Program 4
Packet Processing

Prof. Kredo

Due: By 23:59 Friday, December 12

Introduction

In this program you will accomplish several goals:

- Utilize the pcap library to process raw network packets
- Categorize and print information about packets in a trace file or from a live capture

All programming assignments must be done in pairs.

Requirements

This assignment has you write a program to categorize packets and print select information about those packets. Your program reads packets from a pcap trace file or from a live capture and produces the following output:

- The Ethernet source and destination address and the decoded Ethernet type.
- The source and destination IP (version 4 and 6) addresses, if appropriate.
- The source and destination ports for UDP and TCP.

If you receive a packet with a protocol not specified above, then print the multiplexing key (type or protocol value) and stop. For example, if you receive an IP packet where TCP or UDP is not the encapsulated protocol, then just print the protocol value.

When printing addresses, let the OS do the work for you. Use the functions `inet_ntop(3)` and `ether_ntoa(3)`.

Your program will use the pcap(3) library, which collects packets and provides them to you as an array of bytes. You have two sources for your packets: a saved capture file or a live capture of traffic off the network. A sample program is provided on Learn for you to start with that utilizes both these sources. If you provide a filename as a command line argument, then the capture file is used. If you do not use a command line argument, then a live capture is used. You will need to run the program with root privileges to capture live packets. You may use Wireshark to capture traces (save the packets in pcap format) to use as test input to your program or you may download example traces from http://wiki.wireshark.org/SampleCaptures. When working with traces, ensure they include the full Ethernet header (no not capture on the `any` interface).

You need to link against the libpcap library (`-lpcap`) to build your executable.

There are many sources for the packet formats you will use in this program, but the authoritative sources are: IEEE 802.3 standard for Ethernet, RFC 791 for IPv4, RFC 2460 for IPv6, RFC 793 for TCP, and RFC 768 for UDP.

Additional requirements for this program include:

- You must include a Makefile with your submission that compiles the program upon the command `make`.
- Your program must be called `packets` and take one optional argument, the filename for the packet trace.
- Your program must produce output using an identical format to the example below.
- Your programs must compile cleanly on the lab computers in OCNL 340 or `jaguar` and use the gcc/g++ argument `-Wall`.

- Check all function calls for errors. Points will be deducted for poor code.
- Put both partner names, class, and semester in the comments of all files you submit.
- Submit your source file and Makefile through Learn. Do not submit an archive (zip/tar) file.

Input/Output

Below are input and output examples for your program, where each line consists of `source -> destination`. **Make your program match the formatting of the examples below!** You may define error messages as you wish.

```
$packets trace.cap
Processing file 'trace.pcap'
0:1f:33:cf:2d:c4 -> 8c:a9:82:62:63:6c
    [IPv4] 10.0.0.2 -> 71.139.247.68
    [TCP] 8334 -> 80
8c:a9:82:62:63:6c -> 0:1f:33:cf:2d:c4
    [IPv4] 71.139.247.68 -> 10.0.0.2
    [TCP] 80 -> 8334
8c:a9:82:62:63:6c -> 0:1f:33:cf:2d:c4
    [38]
0:1f:33:cf:2d:c4 -> 8c:a9:82:62:63:6c
    [IPv4] 10.0.0.2 -> 71.139.247.68
    [101]
70:1a:4:ac:ef:b2 -> 33:33:0:0:0:c
    [IPv6] fe80::e171:e8c6:cbe4:ac15 -> ff02::c
    [UDP] 3455 -> 10456
33:33:0:0:0:c -> 70:1a:4:ac:ef:b2
    [IPv6] ff02::c -> fe80::e171:e8c6:cbe4:ac15
    [123]

#packets
Capturing on interface 'eth0'
0:1f:33:cf:2d:c4 -> 8c:a9:82:62:63:6c
    [IPv4] 10.0.0.2 -> 71.139.247.68
    [TCP] 8334 -> 80
```

Grading

Your program will be graded based on the following, which is also a suggested order of implementation.

- 20 points: Print Ethernet information (source and destination address and decoded type)
- 20 points: Print IPv4 information (source and destination address)
- 20 points: Print IPv6 information (source and destination address)
- 20 points: Print UDP and TCP information (source and destination port)
- 20 points: Code quality and neatness

Hints

- Build and implement your program in steps, as outlined above.
- You may find the functions `ntohl(3)`, `ntohs(3)`, and `memcpy(3)` helpful.
- Programs that are functional, but implement fewer requirements are likely to earn more points than programs that are not functional and attempt to implement many requirements. Programs that do not compile or do not run correctly will receive almost no credit.