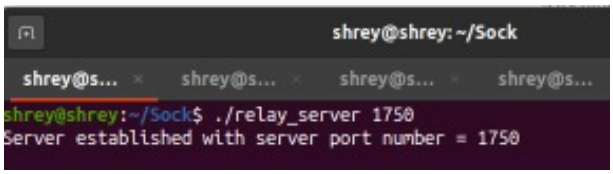
Application ID 6: Relay based Peer-to-Peer System using Client-Server socket programming

Group 30

Rishav Mondal (190101072)
Siddharth Charan (190101085)
Shrey Verma (190101083)
Karan Raj Sharma (190101043)

1 Phase - 1



We first compile and run the server `Relay_Server.c`

1. **Compilation** : `gcc Relay_Server.c -o relay_server`
2. **Running** : `./relay_server <Server port number>`
   The port number should be ≥ 1024 as those ports are reserved for common network functionalities.

After starting the server, we will try to compile and run the various peer nodes. We navigate to the directories indicated by the 3 nodes (`PeerNode1, PeerNode2 and PeerNode3`).

1. **Compilation** : `gcc Peer_Nodes.c -o peer_nodes`
2. **Running** : `./peer_nodes <Server IP Address> <Server port number> <Peer port number>`
   The port number should be ≥ 1024 as those ports are reserved for common network functionalities. We use the loopback IP address for the IP, which can be found by using `ifconfig` command.

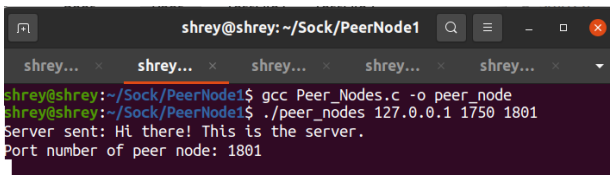


Figure 1: Peer_Node-1

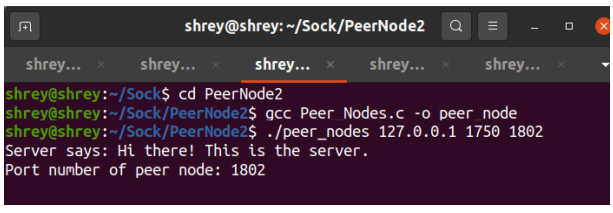


Figure 2: Peer_Node-2

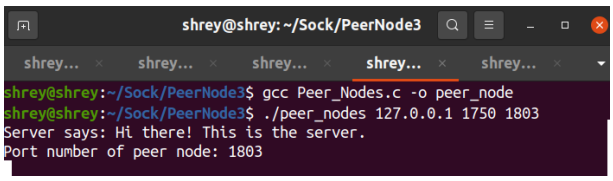


Figure 3: Peer_Node-3

2 Phase - 2

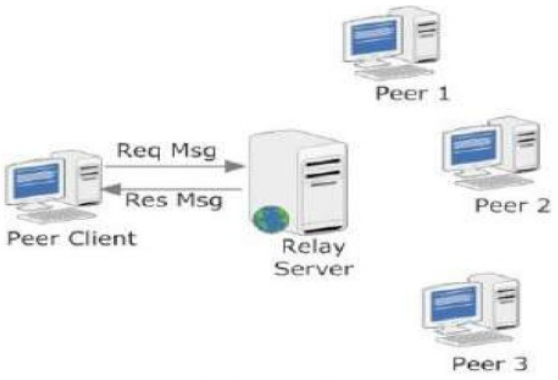
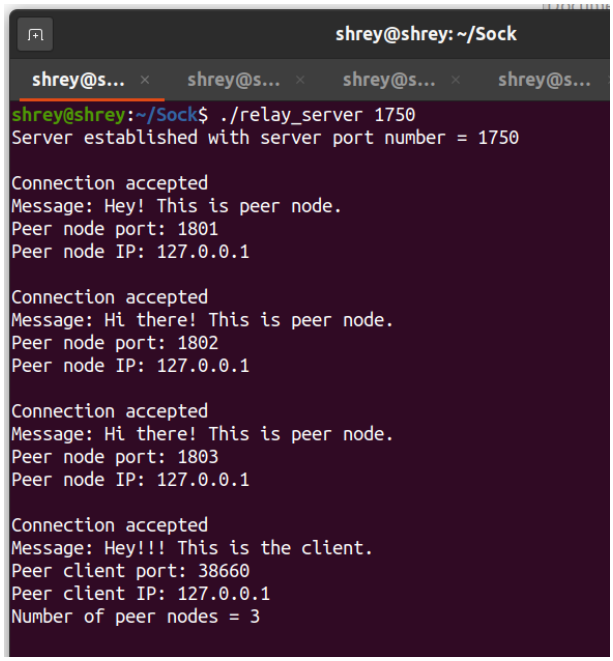
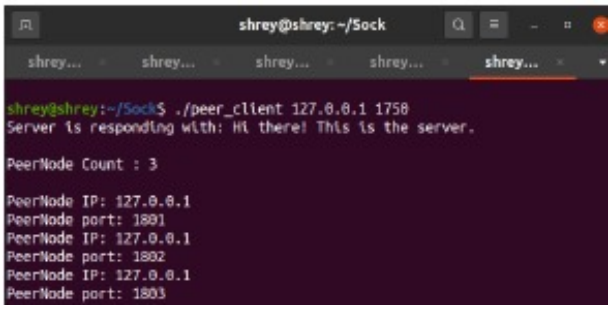


