

Distributed System (COMP90015\_2019\_SM2) – Assignment 1 Report

Multi-threaded Dictionary Server

Name: Boyang Sun

Student ID: 1064064

Email: [boyangs@student.unimelb.edu.au](mailto:boyangs@student.unimelb.edu.au)

Date: 04 Sep, 2019

**1. Introduction**

**1.1 Problem Description**

Using a client-server architecture, design and implement a multi-threaded server that allows concurrent clients to search the meaning(s) of a word, add a new word, and remove an existing word.

Hence, the assignment must make an EXPLICIT use of the two above. By explicit, we mean that in your application, sockets and threads must be the lowest level of abstraction for network communication and concurrency.

**1.2 Functional Requirements**

a. Query the meaning(s) of a given word

b. Add a new word

c. Remove an existing word

d. Failure model and error handling

**1.3 User Interface**

A Graphical User Interface (GUI) is required for this project.

**2. Architecture and System Components**

**2.1 System Architecture**

The system will follow a client-server architecture in which multiple clients can connect to a single multi-threaded server and perform operations concurrently. The multi-threaded server implemented a worker pool architecture. While a request comes in, an existing thread in the thread pool will execute this request. If all the threads are working on requests, new request will be put into a queue to wait for serving.

**2.2 System Components**

The system mainly consists two parts: Dictionary Client and Dictionary Server.

Dictionary Client: Dictionary client implements a graphic user interface to get the user’s input. It is also in charge of encapsulating the request and sending the request via a TCP connection. Error will be displayed to the user with a brief description.

Dictionary Server: Dictionary server manages the dictionary data including adding, removing and querying. Transaction management of concurrency operation is provided by the dictionary server while there is a critical section. Also, the server will decapsulate the request and generate the response. Failure information will also be sent to the client to help the user.

**3. Design and Implementation**

**3.1 Class Design**

This is the system architecture

**3.2 Interaction Design and Interaction Diagram**

All communication will take place via TCP sockets.

**3.3 Failure Model and Exception Handling**

This is the system architecture

(4) Failure Model

All communication between components has to be reliable. If you are using TCP, then the reliability guarantees offered by the protocol are sufficient. If you decide to use UDP, then you have to implement an infrastructure that offers reliable communication over UDP. For those of you with previous experience using TCP, using UDP may be a rewarding challenge.

It is expected that, on both the server and the client side, errors (by means of exception handling) are properly managed. The errors include the following:

Input from the console for what concerns the parameters passed as command line.

Network communication (address not reachable, bad data...).

I/O to and from disk (cannot find the dictionary file, error reading the file, etc...).

Other errors you might come up with.

**4. Creativity elements**

**4.1 Functionality Introduction**

**4.2 Implementation**

**5. Analysis and Conclusion**

**5.1 Analysis**

Advantages and disadvantages of your design choices.

**5.2 Conclusion**

This is the system architecture