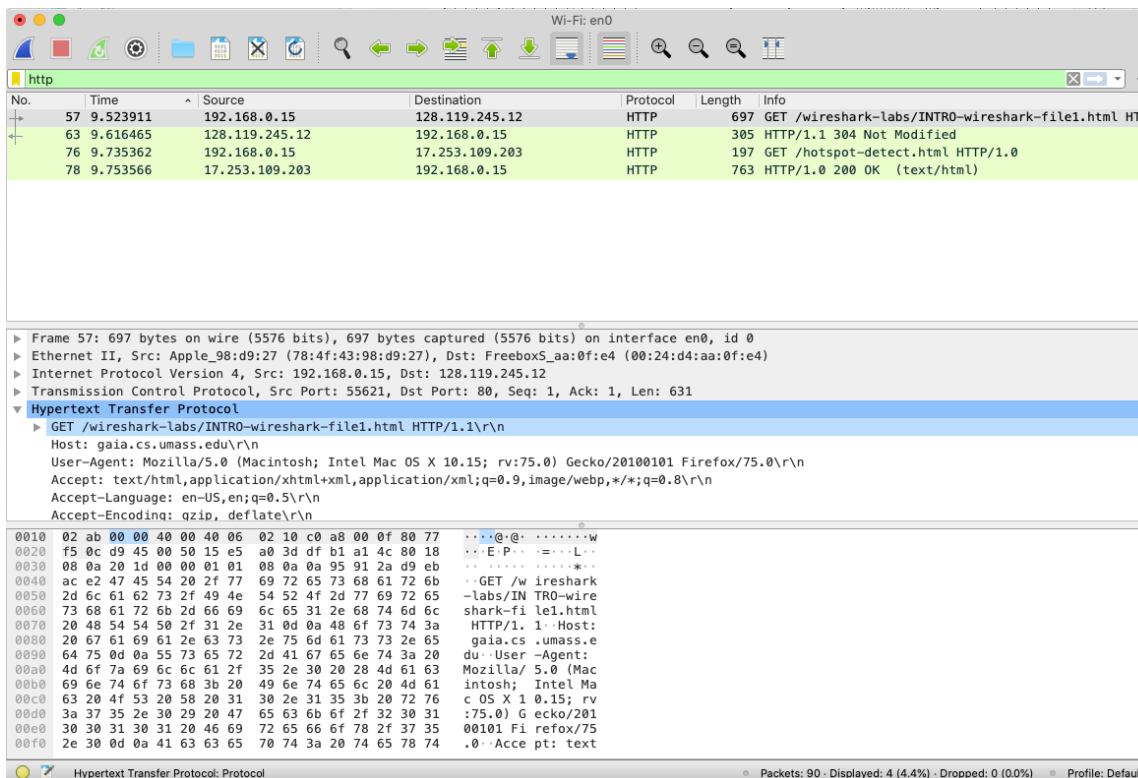


Figure 5: looking at the details of the HTTP message that contained a GET of `http://gaia.cs.umass.edu/wireshark-labs/INTRO-wireshark-file1.html`

9. Find the HTTP GET message that was sent from your computer to the gaia.cs.umass.edu HTTP server. (Look for an HTTP GET message in the “listing of captured packets” portion of the Wireshark window (see Figures 3 and 5) that shows “GET” followed by the gaia.cs.umass.edu URL that you entered. When you select the HTTP GET message, the Ethernet frame, IP datagram, TCP segment, and HTTP message header information will be displayed in the packet-header window³. By clicking on ‘+’ and ‘-’ and right-pointing and down-pointing arrowheads to the left side of the packet details window, *minimize* the amount of Frame, Ethernet, Internet Protocol, and Transmission Control Protocol information displayed. *Maximize* the amount information displayed about the HTTP protocol. Your Wireshark display should now look roughly as shown in Figure 5. (Note, in particular, the minimized amount of protocol information for all protocols except HTTP, and the maximized amount of protocol information for HTTP in the packet-header window).

10. Exit Wireshark

Congratulations! You’ve now completed the first lab!

³ Recall that the HTTP GET message that is sent to the gaia.cs.umass.edu web server is contained within a TCP segment, which is contained (encapsulated) in an IP datagram, which is encapsulated in an Ethernet frame. If this process of encapsulation isn’t quite clear yet, review section 1.5 in the text

What to hand in

The goal of this first lab was primarily to introduce you to Wireshark. The following questions will demonstrate that you've been able to get Wireshark up and running, and have explored some of its capabilities. Answer the following questions, based on your Wireshark experimentation:

1. List 3 different protocols that appear in the protocol column in the unfiltered packet-listing window in step 7 above.
2. How long did it take from when the HTTP GET message was sent until the HTTP OK reply was received? (By default, the value of the Time column in the packet-listing window is the amount of time, in seconds, since Wireshark tracing began. To display the Time field in time-of-day format, select the Wireshark *View* pull down menu, then select *Time Display Format*, then select *Time-of-day*.)
3. What is the Internet address of the gaia.cs.umass.edu (also known as www-net.cs.umass.edu)? What is the Internet address of your computer?
4. Print the two HTTP messages (GET and OK) referred to in question 2 above. To do so, select *Print* from the Wireshark *File* command menu, and select the “*Selected Packet Only*” and “*Print as displayed*” radial buttons, and then click OK.

