# COMP90015 Distributed Systems
# Assignment 2: Distributed White Board

ZHAOFENG QIU 1101584
ZHIHAN GUO 1100249
AOQI ZUO 1028089
SHIRU XIE 1077303

October 23, 2019

## 1 System Description

In this project, a multi-threaded dictionary client-server system is designed and implemented using a client-server architecture. The server of the system allows concurrent clients to modify or query the remote dictionary by using thread-per-request architecture in the server. The system provides reliable communication between the server and clients by using TCP protocol. Also, JSON is used in the system for providing a message exchange protocol between the server and clients. As for failure handling, errors including I/O errors, Network Connection error, and parameters errors, are properly managed on both the server and the client side. Also, illegal requests (such as adding a word that is already in the dictionary, removing or querying a none-exit word and adding a word without meaning) from clients can be detected and handled by the server properly. Moreover, the server can handle concurrent requests at the same time and edit the data in a correct way.

## 2 System Components

### 2.1 Server

The server is implemented using thread-per-request architecture and can detect and handle multiple failures. For every new request sent by clients, the server would create a new thread to handle it and give it a response. Since the server is using multi-thread technology, it can handle requests from different clients concurrently. However, when it comes to data modification, because the relative functions provided by the dictionary controller are synchronized, there will be no ambiguity in modifying the data in the dictionary. According to specific contexts, the server would handle requests in different ways and send back different states(which indicate the specific implementation) and feedbacks to clients. For better-exchanging messages with clients, every response would be packed in JSON format before sending it back to clients. Besides, the length of word and meaning provided by users , the server provides a GUI for logging operations and showing information about the server.

### 2.2 Client

The client is mainly composed of two parts: the Client's Controller and the Client's GUI. It can detect and handle failures such as Network communication problems and Incorrect parameter input problems. The Client's Controller part is responsible for handling the user's operational requirements captured by the GUI. For each ADD, DELETE, QUERY operations given by the user, the controller would establish a new TCP connection with the server and would close it when the controller gets the response of it. In this case, the client would not check whether the server is running before processing the three operations given above. The GUI implemented with Swing can provide visual interfaces to users. In addition to satisfying the user's query and modification operations, it also prompts the user how to use the client, such as asking them to enter a word before clicking the ADD button or REMOVE button, which can improve user's experience.

## 3 Design Details

### 3.1 Client Package

The diagram of classes in the Client Package is shown in Figure 1.

When the client starts to run, it would first check whether the server's address and port are given. After that, it would check whether the formats of parameters are correct. Before the user executes any ADD, DELETE or QUERY operation, the client would not try to check and connect to the server.

When the user tries to execute any of the three operations, the client would first validate the integrity of the information need by the request. Then it would try to create a TCP connection with the server. If the client can not connect to the server, it would advise the user to restart the application with the proper address and port.

When they successfully connect, the client would create a request message in the JSON format shown in Table 1.

Table 1: Client request format (JSON fields)

| | |
|---|---|
| Command | A number indicating different command(detailed in table 3) ) |
| Word | A word for operation |
| Meaning | The meaning of a word(Only need in ADD Command) |

Moreover, the client provides an adaptive window to users which can avoid the Out-Of-Bound text error. Also, some tips would show in the windows when a user tries to do some incorrect operation. The GUI is shown in Figure 2.

### 3.2 Server Package

The diagram of classes in the Server Package is shown in figure 3.

When the server starts to run, it would first check whether the server's port and the dictionary's path are given. After that, it would check whether the formats of parameters are correct and try to read from the dictionary. If the giving dictionary doesn't exist or the format in the dictionary file is not valid, the server would create a default dictionary.

When the server receives a request from a client. It would create a thread to handle the request. According to distinct contexts, the server would handle requests in different ways and send back different states and feedbacks to clients. The response's JSON format is shown in Table 2.

