24 January 2024

CS 558: Computer Systems Lab

Relay based Peer-to-Peer System using Client-Server socket programming

Project Report (Assignment 1)

Group 5

Aditya Deshmukh	234101004	aditya.deshmukh@iitg.ac.in
Akshay Bhosale	234101006	a.bhosale@iitg.ac.in
Ritik Kumar Koshta	234101044	r.koshta@iitg.ac.in
Shardul Nalode	234101049	n.shardul@iitg.ac.in

Title: Implementation Relay based Peer-to-Peer System using Client-Server socket programming

Abstract:

This report outlines the design and implementation of a Relay-based Peer-to-Peer System using three C programs: Peer_Client, Relay_Server, and Peer_Nodes. The system involves the establishment of connections, exchange of information, and file retrieval among the components. The communication is based on TCP sockets, and each program has distinct functionalities to achieve the desired peer-to-peer interaction.

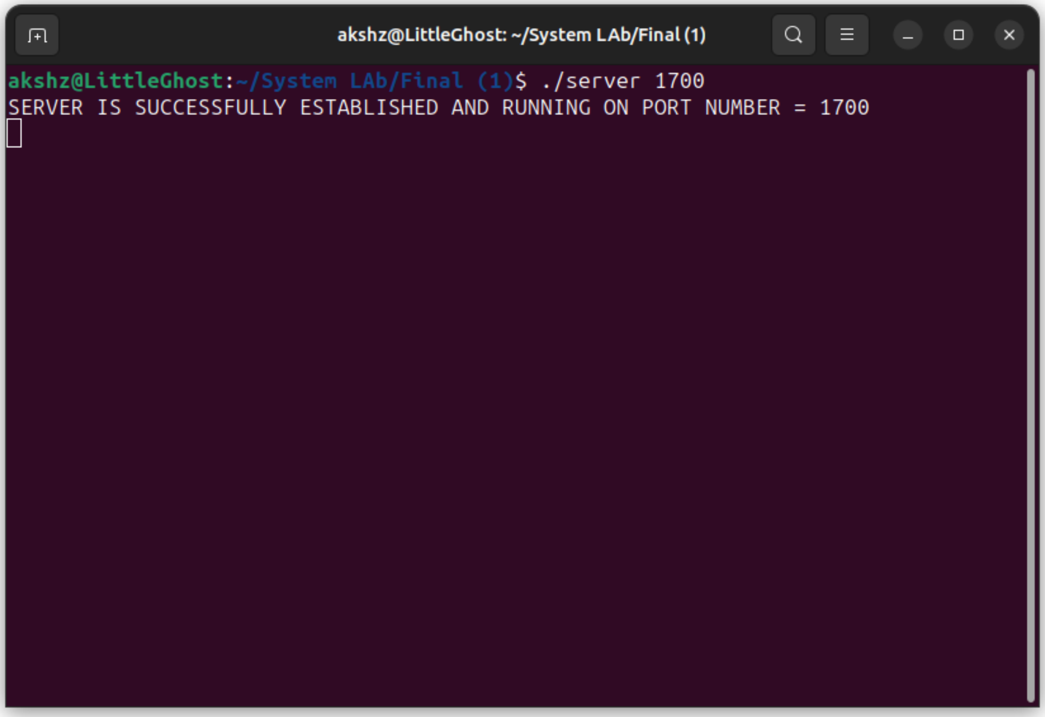
1. Introduction:

The goal of this project is to create a simple peer-to-peer system with a Relay_Server that facilitates communication between Peer_Nodes and Peer_Clients. The system is divided into three phases, each focusing on specific tasks such as registration, information retrieval, and file distribution.

