|  |
| --- |
| Wireshark Lab - 1  Getting Started & HTTP/TCP  Version: February 2016  © Jinwei Cao. All Rights Reserved. Adapted from © 2005 J.F. Kurose, K.W. Ross. |

*“Tell me and I forget. Show me and I remember. Involve me and I understand.”*

Chinese proverb

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 these Wireshark labs, we’ll take the latter approach. You’ll be running various network applications in different scenarios using a computer on your desk, at home, or in a lab. 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.

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 ftp 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. Recall 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. In Figure 1, the assumed physical media is an Ethernet, and so all upper layer protocols are eventually encapsulated within an Ethernet frame. Capturing all link-layer frames thus gives you all messages sent/received from/by all protocols and applications executing in your computer.



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”.

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). Wireshark is a free network protocol analyzer that runs on Windows, Linux/Unix, and Mac 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 (<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 more than 500 protocols, and a well-designed user interface. It operates in computers using Ethernet to connect to the Internet, as well as so-called point-to-point protocols such as PPP.

Part I - Getting Started

1. Getting Wireshark

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

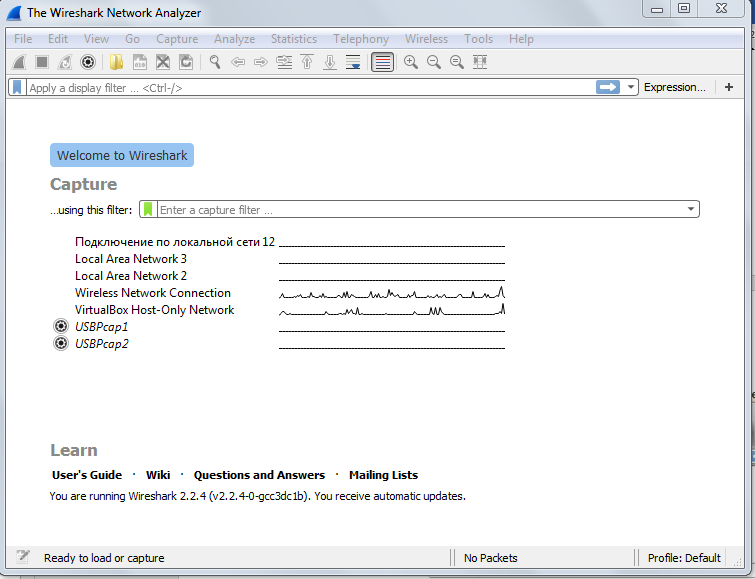
Download and install the Wireshark and (if needed) *libpcap* software:

* If needed, download and install the *libpcap* software. Pointers to the *libpcap* software are provided from the Wireshark download pages. For Windows machines, the *libpcap* software is known as *WinPCap*, and can be found at <http://www.winpcap.org>. See FAQ question #1 at <http://www.winpcap.org/misc/faq.htm> to determine whether or not *WinPCap* is already installed on your machine.
* Go to <http://www.wireshark.org> and download and install the Wireshark binary for your computer.
* Download the Wireshark user guide. You will most likely only need Chapters 1 and 3.

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

2. Running Wireshark

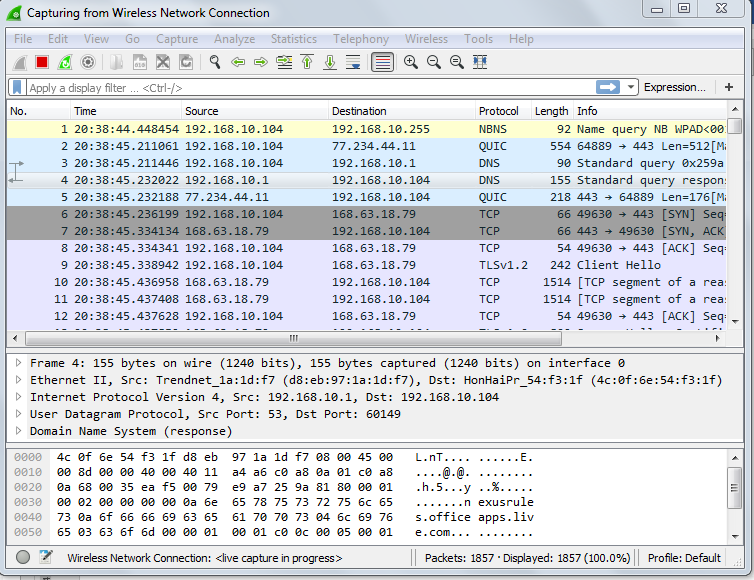
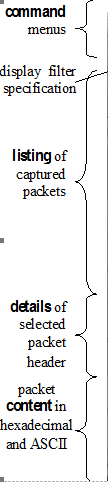
When you run the Wireshark program (Version 2.2.4), the “Wireshark Start Window” shown in Figure 2 will be displayed. Initially, you will be asked to select a network interface (input source) for your packet capture. Before this step, no capturing data will be displayed.



