

Sistema di Chat Client-Server

Luca Marchi

1 Introduzione

La chat è implementata con il modello client-server utilizzando socket programming. Il server consente ai vari utenti (client) di inviare e ricevere messaggi in un'ambiente condiviso (simile ad un gruppo WhatsApp o Telegram), gestendo i vari host contemporaneamente. Il client permette di connettersi al server specificando un indirizzo ip, inoltre l'utente può riconoscersi e identificarsi all'interno della chat tramite uno username univoco. Gli utenti possono interfacciarsi al software tramite una GUI (graphical user interface), la quale permette di mandare messaggi e visualizzare la charoom condivisa. Questo progetto è stato implementato in python, il codice del server si trova nel file *server.py* mentre quello del client si trova in *client.py*.

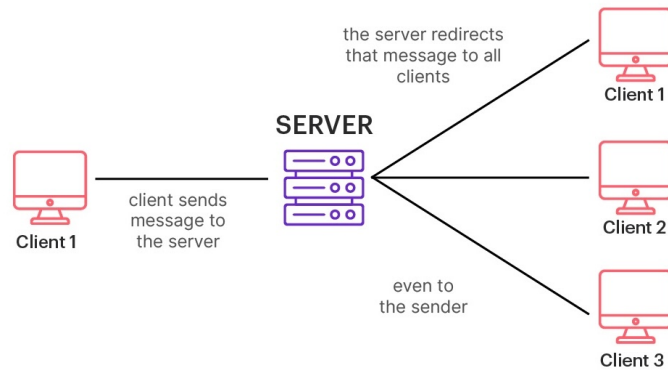


Figure 1: Modello chat client-server

2 Server

2.1 Funzionamento

Il server permette ai client di connettersi attraverso una socket TCP della famiglia *AF_INET* (per la comunicazione di host remoti tramite internet)

usufruendo di un indirizzo ip e di una porta. Una volta avvenuto il collegamento con l'utente, viene creata una nuova socket che rappresenta la connessione tra i due terminali. Il server utilizza un thread per svolgere questo compito

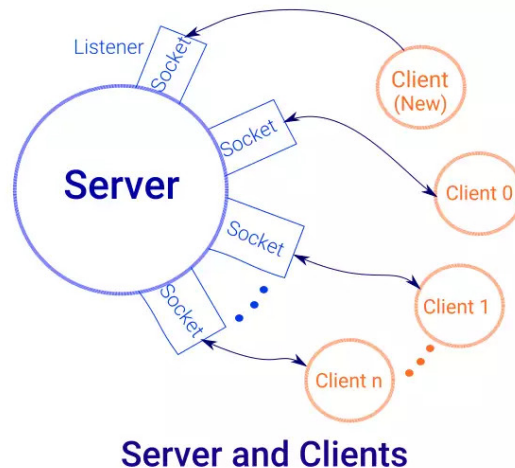


Figure 2: Socket

accept_thread, il quale genera un altro thread figlio per la gestione del relativo client: *manage_client_thread*. Quest'ultimo si occupa della registrazione dell'utente e della ricezione di messaggi per poi inoltrarli a tutti i client (i loro indirizzi vengono salvati in un dizionario che ha come chiave la socket di connessione e come valore il nome utente). I vari utenti non possono avere lo stesso username, quindi viene chiesto all'utente di immettere un nome finché non sarà ritenuto valido. Alla ricezione del messaggio *quit* il server disconnette quel client, chiudendo la socket e rendendo disponibile nuovamente il nome utente dell'host che ha appena terminato la connessione.

2.2 Chiusura

Il server rimane sempre in ascolto sulla socket iniziale che accetta le connessioni, l'unico modo per spegnerlo è premere dal terminale da cui si è avviato il programma *Ctrl - c*. Questo segnale verrà riconosciuto dal main thread che attraverso il metodo *quit_server()* provvederà alla sua chiusura. Il server chiudendosi, invia un messaggio di shutdown a tutti i client collegati.

3 Client

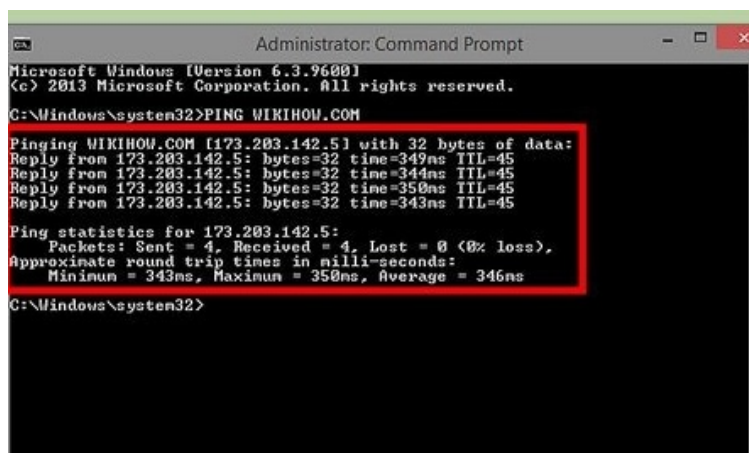
3.1 Funzionamento

Appena il programma si avvia viene chiesto di inserire da terminale l'indirizzo ip del server. Successivamente viene creata una socket TCP analoga a quella del

server, se la connessione non riesce viene lanciata un'eccezione. Il main thread crea la gui per interfacciarsi con l'utente che si compone di:

1. area per visualizzare i messaggi
2. box per scrivere messaggi
3. bottone per inviare i messaggi

Inoltre nel thread principale viene creato anche il processo *receive_thread* che si occupa di ricevere i messaggi e di chiudere il client alla ricezione di *SHUTDOWN*. Il metodo *send()* che invece manda i messaggi al server viene collegato sia al tasto *Return* della tastiera sia al pulsante della gui. Per la gestione della connessione il client manda un ping all'indirizzo del server ogni 5 secondi. Se il ping non va a buon fine allora viene chiuso il client.



```
Administrator: Command Prompt
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.
C:\Windows\system32>PING WIKIHOW.COM

Pinging WIKIHOW.COM [173.203.142.5] with 32 bytes of data:
Reply from 173.203.142.5: bytes=32 time=349ms TTL=45
Reply from 173.203.142.5: bytes=32 time=344ms TTL=45
Reply from 173.203.142.5: bytes=32 time=350ms TTL=45
Reply from 173.203.142.5: bytes=32 time=343ms TTL=45

Ping statistics for 173.203.142.5:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 343ms, Maximum = 350ms, Average = 346ms
C:\Windows\system32>
```

Figure 3: Esempio ping

3.2 Chiusura

Per la chiusura del client esistono varie possibilità:

1. In qualsiasi momento dall'avvio del software premento *Ctrl-c* si chiuderà il client. Se viene premuto nella fase iniziale esso sarà gestito tramite *try-except*, successivamente invece viene mandato un segnale dal main thread alla funzione *signal_handler()* che provvederà alla chiusura (come citato in precedenza per il server).
2. Se viene inserito il messaggio *quit* il client si disconnette.
3. Viene creato un daemon thread *ping_thread* che si occupa di eseguire il ping sull'ip del server e di terminare l'esecuzione del client in caso di riscontro negativo.
4. Si può chiudere il client tramite la croce rossa collocata in alto.

4 Come eseguire il programma

1. Avviare lo script *server.py* da terminale tramite il comando *python3* o *python* (dipende dalla configurazione) seguito da *server.py*.
2. Da un nuovo terminale avviare *client.py* tramite *python3 client.py* e immettere l'indirizzo ip del server. Non c'è bisogno di inserire anche la porta, di default è impostata la 8080. Se si prova il software utilizzando come indirizzo del server *localhost* la funzione *ping_ip* citata in precedenza restituirà sempre un riscontro positivo. Per testare questa funzione cambiare l'ip con un altro desiderato.
3. Ripetere il punto 2 per ogni client che si vuole avviare.
4. Per terminare la connessione eseguire i metodi sopra elencati.