Project 1: Compute per-hop Traceroute times given a tcpdump text trace

Type of Project: Only Individual
Deadline: 2017-02-03, 11:59pm
Language: C/C++ is recommended, though Java or Python are also allowed for this first project
Points: 10 points for C/C++ code, 8 points for Java/Python code

Submission Guidelines: Submit through ELC. Submit ONLY the source in one .tar.gz file. The .tar.gz file should contain your source code plus a Makefile (or a build.xml for Ant, if you use Java), so that the code compiles easily by just using 'make'. Also, be sure to name the output of the compile, i.e., the program file name, as "traceroute_analysis" for C/C++, "TracerouteAnalysis.class" for Java, or "traceroute_analysis.py" for Python.

The program needs to take the file name of the tcpdump trace as the first argument. For example, the program needs to run as:

```
$ ./traceroute_analysis  traceroute_TCP_trace.txt
```

NOTE: project submissions that do not follow the guidelines risk to be discarded wihtout consideration (i.e., 0 points).

Project Description: In this project, you are required to write a program that takes in input a textual tcpdump trace of traffic generated by Traceroute and computes the time between a TCP packet sent by the client and the related ICMP "Time exceeded in-transit" message.

As an example, consider the two packet logs reported below:

```
1291654312.963163 IP (tos 0x0, ttl 1, id 9067, offset 0, flags [none], proto TCP (6), length 60) 128.192.76.177.47212 >
137.138.144.168.80: S, cksum 0xc4d6 (correct), 1135826272:1135826272(0) win 5840 <mss 1460,sackOK,timestamp 2152510109
0,nop,wscale 2>

1291654312.963644 IP (tos 0xc0, ttl 255, id 2503, offset 0, flags [none], proto ICMP (1), length 56) 128.192.76.129 >
128.192.76.177: ICMP time exceeded in-transit, length 36
    IP (tos 0x0, ttl 1, id 9067, offset 0, flags [none], proto TCP (6), length 60) 128.192.76.177.47212 >
137.138.144.168.80:  tcp 40 [bad hdr length 0 - too short, < 20]
```

as we can see from the highlighted fields, the two packets are related to each other.
The fields you should check to match a sent TCP packet with the related ICMP response are: `id 9067, TCP, 128.192.76.177.47212 > 137.138.144.168.80`
Notice that these fields are replicated in the body of the ICMP message (in practice the IP ID field should be sufficient to correctly correlate the two packets).

From the two packets above, the output should be:

```
TTL 1
128.192.76.129
0.481 ms
```

where *128.192.76.129* is the IP addresses of the router that generated the ICMP response, and *0.481 ms* is computed as (1291654312.963644 - 1291654312.963163) * 1000 and rounding to obtain only three digits after the dot.

As another example the output related to the following packets

```
1291654312.963267 IP (tos 0x0, ttl 3, id 9075, offset 0, flags [none], proto TCP (6), length 60) 128.192.76.177.56812 >
137.138.144.168.80: S, cksum 0x5834 (correct), 2778675862:2778675862(0) win 5840 <mss 1460,sackOK,timestamp 2152510109
0,nop,wscale 2>

1291654312.963655 IP (tos 0x0, ttl 62, id 47385, offset 0, flags [none], proto ICMP (1), length 56) 128.192.254.49 >
128.192.76.177: ICMP time exceeded in-transit, length 36
    IP (tos 0x0, ttl 1, id 9075, offset 0, flags [none], proto TCP (6), length 60) 128.192.76.177.56812 >
137.138.144.168.80:  tcp 40 [bad hdr length 0 - too short, < 20]
```

would be

```
TTL 3
128.192.254.49
0.388 ms
```

Consecutive TCP-ICMP packet pairs related to the same ICMP source IP should be listed without repeating the TTL value and IP string. For example, if there are 3 consecutive TCP-ICMP packet pairs related to TTL=3 and with ICMP packets originating from 128.192.254.49, you should list them as

```
TTL 3
128.192.254.49
0.388 ms
0.401 ms
0.398 ms
```

NOTE: there are some differences between what you see from traceroute and what you will get from the analysis. The reason is that traceroute may time-out and show '*' even though the ICMP packet was not lost (it was simply late), while tcpdump will of course report those packets anyway as they arrive to the interface. Also, the time deltas reported by traceroute may be slightly different from the ones reported in the analysis output.

Your software will NOT be tested on the example trace. To make sure your software works as required on other traces, you can generate them on your own by running, for example

```
# tcpdump -tt -i eth3 -p -n -nn -v icmp or tcp port 80
```

```
# traceroute -T -p 80 -n www.cern.ch
```