Facoltà di Ingegneria e Scienze Informatiche

Python Chatroom

Progetto Laboratorio Programmazione di Reti

Pierfederici Edoardo
Matricola: 0001071371

Luglio 2024

Introduzione

Il seguente elaborato consiste in un semplice progetto in Python basato sugli argomenti trattati durante le lezioni di laboratorio del corso di Programmazione di Reti dell'Università di Bologna. Questa documentazione ha lo scopo di descrivere in modo dettagliato il funzionamento del programma e l'implementazione del codice.

Traccia scelta

Implementare un sistema di chat client-server in Python utilizzando socket programming. Il server deve essere in grado di gestire più client contemporaneamente e deve consentire agli utenti di inviare e ricevere messaggi in una chatroom condivisa. Il client deve consentire agli utenti di connettersi al server, inviare messaggi alla chatroom e ricevere messaggi dagli altri utenti.

Requisiti di sistema

Per eseguire correttamente il programma è necessario rispettare i seguenti requisiti:

- Python 3.x
- Connessione stabile e funzionante per la comunicazione client-server
- Accesso ad una porta configurata per il server

Architettura del sistema

Il programma è costituito da due sezioni principali:

- **Server**: gestisce la connessione e la comunicazione con i client e trasmette i messaggi ai client connessi alla chat.
- **Client**: permette agli utenti di connettersi alla chat e interagire con altri client tramite un'interfaccia grafica

Istruzioni per l'esecuzione

Per avviare la chat è necessario inizialmente eseguire lo script *chat_server.py* per permettere al server di mettersi in attesa di connessioni dai client. Una volta che il server è attivo, ciascun utente può avviare il client tramite lo script *chat_client.py*.
A questo punto appare una schermata di login in cui l'utente deve inserire i seguenti dati necessari alla connessione: server host, numero di porta e nickname

(se la porta non viene specificata, il programma assume automaticamente la porta di default 53000). Una volta premuto il tasto "Connetti" la schermata di login si chiude e se la connessione è avvenuta correttamente l'utente accede alla chat e può iniziare a inviare e ricevere messaggi. L'utente può lasciare la chat in qualsiasi momento cliccando il tasto "Quit".

Dettagli implementativi

Server

- Funzione **accept_connections**: accetta una nuova connessione da un client (socket) e il relativo indirizzo; salva l'indirizzo nel dizionario *addresses* e crea un nuovo thread per il client connesso. Inoltre stampa nella console l'indirizzo del client che si è collegato, oppure un messaggio di errore in caso di connessione fallita.

- Funzione **manage_client**: è la funzione centrale lato server che permette di gestire la comunicazione con un client. Inizialmente riceve e decodifica il nickname riportandolo all'interno della chat box. Tramite un ciclo while, attivo finché non viene inviato il comando di quit, riceve i messaggi inviati dal client. Quando l'utente esce dalla chat, il client viene chiuso ed eliminato dal relativo dizionario; lo stesso viene eseguito in caso di errore di comunicazione o perdita di connessione con il client.

- Funzione **broadcast**: trasmette un messaggio, preceduto da un *tag* (nickname del mittente) a tutti i client connessi; produce un messaggio di errore in caso di mancata ricezione.

Client

- Funzione **receive**: riceve e decodifica in formato UTF-8 i messaggi provenienti dal server e li trasmette nella chat box della GUI.

- Funzione **send_message**: permette di inviare un messaggio al server tramite le funzioni *set* (pulisce la variabile *my_msg*) e *send*. Se il messaggio equivale al comando di quit, il socket e la finestra della chat vengono chiusi. Nel caso in cui l'invio del messaggio non andasse a buon fine, viene visualizzato un *message box* di errore.

- Funzione **quit**: invia un messaggio contenente il comando di quit per chiudere la chat.

- Funzione **start_chat**: crea l'interfaccia grafica della chat e stabilisce la connessione con il server. Dichiara le variabili necessarie alla configurazione del client, tra cui *HOST* (indirizzo IP del server), *PORT* (porta su cui il server si mette in ascolto; deve essere un numero intero e il valore predefinito è 53000) e *NICKNAME* ottenute dall'input da tastiera

dell'utente tramite la schermata di login. Una volta configurato il client, viene eseguito un tentativo di connessione al server, al quale viene comunicato il nome utente.

Interfaccia grafica

L'interfaccia grafica della chatroom, implementata a livello client, è stata creata utilizzando il framework grafico di Python **Tkinter**.
Consiste in due frame principali:

- **Schermata di login**: finestra contenente tre campi (*tk.Label* con relative *tk.Entry*) che l'utente deve compilare per connettersi alla chat (indirizzo IP del server host, numero della porta e nickname).
- **Schermata della chatroom**: è costituita da una listbox centrale contenente tutti i messaggi della chat. In basso si trova una Entry che permette all'utente di scrivere il proprio messaggio e inviarlo con il tasto Invio. Di fianco sulla destra si trova anche il pulsante Quit che permette all'utente di lasciare la chatroom.

Conclusioni

Questo progetto mi ha permesso di applicare con il linguaggio Python una parte del programma del corso di Programmazione di Reti. Si tratta di un programma abbastanza semplice che tuttavia garantisce un risultato piuttosto pratico e completo riguardo alla comunicazione tramite chat tra diversi utenti.
Il sistema è stato realizzato mediante socket TCP (Transmission Control Protocol) per garantire una comunicazione affidabile tra client e server. Il server utilizza un sistema multi-threaded per gestire in maniera fluida l'interazione tra i client. Inoltre il programma prevede una gestione di base degli errori e delle eccezioni.
Il sistema risulta scalabile ed è facilmente migliorabile ed estendibile in futuro con nuove funzionalità.