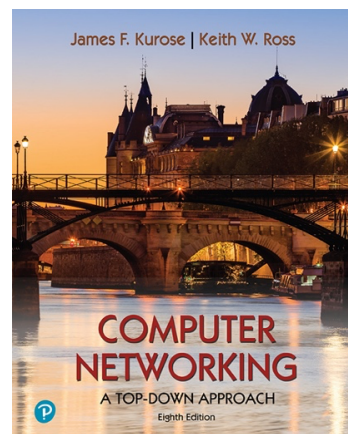


Wireshark Lab: Getting Started v8.1

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

© 2005-2021, 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.

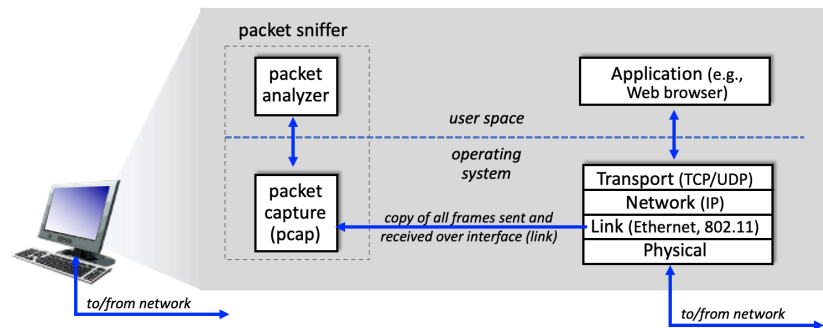


Figure 1: packet sniffer structure

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-

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. Our authors’ website for this book is http://gaia.cs.umass.edu/kurose_ross You’ll find lots of interesting open material there.

guide (http://www.wireshark.org/docs/wsug_html_chunked/), man pages (<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.

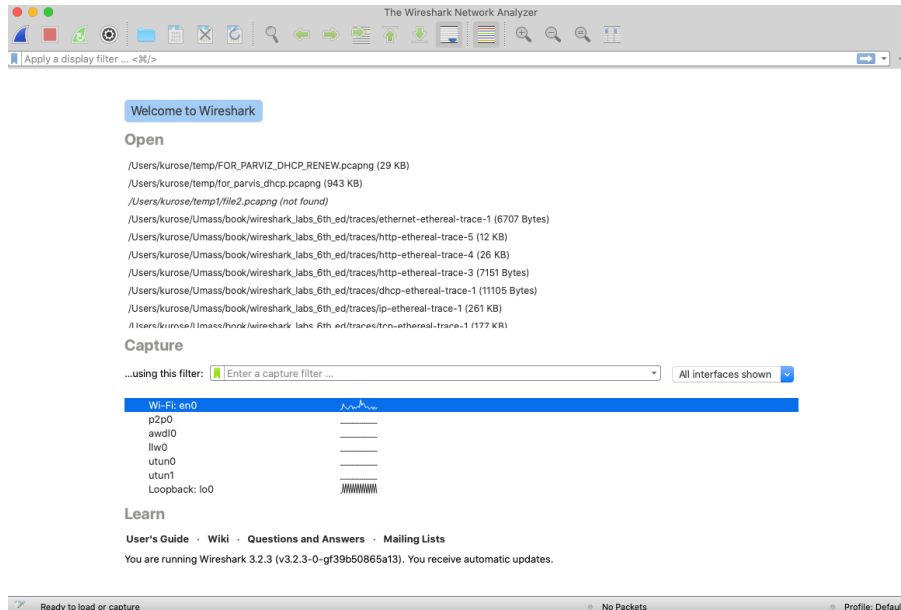


Figure 2: Initial Wireshark Screen

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. You can download the zip file <http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces-8.1.zip> and extract the trace file intro-wireshark-trace-1.pcap. [If you are using an Learning Management System (LMS) to answer questions in this document, you may be instructed to open a different version of this introductory trace file). Once you've downloaded a trace file, you can load it into Wireshark and view the trace using the *File* pull down menu, choosing *Open*, and then selecting the *intro-wireshark-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).

