ASSIGNMENT REPORT

MASTER OF TECHNOLOGY

IN

COMPUTER SCIENCE & ENGINEERING


**Under the Guidance of:**

Prof. Aaditeshwar Seth
Assot. Professor

CSE Department

**Submitted by:**
**dcode_boys**

Ankit (2023MCS2485)
Karan Agrawal (2023MCS2479)

Sourabh Tiwari (2023MCS2487)
Annanya Mehta (2023MCS2484)

**INDIAN INSTITUTE OF TECHNOLOGY, DELHI**

**Hauz Khas, New Delhi -110016, India**

# Report on Client-Server Communication System

## Introduction

The client-server communication system presented in this report aims to demonstrate a practical implementation of network communication between a client and a server. This system is designed to transfer data efficiently and reliably between the two components, simulating a real-world scenario.

## Code Overview

## Client Code (master code)

The client code serves as the master controller for this communication system. It initiates and manages connections to the server and handles data transmission and reception. Key functionalities include:

- Initialization: The client code initializes necessary variables, including the server's IP address and port, and opens a file for writing server responses.

- Thread Management: The code utilizes threading to handle multiple tasks concurrently. Threads are created for retrieving files from the server and for connecting with the master server.

- File Retrieval: The client code connects to the server and retrieves data in chunks of lines. It ensures that data is received and written to the file efficiently.

- File Submission: After retrieving data, the client code sends the data back to the server. It follows a structured format for submission and monitors the submission process.

- Efficiency Plotting: An optional feature is provided for plotting the efficiency of data retrieval over time using Matplotlib.

Server Code
The server code runs on a separate machine and handles incoming connections from clients. It processes requests and sends data back to clients.

Key functionalities include:

- Connection Handling: The server code accepts incoming connections from clients and manages them in separate threads to handle multiple clients concurrently.

- Session Reset: The server supports a "SESSION RESET" command to reset the session for new data transfer.

- Line Data Reception: It receives and processes line data from clients, checking for duplicate lines and updating a flag table to ensure efficient and reliable data reception.

- Message Sending: The server sends messages back to the client for specific line numbers as requested by the master code.

- File Submission: After receiving data from the client, the server combines and processes it for final submission to the master code.

Communication Flow

1. The client establishes a connection with the server using the server's IP address and port.

2. It sends a "SESSION RESET" command to the server to prepare for a new session.

3. The client then requests line data from the server using a "SENDLINE" command and waits for a response.

4. The server sends the requested line data, which the client receives and writes to a file. It also updates a flag table to ensure that duplicate data is not written.

5. Concurrently, the client connects with the master code running on a different thread. It sends line numbers to the master code for message retrieval and waits for messages to be sent back.

