Wireshark lab project 1

Due Sep 25, 2021 by 11:59pm **Points** 100 **Submitting** a file upload
File Types java **Attempts** 1 **Allowed Attempts** 10
Available Sep 14, 2021 at 5pm - Sep 25, 2021 at 11:59pm 11 days

This assignment was locked Sep 25, 2021 at 11:59pm.

CS 352 Wireshark Lab 1

Due date: Saturday, Sept. 25th, 11:59PM

Background

Wireshark (https://www.wireshark.org/), tshark, and **tcpdump (https://www.tcpdump.org)** are programs that use the **pcap library (https://en.wikipedia.org/wiki/Pcap/)**. Pcap (for Packet CAPture) is a library which works with the operation system to make copies of packets inside the operating system and deliver them to user-level programs for analysis. The **pcap library (https://en.wikipedia.org/wiki/Pcap)**, originally written in C, has been ported to numerous languages, including Java, Python, and Go.

Updates:

2021-09-22: The number of UDP packets in small.pcap is 28, not 34. The sample output text file has been updated.

Assignment

The goal of this assignment is for you to familiarize yourself with the Java version of pcap. You will create a small program that counts packets and keeps track of the sizes of the packets. Following the instructions below, you will extended a program called App.java to count the packet types. The goal is to make sure you have the development environment working for the Wireshark 2 project.

You will work individually on this assignment.

App.java accepts a file in the pcap format, which is generated with the tcpdump, wireshark, or tshark commands. The report is an example of simple packet analysis, for example, detecting possible port-scanners and attacks on a set of local IP addresses.
Given a pcap file as input, App.java program prints out the following contents. The contents of each section will be:

1. The number of all packets, which includes all types.
2. The number of UDP packets seen in the file.
3. The number of TCP packets seen in the file.
4. The total bandwidth of the packet trace. That is, the total number of bytes seen divided by the total elapsed time. The bandwidth includes all the headers as well as data.

The header strings for each section are:

1. "Total number of packets,<COUNT>", where <count> is an integer.
2. "Total number of UDP packets,<COUNT>", where count is an integer
3. "Total number of TCP packets,<COUNT>", where count is an integer
4. "Total bandwidth of the packet trace in Mbps,<Bandwidth>", where bandwidth is a floating point number of the megabits per second of the trace.

How to get started:

Your App.java must work on the ilab machines. **If it does not work on the ilab machines, it's wrong.** You can log into the ilab machines using the **instructions at this link. (https://resources.cs.rutgers.edu/docs/computer-systems/student-systems/)**

The Pcap4j examples use **Maven (https://en.wikipedia.org/wiki/Apache_Maven)** as the Java build environment. Follow the instructions in the attached tutorial to get the Pcap4j examples working using Maven on the ilab machines. You will modify the App.java code (included in the attachments) and hand that in.

▶ strategy App.java to generate the report is to write a main loop that reads in all the packets one at a time, extracts the needed information from them, and then sums up the types (TCP or UDP).

In order to read the pcap file and count the packets, look at the example source code -- see the tutorial an attachments.

File Attachments:

1. A skeleton App.java file. This file opens a pcap file and prints out the total number of packets. You must modify this file to report the correct statistics.
2. Tutorial for starting project.

3. 2 test pcap files (small.pcap , http.pcap)
4. 2 example outputs on the pcap files. Your program should match the counts in these files. (small_pcap_out.txt , http_pcap_out.txt)

Use the following commands to run and test the code. (Put the pcap files in the compiled folder)

```
java -jar uber-pcap-1.1.0.jar small.pcap
```

```
java -jar uber-pcap-1.1.0.jar http.pcap
```

What to Hand in:

You must hand in a single file called App.java, which is your program.

Do not hand any zip files, tar files, or additional packaging (e.g. Maven pom files, etc).

Grading:

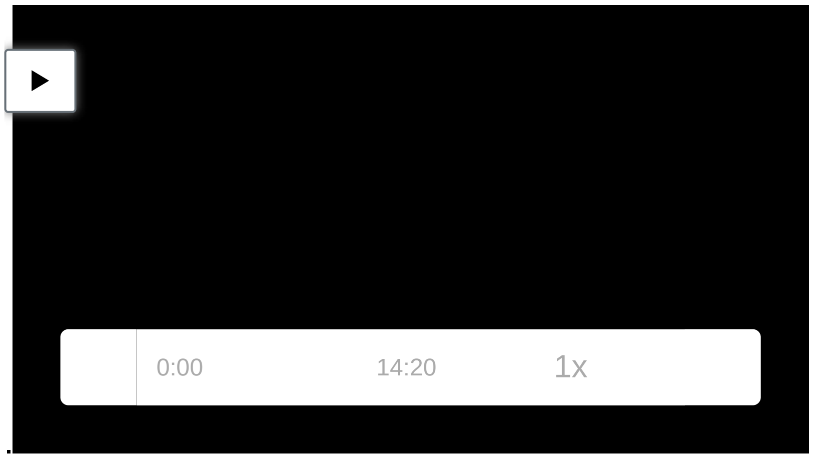
You handed in a working App.java code:

1. Compile success: 20%
2. Correct number of all packets: 20%
3. Correct number of UDP packets: 20%
4. Correct number of TCP packets: 20%
5. Correct total bandwidth of the packet trace: 20%

Files for this assignment:

[PDF tutorial on installing pcap4j library and example App.java skeleton code on the ilab machines.]

Video tutorial how to install pcap4j on the ilab machine:



.

Example skeleton code for App.java.
(https://rutgers.instructure.com/courses/133632/files/19033367/download?download_frd=1)

The follow files should help you test your code. There are 2 pcap files and the output of what your App.java should print out for these files.

First example pcap file: **small.pcap**
(https://rutgers.instructure.com/courses/133632/files/19033779/download?download_frd=1)

Second example pcap file: **http.pcap**
(https://rutgers.instructure.com/courses/133632/files/19034331/download?download_frd=1)

Output for small.pcap: **small_pcap.txt**
(https://rutgers.instructure.com/courses/133632/files/19230879/download?download_frd=1)

Output for http.pcap: **http_pcap.txt**
(https://rutgers.instructure.com/courses/133632/files/19034359/download?download_frd=1)