**Project Title: P2P Communication System and Broadcast**

**Introduction**

This project is a Peer-to-Peer (P2P) communication system that allows clients to communicate with each other directly without involving the server in the message exchange process. The server is responsible for managing client connections, authentication, sharing active client information, and broadcasting messages on behalf of a client. The application also keeps a record of registered users in a CSV file, and user passwords are hashed using SHA-256 for security purpose.

**Features**

1. User authentication through login or signup.
2. Password hashing using SHA-256.
3. List of active clients.
4. Broadcast messages to all connected clients.
5. Establish a peer-to-peer connection to chat with other users.

**Prerequisites**

* A C++ compiler (such as g++).
* OpenSSL library for password hashing.

**Execution steps**

**How to Compile and Run the Server**

1. Compile the server code using the following command:

**g++ -o server server.cpp -lssl -lcrypto -lpthread**

1. Run the server using the following command:

**./server**

1. The server will start listening for connections on port 4400.
2. Compile the client code using the following command:

**g++ -o client client.cpp**

1. Run the server using the following command:

**./client**

1. Follow the prompts and proceed with the execution

**Usage**

1. Compile the code using a C++ compiler.
2. Run the compiled server application on a computer.
3. Run the compiled client application on one or more computers.
4. Clients can connect to the server, authenticate themselves using their username and password, and see the list of active clients.
5. Clients can choose any active client to connect with, and the remote client will be asked to approve the new connection.
6. Once a connection is successful, clients can send messages between themselves without the server.
7. A client can request the server to broadcast a message to every client who is currently active. All clients who will be active within 30 minutes should receive the broadcast message.

**Code Explanation**

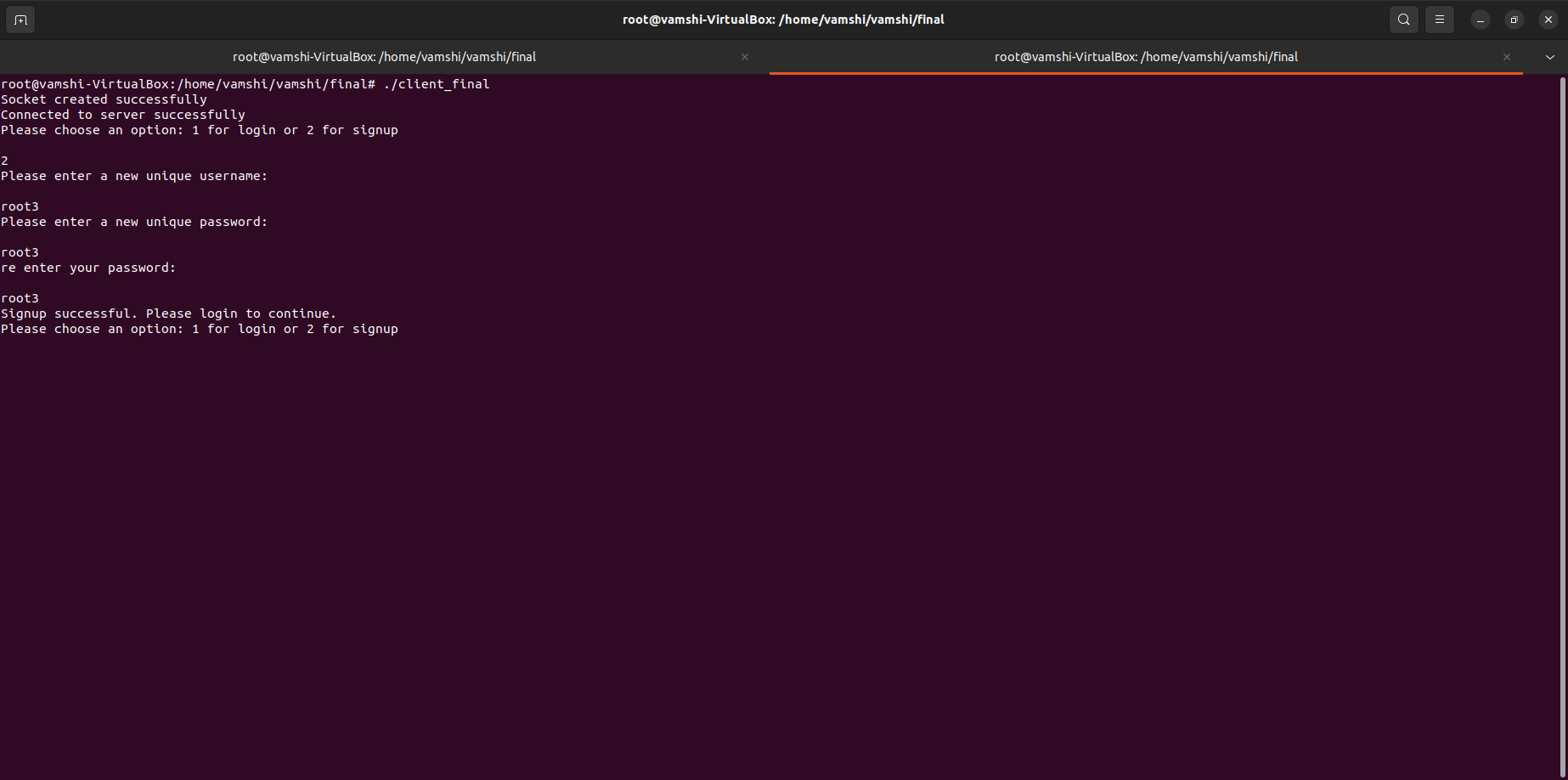
The code consists of the following main parts:

1. Libraries: Standard C++ libraries are included for handling sockets, threads, and other necessary functionality.
2. Global Variables and Constants: Constants for the maximum number of connections and buffer size are defined, along with global variables to manage clients and threads.
3. Main Functions:
   * **hash\_password**: Function to hash a given password using the SHA-256 algorithm.
   * **accept\_clients**: Function to accept incoming client connections and handle their authentication.
   * **main**: Main function that sets up the server socket and starts the threads for accepting clients.
   * **handle\_new\_client**: Function to handle each connected client, providing them with the available options to interact with other users.
4. Server Setup:
   * Create a TCP socket.
   * Bind the socket to the desired address and port.
   * Listen for incoming connections.
5. Client Handling:
   * Authenticate users through login or signup.
   * Provide options for listing active clients, broadcasting messages, and establishing peer-to-peer connections.

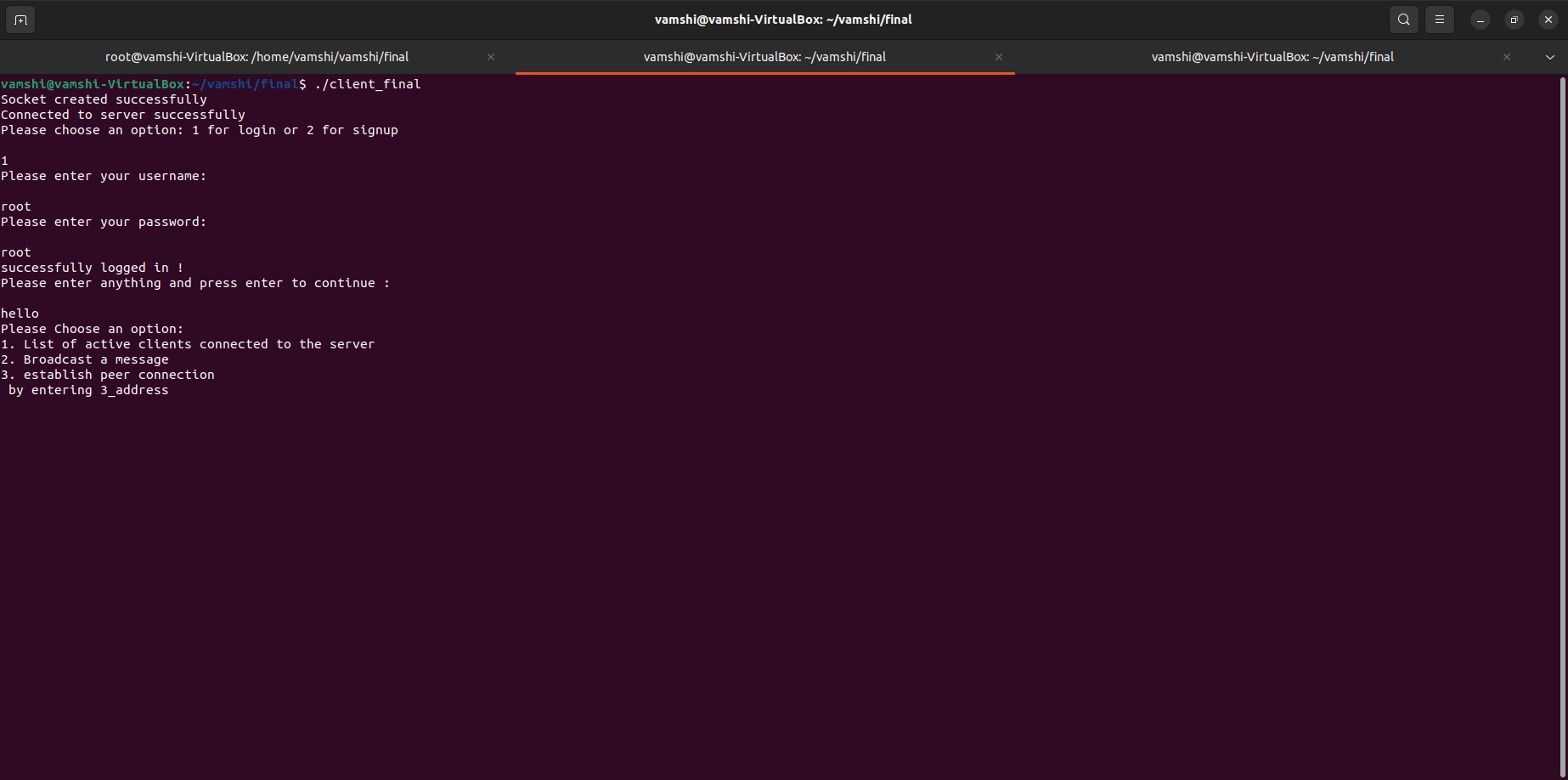
