PyChat

Progetto di Programmazione di reti

Università di Bologna

Martin Tomassi; 0001077737
Francesco Pazzaglia; 0001077423
Luca Casadei; 0001069237

30 aprile 2024

Indice

Capitolo 1

Introduzione

Questa è la documentazione del progetto di Programmazione di Reti relativo alla Traccia 1: *Sistema di Chat Client-Server* in Python. Il progetto mira a implementare una chat che permetta a più utenti di connettersi a un server centrale e di comunicare tra loro in una chatroom condivisa.
Il progetto è suddiviso in due componenti principali: il **client** e il **server**.
Il server si occupa di gestire le connessioni dei client e di instradare i messaggi in modalità broadcast all'interno della chat. Il client, invece, permette agli utenti di connettersi al server tramite il protocollo TCP, di inviare messaggi e di visualizzare i messaggi ricevuti dagli altri utenti nella chatroom. Quando un client si connette al server, può vedere solo i messaggi inviati dopo che la connessione è stata stabilita; i messaggi precedenti non vengono "salvati" nel server.
Nell'implementazione del progetto, sono state adottate diverse scelte architetturali e di progettazione. Ad esempio, la comunicazione tra client e server avviene attraverso il protocollo TCP, scelto per la sua affidabilità nella trasmissione dei dati. Inoltre, l'utilizzo dei thread consente di gestire in modo concorrente l'invio e la ricezione dei messaggi sui due lati della connessione, garantendo un'esperienza fluida agli utenti.
Nel codice del client, è stata implementata una logica per la gestione dei thread separati per l'invio e la ricezione dei messaggi, in modo che l'utente possa scrivere e leggere contemporaneamente senza bloccare il programma.
D'altra parte, nel codice del server, è stata implementata la logica per gestire le connessioni multiple dei client e la distribuzione dei messaggi ricevuti a tutti i client connessi.
In definitiva, il progetto si propone di offrire una soluzione robusta e affidabile per consentire a più utenti di comunicare in tempo reale tramite una chatroom condivisa.

Capitolo 2

Client

Il client utilizza un approccio connection-oriented IPv4 attraverso il modulo *socket* di Python per stabilire una connessione con il server della chatroom. Prima di avviare l'applicazione, l'utente deve specificare l'indirizzo IP del server e un nome utente univoco, che servirà a identificarlo all'interno della chatroom. Questi parametri vengono passati come argomenti al momento dell'esecuzione del client.
Una volta avviato, il client crea un socket e tenta di stabilire una connessione con il server utilizzando l'indirizzo IP e la porta specificati. Se la connessione viene stabilita con successo, il client invia il nome utente al server per identificarsi.
Dopo la connessione, il client avvia due thread separati per gestire la trasmissione e la ricezione dei messaggi.
Il thread di trasmissione consente all'utente di digitare un messaggio dall'input del terminale e inviarlo al server tramite il socket. Prima di inviare il messaggio, viene applicata la codifica UTF-8 per garantire la compatibilità con caratteri speciali come le lettere accentate.
Il thread di ricezione rimane in attesa di messaggi inviati dal server e li decodifica utilizzando sempre la codifica UTF-8 per garantire una corretta visualizzazione. Quando arriva un nuovo messaggio, il client lo visualizza sulla console.
L'interfaccia utente fornisce all'utente le istruzioni necessarie per interagire con l'applicazione, inclusi i comandi per inviare messaggi e le notifiche sullo stato dell'applicazione, ad esempio l'avviso di CTRL-C per terminare l'esecuzione.
Inoltre, il client è dotato di comandi configurati dal server, tra cui il comando /list per tenere traccia degli utenti online nella chatroom, il comando /pvt per inviare messaggi privati a un client specifico e il comando /cmds per visualizzare la lista dei comandi disponibili.

Capitolo 3

Server

Il Server della Chatroom è progettato per consentire agli utenti di connettersi e comunicare utilizzando il protocollo TCP/IP. Il server mantiene una lista dei client connessi e gestisce la trasmissione dei messaggi tra di essi. Il server è implementato in modo tale da essere continuamente in grado di ricevere pacchetti da ogni dispositivo collegato. Per gestire simultaneamente le connessioni multiple, il server utilizza un approccio basato su thread, con un thread separato per ciascuna connessione attiva. Quando arriva un nuovo messaggio da inviare, il server verifica l'effettiva presenza del destinatario controllando una tabella interna degli utenti connessi. Se il destinatario è online, il server inoltra il messaggio al destinatario appropriato. Tuttavia, se il destinatario non è presente nel registro interno del server, viene inviata una risposta al mittente specificando che l'utente richiesto non è connesso o non è stato trovato.
Il Server della Chatroom offre le seguenti funzionalità principali:

- Gestione delle connessioni dei client.
- Trasmissione di messaggi broadcast a tutti i client connessi.
- Invio di messaggi privati tra client specifici.
- Fornitura della lista degli utenti connessi.
- Fornitura della lista dei comandi disponibili.

Il codice del server è suddiviso nelle seguenti sezioni:

- Gestione delle Connessioni: Il server accetta le connessioni dei client e li aggiunge alla lista dei client connessi.

- Trasmissione dei Messaggi: Il server gestisce la trasmissione dei messaggi broadcast e privati tra i client connessi.
- Gestione dei Comandi: Il server supporta diversi comandi che consentono agli utenti di interagire con il server.

Il server supporta i seguenti comandi:

- `/list`: Questo comando restituisce la lista degli utenti attualmente connessi alla chatroom. Quando un client invia questo comando al server, il server invia al client la lista degli utenti connessi.
- `/pvt`: Questo comando consente a un utente di inviare un messaggio privato ad un altro utente. Quando un client invia questo comando al server, il server invia al client la richiesta di specificare il destinatario del messaggio. Successivamente, il client invia il nome dell'utente destinatario al server. Alla fine, il server chiede al client il messaggio da inviare al destinatario. Il server quindi invia il messaggio privato al destinatario specificato. Se l'utente specificato come destinatario non è trovato nella lista degli utenti connessi o è offline, il server invia un messaggio al client mittente indicando che l'utente specificato non è stato trovato o è offline. Se il mittente tenta di inviare un messaggio privato a se stesso, il server invia un messaggio al client mittente indicando che non è possibile inviare messaggi privati a se stesso.
- `/cmds`: Questo comando restituisce la lista dei comandi disponibili nella chatroom. Quando un client invia questo comando al server, il server invia al client la lista dei comandi supportati.

Capitolo 4

Conclusioni

Il progetto di Programmazione di Reti per la realizzazione di una chat client-server in Python ci è stato di aiuto per poter applicare i concetti teorici a quelli pratici acquisiti durante il corso. Attraverso l'implementazione di un sistema di comunicazione, abbiamo dovuto organizzare la gestione delle connessioni, la trasmissione e ricezione dei messaggi e la gestione delle operazioni(quali i comandi personalizzati). Durante lo sviluppo del progetto, abbiamo approfondito i protocolli di rete, in particolare del protocollo TCP, e delle tecniche per gestire in modo efficiente le comunicazioni tra client e server.