6. The master code processes the received line numbers, retrieves messages, and sends them to the client.

7. Once all data is received, the client combines it and sends it back to the server for final submission.

8. The server processes the submission and sends a confirmation to the client.

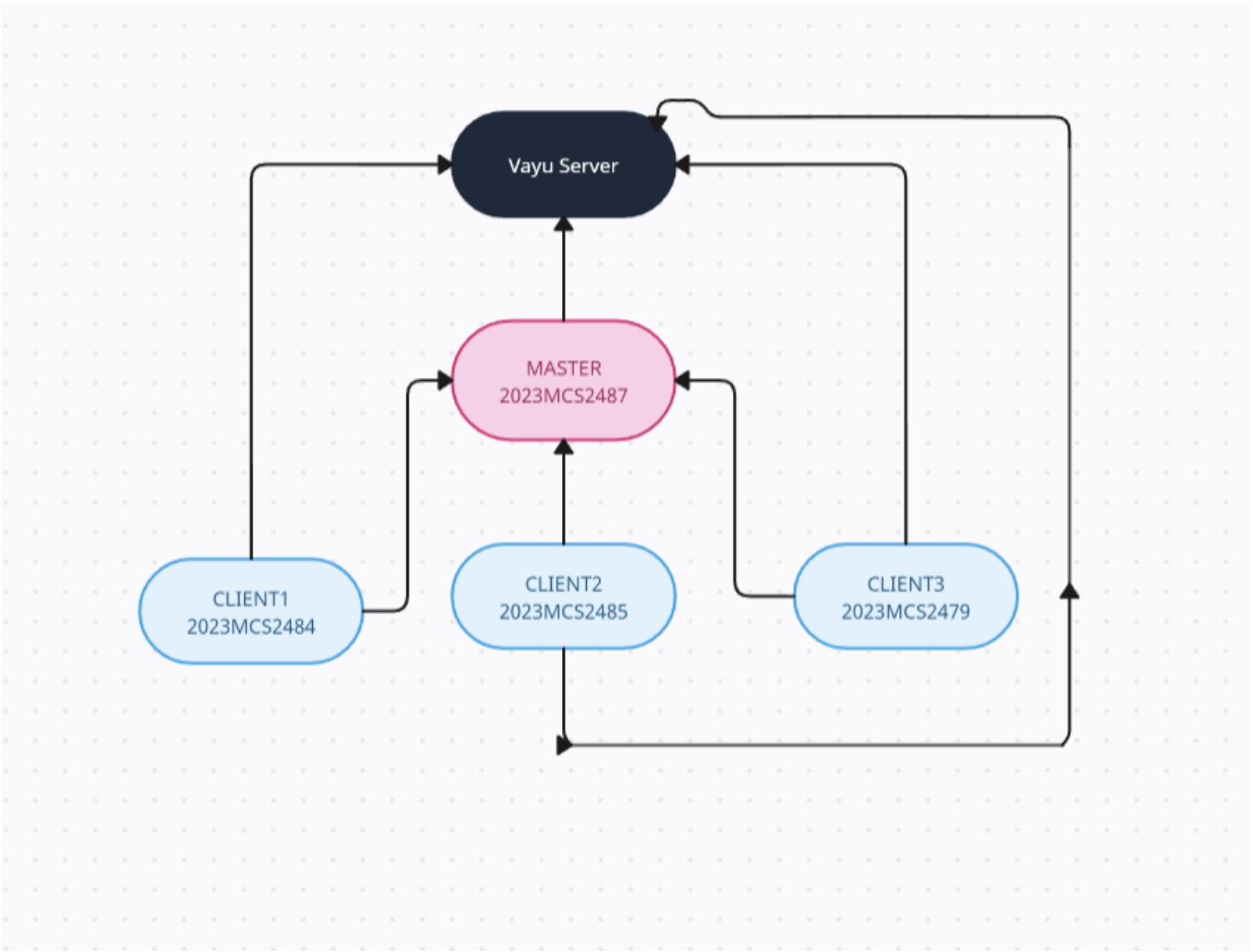
## Master Slave workflow

**Master (Client Code):**
- Controls the entire communication process.
- Manages tasks like data retrieval and connecting with the master server.
- Distributes specific tasks to the server (slave) component.
- Collects and processes responses and results from the server.

## Slave (Server Code):
- Accepts and handles incoming connections from the client (master).
- Executes tasks assigned by the master, such as serving data and messages.
- Responds to client requests and communicates results back to the master.

This summarizes the utilization of the master-slave pattern in the client-server system, enabling efficient and concurrent data exchange.