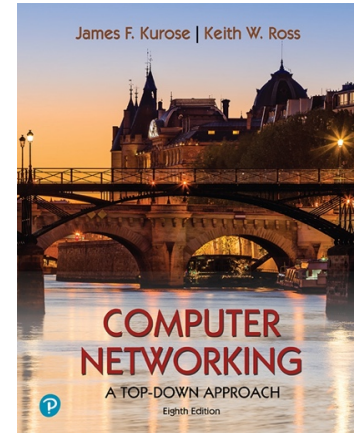
Phase 2:

Ethernet and ARP

Supplement to *Computer Networking: A Top-Down Approach*, 8th 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

© 2005-2020, 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 text¹. 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

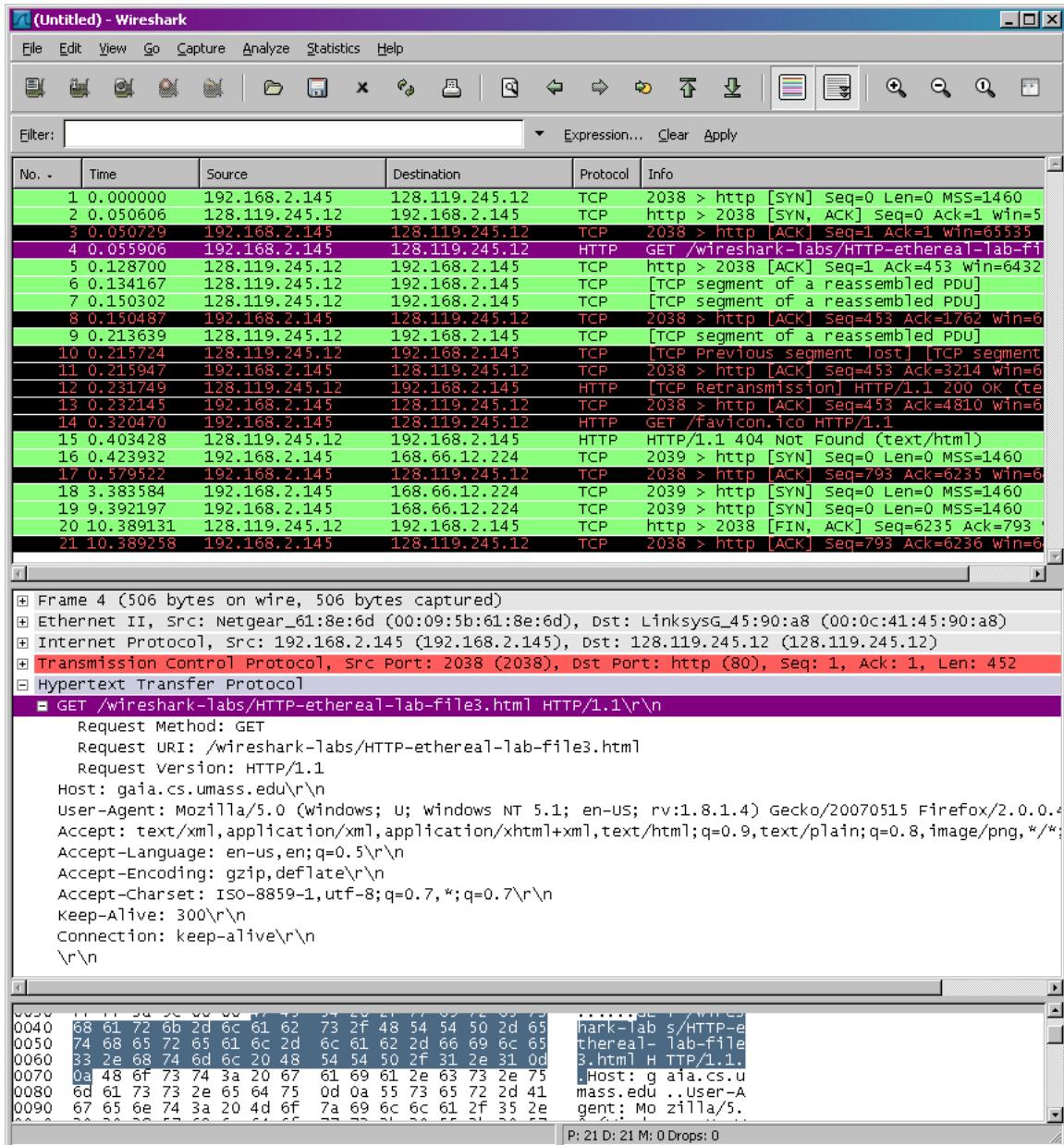
Let's begin by capturing a set of Ethernet frames to study. Do the following²:

- 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.

