# Elaborato Programmazione di Reti
# Traccia 2
# Architettura client-server UDP per trasferimento file

Luca Barattini
Mail: luca.barattini4@studio.unibo.it
Matricola: 0000987404

Anno accademico 2021-2022

# Indice

# Capitolo 1

# Scopo del progetto

Lo scopo di questo elaborato è quello di progettare ed implementare in linguaggio Python un'applicazione client-server per il trasferimento di file che impieghi il servizio di rete senza connessione (socket tipo SOCK_DGRAM, ovvero UDP come protocollo di stato trasporto)

## 1.1 Requisiti del software

Il software deve permettere:

- Connessione client-server senza autenticazione;
- La visualizzazione sul client dei file disponibili sul server;
- Il download di un file dal server;
- L'upload di un file sul server.

La comunicazione tra client e server deve avvenire tramite un opportuno protocollo. Il protocollo di comunicazione deve prevedere lo scambio di due tipi di messaggio:

- Messaggi di comando: vengono inviati dal client al server per richiedere l'esecuzione delle diverse operazioni;
- Messaggi di risposta: vengono inviati dal server al client in risposta ad un comando con l'esito dell'operazione.

## 1.2 Funzionalità del server

Il server deve fornire le seguenti funzionalità:

- L'invio del messaggio di risposta al comando list al client richiedente;
- Il messaggio di risposta contiene la file list, ovvero la lista dei nomi dei file disponibili per la condivisione;
- L'invio del messaggio di risposta al comando get contenente il file richiesto, se presente, od un opportuno messaggi odi errore;
- La ricezione di un messaggio put contenente il file da caricare sul server e l'invio di un messaggio di risposta con l'esito dell'operazione.

## 1.3 Funzionalità del client

Il client deve fornire le seguenti funzionalità:

- L'invio del messaggio list per richiedere la lista dei nomi dei file disponibili;
- L'invio del messaggio get per ottenere un file;
- La recezione di un file richiesta tramite il messaggio di get o la gestione dell'eventuale errore;
- L'invio del messaggio put per effettuare l'upload di un file sul server e la ricezione del messaggio di risposta con l'esito dell'operazione.

# Capitolo 2

# Server

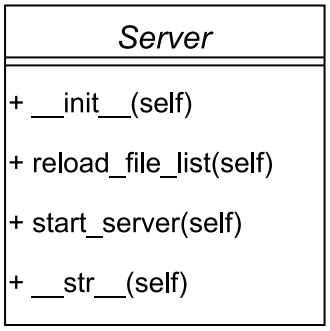


Figura 2.1: Schema UML del server

## 2.1 Gestione comando list

Avvenuta la ricezione del comando list dal client il server chiama una funzione che esegue un ciclo sulla cartella "resources", ovvero la cartella dove sono posizionati i file posseduti dal server, e inserisce ad ogni ciclo il file letto nella lista. Creata la lista il server si occupa dell'invio di quest'ultima al client attraverso una funzione del modulo pickle importabile su python, ovvero la pickle.dumps() che come parametro prende la lista creata in precedenza e la trasforma in un flusso di byte. Il client successivamente ricostruisce la lista mandata con la funzione pickle.load() (sempre del modulo pickle) che come parametro prende il flusso di byte ricevuto.

### Gestione degli errori

Se il server non contiene nessun file nella cartella "resources" provvede, in seguito alla richiesta list, alla spedizione di un messaggio contenente la stringa "EMPTY" in modo che il client capisca che il server non contiene elementi.

