**SA close-up of a computer chip

Description automatically generatedOCKET PROGRAMMING PROJECT**

**CSC3002F**

**2023**

***Authors:*** **Mzimasi Hlabe, Mnelisi Mabuza, Iviwe Booi**

1. Introduction

Peer-2-peer system allows separate end-systems to directly interact with each other without being connected, enabling more distributed and scalable designs. This report documents the design and implementation of a peer-to-peer chat application allowing users to chat with each other in real-time by using UDP to exchange text messages and TCP for connecting and interacting with the server. The server handles queries for available peers and provides necessary responses. We also implemented some protocols for sending messages.

2. Application Features

The following key features were implemented:

- Registration and listing of available peers via a centralized TCP server

- Establishment of direct UDP connections between clients for real-time chatting

- Authentication of clients using shared secret keys

- Privacy controls to specify visibility and permissions of client information

3. Protocol Design

3.1 Message Formats

Three message types were defined - COMMAND, DATA, CONTROL. The header structure contains the message type, sender/receiver IDs, sequence number, payload length.

3.2 Communication Flow

Clients connect to the server via TCP and register. The LIST command returns available peers. Clients can then signal directly via CONTROL messages to initiate a UDP connection. DATA messages carry chat payloads over UDP.

3.3 State Diagram

A state diagram (see Figure 1) shows the different client states - IDLE, CONNECTING, CONNECTED, DISCONNECTED. Transition between states is triggered by incoming messages.

4. Implementation

4.1 Server

The Python server runs multithreaded to handle multiple clients. It maintains dictionaries for registered clients and their connectivity status.

4.2 Client

The client implements a command-line interface for registration and querying peers. It uses Python sockets for TCP signaling and UDP data. Authentication is done via shared keys.

5. Screenshots

Screenshots (Figures 2-4) show the client registration, listing available peers, and real-time chat features.

6. Conclusion

The objectives of implementing a peer-to-peer chat application utilizing both TCP and UDP were achieved. The features implemented address privacy, authentication and flexibility in peer discovery and connectivity. Future work includes adding support for file transfers, integration with a web/mobile frontend.

7. Appendices

Appendix A contains the protocol specification and sequence diagrams. The source code for the server and sample client implementation is attached separately.