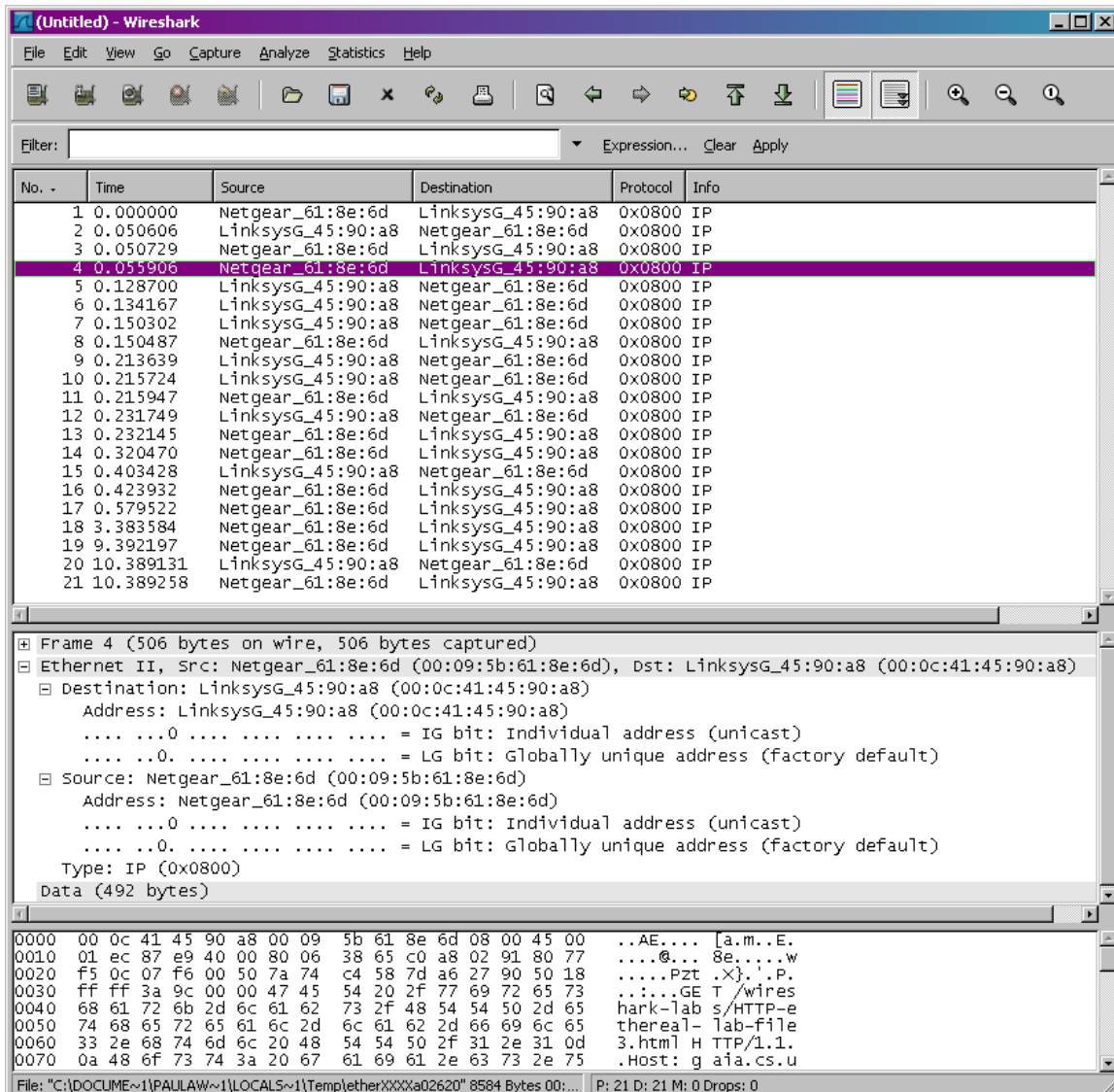
¹ References to figures and sections are for the 8th edition of our text, *Computer Networks, A Top-down Approach*, 8th ed., J.F. Kurose and K.W. Ross, Addison-Wesley/Pearson, 2020.