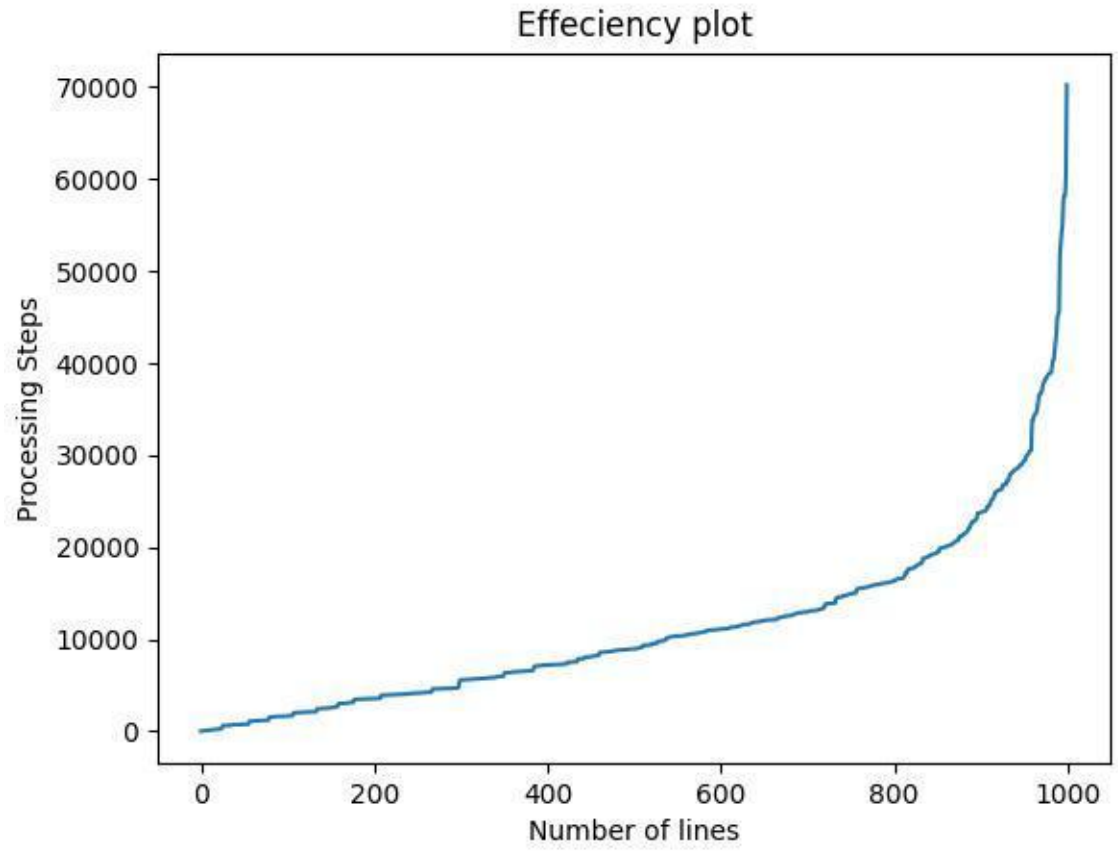
# Client Server model



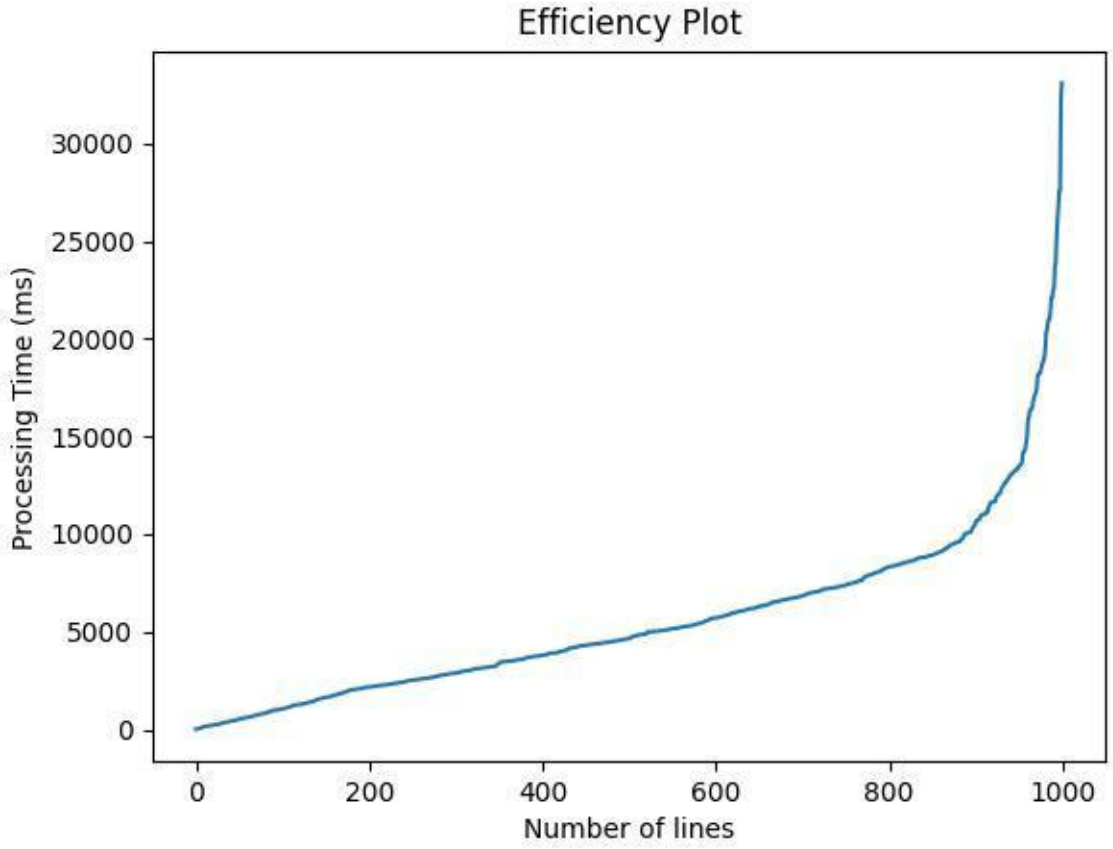
# Running time Graphs

- **When only one master is used**

