“Relazione Chat Client-Server”

per il corso Programmazione di Reti
A.A. 2023/2024

Elisa Yan - 0001070882
elisa.yan@studio.unibo.it

15 maggio 2024

1 Introduzione

Per il progetto di Programmazione di Reti è stato richiesto di sviluppare un sistema che implementa una chat client-server in Python utilizzando socket programming.

Il server deve essere in grado di gestire più client contemporaneamente e di consentire agli utenti di inviare e ricevere messaggi in una chatroom condivisa.

2 Design ed implementazione

Il sistema è composto da due componenti principali: il server e il client.

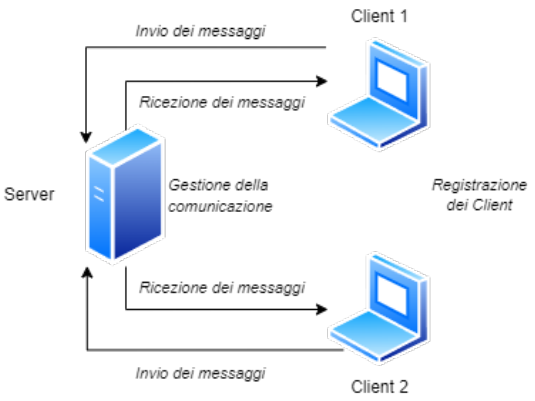

Figura 1: Funzionamento del sistema

2.1 Server

Il server, progettato per gestire una chat room, si mette in ascolto su una socket TCP per accettare le connessioni dei client. Quando un client si connette, gli viene richiesto

di inserire un nickname, che sarà associato alla connessione. Il server gestisce ogni client connesso tramite un thread separato, consentendo una comunicazione simultanea con più client.

Nel caso in cui un client si disconnetta, il server lo rimuove dalla lista degli utenti attivi e aggiorna l'elenco degli utenti, informando gli altri partecipanti della chat.

Il server può essere chiuso dall'amministratore tramite comandi inseriti nella console: "quit" o "shutdown". Quando questi comandi vengono eseguiti, il server invia un messaggio di chiusura a tutti i client connessi, termina tutte le connessioni attive in modo ordinato e chiude la socket.

2.2 Client

Il client è progettato utilizzando l'interfaccia grafica Tkinter per fornire una finestra di chat intuitiva e interattiva. All'avvio viene richiesto all'utente di inserire l'indirizzo del server. Viene verificata la validità dell'indirizzo IP inserito. In caso di errore, viene chiesto all'utente di reinserirlo. Dopo aver ottenuto un indirizzo valido, il client chiede il numero di porta e tenta di stabilire una connessione con il server. Se la connessione fallisce, l'utente è invitato a riprovare.

Una volta stabilita la connessione, il client avvia la GUI. I messaggi digitati dall'utente vengono inviati al server e poi distribuiti a tutti gli altri client connessi. I messaggi ricevuti dal server vengono visualizzati nell'area di testo della chat.

Il client gestisce la connessione al server in modo asincrono utilizzando thread separati: un thread per la GUI, che gestisce l'interazione dell'utente, e un thread per la ricezione dei messaggi dal server. In caso di disconnessione o errore di rete, il client gestisce la chiusura della connessione in modo ordinato e informa l'utente.

L'utente può chiudere la finestra di chat, il che provoca la chiusura della connessione al server e l'uscita dall'applicazione.

