Lab 2b) Binary UDP server

Introduction

This task is the server for Lab1b.

It uses **UDP** as the transport protocol, and on the application-level, it is a **Binary** protocol between the client and server. Messages and formatting instructions are defined in *protocol.h*, found in the base repository https://github.com/patrikarlos/np_assignment2 (Links to an external site.), also see Lab1b.

Requirements

Same communication protocol as in Lab1b.

1. The servers should accept IP:port as the first command-line argument.
2. The server must handle multiple clients, near-simultaneous.
3. The server only supports protocol 1.0, i.e. calcMessage.{type=22, message=0, protocol=17, major_version=1, minor_version=0}.
4. If a client has not replied to an assignment after 10s (or more), that job should be removed, and the client should be considered lost.
5. If a lost client tries to provide a result, it should be rejected with an error.
6. If a client tries to report a result but provides an incorrect ID it should be rejected. Hence, the server needs to track client IP, port, ID, and assignment.

Submission

Your submission **must** be in the form of a *tar.gz* file, containing your solution in a folder called np_assignment2.

The solution should have a well-utilized git repository, source code, and a makefile that is used to build the solutions (client and server). Start from the repository above, then you will have the folder, Makefile, and templates for the solutions. Make sure that you track the progression of the solution with commits to the repository, of course, you should not commit things that did not work, or that was just for testing. I will not examine your repository handling, but I'll check that YOU made progress. I.e. it should not be a clone of my repository, and a commit 'solved'. Once you have completed and tested the solution, run make clean, then compress the entire folder (Links to an external site.), including the git repository. The compressed file (archive) should be named <bth_id>-assignment2b.tar.gz. Then submit that archive here.

Testing

As I'll have several solutions to test, I've automated a large part of the testing. The automation requires that you follow the naming standard, and that 'make' and 'make clean' works. That the solutions are called client and server, nothing else. Both should accept IP:PORT as the only argument.

I'll test both of your solutions against other implementations, hence, protocol compliance is critical.