Programmazione di Reti

Elaborato 2

Gruppo composto da:

Pablo Sebastian Vargas Grateron
Matricola: 0000970487
Email: pablo.vargasgrateron@studio.unibo.it

Luca Babboni
Matricola: 0000971126
Email: luca.babboni2@studio.unibo.it

Scopo del progetto
Progettazione ed implementare in linguaggio Python un'applicazione client-server per il trasferimento di file, basato sul protocollo UDP.

Funzionalità del software:
- Visualizzazione sul client dei file disponibili sul server
- Download di file presenti sul server
- L'upload di un file sul server

Scelte generali di progettazione

Comunicazione client-server
Per la comunicazione client-server è stato scelto di utilizzare un sistema di pacchetti contenenti stringhe. Questi pacchetti possono contenere sia messaggi di errore che comandi da far eseguire all'altra parte di codice.

Sistema di controllo integritá file
Poiché si sta utilizzando il protocollo UDP abbiamo ritenuto necessario un controllo dell'integritá del file ricevuto. Questo controllo avviene alla fine del download del file da parte del receiver.
Durante l'upload il sender invierá un pacchetto contenente l'hash code del file che ha inviato. Questo viene poi controllato con l'hash code del file ricevuto. Se la verifica dà esito positivo, e quindi gli hash code sono identici, viene dato in output un messaggio di avvenuto trasferimento.
In caso di esito negativo il file ricevuto viene scartato dal receiver e viene dato in output un messaggio di fallito trasferimento di file.
Per il calcolo dell'hash code é stato utilizzato l'algoritmo md5.

Trasmissione file
Dal lato del sender viene controllato se il file esiste e viene inviato un primo pacchetto al server contenente i dettagli della richiesta tra cui il nome del file. Il file viene quindi aperto e di volta in volta ne viene letta e inviata una quantità di dati pari alla dimensione del pacchetto (variabile BUFFER_SIZE, settata da noi a 1024 bit).
L'ultimo pacchetto inviato contiene l'hash code del file per la verifica di integrità.

Comandi esposti lato Client:

list

Visualizza sul terminale di esecuzione del Client i file presenti all'interno della cartella di esecuzione del programma Server.
L'implementazione prevede che una volta arrivata la richiesta sul lato server venga creato un file temporaneo txt contenente al suo interno la lista dei nomi dei file presenti nella cartella di esecuzione, che poi verrà inviata al lato Client.
Si è preferita questa implementazione all'inviare un semplice pacchetto contenente una lista perché, in caso di un numero molto elevato di file all'interno della cartella, ci sarebbe stato il rischio di eccedere la dimensione del pacchetto e quindi perdere informazioni.
Una volta inviata e ricevuta la lista il file temporaneo che la contiene viene eliminato.

get *filename*

Se il file esiste nella cartella di esecuzione del programma Server viene trasferito nella cartella in cui è in esecuzione Client. In caso di fallito trasferimento o di file non trovato viene stampato il corrispondente messaggio di errore.

put *filename*

Se il file esiste nella cartella di esecuzione del programma Client viene trasferito nella cartella in cui è in esecuzione Server. In caso di fallito trasferimento o file non trovato viene dato in output al terminale il corrispondente messaggio di errore.

removelocal *filename*

Se il file esiste nella cartella di esecuzione del programma Client viene eliminato, in caso contrario viene dato in output al terminale il corrispondente messaggio di errore.

removeserver *filename*

Se il file esiste nella cartella di esecuzione del programma Server viene eliminato, in caso contrario viene dato in output al terminale il corrispondente messaggio di errore.