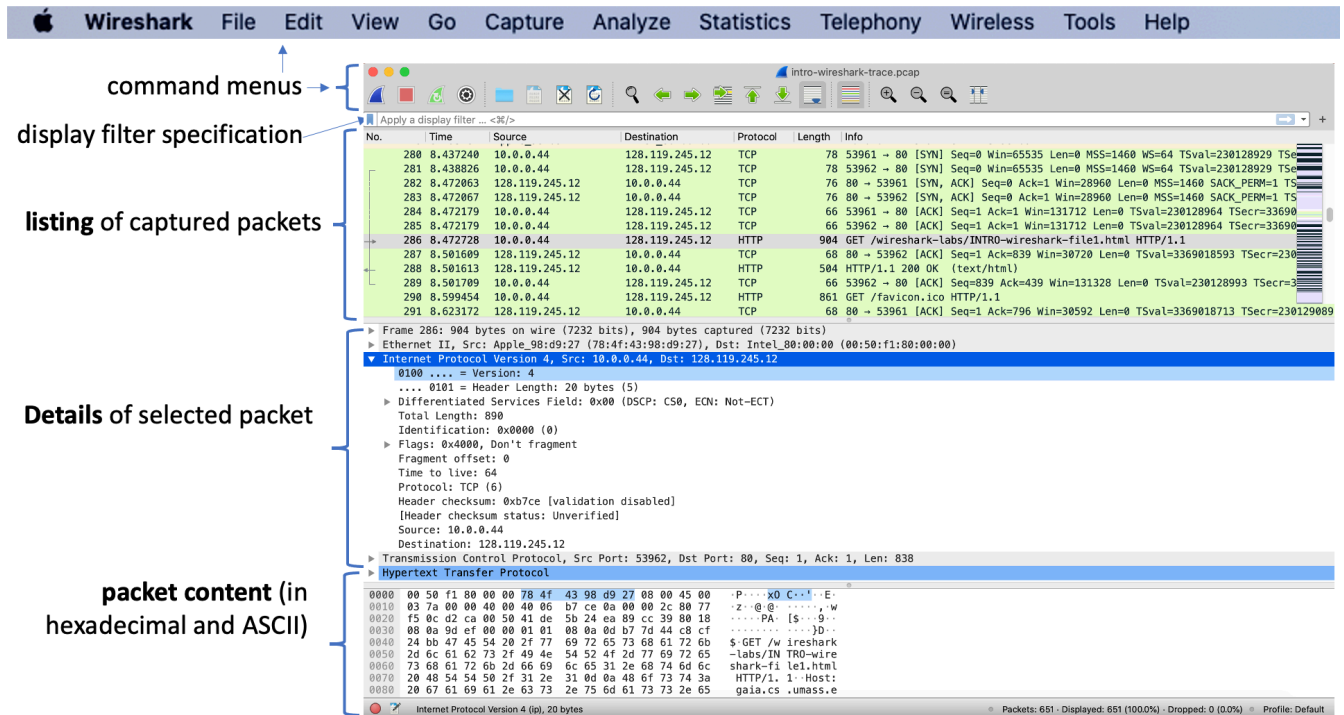


Figure 3: Wireshark window, during and after capture

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).

