Relazione Progetto

“Traccia 2 - Architettura client-server UDP per trasferimento file”

Gianluca Consoli

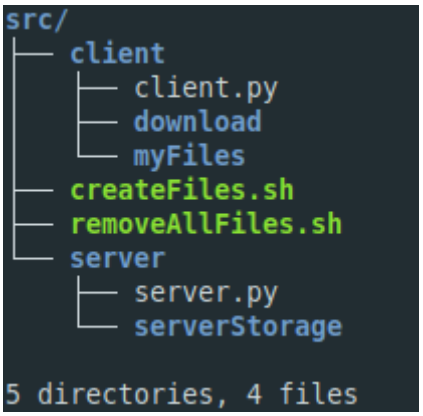
08 luglio 2022

Capitolo 1

Struttura - Progetto

Il progetto realizzato è stato creato per poter avere un trasferimento dei file tra client e server. Come scelta progettuale ho deciso di suddividere la struttura in due semplici file python di nome : client.py e server.py, che sono contenuti nelle directory client/server che si trovano a loro volta dentro una directory chiamata “src”.
I File che verranno scaricati o caricati dal server/client, sono generati automaticamente da uno script di nome ./createFiles.sh (Attenzione!!! Non fare partire questo script direttamente da linea di comando, perché sono progettati per essere chiamati ed eseguiti dal file client.py), per la cancellazione di tutti i file generati è stato creato un’altro script di nome ./removeAllFiles.sh (Attenzione!!! Stessa cosa di prima, non farlo partire direttamente da linea di comando).

STRUTTURA COMPLETA DEL PROGETTO:



Per il client sono state create due cartelle:

- “download”: è la cartella che conterrà tutti i files scaricati dal server;
- “myFiles”: è la cartella personale del client e conterra tutti i suoi file personali.

Mentre per il server è stata creata solo una cartella:

- “serverStorage”: è una cartella con dentro tutti il file del server, che potranno essere scaricati e letti dal client.

Livello di Protocollo

Il livello di protocollo applicativo utilizzato è stato l'HTTP che però usa il servizio UDP, il server accetta la connessione richiesta dal client; mentre, il client inizializza una connessione UDP (crea un socket) al server. Una cosa che non è stata gestita nel progetto è stata la chiusura automatica del socket del client (il timeout), quindi se per esempio il server non è attivo e verrà usato un commando che dovrebbe interagire con il server, il programma client.py si bloccherà per un tempo indefinito in quanto aspetterà una risposta da parte del server.

Strutture dati - Progetto

Per la creazione dei socket (client/server) sono state usate le tuple, per memorizzare i files appartenenti o al client o al server sono state usate o delle stringhe o delle liste.
Mentre, per conoscere tutte le proprietà dei file è stato usato il modulo os, più l'uso di tutti i suoi metodi che sono serviti per la creazione/lettura/scrittura di/su un file.

Funzionamento del Progetto

Prima di tutto si fanno eseguire entrambi i file Python, nel momento in cui viene fatto partire "client.py" verranno generati automaticamente i files del client e del server.
Nella Console del client è possibile impartire un paio di comandi, una volta digitato uno di questi comandi verrà mandato al server e il server di conseguenza eseguirà la richiesta:

['ls', 'upload', 'download', 'exit']

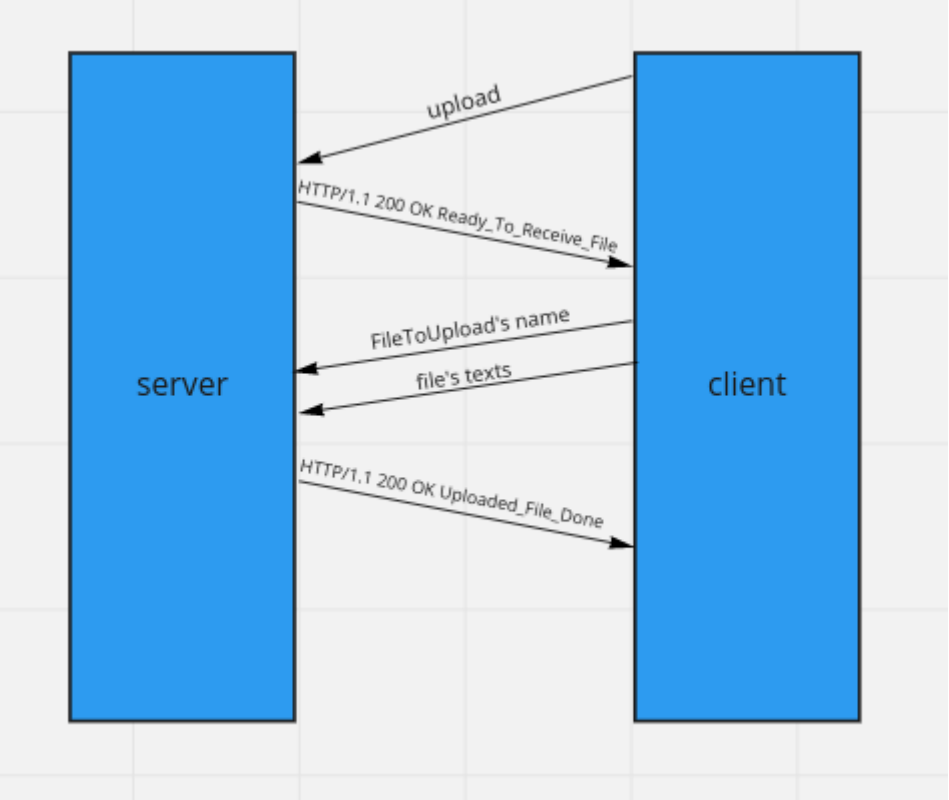
- “ls” sta per listing: permette di far vedere al client quali file sono presenti nel server (attualmente), dopo che è stata mandata la lista di files del server, il client aspetterà da parte del server un 200 OK per avere la conferma che è stata mandata tutta la lista;

