COMPUTER NETWORKS

PROGRAMMING ASSIGNMENT 1

1 Problem description

Write a program that implements a subset of `traceroute` features. It should: display IP address of all routers on path to destination IP address, work in console (no user interface required) and accept the only command line argument, which is destination IP address.

The program should send *ICMP echo request* packets with increasing TTL, same as it's done by `traceroute -I`. For each value of TTL $\in [1, 30]$ perform following steps:

1. Send three *ICMP echo request* packets with given TTL (in one burst, without waiting for reply).
2. Wait for certain, given upfront, time period (e.g. 1 second). [1]
3. Receive packets, that arrived within the time period. It may happen that you receive replies for requests made earlier (i.e. lower TTL values), in such case you should treat them as junk. In order not to block on receive operation, you may choose to use non-blocking sockets. However even more effective method would be to employ `select()` call from standard *Un*x* library.
4. Display an IP address of replying router / host together with average packet round-trip time in milliseconds. However if you don't get all three replies within the time period you should display `???`. In case no router replied with a packet you should display `*`. If you get replies from more than one router you should display all IP addresses.

The program should finish when it receives a reply from destination IP address.

Sample printout your program could display is given below:

```
> ./my-awesome-traceroute 156.17.254.113
1. 156.17.4.254 40ms
2. 156.17.252.34 ???
3. *
4. *
5. 156.17.254.113 156.17.254.114 50ms
6. 156.17.254.113 65ms
```

Your program is supposed to handle malformed argument and indicate that condition by displaying an error message.

[1] Obviously, one could perform wait operation and sending packets from next loop iteration concurrently.

2 Remarks

2.1 Implementation

1. Use *raw sockets* to send and receive *ICMP* messages. Note, that use of these sockets requires administrative permissions. (Programs will be tested in a virtual machine on computers in room 109).
2. Embrace fact that you can match *ICMP* request with reply packets. *ICMP echo request* and *ICMP echo reply* messages have exactly the same values as *identifier* and *sequence number* fields. *ICMP time exceeded* messages, as their payload, have IP header and first 8 bytes of original packet (i.e. complete *ICMP echo request* header).
3. You may use code snippets discussed during the lecture. Especially interesting one is the function calculating control sum of an ICMP header. The snippets are available in the git repository[2].

2.2 Technical remarks

Files The program should be sent to your teacher as a single compressed archive (preferably in *.tar.gz* format). It must contain a single directory and within it:

- Source code in C or C++, i.e. `*.c` and `*.h` files or `*.cpp` and `*.h` files. Each of `*.c` and `*.cpp` source files must contain a comment (at the beginning of file) with following information: first and last name, an "indeks" number.
- `Makefile` file, that enables the program to be compiled through running `make` command.
- Optionally, `README` file containing any remarks from a student.

The directory **must not** any other files that mentioned above, and especially compiled program (in binary form) and linkable objects `*.o` or any other files not required to build the program.

Compilation The program will be compiled and run in Linux 64-bit environment with fairly recent `GCC` compiler.

In case of C language compilation, your program is allowed to use ISO C99 standard, possibly with GNU extensions (compiler option `-std=c99` or `-std=gnu99`).

Compilation will be performed with `-Wall` and `-W` options. The compiler must not report any warning with regards to your code.

Marcin Bieńkowski
Translation: *Krystian Bacławski*

[2] http://github.com/cahirwpz/computer-networks-course/tree/master/programs/icmp