# REPORT 1: CLIENT SERVER PROGRAM

Department of Mathematical Sciences
Computer Science Division
University of Stellenbosch
7600 Stellenbosch
August 2022

## INTRODUCTION

This report server to detailed information on the implementation of a client server program using Java programming language. This project was developed in a time span of 2 weeks and illustrates the general concept of server client communication. The final project is a chat program where several clients can connect to a server and send and receive messages.

## PROJECT MEMBERS INFORMATION

### Group Members:

[Takunda, Charles, Mudima, 23969156]
[Ben, Sibusiso, Mahlangu, 23629428]

## INTRODUCTION / OVERVIEW

The client-server program aims to establish connections between a server and multiple clients running simultaneously using sockets. It constitutes of three main classes namely the Server.java, ClientHandler.java and Client.java. These files work concurrently to establish a network of receiving and sending texts between online clients.

## UNIMPLEMENTED FEATURES

- Handling duplicates: The current program does not cater for duplicates, therefore clients with the same name can be registered on the server.
- Client GUI: The current program does not support a GUI interface for the client. The program only allows for communication on the terminal.

## ADDITIONAL FEATURES IMPLEMENTED

- Server shows the number of clients connected to the server: With the use of a counter, our program prints out the number of connected clients online.
- Server shows an updated list of people who go offline: With the use of an array, when a client enters "exit" they are added to the array of people who are offline.

## DESCRIPTION OF FILES

Our project contains the following three files:

***Client.java*:** Contains a function that handles the threading of messages from the client perspective.

***ClientHandler.java***: Contains several functions that handles the receiving and broadcasting of messages from the server side

***Server.java*:** Initialise the Client Handler class object and performs threading on it.

## EXPERIMENT 1

### TITLE

The relationship between the rate at which messages are sent between multiple clients from one server.

### HYPOTHESIS

As we increase the number of clients running on the program, the rate at which these clients receive messages from each other decreases as we keep the server as constant.

### VARIABLES

- Independent variable: Rate at which messages are received(mb/s)
- Dependent variable: Number of clients in use
- Controlled variable: Server in use

### METHOD

1. Compile and run your java files
2. Run the server
3. Run two clients simultaneously, the record the time taken for the server to send the message to the clients
4. Close all your running files.
5. Repeat step 3 and 4 but increase the number of client's every time you do step 1

## TABLE OF RESULTS

Table 1: Results obtained from experiment 1

| SERVER | Rate at which messages are received(mb/s) | Number of clients in use |
|---|---|---|
| 1 | 120 | 2 |
| | 114 | 4 |
| | 112 | 7 |
| | 110 | 10 |
| | 110 | 12 |
| | 108 | 16 |
| | 103 | 20 |