- “upload”: permette al client di scegliere uno dei suoi file e caricarlo sul server, usando questo comando nella console del client, apparirà questa richiesta:

```
You have these files : ['clientFile4.txt', 'clientFile1.txt',
'clientFile0.txt', 'clientFile3.txt', 'clientFile2.txt']

Digit a file to upload :
```

  Inoltre a mostrarti tutti i file che possiedi e che puoi caricare, ti chiede di selezionarne uno da mandare al sever, se uno metterà un file non esistente ti richiederà di digitare il nome di un file da caricare e non essendo stato creato un modo per ritornare indietro è consigliabile mettere il nome di un file vero (DIGITARE TUTTE LE MAIUSCOLE E MINUSCOLE IN MODO PRECISO, essendo case sensitive è probabile che per una maiuscola o minuscola il programma ti richiederà più volte di digitare il nome del file in modo corretto). Dopo che è stato mandato sia il comando che il nome del file da caricare, avvengono due 200 OK, in cui nel primo il client si assicura che il server è pronto di riceve il file e nel secondo si assicura che il file è stato caricato con successo.



Tips 1.1: dopo che è stato fatto un upload, digitare il comando “ls” per controllare se effettivamente il file è stato caricato nel server.

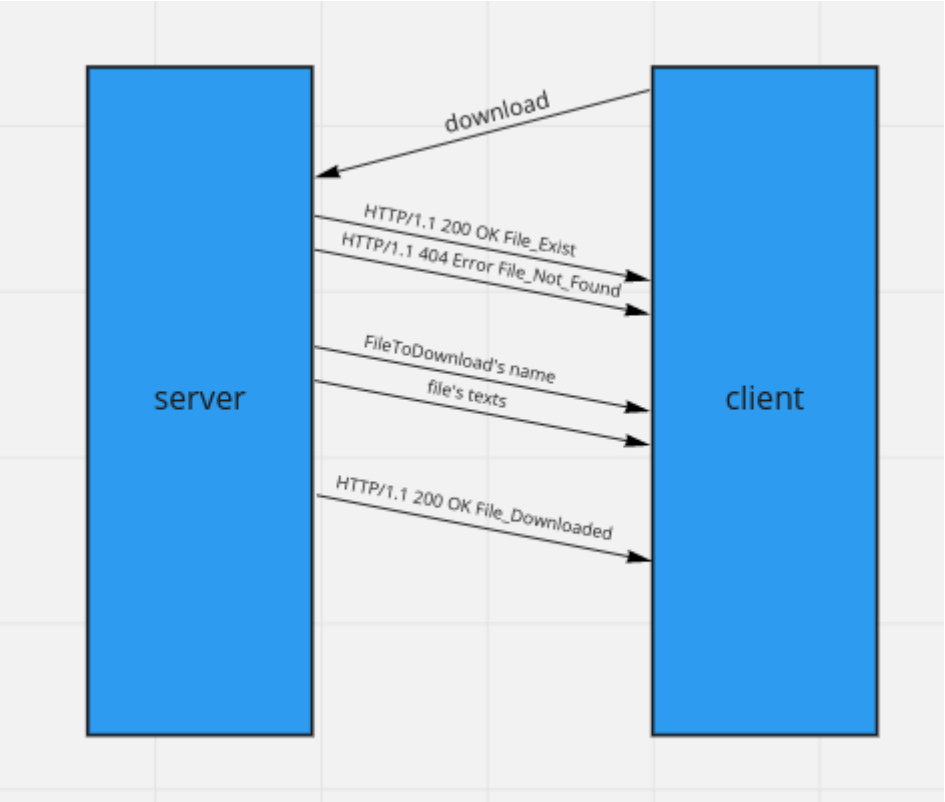
- "download": permette al client di scaricare un file presente nello storage del server, sempre nella console del client, verrà mostrato questo:

```
Choose a file to download : ['serverFile4.txt',
'serverFile2.txt', 'serverFile3.txt', 'serverFile1.txt',
'serverFile0.txt']

Digit the name of file:
```

Ti farà vedere quali sono i file presenti nel server il quel momento (quindi è sempre aggiornato) e poi ti chiederà di digitare il nome di un file che vuoi scaricare, anche in questo caso dopo aver digitato il nome del file, avvengono due 200 OK, il primo servirà al client a capire se il file che vuole scaricare esiste (se non esiste riceverà un 404 File Not Found), e il secondo serve per capire se il file è stato scaricato in maniera corretta.

Tips 1.2: controllare se il file è stato scaricato e si trova nella cartella “download”.



- “exit”: permette al client di terminare il programma in modo sicuro e chiude il socket (Attenzione!!! Ogni volta che uno esce verranno eliminati tutti i files che si trovano nelle cartelle: ‘serverStorage’, ‘myFiles’, ‘download’; quindi se uno vuole controllare la presenza di certi files, assicurarsi di avere ancora in esecuzione il programma client.py)