A Technical Report of Distributed Dictionary System

COMP90015: Distributed Systems

Lujie Ma

1391364

The University of Melbourne

March 21, 2023

Lujie Ma 1391364

Introduction

Due to technological and economic reasons plants, manufacturing systems and networks are developed with an ever-increasing complexity.

Based on the requirements mentioned in the assignment, I create the Distributed Dictionary System, which is a network-based application that allows multiple clients to connect and send commands to a shared dictionary stored on the server. My distributed dictionary system can handle four types of commands from the client, it allows the user to search for words' meanings, add new words, remove existing words, and update the words' meanings. Furthermore, I create a Graphical User Interface for users, which can provide users with a user-friendly and more intuitive experience for using my distributed dictionary system.

Features

The features of the distributed dictionary system include two components: a server-side (Dictionary Server) and a client-side (Dictionary Client).

- ## Server
  - ### Concurrent Issue

    In my distributed dictionary system, I import the concurrentHashMap class from the package java.util.concurrent to store the data from the Dictionary file. By using concurrentHashMap, our server can handle concurrency issues very well. ConcurrentHashMap can make sure all operations are thread-safe.

  - ### Multi-threads

    The dictionary server applies a multi-thread approach to handle multiple client connections. It allows the server can serve multiple clients simultaneously. By using the package of java.util.concurrent.atomic.AtomicInteger, I number each thread from 1, such as ClientHandler-1, ClientHandler-2 and so on. It makes managing threads easier.

    In the implementation of the code, the main thread listens for incoming client connections by calling the method in a while loop. When a new client connects to the server, the main thread will create a new class, transmitting the new client socket and the dictionary data as arguments.

  - ### Class ClientHandler

    The client Handler is a very important part of the server side (Dictionary Server). ClientHandler supports search, add, delete, and update commands send from client-side (Dictionary Client).

To search for multiple meanings of the word, I make a little design. In the search word function, I create a loop to traverse all meanings if the searched word is existing in the dictionary, and then send an “END” command to the client side, letting the loop end.

○ Error & Status Report

My distributed dictionary system can handle multiple error reporting, it always can report accurate errors and give users tips, so users can have a better way to identify where the problem lies and then solve it. The status reporting is also a highlight of my system, for every command sent from the clients, the server will always return the correct status ([SUCCESS] or [FAIL]) to let the user know if the operation works.

The distributed dictionary system is allowed to report these errors on the server side (Dictionary Server):

1. If you are typing the wrong command to start my distributed dictionary system, we can give the tips:
   ‘Usage: java DictionaryServer <port_number> <dictionary_file>’

2. Our system can catch any potential IO exception that might occur when reading the file:
   ‘Error reading JSON file: {error}’

3. Our system can catch any potential IO exception that might occur when have trouble creating the objects:
   ‘Error starting the server: {error}’

4. Our system can catch any potential IO exception that might occur when having an error communicating with the client:
   ‘Error communicating with the client: {error}’

5. Our server side can check the commands sent from the clients. If the length of the commands is less than 2, our system will provide the tips to the client:
   ‘Invalid command: Arguments missing.’

- Client

○ Error & Status Report

My distributed dictionary system can handle multiple error reporting, it always can report accurate errors and give users tips, so users can have a better way to identify where the problem lies and then solve it. The status reporting is also a highlight of my system, for every command sent from the clients, the server will always return the correct status ([SUCCESS] or [FAIL]) to let the user know if the operation works.

