Relazione Progetto di Reti

Programmazione di Reti - Anno Accademico 2020/2021

Architettura client-server UDP

Valentina Pieri

valentina.pieri5@studio.unibo.it

0000974789

Indice

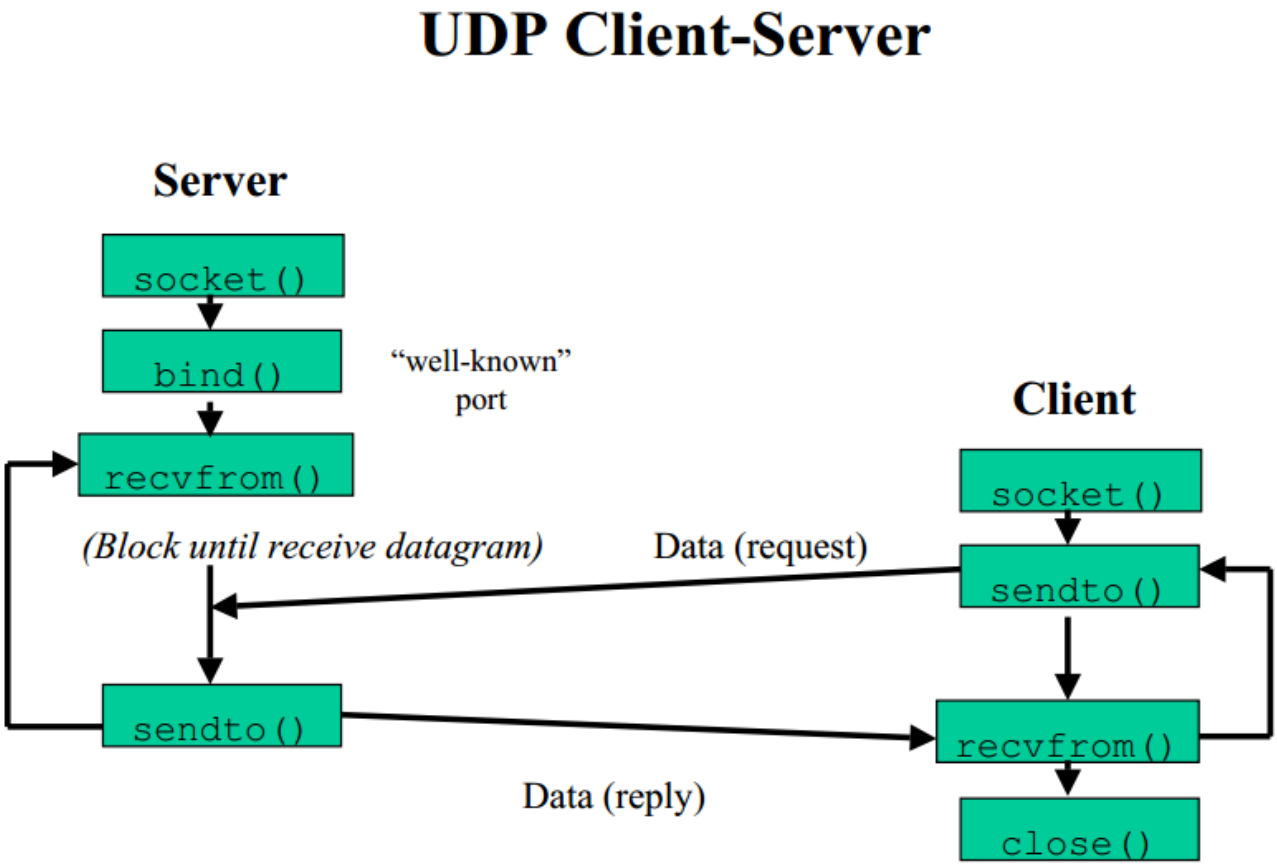
Dettagli per l'uso

Il programma è strutturato per essere usato dal terminale, dal quale bisogna accedere alle directory server o client e poi eseguire, rispettivamente all'interno di quale cartella ci si trova:

- py server.py
- py client.py

Per quanto riguarda i file questi sono da inserire all'interno della directory desiderata, client o server.
Quando si desidera terminare il programma, basta digitare il comando "exit" sul client, il quale chiuderà sia il client che il server. In caso ci fossero più client in uso, il primo che verrà chiuso chiuderà anche il server, lasciando funzionanti i restanti client, i quali dovranno essere chiusi a loro volta.

Descrizione generale



L'obbiettivo del codice è quello di permettere il trasferimento di file tra un client-server, che impieghi il servizio di rete senza connessione. Il codice è suddiviso in due directory, client e server, al cui interno sono presenti rispettivamente le classi client.py, client_lib.py

e server.py, server_lib.py.
All'interno della classe server.py è presente la funzione command_thread, al cui interno controllo che tipo di comando gli è stato passato dal client, questa funzione viene poi ripresa in un while true in cui, per ogni comando che i client inviano al server, quest'ultimo crea un thread apposito che ne gestisce la relativa risposta.
Quando viene digitato dal client il comando "exit", all'interno della classe client.py e server.py, vengono chiusi i loro socket, chiamati sock, e poi si chiudono.

Descrizione delle funzioni principali

- **Get**

  In server_lib.py è presente la funzione get_server, che dopo aver ricevuto il comando "get filename" dal client inizia la ricerca del file. Se quest'ultimo è presente all'interno della cartella server, viene inviato al client "ok", per confermarne la sua presenza, per poi aprire e leggere il file fino a quando non si è giunti alla fine di quest'ultimo. A quel punto il file viene inviato al client e compare il messaggio "Successfully sent file from Server". Ma se in caso contrario il file non dovesse essere presente allora viene inviato solamente un messaggio di "error".
  In client_lib.py è presente invece la funzione get_client, la quale manda il comando iniziale "get filename" al server. Se il client riceve in risposta "ok", allora all'interno della directory client viene creato, e aperto in modalità scrittura, il file ricevuto dal server e compare il messaggio "Successfully received file from Server". Se invece il client riceve come risposta "error", allora viene stampato "File not Found".

- **Put**

  In client_lib.py è presente put_client, usando split viene salvata in una variabile chiamata filename, il nome del file inserito dal client e controlla se è presente all'interno della directory client. Se il file esiste invia il comando "get filename" al server, apre il file in lettura binaria e grazie a un while true invia i dati del file al server fino a quando non arriva alla fine del file, il quale viene chiuso e compare il messaggio "Successfully sent file from Client". Se invece il file non è presente allora comparirà il messaggio "File not Found".
  In server_lib.py è presente put_server, in cui come in put_client, viene salvata in una variabile, chiamata filename, il nome del file, utilizzando split. Il file viene poi successivamente creato con scrittura binaria e viene scritto con un while true fino a quando non si arriva alla fine del file e viene chiuso. Alla fine appare il messaggio "Successfully received file from Client".

- **List**

In client_lib.py è presente list_client, nel quale è inviato il comando "list" al server e dopo aver salvato il numero di file ricevuti dal server, con un loop stampo i nomi di tutti i file appartenenti alla lista e quando è stato completato questo procedimento compare il messaggio "Successfully received list from Server".
Invece in server_lib.py è presente list_server e quando riceve il comando dal client, va subito a creare una lista dei file presenti all'interno della directory server, escludendo server.py, server_lib.py e pycache. Viene inviato il numero di file presenti nella lista al client, con un loop for ognuno dei file è inviato ed infine compare il messaggio "Successfully sent list from Server".