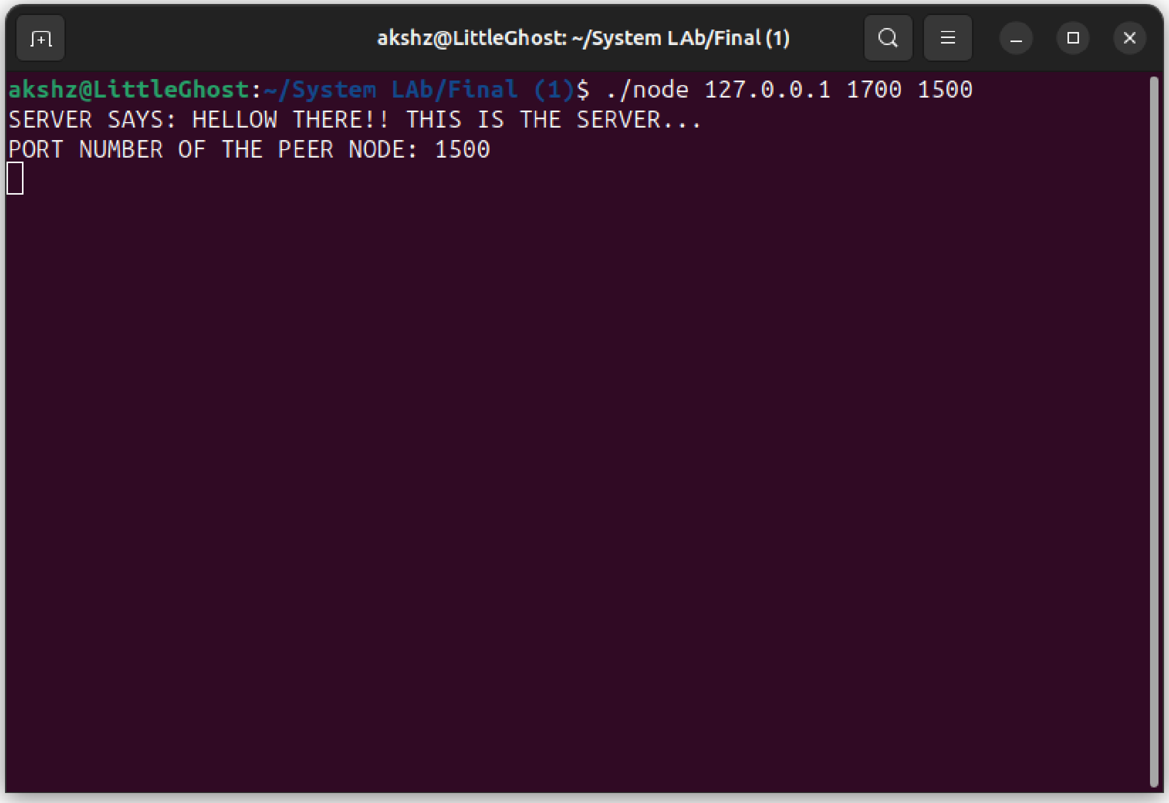
2. Implementation:

2.1 Phase One - Registration:

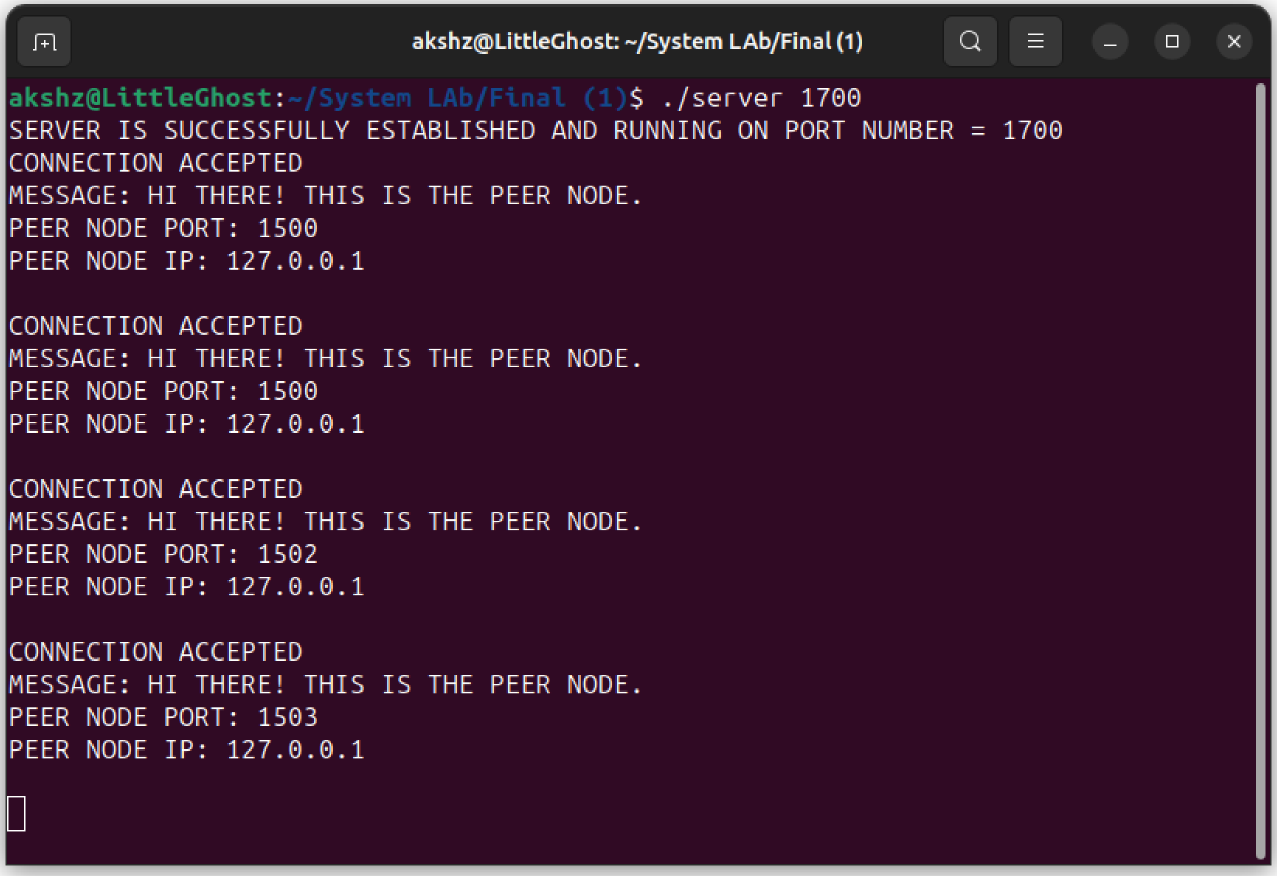
- ➢ Peer_Nodes connect to the Relay_Server using known TCP ports.
- ➢ Upon successful connection, Peer_Nodes send their IP address and port information to the Relay_Server.
- ➢ After sending the information, the Peer_Nodes gracefully close the connections.



```
akshz@LittleGhost:~/System LAb/Final (1)$ ./server 1700
SERVER IS SUCCESSFULLY ESTABLISHED AND RUNNING ON PORT NUMBER = 1700
```