The distributed dictionary system is allowed to report these errors on the client side (Dictionary Client):

1. If you are typing the wrong command to start my distributed dictionary system, we can give the tips:
   'Usage: java DictionaryClient <server-address> <port_number>'

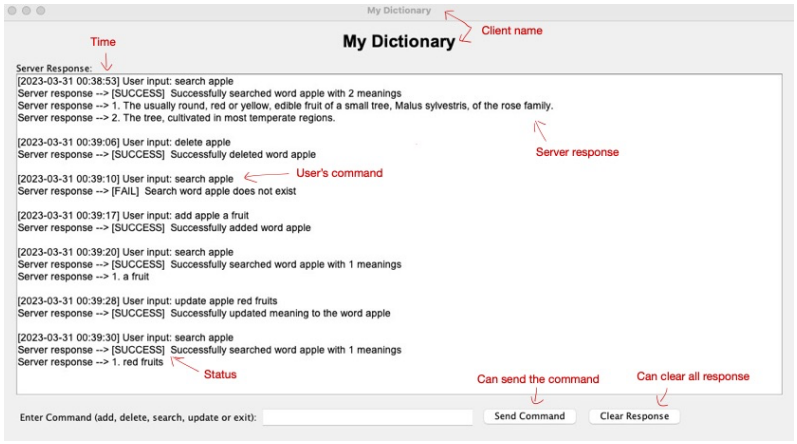
2. Our system can catch any potential IO exception that might occur in the client:
   'Error reading response from the server: {error}'

3. Our system can catch any potential IO exception that might occur when cannot connect to the server:
   'Error connecting to the server: {error}'

4. Our system can define if the user enters the wrong command and give the tips:
   'Add / update command with the wrong syntax'
   'Delete / search word {word} does not exist'

- ## GUI

The Client GUI is implemented by using Swing. Here shows a demo of the server:

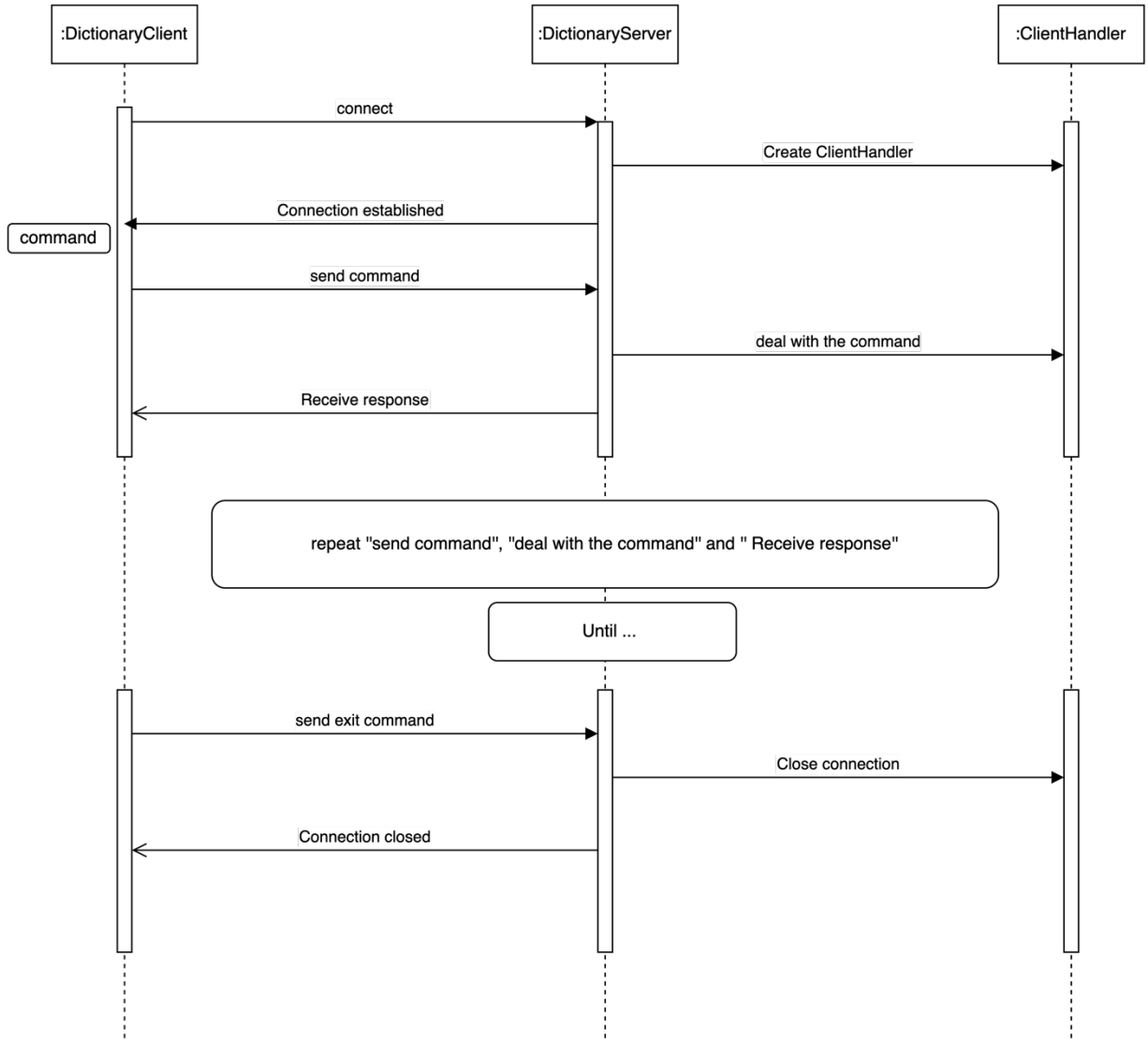


GUI Features

1. The title of the Client GUI is "My Dictionary".
2. On the client side, my dictionary can return the status of each sent command, which makes users easily know if their command works or not.
3. There is a suitable input box to handle user input. On the Client side, I also create buttons for users can send commands and clear all responses. By pressing "Clear Response", it will back to a blank board of the Server Response.

Lujie Ma 1391364

Interaction Diagram



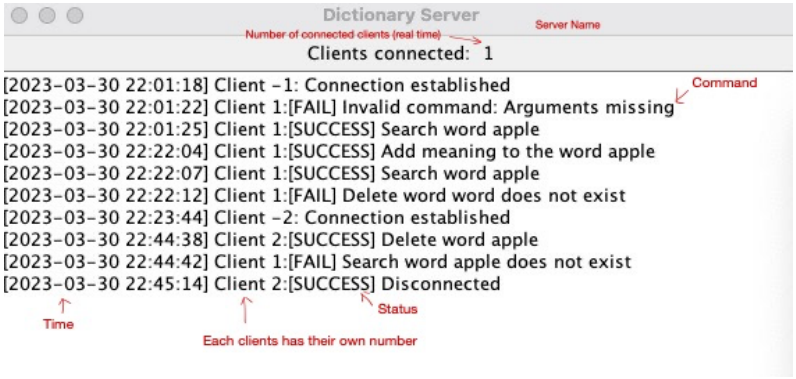
Excellence elements

- Our systems allow the server can print each status and command from each client. It allows the admin to have a better and clearly management environment and check what commands are sent to each client and confirm the status.
- Our systems can show the time on both the client side and the server side.
- On our client side, I make a design for a better user experience that allows users can press enter to send a command instead of pressing the “Send Command” button.

Creativity elements

- **Server GUI (Creativity elements)**

  To ensure that the server can have an intuitive expression, I create a GUI by using Swing. Here shows a demo of the server:



  GUI Features:

  1. The title of the Server GUI is “Dictionary Server”.
  2. In the Server GUI, we can see how many clients are connected to the current server.
  3. We can know what commands are sent by the clients and know their status.
  4. Each client which successfully connected to the server will be given a unique number. This is to make management easier.

Critical analysis & Conclusion

In conclusion, my dictionary can provide the basic dictionary services over a network. The server side efficiently manages multiple concurrent clients with reliable network connections. The client-side allows users to interact with the server.

the multi-threaded server is implemented by a thread-per-connection architecture. My dictionary adopts **TCP** to make the communication between the server side and the client side. The ‘socket’ and ‘ServerSocket' objects handle the TCP communication.

My dictionary still got limitations in the following part:

1. To make sure of a reliable data transfer, my dictionary adopts TCP to make the communication between the server side and the client side. However, it is slower than using UDP.
2. My dictionary has security issues which can not make sure all messages are protected when in communication between the server and client.