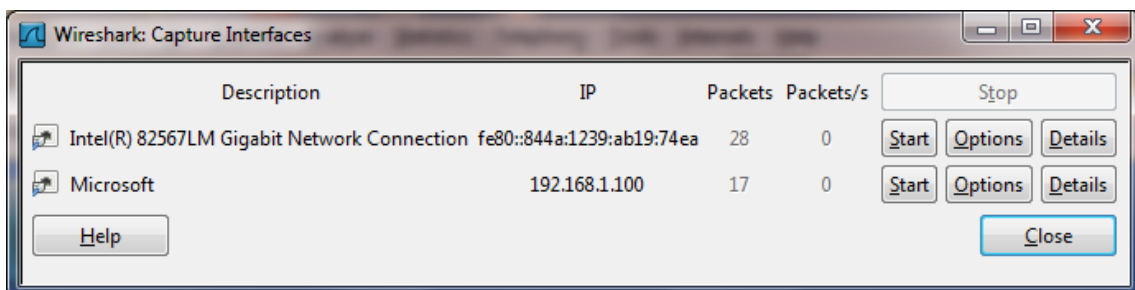


Figure 4a: Wireshark Capture interface window, on a Windows computer

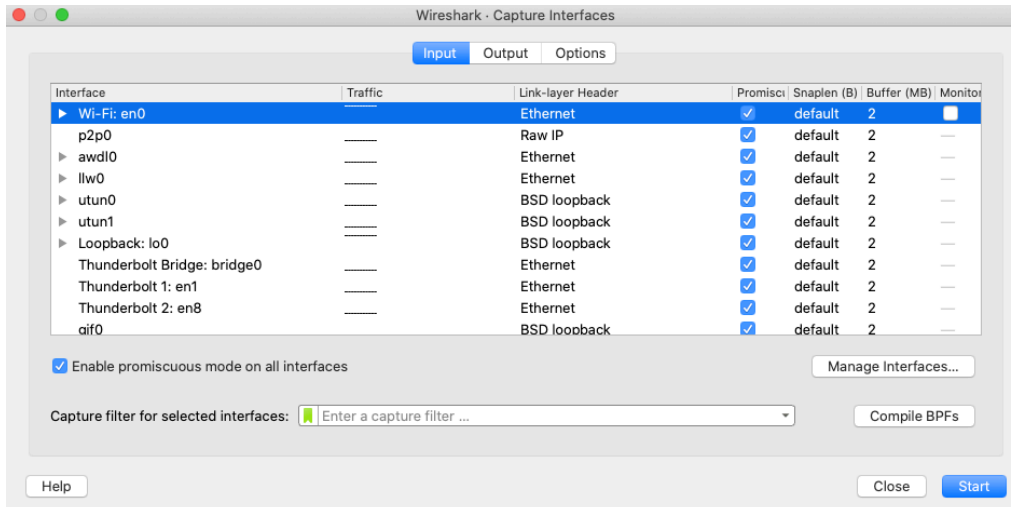


Figure 4b: Wireshark Capture interface window, on a Mac computer

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”!

- Type in “http” (without the quotes, and *in lower case* – all protocol names are in lower case in Wireshark, and make sure to press your enter/return key) 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).

