Elaborato Programmazione di Reti

Marco Paggetti

17 maggio 2024

Traccia 1: Sistema di Chat Client-Server

Implementare un sistema di chat Client-Server in Python utilizzando socket programming. Il server deve essere in grado di gestire più client contemporaneamente e deve consentire agli utenti di inviare e ricevere messaggi in una chatroom condivisa. Il client deve consentire agli utenti di connettersi al server, inviare messaggi alla chatroom e ricevere messaggi dagli altri utenti.

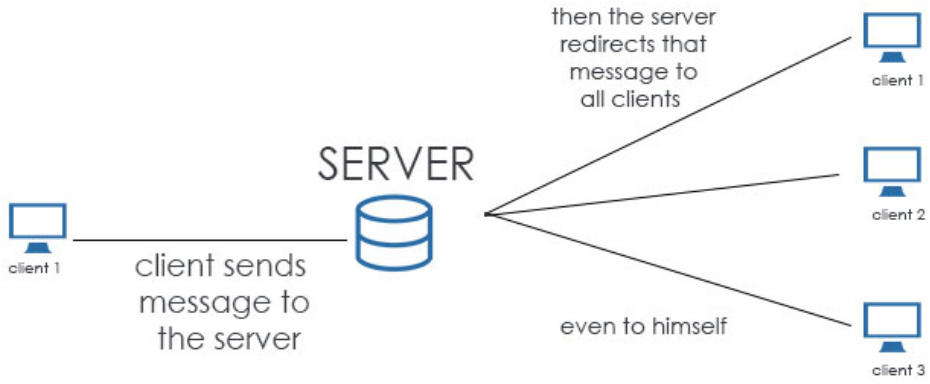

Figura 1: Architettura della Chat Client-Server

Indice

1 Funzionamento

Il codice si compone di due parti: il programma relativo al Server della Chat e quello relativo al Client della Chat (entrambi scritti in linguaggio Python). Entrambi i programmi fanno uso di socket (per la comunicazione di rete) e di thread (per la gestione di più azioni contemporaneamente).

1.1 Applicazione Server

1.1.1 Definizione degli elementi iniziali

Al momento del lancio, il Server richiede all'utente di inserire due elementi: l'indirizzo IP e la porta su cui il Server opera. Utilizzando queste informazioni, viene generato l'indirizzo del Server, che è la coppia formata dai due dati richiesti in input. Successivamente, viene creato un socket TCP per il Server, che rappresenta la sua interfaccia per la comunicazione. Il protocollo TCP è stato scelto al posto del protocollo UDP in quanto è più adatto al progetto in questione. Il socket viene poi associato all'indirizzo del Server. Una volta associato l'indirizzo del Server al socket, quest'ultimo viene messo in ascolto per eventuali connessioni in ingresso. Infine, vengono definite due liste: una per registrare i Client connessi al Server e una per registrare i relativi nickname. Inoltre, vengono create altre due liste: una che contiene, in ogni indice, un valore che funge da flag per la chiusura del relativo thread di gestione dei messaggi del Client, mentre l'altra serve per tenere traccia dei thread di gestione delle connessioni, attivati.

1.1.2 Avvio del Server

Dopo la fase di configurazione iniziale, il Server procede creando e avviando un thread dedicato all'accettazione delle connessioni in entrata. Una volta creato il thread, il flusso principale del Server si dedica a scrivere un messaggio ogni 5 secondi, per indicare che il Server è attivo. In caso di interruzione da tastiera (CTRL+C), il Server inizia la procedura di chiusura: disconnette tutti i Client connessi in quel momento e svuota le liste dei Client e dei relativi nickname. Prima di terminare completamente, il Server si preoccupa di chiudere il socket e di attendere la terminazione di tutti i thread di gestione delle connessioni presenti nella lista apposita.

