A News System Implementation
C++ Programming, EDA031

Erik Hedblom, [tpi11ehe] Tobias Landelius, [ael10tla]
Oskar Jermakowicz, [dat12oje] Martin Larsson, [dat12mla]

April 18, 2016

Table of contents

1 System description **3**
1.1 Server 3
1.2 Client 3

2 Conclusion **4**

Appendix A **5**

Appendix B **6**

1 System description

The application is a simple implementation of a news system where newsgroups and articles are stored on a server which communicates with clients that connect to it. The client is a Command Line Interface where users can execute commands to receive, edit or delete data on the server.

The implementation is divided into three packages as can be seen in Figure 1 below; `client`, `library` and `server`.

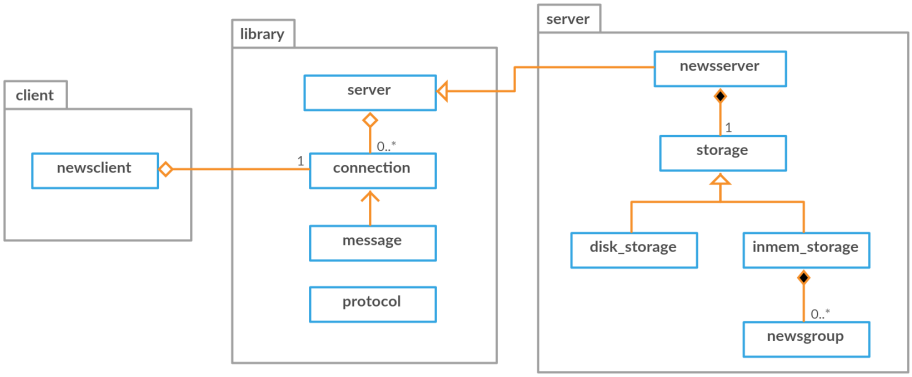

Figure 1: UML diagram of the system.

The purpose of the library package is to simply handle the interaction between the server and the client. During the execution of commands on the server a given protocol is used for the communication which is defined in `protocol.h` in the library package. The purpose of both the server and the client will be further explained in detail in section 1.1 and 1.2 of this report. In Appendix A an example sequence diagram can be seen, this shows the client and server interaction steps that are taken during the removal of a newsgroup when such a request comes from the client.

1.1 Server

The provided classes `Connection` and `Server` manages the actual network sockets and connected clients. From them we receive a stream of bytes to decode as commands from the clients. The decoding and execution of commands is done by `Newsserver` using the utility functions in `message`. The commands may result in changes in the database which is used to store all newsgroups and articles. There is source code for two types of databases: one only using the primary memory (`inmem_storage` and `newsgroup`) and one working against the disk (`disk_storage`). Both of these classses inherit from the base class `storage` making it easy to swap between the two, although `disk_storage` is the one currently used. There is also a set of possible exceptions defined in `storage_exception` that can happen when interacting with the database.

1.2 Client

In order to start a client, one has to run the following command on the command line in the directory of newsclient; `newsclient host-name port-number`. If the

connection is successful, this will establish a connection with the server. The establishment of the connection is handled by the `NewsClient` constructor which creates a `Connection` object which will be used to communicate with the server.

Once a connection is established the `run()` method of `NewsClient` is run, and the user can now execute commands. All possible commands are explained if the user runs the command `help`, the printout of the `help` command can be seen in Appendix B.

The client parses each user input through regex to see whether the input is a valid command. This is done through `parse_line(string)` which matches any valid commands and parameters and stores them in a vector. This vector is thereafter compared with the available commands in `handle_command(string)`, and if it turns out the commands is valid, the corresponding method for that particular command is run in `NewsClient`. These methods simply call the different send commands in the `message` class in the `library` package. The start and end of the command is sent with the `send_code(Connection, int)` method in `message`, where the int is the corresponding protocol definition from `protocol.h`. If the command has any parameters, these are sent with either `send_int_parameter(Connection, int)` or `send_string_parameter(Connection, string)`. Once the whole command has been sent to the server, the same method will check the servers response with `consume_code(Connection, int)` (where int is a protocol definition) and `recv_code(Connection)` in `message` and print the servers response to the user.

2 Conclusion

We fulfilled all requirements given to the extent that is possible. For example, the requirement that there should be no limitation on the number of newsgroups or articles is not realizeable. The newsserver is not particulary safe in the sense that anyone can connect to a server and modify the database. A better solution would involve users logging in and have restricted access to perhaps only read articles or only be able to delete articles created by the user.

While the disk database does work as intended it does not handle all possible errors that can arise when working against disk (files can go missing, corrupted files, permissions, etc). We did not implement all this error handling as it felt out of scope for the project.