Table 2: Server response format (JSON fields)

| | |
|---|---|
| State | A number indicating the state (detailed in table 3) |
| Meaning | The meaning of a word(Only need in QUERRY mode) |

Besides, the server also provides an adaptive window that can avoid the Out-Of-Bound text error. The use of it is to provide useful information about the server and recording the operation of the system. The GUI is shown in Figure 4.

### 3.3 StateCode Package

The StateCode class is responsible for providing status code comparison tables shared by both the server and clients. The comparison tables is shown in Table3.

Table 3: Status code reference

| | |
|---|---|
| 0 | QUERY Command, used in request. |
| 1 | ADD Command, used in request. |
| 2 | REMOVE command, telling the client the operation can not be executed. |
| 3 | Successful Operation, telling the client the operation is successfully executed. |
| 4 | Unsuccessful Operation, used in respones. |
| 403 | Connection Failure, used in connection. |

### 3.4 Interaction Diagram

The interaction diagram between a single client and the server is shown in

# 4 Critical Analysis

### 4.1 Failure Handling and Recovery

The two major errors that may occur in the system are operation errors and connection errors. For each kind of errors, the system has provided proper notification to help the users understand what goes wrong.

1. *Operation Errors* Operation errors occurs when the user started the program in the wrong way or performed the wrong operation. The problems and their solutions are listed in Table 4.

Table 4: Operation errors and solution

| Description | Solution |
|---|---|
| Inadequate startup paramaters (both sides) | Prevent startup and show sample. |
| Input invalid port number (both sides) | Prevent startup and ask for valid port number. |
| Empty input (client side) | Show GUI tips and ask for input. |
| The dictionary file not exists (server side) | Use default fictionary |
| Default dictionary file not exists (server side) | create default fictionary |
| Get request to add a exist word (server side) | Send word-exist feedback to client |
| Get request to query a non-exist word (server side) | Send word-non-exist feedback to client |
| Get request to remove a non-exist word (server side) | Send word-non-exist feedback to client |

2. *Connection Errors* In the real world, the network is unstable. Besides, the server may not work, and the port may be wrong. The problems and their solutions are listed in Table 5.

Table 5: Connection errors and solution

| Description | Solution |
|---|---|
| Connection Timed Out (client sides) | Show GUI tips and ask for a connectable server. |
| Connectionless Port (client sides) | Show GUI tips and ask for a connectable port. |

## 4.2 Analysis of the System

### 4.2.1 Advantage

1. *Scalability* Based on the feature that users operate infrequently in dictionary application, using Thread-per-request scales better than thread-per-connection. Java threads are rather expensive, typically using a 1Mb memory segment each, whether they are active or idle. In the thread-per-request model, the thread is only associated while a request is being processed. That usually means that the service needs fewer threads to handle the same number of users. And since threads use significant resources, that means that the service will be more scalable.

2. *Heterogeneity* Using Java to implement both the server can solve the problem of heterogeneity. This is because JVM provides basic cross-platform.

3. *Notification of Error* Most of the errors which may occur during the using period have been handle properly. Through giving feedback on the GUI, users can easily find out the error.

#### 4.2.2 Disadvantage

1. *Problem of thread-per-request* It is expensive to create a new thread for each request. The server that creates a new thread for each request may spend more time and system resources on creating and destroying threads than it spends on processing actual user requests. Using worker pool architecture can solve the problem.

2. *Security Concern* Since access to the server is unrestricted, every one can send a request to the server. And there is no data format check on the request. The server can be easily attacked. There is a need to guarantee the privacy, integrity of resources in the server. Besides, the length and the format of word and meaning provided by users would not be check, which may also cause errors.

### 4.3 Creativity elements

1. *Server GUI* A GUI for managing the server is provided. The use of it is to provide useful information about the server and recording the operation of the system. The GUI is shown in Figure 4.

2. *Timeout Handler* In the real world, the network is unstable. Adding timeout mechanism can effectively avoid the problem of program jamming. The relative class's Class Diagram is shown in Figure 6.

3. *Adaptive GUI* Adaptive GUI to users which can avoid the Out-Of-Bound text error are provided on both the server and clients sides.