  It takes approximately 70 seconds.

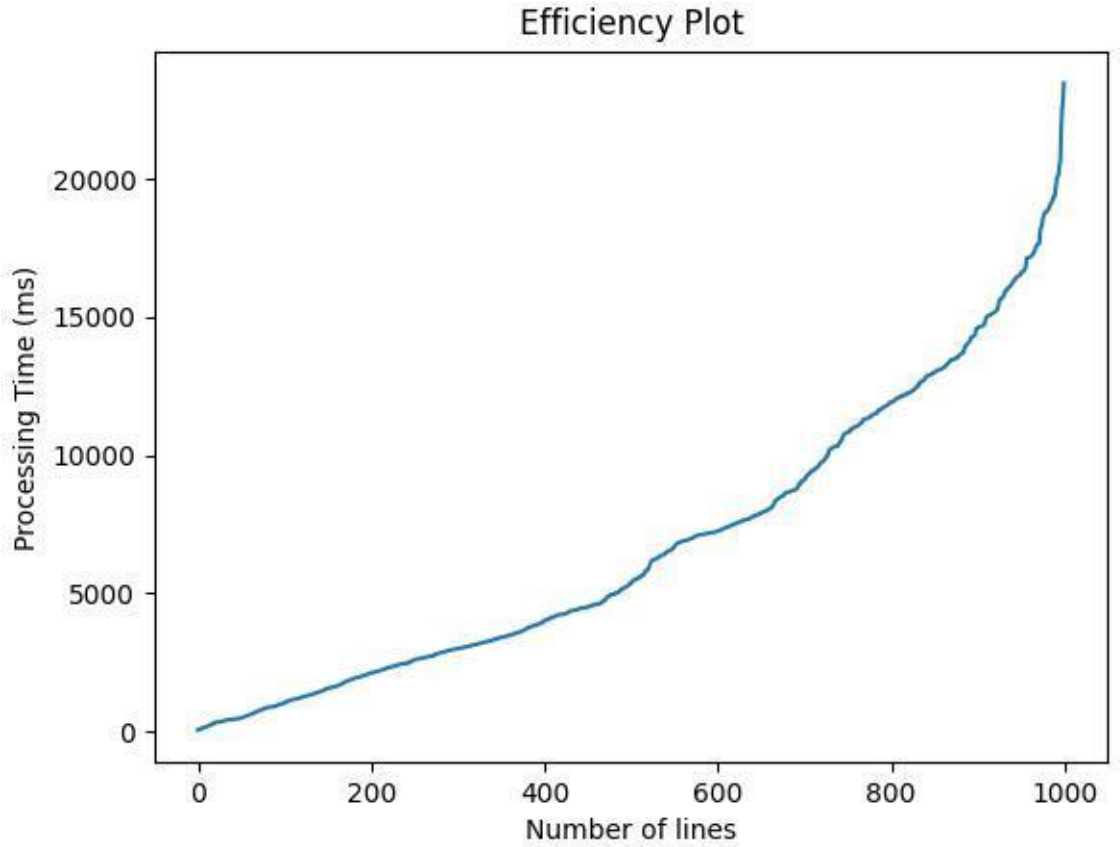


- **When one master and 1 client is used.**
  It takes approximately 34 seconds.



