

CSCI 466: Networks

Fall 2023

Wireshark Lab 1: HTTP and DNS

Due: Wednesday September 20th @ 11:59PM

Overview

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.

Getting Wireshark

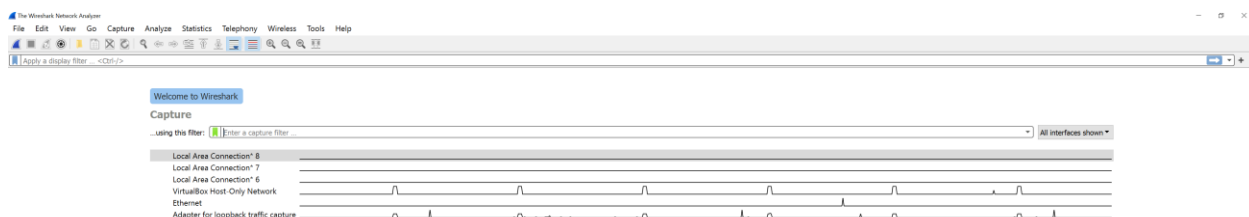
In order to run **Wireshark**, the packet sniffing we will use in this class, you will need to have access to a computer that supports both Wireshark and the *libpcap* or *WinPCap* packet capture library. The *libpcap* software will be installed for you, if it is not installed within your operating system, 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 several helpful hints and interesting tidbits of information, particularly if you have trouble installing or running Wireshark

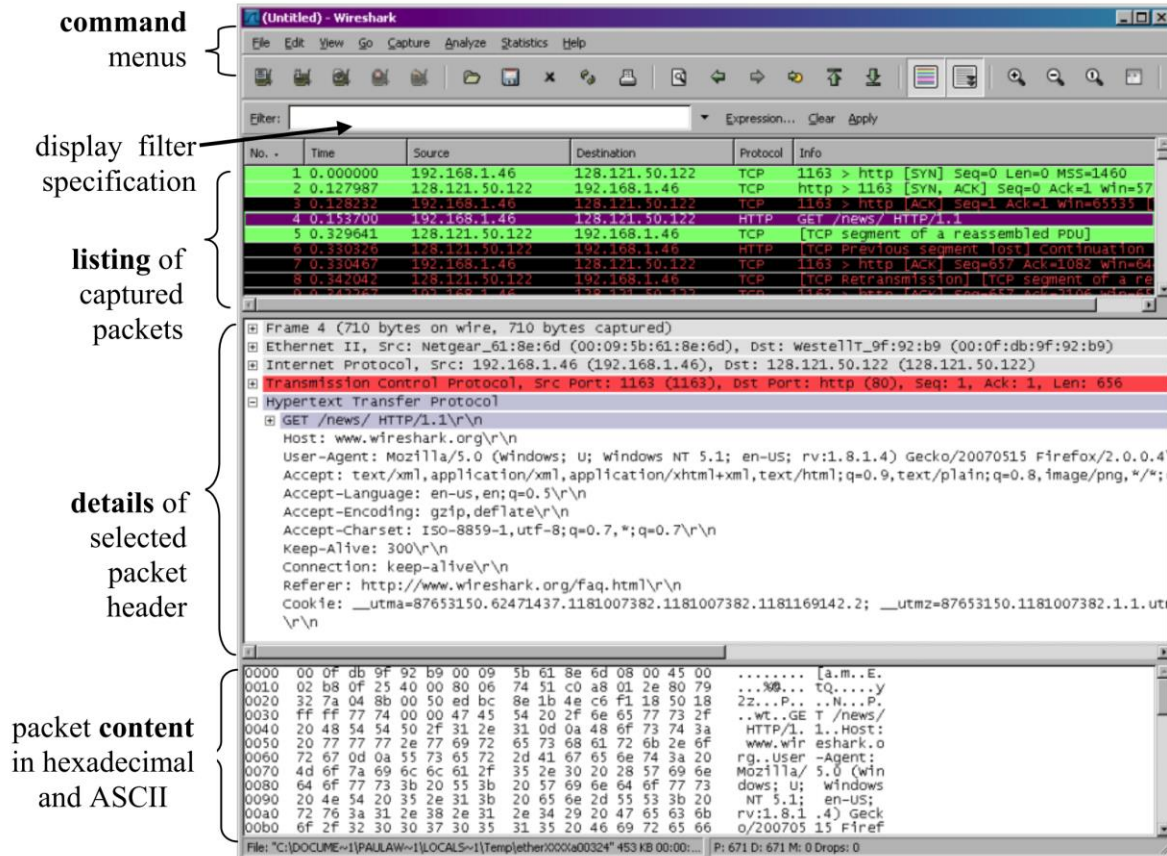
Running Wireshark

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.



Under the “Capture” section, there is a list of so-called interfaces. On my desktop, I do not have a Wi-Fi interface, but you probably will. All packets to/from this computer will pass through one of the interfaces. You will likely select ethernet (if you have a wired internet connection), or Wi-Fi (if you have a wireless internet connection) for your network captures in these labs.