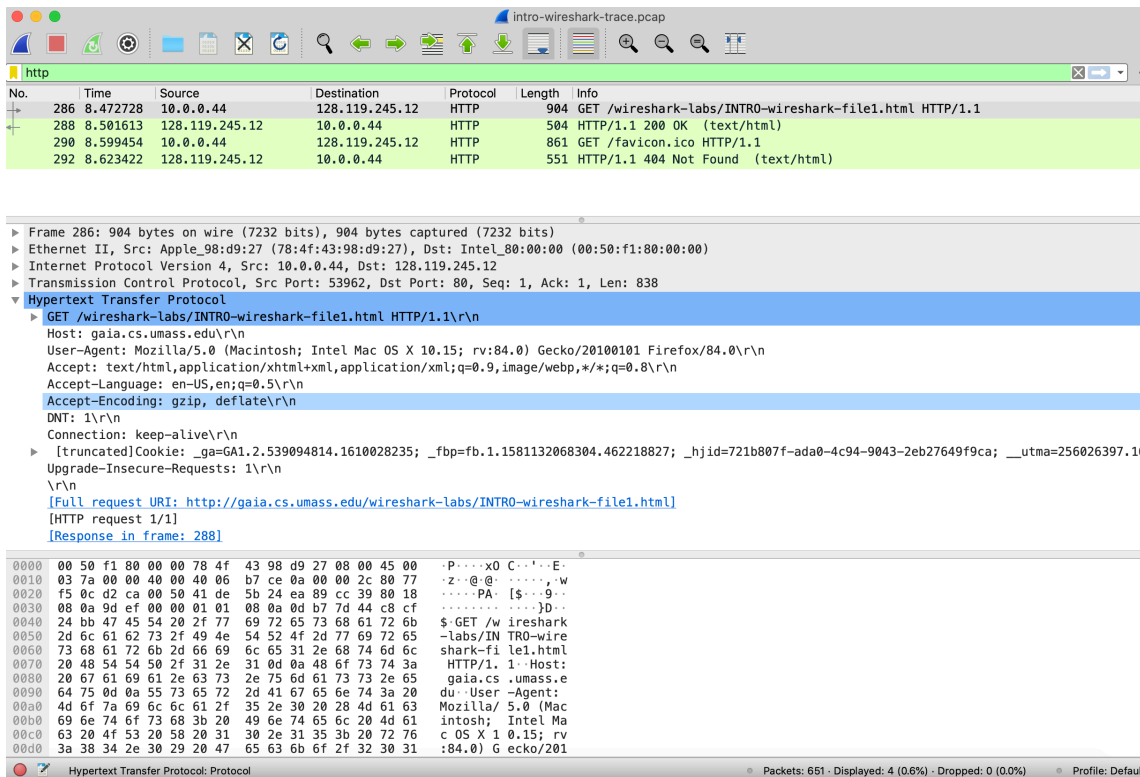


Figure 5: looking at the details of the HTTP message that contained a GET of `http://gaia.cs.umass.edu/wireshark-labs/INTRO-wireshark-file1.html`

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!

Now answer the questions below. If you’re doing this lab as part of class, your teacher will provide details about how to hand in assignments, whether written or in a learning management system (LMS).⁴ If you’re unable to run Wireshark on a live network connection or are answering questions via an LMS, you can download a packet trace file that was captured while following the steps above⁵.

1. Which of the following protocols are shown as appearing (i.e., are listed in the Wireshark “protocol” column) in your trace file: TCP, QUIC, HTTP, DNS, UDP, TLSv1.2?
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. (If you want 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 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

⁴ For the author’s class and written answers, students print out the GET and response messages and indicate where in the message they’ve found the information that answers a question. They do this by marking paper copies with a pen or annotating electronic copies with text in a colored font. There are LMS modules for teachers that allow students to answer these questions online and have answers auto-graded for these Wireshark labs at http://gaia.cs.umass.edu/kurose_ross/lms.htm

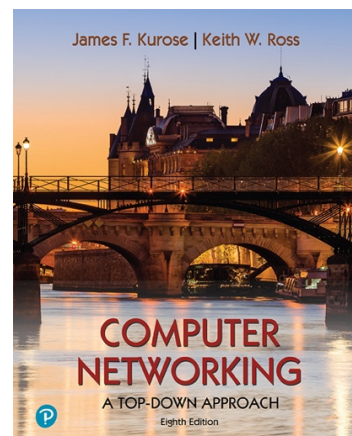
⁵ You can download the zip file <http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces-8.1.zip> and extract the trace file `intro-wireshark-trace1-1`. This trace file can be used to answer these Wireshark lab questions without actually capturing packets on your own. Each trace was made using Wireshark running on one of the author’s computers, while performing the steps indicated in the Wireshark lab. Once you’ve downloaded a trace file, you can load it into Wireshark and view the trace using the *File* pull down menu, choosing *Open*, and then selecting the trace file name.

Wireshark Lab: HTTP v8.1

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

© 2005-2021, J.F Kurose and K.W. Ross, All Rights Reserved



Having gotten our feet wet with the Wireshark packet sniffer in the introductory lab, we’re now ready to use Wireshark to investigate protocols in operation. In this lab, we’ll explore several aspects of the HTTP protocol: the basic GET/response interaction, HTTP message formats, retrieving large HTML files, retrieving HTML files with embedded objects, and HTTP authentication and security. Before beginning these labs, you might want to review Section 2.2 of the text.¹

1. The Basic HTTP GET/response interaction

Let’s begin our exploration of HTTP by downloading a very simple HTML file - one that is very short, and contains no embedded objects. Do the following:

1. Start up your web browser.
2. Start up the Wireshark packet sniffer, as described in the Introductory lab (but don’t yet begin packet capture). Enter “http” (just the letters, not the quotation marks, and in lower case) in the display-filter-specification window, so that only captured HTTP messages will be displayed later in the packet-listing window. (We’re only interested in the HTTP protocol here, and don’t want to see the clutter of all captured packets).
3. Wait a bit more than one minute (we’ll see why shortly), and then begin Wireshark packet capture.
4. Enter the following to your browser
<http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file1.html>
Your browser should display the very simple, one-line HTML file.
5. Stop Wireshark packet capture.

