UDP Chat Application

UDP Chat Application

Version <1.0>

Glenn Harper

Liathe Najdawi

2019-03-11

Revision History

|  |  |  |
| --- | --- | --- |
| Date | Version | Description |
| 03-11-19 | 1.0 | Initial Version of Document |
|  |  |  |
|  |  |  |

Software Design Document

1 Introduction

The UDP transport protocol is a fast and easy way to send data over the Internet. It’s been proven helpful for video streaming, video chat services like Skype, and many other applications which value high speed transportation of their data, over the data integrity promised by TCP. Although UDP has a risk of packet loss that is not present in TCP, there are certain measures that can be implemented in the application layer which can provide acknowledgement of packets received between client and server processes. This allows UDP to maintain the benefits of speed and control over packet contents that give it an advantage of TCP, which requires additional time and bandwidth for flow-control, congestion control, and control messages for the initial handshake. Implemented in a Simple Chat Messenger Application, UDP can be a fast and efficient method of communicating over long distances.

1.1 Purpose

The purpose of this project is to create a Chat Application which makes use of the UDP protocol to allow 2 or more clients to seamlessly send messages between one another instantly through a server facilitating connections between clients. The application should have a reliable connection between any two clients. On the client side, finding and messaging other users on the same chat server should be easily accessible. The server will assign user ids to each client and present a list of all clients connected to the server.

2 Software Requirement Specification

2.1 Project Scope and Features

2.1.1 Functional Requirements

1. User Registration

Users must be able to connect to the server using the client application running on their end system. When connecting for the first time, the server should assign the user an id and add them to the list of active users, where they can be seen and messaged by any other active users on the sever. The user should also be given an inbox which will store all messages received on the server. Users should have an option to change their user names. Changing your username to ‘admin’ should prompt the server to request the administrator password. Once successfully entered this should allow the chat server administrator to access the chat server application and make changes.

2. User Instant Messaging

Users must be able to send and receive messages with any users that are active on the server. Upon receiving a message from another active user, the user should be notified that they have received a message via an alert with the user id of the sender as well as a short preview of the message. All messages received by any user should go directly to the user’s inbox for later review.

3. User Status

Usernames displayed on the server should also specify the availability of a user. On the list of users presented upon login, all users registered on the server should have an indicator by their user id which denotes whether each registered user is online, away. Connected to the server and prepared to chat.

4. Instant Message Status Tracking

When a user sends a message, they should be notified once the message has been received by the recipient. Additionally, any successfully sent messages which have been open and read by the recipient should be marked as read for the sender to distinguish them from those which have been successfully sent, but not opened or read by the recipient.

2.1.2 Non-Functional Requirements

1. Performance

Both Client applications and Server applications should be lightweight and allow for instant exchanges of messages between any clients actively connected to the running server application hosting the chat server.

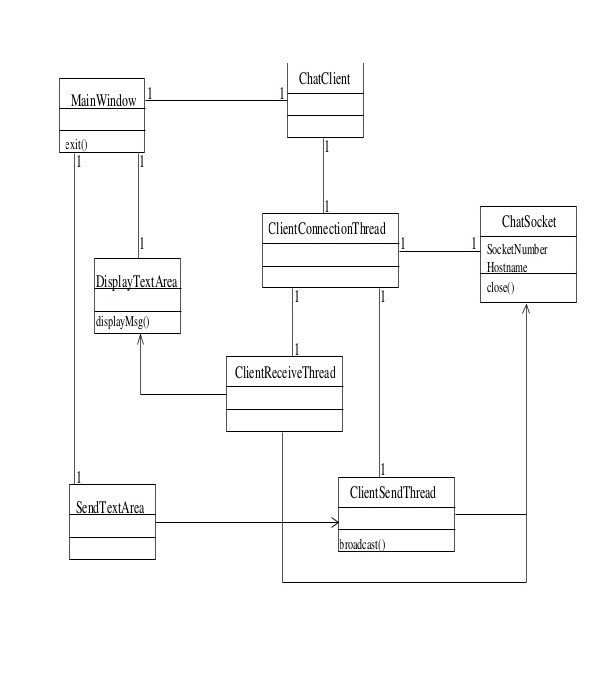
2. Data Integrity and Recoverability

Server users lists and message history should be properly backed up, so that in the case of an unexpected client disconnect, messages and user ids are not immediately lost. There should application-layer measures to tackle the issue of packet loss, which is present in UDP.

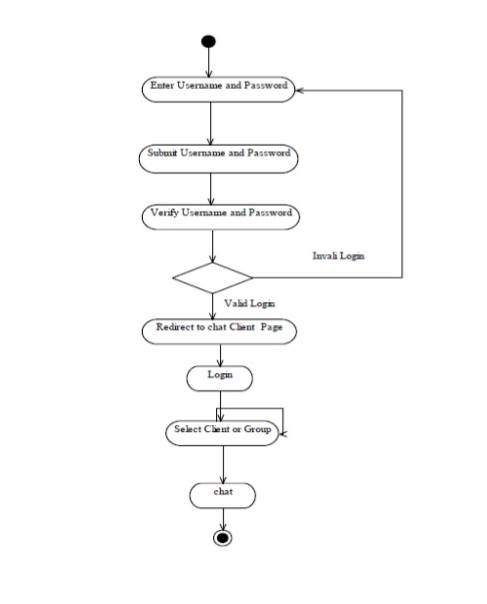
3. User Privacy

Users should not be able to read messages of a chat session of which they are not a participating member. Users should be distinguished by IP address by the server for authentication, but the IP address of any user should only be visible to the server to match to an id on the user list, and not visible to any users on the chat server. Messages sent between clients over the chat server should be encrypted, so that the message contents are only available to the receiving user active on the server.

2.3 Class Chart



2.4 Flow Chart



3 Implementation and testing

**Databases**: Firebase, MySQL, MongoDB

**Programming Languages**: Python, Java, JavaScript

**Libraries, technologies and frameworks**:

**Python**: Socket, wxPython, Tkinter, ssl

**Java**: JSP, Spring, Hibernate, JEE/J2EE

**JavaScript**: Node.Js

**IDE**:

Visual Studio Code

**Version Control**:

Git, Github

**Build Tools**:

Maven, Gradle

**Application and Web Server**:

Apache Tomcat, Jboss, WebLogic, BaseHTTPServer, SimpleHTTPServer, CGIHTTPServer

**Testing**:

Selenium Framework, Functional Testing, Load Testing, Data Integrity testing, Automated testing and Maintentance