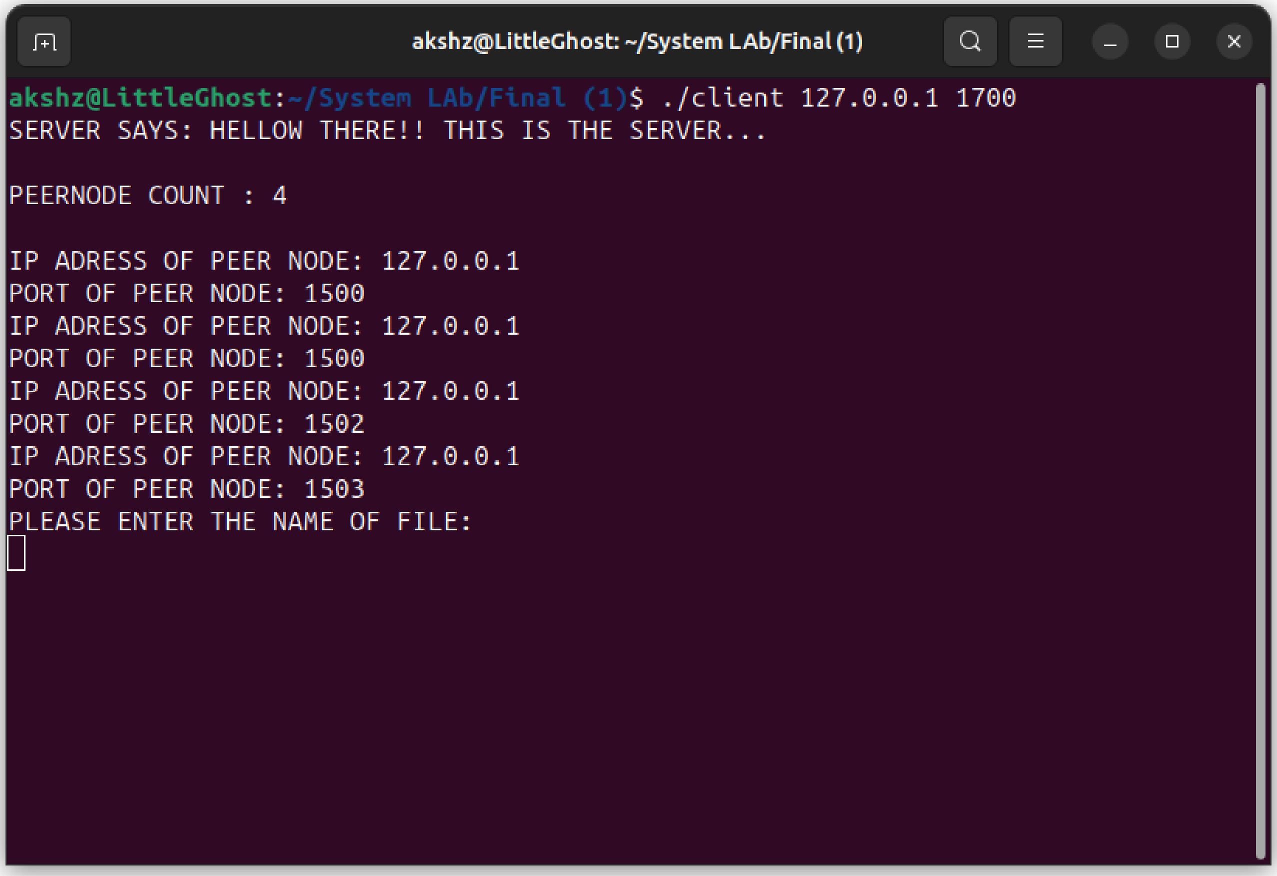
```
akshz@LittleGhost:~/System LAb/Final (1)$ ./node 127.0.0.1 1700 1500
SERVER SAYS: HELLOW THERE!! THIS IS THE SERVER...
PORT NUMBER OF THE PEER NODE: 1500
```



```
akshz@LittleGhost:~/System LAb/Final (1)$ ./server 1700
SERVER IS SUCCESSFULLY ESTABLISHED AND RUNNING ON PORT NUMBER = 1700
CONNECTION ACCEPTED
MESSAGE: HI THERE! THIS IS THE PEER NODE.
PEER NODE PORT: 1500
PEER NODE IP: 127.0.0.1

CONNECTION ACCEPTED
MESSAGE: HI THERE! THIS IS THE PEER NODE.
PEER NODE PORT: 1500
PEER NODE IP: 127.0.0.1

CONNECTION ACCEPTED
MESSAGE: HI THERE! THIS IS THE PEER NODE.
PEER NODE PORT: 1502
PEER NODE IP: 127.0.0.1

CONNECTION ACCEPTED
MESSAGE: HI THERE! THIS IS THE PEER NODE.
PEER NODE PORT: 1503
PEER NODE IP: 127.0.0.1
```