**Figure 2:** Wireshark Start Window

**Note**: For Windows OS if you don’t see all your network interfaces, try running the wire-shark application with administrative privileges.

To start capturing packets, double click an input source of your choice. For this lab, since we currently connect to a Wi-Fi network, we will select “Wireless Network Connection” as an example, after double clicking it the “Graphical User Interface” shown in Figure 3 will be displayed.



**Figure 3:** Wireshark Graphical User Interface

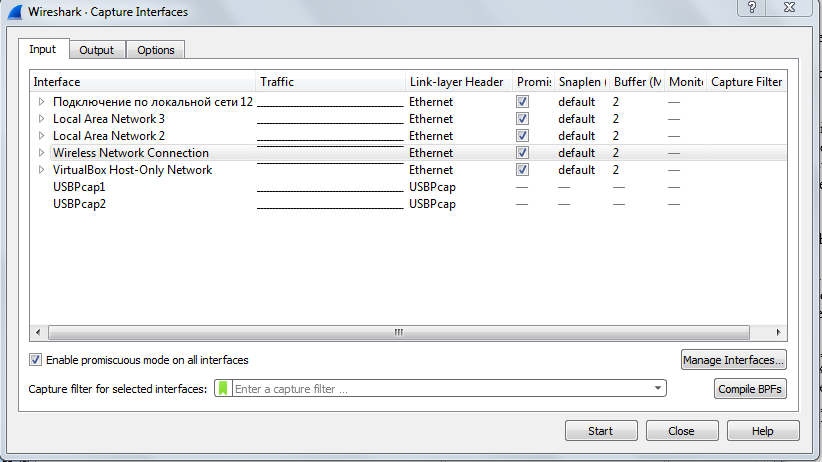
The Wireshark interface has five major components:

* The **command menus** are standard pulldown menus located at the top of the window. 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; 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 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 right-pointing or down-pointing arrowhead 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.

3. Taking Wireshark for a Test Run

The best way to learn about any new piece of software is to try it out! Do the following

1. Start up your favorite web browser first, which will display your selected homepage.
2. Start up the Wireshark software. You will initially see a window similar to the one shown in Figure 2. If you wait 5 seconds, you will see that if there is any network activity under a network interface listed on the “Wireshark Start Window”, there will be a vibrating line to the right hand side of the network interface. If there isn’t any network activity, the line will be straight.
3. To begin packet capture, double click the network interface of your choice, and the “Wireshark User Graphical Interface window” shown in Figure 3 will be displayed. Alternatively, you can select the **Capture** pull down menu and select “**Options**”. This will cause the “Wireshark: Capture Interfaces” window to be displayed, as shown in Figure 4. On the “Wireshark Capture Interfaces” window, click on a network interface, and then click the Start button. Once the capture process begins, on the “Wireshark User Graphical Interface” window, you will start to see captured packets. If no packet data is displayed in the packet-listing, packet-header, or packet-contents section, that is probably because there are no activity captured with the network interface you selected. For this lab, if your computer is connected to a Wi-fi network, please select “Wireless Network connection” as your network interface.



