**Features List**

We have designed and implemented a peer-to-peer chat application that utilizes both UDP and TCP sockets. Our application allows for real-time communication between clients while using UDP for media transfer and TCP for signaling and initializing peer communication. A server is used to coordinate initial client connections, maintain a list of active peer clients, and facilitate tasks such as signaling and obtaining communication parameters.

Below are the main features of our application:

* **Server-Client Communication**: The program facilitates communication between a server and multiple clients. Clients connect to the server over TCP/IP sockets, allowing them to send and receive messages. The server uses threading to handle multiple client connections concurrently. Client connections are managed by separate threads which allows for simultaneous communication with multiple clients.
* **Peer-to-Peer Chat**: The application enables peer-to-peer chat between two clients. When one client wants to chat to another client who is online, the application will send a message to that client asking if they want to chat. It will use threading for sending and receiving messages between the two clients. If one client types ‘->exit’, they will leave the chat.
* **User Authentication**: An added feature which ensures that there are no duplicate clients, clients are prompted to provide a username upon connection. The server ensures that each username is unique among connected clients.
* **Status Management**: Clients can choose their online visibility status (online or offline). This status is managed by the server and broadcasted to other clients.
* **Menu Options for Clients**: Clients have a set of menu options to choose from, including viewing online clients, initiating chats with other clients, managing their online status, and disconnecting from the server. The menu options provide a user-friendly interface for clients to interact with the server and other clients.
* **Broadcasting Messages**: The server can broadcast messages to all connected clients. This feature allows for global announcements or notifications.

Overall, the program serves as a basic chat server with features such as user authentication, status management, menu-driven interaction, broadcasting, and support for peer-to-peer communication.

**Client Implementation**

BLLDAA001:

CNXSIA001:

CMPCLA004:

**Sequence Diagrams**

**Protocol Specification**

We came up with two protocol designs that have been implemented, the one is the signaling protocol enabling client-server connection and the other is the peer-to-peer communication protocol.

1. The signaling protocol
   * This protocol defines how clients connect to the server using TCP sockets.
   * Clients are authenticated when they are trying to connect to the server to ensure that only authorized clients can connect to the server.
2. Peer-to-peer communication protocol
   * This protocol is what allows real-time communication between peers using UDP sockets. It defines how peers can be discoverable on the network in order for direct communication amongst each other.
   * This protocol defines the structure of message headers containing information on message types and message structure.
     + There are three types of messages:

**Command messages:** this how communication is initiated and terminated and allows clients to query available peers.

**Data transfer messages:** this allows transfer of real-time communication and signaling to connect to server.

**Control messages:**  this allows for the acknowledgement of received messages and requests for re-transmission in the event that the message has not been received.

* + - The structure of messages is as follows:

**The header:**

Message Type: Command, Data, or Control

Sender ID

Receiver ID

Message Length

**The body:**

Command details (if command message)

Media stream data (if data transfer message)

Acknowledgement status (if control message)

* + The protocol also defines communication rules for transmission. This includes the allowed client states, the exchange of messages, and the transition between states.
    - There are three types of allowed client states. These include:

**Available:** Client is ready and available to communicate.

**Connected:** Client is currently in an active chat.

**Away:** Client is temporarily inactive.

* + - Message exchange is defined as follows:

**Available clients query:** Client sends a command message to the server to request a list of available peers.

**Signaling:** Client uses TCP to send and receive signaling messages for establishing a connection to the server and then to a peer.

**Real-time communication:** Once connected, clients use UDP for transmitting media.

* + - The types of state transitions is defined as follows:

**Available to connected:** This occurs upon successful connection initiation between two peers.

**Connected to away:** When a client becomes inactive or disconnects.

**Away to available:** When a client becomes active again.

**Screenshots of the Application**