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 a 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 hand in a printout of the packet(s) within the trace that you used to answer the question asked. Annotate the printout³ 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?

³ What do we mean by “annotate”? If you hand in a paper copy, please highlight where in the printout you’ve found the answer and add some text (preferably with a colored pen) noting what you found in what you’ve highlight. If you hand in an electronic copy, it would be great if you could also highlight and annotate.

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 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).

Using "*arp -a*" command without quotation mark 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?

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.

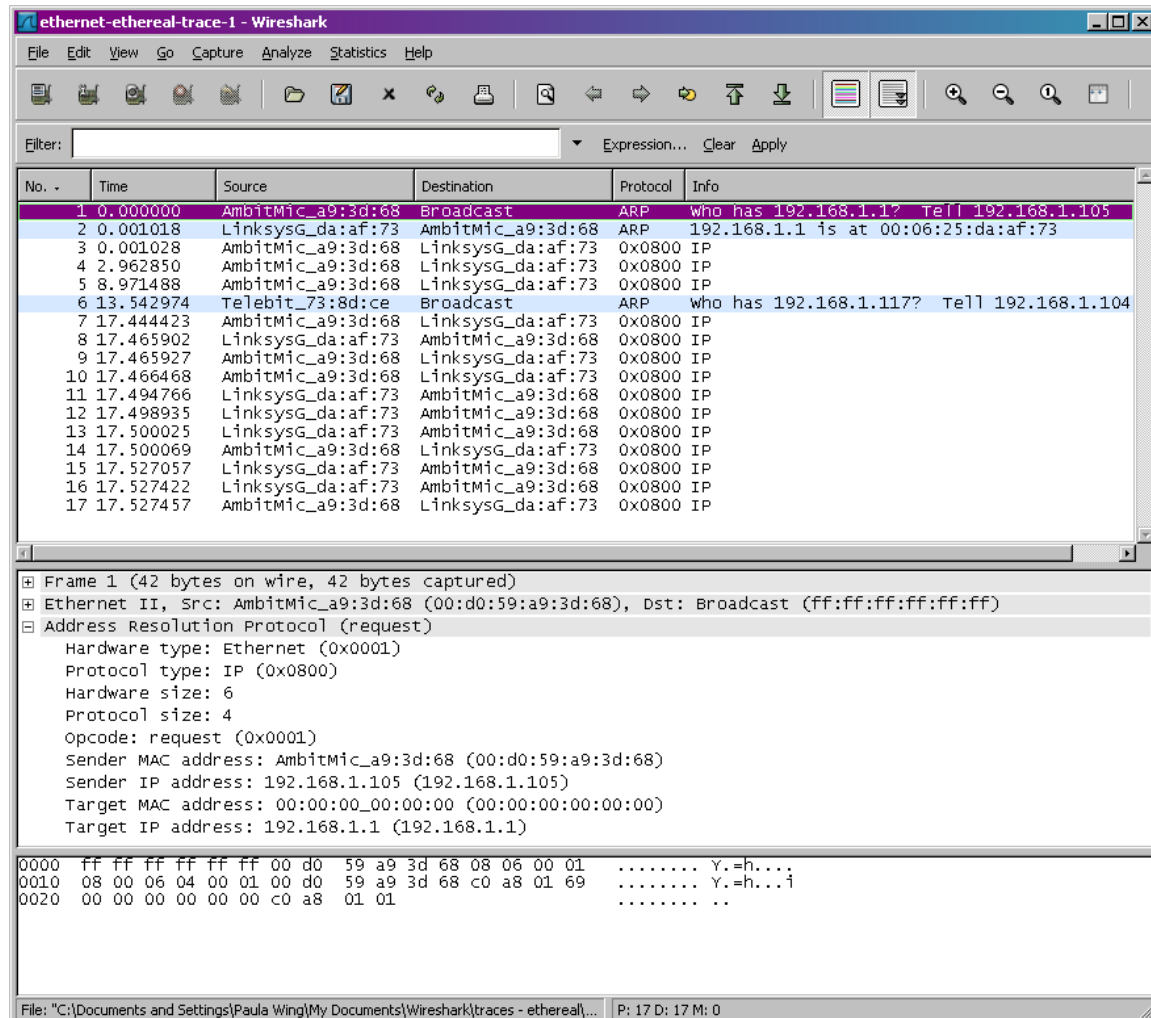
- **MS-DOS.** The MS-DOS *arp -d ** command will clear your ARP cache. The *-d* flag indicates a deletion operation, and the *** is the wildcard that says to delete all table entries.
- **Linux/Unix/MacOS.** The *arp -d ** will clear your ARP cache. In order to run this command you'll need root privileges. If you don't have root privileges and 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 following⁴:

- 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:

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).



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 <http://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>.
 - 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?
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.