**Screenshots**

**1. Client Login/Signup:**

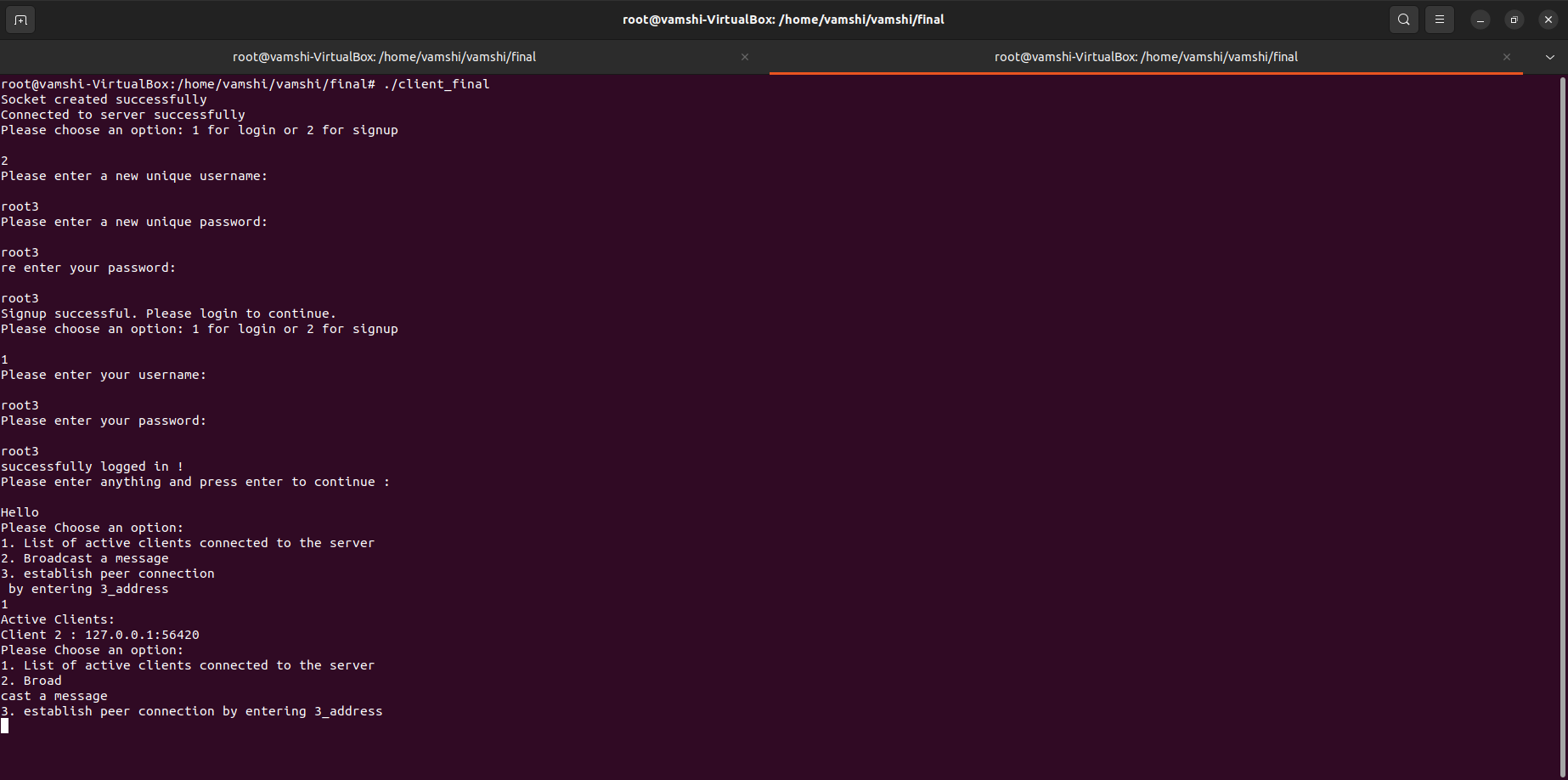
**Signup:**

****

**Login:**

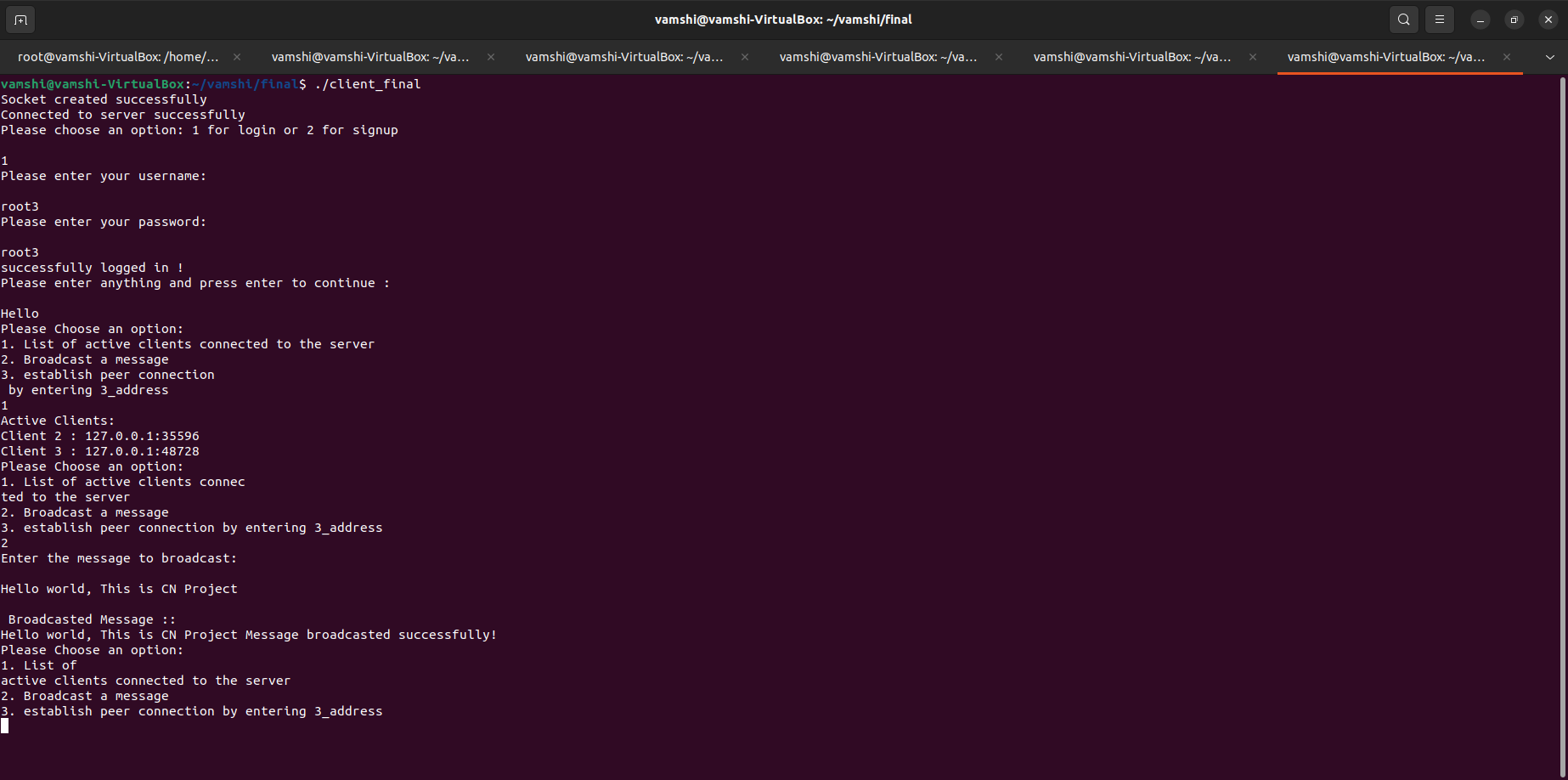
****

**2. List of Active Clients:**

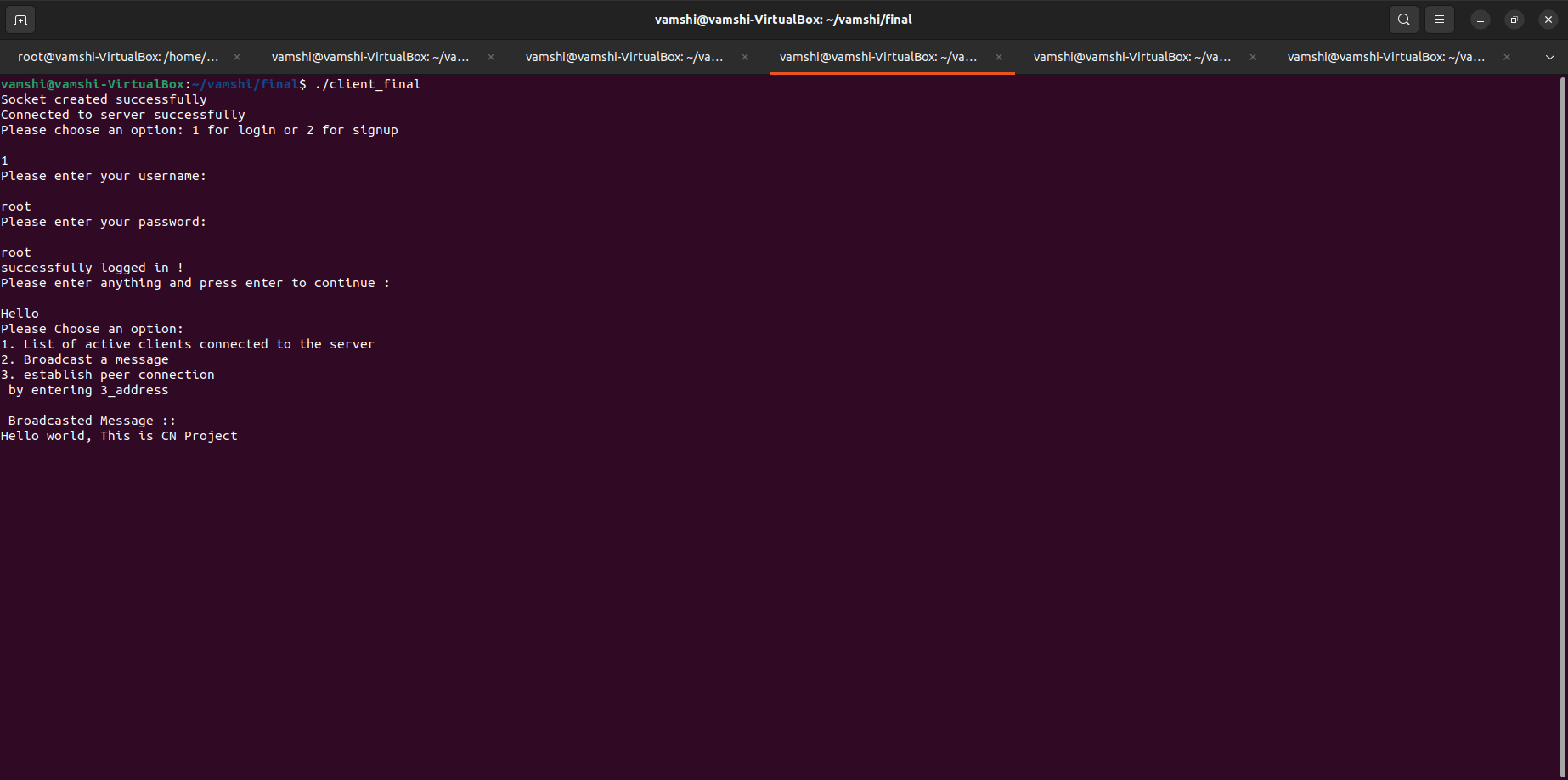
****

**3. Broadcasting a Message:**

**3.1 Client sending broadcast message :**

****

**3.2 other client receiving broadcasted message :**

****

**4. Establishing