1.1.3 Accettazione delle connessioni

Il thread dedicato all'accettazione delle connessioni in entrata si mette in ascolto sul socket. Non appena arriva una richiesta di connessione da un Client, il thread attende la ricezione del primo messaggio del Client, che conterrà esclusivamente il nickname dell'utente connesso. Successivamente, il Client e il relativo nickname vengono registrati nelle liste corrispondenti. A questo punto, il Server invia un messaggio di benvenuto al Client e diffonde immediatamente un messaggio broadcast (a tutti gli utenti connessi alla chat in quel momento), annunciando che il nuovo Client si è unito alla conversazione. Contemporaneamente alla creazione e all'avvio di un thread per gestire le connessioni con il Client appena connesso, viene aggiunto un flag 0 (che significa "non chiudere") alla lista dei flag di chiusura, e l'indice del corrispondente valore di flag nella lista, viene salvato per il thread. Tale thread viene salvato nella lista appositamente creata durante la fase di configurazione iniziale. Il codice gestisce i tentativi di messa in ascolto su socket chiuse terminando il thread e gestisce eventuali eccezioni ulteriori stampando il traceback delle istruzioni che hanno generato l'eccezione, per poi procedere alla chiusura del thread.

1.1.4 Gestione delle connessioni

Il thread di gestione delle eccezioni ha due compiti principali. In primo luogo, avvia il thread che si occuperà di inviare i ping al Client. In secondo luogo, imposta un timeout di 10 secondi sul Client, dopo il quale, se non riceve alcun messaggio, genera un'eccezione. Dopo aver impostato il timeout, il thread attende l'arrivo di un messaggio dal Client. Viene effettuato un controllo preliminare sul messaggio ricevuto per verificare che non sia vuoto, indicando che la connessione si è interrotta e il Client si è disconnesso. Se il messaggio è vuoto, il Client disconnesso (se non lo è già) e il thread di gestione delle connessioni viene terminato. Se il messaggio non è vuoto, vengono effettuati ulteriori controlli: se è un messaggio di ping, viene semplicemente stampata a terminale la notifica di ricezione. Altrimenti, si verifica che il messaggio sia diverso dal messaggio di quit. Se lo è, al messaggio viene aggiunto in testa il nickname del Client che lo ha inviato e il messaggio completo viene inviato a tutti i Client connessi. Se invece è un messaggio di quit, il flag di chiusura del Client specifico viene impostato a 1 ("chiudi") e si attende la terminazione del thread che gestisce il ping. Non appena questo termina, sia il Client sia il relativo nickname vengono rimossi dalle liste corrispondenti. Infine, dopo aver inviato un messaggio a tutti i Client connessi in quel momento, annunciando che il Client ha lasciato la chat, viene stampata a terminale la lista dei nickname dei Client rimasti nella chat. Il codice gestisce l'eccezione generata dallo scadere del timeout del Client (ovvero quando il Server non riceve più alcun messaggio dal Client per più di 10 secondi), disconnettendo il Client e terminando il thread di gestione della connessione del Client corrispondente. Gestisce anche l'eccezione generata da un tentativo di ricezione di un messaggio su un socket chiuso, terminando semplicemente il thread dopo aver lanciato un messaggio che spiega la causa dell'eccezione. Infine, tutte le altre eccezioni vengono gestite stampando l'errore, accompagnato dal traceback delle istruzioni che hanno portato al problema, e chiudendo il thread.

1.1.5 Disconnessione dei Client

Ogni volta che un Server disconnette un Client, invia inizialmente un messaggio di "quit" al Client da disconnettere. Questo permette al Client di rilevare la richiesta e di disconnettersi autonomamente. Successivamente, il Server rimuove il Client e il suo nickname dalle liste corrispondenti e invia un messaggio a tutti gli utenti connessi alla chat, per informarli della disconnessione del Client. Il codice gestisce l'eccezione generata dal tentativo di inviare un messaggio su un socket già chiuso, stampando semplicemente un messaggio sul terminale per indicare che l'operazione non è stata eseguita correttamente.

1.1.6 Messaggio broadcast

Ogni volta che il Server deve mandare un messaggio broadcast (cioè a tutti i Client connessi alla chat), scorre la lista dei Client attualmente connessi alla chat e invia lo stesso messaggio a ciascuno di essi.

1.1.7 Gestione dell'invio dei ping al Client

Il thread responsabile della gestione dei ping invia un messaggio contenente "[ping]" al Client di riferimento ogni 3 seondi. Questo thread continua a svolgere le sue funzioni fino a quando non rileva che il proprio flag di

chiusura è stato impostato ad 1. Il codice gestisce due tipi specifici di eccezioni: quelle generate dalla chiusura inaspettata della connessione e quelle generate dal tentativo di inviare un messaggio su un socket chiuso. In entrambi i casi, viene visualizzato un messaggio sul terminale che spiega brevemente il problema e informa che il thread responsabile del ping verrà interrotto. Tutte le altre eccezioni vengono gestite visualizzando un messaggio di errore sul terminale, accompagnato dal traceback, seguito dalla chiusura del thread.

1.2 Applicazione Client

1.2.1 Definizione degli elementi iniziali

Al momento dell'avvio, il Client richiede all'utente di fornire tre informazioni: l'indirizzo IP del Server a cui connettersi, la porta su cui il Server è in esecuzione e il Nickname che l'utente desidera utilizzare all'interno della chat. Utilizzando le prime due informazioni, viene generato l'indirizzo del Server e successivamente viene creato il socket del Server. Una volta completati questi passaggi, il Client tenta di stabilire una connessione con il Server. Dopo aver stabilito la connessione, viene creata l'interfaccia grafica della chat. Questa interfaccia è composta da una sezione in cui vengono visualizzati tutti i messaggi della chat (dotata di una barra di scorrimento per navigare all'interno della lista), una sezione dedicata all'inserimento del messaggio da inviare (in cui viene immediatamente visualizzato un messaggio che indica che quella è l'area per la scrittura) e un pulsante "Invio" che consente di inviare il messaggio (è anche possibile inviare il messaggio premendo il tasto di invio sulla tastiera). L'interfaccia grafica implementa un protocollo di chiusura, che automaticamente imposta e invia un messaggio di quit. Successivamente, vengono creati due flag di chiusura (dove il valore 0 indica "non chiudere" e il valore 1 indica "chiudere"), uno dei quali si riferisce al thread di gestione del ping, mentre l'altro si riferisce al thread di ricezione dei messaggi.

1.2.2 Avvio del Client

Dopo la fase di configurazione iniziale, vengono creati e avviati due thread. Uno di questi è responsabile dell'invio del ping al Server, mentre l'altro si occupa della gestione della ricezione dei messaggi. Infine, il flusso principale del Client si dedica a mantenere attiva l'interfaccia grafica, che viene avviata immediatamente dopo l'avvio dei thread precedenti.

1.2.3 Connessione al server

Quando il Client tenta di connettersi al Server, cerca di stabilire una connessione con l'indirizzo del Server, che è stato precedentemente creato utilizzando le informazioni fornite dall'utente. Una volta stabilita la connessione, il Client invia automaticamente un primo messaggio al Server, contenente esclusivamente il nickname scelto dall'utente. Il codice gestisce i tentativi di connessione a un Server non avviato, visualizzando un messaggio che specifica che il Server non è attualmente raggiungibile. Questo viene implementato attraverso l'uso di un ciclo while, che permette all'utente di ritentare la connessione in modo automatico. In caso di altre eccezioni, il codice prevede la visualizzazione di un messaggio di errore sul terminale, accompagnato dal traceback dell'errore.

