

Hybrid Chat

Christian Battista Giannarelli #tecnologie

Che cos'è?

Un applicativo in Python3 che permette di comunicare nella rete locale attraverso lo scambio di messaggi di testo.

Perché "hybrid"?

L'applicativo può essere utilizzato in due modalità:

- Server-client hybrid:
 - invio di messaggi ai client in modo diretto;
 - ricezione dei messaggi dai client;
 - inoltra a tutti i client dei messaggi ricevuti;
- client-only:
 - invio dei messaggi al server;
 - ricezione dei messaggi dal server.

Moduli utilizzati

Sono impiegati i seguenti moduli:

- `socket` :
 - per le funzionalità di rete;
- classe `Thread` da `threading` :
 - per l'utilizzo dei thread;
- `errno` :
 - per i codici d'errore.

Analisi del codice

1. Inclusione dei moduli:

```
import socket
from threading import Thread
import errno
```

2. Creazione e/o apertura del file di log:

```
file = open("connections.txt", "a")
```

3. Creazione della lista di connessioni:

```
connections = []
```

4. SOLO SERVER - Thread di gestione di una singola connessione (salvataggio log, ricezione messaggi, passaggio alla funzione d'inoltro):

```
def new_connection(connection, address):  
    with connection:  
        print("New connection: ", address)  
        file.write(str(address))  
        file.flush()  
  
        while True:  
            data = connection.recv(1024)  
  
            # Dati corrotti, thread ucciso  
            if not data:  
                break  
  
            print(address, data.decode())  
            send_to_clients(data)
```

5. SOLO SERVER - La classe `Connection`, che prevede l'avvio del thread mostrato al passaggio precedente (viene generato un oggetto di questo tipo per ogni connessione):

```
class Connection:  
    def __init__(self, connection, address):  
        self.connection = connection  
        self.address = address  
        self.thread = Thread(target = new_connection, args = (self.connection,  
self.address,))  
        self.thread.start()
```

6. SOLO CLIENT - Thread per la ricezione e stampa dei dati:

```
def receive_as_client():  
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as my_socket:  
        my_socket.connect((server_address, int(port)))  
  
    while True:  
        data= my_socket.recv(1024)  
        print(data.decode())
```

7. Invio semplice di un messaggio (server-client hybrid → client-only oppure client-only → server-client hybrid):

```
def send_as_client():  
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as my_socket:  
        my_socket.connect((server_address, int(port)))  
  
    while True:  
        text = input()  
        my_socket.sendall(bytes(username + " : " + text, encoding =  
"utf8"))
```

8. SOLO SERVER - Inoltro a tutti i client di un messaggio ricevuto:

```
def send_to_clients(data):  
    for connected in connections:  
        connected.connection.sendall(data)
```

9. Guida d'utilizzo:

```
print("Welcome!\na) Type \"localhost\" and your chosen port to operate as  
server-client hybrid.\nb) Type the server IP address and the port it uses to  
operate as client-only.")
```

10. Acquisizione dell'indirizzo IP del server:

```
server_address = input("Server IP address: ")
```

11. Acquisizione del numero di porta:

```
port = input("Port: ")
```

12. Acquisizione del nome utente:

```
username = input("Username: ")
```

13. Avvio del thread per l'invio semplice dei messaggi:

```
send_as_client_thread = Thread(target = send_as_client)
send_as_client_thread.start()
```

14. Operatività effettiva, selezione della modalità in base alla disponibilità della porta selezionata:

```
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as my_socket:
    # Operatività server-client hybrid (porta inutilizzata)
    try:
        my_socket.bind((server_address, int(port)))
        my_socket.listen()

        # Nuovo username per il server
        username = username + " SERVER"

        print("Server started!\nYour new username is: " + username + "!\nType
some text and press enter to send as " + username + ".")

        # Accettazione di nuove connessioni, creazione di oggetti Connection
```

```

while True:
    connection, address = my_socket.accept()
    connected = Connection(connection, address)
    connections.append(connected)

# Operatività client-only (porta in utilizzo)
except socket.error as e:
    if e.errno == errno.EADDRINUSE:
        print("Port is already in use. Client-only mode starting...\nType
some text and press enter to send ('q' to quit).")

        receive_as_client_thread = Thread(target = receive_as_client)
        receive_as_client_thread.start()

        while True:
            pass
    else:
        print(e)

```

N.B.: la modalità client-only utilizza due socket differenti: uno per l'invio ed uno per la ricezione.

Galleria

```
christian@192 desktop % python3 chat.py
Welcome!
a) Type "localhost" and your chosen port to operate as server-client hybrid.
b) Type the server IP address and the port it uses to operate as client-only.
Server IP address: localhost
Port: 8000
Username: cibigi
Server started!
Your new username is: cibigi SERVER!
Type some text and press enter to send as cibigi SERVER.
New connection: ('127.0.0.1', 50445)
New connection: ('127.0.0.1', 50446)
New connection: ('127.0.0.1', 50447)
New connection: ('127.0.0.1', 50448)
New connection: ('127.0.0.1', 50449)
ciao sono il server
('127.0.0.1', 50445) cibigi SERVER : ciao sono il server
('127.0.0.1', 50446) taylor-swift : ciao sono taylor
('127.0.0.1', 50448) lorde : ciao sono lorde

```

```
-/desktop -- Python chat.py
christian@192 desktop % python3 chat.py
Welcome!
a) Type "localhost" and your chosen port to operate as server-client hybrid.
b) Type the server IP address and the port it uses to operate as client-only.
Server IP address: localhost
Port: 8000
Username: taylor-swift
Port is already in use. Client-only mode starting...
Type some text and press enter to send ('q' to quit).
cibigi SERVER : ciao sono il server
ciao sono taylor
taylor-swift : ciao sono taylor
lorde : ciao sono lorde

```

A destra il server cibigi SERVER, a sinistra il client taylor-swift.

