Project 4: TCP traceroute

Type of Project: Individual or Pair
Deadline: 2017-04-30, 11:59pm
Language: C/C++
Points: 30 points

Submission Guidelines: Submit through ELC, as usual. Name the project file as "**LastName**-tcp_traceroute" (e.g., "Roberto-tcp_traceroute"). Submit the source code pluse the "**Makefile**", so that the code can be compiled simply by executing 'make'. Also, make sure the compiler will name the executable file as "tcp_traceroute".

NOTE: project submissions that do not follow the guidelines risk to receive 0 points.

Project Description: In this project, you are required to **use raw sockets** to write a simple traceroute-like program. This program will craft IP packets with an increasing value of TTL. Each IP packet will carry a TCP segment. The TCP header must have the PUSH flag on (all other flags need to be off). The TCP payload needs to carry the string "CSCI6760-S11" (nothing else). Also, you need to make sure that both the IP header checksum and TCP checksum are correct.

Your program needs to run as

```
# ./tcp_traceroute <SRC IP> <DST IP> <TCP port>
```

for example

```
# ./tcp_traceroute  128.192.76.177  74.125.157.104  80
```

The output should look something like:

```
 1  128.192.76.129  0.507 ms  0.614 ms  0.756 ms
 2  128.192.0.5  0.580 ms  0.899 ms  1.226 ms
 3  128.192.254.49  0.355 ms  0.444 ms  0.479 ms
 4  128.192.166.69  0.437 ms  0.463 ms  0.534 ms
 5  70.33.127.97  2.561 ms  2.281 ms  2.052 ms
 6  128.192.166.33  4.135 ms  4.097 ms  3.715 ms
 7  74.125.48.33  13.782 ms  13.665 ms  13.586 ms
 8  72.14.239.100  4.046 ms 64.233.174.2  4.474 ms 72.14.239.100  4.331 ms
 9  209.85.254.249  5.487 ms  5.364 ms 72.14.239.131  4.814 ms
10  209.85.252.66  4.346 ms 209.85.252.102  4.147 ms  17.470 ms
11  74.125.157.104  5.295 ms  6.790 ms  7.464 ms
```

The TTL must start from 1 and go to a max of 30. Each TTL needs to be "tested" 3 times. For example, your program will send 3 packets with TTL 1, 3 packets with TTL 2, etc. Of course, to calculate the time deltas you need to wait for an ICMP "time exceeded" packet to come back for each IP packet you send.

Notice that in some scenarios (in particular when no Firewall is blocking incoming non-SYN TCP packets) you may receive a TCP RST (reset) packet from the DST IP, in which case (if the TCP port matches) you should stop sending any packets and exit.

Of course, in some cases packets may get lost. You should use a timeout of no more than 1s within wich you'll wait for an ICMP "time exceeded" or a TCP RST packet to come back. After 1s wihtout receiving anything you can assume that the reponse packet got lost and move on to the next one.

TESTING YOUR CODE

You should test your code in multiple scenarios, including cases of DST IPs behind a firewall and DST IPs that are completely "open". Find some DST IPs for which the routes are pretty stable and compare your results with the output of "traceroute" (use the -T and -p and -n options).

MAILING LIST: If you have questions about the projects, the best place to ask is the course mailing list.

PROJECT EVALUATION

I will run your progarm on 5 different combinations of <DST IP> and <TCP port>. You will be assigned 4 points for each test that produces the expected output (as compared to traceroute and my own implementation of the program you are requested to write).

RESOURCES

You can find information about how to use raw sockets here (you may be able to find better tutorials than the ones I'm reporting below):

http://www.tenouk.com/Module43a.html
http://sock-raw.org/papers/sock_raw