Assignment 1
Project report: Stage 1
COMP3100 Distributed Systems, S1, 2023
SID: 46131434, Name: Ravi Inder Singh

Contents

1 Introduction **1**

2 System Overview **1**

3 Design **3**
3.1 Philosophy 3
3.2 Constraints and considerations 3
3.3 Functionalities of the Client 3

4 Implementation **3**
4.1 Technologies used 3
4.2 Data structures, Techniques and software libraries used 3

5 References **4**

1 Introduction

This is stage 1 of the project. The project focus on the working of a scheduler in distributed systems. Distributed systems are network of various computing components working together as if a single system. This environment is helpful in resource allocation while scheduling set of jobs. It ensures the system availability and fault tolerance due to readily availability of other nodes/computers in case of breakdown [1].

This stage investigates the working of server and client using client side simulator. Server gives the list of jobs and client has to do the task of job scheduler.It is achieved using simple yet effective method by using Largest-Round-Robin (LRR) algorithm. In this algorithm, first server with the highest number of cores is chosen and all the jobs are scheduled on the servers of that type [2].

The rest of this report is organised into 5 sections. Section 2 gives high level of understanding on the working of the project and technical aspects of it. Section 3 talks about the computing constraints of the project and design philosophy. Section 4 is all about the implementation, technologies and development libraries used. Finally we have list of references and link to **GitHub** repository.

2 System Overview

This section gives a high level of description of the project and ds-client simulator in particular. For the stage 1, ds-server is already pre-compiled and provided in the project folder from the GitHub repository. However, students are expected to produce a client which imitates the working of ds-client i.e., schedule jobs.

First, server and client needs to set up a communication channel. By default, they communicate using sockets and on port 50000. After the connection is established, three-way handshake algorithm is used to initiates the conversation. Client sends 'HELO' to the server. To which server responds with 'OK'. Followed by 'AUTH username' and then server sends welcome message to the user.

Now client is ready to schedule the jobs for the server. It asks for the job using 'REDY' keyword. Server promptly reply with the JOBN a b c d e where 'a' is start time, 'b' is job ID, 'c' is number of cores required, 'd' and 'e' are memory and disk requirements respectively. Now client can check number of servers available on the main server and their state using 'GETS all' as shown in figure1.

By studying the capacity and states of the servers, client can schedule jobs and allocate resources accordingly

using LRR algorithm. This can achieved by command 'SCHD x y z' where x refers to job Id, 'y' is serverType and 'z' refers to the serverID of that type of server.

