Lauri Tirkkonen

Design exercise

Overall architecture of the system

The chosen socket technology to be used is TCP, as it is more reliable. TCP is a connection-based protocol, meaning that after the connection has been made, the data can be transferred in two directions. In a multi-user chat system, it would be beneficial to confirm that the message was send and received.

The chosen execution style for the system was multithreading, as it is more quick, efficient and failure proof. The threads are independent of each other and issue in one thread doesn't affect other threads.

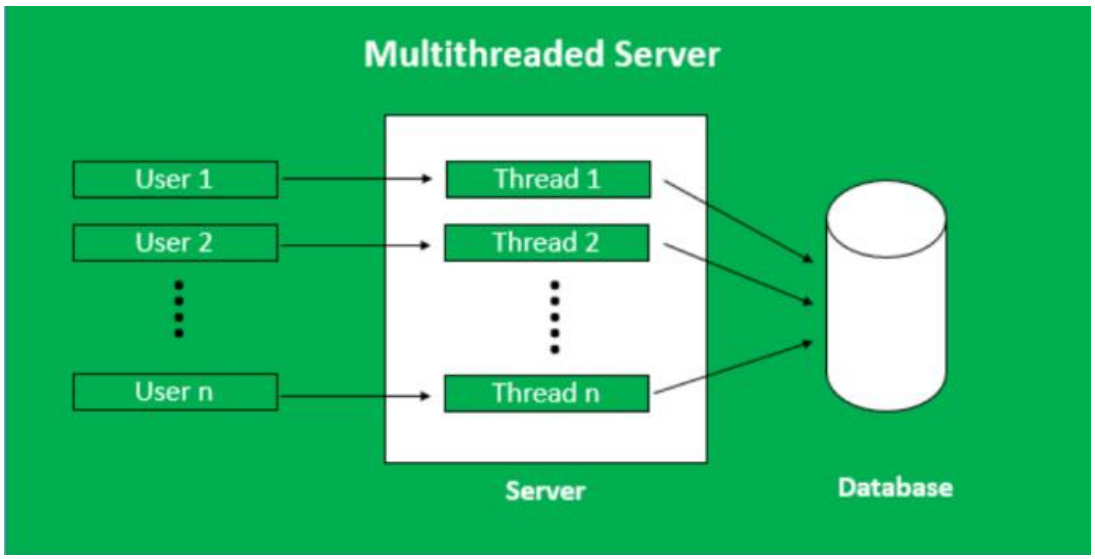

Diagram describing the overall architecture of the system.

Development / Execution

The main ideas remained unchanged in the development, despite changes made to some details. The system was made with Python. The server uses multithreading in the form of utilization of threading python-module and TCP in the form of utilization of socket python-module.

The confirmation is not utilized very well, as the functionality resembles more UDP style than TCP. The server listens for messages and if it receives a message, it will send the message to recipients as well as the sender (confirmation). There is no database, the data is handled with lists of active users, that is located inside the server.

Transparency: has been catered for, except some information messages are send in the name of server.

Scalability: Can be scaled with no limitations. Server threads are not killed, which could cause issues.

Failure handling: Handled in a way that failures can be ignored. Multiple servers could be utilized, so failure in a single server wouldn't cause problems.