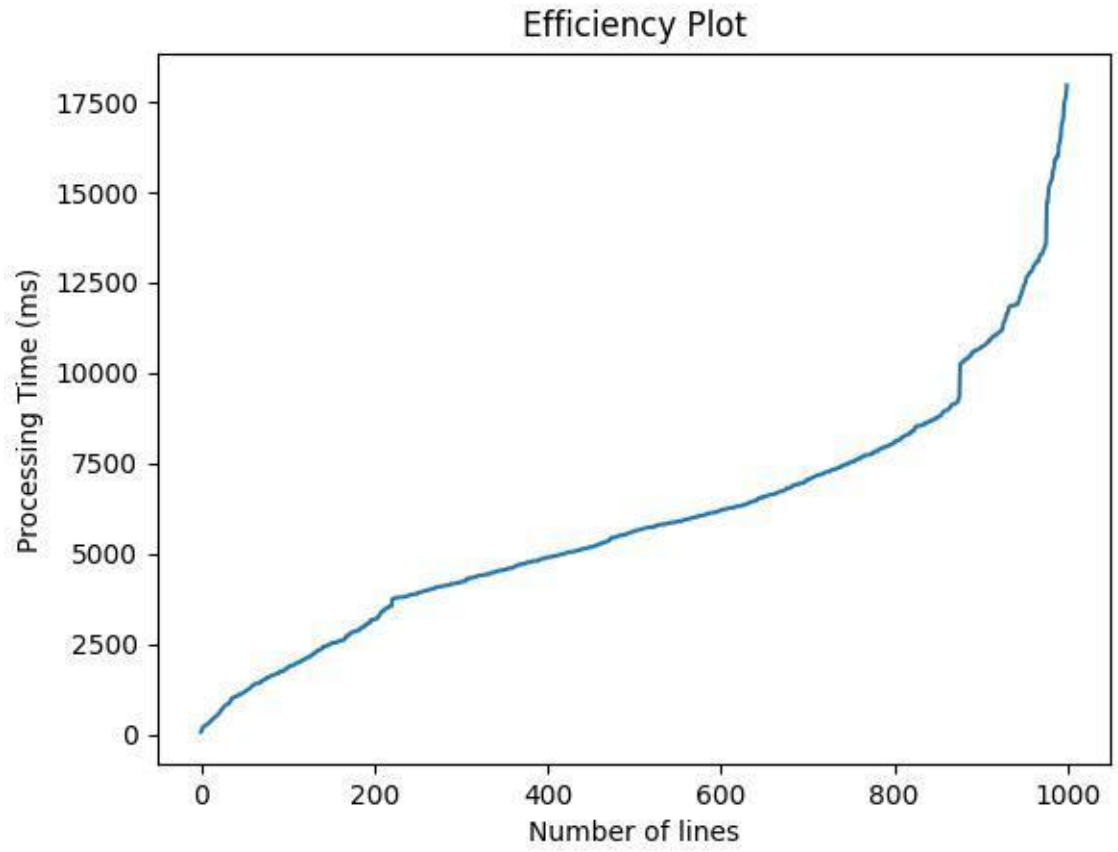
- **When one master and 2 clients are used**

  It takes approximately 24 seconds.



- **When one master and 3 clients are used**

  It approximately takes 18 seconds.




As we increase the number of clients the time decreases is represented by the below chart.

# Conclusion

The client-server communication system successfully demonstrates the efficient transfer of data between a client and a server. It showcases the use of multithreading to manage concurrent tasks, efficient data reception, and message exchange. Additionally, the system provides an option to plot efficiency over time.

This system can serve as a foundation for more complex network communication applications and can be further extended and customized to suit specific use cases and requirements.