Programing Project #4 - Protocol Analyzer

Overview

In this project you will Protocol Analyzer, giving you an opportunity to review three of the four layers studied this semester. This assignment is due Sunday, December 5th.

For full credit your submission must have the ability to read a pcap packet capture file and compute the following information:

1. The total number of packets in the file.
2. The number of packets matching each of the listed protocols as well as identifying largest, smallest, and average datagram/packet/frame size of each type. Each length is the length of the header at that layer plus the payload. Protocols you must recognize are:
    a. Link layer: Ethernet II, ARP, Other-Link-Layer
    b. Network layer (Ethernet II only): IPv4, IPv6, Other-Network-Layer
    c. Transport Layer (IPv4 only): TCP, UDP, ICMP, Other-Transport-Layer
3. The number of unique instances of each of the following types as well as listing the values in human-readable format:
    a. Source and Destination Mac address (all frames).
    b. Source and Destination IPv4 addresses in Ethernet II frames.
    c. Source and Destination UDP port numbers in IPv4 packets.
    d. Source and Destination TCP port numbers in IPv4 packets.
4. Within the TCP protocol (IPv4 packets only)
    a. The number of packets with the TCP-SYN bit set in IPv4 packets.
    b. The number of packets with the TCP-FIN bit set in IPv4 packet.

Details

Skeleton code and the pcap library:

Programs like tcpdump and wireshark use a common packet capture library (pcap). Using the pcap library allows users to save packet captures in a common file format that can be shared with and analyzed by a wide range of tools. You will use the pcap library to read capture files, parsing the packets and identifying protocol information from the link, network, transport, and application layers.

At the heart of the pcap library is the call-back function pk_processor(). As the library reads through the file this function is called once for each packet. Each time three parameters are passed (1) a generic, user-defined pointer, (2) a pointer to the pcap header for the packet and (3) a pointer to the raw packet.

I've created a skeleton program that reads a dump file and counts the number of packets in the file. You don't have to use my sample code, but if you do you won't need to worry about the mechanics of the pcap library. You need only process the packets as they are passed to pk_processor (I've also supplied a simple class that you can use to accumulate statistics called resultsC).

Note that I would encourage you to use the system header files to aid in the decoding of the packets, however you may not use the pcap library filter functions (those would basically do all the work for you…).

Command line options:

The binary must be called "packetstats". Your program must accept the following command line arguments:

- -f <filename>, where filename is the pcap file to process.
- -d #, Turn on debugging messages. The is a digit that indicates how verbose the messages should be.
- -m, list unique mac addresses.
- -a, list unique IPv4 addresses.
- -t, list unique TCP port numbers.
- -u, list unique UDP port numbers.

Output Format:

To facilitate grading (since the program is not due until the last week of class, I need to automate grading as much as possible), each statistic must be displayed on a line by itself in the format "Identifier = value". The whitespace before and after the equality is important.

For the packet counts and sizes the Identifier must be "Total X", "Min X", Max X" and "Average X", where X is one of "Ethernet", "ARP", "OtherLink", "IPv4", IPv6", "OtherNetwork", "TCP", "UDP" and "OtherTransport".

For the protocol detail counts the Identifier must be "Unique X", where X is one of "srcMac", "dstMac", "srcIPv4", "dstIPv4", "srcUDP", "dstUDP", "srcTCP" and "dstTCP".

For the flag and total packet counts the Identifier must be "synCount", "finCount", "totalPacketCount".

If one or more list is requested you should print a header identifying the list, followed by each of the unique values on a line by itself. Order does

not matter. Note that for ICMP and the transport layer protocols you only need to decode IPv4 packets. IPv6 packets need to be identified, but you don't need to count IPv6 addresses or decode the transport layer headers.

If you use the supplied statistics and results classes, the format should be correct. If there are questions or ambiguities, please run my sample code (~promig3/pub/bin/packetstats) and duplicate the output format there.

Submission

This assignment is due by the end of the day on Sunday, December 5th. There will be a 5% per day penalty for late submissions with a maximum late penalty of 40% (**to allow us time to grade no submissions can be accepted after December 8th**).

You should submit a single tarball of a single directory containing a makefile, a README.txt with your name and any information I need to compile the program and the source files needed to build your program. The single directory must be named with your username. The grader should not need to do anything other than untar your files, cd into your direc-tory, type make and begin testing.

Do not include any core, object, or binary files in the tarball. The Makefile provided with the sample code includes a target named submit that will create the tarball in the format we are looking for. All you need to do is:

$> make submit

In addition to functionality, you will be graded on the quality of the code, including readability, comments, and the use of proper programing practices.