**LAB NO. 5**

**INTRODUCTORY WIRESHARK LAB**



**DATA COMMUNICATION AND COMPUTER NETWORKS LAB**

Submitted by:

Name : **Naveed Ahmad**

Reg no**. : 22PWCSE2165**

ClassSection **: B**

24 April 2025

**Submitted to:** **Dr. Yasir Saleem Afridi**

**Department of Computer Systems Engineering**

**University of Engineering and Technology, Peshawar**

**LAB 05 INTRODUCTORY WIRESHARK LAB**

**Objectives:**

Introduction to Wireshark

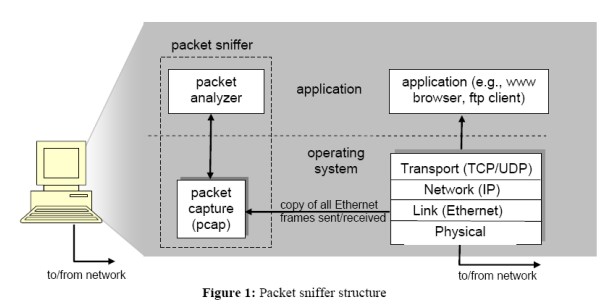
**Introduction:**

The 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 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 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 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. 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, Mac, and Linux/Unix computer. 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/>manual pages website link [http://www.wireshark.org/docs/man-pages,](http://www.wireshark.org/docs/man-pages) and a detailed FAQ Frequently asked Questions [http://www.wireshark.org/faq.html,](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 and SLIP), 802.11 wireless LANs, and many other link-layer technologies (if the OS on which it's running allows Wireshark to do so).

**Getting Started:**

In order to run Wireshark, 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 a number of helpful hints and interesting tidbits of information, particularly if you have trouble installing or running Wireshark.

**Running Wireshark**:

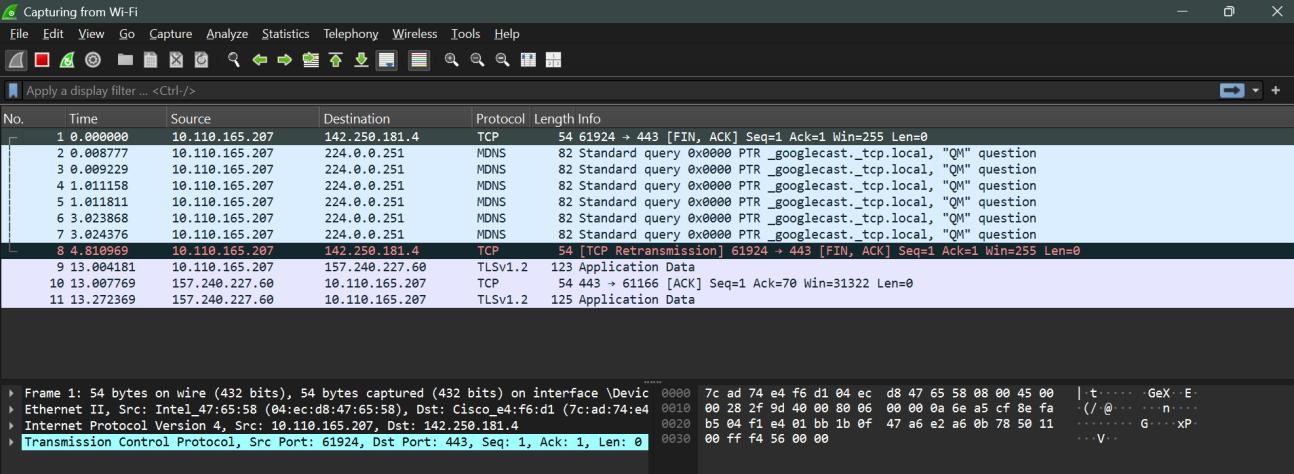
When you run the Wireshark program, you’ll get a startup screen that looks something like the screen below. Different versions of Wireshark will have different startup screens – so don’t panic if yours doesn’t look exactly like the screen below! The Wireshark documentation states “As Wireshark runs on 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*



There’s not much interesting on this screen. But note that under the Capture section, there is a list of so-called interfaces. The computer we’re taking these screenshots from has just one real interface – “Wi-Fi en0,” 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 want to capture packets. Let’s take Wireshark out for a spin! If you click on one of the available interface 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

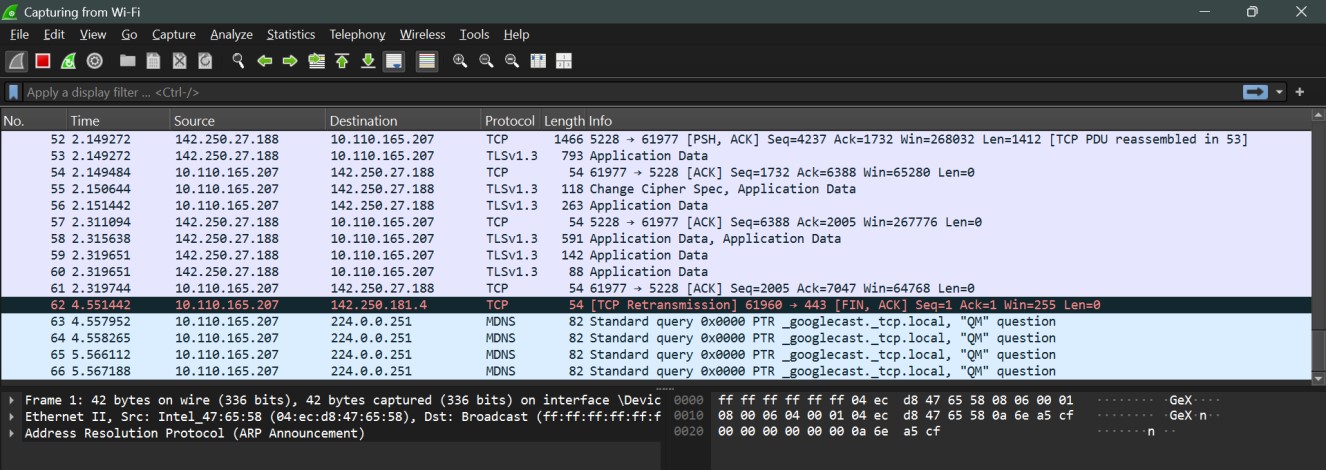


*Figure 3*

**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. Indeed, I recommend that you do this first lab on a computer that has a wired Ethernet connection, rather than just a wireless connection. 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 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, as shown in Figure below.



