Trasferimento file Client - Server UDP

Cristina Zoccola - cristina.zoccola@studio.unibo.it - 0000969874
Sara Romeo - sara.romeo3@studio.unibo.it - 0000969946

May 2022

Indice

1 Indicazioni per l'uso dell'applicativo

Il programma è strutturato per essere eseguito da linea di comando. Sarà necessario dividere l'esecuzione in due prompt dei comandi windows, nei quali rispettivamente bisogna:

- entrare nella cartella Server ed eseguire Server.py
- entrare nella cartella Client ed eseguire Client.py

L'applicativo consente di caricare e scaricare file dal server di tutti i tipi. In particolare vi sono file di esempio in formato txt, png, mp3, mp4:

- **Client:**
  - file1.txt 1 Kb
  - file2.png 2 Kb
  - file3.mp3 78 Kb
- **Server:**
  - list1.txt 1 Kb
  - list2.png 2 kb
  - list3.mp3 83 kb
  - list4.mp4 546 Kb

Per richiedere la lista dei file disponibili alla condivisione sarà sufficiente scrivere a lato client il comando "list", mentre per caricare e scaricare file il formato sarà rispettivamente "put file-name" e "get file-name". Infine per chiudere il client e il server contemporaneamente il comando è "exit".

2 Descrizione generale delle scelte di progetto effettuate

È stato scelto di implementare tutte le funzioni di Client e server in un file esterno chiamato Commands. In questo modo risulta più chiara la lettura dell'implementazione e permette un riuso di codice da parte di entrambi per funzioni in comune. La dimensione dei pacchetti per inviare dati convenuta è di 4096 KB. Ogni file viene aperto, letto e scritto in maniera binaria. Nel momento di download e upload dei file, si è reputato necessario inserire un timeout per la ricezione e invio dei pacchetti. Questo perché se il ricevente

non ottiene pacchetti durante quel periodo di tempo significa che la ricezione del file è terminata, quindi grazie a ciò si evita di lasciarlo in costante attesa. All'inizio del loop principale viene impostato a none in modo tale che possa rimanere in ascolto del prossimo messaggio.

3 Descrizione delle funzioni

All'avvio il Server si mette in attesa dentro un loop di ricevere messaggio dal Client. Se questo invia un comando corretto procede con l'esecuzione richiamando la funzione relativa, altrimenti restituisce "Unkown input" e si rimette in attesa. Dall'altro lato quando l'utente inserisce un input il Client controlla qual'è il comando e chiama la funzione relativa. Se il comando è sbagliato riceve il messaggio di unknown input dal server e si rimette in attesa di inserimento dell'utente.

3.1 Get

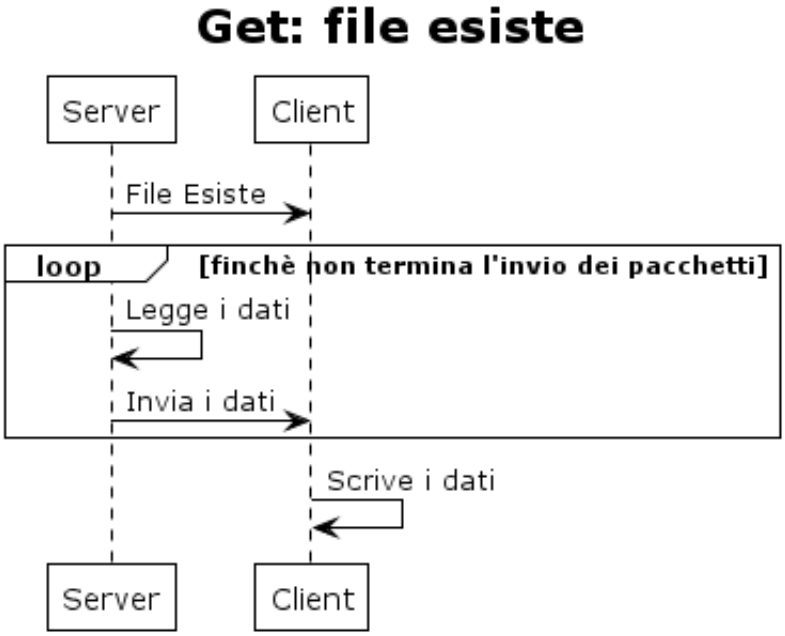

Figura 1: Get file esiste

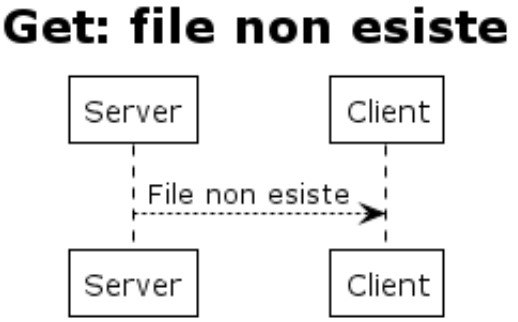

Figura 2: Get file non esiste

Il Server chiama ServerGet(): se il file esiste nella directory lo apre e lo legge, mandando al Client i pacchetti che lo contengono altrimenti invia un messaggio di file non esiste e si rimette in attesa. In ClientGet() si riceve un messaggio per sapere se il file esiste o meno. Se questo non è presente lo comunica all'utente ed esce dalla funzione, altrimenti procede con il download del file. Apre un file, ottiene tutti i pacchetti relativi e successivamente li scrive nel file aperto. Questo verrà chiuso e l'operazione termina tornando al loop principale.

