COMP90015: Distributed Systems

Assignment 1 Report

Student ID: 1114028
Jiawei Luo

1. Project description

In this project, a dictionary system was designed. The dictionary system basically uses a client-server architecture while using a multi-threaded structure to support multiple users at the same time access. Each thread will support one connection from a client. It supports numerous user operations on the dictionary system, including synchronously searching, adding, removing, and updating. This system uses TCP communication between client and server to provide a stable and reliable communication socket. The client-server communication protocol and the server-local dictionary data store are in JSON format, providing simple and reliable data storage and exchange.

Concerning the failure model, the client's incorrect requests will be processed on the server and fed back to the client. At the same time, incorrect I/O exceptions, connected exceptions, JSON parse Exception, and incorrect arguments (including incorrect port numbers, incorrect server address, incorrect dictionary address) will be processed on both the client and server.

2. Project architecture

2.1. server

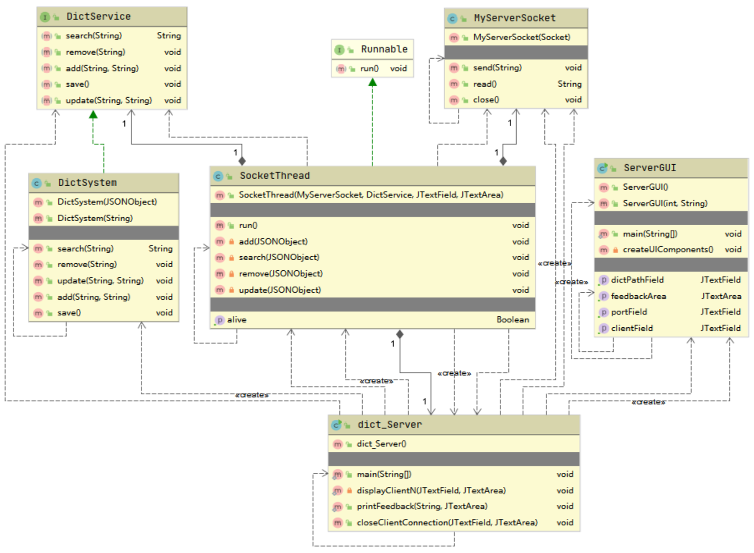


Figure 1. Server class diagram

The class dict_Server is the main class of the dictionary server. When it starts, it first checks whether the console arguments are correct: If no argument, the program will run by default; if there are two arguments <port> and <dictionary-file>, which is the port number and the path of the dictionary data, respectively, it will correctly run; otherwise, the program will show the correct format of running program and exit.

When dict_Server is correctly running, it will create a dictionary instance from DictService and DictSystem, including all operation that regards the dictionary and load dictionary data from the given <dictionary-file> path. The data is store in a JSON file and follow the format:

{"word1":"meaning1","word2":"meaning2"}

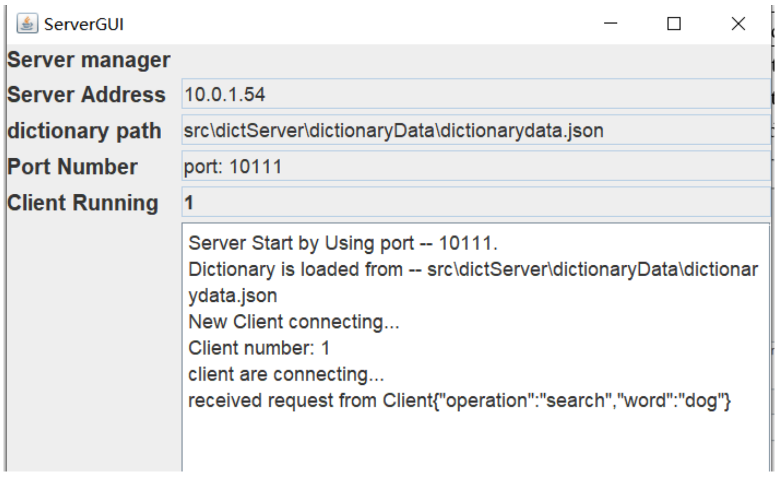
Once the server received a client's request, it created a thread from SocketThread to handle that client's message. The SocketThread basically parses the JSON message from the client and feeds the operation to the dictionary system, which completes the operation before sending a reply back to the client. It sends JSON message to client follow the format:

{status: the request status, content: the content information from server}

status	The status of request; can only be "success" or "fail"
content	The fail or success message from server; or the meaning of Search operation

The SocketThread also handles reading and sending the socket MyServerSocket, a simple program that extends the socket and read and write the socket's output stream.