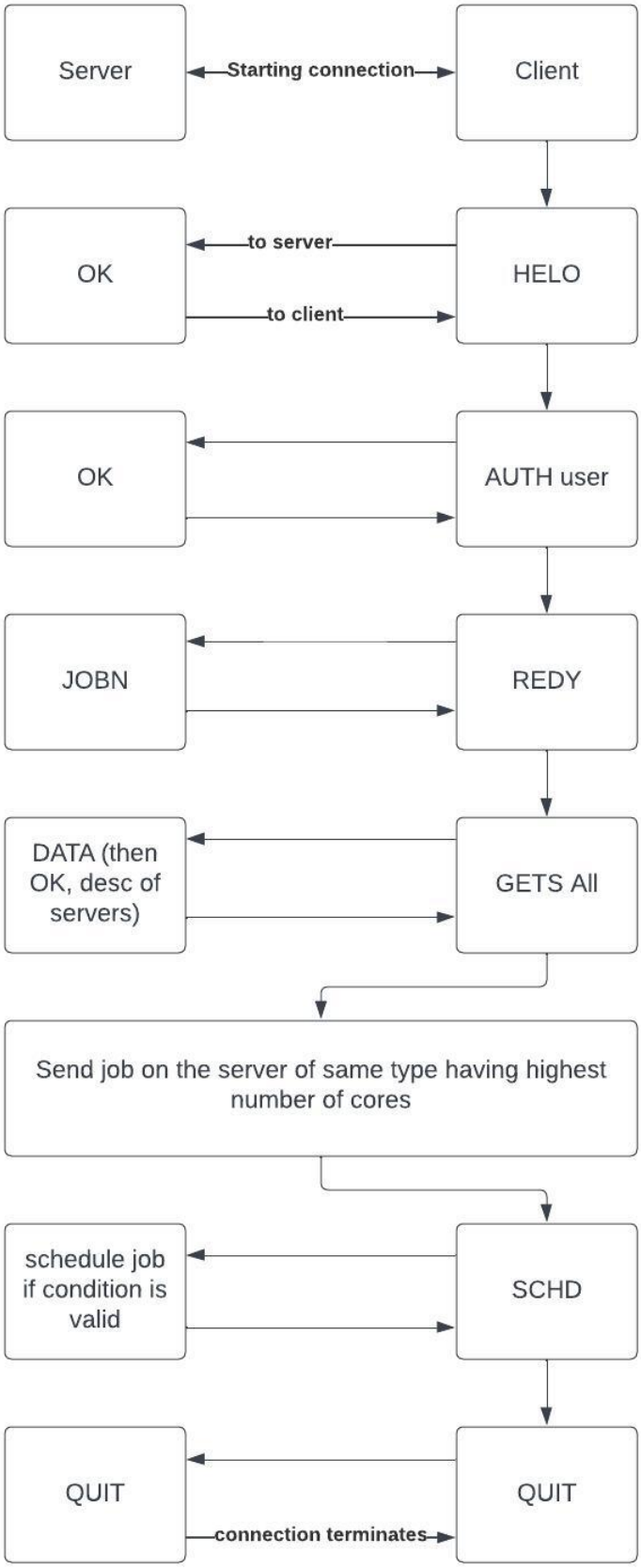

Figure 1: Workflow

3 Design

This section gives the overview of design philosophy, constraints for the projects and its functionalities.

3.1 Philosophy

The design is straightforward and easy to follow. The design philosophy behind was to make a neat and descriptive code that can be easily revisited and understood in quick span of the time. It is well known that memory consumption of the program plays crucial role in its efficiency. Therefore, the code is more focused on making efficient use of the data-structures. This code is written while considering further expansion of the program in the stage 2 of this project.

3.2 Constraints and considerations

The constraints have played a vital role in the code management. These helped to get rid of the redundant code and magic numbers. Proper naming conventions are used which are short but descriptive enough to improve the code readability. Moreover, indentation style has been taken into consideration while writing the code.

3.3 Functionalities of the Client

At this stage of the project, pre-complied ds-server is provided with project repository. It acts as job dispatcher to the client. Goal of the stage 1 is to develop client alike ds-client using java. Firstly, the javaClient set up the connection with the server using three-way handshake on the default (mutual) port numbers. Then, it receives job instructions from the server using set of keywords. These jobs are assigned to the server having highest number of cores and are distributed among the servers of the same type using Largest Round Robin (LRR) algorithm. JavaClient act as if the jobs were scheduled using pre-compiled ds-client.

4 Implementation

The program is implemented using various resources such as different set of technologies, techniques, in-built software libraries and lecture notes.

4.1 Technologies used

Used Linux based OS Ubuntu on Oracle virtual machine which is installed on Intel powered Windows 11 device. Efficient use of sharedFolder was made to transfer files or from Windows to virtual machine. IDE such as Visual Studio Code helped in code indentation.

4.2 Data structures, Techniques and software libraries used

Proper use of data structures such as arrays and strings help in saving data from the incoming data stream. These strings were later processed using split function. The resulted strings are then passed to the loop or other functions accordingly. These strings form the major part of the processing the request as these encloses the specific structure of the command followed by first keyword. Identifying these different set of strings solves the majority part of the stage 1.

Socket programming helped in the communication of server and javaClient using port 50000. The three-way handshake is crucial in setting up the connection and they were able to exchange messages using set of 4 alphabet keywords. In-built libraries such as Java Net and I/O provides classes to create sockets along with data input and output streams.

Challenge in this stage is to figure out the largest server and to know the number of its types. This was accomplished using clever design and making use of Boolean variable which acted as one way switch once the program starts. This saves the computing power and resources by not allowing the program to evaluate the server with highest number of cores, again and again. Once this is set-up, then next part has to deal with the job scheduling. Overall, java split() method is quite useful in splitting the messages and strings into arrays and from these data structures information can be easily extracted.

I have made sure, using a Boolean condition, that the GETS All run only once. I have used two string

arrays. First one to handle the strings of the sentences such as ”super-silk 0 inactive -1 8 4000 16000 0 0” and second string array to store and access individual words such as ”super-silk”, ”0” etc. This makes easier to access the required specs and can be assigned to the variables. The values then can be easily used in client messages such as ”SCHD jobId LServer LServerID”. All the **JCPL** messages must be ignored and only **JOBN** needs to be assigned. After successful scheduling of the jobs, server sends the message **NONE** and then the while loop terminates, followed by closing output stream and the socket.

5 References

Link to the **GitHub** project is https://github.com/Ravi-mq/COMP3100DSAssignment.git

References

[1] Splunk, “What are distributed systems?.” https://www.splunk.com/en_us/data-insider/what-are-distributed-systems.html, 02 2021.

[2] A. Arshad, “What is the round-robin load balancing technique?.” https://www.educative.io/answers/what-is-the-round-robin-load-balancing-technique.