Project report: Stage 1
COMP3100 Distributed Systems, S1, 2023

SID: 47118318, Name: Max Frohlich

1 Introduction

Ds-Server is a means of communicating data between two devices across a network, these connections let computers talk to each other and send data. This requires a Ds-server and a DS-Client. This project will be focused on the connection between two connections, in this case, two terminals that will be acted as two computers. This project will be focusing on a simple job dispatcher/scheduler, which is called Largest-Round-Robin, this will send a job to the server. [1].

In this report, this report will describe the aim to program a Ds-server and a Ds-CLient using socket programming that can connect to each other and will send and receive jobs. this will be a vanilla implementation and the client will act as the job. This report will also go into many parts of the process of making and implementing this piece program, the first section will handle the system overview where it will describe the high-level description of the system. This will show the work flow of the system and show the working of it. the second section will describe the design where the functionalities and constraints of the simulator. the implementation section will describe the technologies, techniques, and software libraries used to make the Program and how each of the components/functions of this simulator is to be implemented. The rest of this report is organized as follows. Section 2 gives an overview of ...

2 System Overview

This system will have a Client-side and a server-side, on which will be called client and server. In this system, the client side will be connecting to the server and making where the client will send over ”jobs”, This will use the largest round-robin algorithm, which will send the job to the largest type in the same way as LRR.

The flow on which will this work is as follows:

1. first the Client and server will need to establish a connection with each other in order to start communicating.
2. The client will then send a ”HELO” to the server and will then respond with an ”OK” if everything is in order.
3. The client will then need to send an AUTH message with a username to login to the server, after this is successful it will respond with an ”Ok” message.
4. The Client will need to send a ”REDY” message to the server, to start to begin the job dispatching process, the server will respond with an ”OK”.
5. The client will need to also identify the largest server by sending a ”GETS -a” command so that the server can respond with a list of servers available and then all their functionalities with them.
6. The Client will need to find a choose the correct server type with the most amount cores in them.
7. Then the ”SCHD” command will be used to message the server and have the selected job details.
8. the previous steps will need to be done over and over again until all the jobs have been dispatched.

These are all the steps that will need to be taken in order to achieve all the jobs and dispatch them all. this can take time because the client will have to go through each job one by one. This connection will go over port 50000 as it is the default port.

3 Design

The design of this system should be as efficient as possible at doing the job scheduling, this should also be able to scale well because it can grow in size easily. The client should be able to schedule the jobs that will be sent to the server, where then the server program should be able to send these job requests easily.

3.1 Sockets

There are many considerations that will need to be implemented to make the design of this scheduler work well, first it will need to be coded with socket communication, this is so that the server and client can communicate with each other, socket programming is good because it can be useful for both the stand-alone and network applications. these sockets let the machine communicate between networks and distribute the information to the most efficient machine. (1) Socket programming is one of the most fundamental parts of the operating system when it wants to communicate over a network. generally, these sockets are provided by the client-server applications where the client will create a socket and then attempt to connect to a network where it has a server socket, once the connection is established it will start to transfer data. there are two types of sockets which are datagram sockets and stream sockets, there are many more but are more specialized.(3) This program will use data datagram sockets because this type of socket will allow being used in two-way communication between the server and the client. Stream sockets allow communication with the use of TCP this socket is very reliable and allows two-way authentication which is also a good choice of a socket.

3.2 Client-side

One of the most important design philosophies would be the job scheduling itself, where the client receives the job and will need to schedule each of them, this will need to make it as easy as possible for both the client and the server so that both can run as fast as possible. this job scheduler should also run on the Largest round robin scheduling algorithm, this algorithm hosts a number of advantages that can help the whole process of this code.

1. All the jobs are added to a queue that will be eventually processed.
2. the time to complete tasks is sometimes fast as a quantum CPU.
3. on completion the task will terminate, which leads to more efficient work for other tasks.

Overall there are many advantages to the LRR algorithm and should be used when thinking about the design of this job scheduling program.