2.2 Phase Two - Information Retrieval:

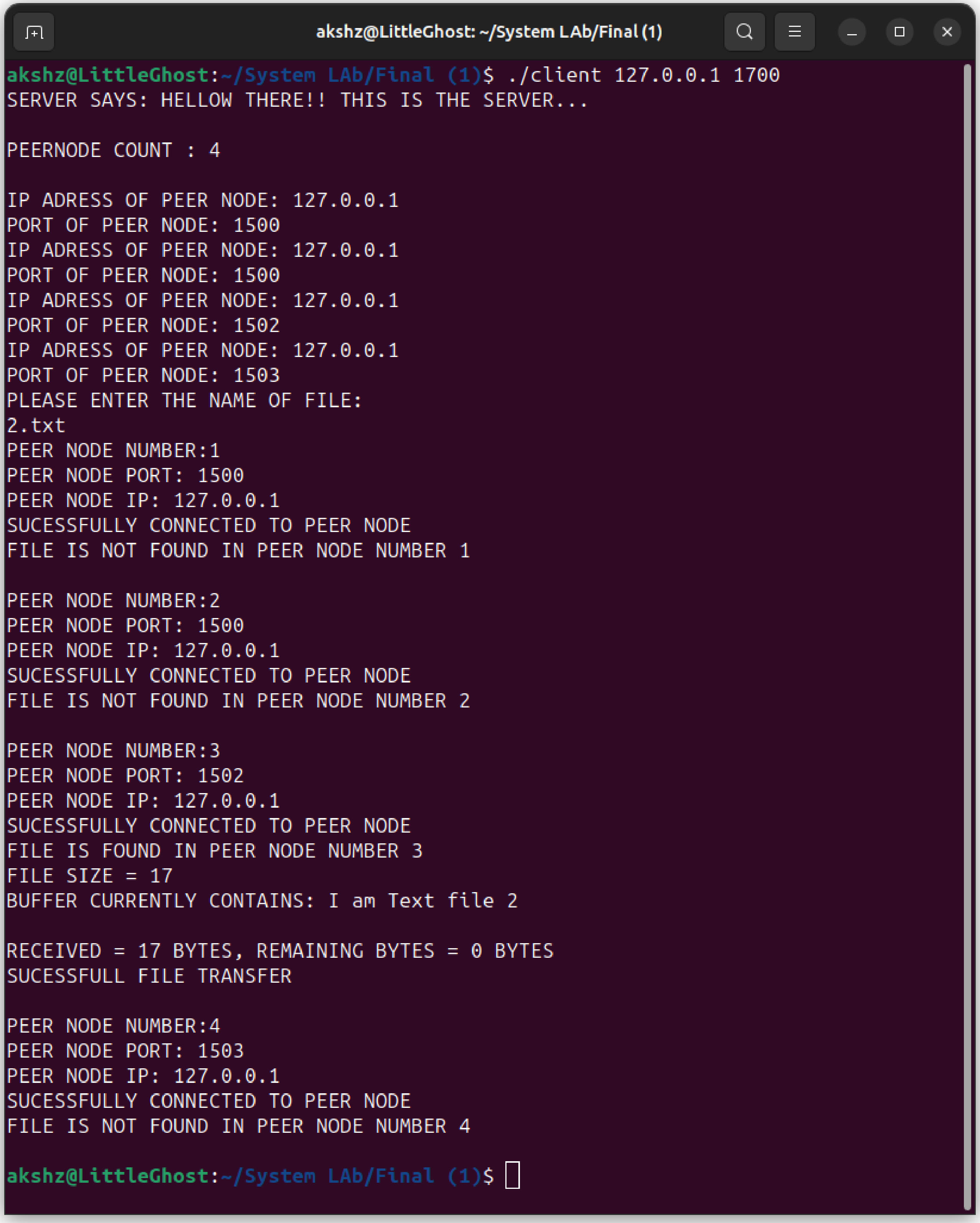
- Peer_Clients connect to the Relay_Server using a known TCP port.
- Upon successful connection, Peer_Clients request active Peer_Node information.
- Relay_Server responds with the active Peer_Node information it maintains



```
akshz@LittleGhost:~/System LAb/Final (1)$ ./client 127.0.0.1 1700
SERVER SAYS: HELLOW THERE!! THIS IS THE SERVER...

PEERNODE COUNT : 4

IP ADRESS OF PEER NODE: 127.0.0.1
PORT OF PEER NODE: 1500
IP ADRESS OF PEER NODE: 127.0.0.1
PORT OF PEER NODE: 1500
IP ADRESS OF PEER NODE: 127.0.0.1
PORT OF PEER NODE: 1502
IP ADRESS OF PEER NODE: 127.0.0.1
PORT OF PEER NODE: 1503
PLEASE ENTER THE NAME OF FILE:
```

