matricola: 0000971342
email: Federico.muccioli5@studio.unibo.it

Traccia 2

Architettura client-server UDP per trasferimento file

Lo scopo de progetto 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 strato di trasporto).

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 messaggi:
- 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. **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
  messaggio di 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.
  Funzionalità del client: I 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 ricezione 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.

matricola: 0000971342
email: Federico.muccioli5@studio.unibo.it

Esecuzione:

Lanciare entrambe le applicazioni, Server.py e Client.py. Dall'interfaccia del client sarà possibile impartire diversi ordini per la gestione del server, più in dettaglio:

-LIST stampa in output la lista dei file presenti nel server, situati nella cartella "server_data".

-GET<filename> scarica il file, di cui specificato il nome, nella cartella "client_data".

-PUT<path> carica il file, di cui specificato il percorso, nel server, nella cartella "server_data".

-DELETE<filename> elimina il file, di cui specificato il nome, dal server, dalla cartella "server_data".

-HELP stampa in output la lista dei comandi impartibili al server.

matricola: 0000971342
email: Federico.muccioli5@studio.unibo.it

Scelte progettuali:

La comunicazione tra le due entità (Server e Client) è gestita tramite l'invio e la ricezione di comandi, messaggi di conferma/errore e dati di file. Per poter gestire ogni tipo di file è stato scelto di comunicare ogni dato trasmesso in codifica binaria.

Il buffer di comunicazione è stato scelto come potenza di 2 ovvero da 1024 byte. In caso il codice venga eseguito su una macchina di bassa potenza o sono state implementate nuove funzioni che rallentano lo svuotamento del buffer, è possibile aumentare le dimensioni in modo da evitare che il sistema operativo debba eliminare alcuni pacchetti in entrata e perdere preziosi dati.

Durante l'invio dei file, tra una trasmissione e l'altra, è stato scelto di impostare al mandante un delay minimo che possa permettere al ricevente di salvare i dati e poter tornare in ascolto evitando la perdita di pacchetti. Questo delay non dovrà comunque essere troppo elevato o si rischia di rallentare la connessione.

Il Server ha la caratteristica di essere strutturato utilizzando un ciclo infinito nell'attesa di ricevere un qualsiasi comando da parte di un qualsiasi Client. Nel momento della ricezione di un messaggio contenente un comando, il Server si presterà a creare un Thread che avrà lo scopo di soddisfare la richiesta di tale comando appena ricevuto. Questo da garantire il continuo funzionamento del Server, anche in presenza di una molteplicità di Client che eseguono richieste nello stesso momento.

Ogni file caricato sul server o scaricato dal client verrà salvato in una directory presente nello stesso percorso del sorgente ("server_data" nel caso del server, "client_data" nel caso del client).
Questa scelta è stata fatta in modo tale da non permettere l'accesso ai sorgenti e non poterne modificare il contenuto. Ad esempio, con la funzione DELETE, sarebbe possibile eliminare il server.py.

matricola: 0000971342
email: Federico.muccioli5@studio.unibo.it

Nella gestione del server è stato scelto di implementare una nuova funzione DELETE che possa permettere di eliminare un file residente nel server. Questa funzione è necessaria in quanto non è possibile sovrascrivere un file grazie ai controlli implementati e nel caso si volesse aggiornare un file è possibile prima eliminarlo dal repository per poi caricare la versione aggiornata. Questo comando è anche necessario nel caso non servissero determinati file e si volesse liberare lo spazio di archiviazione.

E' stato scelto di implementare anche un'altra nuova funzione HELP. Semplicemente impartendo il comando viene ritornata una lista dei comandi impartibili al server. Questa funzione non cambia il funzionamento della gestione dei file ma ritorna comunque utile a chi impartisce dal lato client ai fini di chiarezza.

Se viene impartito un comando non riconosciuto, in output viene mandata una stringa d'errore e la lista dei comandi impartibili; all'interno del codice dopo aver inviato il messaggio di errore viene richiesto il comando HELP per ottenere la lista di comandi validi.

Scelte strutturali:

Per la gestione delle richieste è stato scelto di utilizzare messaggi composti da stringhe poi codificate all'invio e decodificate alla ricezione. I comandi GET, PUT, LIST, DELETE, HELP sono case sensitive. Nel caso di comandi che necessitano di un argomento, come per il caso di GET, che è necessario specificare il nome del file da scaricare, l'argomento verrà aggiunto alla stringa del comando separato da uno spazio.

matricola: 0000971342
email: Federico.muccioli5@studio.unibo.it

Comandi nello specifico:

-LIST stampa in output la lista dei file presenti nel server, situati nella cartella “server_data”.
Nel caso non ci fossero file presenti stampa a video "The server directory is empty".

-GET<filename> scarica il file, di cui specificato il nome, nella cartella “client_data”.
Se non viene specificato il nome del file stampa a video "Error: enter the name of the file to download".
Per evitare di sovrascrivere involontariamente un file già presente se esiste già un file con lo stesso nome del file richiesto nella cartella del client verrà stampato "Error: file with the same name already present on directory".
Nel caso il file di cui specificato il nome non fosse presente nel server verrà stampato "Error: file not found".
Se durante la scrittura o lettura dei file incombessero problemi verrà stampato a video “Error: retry the operation”.
Nel caso del corretto funzionamento e del download del file verrà stampato “File downloaded successfully".

-PUT<path> carica il file, di cui specificato il percorso, nel server, nella cartella “server_data”.
Se non viene specificato il nome del file stampa a video "Error: enter the path of the file to upload".
Se non viene trovato il file, di cui specificato il percorso, stampa a video "Error: enter the path of a valid file".
Per evitare di sovrascrivere involontariamente un file già presente se esiste già un file con lo stesso nome del file da caricare nella cartella del server verrà stampato "Error: file with the same name already present on database".
Se durante la scrittura o lettura dei file incombessero problemi verrà stampato a video “Error: retry the operation”.

matricola: 0000971342
email: Federico.muccioli5@studio.unibo.it

Nel caso del corretto funzionamento e dell'upload del file verrà stampato "File uploaded successfully".

-DELETE<filename> elimina il file, di cui specificato il nome, dal server, dalla cartella "server_data".
Se non viene specificato il nome del file stampa a video "Error: enter the name of the file to delete".
Nel caso la directory del server fosse vuota verrà stampato "Error: the server directory is empty".
Nel caso il file non venisse trovato stampa a video "Error: file not found".
Nel caso incombessero problemi nell'eliminazione del file viene stampato a video "File not deleted".
Nel caso dell'avvenuta eliminazione del file viene stampato a video "File deleted successfully".

-HELP stampa in output la lista dei comandi impartibili al server.

-Se non viene richiesto un valido comando stampa a video "Error: insert a valid comand:" e la lista dei comandi impartibili al server.

matricola: 0000971342
email: Federico.muccioli5@studio.unibo.it

Schema UML: