CS 353: Programming Assignment 2

Revision History:
3/9 Release

The main goal of this assignment is to give you hands-on experience in developing tools for network traffic analysis and management. In this assignment you will use RAW sockets and libpcap-based tools to generate and analyze network traffic.

This assignment is organized in two parts.

The PARTS

I. Develop a ping tool to send periodic ping messages to the specified IP address and compute the estimated RTT based on the received responses. The program will need to use RAW sockets. More information about RAW sockets is located at http://sock-raw.org/papers/sock_raw

Command Line Options:

> pinger -p "data" -c N -d IP

where

```
-l, --logfile     Write the debug info to the specified log
                  file
-p, --payload     The string to include in the payload
-c, --count       The number of packets used to compute RTT
-d, --dst         The destination IP for the ping message
```

When run without any options, it should print the usage instructions and exit. You can create a log file to help debug your program.

Expected Command Line Output:

```
[hussain@laptop:~] sudo pinger -d 206.190.36.45 –c 4 –p "hello"
Pinging 206.190.36.45  with 5 bytes of data "hello":
   Reply from 206.190.36.45: bytes=5 time=69ms TTL=47
   Reply from 206.190.36.45: bytes=5 time=70ms TTL=47
   Reply from 206.190.36.45: bytes=5 time=69ms TTL=47
   Reply from 206.190.36.45: bytes=5 time=69ms TTL=47
 Ping statistics for 206.190.36.45:
  Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
  Approximate round trip times in milli-seconds:
  Minimum = 69ms, Maximum = 70ms, Average = 69ms
```

II. Develop a viewer tool to observer the IPv4 ICMP traffic on the network or in a file. The tool should use libpcap to capture the network packets. More information on the pcap library is located at http://www.tcpdump.org

Command Line Options:

> viewer -i interface -c N –r filename

where

```
 -i, --int           Listen on the specified interface
 -r, --read          Read the pcap file and print packets
 -c, --count         print count number of packets and quit
 -l, --logfile       write debug info to the specified log
                     file
```

When run without any options, it should print the usage instructions and exit. You can create a log file to help debug your program.

Expected Command Line Output:

```
[hussain@laptop:~] sudo viewer –i en0 –c 6
viewer: listening on en0
1489165579.494482 192.168.1.155 > 206.190.36.45: ICMP echo request, id 19735, seq 0, length 5
1489165579.538329 206.190.36.45 > 192.168.1.155: ICMP echo reply, id 19735, seq 0, length 5
1489165580.497752 192.168.1.155 > 206.190.36.45: ICMP echo request, id 19735, seq 1, length 5
1489165580.540739 206.190.36.45 > 192.168.1.155: ICMP echo reply, id 19735, seq 1, length 5
1489165581.498975 192.168.1.155 > 206.190.36.45: ICMP echo request, id 19735, seq 2, length 5
1489165581.545551 206.190.36.45 > 192.168.1.155: ICMP echo reply, id 19735, seq 2, length 5
```

Code and Collaboration Policy

You are encouraged to refer to the socket programming tutorials. You can discuss the assignment and coding strategies with your classmates. However, your solution **must** be coded and written by yourself. Please refer to the plagiarism policy in the course syllabus.

The submissions will be run through code similarity tests. Any flagged submissions will result in a **failing score**. Keeping your code private is your responsibility.

You are strongly encouraged to pair and test your client and server implementations with your peers in class.

Submission Instructions

You can develop and test your code on aludra or using your own machines. Create a compressed tar file which includes a README, Makefile for C/C++, and the source code.

To submit, create a folder called assign1 with the above files. Tar the folder assign1. Submit the tar file on blackboard.

The README must contain: USCID, compiling instructions, additional notes on usage if needed.

For C/C++: to evaluate your project we will untar and type
% make

It is a project requirement that your code build without any warnings (when compiled with -Wall flag). Structure the Makefile such that it creates the client and server executable (not a.out). For more information please read the make(1) man page.

For Python: Make sure you add the directives to support direct execution

We will then run your programs using a suite of test inputs. After running the program, we will grade your work based on the output. It is recommended that your implementation be somewhat modular. This means that you should follow good programming practices—keep functions relatively short, use descriptive variable names.

Deadlines

Due on April 4th by 6pm