2.3 Phase Three - File Distribution:

- Peer_Clients prompt the user for a file name.
- Using the acquired Peer_Node information, Peer_Clients connect to each Peer_Node one at a time.
- Peer_Clients attempt to fetch the file from the connected Peer_Node.
- If the file is present, the content is provided to the Peer_Client, which prints it to the terminal.
- If not, the Peer_Client proceeds to the next Peer_Node in the list.
- This process continues until the file content is obtained or all entries in the Relay_Server response are exhausted.



```
akshz@LittleGhost:~/System LAb/Final (1)$ ./client 127.0.0.1 1700
SERVER SAYS: HELLOW THERE!! THIS IS THE SERVER...

PEERNODE COUNT : 4

IP ADRESS OF PEER NODE: 127.0.0.1
PORT OF PEER NODE: 1500
IP ADRESS OF PEER NODE: 127.0.0.1
PORT OF PEER NODE: 1500
IP ADRESS OF PEER NODE: 127.0.0.1
PORT OF PEER NODE: 1502
IP ADRESS OF PEER NODE: 127.0.0.1
PORT OF PEER NODE: 1503
PLEASE ENTER THE NAME OF FILE:
2.txt
PEER NODE NUMBER:1
PEER NODE PORT: 1500
PEER NODE IP: 127.0.0.1
SUCESSFULLY CONNECTED TO PEER NODE
FILE IS NOT FOUND IN PEER NODE NUMBER 1

PEER NODE NUMBER:2
PEER NODE PORT: 1500
PEER NODE IP: 127.0.0.1
SUCESSFULLY CONNECTED TO PEER NODE
FILE IS NOT FOUND IN PEER NODE NUMBER 2

PEER NODE NUMBER:3
PEER NODE PORT: 1502
PEER NODE IP: 127.0.0.1
SUCESSFULLY CONNECTED TO PEER NODE
FILE IS FOUND IN PEER NODE NUMBER 3
FILE SIZE = 17
BUFFER CURRENTLY CONTAINS: I am Text file 2

RECEIVED = 17 BYTES, REMAINING BYTES = 0 BYTES
SUCESSFULL FILE TRANSFER

PEER NODE NUMBER:4
PEER NODE PORT: 1503
PEER NODE IP: 127.0.0.1
SUCESSFULLY CONNECTED TO PEER NODE
FILE IS NOT FOUND IN PEER NODE NUMBER 4

akshz@LittleGhost:~/System LAb/Final (1)$
```

2.4 Flow

```
Start

Peer_Nodes -> Connect to Relay_Server -> Send Information

Relay_Server -> Receive Information -> Store Information

Peer_Clients -> Connect to Relay_Server -> Request Information

Relay_Server -> Receive Request -> Send Information

Peer_Clients -> For Each Peer_Node Information Received:
        - Connect to Peer_Node
        - Request File
        - If File Present:
          - Receive File Content
          - Print Content
          - End
        - If File Not Present:
          - Proceed to Next Peer_Node Information

End
```

3. Communication Mechanism:

- Components exchange data using 'send' and 'receive' methods.
- 'send' transmits information, 'receive' captures responses.
- Defined message structure ensures secure communication.

4. Command Line Prototype:

- The system accepts IP addresses and port numbers from the command line.
- No hard-coded port numbers are used to enhance flexibility and configurability.

5. Flow of Operation:

- Peer_Nodes connect to the Relay_Server, provide information.
- Peer_Clients connect to the Relay_Server, request Peer_Node information.
- Peer_Clients attempt to fetch files from Peer_Nodes based on received information.
- Connections are established, information is exchanged, and file content is retrieved as per the relayed data.

6. Conclusion:

- ➢ The implementation successfully achieves a simple Relay-based Peer-to-Peer System using TCP sockets. The system allows for the registration of nodes, retrieval of active nodes, and distributed file access.

Future Improvements:

- ➢ Enhance security measures such as encryption for communication between components.
- ➢ Implement error handling mechanisms for robustness.
- ➢ Scale the system for a larger number of nodes and file distribution.

In summary, the project demonstrates a functional Relay-based Peer-to-Peer System, adhering to the specified guidelines and requirements. The system allows for flexible communication and efficient file distribution among connected nodes.