Relazione di Programmazione di Reti

Elaborato - Traccia 1: Sistema di Chat Client-Server

Introduzione

La seguente documentazione descrive l'implementazione e l'utilizzo di un sistema di chat client/server realizzato ed implementato utilizzando il linguaggio di programmazione Python e la libreria socket.

Il presupposto di partenza consiste nel creare un sistema che consente a più client di connettersi ad un server centrale che ospita una chatroom condivisa tra tutti gli utenti, inviando loro in broadcast tutti i messaggi che ognuno spedisce dal proprio client, il tutto in tempo reale.

Scopo

Lo scopo della realizzazione di questo elaborato è di dimostrare la capacità e la comprensione di programmazione di rete (e concorrente) facendo uso del concetto di *socket programming* e di instaurazione delle connessioni, nonché elaborazione della comunicazione, tra host nello standard ***TCP/IP***

Architettura del sistema

Il sistema è sostanzialmente composto da due componenti attive e principali: un server e (diversi) client.

Server

Il server è il responsabile della gestione delle connessioni dei client e del funzionamento logico della chatroom.

È costantemente in ascolto su una specifica porta ed accetta in loop le richieste di connessione dei vari client.

Una volta instaurata la connessione, ogni client può inviare messaggi al server che, a sua volta, procederà ad inoltrare i messaggi agli altri client connessi, quindi in modalità broadcast.

Il server utilizzerà il principio di multithreading per poter gestire le molteplici connessioni con i client in maniera contemporanea e concorrente, consentendo così ai diversi client di partecipare alle comunicazioni senza interferire l'uno con l'altro.

Client

Il client è la logica presente dietro l'interfaccia utente (command-line, per questo elaborato) che consente all'utilizzatore di connettersi al server, inviare i propri messaggi e ricevere quelli inviati dagli altri utenti nella chatroom condivisa.

Ogni client si connette al server tramite un socket TCP/IP e può inviargli messaggi testuali che saranno poi da lui elaborati ed inoltrati.

Implementazione

Server

Il server è implementato mediante l'uso del modulo **core** e della libreria **socket** del linguaggio Python.

Egli utilizza un socket TCP per mettersi in ascolto e captare le richieste di connessione in arrivo dai client su una porta specifica (nel nostro esempio è utilizzata una porta non well-known, nello specifico la numero 53000).

Durante l'instaurazione della connessione viene inoltre tenuta traccia di eventuali eccezioni generate da un fallimento della stessa e all'utente viene debitamente notificato, in tempo reale, il tipo di errore.

Quando un client si connette, il server genera un thread indipendente a cui viene assegnata quella specifica comunicazione con quello specifico client così da liberare il main thread che sarà quindi in grado di rimettersi in ascolto ed accogliere le richieste di connessione successive.

Parlando di strutture dati, il server tiene traccia dei client connessi aggiornando una lista di oggetti socket ogni qual volta viene instaurata o chiusa una connessione con un client.

All'interno dell'implementazione del server è definita una funzione utilizzata dai suoi thread per gestire la comunicazione broadcast con tutti i client connessi. Questa funzione avrà il compito di scorrere la lista dei client connessi ed inviare uno ad uno il messaggio che uno dei client ha inviato.

Anche in questo caso sono gestiti gli errori di comunicazione in caso il server si trovasse impossibilitato ad inviare il messaggio a qualcuno dei suoi client connessi.

Client

Il client non dispone di un'interfaccia grafica, è bensì implementato come un'applicazione a riga di comando.

Anche in questo caso si utilizza un socket TCP per connettersi al server comunicando l'indirizzo IP, per la localizzazione della scheda di rete, e la porta sulla quale il server dovrebbe essere in ascolto.

Una volta stabilita la connessione, il client potrà avviare un thread (esentando così il main thread dei vari client dal dover effettuare operazioni secondarie che riguardano solo la comunicazione in entrata) che si occuperà della gestione della ricezione dei messaggi altrui.

Anche in questo caso viene definita una funzione specifica per la ricezione dei messaggi ed anche qui vengono sistematicamente controllate le andate a buon fine della comunicazione. Nel caso in cui dovessero esserci problemi nella ricezione dei messaggi, opportuni costrutti di generazione e cattura delle eccezioni provvederanno a notificare all'utente la caduta in errore.

L'invio dei messaggi è invece gestito dal main thread dei client. L'utente può scrivere il suo messaggio in console ed inviarlo, il client si assicurerà che la codifica sia in UTF-8 (convenzione stabilita preventivamente tra client e server per permettere la certezza di leggibilità dei messaggi) ed invierà il messaggio al server tramite l'oggetto socket.

Anche qui sono implementati costrutti di cattura e gestione di eventuali eccezioni generate da errori nell'invio del messaggio, in tali casi viene notificato il problema all'utilizzatore.

Requisiti ed utilizzo del software

Requisiti software:

- Accertarsi di avere Python installato (per sicurezza, alla versione più recente) sulla propria macchina. È possibile scaricare il setup dal sito ufficiale di Python.
- Sebbene siano librerie standard di Python, si specifica per sicurezza che il software utilizza le librerie **socket**, **threading** e **sys** rispettivamente per la comunicazione/connessione, per il la gestione del multithreading e per la gestione dell'applicativo.

Requisiti di esecuzione:

- Due o più terminali o macchine:
  Se si vuole eseguire tutto sulla stessa macchina (i test vengono eseguiti così e così funziona l'applicazione di default), è necessario aprire più terminali per eseguire il server ed ogni client separatamente.
  Se invece ogni host della chatroom è ospitato su diverse macchine (fisiche o virtuali) sarà necessario che tutte siano collegate sulla stessa rete (il collegamento ad internet non è comunque richiesto) e bisognerà utilizzare il corretto indirizzo IP del server per permettere ai client di raggiungerlo (per farlo è però necessario modificare il contenuto della variabile all'interno dei file server.py e client.py).

N.B. Assicurarsi che la porta utilizzata dall'applicativo e su cui il server dovrebbe essere in ascolto non sia bloccata da firewall o altre restrizioni di rete definite dal sistema.

Utilizzo ed esecuzione del codice

Avvio del server

Assicurarsi di eseguire il server prima di avviare qualsiasi client. È assolutamente necessario che sia il server a mettersi in ascolto sulla porta data prima che qualcuno cerchi di instaurare una connessione.

Avviare un terminale e spostarsi quindi nella locazione del file system nel quale si trova il codice sorgente del server ed infine eseguirlo come di seguito:

```
python server.py
```

Avvio dei client

Avviare un terminale (diverso per ogni client se ci si trova nella stessa macchina che ospita il server o altri client), spostarsi nella locazione del file system nel quale si trova il codice sorgente dei client ed infine eseguirlo come di seguito:

```
python client.py
```

Se tutto è stato eseguito a dovere allora sarà possibile utilizzare il software per comunicare nella chatroom ospitata dal server.