Windows Operating System Version

**Figure 4:** Wireshark Capture Interfaces Window

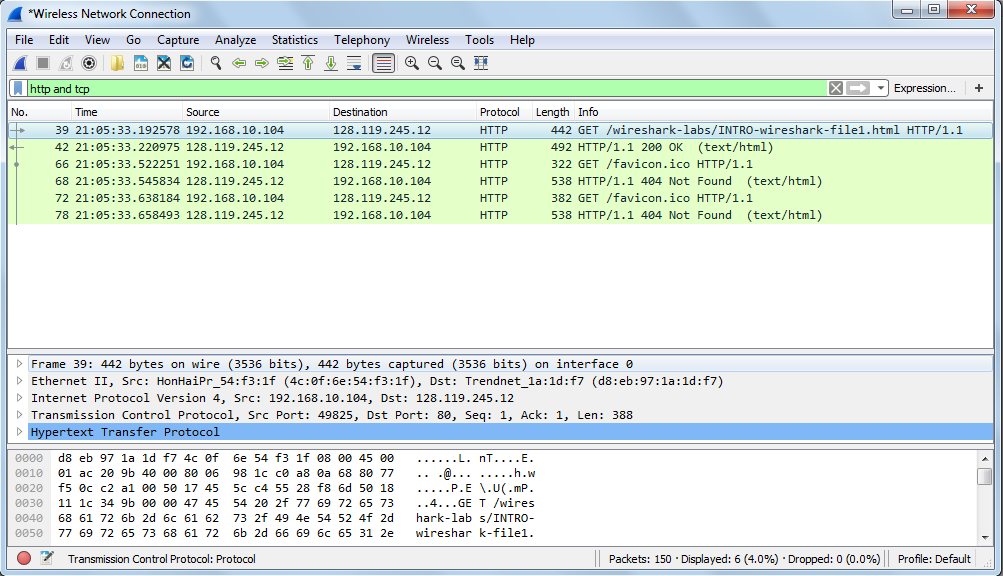
1. You can use all of the default values in this window. The network interfaces (i.e., the physical connections) that your computer has to the network will be shown in the Interface pull down menu at the top of the **Capture Interfaces** window. In case your computer has more than one active network interface (e.g., if you have both a wireless and a wired Ethernet connection), you will need to select an interface that is being used to send and receive packets (mostly likely the wired interface; if you use the wireless interface, you might need to **uncheck** the “**Enable promiscuous mode on all interfaces**” option.). After selecting the network interface, click Start. Packet capture will now begin - all packets being sent/received from/by your computer are now being captured by Wireshark!

1. While Wireshark is running, enter the URL:   
    [**http://gaia.cs.umass.edu/wireshark-labs/INTRO-wireshark-file1.html**](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. The Ethernet frames containing these HTTP messages will be captured by Wireshark.
2. After your browser has displayed the INTRO-wireshark-file1.html page, stop Wireshark packet capture by clicking on the **Stop** button (the red square on top of the User Graphical Interface window. Alternatively, you can click on Capture on the menu on top and then click on the Stop button from the pull down list. This will cause the main Wireshark window to stop capturing any more packets, but it will still display all packets already captured since you began packet capture. 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. For example, if you use Wireshark in the Purnell 028 lab, you might find that many of the packets are using the **SSDP** protocol. SSDP, which stands for Simple Service Discovery Protocol, is a protocol that allows networked computers to discover remote services and devices that are available for them to use. It also allows networked computers to announce new services and devices that they’re willing and able to share. Currently, most SSDP-capable devices are printers, fax machines, and other computers on a network. However in the future it may even be possible to have an SSDP-capable coffee maker that is automatically detected by your computer and that can be easily programmed to make particular types of coffee on different days and at different times for each network client.

We’ll learn much more about these protocols as we progress through the course! For now, you should just be aware that there is often much more going on than “meets the eye”!

1. Type in “http and tcp” (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 press Enter key on your keyboard, or select *Apply* (the blue arrow to the right of where you entered “http and tcp”). This will cause only HTTP message to be displayed in the packet-listing window.
2. Select the first http message shown in the packet-listing window. This should be the HTTP GET message that was sent from your computer to the gaia.cs.umass.edu HTTP server. 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 window[[1]](#footnote-1). By clicking on the arrow that points to the right hand side or the arrow that points down to the left side of the packet details window, you can *maximize* or *minimize* the amount of protocol information displayed. Your Wireshark display should now look roughly as shown in Figure 5.
3. Exit Wireshark.

Congratulations! You’ve now completed the first half of the lab.