a Peer-to-Peer Connection:**

**4.1: Client sending a p2p message using the command:**

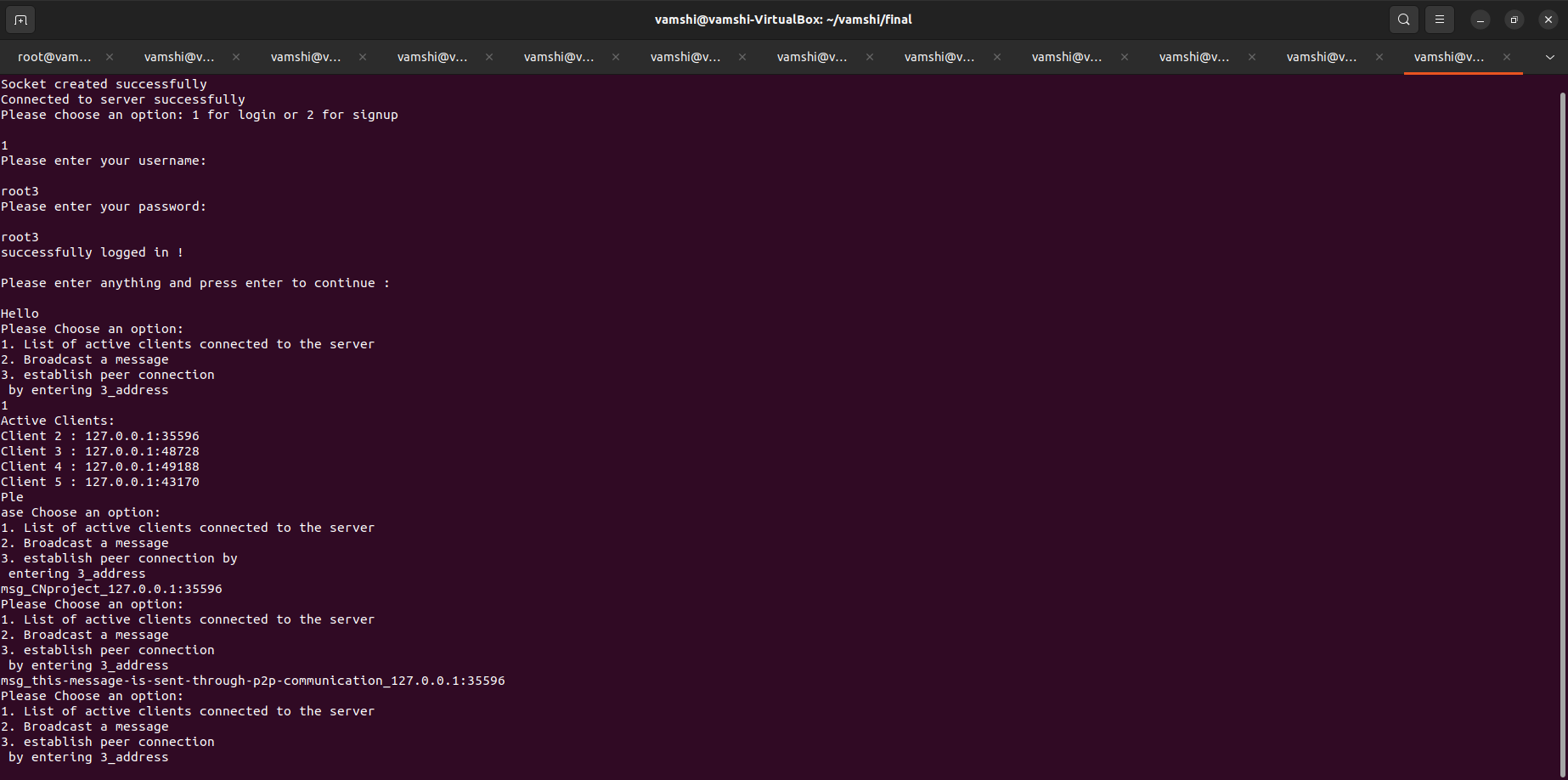
Command:

**msg\_”your-message”\_”IP-and-port-from-list-of-client”**

Example:

1. **msg\_CNproject\_127.0.0.1:35596**

**2. msg\_this-message-is-sent-through-p2p-communication\_127.0.0.1:35596**

****

**4.2** Client receiving the p2p message:

