Federico De Leonardis matr:0001071569

1. Introduzione

Il sistema di chat client-server implementato utilizza il socket programming in Python per permettere la comunicazione in tempo reale tra più client attraverso un server centrale. Questo documento descrive l'implementazione e le caratteristiche del sistema, inclusi i requisiti per l'esecuzione del codice.

2. Requisiti

- Python 3.x
- Modulo `socket` di Python (incluso nella libreria standard)
- Modulo `threading` di Python (incluso nella libreria standard)
- Modulo `tkinter` per la GUI usata come interfaccia con l'utente

3. Implementazione del Server

Il server (`chat_server.py`) è progettato per accettare connessioni da più client e gestire la trasmissione dei messaggi tra di loro.

Abbiamo innanzitutto due dizionari per la registrazione dei client in ingresso e dei relativi nomi associati agli indirizzi, clients e addresses rispettivamente.

Configurazione del Server: Il server viene configurato per ascoltare le connessioni in ingresso sulla porta 53000.

- **Accettazione delle Connessioni:** Quando un nuovo client si connette, il server accetta la connessione e memorizza le informazioni dell'utente.

  Definiamo la funzione accept_entrance_connections che gestisca le connessioni in entrata che si mette in ascolto sul socket quando viene invocata. Appena arriva una richiesta di connessione da un client, il server gli restituisce un messaggio di saluto accompagnato da alcune indicazioni sull'utilizzo della chat. Viene quindi preso il dizionario addresses e aggiornandolo con il dato relativo al nuovo client

  Infine attiva il Thread() verso questo client andando ad invocare la funzione handle client

- **Ricezione dei Messaggi:** I messaggi inviati dai client vengono ricevuti e trasmessi a tutti gli altri client connessi.

  La funzione handle client controlla il corretto funzionamento del client socket, per poi mandare, con la funzione broadcast, il messaggio ricevuto da un client a tutti gli altri, incluso il client mittente

- **Gestione delle Disconnessioni:** Se un client si disconnette, il server rimuove il client dall'elenco delle connessioni attive con la funzione remove.

4. Implementazione del Client

Il client (`chat_client2.py`) consente agli utenti di connettersi al server, inviare messaggi e ricevere messaggi dagli altri utenti nella chatroom.

- **Connessione al Server:** Il client si connette al server specificando l'IP e la porta del server, viene creato il socket del client per avviare la connessione e un thread per la ricezione dei messaggi.
- **GUI:** A questo iniziamo il ciclo principale della gui, scritta con l'ausilio di tkinter (strumento incluso in python). Il ciclo con cui l'utente è in grado, tramite una finestra, di visualizzare i messaggi da parte degli altri utenti e scriverne di nuovi deve essere infinito. Questo perché finché è connesso deve poter visualizzare e inviare dati a suo piacimento. Termina l'esecuzione quando riceve un messaggio, e quindi prosegue aggiungendo il messaggio a msg_list (cioè la lista di stringhe che viene poi visualizzata sul frame)
- **Invio dei Messaggi:** L'utente può inserire un messaggio che viene inviato al server e quindi distribuito agli altri client.
- **Ricezione dei Messaggi:** Il client ascolta i messaggi in arrivo dal server e li visualizza sulla console.

5. Gestione degli Errori

Entrambe le implementazioni gestiscono gli errori e le eccezioni in modo appropriato. Il server gestisce le connessioni perse e i client gestiscono errori di lettura e scrittura, assicurando che il sistema sia robusto e affidabile.

6. Esecuzione del Codice

Si può creare un eseguibile da condividere con altre persone, installando il pacchetto PYINSTALLER (da shell di DOS: python -m pip install pyinstaller) e fornendo a Pyinstaller in ingresso il file relativo al client CHAT

1. **Avviare il Server:**
    - Aprire un terminale (o una finestra di comando).

Eseguire il comando seguente per avviare il server in ambiente bash :

```
python chat_server.py
```

    -
    - Il server inizierà ad ascoltare le connessioni in ingresso sulla porta 53000.
2. **Avviare uno o più Client:**
    - Aprire un altro terminale per ogni client che si desidera connettere al server.

Eseguire il comando seguente per avviare il client in ambiente bash:

```
python chat_client2.py
```

    - Quando viene avviato il client, verrà richiesto di inserire un nome utente. Dopo aver inserito il nome utente, il client si connetterà al server e sarà pronto per inviare e ricevere messaggi.
3. **Comunicare nella Chatroom:**
    - Ogni client può inviare messaggi che verranno distribuiti a tutti gli altri client connessi.

    - I messaggi inviati da un client appariranno nei terminali di tutti gli altri client connessi.

7. Considerazioni Aggiuntive

- La comunicazione avviene su localhost, quindi server e client devono essere eseguiti sulla stessa macchina o su macchine diverse all'interno della stessa rete locale.
- È possibile migliorare la sicurezza della chat implementando la crittografia dei messaggi.
- L'interfaccia attuale è minimale e potrebbero essere fatti cambiamenti per aumentare l’accessibilità