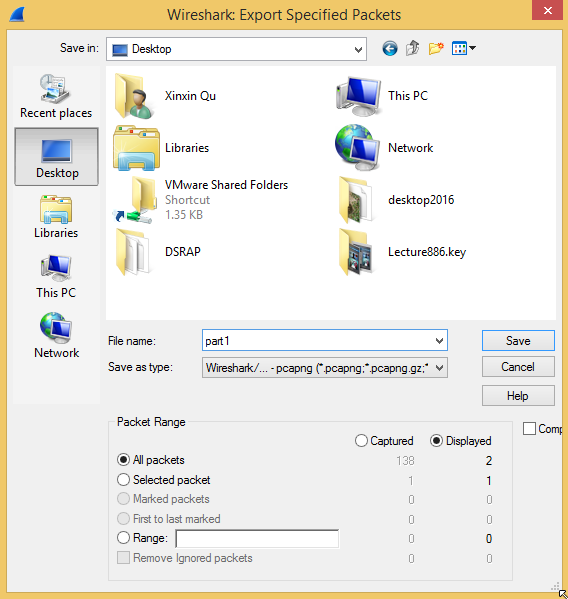


**Figure 5:** Wireshark display after step 9

***What to hand in***

The goal of the first half of the 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 five 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 (IP) address of the gaia.cs.umass.edu? What is the Internet (IP) address of your computer?
4. Save the two HTTP messages displayed in step 9 above. To do so, select *“****Export Specified Packets****”* from the Wireshark***File*** command menu, and select “***All Packets****”* and *“****Displayed****” as shown in Figure 6*, specify the location where you want to save the files, and then click Save.



**Figure 6:** Wireshark display for saving displayed messages (Windows Version)

Part II - HTTP

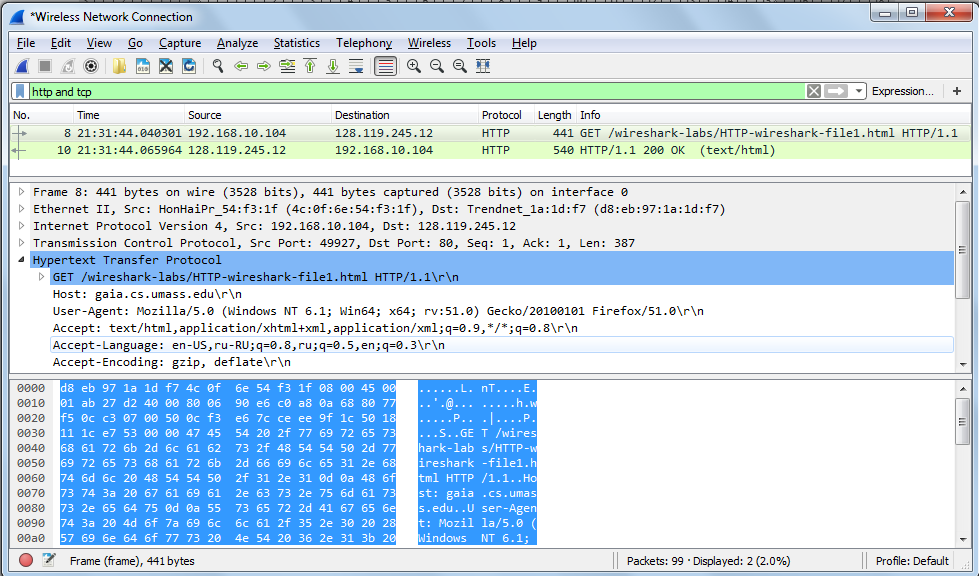
Having gotten our feet wet with the Wireshark packet sniffer, we’re now ready to use Wireshark to investigate protocols in operation. In this second half of the lab, we’ll explore several aspects of the HTTP protocol: the basic GET/response interaction, HTTP message formats, retrieving HTML files with embedded objects, and HTTP caching.

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 part I (but don’t yet begin packet capture). Enter “http and tcp” (just the letters, not the quotation marks) 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. 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.

Your Wireshark window should look similar to the window shown in Figure 1. If you are 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]](#footnote-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 GET 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 an arrow that points to the right (which means there is hidden, undisplayed information), and the HTTP line has an arrow that points down (which means that all information about the HTTP message is displayed).

(*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.).

***What to hand in***

By looking at the information in the HTTP GET and response messages, answer the following questions. When answering the following questions, you should save the GET and response messages and indicate where in the message you’ve found the information that answers the following questions.

1. Is your browser running HTTP version 1.0 or 1.1? 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? Of the gaia.cs.umass.edu server?
4. What is the status code returned from the server to your browser?
5. When the HTML file that you are retrieving was last modified at the server? When was it returned to your browser?
6. How many bytes of content are being returned to your browser?
7. Save the HTTP messages displayed. To do so, select *“Export Specified Packets”* from the Wireshark *File* command menu, and select “*All Packets”* and *“Displayed”.* Specify the location where you want to save the HTTP messages, type in a name and then click Save.