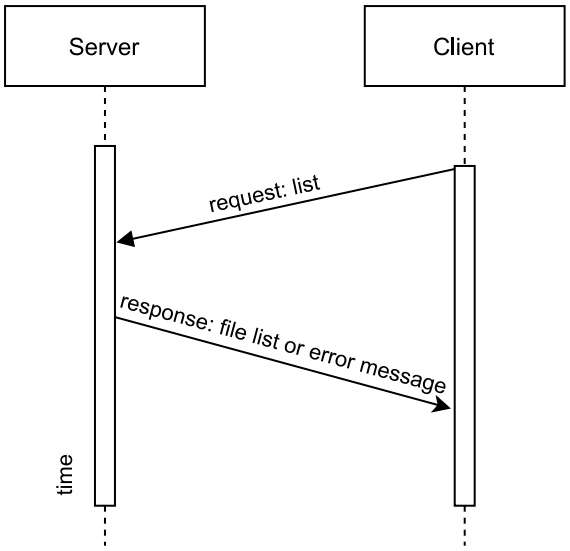


Figura 2.2: Comando list

## 2.2 Gestione comando get

Avvenuta la ricezione del comando get il server controlla se il nome del file è presente all'interno della lista degli elementi in possesso. Se non lo è manda al client un messaggio di errore, in caso contrario ne invia uno contenente la stringa "OK", in modo da accordarsi per l'inizio della ricezione del file. Prima della trasmissione il server invia al client delle informazioni sul file, il nome e la dimensione. A questo punto entra in un ciclo continuo, finché la quantità di byte inviati non raggiunge la dimensione totale del file. In questo ciclo il file viene letto con un buffer da 32768 Byte (32KB) e trasmesso al client che si occupa di ricostruirlo. Alla fine della trasmissione il server invia un messaggio contenente la stringa "FILETRANSMITTED" al client, che compara la dimensione del file ricevuta all'inizio con quella effettivamente scritta, in modo da capire se esso è stato ricevuto per intero.

### Gestione degli errori

Nel caso il file non esista il server trasmette al client un messaggio di errore contenente la stringa "ERROR", in caso di eccezione invia un messaggio con la stringa "EXCEPTION". In queste circostanze il file non viene ricevuto.

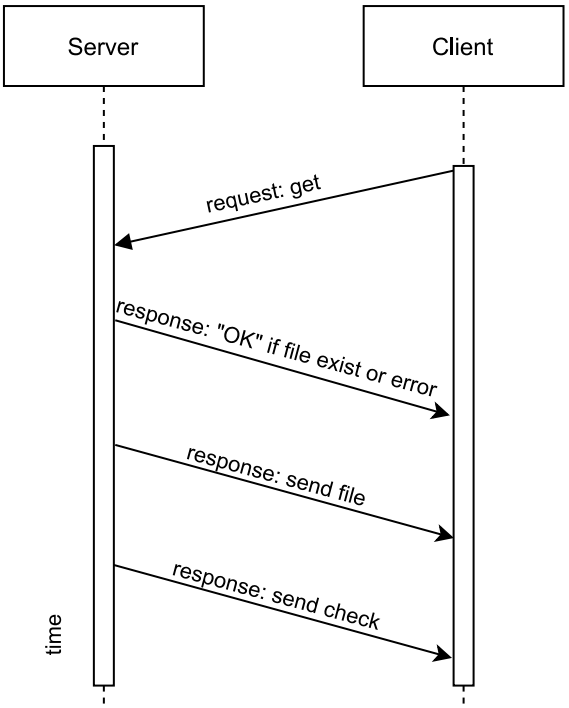


Figura 2.3: Comando get

## 2.3 Gestione comando put

Alla ricezione del comando put il server si mette in ascolto. Il server attende che il client verifichi all'interno della cartella "myFiles" l'esistenza del file che si vuole caricare. Nel caso in cui non esista, il file non viene trasmesso. Ottenuto l'ok riceve le informazioni sul file (nome e dimensione) e successivamente lo stesso. Il buffer è sempre di 32768 Byte che corrispondono a 32 KB. Alla fine della trasmissione il server ottiene dal client una stringa di controllo "FILETRANSMITTED". In seguito alla ricezione del messaggio il server confronta la dimensione del file ricevuta all'inizio con quella effettivamente scritta, e restituisce "FileOK" se è arrivato per intero, altrimenti "FileNotOK".

## Gestione degli errori

Se confrontando le due quantità inviate il server non dovesse riscontrare un'uguaglianza allora procederebbe all'invio di un messaggio contenente la stringa "FileNotOK", in modo che il client capisca che il file non è arrivato correttamente. Il server invia la stringa "FileOK" in caso contrario.