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. Our authors’ website for this book is http://gaia.cs.umass.edu/kurose_ross You’ll find lots of interesting open material there.

Your Wireshark window should look similar to the window shown in Figure 1. If you're unable to run Wireshark on a live network connection, you can download a packet trace that was created when the steps above were followed.²

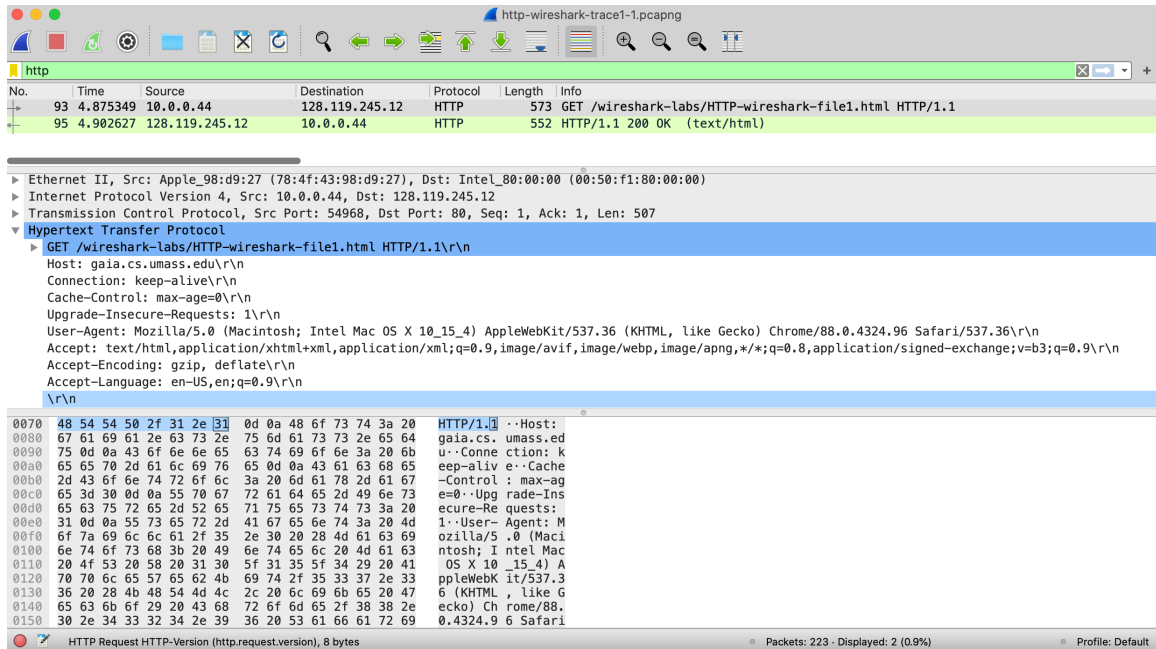


Figure 1: Wireshark Display after `http://gaia.cs.umass.edu/wireshark-labs/ HTTP-wireshark-file1.html` has been retrieved by your browser

The example in Figure 1 shows in the packet-listing window that two HTTP messages were captured: the GET message (from your browser to the `gaia.cs.umass.edu` web server) and the response message from the server to your browser. The packet-contents window shows details of the selected message (in this case the HTTP OK message, which is highlighted in the packet-listing window). Recall that since the HTTP message was carried inside a TCP segment, which was carried inside an IP datagram, which was carried within an Ethernet frame, Wireshark displays the Frame, Ethernet, IP, and TCP packet information as well. We want to minimize the amount of non-HTTP data displayed (we're interested in HTTP here, and will be investigating these other protocols in later labs), so make sure the boxes at the far left of the Frame, Ethernet, IP and TCP information have a plus sign or a right-pointing triangle (which means there is hidden, undisplayed information), and the HTTP line has a minus sign or a down-pointing triangle (which means that all information about the HTTP message is displayed).

