Progetto per il corso di "Programmazione di reti"

Sistema di Chat Client-Server

A.A. 2023/24

Gioele Bucci
gioele.bucci@studio.unibo.it

7 maggio 2024

0.1 Introduzione

L'applicazione realizzata è una semplice chatroom basata su architettura client-server multithreaded, implementata in Python. La comunicazione tra client e server avviene tramite connessione TCP, impiegando la libreria `socket`.

0.1.1 Funzionalità

- **Messaggistica**: I client possono inviare messaggi nella chat: tali messaggi vengono ricevuti dal server e ritrasmessi a tutti gli altri client.
- **Comandi**: I client possono inviare comandi al server, che li elabora ed esegue le azioni corrispondenti. I comandi si distinguono in comandi utente e comandi riservati, utilizzabili solo dall'amministratore.
- **Gestione di errori e disconnessione**: Quando un client si disconnette, volontariamente o a causa di un errore, il server lo rimuove dalla lista dei client e trasmette un messaggio a tutti gli altri client notificando la disconnessione.

0.2 Funzionamento

Il server mantiene una porta in ascolto per le connessioni in ingresso. All'arrivo di un tentativo di connessione viene creato un thread per la gestione del relativo client.
All'avvio, il client richiede all'utente di inserire un nickname. Se il nickname inserito è "admin", viene richiesta anche una password. Entrambe le informazioni vengono memorizzate localmente in attesa di essere verificate dal server. Successivamente il client istanzia un socket per stabilire la connessione con il server e crea due thread: uno per la ricezione dei messaggi e uno per l'invio.

0.2.1 Verifica della connessione del client

1. Il client richiede all'utente un nickname (ed eventualmente una password, se il nickname era "admin") ed effettua la connessione al server.
2. Il server invia un segnale di `NICK`, al quale il client risponde con il nickname. Se tale nickname è già in uso il server termina la connessione, altrimenti risponde con `NICK_OK`
3. Se il nickname è "admin", il server invia un segnale aggiuntivo di `PASSW`, al quale il client risponde con la password. Il server confronta la password ricevuta con quella memorizzata e se corrispondono invia un segnale di `PASSW_OK` e autentica il client come amministratore: in caso contrario la connessione viene terminata.

   Nota: la password è memorizzata in chiaro nello script del server ma in un'applicazione reale sarebbe necessario utilizzare un metodo di più sicuro di memorizzazione, come ad esempio l'hashing delle password in un database.

0.3 Comandi

Di seguito una lista dei comandi disponibili, con la relativa sintassi. Per ciascun comando si specifica anche se è riservato, ovvero utilizzabile solo dall'admin. Ciascun comando è inviato da un client e viene elaborato dal server, con l'eccezione del comando `exit` che viene eseguito direttamente sul lato client.

Comando	Descrizione	Riservato
`kick <user> <reason>`	espelle un utente dalla stanza	SI
`list`	elenca tutti gli utenti connessi	NO
`msg <user> <message>`	invia un messaggio privato ad un utente	NO
`whoami`	restituisce il proprio nickname	NO
`whois <user>`	restituisce informazioni relative al client	SI
`exit`	*(client-side)* termina la connessione	NO

0.4 Utilizzo

Per utilizzare l'applicazione, aprire prima il terminale e avviare prima il server e in seguito il client. Assicurarsi di specificare l'indirizzo IP e la porta come argomenti durante l'avvio.

```
python .\server.py <ip> <port>
python .\client.py <ip> <port>
```

È anche possibile non fornire alcun argomento agli script, in tal caso verrà utilizzato il valore di default `127.0.0.1:55555`.
Per effettuare l'accesso come amministratore, utilizzare come nickname *"admin"* e come password *"password"*.

0.5 Modularizzazione dell'applicazione

L'applicazione è stata modularizzata per favorire una maggiore estensibilità e manutenibilità. Di seguito si elencano i moduli aggiunti, ciascuno corredato da una breve descrizione.

- `utils.py`: Contiene funzioni comuni utilizzate sia dal client che dal server. Contiene le funzioni per inviare e ricevere messaggi ad un socket e per estrarre l'indirizzo di connessione dagli argomenti passati a riga di comando.
- `commands.py`: Definisce i comandi eseguibili dal server, al quale è lasciato il compito di definire quali di questi comandi sono disponibili e specificarne il livello di accesso.
- `signals.py`: Definisce i segnali che il server può trasmettere ai client.
- `serverutils.py`: Modulo riservato al server contenente funzioni per l'invio di comandi e messaggi broadcast. Mantiene anche la lista dei client connessi e i relativi nickname.