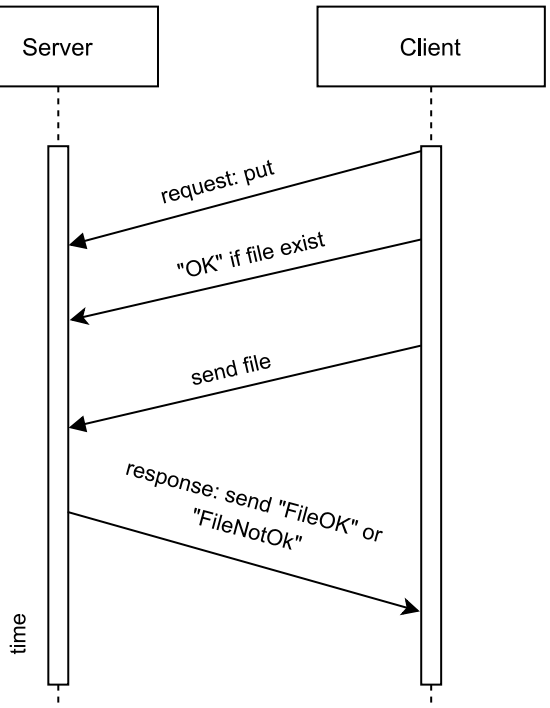


Figura 2.4: Comando put

# Capitolo 3

# Client

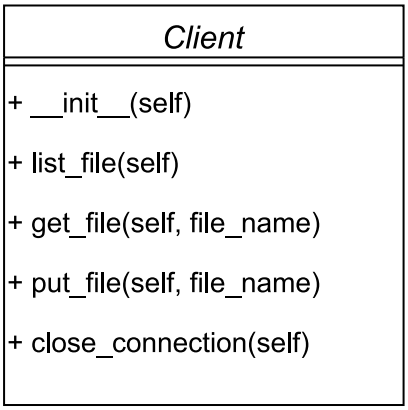


Figura 3.1: Schema UML del client

## 3.1 Gestione dell'input

Per il controllo dell'input del comando list e close, dato che non hanno parametri, si controlla che la stringa inserita nella console del client sia uguale all'istruzione scelta. Per quanto riguarda put e get si interviene controllando che la stringa ricevuta inizi per il comando inserito e che lo split (ad esempio string.split('put ') ) sia diverso da stringa vuota, in modo da poter capire se è stato passato un parametro dopo l'istruzione. Nel caso in cui la stringa non inizi con il comando scelto o in cui il comando e il nome file non contengano spazi tra loro verrà stampato un errore e richiesto un nuovo input. I comandi non sono case sensitive, mentre i nomi dei file lo sono.

## 3.2 Gestione comando list

Il client, ricevuto in input dall'utente il comando list, si occupa di mandare una richiesta al server chiedendo la lista dei file contenuti nella cartella "resources". Successivamente riceve in risposta una lista sotto forma di flusso di byte (grazie alla funzione pickle.dumps() del modulo pickle) che dovrà ricomporre attraverso la funzione pickle.load() che come parametro prende i dati appena ricevuti. Finito il tutto il client si occupa di stampare a schermo la lista ricevuta.

### Gestione degli errori

Se il server non possiede files allora invia un messaggio al client contenente la stringa "EMPTY".

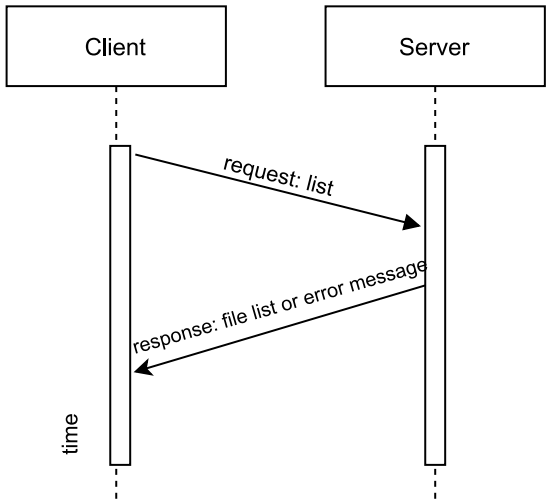


Figura 3.2: Comando list

## 3.3 Gestione comando get

Il client, ricevuto l'input del comando get con il relativo parametro, invia la richiesta al server. Appena ricevuta il server si occupa di controllare se il file passato come parametro è presente all'interno della sua lista. Se è presente allora viene inviata una stringa "OK" ed inizia la trasmissione del file, preceduta dalla trasmissione del nome e della dimensione. Alla fine il server manda al client la stringa "FILETRANSMITTED". A questo punto

il client si occupa di confrontare la dimensione ricevuta all'inizio con quella effettiva, in modo da capire se il file è stato scaricato per intero.

### Gestione degli errori

Se il file passato come parametro non esiste il client riceve dal server una stringa contenente il messaggio "ERROR". In caso di eccezione invece riceve la stringa "EXCEPTION". In entrambi i casi il file non viene ricevuto.