The project as a whole felt small for the course, though, perhaps because we were a full team of four students. It would have been interesting to explore more obscure parts of C++ as there were plenty of time to complete the project.

Appendix A

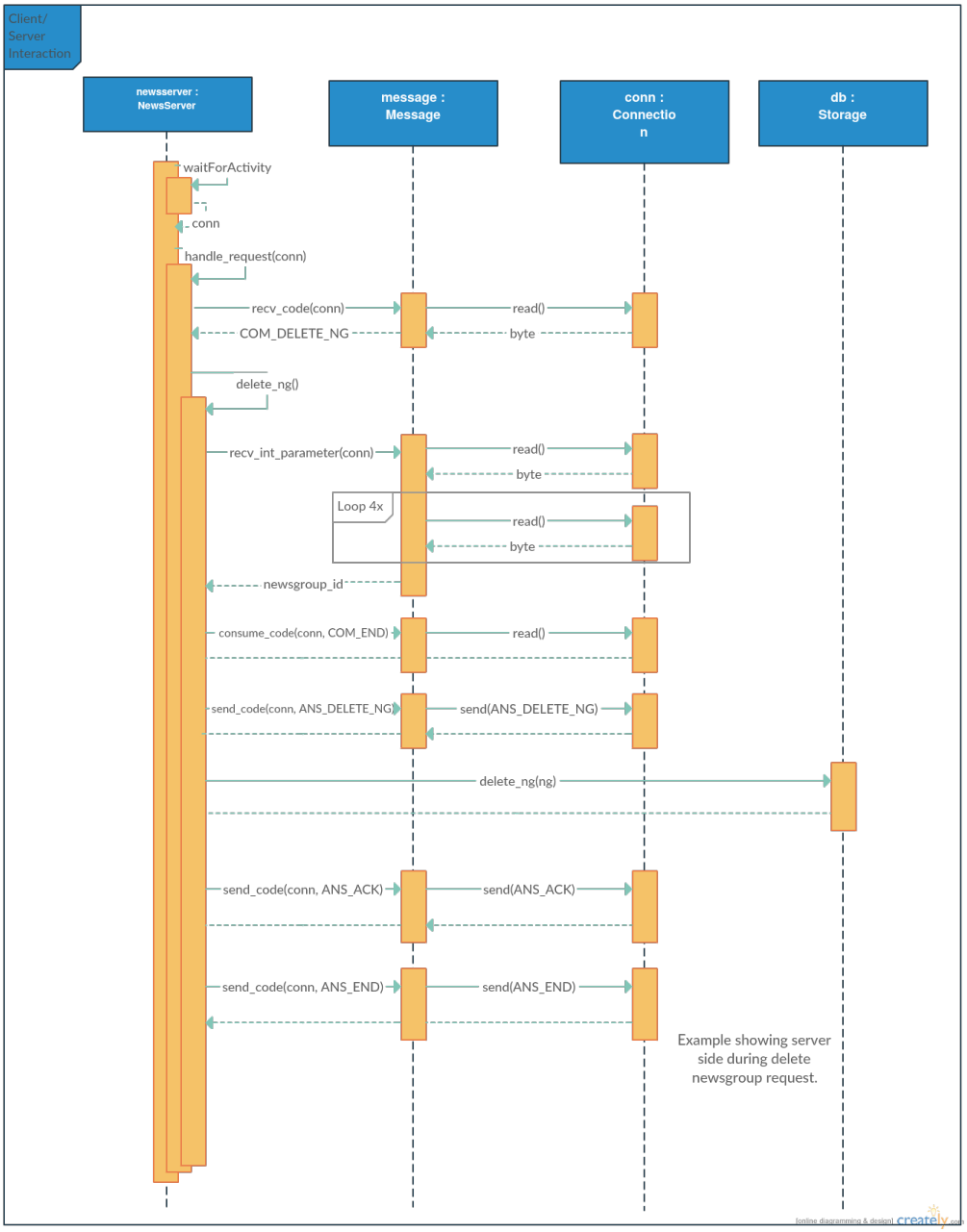

Figure 2: The interaction between server and client.

Appendix B

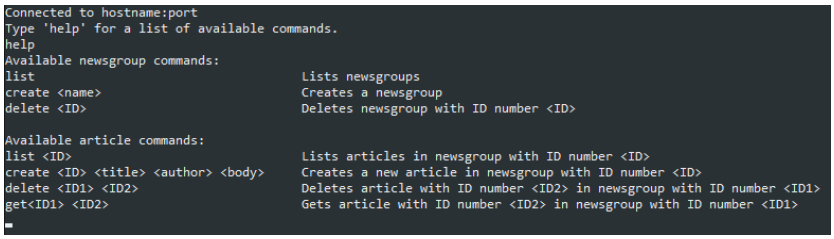

Figure 3: Available commands, shown when the `help` command is run.