exit

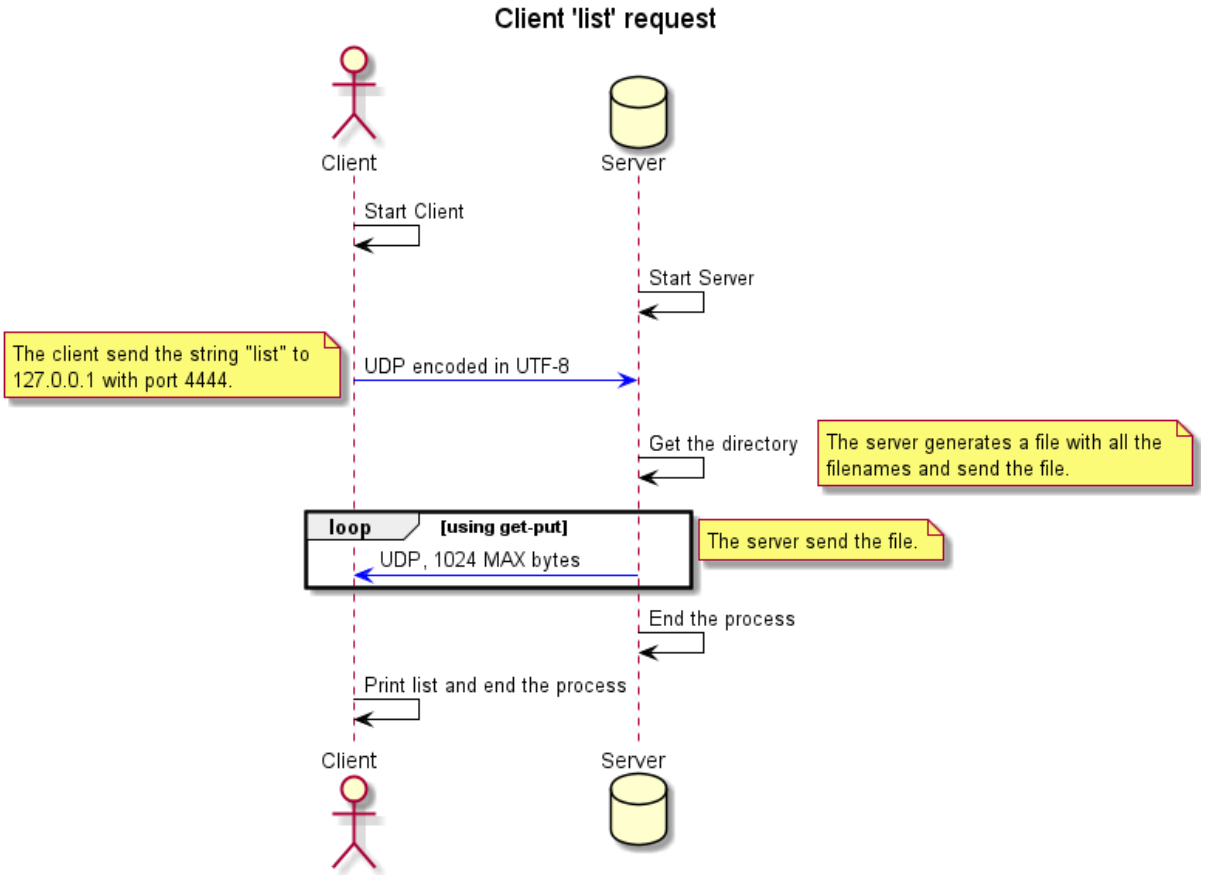
Termina l'esecuzione del programma Client

