Text Conferencing Implementation Outline

Chosen Feature 1: Private Messaging

Command Format:
/pm <message destination client ID> <message>

Protocol Format:

Type	Packet Source	Packet Data
PRIVATE_MESSAGE	<Source Client ID>	<Message Destination Client ID> <Message>

Design Choices:
Space is used as a delimiter for usability, unlike other delimiters. Specifying the destination client ID in every private message allows users to quickly message multiple clients without complicated one-on-one private message sessions, which function the same as sessions.

Implementation:
This feature repurposes the old send message function code on the client side. The client strips the command /pm from the input and sends the rest of the command input to the server with the message type PRIVATE_MESSAGE. The server repurposes the old send_message and get_client_info functions. The server splits the string delimited by space to find the destination client ID and the message. The server then looks for the client info with the get_client_info function and send_message to send the message with the client info if the requested client ID exists in the database.

Chosen Feature 2: User Registration

Command Format:
/register <client ID> <password> <server-IP> <server-port>

Protocol Format:

Type	Packet Source	Packet Data
REGISTER	<client ID>	<password>

Design Choices:
Users must log in after registration. This design is chosen to dissuade ambiguity of features and ensure each feature does not overlap the other. Users must also log out before registering another client. This design is chosen to ensure no two users are logged in on the same terminal.

Implementation:
The old login code is repurposed on both the server side and the client side. The only difference is that the new login information is added by writing to a text file database.