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.



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 packetlisting 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 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.

Lab Tasks

Task 1 HTTP Traffic.

Restart packet capture and 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 web 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 frames containing these HTTP messages (as well as all other frames passing through your Ethernet adapter) will be captured by Wireshark.

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. 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.

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"). This will cause only HTTP message to be displayed in the packet-listing window.

- **Take a screenshot of your Wireshark screen that shows all HTTP traffic from your capture**

Task 2 HTTP Get Request.

Locate the HTTP GET request that was sent from your computer to the gaia.cs.umass.edu HTTP server. Make sure you are **not** looking at the GET request for favicon.ico. Answer the following questions:

- 2.1. What is the IP address of gaia.cs.umass.edu?
- 2.2. What is the IP address of your machine?
- 2.3 What was the destination port for this packet?
- 2.4 What version of HTTP is being used for the HTTP request?
- 2.5 Take a screenshot of the header fields of the HTTP GET request.
- 2.6 Before an HTTP request can be sent, a TCP connection must be established with the server. There is a series of 3 TCP packets that are sent that establish this connection before the HTTP request is sent. Find these three packets and take a screenshot.

Task 3 HTTP Response.

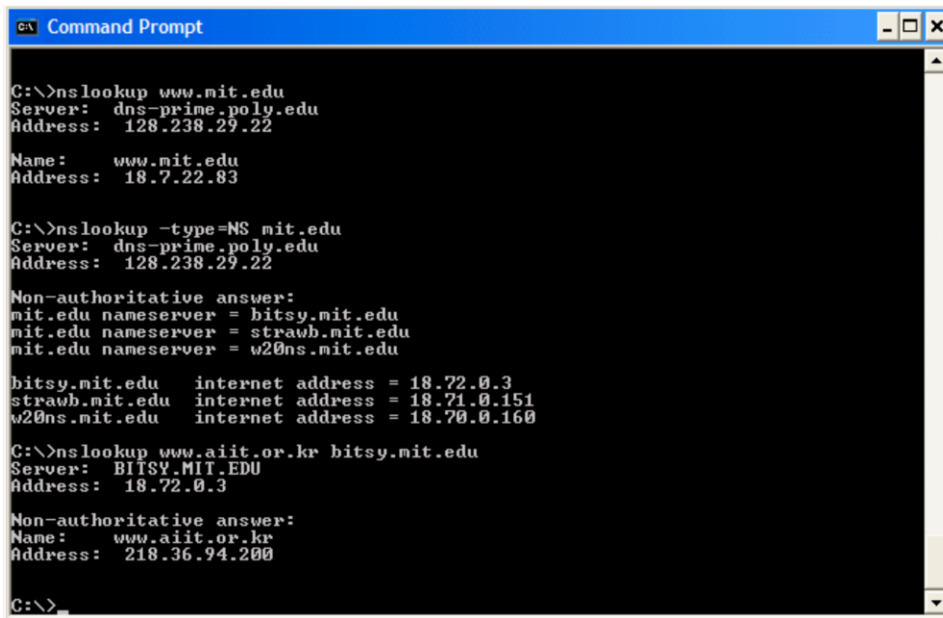
Locate the HTTP response that was sent from the gaia.cs.umass.edu HTTP server to your computer. Answer the following questions:

- 3.1 What is the response code?
- 3.2 What type of content was returned from this response message?
- 3.3 How long did it take from when the HTTP GET message was sent until the HTTP OK reply was received?
- 3.4 What version of HTTP is being used for the HTTP response?
- 3.5 Take a screenshot of the header fields of the HTTP response.

(I highly recommend finishing Tasks 1,2 and 3 before proceeding)

Task 4 nslookup.

In its most basic operation, the *nslookup* tool allows the host running the tool to query any specified DNS server for a DNS record. The queried DNS server can be a root DNS server, a top-level-domain DNS server, an authoritative DNS server, or an intermediate DNS server (see the textbook for definitions of these terms). To accomplish this task, *nslookup* sends a DNS query to the specified DNS server, receives a DNS reply from that same DNS server, and displays the result.



```
C:\>nslookup www.mit.edu
Server: dns-prime.poly.edu
Address: 128.238.29.22

Name: www.mit.edu
Address: 18.7.22.83

C:\>nslookup -type=NS mit.edu
Server: dns-prime.poly.edu
Address: 128.238.29.22

Non-authoritative answer:
mit.edu nameserver = bitsy.mit.edu
mit.edu nameserver = strawb.mit.edu
mit.edu nameserver = w20ns.mit.edu

bitsy.mit.edu internet address = 18.72.0.3
strawb.mit.edu internet address = 18.71.0.151
w20ns.mit.edu internet address = 18.70.0.160

C:\>nslookup www.aiit.or.kr bitsy.mit.edu
Server: BITSY.MIT.EDU
Address: 18.72.0.3

Non-authoritative answer:
Name: www.aiit.or.kr
Address: 218.36.94.200

C:\>
```