3.2 Put

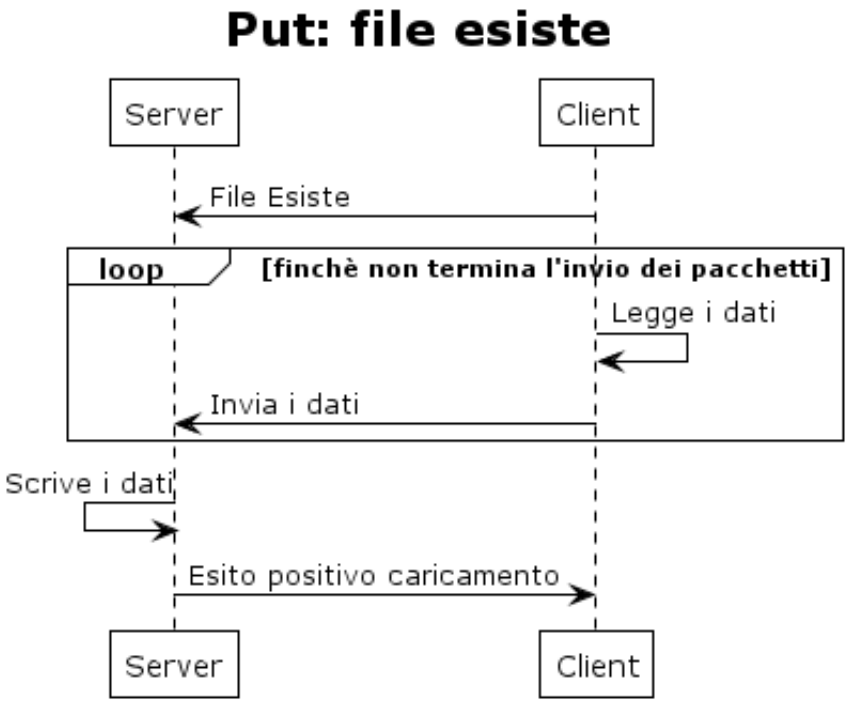

Figura 3: Put file esiste

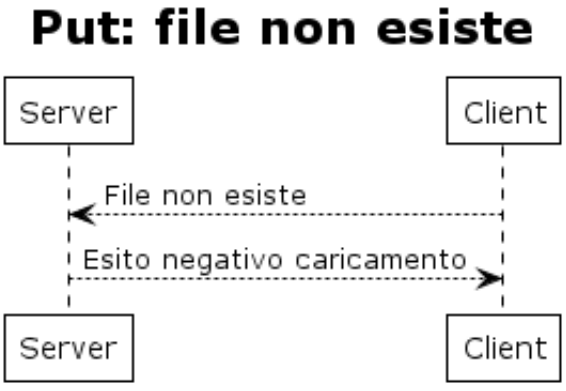

Figura 4: Put file non esiste

Per prima cosa viene chiamata la ClientPut() che controllerà se il file esiste. Se non esiste il processo invia un messaggio di errore al server il quale

a sua volta manda un'avviso di file non caricato tornando in attesa. Nel caso contrario continua aprendo e leggendo il file inviandolo al server. Quest'ultimo nella ServerPut() riceverà tutti i pacchetti necessari e in seguito li scriverà nel file aperto all'inizio dell'upload, inviando infine l'esito positivo dell'operazione che verrà ricevuto dal Client().

3.3 List

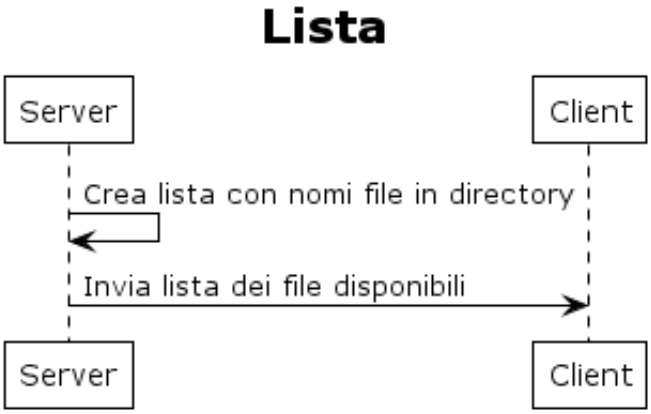

Figura 5: List

ServerList() si occupa di prelevare il percorso della directory corrente grazie alle funzioni della libreria os di Python. Dopodiché verrà creata una lista e ciclando i file esistenti nella cartella si inserisce il nome di ognuno. Finito di memorizzare l'elenco dei file disponibili, si rimuove quello del sorgente Server.py, in quanto non disponibile alla condivisione. Infine la lista viene trasformata in una stringa, a sua volta codificata di default in utf-8 per l'invio al Client. Questo si occuperà solo di rimanere in attesa del messaggio.

3.4 Exit

Si chiude il socket sia lato Server che Client e il processo termina.