Chat Client-Server

Nicolò Penserini 0001080348
Alessandro Ambrogiani 0001080493

Maggio 2024

Capitolo 1

Server

Per realizzare le funzionalità del server è necessario importare, dalla libreria socket, la funzione socket per poter creare un socket e le costanti AF INET, per indicare che il protocollo usato è di tipo IPv4, e SOCK STREAM, che indica che il socket deve essere orientato alla connessione secondo il protocollo TCP.
Le funzionalità necessarie sono svolte da thread diversi che implementano 2 funzioni:

- **accetta connessioni in entrata** rimane attiva dall'avvio dell'applicazione, svolgendo il la funzione di ascolto del server, e ogni volta che avviene una richiesta di connessione da parte di un client (attraverso un flag SYN) la accetta, memorizzandone l'indirizzo, e dà inizio a un nuovo thread che si occuperà della gestione del client.

- **gestisce client** si occupa della gestione di un singolo client. Per ognuno di essi riceve i messaggi inviati e li invia a tutti i client connessi, sfruttando la funzione **broadcast**, che manda il messaggio con il nome del mittente a tutti i client memorizzati. Quando il messaggio che riceve è quello scelto per la terminazione della connessione, manda l'ultimo messaggio al client, termina la connessione chiudendo il socket associato al client e manda un messaggio agli altri client scrivendo che l'utente si è disconnesso.

Nel thread main si crea il nuovo thread che si occupa di accettare le connessioni e si aspetta che termini attraverso la funzione join.

Capitolo 2

Client

Per realizzare le funzionalità del client, oltre alle librerie già utilizzate nel server, abbiamo sfruttato la libreria tkinter per realizzare un'interfaccia grafica attraverso cui leggere i messaggi in arrivo e inviarne di nuovi.
Per un client viene creato un socket per la connessione al server, che avviene attraverso la funzione connect, che invia un messaggio con flag SYN.
Una volta stabilita la connessione viene creato un thread che si occupa di ricevere i messaggi attraverso la funzione recv, che è una funzione bloccante e quindi deve essere chiamata da un thread diverso da quello principale. Quando si ricevono dei messaggi vengono mostrati nell'interfaccia grafica.
Per inviare messaggi si sfrutta la funzione **invio** che invia messaggi sul socket connesso con il server. Se il messaggio inviato è quello per terminare la connessione si attende l'ultimo messaggio inviato dal server e si chiudono il socket e la finestra dell'interfaccia grafica.
Alla chiusura della finestra viene inviato un messaggio di chiusura.

Appendice A

Utilizzo

Per eseguire i due script è necessario un interprete python. Prima deve essere eseguito lo script Server.py, che si mette in attesa di ricevere connessioni e ogni volta che si collega un nuovo client viene stampato l'indirizzo e il numero di porta, che vengono stampati anche quando un client si disconnette. Quando il server è attivo in attesa di connessioni possono essere lanciate più istanze dello script client.py. Appena lanciato lo script client.py viene richiesto di inserire l'indirizzo IP e la porta del server in ascolto (per l'implementazione attuale 127.0.0.1:53000); se le informazioni non sono inserite in modo corretto viene chiesto di ripetere l'inserimento.
Una volta effettuato il collegamento al server si apre la finestra di interfaccia grafica, in cui bisogna inserire il nome dell'utente appena collegato. Dopodichè è possibile inviare messaggi, leggere i messaggi in arrivo e uscire dalla chat scrivendo "{quit}" in chat o chiudendo la finestra.