**Lab Report: Peer-to-Peer (P2P) Chat Application with Group Chat**

INRODUCTION  
Peer-to-peer (P2P) networking offers a decentralized approach to communication, eliminating the single point of failure and potential bottlenecks of centralized server-based systems. The objective of this lab is to design and implement a decentralized peer-to-peer (P2P) chat application using socket programming. The application allows users to communicate directly with each other without relying on a central server, emphasizing the challenges of decentralized communication, such as peer discovery, message routing, and user presence management. Additionally, the application supports **group chat functionality**, enabling multiple users to participate in a shared conversation.   
  
**Goals:** The primary goals of this lab were to:

* + Develop a functional P2P chat application using Python and tkinter for the GUI.
  + Implement both direct connection (IP/port) and presence-based connection methods.
  + Enable users to exchange text messages and files in both individual and group chat settings.
  + Display user presence information (online/offline status) in the chat interface.

ARCHITECTURE

High Level System (Component Diagram)

A diagram of a diagram

Description automatically generated

**Explanation**:

* Each **Peer** encapsulates:
  + A **GUI** (Chat UI) to handle user interaction.
  + The **P2PChat** component for managing peer connections and message routing.
  + A **File\_Transfer** module for chunk-based file sending/receiving.
  + A **Presence Client** that communicates with the **Presence.**
* Lines (<-->) indicate TCP connections.
* Peers can connect directly to each other **and** optionally to the presence server.

**System Components**

* **ChatUI (tkinter-based):** The main GUI managing user interactions, messages, and system notifications. It connects with P2PChat, FileTransfer, and PresenceClient.
* **P2PChat:** Manages peer-to-peer messaging using TCP sockets, handles group chats, and maintains connections.
* **FileTransfer:** Splits files into chunks, sends them via sockets, and reassembles them on the recipient’s end. Supports individual and group file transfers.
* **PresenceClient:** Connects to a presence server to register users, discover peers, and update online statuses.

Presence-Based Discovery

A diagram of a system

Description automatically generated

**Explanation**:

* The **user interface** (U) calls methods on the PresenceClient (PC) to register or query users.
* The **presence server** (PS) responds with success or error messages, and eventually provides a list of online users.
* Once the **UI** receives that list, the user can **initiate a direct P2P connection** to any desired peer’s IP/port.

File Transfer Flow

A screenshot of a computer screen

AI-generated content may be incorrect.

**Explanation**:

* The **Sender’s UI** triggers the send\_file method in the FileTransfer object (SFT).
* FileTransfer reads the file, chunk by chunk, and instructs P2PChat to send each chunk as a JSON message to the **Receiver’s P2PChat**.
* The receiver’s P2PChat calls back into its own FileTransfer module to handle the chunk (file\_chunk\_callback).
* Chunks are buffered until the last chunk is detected, then the receiver writes the file to disk.
* The UI can display progress or completion messages as needed.

IMPLEMENTATION

TESTING

DISCUSSION  
**Challenges Faced:**

CONCLUSION  
This project provided helpful practice in designing and creating a decentralized P2P chat application with group chat functionality. The application is successfully able to utilize peer discovery, message routing, user presence, group chat, and file transfer in a decentralized environment. Though the current implementation is okay for small networks, future efforts can be towards enhancing scalability and making advanced functionality such as DHT-based peer discovery and encryption available.

APPENDIX (Code)  
  
The well-commented source code for the project is included in the following files:

* chat.py: Main GUI and application logic.
* p2p\_chat.py: Core P2P chat functionality, including group chat.
* message\_handler.py: Handles incoming messages.
* file\_transfer.py: Handles file transfer between peers.
* utils.py: Utility functions for message framing and socket communication.

<https://github.com/keshavghimire/p2pchat>  
  
**AI Usage in Coding**

**Prompt Used for AI Assistance:**