Socket

Cos'è un socket?

Un socket è un oggetto per la comunicazione tra diversi endpoint (indirizzo IP, porta) che possono risiedere sulla stessa macchina o su macchine diverse collegate in rete.
I socket si trovano al livello di trasporto ed espongono un'interfaccia di programmazione a livello applicativo che permette di sfruttare i meccanismi di comunicazione sottostanti.
Comunemente i socket sfruttano i layer TCP e UDP per inviare e ricevere dati.

TCP (Transmission Control Protocol)

Protocollo del livello di trasporto che gestisce le connessioni, rileva gli errori e controlla il flusso di dati in rete tra mittente e destinatario.

UDP (User Datagram Protocol)

Protocollo del livello di trasporto che a differenza del TCP è senza connessione, più semplice, veloce, ma inaffidabile perchè non esegue controlli di consegna e di conseguenza i pacchetti possono essere persi, duplicati, arrivare in ordine sparso.

Sistema client/server

Una macchina (server) mette a disposizione uno o più servizi in rete per altre macchine (client) che ne usufruiscono.
La richiesta è inviata dal client e viene processata dal server.

Libreria socket Python principali metodi;

- **socket()**: istanzia il socket, ha 2 parametri obbligatori
  - **family**: tipo indirizzo (IPV4 = AF_INET, IPV6 = AF_INET6)
  - **type**: tipo di socket (TCP = SOCK_STREAM, UDP = SOCK_DGRAM)
- **bind()**: associa il socket all' endpoint (indirizzo IP, porta).
- **listen()**: reasta in attesa di connessioni.
- **accept()**: accetta la connessione di un client e crea un nuovo socket per la comunicazione con quel client.
- **connect()**: stabiliste la connessione con un server (endpoint).
- **send()**: invia I dati tramite il socket TCP.
- **sendTo()**: invia I dati tramite il socket UDP.
- **recv()**: Riceve i dati dal socket TCP.
- **recvFrom()**: Riceve i dati dal socket UDP.
- **close()**: chiude la connessione.

Realizzazione di un semplice echo server

Un server riceve dei dati da un client e li reinvia.

Il server deve:

- Restare in ascolto su una determinata porta (endpoint)
- Accettare la connessione
- Ricevere i dati
- Reinviare i dati
- Terminare

Il client deve:

- Collegarsi al server (endpoint)
- Inviare i dati
- Ricevere la risposta
- Terminare

Server UDP

```
import socket

ip = '127.0.0.1'
porta = 5006
endpoint = (ip,porta)
buf_size = 1024
try:
    # creo socket UDP
    skt = socket.socket (socket.AF_INET, socket.SOCK_DGRAM)
    skt.bind(endpoint)
    print("Server in ascolto...")
    data_bytes, addr = skt.recvfrom(buf_size)
    print("Connessione da: ", addr)
    data_str = data_bytes.decode("utf-8") #converto byte in stringa
    print ("Ricevuto dal client: ", data_str)
    skt.sendto(data_bytes,addr)
    skt.close()
except socket.error as e:
    print("Si è verificato un errore: ", e)
```

Client UDP

```
import socket

ip = '127.0.0.1'
porta = 5006
endpoint = (ip,porta)
buf_size = 1024
msg = bytes("Hello World","utf-8") #converte stringa in byte
try:
    # creo socket UDP
    skt = socket.socket (socket.AF_INET, socket.SOCK_DGRAM)
    skt.connect(endpoint)
    skt.sendto(msg,endpoint)
    #attendo risposta dal server
    data_bytes, addr = skt.recvfrom(buf_size)
    data_str = data_bytes.decode("utf-8") #converto byte in stringa
    print ("Risposta: ", data_str)
    skt.close
except socket.error as e:
    print("Si è verificato un errore: ", e)
```

Server TCP

```
import socket

ip = '127.0.0.1'
porta = 5006
endpoint = (ip,porta)
buf_size = 1024
try:
    # creo socket TCP
    skt = socket.socket (socket.AF_INET, socket.SOCK_STREAM)
    skt.bind(endpoint)
    skt.listen()
    print("Server in ascolto...")
    con, addr = skt.accept()
    print("Connessione da: ", addr)
    data_bytes = con.recv(buf_size)
    data_str = data_bytes.decode("utf-8") #converto byte in stringa
    print ("Ricevuto dal client: ", data_str)
    con.send(data_bytes)
    con.close()
except socket.error as e:
    print("Si è verificato un errore: ", e)
```

Client TCP

```
import socket

ip = '127.0.0.1'
porta = 5006
endpoint = (ip,porta)
buf_size = 1024
msg = bytes("Hello World","utf-8") #converte stringa in byte
try:
    # creo socket TCP
    skt = socket.socket (socket.AF_INET, socket.SOCK_STREAM)
    skt.connect(endpoint)
    skt.send(msg)
    #attendo risposta dal server
    data_bytes = skt.recv(buf_size)
    data_str = data_bytes.decode("utf-8") #converto byte in stringa
    print ("Risposta: ", data_str)
    skt.close
except socket.error as e:
    print("Si è verificato un errore: ", e)
```