3.3 Scale-ability

The Client should be able to handle multiple different server types so that it can be easily scalable and easy to implement with other machines. There should also be some sort of authentication added to the code so that the client can send a message with a username so a user can be authenticated. This can make it easier for a person to record records of what a person has done and can let progress be saved to the user so that the person can continue it later if they choose to do so. (2)

3.4 Further improvements

There can be many improvement's to the code and the overall design of the program itself. one of the improvements that could have been added could be the use of a different language, like C, C, or even python. This is because each language can provide many different use cases that could make it easier to code or make it more efficient at the job it is supposed to do. C can be used if you would want to make it very efficient because of how low level it is, this can make the server and client run faster so it can do the job faster. Python can be used to make understanding the coding the logic of the sockets easier because python is easier to code in.
Criticism/reflection and improvement suggestions

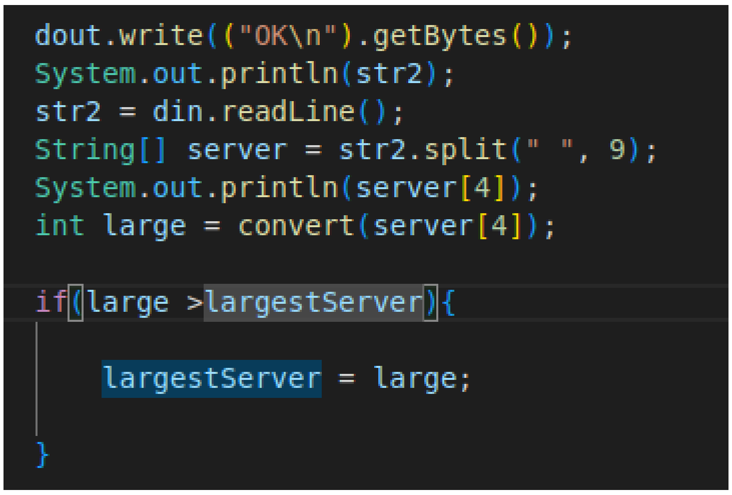
4 Section Implementation

4.1 Starting up a server

Largest Round Robin The first start that should happen is making a vanilla server so that the server and the client can connect to each other in the first place. As stated above socket programming will be used to implement a connection between each other. the first implementation would be the use of imports which are ”import java.io.* ;” and ”import java.net.*;” where the java.net.Socket this will allow the program to know socket-related functions so that it can be coded. once that is done the client and server will need to know which port to send information, this can be from a range of 0-65535 where the default port on which also should be left is 50000, other parts of the socket address would be by (TCP, hostname,1028). further, there will be a need for BufferedReader and DataOutPutStream classes that are used to read and write the data to the socket streams. all of these are used for both the Client and the Server side so that they can connect and interact with each other so there can be a handshake. these are vital to the program and are the foundation of all the next implementations to the server and client. (4)

4.2 Largest Round Robin

The largest round-robin (LRR) is a scheduling algorithm widely used scheduling algorithm that is used in multitasking, which ensures a process that will allow it to run efficiently. This will be used in the system so that it can run efficiently which is important for the server and the client. to do this, there will need to be a check for the number of servers running, to do this, Nrec will need to be extracted from the reply from the server this is gotten by using the ”GETS All” command and will come back with a number of records and the record length. There will be a need to first separate these two numbers by using the .split() function that will split it into a string array and then convert the Nrec into an integer using .parsint function, this will then have the number of records to use in a loop to go through each record.



```
dout.write(("OK\n").getBytes());
System.out.println(str2);
str2 = din.readLine();
String[] server = str2.split(" ", 9);
System.out.println(server[4]);
int large = convert(server[4]);

if(large >largestServer){

    largestServer = large;

}
```

The use to find the largest server
Once the number of records is fetched, this will need to be used to find the biggest server by the number of CPU cores. To do this there will be a need for a for loop that will go through each record and then split it into a string array again consisting of 9 parts to the array. where there will be a need to get the 4th item in the array to extract the core count of the server. This will also need to be converted from a string into an integer again so that there can be manipulation with it again.

4.3 next one

Heading 1	Heading 2	Heading 3	Heading 4	...
A	\$100.0			
B	$\sim$ \$93			
C	10^2			
D	C_2			
E	85%			

Table 1: Average Performance.

References

[1] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. D. Joseph, R. Katz, S. Shenker, and I. Stoica, "Mesos: A platform for fine-grained resource sharing in the data center," in *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation*, NSDI'11, (USA), p. 295–308, USENIX Association, 2011.

1) https://www.ibm.com/docs/en/i/7.1?topic=communications-socket-programming 2) https://www.scaler.com/topics/roun
robin-scheduling-in-os/ 3) https://www.geeksforgeeks.org/socket-in-computer-network/ 4) https://www.ibm.com/docs/en/z
understanding-sockets 5) https://www.researchgate.net/profile/Lipika-Datta/publication/281615591$_{E}fficient_{R}ound_{R}obin_{S}$
$Round - Robin - Scheduling - Algorithm - with - Dynamic - Time - Slice.pdf$