CS 254 – Computer Networks
Lab 6 – Ethernet Packet Traces

The purpose of this lab is to help you understand Ethernet frames and network traffic through the analysis of packet trace files. You will also learn how to generate your own packet trace files.

Section 1: Analyzing Packet Trace Files Using tcpdump

You will be using Ubuntu Linux on a virtual machine for this lab. Boot the virtual machine by starting up VMWare Player and then browsing to the **NetworkLab** virtual machine on the C: drive. Sign on to the virtual machine using the following credentials:

Username: **csstudent**
Password: **abc123**

Step 1.1 – Open a terminal window on the virtual machine. Use the **man** command to display the help for the **tcpdump** and **tshark** commands. Use the arrow keys and Pg-Up/Pg-Dn keys to navigate the window. Type the letter ‘q’ to quit viewing the help.

Question: When using **tcpdump**, what does the –r flag do?

____

Question: When using **tcpdump**, what does the –x flag do?

____

Question: When using **tcpdump**, what does the –e flag do?

____

Question: When using **tcpdump**, what does the –n flag do?

____

Question: Construct a command using these switches to read a packet trace file with the following constraints:

- Use the **tcpdump** command
- The trace file to read is named **icmp.trace**
- Don’t convert host addresses to names
- Print hex/ASCII representation of packet/frame data
- Print link-level headers

____

Step 1.2 – In your local directory (**/home/csstudent**), there is a trace file named **icmp.trace** provided for you to analyze. Enter the command you just came up with in the previous question to view the Ethernet frames that have been provided to you in the trace file. If the command doesn't appear to function properly, refer back to the man page for the **tcpdump** command to adjust it.

Step 1.3 – Enter your command from the previous step into the system. If you performed the operation successfully, you will see the frames as they were captured previously. To allow for you to see the output one screen at a time, use the vertical bar character to "pipe" your command's output through **less**. For example:

```
[csstudent@NetworkLab]$ ls –l | less
```

This would display a long directory listing, one screen at a time. Use the "space-bar" to forward the display one screen, or the "Enter" key to forward the display one line at a time. Type "q" to quit viewing the output.

Question: Did your command work? Give the command you used to view the dump.

____

By looking at the output from the command, answer the following questions:

- How many frames were captured? _____________ (I know, this is tedious.)
- Locate the first frame in the trace that is an ICMP echo request. Which frame (number) is this? ______________
- What is the MAC address of the computer sending the ICMP echo request frame? _______________
- What is the MAC address of the recipient of the ICMP echo request frame? ______________
- How do you think the MAC address of the recipient is discovered? (How did you find the MAC address of a remote machine in Lab 1?)

  ____

Step 1.4 – Now we will use the **tshark** command to view the same file. This utility performs a similar function to **tcpdump**, but can represent the frame data in a different format.

Question: What **tshark** switches are used to perform the following:

- Disable network object name resolution: _____________________
- Print hex and ASCII dump of the packet data: _________________
- View the packet details: ______________________
- Identify the trace file that stores the results: ____________________

(Use the **man** page to obtain the correct switch values.)

Question: What **tshark** command is used to view the same packet trace file as viewed in step 1.3? (Use the switches identified in the last question.)

____

Question: What is the destination MAC address of the first packet as displayed by **tshark**? ______________________________________

Question: What is the frame length of an echo request as displayed by **tshark**? (First find the frame.) ____________________________________

Question: What is the hex value of the Ethernet frame type for the IP frames? (The frame found for the previous question is an IP frame.)

____

Section 2: Capturing Network Frames into a Trace File (IGNORE/SKIP THIS SECTION!)
In this section, you will generate some network traffic of your own and capture the frames into a trace file. You will then be able to use the trace file to analyze traffic.

Step 2.1 – Construct a command to use the **tcpdump** utility that captures frames to a trace file named **netlab.trace**. *(We only want packets that originate or are destined to our machine, and we should limit the trace file to a maximum of 250 frames.)*

Question: Enter the command you came up with:

____

Start the capture by entering the command into the system.