1.2.4 Gestione della ricezione dei messaggi

Il thread responsabile della ricezione dei messsaggi inizia impostando un timeout di 10 secondi sul Server. Alla scadenza di questo timeout, viene generata un'eccezione. Dopo aver impostato il timeout, il thread entra in modalità di attesa per ricevere un messaggio dal Server. Non appena riceve un messaggio, effettua immediatamente un controllo su di esso: se il messaggio è un ping, viene visualizzata una conferma di ricezione del ping del Server sul terminale; se invece il messaggio non è né un messaggio di quit né un ping, viene aggiunto alla lista dei messaggi visibili nell'interfaccia grafica della chat (oltre ad essere visualizzato sul terminale). Nel caso in cui il messaggio sia un messaggio di quit, viene prima visualizzato un messaggio che indica che sta avvenendo la disconnessione. Successivamente, viene impostato a 1 il flag di chiusura del thread legato al ping e si attende la sua terminazione. Una volta terminato il thread del ping, vengono chiusi sia il socket che l'interfaccia grafica della chat, e viene terminato il thread della ricezione dei messaggi.

1.2.5 Gestione dell'invio dei messaggi

Ogni messaggio viene raccolto dalla sezione di inserimento dedicata quando si preme il pulsante "Invio" o il tasto di invio sulla tastiera. Successivamente, il contenuto della sezione di inserimento viene azzerato e il messaggio viene inviato al Server. Viene immediatamente effettuato un controllo sul messaggio: in caso di messaggio di quit, entrambi i flag di chisura vengono impostati ad 1. Dopo aver atteso la terminazione dei due thread (quello di ping e quello di ricezione dei messaggi), il socket e l'interfaccia grafica dell'applicazione vengono chiusi.

1.2.6 Gestione dell'invio dei ping al Server

Il thread responsabile della gestione dei ping invia un messaggio contenente "[ping]" al Server ogni 3 secondi. Questo thread continua a svolgere le sue funzioni fino a quando non rileva che il proprio flag di chiusura è stato impostato a 1, momento in cui il thread viene terminato. Tutte le eccezioni che possono verificarsi vengono gestite visualizzando un messaggio di errore sul terminale, accompagnato dal traceback dell'errore, seguito dalla chiusura del thread.

2 Requisiti

Per eseguire questi script, è indispensabile avere Python 3 installato sul proprio sistema. Inoltre, il sistema deve supportare le librerie "socket", "threading" e "tkinter" di Python. Queste librerie sono incluse nella distribuzione standard di Python, quindi non dovrebbero essere necessarie installazioni aggiuntive. Per utilizzare l'applicazione, è necessario avviare prima il Server e successivamente il Client. Entrambi gli script richiedono che l'utente fornisca l'indirizzo del Server e la porta su cui il Server è in ascolto. Nel client, l'utente deve anche inserire un nickname che verrà utilizzato all'interno della chat. Per eseguire l'applicazione su più dispositivi fisici, è necessario che questi siano connessi alla stessa rete LAN. Il Server dovrà essere lanciato su uno di questi dispositivi. Per connettere i vari dispositivi, è sufficiente specificare, come indirizzo del Server, l'indirizzo IP del dispositivo fisico all'interno della LAN (che può essere visualizzato utilizzando il comando "ipconfig" nel terminale).

3 Considerazioni finali

Questi script rappresentano una semplice implementazione di un sistema di chat Client-Server. Tuttavia, esistono diverse aree che potrebbero beneficiare di miglioramenti. Ad esempio, potrebbe essere implementata una funzionalità di controllo di sicurezza, così come l'invio di messaggi in forma criptata. Il protocollo di comunicazione attualmente è molto semplice e non supporta funzionalità avanzate come la trasmissione di file o la chat privata tra due client. Inoltre, i messaggi che vengono visualizzati sui vari terminali potrebbero essere migliorati per descrivere in modo più dettagliato ciò che sta accadendo, al fine di evitare possibili incomprensioni. Infine, l'interfaccia utente del client è stata progettata per essere il più semplice possibile, dando priorità alla gestione della chat rispetto all'aspetto grafico.