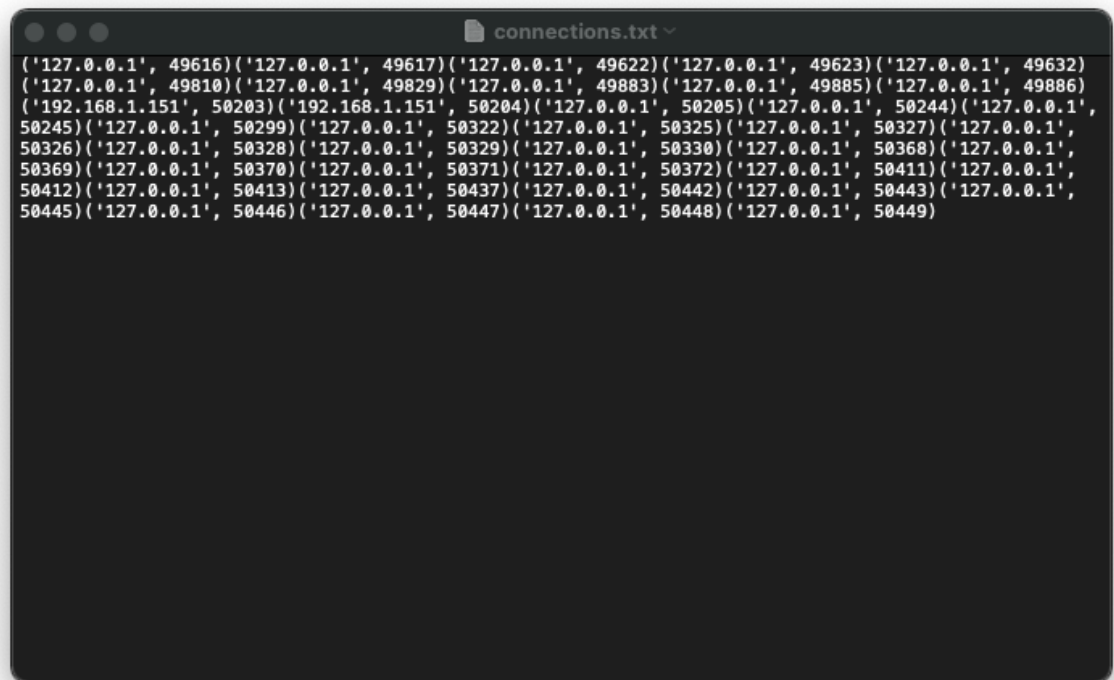
```
christian@192 desktop % python3 chat.py
Welcome!
a) Type "localhost" and your chosen port to operate as server-client hybrid.
b) Type the server IP address and the port it uses to operate as client-only.
Server IP address: localhost
Port: 8000
Username: cibigi
Server started!
Your new username is: cibigi SERVER!
Type some text and press enter to send as cibigi SERVER.
New connection: ('127.0.0.1', 50445)
New connection: ('127.0.0.1', 50446)
New connection: ('127.0.0.1', 50447)
New connection: ('127.0.0.1', 50448)
New connection: ('127.0.0.1', 50449)
ciao sono il server
('127.0.0.1', 50445) cibigi SERVER : ciao sono il server
('127.0.0.1', 50446) taylor-swift : ciao sono taylor
('127.0.0.1', 50448) lorde : ciao sono lorde

```

```
-/desktop -- Python chat.py
christian@192 desktop % python3 chat.py
Welcome!
a) Type "localhost" and your chosen port to operate as server-client hybrid.
b) Type the server IP address and the port it uses to operate as client-only.
Server IP address: localhost
Port: 8000
Username: lorde
Port is already in use. Client-only mode starting...
Type some text and press enter to send ('q' to quit).
cibigi SERVER : ciao sono il server
taylor-swift : ciao sono taylor
ciao sono lorde
lorde : ciao sono lorde

```

A destra il server cibigi SERVER, a sinistra il client lorde.



```
connections.txt
('127.0.0.1', 49616)('127.0.0.1', 49617)('127.0.0.1', 49622)('127.0.0.1', 49623)('127.0.0.1', 49632)
('127.0.0.1', 49810)('127.0.0.1', 49829)('127.0.0.1', 49883)('127.0.0.1', 49885)('127.0.0.1', 49886)
('192.168.1.151', 50203)('192.168.1.151', 50204)('127.0.0.1', 50205)('127.0.0.1', 50244)('127.0.0.1',
50245)('127.0.0.1', 50299)('127.0.0.1', 50322)('127.0.0.1', 50325)('127.0.0.1', 50327)('127.0.0.1',
50326)('127.0.0.1', 50328)('127.0.0.1', 50329)('127.0.0.1', 50330)('127.0.0.1', 50368)('127.0.0.1',
50369)('127.0.0.1', 50370)('127.0.0.1', 50371)('127.0.0.1', 50372)('127.0.0.1', 50411)('127.0.0.1',
50412)('127.0.0.1', 50413)('127.0.0.1', 50437)('127.0.0.1', 50442)('127.0.0.1', 50443)('127.0.0.1',
50445)('127.0.0.1', 50446)('127.0.0.1', 50447)('127.0.0.1', 50448)('127.0.0.1', 50449)
```

Log delle connessioni.