PyChat

Progetto di Programmazione di reti

Università di Bologna

Martin Tomassi; 0001077737
Francesco Pazzaglia; 0001077423
Luca Casadei; 0001069237

29 aprile 2024

Indice

1 Introduzione **2**

2 Client **3**

3 Server **4**
3.1 Introduzione 4
3.2 Funzionalità Principali 4
3.3 Implementazione 5
3.3.1 Struttura del Codice 5
3.4 Utilizzo 5
3.4.1 Comandi Supportati 5

Capitolo 1

Introduzione

Questa è la documentazione del progetto di Programmazione di Reti relativo alla Traccia 1: *Sistema di Chat Client-Server* in Python. Il progetto prevede la realizzazione di una chat che consente a più utenti di connettersi a un server centrale e comunicare tra loro in una chatroom condivisa.
Il progetto è composto da due componenti principali: il **client** e il **server**.
Il server gestisce le connessioni dei client e invia i messaggi in modalità broadcast all'interno della chat. Il client, invece, permette agli utenti di connettersi al server tramite protocollo TCP, di inviare messaggi e di visualizzare i messaggi ricevuti dalla chatroom degli altri utenti. Quando un client si connette al server, può vedere solo i messaggi inviati dopo che la connessione è stata stabilita; i messaggi precedenti non vengono "salvati" nel server.
E' stato scelto di utilizzare il protocollo a livello di trasporto chiamato **T**ransimission **C**ontrol **P**rotocol perché garantisce la consegna affidabile dei dati, quindi nel caso di una chat è fondamentale che i messaggi vengano inviati e ricevuti in maniera corretta senza alcun tipo di perdita.

Capitolo 2

Client

Il client utilizza un approccio connection-oriented IPv4 attraverso il modulo *socket* di Python per stabilire una connessione con il server della chatroom. Prima di avviare l'applicazione, l'utente deve specificare l'indirizzo IP del server e un nome utente univoco, che servirà a identificarlo all'interno della chatroom. Questi parametri vengono passati come argomenti al momento dell'esecuzione del client.
Una volta avviato, il client crea un socket e tenta di stabilire una connessione con il server utilizzando l'indirizzo IP e la porta specificati. Se la connessione viene stabilita con successo, il client invia il nome utente al server per identificarsi.
Dopo la connessione, il client avvia due thread separati per gestire la trasmissione e la ricezione dei messaggi.
Il thread di trasmissione consente all'utente di digitare un messaggio dall'input del terminale e inviarlo al server tramite il socket. Prima di inviare il messaggio, viene applicata la codifica UTF-8 per garantire la compatibilità con caratteri speciali come le lettere accentate.
Il thread di ricezione rimane in attesa di messaggi inviati dal server e li decodifica utilizzando sempre la codifica UTF-8 per garantire una corretta visualizzazione. Quando arriva un nuovo messaggio, il client lo visualizza sulla console.
L'interfaccia utente fornisce all'utente le istruzioni necessarie per interagire con l'applicazione, inclusi i comandi per inviare messaggi e le notifiche sullo stato dell'applicazione, ad esempio l'avviso di CTRL-C per terminare l'esecuzione.
Inoltre, il client tiene traccia degli utenti online all'interno della chatroom e offre la possibilità di visualizzare la lista degli utenti connessi in un determinato momento tramite il comando */list*.

Capitolo 3

Server

3.1 Introduzione

Questo documento fornisce la documentazione per il Server della Chatroom, sviluppato come parte del progetto di Programmazione di Reti. Il server è progettato per consentire agli utenti di connettersi e comunicare tramite una chatroom utilizzando il protocollo TCP/IP.
Il server è implementato in modo tale da essere continuamente in grado di ricevere pacchetti da ogni dispositivo collegato. Per gestire simultaneamente le connessioni multiple, il server utilizza un approccio basato su thread, con un thread separato per ciascuna connessione attiva.
Quando arriva un nuovo messaggio da inviare, il server verifica l'effettiva presenza del destinatario controllando una tabella interna degli utenti connessi. Se il destinatario è online, il server inoltra il messaggio al destinatario appropriato. Tuttavia, se il destinatario non è presente nel registro interno del server, viene inviata una risposta al mittente specificando che l'utente richiesto non è connesso o non è stato trovato.

3.2 Funzionalità Principali

Il Server della Chatroom offre le seguenti funzionalità principali:

- Gestione delle connessioni dei client.
- Trasmissione di messaggi broadcast a tutti i client connessi.
- Invio di messaggi privati tra client specifici.
- Fornitura della lista degli utenti connessi.

- Fornitura della lista dei comandi disponibili.

3.3 Implementazione

Il server è implementato in Python e utilizza il modulo `socketserver` per gestire le connessioni dei client. Il server mantiene una lista dei client connessi e gestisce la trasmissione dei messaggi tra di essi.

3.3.1 Struttura del Codice

Il codice del server è suddiviso nelle seguenti sezioni:

- **Gestione delle Connessioni**: Il server accetta le connessioni dei client e li aggiunge alla lista dei client connessi.
- **Trasmissione dei Messaggi**: Il server gestisce la trasmissione dei messaggi broadcast e privati tra i client connessi.
- **Gestione dei Comandi**: Il server supporta diversi comandi che consentono agli utenti di interagire con il server.

3.4 Utilizzo

Per avviare il server, eseguire il file Python `server.py`. Il server rimarrà in ascolto per le connessioni dei client sulla porta specificata nel file di configurazione.

3.4.1 Comandi Supportati

Il server supporta i seguenti comandi:

- `/list`: Fornisce la lista degli utenti connessi alla chatroom.
- `/pvt`: Consente di inviare un messaggio privato a un altro utente.
- `/cmds`: Fornisce la lista dei comandi disponibili.