2. HTML Documents with Embedded Objects

Now let’s 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. (To do this under Firefox, select *Tools->Clear Recent History*, or for Internet Explorer, select *Tools*

*->Internet Options->Delete File;* these actions will remove cached files form your browser’s cache.) Now do the following:

* Start up the Wireshark packet sniffer
* Enter the following URL into your browser  
  **<http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file4.html>**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. Your browser will have to retrieve these logos from the indicated web sites. The publisher’s logo is retrieved from the www.aw-bc.com web site. The image of our book’s cover is stored at the manic.cs.umass.edu server.
* Stop Wireshark packet capture, and enter “http and tcp” in the display-filter-specification window, so that only captured HTTP messages will be displayed.
* (*Note:* If you are unable to run Wireshark on a live network connection, you can use the http-ethereal-trace-4 packet trace to answer the questions below; see footnote 1. This trace file was gathered while performing the steps above on one of the author’s computers.)

***What to hand in***

Answer the following questions:

1. How many HTTP GET request messages were sent by your browser?
2. To which Internet addresses were these GET requests sent?
3. How many HTTP response messages were received by your browser?
4. Save the HTTP messages displayed. To do so, select *“Export Specified Packets”* from the Wireshark *File* command menu, and select “*All Packets”* and *“Displayed”,* Specify the location where you want to save the HTTP messages, type in a name and then click Save.

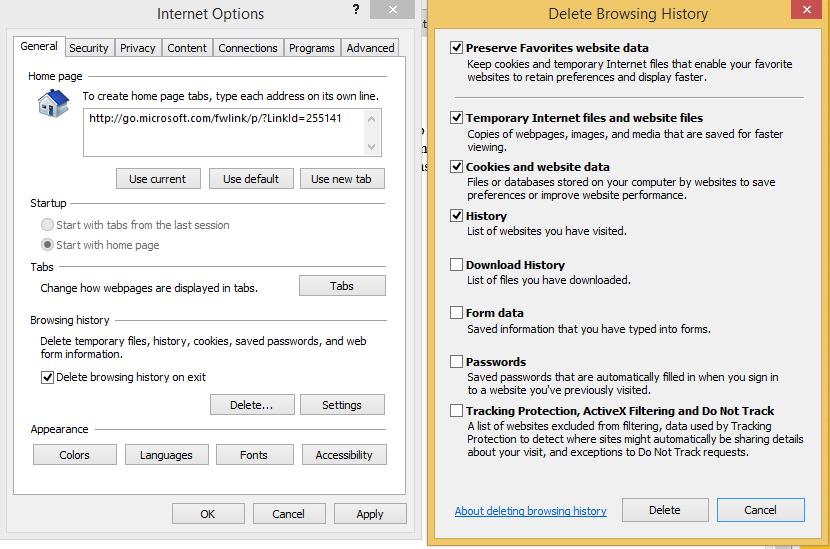
Part III – HTTP Message to TCP Segments

Retrieving Long HTML Documents

In our examples in lab 1, the documents retrieved have been simple and short HTML files. Let’s see what happens when we download a long HTML file. Do the following:

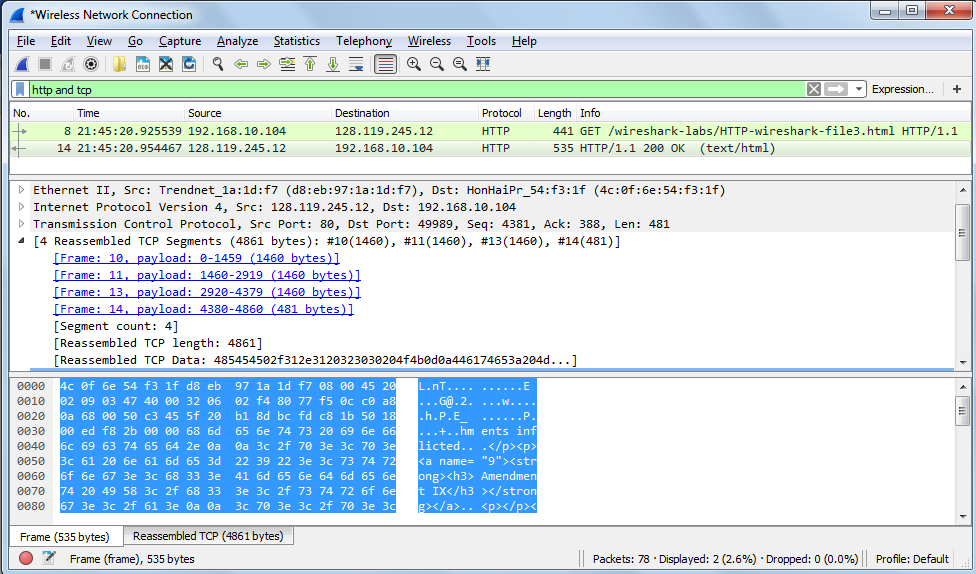
* Start up your web browser, and make sure your browser’s cache is cleared. (To do this under Firefox, select Tools->Clear Recent History, or for Internet Explorer, select Tools->Internet Options->Delete File (if you cannot find Tools in your Internet Explorer 11, click on the setting icon on the right top corner as shown in Figure 1, or for some version of internet explorer, you can move your curser to the top right corner, and the internet option charm will appear, you can click the “Internet Options” from there and clear the browsing history); these action will remove cached files form your browser’s cache.) Now do the following:
* Start up the Wireshark packet sniffer
* Enter the following URL into your browser  
  [**http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file3.html**](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 and tcp**” in the display-filter-specification window, so that only captured HTTP messages will be displayed.
* (*Note:* If you are unable to run Wireshark on a live network connection, you can use the http-ethereal-trace-3 packet trace to answer the questions below; see footnote 2. This trace file was gathered while performing the steps above on one of the author’s computers.)