After you have started the capture switch to a new terminal window and issue a ping to one of the local servers, such as **kant** or **hegel**. After about 10 or 12 seconds, switch back to the other terminal window and stop the capture by pressing **Ctrl-C**.

Step 2.2 – Analyze your trace file to determine what has happened on the network during the execution of the ping command. Use the **tshark** command, but this time remove the switch to view the packet details.

Question: What is the hex value of the Ethernet frame type for the ARP frames?

____

Question: What is the hex value of the Ethernet frame type for the IP frames?

____

Question: How many ICMP requests did you receive during your capture window? ______________

Question: How many ICMP replies did you receive during your capture window? ______________

Step 2.3 – Using the command from step 2.1, capture a trace file named **netlab2.trace**. This time exclude the option to limit packets to your current host. You will now be capturing all packets received from the local network segment.

Identify three different types of network traffic in this trace file. For each one, also identify the source and destination MAC addresses.

____

____

____

Section 3: Using the wireshark Tool

In the previous three sections, you were able to read a trace file, create a trace file, and pull specific information about Ethernet frames from your analysis of the trace files. Although the command line tool is very functional for providing this analysis, the **wireshark** tool provides a graphical user interface that is much easier to use and has some additional features that are quite powerful.

Step 3.1 – Start the **wireshark** application by typing **wireshark** at a command prompt in your terminal window. A window should appear in a traditional GUI interface form. Select **File -> Open** from the menu, and open **icmp.trace** in **/home/csstudent**. Examine the dump file, paying particular attention to the middle and bottom panes. The middle pane will parse the frame into a human readable format (you may need to expand items in the middle frame to see details), and the bottom pane will show the hex and ASCII representation of the frame.

Use **wireshark** to answer the questions previously posed in Step 1.3:

- How many frames were captured? _____________
- Locate the first frame in the trace that is an ICMP echo request. What frame number is this? ______________
- What is the MAC address of the computer sending the ICMP echo request frame? ______________
- What is the MAC address of the recipient of the ICMP echo request frame? ____________
- What is the IP address of the computer sending the ICMP echo request frame? ______________

- What is the IP address of the recipient of the ICMP echo request frame? ____________

Step 3.2 – With the **wireshark** tool, you are able to capture and analyze real-time packet traces without the need to save them to intermediate trace files. Before proceeding you will need to run wireshark as *root*. Exit wireshark then enter su from the command line to gain root access. Your instructor will need to supply the root password. Before proceeding open a second terminal window. In the first terminal window return to the **wireshark** tool, select **Capture -> Options** from the menu bar, specify **eth0** as the interface, and then click **Start**. You will then be capturing packets until you click **Stop**. At this point you are presented with the trace analysis window, and you can view the real-time data.

Start a capture. Go to the second terminal window and generate ICMP traffic by ping'ing kant. After around 10 seconds terminate the ping with **Ctrl-C**. Return to the trace analysis window and stop the capture. Now examine the traffic you captured.

Question: How many ICMP packets did you capture?

____

Question: What is the frame size of an Echo request as reported by **wireshark**?

____

Question: Write down any additional frame types you may have captured in this way.

____

Step 3.3 - Use **wireshark** to analyze a packet trace named **email.trace** that contains several e-mails.

Note: You can isolate just the email traffic by setting the filter to **smtp** and click apply.

Answer the following questions:

- How many emails messages were sent in this trace (simply count the number of **EHLO** commands)? _________________

- For each email identify the sender, the recipient, the subject, and the text of the message:
  (Hint: right-click on an **EHLO** command and select **Follow TCP stream** for a readable form of emails.)

  ____
  ____
  ____
  ____
  ____
  ____
  ____
  ____

- Which of the emails has an attachment? ______________________

- What is the name of the attachment? _________________________

For extra credit: Go back to the file **icmp.trace** and find the password for asmith.