² You can download the zip file <http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces-8.1.zip> and extract the trace file `http-wireshark-trace1-1`. These trace files can be used to answer these Wireshark lab questions without actually capturing packets on your own. Each trace was made using Wireshark running on one of the author's computers, while performing the steps indicated in the Wireshark lab. Once you've downloaded a trace file, you can load it into Wireshark and view the trace using the *File* pull down menu, choosing *Open*, and then selecting the trace file name. The resulting display should look similar to Figure 1 (for the `http-wireshark-trace1-1` trace file for this HTTP lab). The Wireshark user interface displays just a bit differently on different operating systems, and in different versions of Wireshark.

(*Note:* You should ignore any HTTP GET and response for favicon.ico. If you see a reference to this file, it is your browser automatically asking the server if it (the server) has a small icon file that should be displayed next to the displayed URL in your browser. We'll ignore references to this pesky file in this lab.).

By looking at the information in the HTTP GET and response messages, answer the following questions. If you're doing this lab as part of class, your teacher will provide details about how to hand in assignments, whether written or in an LMS.³

1. Is your browser running HTTP version 1.0, 1.1, or 2? What version of HTTP is the server running?
2. What languages (if any) does your browser indicate that it can accept to the server?
3. What is the IP address of your computer? What is the IP address of the gaia.cs.umass.edu server?
4. What is the status code returned from the server to your browser?
5. When was the HTML file that you are retrieving last modified at the server?
6. How many bytes of content are being returned to your browser?
7. By inspecting the raw data in the packet content window, do you see any headers within the data that are not displayed in the packet-listing window? If so, name one.

In your answer to question 5 above (assuming you're running Wireshark "live", as opposed to using an earlier-recorded trace file), you might have been surprised to find that the document you just retrieved was last modified within a minute before you downloaded the document. That's because (for this particular file), the gaia.cs.umass.edu server is setting the file's last-modified time to be the current time, and is doing so once per minute. Thus, if you wait a minute between accesses, the file will appear to have been recently modified, and hence your browser will download a "new" copy of the document.

2. The HTTP CONDITIONAL GET/response interaction

Recall from Section 2.2.5 of the text, that most web browsers perform object caching and thus often perform a conditional GET when retrieving an HTTP object. Before performing the steps below, make sure your browser's cache is empty⁴. Now do the following:

- Start up your web browser, and make sure your browser's cache is cleared, as discussed above.
- Start up the Wireshark packet sniffer

³ For the author's class, when answering the following questions with hand-in assignments, students print out the GET and response messages (see the introductory Wireshark lab for an explanation of how to do this) and indicate where in the message they've found the information that answers a question. They do this by marking paper copies with a pen or annotating electronic copies with text in a colored font. There are LMS modules for teachers that allow students to answer these questions online and have answers auto-graded for these Wireshark labs at http://gaia.cs.umass.edu/kurose_ross/lms.htm

⁴ See <https://www.howtogeek.com/304218/how-to-clear-your-history-in-any-browser/> for instructions on clearing your browser cache.

- Enter the following URL into your browser
<http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file2.html>
Your browser should display a very simple five-line HTML file.
- Quickly enter the same URL into your browser again (or simply select the refresh button on your browser)
- Stop Wireshark packet capture, and enter “http” (again, in lower case without the quotation marks) in the display-filter-specification window, so that only captured HTTP messages will be displayed later in the packet-listing window.

If you’re unable to run Wireshark on a live network connection (or unable to get your browser to issue an If-Modified-Since field on the second HTTP GET request), you can download a packet trace that was created when the steps above were followed.⁵ Answer the following questions:

8. Inspect the contents of the first HTTP GET request from your browser to the server. Do you see an “IF-MODIFIED-SINCE” line in the HTTP GET?
9. Inspect the contents of the server response. Did the server explicitly return the contents of the file? How can you tell?
10. Now inspect the contents of the second HTTP GET request from your browser to the server. Do you see an “IF-MODIFIED-SINCE:” line in the HTTP GET? If so, what information follows the “IF-MODIFIED-SINCE:” header?
11. What is the HTTP status code and phrase returned from the server in response to this second HTTP GET? Did the server explicitly return the contents of the file? Explain.