**Figure 1 Delete History for Internet Explorer 11**

In the packet-listing window, you should see your HTTP GET message, followed by a multiple-packet response to your HTTP GET request. This multiple-packet response deserves a bit of explanation. Recall 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 segment. The single HTTP response message is thus broken into several pieces by TCP, with each piece being contained within a separate TCP segment. Each TCP segment is recorded as a separate packet by Wireshark, and the fact that the single HTTP response was fragmented across multiple TCP packets is indicated by an extra line of “XXX Reassembled TCP Segments …” listed under the Transmission Control Protocol.



***What to hand in***

Answer the following questions:

1. How many HTTP GET request messages were sent by your browser?
2. How many data-containing TCP segments were needed to carry the single HTTP response?
3. What is the length of each of these TCP segments?
4. What is the status code and phrase associated with the response to the HTTP GET request?
5. What is the IP address and port number used by gaia.cs.umass.edu to send the file? What is the IP address and TCP port number used by your client computer (source) to receive the file?
6. What are the sequence number and ACK number of the TCP segment containing the HTTP GET command?
7. Save the HTTP messages displayed. To do so, select *“Export Specified Packets”* from the Wireshark *File* command menu, and select “*All Packets”* and *“Displayed”*, Specify the location where you want to save the HTTP messages, type in a name and then click Save.

The Next Step

Many of the lab exercises will include a section entitled “The Next Step”. I will ask you to take “the next step” by exploring some concepts on your own. You might need to conduct some search in the online textbook or some Web search. Finishing these steps will increase your lab exercise grade, and will assist you in understanding the networking concepts.

1. Additional Exercises

1. Become familiar with the “expression,” “clear,” and “apply” buttons in the user interface. Display only the packets sent by your host. Do this by filtering IP addresses.
   1. Go to the Help menu, and skip through the man page for “Wireshark Filter”.
2. Display only the packets that have been received by your host. Do this by filtering IP addresses.
3. Display only the HTTP packets that have been sent by your host.
4. Additionally explore the various buttons and menus in the Wireshark interface.

2. The HTTP CONDITIONAL GET/response interaction

Most web browsers perform object caching and thus perform a conditional GET when retrieving an HTTP object. Before performing the steps below, make sure your browser’s cache is empty. (To do this under Firefox, select Tools->Clear Recent History, or for Internet Explorer, select Tools->Internet Options->Delete File; these action will remove cached files form your browser’s cache.) 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
* Enter the following URL into your browser  
  [**http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file2.html**](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 and tcp” 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 http-ethereal-trace-2 packet trace to answer the questions below; see footnote 1. This trace file was gathered while performing the steps above on one of the author’s computers. Again, 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.)

***What to hand in***

Answer the following questions:

1. 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?
2. Inspect the contents of the server response. Did the server explicitly return the contents of the file? How can you tell?
3. 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?
4. 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.
5. Save the HTTP messages displayed. To do so, select *“Export Specified Packets”* from the Wireshark *File* command menu, and select “*All Packets”* and *“Displayed”*. Specify the location where you want to save the HTTP messages, type in a name and then click Save.

1. 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. [↑](#footnote-ref-1)
2. Download the zip file <http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.zip> and extract the file http-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 http-ethereal-trace-1 trace file. The resulting display should look just like Figure 1. [↑](#footnote-ref-2)