Operating Systems and Networks

Assignment 5 - Question 3

Multithreaded-Client and Server

1) Server

- The server runs on the port `6969`. It can be changed in the `q3/src/common.h` file by changing the `SERVER_PORT` macro.
- The server has a `BACKLOG` of `1024` that means it can listen to a maximum of 1024 connections. This can be changed in `q3/src/server/server.cpp`.
- The server first takes the number of threads in the thread pool that handle the connections from the command line arguments.
- It then initializes the dictionary, mutex locks for each key.
- It also initializes a queue where all the incoming connections will be enqueued. A lock is also initialized for it. A conditional variable is also kept.
- After that the server starts the threads to keep looking for work in the queue.
- It then binds a socket on the `SERVER_PORT` and then starts listening on it.
- It then starts an infinite loop and starts trying to accept new connections from the clients.
- Whenever a new connection comes, the server acquires the lock of the connection queue and then enqueues the connection into it.
- It then broadcasts to the conditional variable related to the queue to tell the thread pool's threads that a new connection has arrived.
- The main thread keeps on doing this infinitely.

1.1) Thread Pool

- In the pool threads' function, first of all we acquire the lock of the queue as we intend to dequeue a connection from it.
- Then we start a while loop till the queue is empty and inside we wait on the conditional variable related to the queue.
- After getting some connection, the thread passes it to another function that communicates with the connection.

1.2) Communicating Function

- This function just reads data from the connection fd, parses it, executes the command, and writes the result back to the connection fd.
- In the command functions, **whenever a key is to be accessed/modified, the lock for the key is acquired and only then the operation is performed. This allows multiple commands that use different keys to run in parallel.**

2) Clients

- Each client has its own thread.
- First of all it sleeps till the time for its sending the command to the server arrives.
- Then it binds a new socket to connect with the socket of the server.
- It then attempts to connect with the server.
- After the connection is successful, it sends the command that is is supposed to run and waits for the response.
- Before printing anywhere in the clients, I acquire a mutex lock so that the outputs of different functions don't get intermingled and after printing, the lock is freed.