3. Retrieving Long Documents

In our examples thus far, the documents retrieved have been simple and short HTML files. Let’s next see what happens when we download a long HTML file. Do the following:

- Start up your web browser, and make sure your browser’s cache is cleared, as discussed above.
- Start up the Wireshark packet sniffer
- 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, and enter “http” in the display-filter-specification window, so that only captured HTTP messages will be displayed.

⁵ If you’re unable to run Wireshark on a live network connection, you can download the zip file <http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces-8.1.zip> and extract the trace file http-wireshark-trace2-1.

⁶ *Hint:* ideally, you should see an If-Modified-Since header since you’ve just downloaded this page a few seconds ago. However, depending on the browser you’re using, and the format of the server’s earlier response to your initial GET, your browser may not include an If-Modified-Since even if the document has been downloaded and caches. The Chrome browser is pretty good at regularly using If-Modified-Since. But Safari and Firefox are much more finicky about when to use If-Modified-Since. Life isn’t always as easy in practice as it is in theory!

In the packet-listing window, you should see your HTTP GET message, followed by a multiple-packet TCP response to your HTTP GET request. Make sure your Wireshark display filter is cleared so that the multi-packet TCP response will be displayed in the packet listing.

This multiple-packet response deserves a bit of explanation. Recall from Section 2.2 (see Figure 2.9 in the text) that the HTTP response message consists of a status line, followed by header lines, followed by a blank line, followed by the entity body. In the case of our HTTP GET, the entity body in the response is the *entire* requested HTML file. In our case here, the HTML file is rather long, and at 4500 bytes is too large to fit in one TCP packet. The single HTTP response message is thus broken into several pieces by TCP, with each piece being contained within a separate TCP segment (see Figure 1.24 in the text). In recent versions of Wireshark, Wireshark indicates each TCP segment as a separate packet, and the fact that the single HTTP response was fragmented across multiple TCP packets is indicated by the “TCP segment of a reassembled PDU” in the Info column of the Wireshark display.

Answer the following questions⁷:

12. How many HTTP GET request messages did your browser send? Which packet number in the trace contains the GET message for the Bill of Rights?
13. Which packet number in the trace contains the status code and phrase associated with the response to the HTTP GET request?
14. What is the status code and phrase in the response?
15. How many data-containing TCP segments were needed to carry the single HTTP response and the text of the Bill of Rights?

4. HTML Documents with Embedded Objects

Now that we’ve seen how Wireshark displays the captured packet traffic for large HTML files, we can look at what happens when your browser downloads a file with embedded objects, i.e., a file that includes other objects (in the example below, image files) that are stored on another server(s).

Do the following:

- Start up your web browser, and make sure your browser’s cache is cleared, as discussed above.
- Start up the Wireshark packet sniffer
- Enter the following URL into your browser
<http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file4.html>

⁷ If you’re unable to run Wireshark on a live network connection, you can download the zip file <http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces-8.1.zip> and extract the trace file `http-wireshark-trace3-1`.

Your browser should display a short HTML file with two images. These two images are referenced in the base HTML file. That is, the images themselves are not contained in the HTML; instead the URLs for the images are contained in the downloaded HTML file. As discussed in the textbook, your browser will have to retrieve these logos from the indicated web sites. Our publisher's logo is retrieved from the gaia.cs.umass.edu web site. The image of our 8th edition cover (one of our favorite covers) is stored at a server in France.

- Stop Wireshark packet capture, and enter “http” in the display-filter-specification window, so that only captured HTTP messages will be displayed.

Answer the following questions⁸:

16. How many HTTP GET request messages did your browser send? To which Internet addresses were these GET requests sent?
17. Can you tell whether your browser downloaded the two images serially, or whether they were downloaded from the two web sites in parallel? Explain.