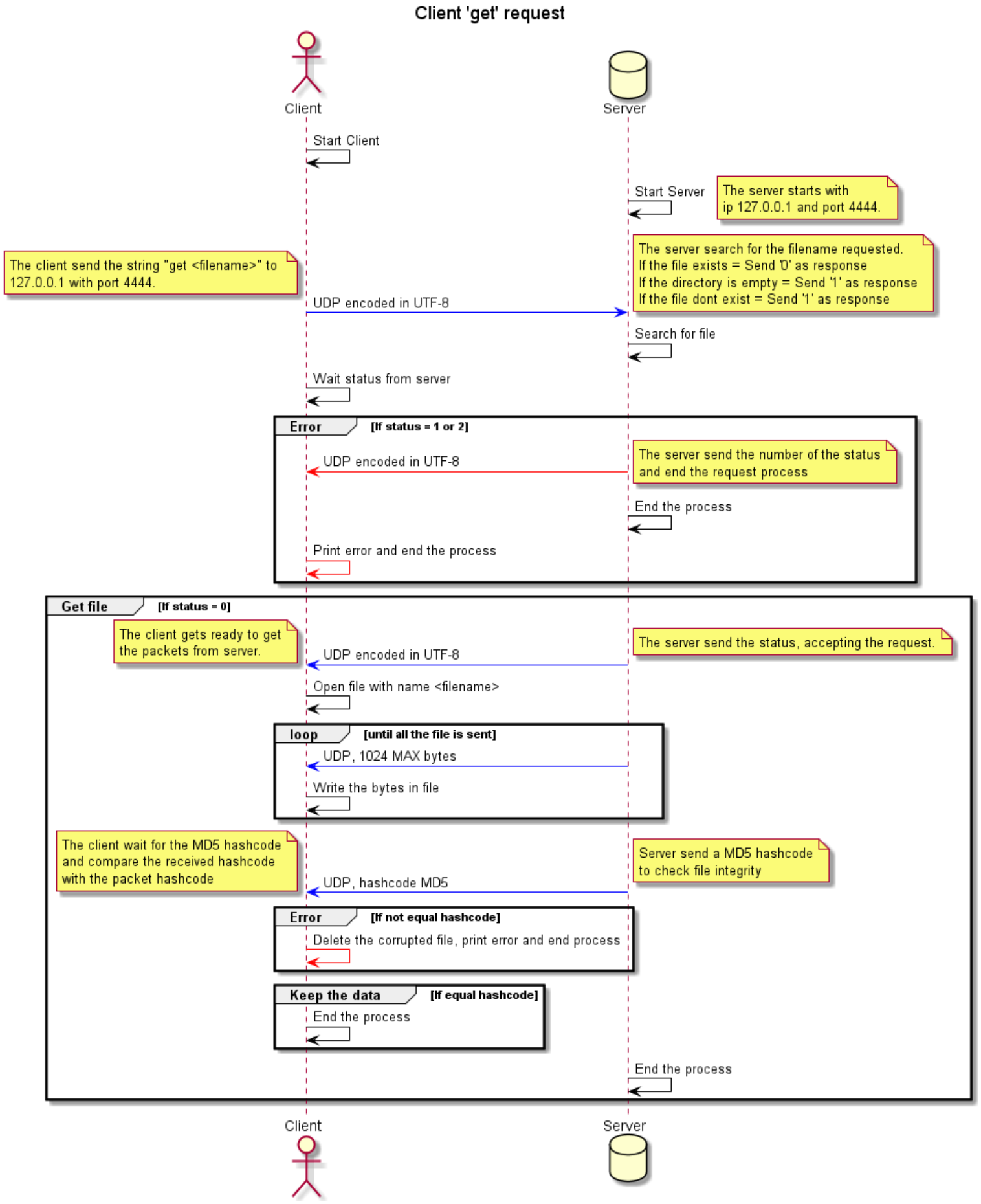
exitserver

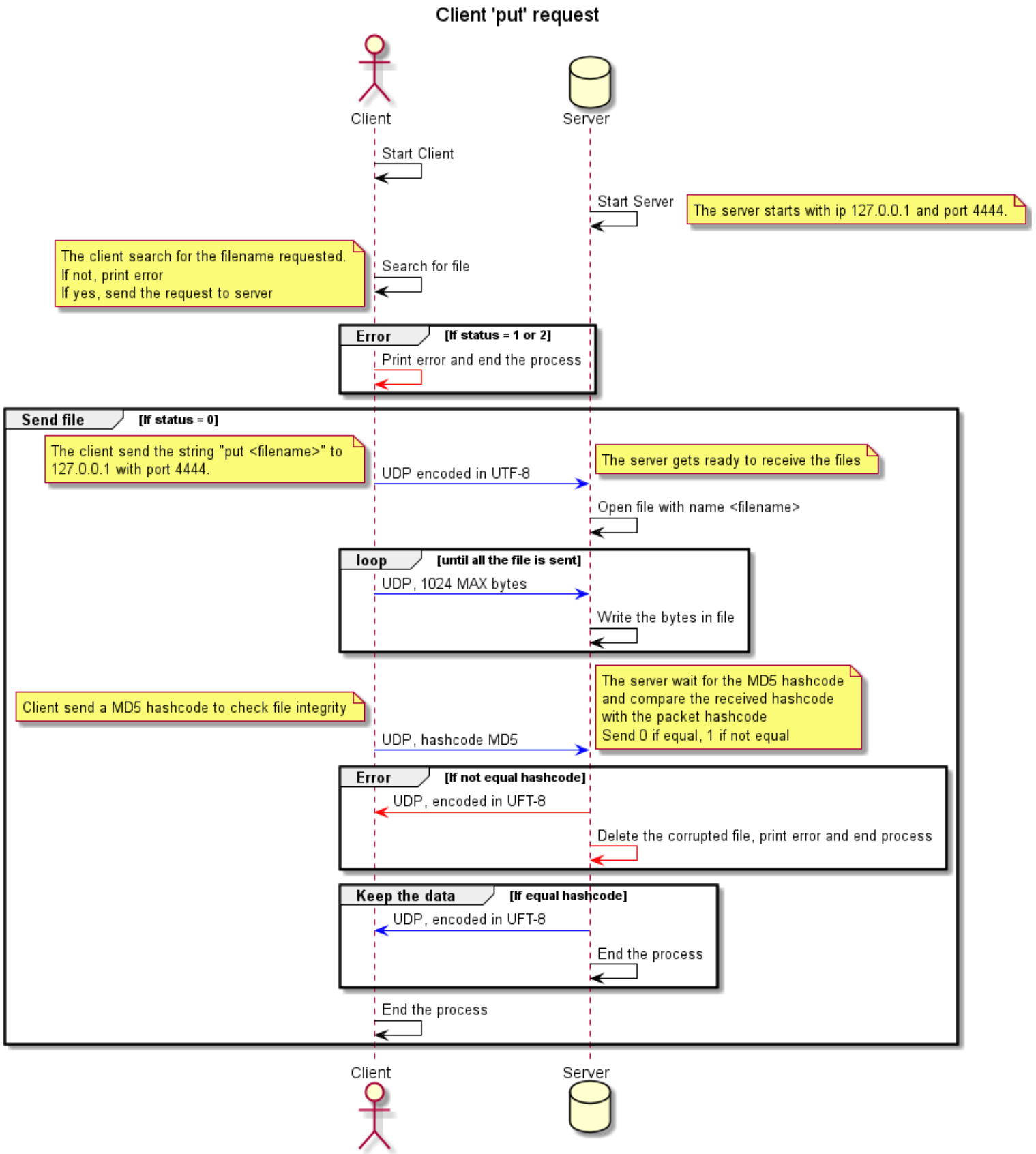
Termina l'esecuzione del programma Server

UML

Di seguito vengono riportati gli uml che spiegano il funzionamento delle funzioni principali precedentemente descritte




Client 'get' request
Client
Server
Start Client
Start Server
The server starts with ip 127.0.0.1 and port 4444.
The client send the string "get <filename>" to 127.0.0.1 with port 4444.
The server search for the filename requested.
If the file exists = Send '0' as response
If the directory is empty = Send '1' as response
If the file dont exist = Send '1' as response
UDP encoded in UTF-8
Search for file
Wait status from server
Error
[If status = 1 or 2]
UDP encoded in UTF-8
The server send the number of the status and end the request process
End the process
Print error and end the process
Get file
[If status = 0]
The client gets ready to get the packets from server.
The server send the status, accepting the request.
UDP encoded in UTF-8
Open file with name <filename>
loop
[until all the file is sent]
UDP, 1024 MAX bytes
Write the bytes in file
The client wait for the MD5 hashcode and compare the received hashcode with the packet hashcode
Server send a MD5 hashcode to check file integrity
UDP, hashcode MD5
Error
[If not equal hashcode]
Delete the corrupted file, print error and end process
Keep the data
[If equal hashcode]
End the process
End the process
Client
Server


Client 'put' request
Client
Server
Start Client
Start Server
The server starts with ip 127.0.0.1 and port 4444.
The client search for the filename requested.
If not, print error
If yes, send the request to server
Search for file
Error
[If status = 1 or 2]
Print error and end the process
Send file
[If status = 0]
The client send the string "put <filename>" to 127.0.0.1 with port 4444.
UDP encoded in UTF-8
The server gets ready to receive the files
Open file with name <filename>
loop
[until all the file is sent]
UDP, 1024 MAX bytes
Write the bytes in file
Client send a MD5 hashcode to check file integrity
The server wait for the MD5 hashcode and compare the received hashcode with the packet hashcode
Send 0 if equal, 1 if not equal
UDP, hashcode MD5
Error
[If not equal hashcode]
UDP, encoded in UFT-8
Delete the corrupted file, print error and end process
Keep the data
[If equal hashcode]
UDP, encoded in UFT-8
End the process
End the process
Client
Server