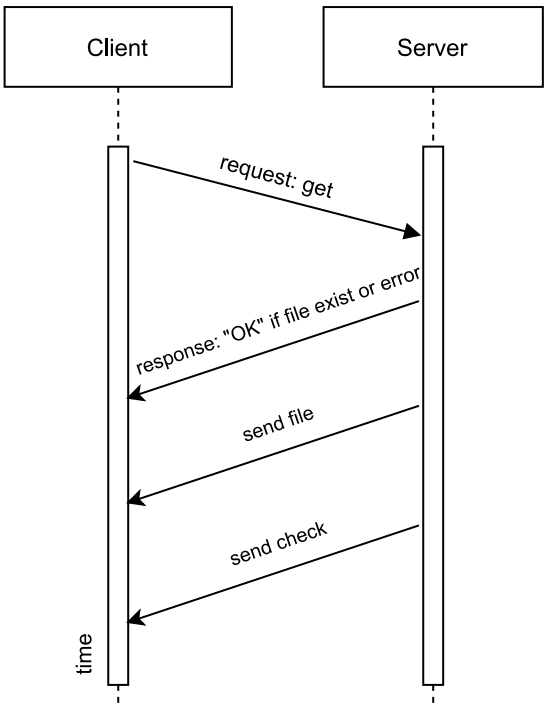


Figura 3.3: Comando get

## 3.4 Gestione comando put

Il client, ricevuto l'input del comando get con il relativo parametro, invia la richiesta al server che si mette in ascolto. A questo punto si occupa di controllare l'esistenza del file passato come parametro nella cartella "myFiles" ed in caso positivo trasmette una stringa contenente il messaggio "OK" al server, in modo da permettere l'inizio della trasmissione. Appena inviato l'ok il client invia anche il nome e la dimensione del file al server. In seguito comincia la trasmissione. Al termine della trasmissione il client manda al

server una stringa di controllo "FILETRANSMITTED". Quando il server la riceve controlla se il file è stato ricevuto per intero. Alla fine il client aspetta la ricezione di un messaggio "FileOK" se il file è arrivato correttamente, altrimenti "FileNotOK".

### Gestione degli errori

Se il file non esiste il client si occupa di visualizzare un errore. In caso di trasmissione non corretta il server invia al client un messaggio di errore "FileNotOK".

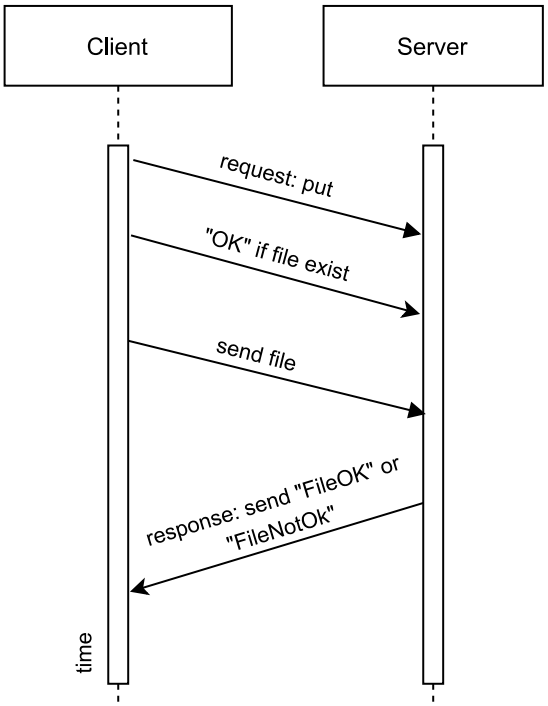


Figura 3.4: Comando put

## 3.5 Comando close

Comando che si occupa di terminare la connessione.

# Capitolo 4

# Istruzioni d'uso

Aprire e avviare il file server.py e il file client.py.
Digitare i comandi sulla console del client.
All'interno dell'elaborato consegnato vi saranno due cartelle, client e server, contenenti a loro volta delle sottocartelle. Per il client troviamo le sottocartelle "myFiles", dove è possibile inserire alcuni file di cui si può eseguire il comando put su server, e "received", cartella contenente tutti i file ricevuti dal server. Per il server troviamo solo la cartella "resources" che contiene tutti i file in suo possesso. Se si necessita di inserire dei file manualmente su server si proceda a caricare il materiale nella cartella "resources", altrimenti si inseriscano nella cartella "myFiles" del client e si proceda ad comando di put.

## 4.1 Sintassi

Si noti che le parole chiave dei comandi (list, get, put e close) non sono case sensitive, ovvero possono essere sia scritte in minuscolo, maiuscolo oppure solo qualche lettera in maiuscolo o minuscolo, mentre i nomi dei file sono case sensitive. In caso di scrittura del nome file in modo scorretto il file non verrà trovato e trasmesso. In caso di assenza di estensione il file non verrà trovato. Ci si assicuri anche che non vi siano spazi a seguire il nome del file oppure quest'ultimo non sarà trovato.

- Comando List → list
- Comando Get → get ⟨nomeFile.estensione⟩
- Comando Put → put ⟨nomeFile.estensione⟩
- Comando Close → close

## 4.2 Link repository github

This is my link: `https://github.com/lucabarattini4/Client-Server-UDP-for-file-transfer`