Computer Networking - EX. 1

Roi Koren 305428369 roikoren@gmail.com
Bar Zamir 204002570 bar820@gmail.com

April 7, 2019

1 Protocol

Before getting to the protocol itself, we'll describe the basics of sending a non-negative integer between client and server, and sending a string. To send an integer, the sending side must convert it to big-endian (network standard), and send the 4 bytes resulting from converting it to char* (in C). To send a string, the sending side must first send the string's length, as described above, and then send the wanted string. We provide an implementation of this mechanism in the seker_helpers.c & .h files.

We'll also define SUCCESS to be 0, and ERROR to be 1.

The protocol we used is as such:

Firstly, the client connects to the server with a TCP socket, connecting to the server's address and port. Upon receiving the connection, the server replies with SUCCESS.

The client then sends a username and password strings to the server. If they are valid, the server responds with SUCCESS. Otherwise, it responds with ERROR, and the client must send another pair of username and password strings. The server will not disconnect from the client, and will keep waiting for a new pair of username and password strings, until a match is made.

The client may then send any of the 5 defined requests, by sending its number to the server. The requests and their numbers are defined in seker_helpers.h, and are 1 for list_of_courses, 2 for add_course, 3 for rate_course, 4 for get_rate, and 5 for quit. After handling the request finishes, the client may send another request, until a quit request is made, after which the user is logged-out and the client disconnected.

Next we'll describe the handling of the different requests, on both the client and server sides.

Upon sending a list_of_courses request, the client then should expect the receive a list of strings, each string containing a course number and name, separated by a tab character, the name surrounded with quotation marks. This list is appended with a newline character. The list ends with an end_of_list string, also defined in seker_helpers.h to be "end_of_list". The server should send this list of courses, and the end_of_list string. Note that the course number is part of the string, and not sent as a non-negative integer.

If the client sends an add_course request, it must then send a desired course number, between 0 and 9999. The client should send this number as an integer. If that number is not used by an existing course, the server will reply with SUCCESS, and the client may continue. Otherwise, the server replies with ERROR, and the request is ended. If the course number is not in use, the client must then send the course's name, surrounded with quotation marks, as a single string. The server should add this new course, and be ready to list it, receive ratings for it, and return its ratings, if requested, from now on.

If the client sends a rate_course request, it should then send the course number which is to be rated, as an integer. The server responds with ERROR if no course was found with that number. Otherwise, it replies with SUCCESS. The client may then send the rate itself, as an integer between 0 and 100. Then, the client should send the rate text to the server, as a single string surrounded with quotation marks. The server should save this rating, together with the username of the user supplying the rating, for future queries on ratings of this course.

A get_rate request from a client should be followed with the client sending the desired course number as an integer. The server will reply with ERROR if such a course number is not found, and the request will end. Otherwise, the server will reply with SUCCESS, after which it will send the client a list of ratings of the desired course, followed by the end_of_list string. Each rating will be a string containing the username for the user who gave the rating, the rating the user gave, and the textual rating surrounded with quotation marks, the three separated with tab characters. Each rating ends with the newline character. The server must send these ratings, in this format, of course.

Finally, the quit request is not followed by any more data being sent between the client and server.

2 Notes

- Both client and server programs depend on the common seker_helpers.c & .h files to be compiled with them.
- When providing user input to the client program, the parameters are separated by whitespaces, namely spaces and tabs. This means that ‘add_course TAB TAB 1 "Computer Networking 101"‘ is equivalent to ‘add_course 1 "Computer Networking 101"‘, with TAB being the tab character.