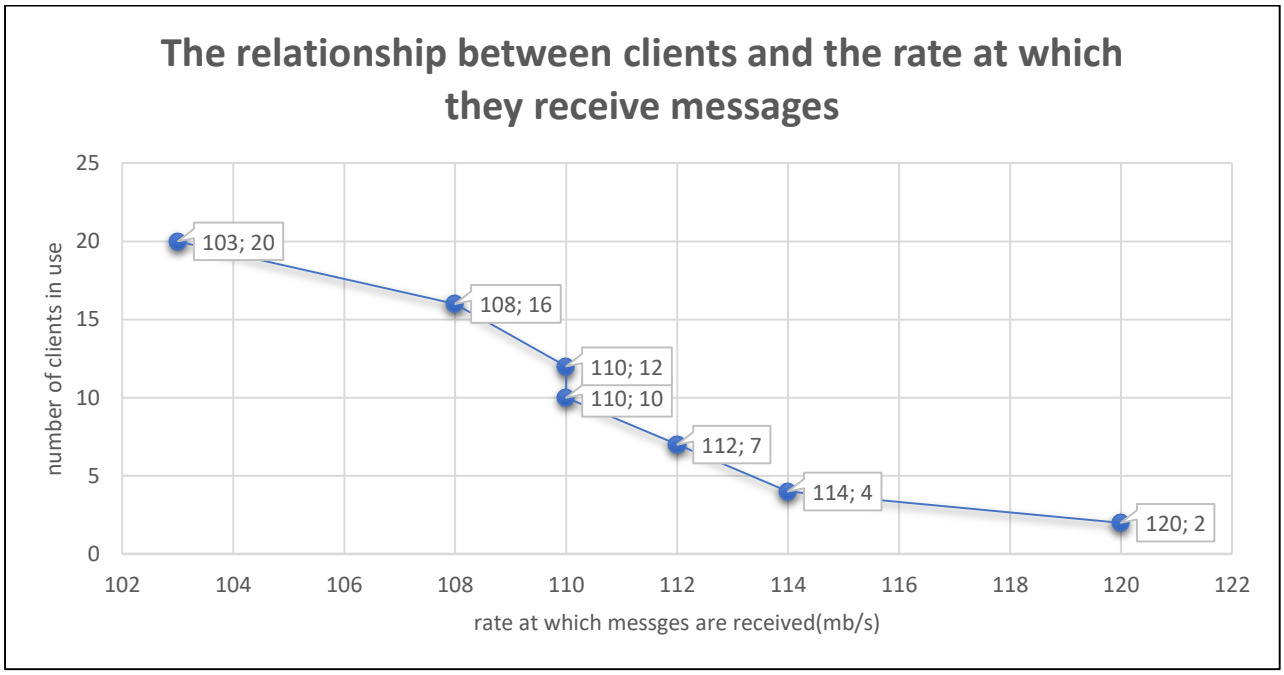


Figure 1

## CONCLUSION

As the number of clients running in the system increase, the rate at which they receive the messages from other clients decreases while keeping the server as a constant factor.

# EXPERIMENT 2

## TITLE

The relationship between a different sets of RAM size, OS types and processors with the rate at which messages are sent between multiple clients from one server.

## HYPOTHESIS

As we increase the size of the RAMs, use faster processors and bigger OS types, the rate at which multiple clients receive and sent their messages increases while the server is kept constant

## VARIABLES

- Independent variable: Different set of RAMs, processors, and OS types
- Dependent variable: Performance types (fast, average, and slow)
- Controlled variable: Server in use

## METHOD

1. Compile and run your java files
2. Run the server
3. Run two clients simultaneously, the record the time taken for the server to send the message to the clients.
4. If it was fast, assign that set a score of 80, if it was average assign that set a score of 50 then lastly if it was slow then assign that set a score of 25.
5. Close all your running files.
6. Repeat step 3 and 4 but use a different set of OS type, processor and RAM size from the set that's been already used every time you do step 1

## TABLE OF RESULTS

KEY: note that the scores were randomly assigned to give the types of performance a numerical value.

Table 2: Performance scores

| PERFORMANCE | SCORES ASSIGNED |
|---|---|
| FAST | 80 |
| AVERAGE | 50 |
| SLOW | 25 |

Table 3: Specification comparison table

| | OS TYPES(BITS) | Processors | RAM SIZE(GB) |
|---|---|---|---|

| | 64 | 32 | i7 | i5 | i3 | Intel | 4 | 8 | 16 |
|---|---|---|---|---|---|---|---|---|---|
| RESULTS | FAST | AVERAGE | FAST | FAST | AVERAGE | AVERAGE | AVERAGE | FAST | FAST |

**CONCLUSION**

The bigger the RAM size, the faster the processors and the higher the OS type, the faster the chat program can handle multiple client's communications

## ISSUES ENCOUNTERED

- Client messages being printed on the server
- Hamachi disconnecting itself while in use
- Implementing threading on the messages received
- Clients crashing after one client had left the chat
- Clients crashing after disconnecting the server

## SIGNIFICANT DATA STRUCTURES

1. A vector data structure, which through its dynamics was able to help us store our ClientHandler objects without manually changing the dimension of the vector since it can grow or shrink on its own.
2. An Arraylist data structure, this simplified the way we broadcast the number of clients that are offline on the server.

## COMPILATION

Explanation on how to compile the program:

1. Firstly, ensure that you have the files in the same folder/directory
2. Run javac *.java to compile all the files
3. Java Server to run Server
4. java Client to run Client

## EXECUTION

To run whispering the client that wants to whisper would have to type the following, <ClientToWhisperTo>**[Message]

# REFERENCES

Wittcode (2021) *Java Socket Programming-Multiple Clients chat.* [online video] Available at: https://youtu.be/gLfuZrrfKes. [Accessed 30/06/2022]

GeeksforGeeks (2021). *Socket Programming in Java. [*online].GeeksforGeeks. Available at: https://www.geeksforgeeks.org/socket-programming-in-java/?ref=lbp. [Accessed 30/06/2022]

Usman Saleem, 2002. A Patter/Framework for Client/Server Programming in Java. Available at: https://www.developer.com/java/enterprise-java/a-pattern-framework-for-client-server-programming-in-java/ [Accessed: 29/07/2022]