3 Esempio di utilizzo

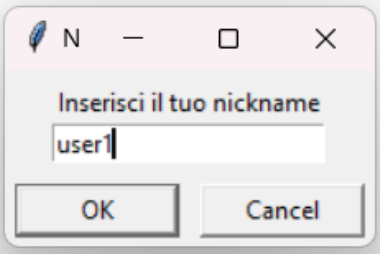

Figura 2: Inserimento del nickname

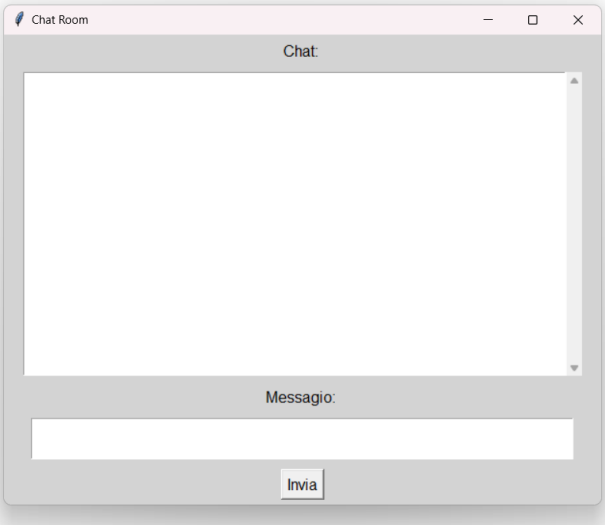

Figura 3: Entrata nella chat

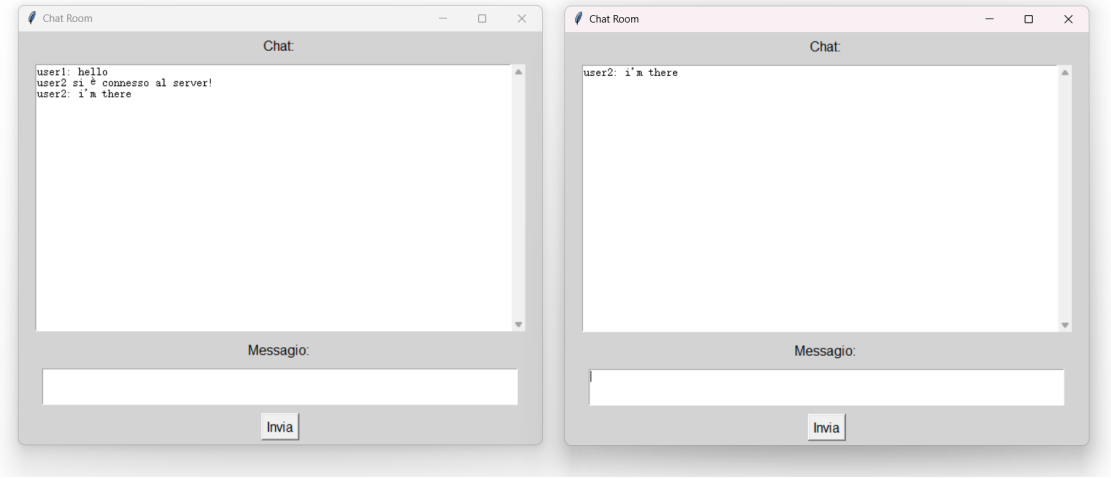

Figura 4: Entrata di un altro utente

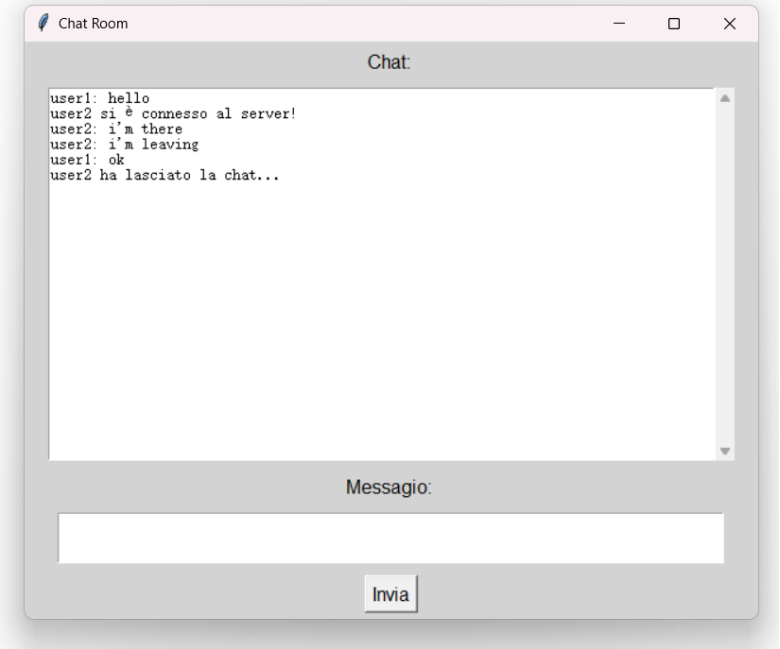

Figura 5: Uscita di un utente

4 Esecuzione dell'applicazione

Per eseguire l'applicazione del sistema di chat client-server, è necessario avere installato Python sul proprio sistema.

Prima deve essere avviato il server eseguendo il seguente comando sul terminale:

```
python .\server
```

Questo comando avvia il server, che si mette in ascolto dei client che cercano di connettersi. Successivamente il client può essere eseguito digitando il seguente comando sul terminale:

```
python .\client
```

Durante l'esecuzione del client, all'utente viene richiesto di specificare l'indirizzo IP e la porta del server a cui desidera connettersi. Per avviare più client si ripete la stessa operazione ma su un nuovo console.

Una volta che il client si connette con successo al server, può inviare messaggi nella chat e ricevere messaggi dagli altri client connessi.

5 Commenti finali

Il sistema attuale fornisce solo funzionalità di base per la trasmissione di messaggi, ma può essere esteso ulteriormente con nuove funzionalità e miglioramenti.