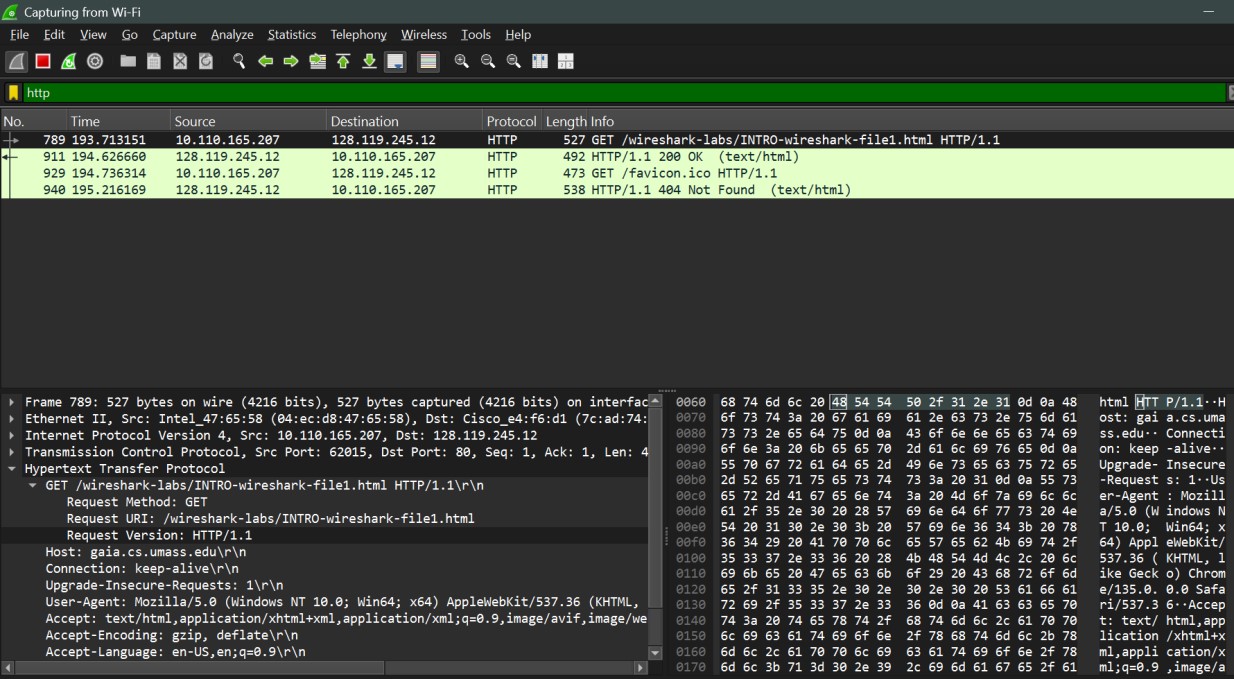
*Figure 4*

1. 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. Click on Start for the on which you want to begin packet capture (in the case, the Gigabit network Connection). Packet capture will now begin - Wireshark is now capturing all packets being sent/received from/by your computer
2. 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, 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
3. While Wireshark is running, enter the URL: [http://gaia.cs.umass.edu/wiresharklabs/INTRO-wireshark-file1.html a](http://gaia.cs.umass.edu/wireshark-labs/INTRO-wireshark-file1.html)nd 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 (as well as all other frames passing through your Ethernet adapter) will be captured by Wireshark.



*Figure 5*

1. 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. For now, you should just be aware that there is often much more going on than “meet’s the eye”!
2. 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.
3. 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 8 | P a g e the Wireshark window (see Figure 3) 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 window2. By clicking on ‘+’ 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).
4. Exit Wireshark Congratulations! You’ve now completed the first lab.



*Figure 6*

**CSE 303L: Data Communication and Computer Networks**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Demonstration of Concepts** | **Poor (Does not meet expectation**  **(1))**    The student failed to demonstrate a clear understanding of the assignment concepts | **Fair (Meet**  **Expectation**  **(2-3))**    The student demonstrated a clear understanding of some of the assignment concepts | **Good (Exceeds**  **Expectation (4-5)**    The student demonstrated a clear understanding of the assignment concepts | **Score**      **30%** |
| **Accuracy** | The student misconfigured enough network settings that the lab computer couldn't function properly on the network | The student configured enough network settings that the lab computer partially functioned on the network | The student configured the network settings that the lab computer fully functioned on the network | **30%** |
| **Following**  **Directions** | The student clearly failed to follow the verbal and written instructions to successfully complete the lab | The student failed to follow the some of the verbal and written instructions to successfully complete all requirements of the lab | The student followed the verbal and written instructions to successfully complete requirements of the lab | **20%** |
| **Time Utilization** | The student failed to complete even part of the lab in the allotted amount of time | The student failed to complete the entire lab in the allotted amount of time | The student completed the lab in its entirety in the al | **20%** |