Figure 2

1. **Compilation** : `gcc Peer_Client.c -o peer_client`
2. **Running** : `./peer_client <Server IP address> <Server port number>`



The `Peer_Client` connects with `Relay_Server` and requests it for information of active `Peer_Nodes`. Then the `Relay_Server` responds with the information of the active `Peer_Nodes` which it currently has. On receiving the response message from the `Relay_Server`, the `Peer_Client` closes the connection gracefully.



3 Phase - 3

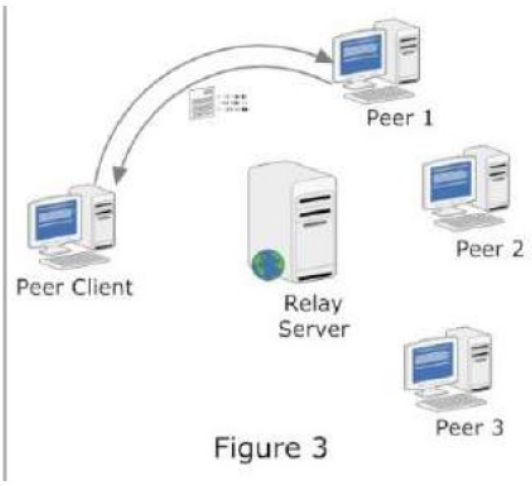
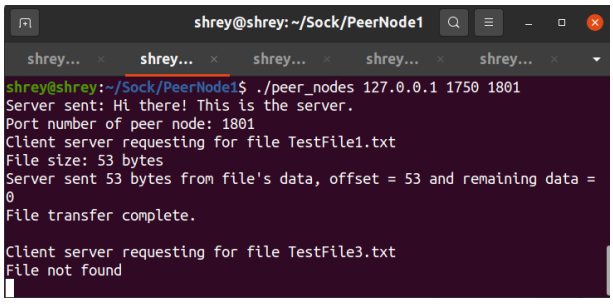


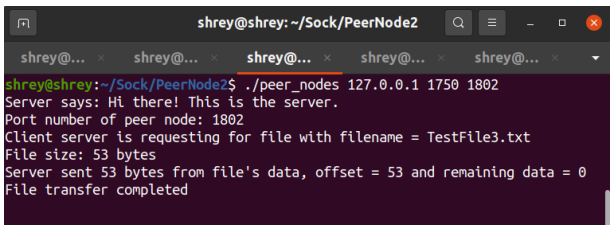
Figure 3

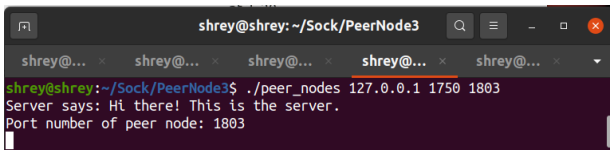
We distribute the test files in the following way:

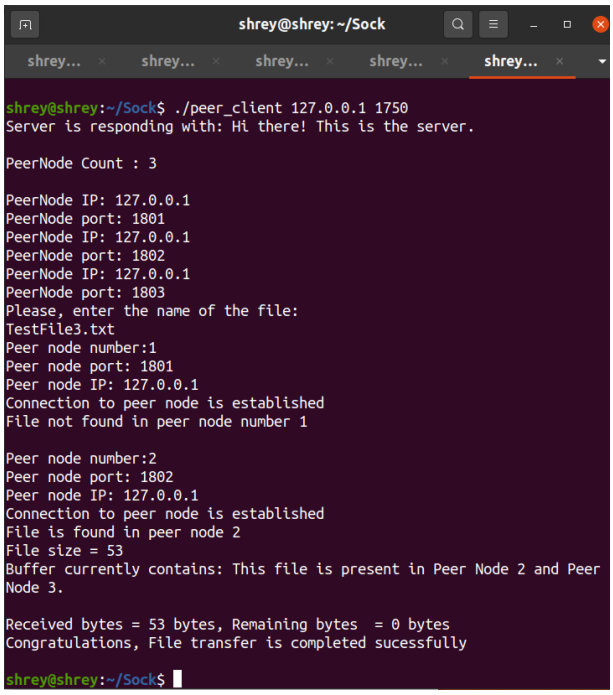
- `PeerNode1` contains `TextFile1.txt` and `TextFile2.txt`
- `PeerNode2` contains `TextFile2.txt` and `TextFile3.txt`
- `PeerNode3` contains `TextFile1.txt` and `TextFile3.txt`

On requesting `TestFile1.txt`, which is present in `Peer_Nodes` 1 and 3 then they will transfer the data back and in whichever one the file is found first is returned as shown:









In case the requested file was not found in any node, we get the output as following: