# Applicazione Client-Server UDP per Trasferimento File

Marco Costantini - marco.costantini3@studio.unibo.it
Daniele Pancottini - daniele.pancottini@studio.unibo.it

27 Luglio 2022

# Indice

# 1 Analisi del Problema

Il progetto prevede la realizzazione di un'applicazione per il trasferimento di file (sviluppata in linguaggio Python) basata su un'architettura di tipo client-server in cui verrà usato il protocollo UDP.

L'applicazione dovrà soddisfare le seguenti funzionalità:

- Connessione client-server senza autenticazione
- Visualizzazione dei file caricati sul server
- Download di un file dal server
- Upload di un file sul server

Il client potrà richiedere al server l'elenco dei file caricati attraverso il comando ***list***, ed il server dovrà restituire la lista dei nomi dei file caricati, e quindi disponibili per il download.

Il client potrà scaricare un determinato file dal server attraverso il comando ***get filename***, ed il server dovrà restituire al client il file, mediante un opportuno protocollo di comunicazione.

Il client potrà caricare un determinato file sul server attraverso il comando ***put filename***, ed il server dovrà ricevere il file dal client, mediante un opportuno protocollo, e lo scriverà in memoria.

# 2 Design

## 2.1 Protocollo

Il client e il server comunicano tramite specifici comandi inviati dal client al momento della selezione relativa all'opzione desiderata (visulizzata nel menù), nello specifico i comandi inviati sono:

- **list**: richiesta della lista dei file disponibili
- **put** + ***'nome file'***: upload di un file sul server
- **get** + ***'nome file'***: download di un file

## 2.2 Comandi

### 2.2.1 Comando List

Il comando ***list*** permette al client di ricevere, dal server, la lista dei file caricati e disponibili per il download. Il server invierà al client la lista dei file sotto forma di vettore di stringhe, basterà quindi stampare il contenuto del vettore per soddisfare la prima funzionalità.

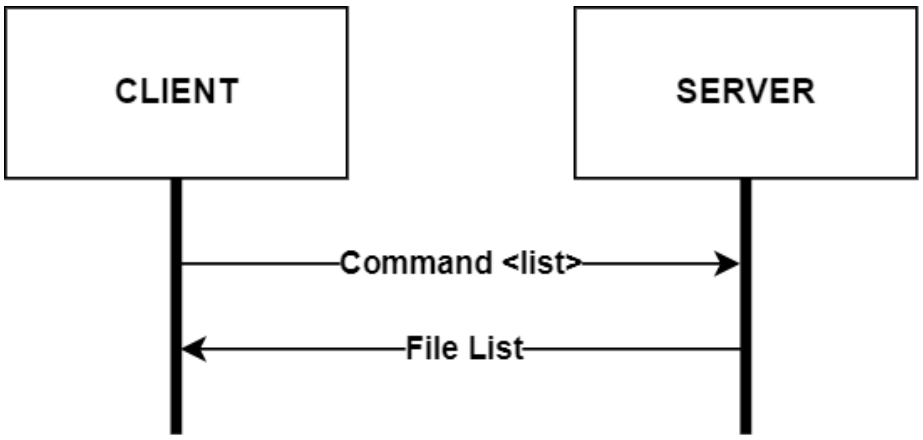


Figure 1: Schema Comando List

### 2.2.2 Comando Get

Il comando ***get filename*** permette al client di scaricare un file dal server. Il server, dopo aver ricevuto il comando ed il nome del file da scaricare ne verifica l'esistenza, se il file dovesse essere presente in memoria inizierà la trasmissione con il client per il download (utilizzando un sistema basato sul protocollo rdt 3.0), altrimenti restituirà un messaggio di errore.

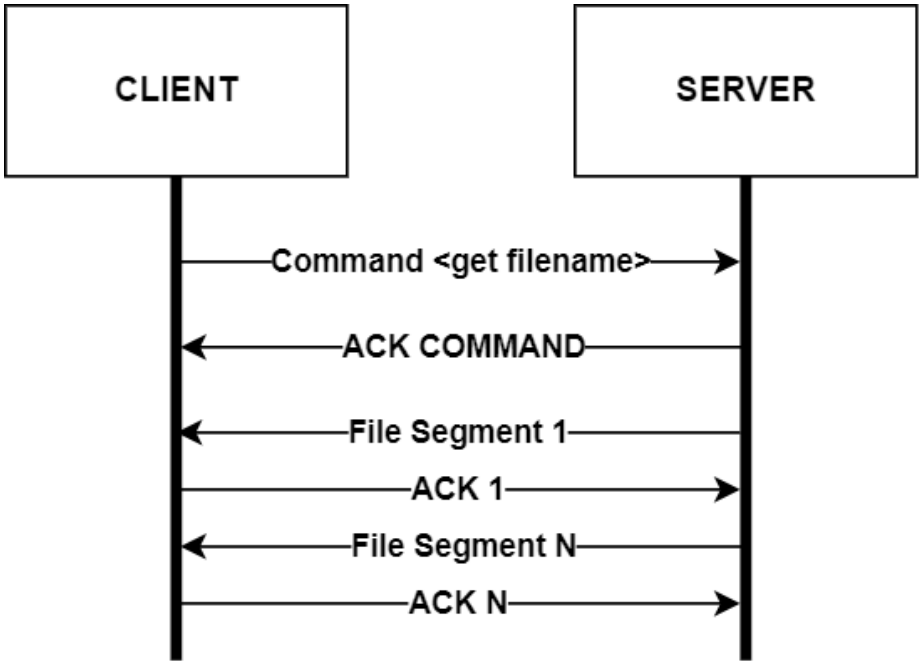


Figure 2: Schema Comando Get

### 2.2.3 Comando Put

Il comando ***put filename*** permette al client di caricare un file sul server. Il server, dopo aver ricevuto il comando ed il nome del file da caricare, risponde con un messaggio di ACK, ed attende l'inizio della trasmissione del file da parte del client, sempre utilizzando un sistema basato sul protocollo rdt 3.0.

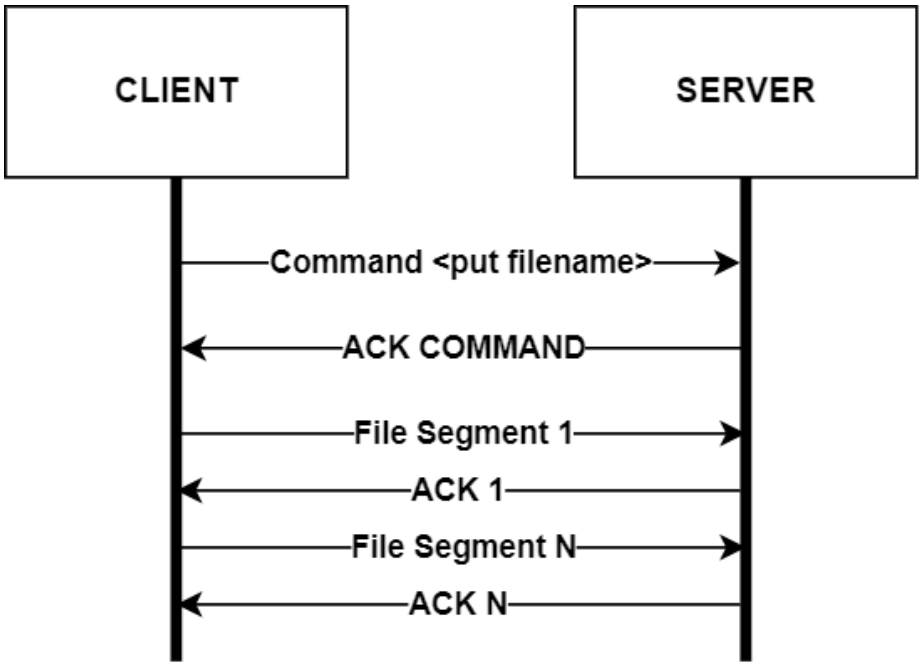


Figure 3: Schema Comando Put

# 3 Sviluppo

## 3.1 Client

La classe Client definisce il socket UDP per la comunicazione con il server. Tramite un menù di scelta vengono inviati i comandi al server (con il comando *sendto*) che li gestirà e restituirà il risultato al client. Ricevuto il risultato dal server (tramite il comando *recvfrom*) eseguirà a seconda del comando inviato:

- **List** stampa nome dei files ricevuti dal server.
- **Put** utilizzo della funzione rdt per la gestione dei pacchetti.
- **Get** controllo l'effettiva esistenza del file e successivamente utilizzo della funzione rdt per la gestione dei pacchetti.

## 3.2 Server

Il modulo ***server.py*** definisce il socket UDP del server attraverso il modulo standard Python: ***socket.py***. Il server, per come è definito, riesce a supportare le richieste di un client alla volta, utilizzando uno switch per smistare i vari comandi (list, get, put). In caso di richieste di tipo GET e PUT, il server delega la trasmissione o la ricezione dei FILE SEGMENT al modulo ***rdt_handler.py***,

che fornisce i metodi necessari per trasmettere o ricevere file utilizzando il protocollo RDT 3.0.

## 3.3 RDT Handler

Il modulo ***rdt_handler.py*** definisce le strutture dati necessarie per l'implementazione del potocollo RDT 3.0, ed una classe simil controller che espone i metodi per l'interscambio di file utilizzando sempre il protocollo RDT 3.0.
Le strutture dati definite da questo modulo sono due:

- Classe ***Packet***: struttura dati principale per l'interscambio dei file segment
- Enum ***PacketType***: viene utilizzato nella classe Packet e definisce i tipi di pacchetti gestibili (ACK PACKET, DATA PACKET).

La classe ***RdtFileTransferHandler*** espone i seguenti metodi:

- ***rdtFileDataReceiver***: metodo che implementa la logica per la ricezione di un file
- ***rdtFileDataSender***: metodo che implementa la logica per la trasmissione di un file

# 4 Guida Utente

## 4.1 Client

## 4.2 Server

Per avviare il server è sufficiente eseguire il comando:

```
python3 server.py
```