Relazione progetto di Programmazione di Reti

Matte Bagnolini : 0001071301

June 26 2024

Contents

1 Introduzione

Per questo progetto ho voluto realizzare un sistema di chat client-server (traccia 1). L'applicativo permette di creare una chat room, dove più clients possono scambiarsi messaggi in tempo reale. Il server ha il compito di gestire le connessioni dei clients.
Per realizzare questo progetto ho deciso di utilizzare il protocollo TCP per l'affidabilità che garantisce rispetto a UDP, che tuttavia risulta più veloce.

2 Requisiti

Di seguito vengono specificati i requisiti per utilizzare l'applicativo:

- Interprete per Python 3.*
- modulo 'socket' per la gestione di socket TCP
- modulo 'threading' per la gestione di thread
- modulo 'tkinter' per la realizzazione della GUI

3 Funzionamento del sistema

Di seguito viene descritto il funzionamento dell'applicativo

3.1 Server

Il server ha il compito di gestire le connessioni degli utenti e di inviarne i messaggi che riceve da essi.
Prima di tutto viene inizializzato il server presso un indirizzo ip e una porta esplicitati (nel nostro caso 127.0.0.1:53000) con il metodo .bind() del socket TCP. Successivamente il server viene messo in ascolto di un numero predefinito di client (nel nostro caso 10) e viene creato un nuovo thread che ha il compito di accettare nuove connessioni.
Una volta ricevuta una connessione da un client tramite metodo .accept(), l'indirizzo del client viene registrato in un dizionario, che associa il socket client al suo indirizzo. Successivamente viene creato un nuovo thread, dove vengono gestite le comunicazioni col client. Quindi per ogni client, viene avviato un thread che lo gestisce
In questo thread, il server gestisce il client, attribuendoli un username (che è reso univoco) e si mette in ascolto di messaggi. Una volta ricevuto, viene inviato in broadcast a tutti i client connessi alla chat. Se il messaggio è '{quit}', allora il server lo interpreta come segnale per rimuovere il client dalla chat, chiudendo la connessione e eliminandolo dal dizionario.

Il server è quindi in grado di:

- Gestire connessioni in ingresso
- Gestire messaggi in arrivo
- Inviare messaggi in broadcast
- Gestire la chiusura di connessione con i client

3.2 Client

Il client ha il compito di inviare messaggi al server, e di visualizzare tramite una semplice GUI i vari messaggi della chat.
All'avvio di un nuovo client, viene visualizzata una finestra di setup, dove viene chiesto di inserire l'indirizzo ip e il numero di porta del server a cui ci si vuole connettere. Successivamente, se i dati inseriti risultano corretti, ci si connette alla chat tramite l'utilizzo di un socket TCP. Una volta stabilita la connessione viene creato un nuovo thread che ha il compito di ricevere messaggi inviati dal server.
Tramite la GUI, è possibile scrivere un messaggio in un box di testo e inviarlo alla chatroom (cioè al server, che poi lo spedirà in broadcast) utilizzando un bottone o il tasto INVIO.

Il primo messaggio inviato dev'essere il nome utente con cui si vuole essere identificati (che verrà reso univoco dal server). I successivi messaggi saranno inviati nella chat normalmente, e sarà specificato per ognuno di essi il nome dell'utente e l'ora in cui è stato inviato (in realtà è il server che aggiunge alla stringa di messaggio l'orario in cui il messaggio viene ricevuto).
Nella chat vengono visualizzati i messaggi degli utenti, e gli avvisi di ingresso o uscita dalla stanza virtuale.

Il client è quindi in grado di:

- Connettersi a un server specificando indirizzo ip e porta
- Inviare e ricevere messaggi dal server
- Visualizzare tramite una semplice GUI i messaggi ricevuti
- Terminare la connessione con la chatroom

4 Guida Utente

L'applicazione può essere avviata tramite l'uso del terminale. Prima di avviare i vari client, è necessario far partire il server.
Per iniziare bisogna aprire un pannello terminale nella cartella del repository.

4.1 Server

Per avviare il server, eseguire il comando 'python3 src/server.py' nel terminale. Una volta partito esso rimane in attesa di connessioni finchè non viene interrotto manualmente da chi l'ha lanciato.
Nel pannello terminale dove viene eseguito il server viene stampato un log dove sono registrati tutti gli eventi della chat (ingressi, uscite, messaggi).
Per interromperne l'esecuzione bisogna digitare il comando 'Ctrl+C' nel terminale che lo ha avviato.

4.2 Client

Una volta avviato il server, è possibile avviare i vari client (fino a 10 contemporaneamente) utilizzando il comando 'python3 src/client.py'. All'esecuzione del comando viene aperta una finestra di setup dove bisogna inserire l'indirizzo ip e la porta del server a cui vogliamo collegarci (nel nostro caso 127.0.0.1:53000 come detto sopra). Successivamente viene aperta la GUI con cui possiamo inviare e ricevere messaggi.
È possibile interrompere l'esecuzione del client inviando nella chat la stringa '{quit}', oppure chiudendo la rispettiva finestra della GUI. La disconnessione dal server è gestita automaticamente.