A server GUI ServerGUI is designed to show information regarding the server running, involving local server address(IP address), local dictionary data path, port number and client number connected to the server. besides, the system response will be shown in GUI.



Server manager

Server Address	10.0.1.54
dictionary path	src\dictServer\dictionaryData\dictionarydata.json
Port Number	port: 10111
Client Running	1

```
Server Start by Using port -- 10111.
Dictionary is loaded from -- src\dictServer\dictionaryData\dictionar
ydata.json
New Client connecting...
Client number: 1
client are connecting...
received request from Client{"operation":"search","word":"dog"}
```

Figure 2. Server GUI

2.2. client

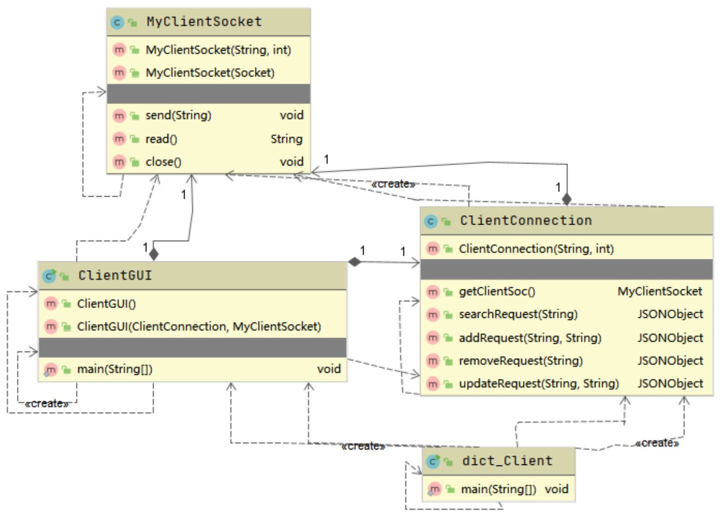

MyClientSocket
MyClientSocket(String, int)
MyClientSocket(Socket)
send(String) void
read() String
close() void
ClientConnection
ClientConnection(String, int)
getClientSoc() MyClientSocket
searchRequest(String) JSONObject
addRequest(String, String) JSONObject
removeRequest(String) JSONObject
updateRequest(String, String) JSONObject
ClientGUI
ClientGUI()
ClientGUI(ClientConnection, MyClientSocket)
main(String[]) void
dict_Client
main(String[]) void
«create»
1

Figure 3. Client class diagram

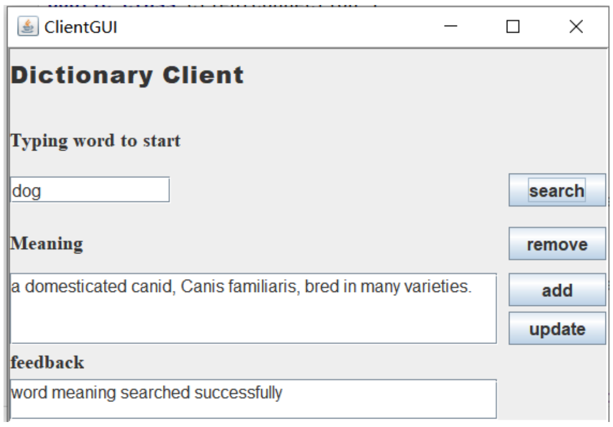
The dict_Client is also the main class of the client program, and it checks the argument follow the format <server-address> <server-port>; meanwhile, it can also be run by default if no argument is provided, but the default running has to be both client and server on the same localhost. Otherwise, the IP address has to be provided.

When the program starts, an instance of ClientConnection is created to generate request to the server, the request should follow the format in JSON:

{"operation":"…", "word":"…", "meaning","…"}

"operation"	The request operation; can only be "add","search","remove" or "update"
"word"	The word that client wants to search, add, remove or update
"meaning"	The meaning of the word, only used for add or update

Also, it creates a socket MyClientSocket instance to send the request to the server, TCP handles the communication, and it is reliable and stable. After receiving the response, the content of the message will be displayed in the client GUI ClientGUI:



a domesticated canid, Canis familiaris, bred in many varieties.

word meaning searched successfully

Figure 4. Client GUI

The GUI has three text field/area, the word field is used to enter a single word, and the meaning area can be used to display and update the meaning to the word. Moreover, the feedback area is not editable, and it shows the system message and the response message from the client.

