Computer Networks I — Fall 2013

Programing Project #4 - Protocol Analyzer — 2013-12-11

In this project you will use the pcap library to write a Protocol Analyzer, giving you an opportunity to review all four network layers studied this semester. The assignment will be graded in three parts

1. For full credit your submission must have the ability to read the saved file and report basic statistics about the major TCP/IP protocols.
2. For an extra 10% you may include statistics about information inside the protocol headers.
3. You can get an additional 10% by adding code that recognize and reports on at least one application layer protocol.
4. If you want to be really industrious you can get yet another 10% by computing statistics about individual TCP flows.

Details

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. It also gives us a chance to look at compete ethernet frames without having root or administrator access on a system.

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.

Part 1 - Count the basic protocols

I've created a skeleton program for you to start with and put it on isengard in ~promig3/pub/project4. This program 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 should not need to worry about the mechanics of the pcap library. The code I provided will call a function called **`pk_processor`** once for each packet in the file, passing in a pointer to the current packet. 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`**).

For the first part of the assignment your program should collect and report:

- At the link layer:
  - The number Ethernet II frames.
  - The number 802.3 frames.
- At the network layer:
  - The number of, as well as the average, maximum and minimum size of ARP, IPv4, and IPv6 packets.
  - The total number of packets of all other network layer protocols.
- At the transport layer:
  - The number of, as well as the average, maximum and minimum size of ICMP, TCP and UPD packets.
  - The total number of packets of all other network layer protocols.

Part 2 - Protcol Details (extra credit #1)

Once you can identify the basic protocols you should add:

- The total number of unique source/destination MAC addresses.
- The total number of unique source/destination IP addresses.
- The total number of unique source/destination UDP port numbers.
- The total number of unique source/destination TCP port numbers.
- The total number of SYN and FIN packets seen.
- The total number of fragmented packets (i.e., IP packets with the more-fragments flag set).

Feel free to include any other statistics you think might be interesting/useful.

Part 3 - Application Layer Protocol

Pick at least one application layer protocol (i.e., HTTP, FTP, DNS) and report basic statistics about the protocol. For example if you select HTTP you should report the number of GET, PUT, POST etc. requests, the number of requests using each each version of the protocol (e.g., 1.0, 1.1). You may assume that packets are sent/received by the default port, but you need to report something that indicates you analyzed the protocol information.

Part 4 - TCP Flows:

A TCP flow is one half of a TCP session - all the packets from the client to the server make one flow, all the packets from the server to the client make up the other flow. You need only include complete flows (i.e., a flow where you see both the SYN and the FIN packet; don't worry about the FIN from the other flow and/or ACKs).

- Report the number of flows found, the average, maximum and minimum length of the flow in both packets and seconds.

Submission

This assignment is due by the end of the day on Thursday December 5th.

Students should submit a single tarball containing a Makefile, a README.txt with information about which application layer protocol you worked on and the source files needed to build your program. All these files should be include in a single directory, named with your username. The grader should not need to do anything other than untar your files, type make and begin testing.

Do not include any core, object or binary files in the tarball (also don't include any pcap dump files, we will use our own for testing).

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