The above screenshot shows the results of three independent *nslookup* commands (displayed in the Windows Command Prompt). In this example, the client host is located on the campus of Polytechnic University in Brooklyn, where the default local DNS server is dns-prime.poly.edu. When running *nslookup*, if no DNS server is specified, then *nslookup* sends the query to the default DNS server, which in this case is dns-prime.poly.edu. Consider the first command:

```
nslookup www.mit.edu
```

In words, this command is saying “please send me the IP address for the host www.mit.edu”. As shown in the screenshot, the response from this command provides two pieces of information: (1) the name and IP address of the DNS server that provides the answer; and (2) the answer itself, which is the host name and IP address of www.mit.edu. Although the response came from the local DNS server at Polytechnic University, it is quite possible that this local DNS server iteratively contacted several other DNS servers to get the answer, as described in Section 2.4 of the textbook.

Now consider the second command:

```
nslookup -type=NS mit.edu
```

In this example, we have provided the option “-type=NS” and the domain “mit.edu”. This causes *nslookup* to send a query for a type-NS record to the default local DNS server. In words, the query is saying, “please send me the host names of the authoritative DNS for mit.edu”. (When the -type option is not used, *nslookup* uses the default, which is to query for type A records.) The answer, displayed in the above screenshot, first indicates the DNS server that is providing the answer (which is the default local DNS server) along with three MIT nameservers. Each of these servers is indeed an authoritative DNS server for the hosts on the MIT campus. However, *nslookup* also indicates that the answer is “non-authoritative,” meaning that this answer came from the cache of some server rather than from an authoritative MIT DNS server. Finally, the answer also includes the IP addresses of the authoritative DNS servers at MIT. (Even though the type-NS query generated by *nslookup* did not explicitly ask for the IP addresses, the local DNS server returned these “for free” and *nslookup* displays the result.) Now finally consider the third command:

```
nslookup www.aiit.or.kr bitsy.mit.edu
```

In this example, we indicate that we want to the query sent to the DNS server bitsy.mit.edu rather than to the default DNS server (dns-prime.poly.edu). Thus, the query and reply transaction takes place directly between our

querying host and bitsy.mit.edu. In this example, the DNS server bitsy.mit.edu provides the IP address of the host www.aiit.or.kr, which is a web server at the Advanced Institute of Information Technology (in Korea).

Now that we have gone through a few illustrative examples, you are perhaps wondering about the general syntax of *nslookup* commands. The syntax is:

```
nslookup -option1 -option2 host-to-find dns-server
```

In general, *nslookup* can be run with zero, one, two or more options. And as we have seen in the above examples, the dns-server is optional as well; if it is not supplied, the query is sent to the default local DNS server.

Now that we have provided an overview of *nslookup*, it is time for you to test drive it yourself. Do the following (and write down the results):

Now answer the following questions (wireshark is not needed for this task)

- 4.1 Run *nslookup* on montana.edu. What is the IP address of that server? Take a screenshot of this response.
- 4.2 Run that IP through <https://who.is/> and provide a screenshot of the registration information for that IP address. **Note:** If who.is is giving you no response, try another online whois service website.
- 4.3 Run *nslookup -type=MX* followed by a URL of your choice. What does the -type=MX flag mean? What was the answer for the DNS query you requested?

Task 5: DNS Traffic

ipconfig is a helpful tool for identifying your IP address. In a command prompt/terminal, run *ipconfig* and identify your IPv4 address (on Linux, you can use *ifconfig*). Now that we are familiar with *nslookup*, we're ready to get down to some serious business. Let's first capture the DNS packets that are generated by ordinary Web-surfing activity. If you having issues getting your IP on Mac, try *ipconfig getifaddr en0*

1. Run the command *ipconfig /flushdns*. This will remove any DNS answers you have stored in your local cache (*you might need to google the correct command for macs*)
2. Open Wireshark and enter "*ip.addr == your_IP_address*" into the filter, where you obtain *your_IP_address* with *ipconfig*. This filter removes all packets that neither originate nor are destined to your host.
3. Start packet capture in Wireshark
4. With your browser, visit the Web page: <http://www.ietf.org>
5. Stop packet capture

Now answer the following questions:

- 5.1. Locate the DNS query and response messages for www.ietf.org. and take a screenshot of your screen
- 5.2. Are the DNS query and response messages sent over UDP or TCP?
- 5.3. What is the destination port for the DNS query?

- 5.4. To what IP address is the DNS query message sent?
- 5.5. Examine the DNS response message. How many “answers” are provided?
- 5.6. Based on your DNS response message, what is an IP address for <http://www.ietf.org>?

Submission

You will submit a lab report to D2L. This lab report will contain your answers for all the tasks, and any screenshots that were needed. Save it as a PDF and submit it to D2L.