3. Error handling

Errors can occur on both the client and the server-side. The most common errors are handled. There are four kinds of errors: Input from the console errors, dictionary system errors from disk, Network communication errors, and Client Operation errors. Errors are handled in the program and listed below:

3.1. Input from the console errors

Server command error	Print "With two argument, please Run as -> java –jar DictionaryServer.jar <port> <dictionary-file>" and exit, require run again with correct arguments
Client command error	Print "With two argument, please Run as -> java –jar DictionaryClient.jar <server-address> <server-port>" and exit, require run again with correct arguments
Not input a number as port	Print"The <port> should be port number that in the interval of [1024, 65536]" and exit, require run again with a vaild port

Not an open port error	Print"The <port> should be port number that in the interval of [1024, 65535]" and exit, require run again with a vaild port
BindException in server	Print "port number of address is being use, try a different port number..." and exit, require run again with a vaild port

3.2. dictionary system errors from disk

Server:

Dictionary File Not Found Exception	Print "File can not be found in " + filePath, and exit, require run again with correct file path
Dictionary file parse error	Print "JSON parse error in dictionary system..." and exit, require run again with correct file format
Dictionary IO exception when parsing JSON data	Print "JSON parse error in dictionary system..." and exit, require run again with correct file format
Dictionary IO exception when saving JSON data	Print "I/O operations interrupted in saving dictionary."

3.3. Network communication errors

Server:

if there is an error in the underlying protocol, such as a TCP error---SocketException	Print "Socket error happened"
SocketTimeoutException	Print "Client socket connect time out..."
Bad message from client	Print "UNKNOWN JSON parse error..."
Bad message from client	Print "Wrong operation from client..."
IO exception when sending response to client—happen when client close GUI window	Print "I/O operations interrupted, Client closed."
Null pointer exception, error when get null JSON object	Print "I/O operations closed, Client closed."

Client:

Send request when server closed	""IO error...Server may be closed, try reopen client after server runs...""
Bad message from server.	"invalid response"
Server address is not correct	"Network is unreachable, please use correct address..."
Client start when server close; user a different address	"Server is not reachable, try again after server runs..."

3.4 Client Operation errors

Empty word for search/remove	Print "Please enter you word"
Empty word or meaning for add/update	Print "Please enter both your word and meaning"
Word not exist in dictionary	Send message and display in client: "word not exist in dictionary..."
Adding duplicate word	Send message and display in client: "word already exist in dictionary..."

4. Interaction diagram

The interaction of clients and the server is shown below:

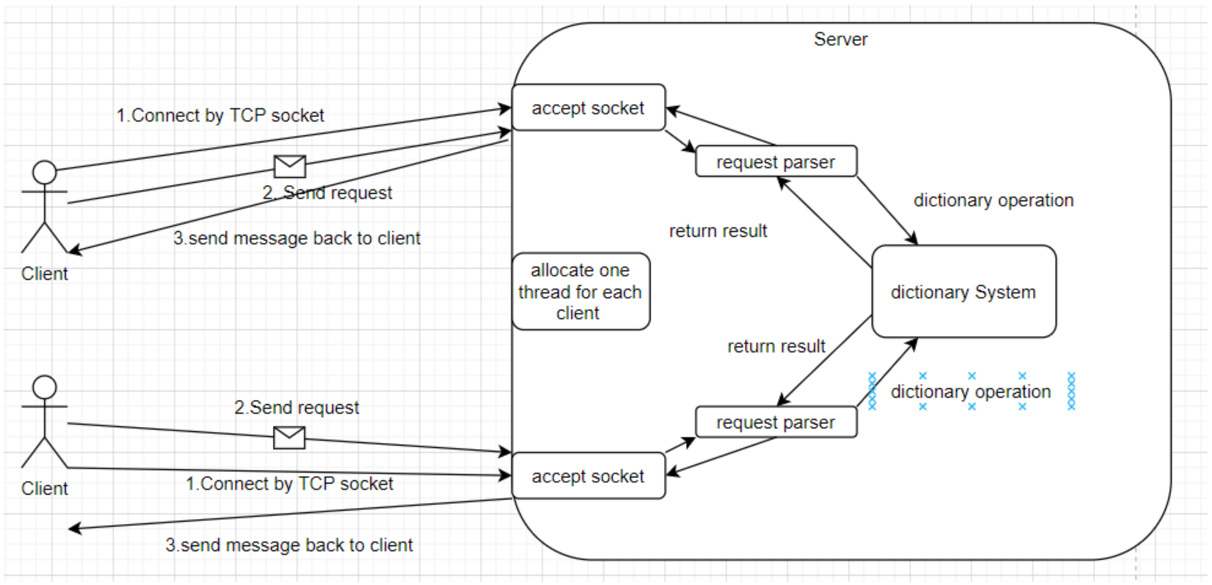


Figure 5. the interaction diagram 1

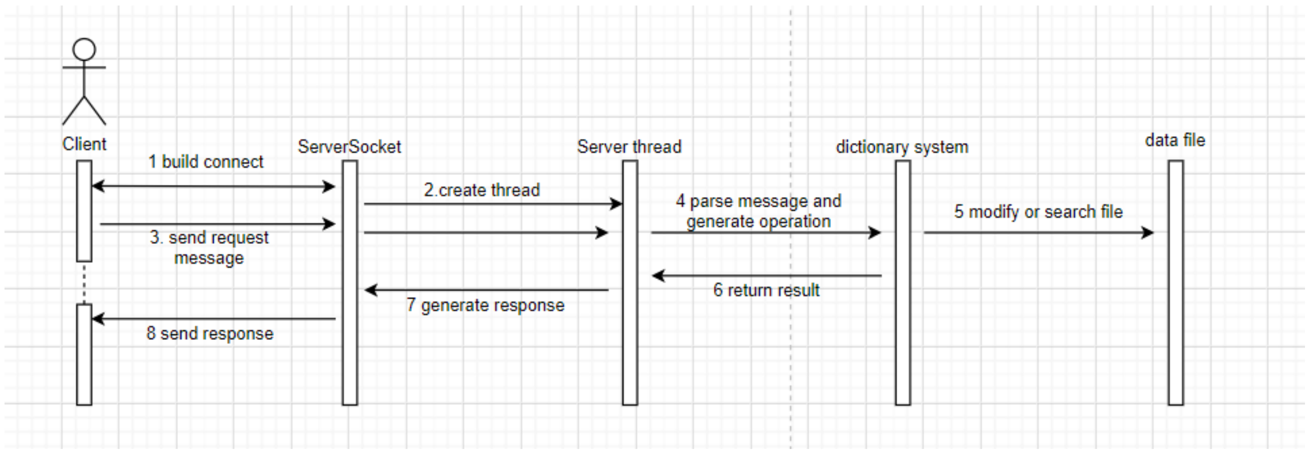


Figure 6. the interaction diagram 2

5. Critical analysis on system

Advantages:

1. Thread per connection: I used thread per connection architecture because it is easy to implement. Compare with Thread-per-request and Thread pool, and the thread-per-connection model has the advantage of being simple and independent. No thread has to wait for another thread to do what it's doing, and it doesn't have to wait for something in the queue it doesn't need. Therefore, client threads can all run at maximum thread speed.
2. Thread per connection: Compared to the thread-per-request structure, thread-per-connection can get better thread resource allocation in the case of multiple clients logging in for a long time and sending numerous requests, and each client can get the same amount of resource.

For the Goals of Distributed Systems

3. Connecting Users and Resources: server can accept socket with correct request format, ip address and port.

4. Transparency: user just need to do operation on the client GUI and wait for response, Server manager can also know the number of clients running, the system information on server GUI
5. Openness: the message protocol is defined using JSON. JSON has very short, readable message style, while the XML message is more complex and long. The data type is not very important in my program, that is why I choose JSON.
6. Scalability: thread-per-connect only start a thread when a client is connecting, it is scalable when several clients sending a large number of requests.
7. Availability: TCP connection provided by java is reliable and stable.
8. Error handling: Errors are handled by using messages on client GUI and feedback on Server GUI in multiple ways

Disadvantages:

1. Thread-per-connection architecture can not efficiently and selectively allocate memory when many clients are running at the same time and result in huge thread resource waste. Compared with thread-per-connection, thread-pool has more advantages in managing thread allocation, which is more in line with the reality of client-server structure.

2.   If each client sends only a small number of requests or one request, it will be as efficient as thread-per-request, which is low.

3. Safety: This message transfer is not encrypted so that anyone can send a request to the server to get the information. In practice, both the server and the client should set up message encryption protocol to prevent some people from stealing and attacking the server information.

6. Creativity

1. A GUI for server to display the system information, such as dictionary path, port number, address, and client connecting.
2. A real-time client number display is show in GUI, that demonstrate the number of clients online, once a client offline, the number will change.

7. Conclusion

Overall, this multi-threaded dictionary system allows multiple users to add, query, remove, and update dictionary content concurrently and give the client feedback. This system needs to be upgraded in the future to solve some concurrent scheduling problems in the current system architecture (thread-per-connection) to improve the system’s concurrency efficiency and throughput. More interesting features can be added to the system, and the GUI interface needs to be more aesthetically pleasing.