Relazione per progetto di programmazione di reti

Architettura client-server UDP per trasferimento file

Francesca Lanzi - 0000970828
francesca.lanzi6@studio.unibo.it

1 Luglio 2022

Indice

Capitolo 1

Funzionalità del progetto

Lo scopo dell'applicazione è di poter trasferire file da un client a un server e viceversa e di ottenere una lista completa dei file presenti in quest'ultimo. In modo che risultasse più logica la struttura del programma, è stata presa la decisione di dividere i file relativi al server e al client in due cartelle separate, ognuna contenente una copia del file `messages.py`, in cui sono contenute le funzioni che sono necessarie a entrambi i file, cioè ottenere la lista dal server, ricevere e mandare un file. Non sono state utilizzate strutture dati particolari, sono state usate principalmente le funzionalità delle stringhe, dei file e dei bytes, oltre a quelle necessarie dei socket. Per la lettura e la scrittura dei file da inviare e ricevere sono stati usati buffer della stessa dimensione, cioè da 4096 bytes.

1.1 Funzionalità in dettaglio

1. Comando `list`

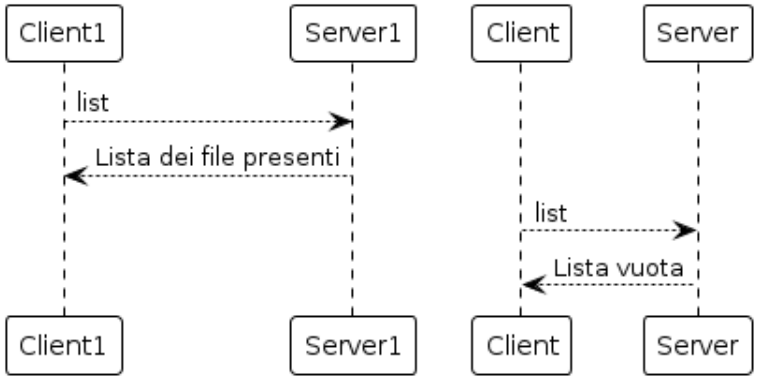

Figura 1.1: Rapprentazione grafica del funzionamento di `list`

Il comando `list` consiste nella richiesta al server da parte del client di una lista completa dei file presenti nella cartella primo; a questo scopo, per evitare di leggere possibili cartelle, è stato necessario filtrare e selezionare i file soli. Quindi, come si vede nella Figura 1.1, nel caso in cui saranno presenti dei file il server risponderà al client mandando una lista di stringhe con i nomi dei file esistenti; altrimenti se i file sono assenti, il server restituirà semplicemente una lista vuota. Infine viene scritto a linea di comando se l'invio di pacchetti è andato a buon fine o meno.

2. Comando `get <<filename>>`

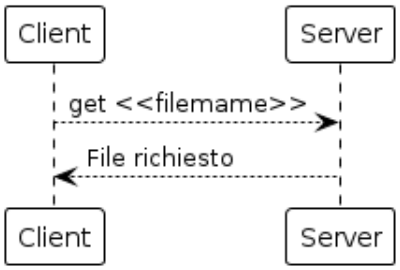

Figura 1.2: Rapprentazione grafica del funzionamento di `get`

Il comando `get` permette al client di richiedere al server un file presente in esso. Il funzionamento è spiegato nella Figura 1.2: la parte del client manda la richiesta con il nome del file che si vuole ottenere; quindi il server, ricevendo il nome del file, usa il metodo `sendFile` per spedire il file sottoforma di bytes al client; ottenuto il file desiderato con `receiveFile`, il client decodifica il binario ricevuto e stampa un messaggio di operazione riuscita.

3. Comando `put <<filename>>`

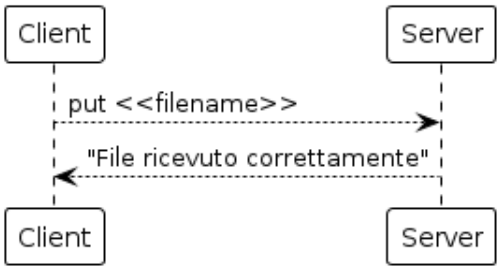

Figura 1.3: Rapprentazione grafica del funzionamento di `put`

Con il comando `put` il client ha la possibilità di mandare un file nel server. Il principio di funzionamento è simile al precedente comando ma con i ruoli scambiati (vedi Figura 1.3): ora è il client che spedisce il binario del file in questione al server usando il metodo `sendFile`, mentre

il server, usando `receiveFile`, riceve e decodifica il suddetto binario e manda al client un messaggio di operazione svolta correttamente.

4. Comando `exit`

Il comando exit semplicemente fa terminare il programma; vengono inoltre chiusi i socket del client e del server.

Capitolo 2

Uso dell'applicativo

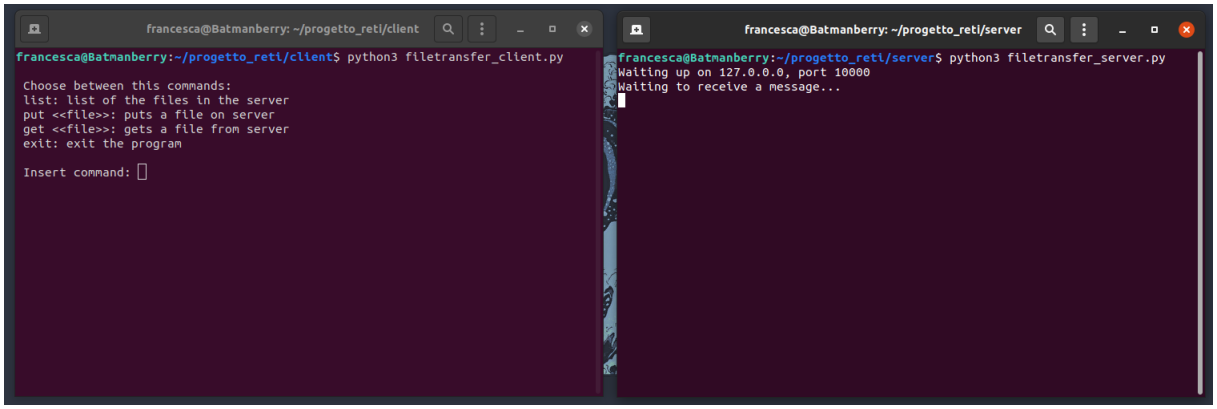

Figura 2.1: Terminali del client e del server

Il programma è progettato il modo da essere eseguito da riga di comando; è necessario aprire due terminali e avviare in uno il client e nell'altro il server, entrando nelle cartelle ed eseguendo il corrispondente file Python come mostrato nella figura soprastante. Quindi si potrà interagire con il programma scrivendo nella linea di comando del terminale client l'operazione desiderata tra quelle elencate in precedenza. Se si vuole terminare il programma è sufficiente scrivere il comando `exit`.