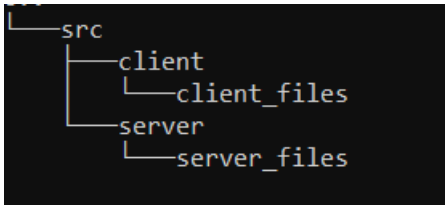
Relazione Progetto Programmazione di Reti

Leonardo Randacio
matricola 0000970515
leonardo.randacio@studio.unibo.it

Introduzione

Ho deciso di svolgere la traccia 2, presente su https://virtuale.unibo.it/pluginfile.php/1221482/mod_resource/content/1/tracce%20Progetti%20di%20fine%20corso%202021_2022.pdf
Ho creato una repository su github contenente il progetto e la relazione:
https://github.com/Oldranda1414/ProgettoReti

Struttura del progetto



Il progetto comprende una cartella src, contenente due cartelle, client e server, e uno script bash fileBuilder.sh. All'interno delle cartelle client e server sono presenti i due script corrispondenti, oltre a delle cartelle contenenti i file txt che permettono di simulare contenuti presenti sul client e sul server.

Dettagli progettuali

Il client e server comunicano facendo uso di UDP.
Il protocollo usato dal client e dal server per comunicare comprende un primo messaggio di comando inviato dal client e una o più riposte dal server. Se il client deve fornire indicazioni aggiuntive il primo messaggio di comando viene seguito dai dati necessari al server per completare l'operazione. Nelle operazioni più complesse il client attende un codice specifico di conferma dal server, oltre ai dati richiesti. Per facilitare la comunicazione a volte il server e il client comunicano con più messaggi diversi, per semplificare la complessità del codice. I dettagli del protocollo vengono esposti in seguito

Il server e il client possono svolgere 3 operazioni.

1) Il client può richiedere una lista dei file presenti sul server:
Questa operazione viene svolta usando 3 messaggi UDP. Prima il client fa la richiesta al server, poi il server invia 2 messaggi: con il primo il server invia il numero di file presenti su di esso e poi invia la lista composta dai nomi di ogni singolo file. In caso di un errore interno al server esso invia un messaggio contenente l'errore e il client riconosce che il messaggio non è corretto e mostra a terminale l'errore riportato dal server

2) Il client richiede un download di un file presente sul server:
Questa operazione viene svolta usando 4 messaggi UDP. Il client manda due messaggi, il primo il messaggio di comando corrispondente, mentre il secondo contenente il nome del file richiesto. Il server invia un messaggio che comunica al client se il file è presente sul server e poi, se lo è, invia i contenuti del file richiesto. In caso di un errore interno al server esso invia un messaggio contenente l'errore e il client riconosce che il messaggio non è corretto e mostra a terminale l'errore riportato dal server.

3) Il client richiede di caricare un file sul server
Questa operazione viene svolta usando 4 messaggi UDP. Il client invia 3 messaggi in successione, contenenti in ordine: il comando corrispondente, il nome del file che si vuole caricare sul server e i contenuti del file. Il server una volta scaricato il file invia un messaggio di conferma al client. In questo modo il client si può rendere conto se l'operazione è andata a buon fine. In caso di un errore interno al server esso invia un messaggio contenente l'errore e il client riconosce che il messaggio non è corretto e mostra a terminale l'errore riportato dal server.

Guida dell'utente

Si possono avviare i due script di python su due macchine diverse, oppure sulla stessa macchina in due terminali differenti

```
1) get the list of file in the server
2) get the contents of a file from the server
3) upload a file on the server
4) exit
Input a valid command[1/2/3/4]:
```

```
starting up the server on localhost on port 12000
ready to receive a message...
```

A sinistra si può vedere il terminale del client, sulla destra quello del server

Il client può eseguire una delle 3 operazioni inserendo il codice corrispondente, oppure chiudere il client inserendo il codice corrispondente.
Nel caso in cui l'utente voglia svolgere l'operazione 2 o 3 dovrà poi inserire il nome del file che vuole scaricare dal server o caricare sul server

```
1) get the list of file in the server
2) get the contents of a file from the server
3) upload a file on the server
4) exit
Input a valid command[1/2/3/4]: 1
command accepted
there are 10 files on the server
['server_file_1.txt', 'server_file_10.txt', 'server_file_2.txt', 'server_file_3.txt', 'server_file_4.txt
', 'server_file_5.txt', 'server_file_6.txt', 'server_file_7.txt', 'server_file_8.txt', 'server_file_9.tx
t']



1) get the list of file in the server
2) get the contents of a file from the server
3) upload a file on the server
4) exit
Input a valid command[1/2/3/4]: 2
command accepted
input the name of the file you want to download: server_file_1.txt
file found on the server
downloading the file...
file downloaded



1) get the list of file in the server
2) get the contents of a file from the server
3) upload a file on the server
4) exit
Input a valid command[1/2/3/4]:
```

terminale lato client dopo aver svolto le operazioni 1 e 2

```
starting up the server on localhost on port 12000
ready to receive a message...
message received: get list
list of files sent to the client




ready to receive a message...
message received: get file
waiting for filename...
file sent to the client
```

terminale lato server dopo aver svolto le operazioni 1 e 2

```
1) get the list of file in the server
2) get the contents of a file from the server
3) upload a file on the server
4) exit
Input a valid command[1/2/3/4]: 3
command accepted
input the name of the file you want to upload: client_file_1.txt
sending client_file_1.txt to the server...
client_file_1.txt uploaded correctly to the server




1) get the list of file in the server
2) get the contents of a file from the server
3) upload a file on the server
4) exit
Input a valid command[1/2/3/4]:
```

terminale lato client dopo aver svolto l'operazione 3

```
ready to receive a message...
message received: upload
waiting for filename...
filename received
waiting for the file contents...
file contents received
file uploaded to the server




ready to receive a message...
```

Terminale lato server dopo aver svolto l'operazione 3