**ECE 5600 Project (Phase 1)**

**OBJECTIVES**

1. Become familiar with Wireshark and Linux OS.

2. Learn how to capture traffic and use C/C++ programming in a Linux OS environment.

3. Learn how to customize a Wireshark filter to capture and analyze data frames.

**REPORT REQUIREMENTS**

1. Read the lab report requirements from the syllabus carefully.

2. Do not share your code or data with other students.

3. Be sure to add your partner’s name to the report so that you can submit the report as a team. Each team only needs to submit one file.

4. Screenshot the Wireshark window (with detailed output text in the bottom window) and your command window’s output after you successfully capture the frames and compare the two frames.

5. Online submissions only. Please submit your report in PDF format titled **lastname\_firstname\_lab1.pdf** and any other necessary files including the code files in a .zip or .rar folder titled **lastname\_firstname\_lab1.zip**.

**BACKGROUND**

During the course of this semester, we will implement TCP/IP protocol stacks using Linux. Linux allows us to configure a socket in packet mode so we can send and receive raw packets over Ethernet. We will be able to build up our own network stack without interfering with the normal operation of the network connection. The IP addresses of computers in the designed lab room have the following format:

*129.123. x. xxx*

You can use the command **ifconfig** to get the IP address.

**PRE-LAB READING**

Chapter 4 P 281-286, P 465-469.

**PRELIMINARY**

In order to understand Project 1, you need to have some knowledge of Ethernet frames. Ethernet frame formatting is shown in Figure 1. The preamble is omitted since it is used for synchronization.

Graphical user interface, text

Description automatically generated with medium confidence

**Figure 1** **Ethernet frame formatting.**

The MAC header contains three parts with a total length of 14 bytes: Destination MAC address (6 bytes), Source MAC address (6 bytes), and frame type (2 bytes). Here, frame type (Ether Type) indicates the payload data type. Two common types are 0x0800 (IP) and 0x0806 (ARP).

**COMPUTER SETUP**

1. Log in to one of the computers in the lab using your A-number and password.

If you work outside the lab, please read **remoteREADME.pdf**.

1. You may use **sudo** to run some commands. The password is your A-number’s password. **Be careful to use the sudo command.**
2. Get the C++ code for Frame I/O from the course website. Read **frameio.h** and **frameio.cpp** carefully; make sure you understand them for future course projects.
3. Read the example code example1.cpp. This example gives you a template for using Frame I/O.
4. Run **which wireshark** to ensure Wireshark software is installed on that computer.

**PRACTICE**

For practice run example1.cpp first:

1. Find the network interface name using the ifconfig command in your terminal.
2. Use your network interface name to replace the name in example1.cpp. The old name is "eth1" in the **main** function.
3. Compile the code by running g++. Run "**g++ example1.cpp frameio2.cpp util.cpp -lpthread -o out**" from the terminal. You can search online for a better understanding of the g++ compiler.
4. Run the executable “out” you created using **sudo ./out** in the terminal. *Hint: Remember sudo enables admin permissions.*

You should see this example code capturing IP and ARP frames. *Hint: If there is no traffic information, make sure you change the interface name. If there is still no output, ping your machine from another computer.*

**PROJECT PROCEDURE**

The following steps outline this project:

1. Write a program to read an incoming packet (either IP packet or ARP packet) and print the first 42 bytes in “%02x” format to the terminal. Put a space between each byte and add a new-line character (\n) after the 22nd and 42nd bytes for clarity. Separate each set of 42 bytes with a blank line.

2. Run your program and verify that it monitors network traffic. *Hint: Network traffic in a public network will be heavy. Use a filter on Wireshark to find your printed packet.*

3. Run Wireshark. *Hint: Use* **sudo /user/local/bin/wireshark** *to run Wireshark with admin permissions at terminal Konsole.*

4. Start capturing traffic (the device to monitor is probably “**eno1**”; you should check it using **ifconfig** before experimenting with Wireshark). From another machine, ping your IP address. Observe the packet’s transmission over the network. There are many unrelated frames captured in Wireshark; you can define a capture filter in **Capture Options** or in the Filter bar. *Hint: search online for tips for capture filtering in Wireshark.*

5. Stop capturing traffic in Wireshark. **Ctrl+C** can be used to terminate your Wireshark program. Use Wireshark to find an IP/ARP packet **containing your IP address**. Select the IP/ARP packet with your mouse and view the raw data in the bottom window. Screenshot the Wireshark window showing this data for your report. *Hint: Centos has the screenshot tool.*

6. Locate the **corresponding packet** in your program’s output. Screenshot the terminal with the corresponding packet highlighted for your report.

**Hints for Project 1**

You need to output the MAC header (14 bytes) and the first 28 bytes from the payload in Project 1.

For your program, you can borrow code pieces from example1.cpp.

1. Define a frameio structure.
2. Open the interface.
3. Use a while loop to receive frames. If the frame length is less than 42, discard it.
4. Print the first 42 bytes with a space between each byte, and add a new line after 22 bytes for extra clarity. Add a blank line between each set of 42 bytes. *Hint: use \n.*