5 HTTP Authentication

Finally, let's try visiting a web site that is password-protected and examine the sequence of HTTP message exchanged for such a site. The URL http://gaia.cs.umass.edu/wireshark-labs/protected_pages/HTTP-wireshark-file5.html is password protected. The username is “wireshark-students” (without the quotes), and the password is “network” (again, without the quotes). So let's access this “secure” password-protected site. Do the following:

- Make sure your browser's cache is cleared, as discussed above, and close down your browser. Then, start up your browser
- Start up the Wireshark packet sniffer
- Enter the following URL into your browser
http://gaia.cs.umass.edu/wireshark-labs/protected_pages/HTTP-wireshark-file5.html
Type the requested user name and password into the pop up box.
- Stop Wireshark packet capture, and enter “http” in the display-filter-specification window, so that only captured HTTP messages will be displayed later in the packet-listing window.
- *Note:* If you are unable to run Wireshark on a live network connection, you can use the “classic” http-ethereal-trace-5 packet trace, or other additional traces, as notes in footnote 2, to answer the questions below.

Now let's examine the Wireshark output. You might want to first read up on HTTP authentication by reviewing the easy-to-read material on “HTTP Access Authentication Framework” at [http://frontier.userland.com/stories/storyReader\\$2159](http://frontier.userland.com/stories/storyReader$2159)

⁸ If you're unable to run Wireshark on a live network connection, you can download the zip file <http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces-8.1.zip> and extract the trace file `http-wireshark-trace4-1`.

Answer the following questions⁹:

18. What is the server's response (status code and phrase) in response to the initial HTTP GET message from your browser?
19. When your browser's sends the HTTP GET message for the second time, what new field is included in the HTTP GET message?

The username (wireshark-students) and password (network) that you entered are encoded in the string of characters (d2lyZXNoYXJrLXN0dWRlbnRzOm5ldHdvcms=) following the "Authorization: Basic" header in the client's HTTP GET message. While it may appear that your username and password are encrypted, they are simply encoded in a format known as Base64 format. The username and password are *not* encrypted! To see this, go to <http://www.motobit.com/util/base64-decoder-encoder.asp> and enter the base64-encoded string d2lyZXNoYXJrLXN0dWRlbnRz and decode. *Voila!* You have translated from Base64 encoding to ASCII encoding, and thus should see your username! To view the password, enter the remainder of the string Om5ldHdvcms= and press decode. Since anyone can download a tool like Wireshark and sniff packets (not just their own) passing by their network adaptor, and anyone can translate from Base64 to ASCII (you just did it!), it should be clear to you that simple passwords on WWW sites are not secure unless additional measures are taken.

Fear not! As we will see in Chapter 8, there are ways to make WWW access more secure. However, we'll clearly need something that goes beyond the basic HTTP authentication framework!

⁹ If you're unable to run Wireshark on a live network connection, you can download the zip file <http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces-8.1.zip> and extract the trace file http-wireshark-trace5-1.

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 or (if you are using the trace file) the computer that sent the HTTP GET message?

To answer the following two questions, you'll need to select the TCP packet containing the HTTP GET request (hint: this is packet number 286⁶). The purpose of these next two questions is to familiarize you with using Wireshark's "Details of selected packet window"; see Figure 3. To do this, click on Packet 286 (your screen should look similar to Figure 3). To answer the first question below, then look in the "Details of selected packet" window toggle the triangle for HTTP (your screen should then look similar to Figure 5); for the second question below, you'll need to expand the information on the Transmission Control Protocol (TCP) part of this packet.

4. Expand the information on the HTTP message in the Wireshark "Details of selected packet" window (see Figure 3 above) so you can see the fields in the HTTP GET request message. What type of Web browser issued the HTTP request? The answer is shown at the right end of the information following the "User-Agent:" field in the expanded HTTP message display. [This field value in the HTTP message is how a web server learns what type of browser you are using.]
 - Firefox, Safari, Microsoft Internet Edge, Other
5. Expand the information on the Transmission Control Protocol for this packet in the Wireshark "Details of selected packet" window (see Figure 3 in the lab writeup) so you can see the fields in the TCP segment carrying the HTTP message. What is the destination port number (the number following "Dest Port:" for the TCP segment containing the HTTP request) to which this HTTP request is being sent?

And finally ...

6. 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.

⁶ Remember that this "packet number" is assigned by Wireshark for listing purposes only; it is NOT a packet number contained in any real packet header.