

RELAZIONE ELABORATO PROGRAMMAZIONE DI RETI TRACCIA 2

Gabriele Rubboli Petroselli
Marco Ravaioli

Introduzione

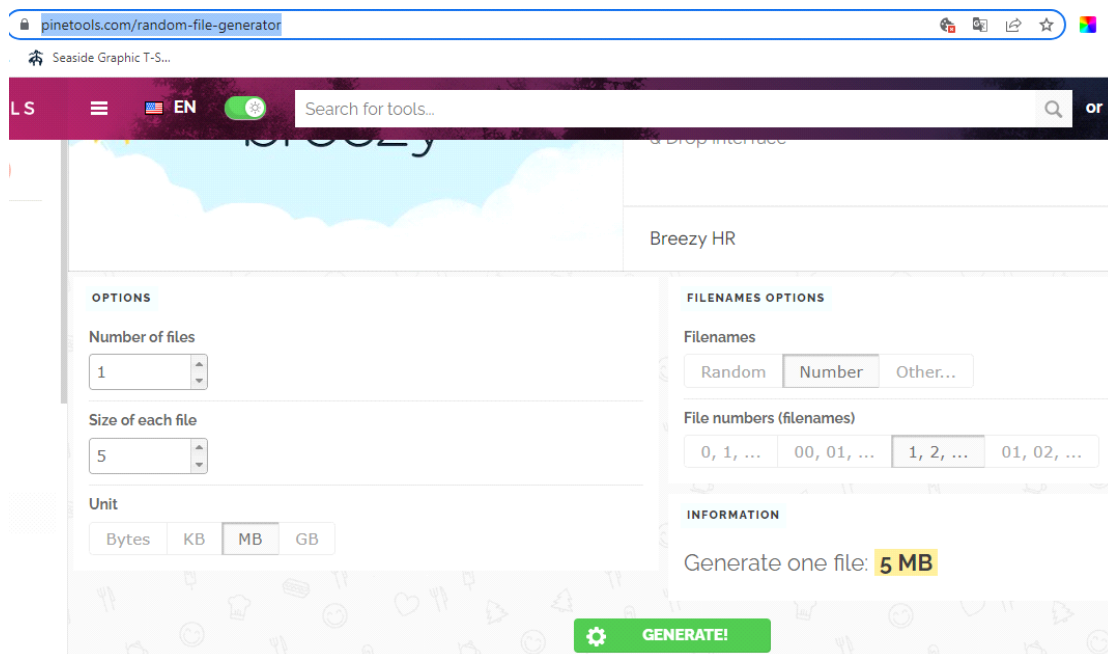
Sviluppo dell'elaborato di programmazione di reti, Traccia 2.

Struttura progetto

la struttura si divide nelle directory del Server e del Client, al cui interno sono rispettivamente contenuti i File .py di Server e Client.

All'interno delle cartelle contenenti i .py è necessario inserire dei file di prova per il corretto testing dell'elaborato. Per praticità viene inserito di seguito un link per la generazione di file online:

<https://pinetools.com/random-file-generator>



Come si presenta il File Generator

Gli Script Python si possono lanciare da da terminale, riservando un terminale differente per ciascuno di essi:

```

Prompt dei comandi
Microsoft Windows [Versione 10.0.19043.1826]
(c) Microsoft Corporation. Tutti i diritti sono riservati.

C:\Users\mrava>cd "OneDrive\Desktop\Programmazione di Reti\src\Server"
C:\Users\mrava\OneDrive\Desktop\Programmazione di Reti\src\Server>python ServerUDP.py 127.0.0.1 8080

Prompt dei comandi
Microsoft Windows [Versione 10.0.19043.1826]
(c) Microsoft Corporation. Tutti i diritti sono riservati.

C:\Users\mrava>cd "OneDrive\Desktop\Programmazione di Reti\src\Client"
C:\Users\mrava\OneDrive\Desktop\Programmazione di Reti\src\Client>python ClientUDP.py 127.0.0.1 8080_

```

Esempio del procedimento di lancio degli script dai terminali

Guida alle funzionalità

Si eseguano gli script come nell'esempio sopra riportato, assicurandosi di inserire correttamente anche Indirizzo IP e Porta del Server.

E' necessario che la scelta di indirizzo e porta sia uguale in entrambi i terminali poiché in ambedue i casi viene richiesto l'inserimento dei dati del Server.

I programmi così composti funzionano solo se lanciati in locale quindi con indirizzo IP 127.0.0.1 o localhost.

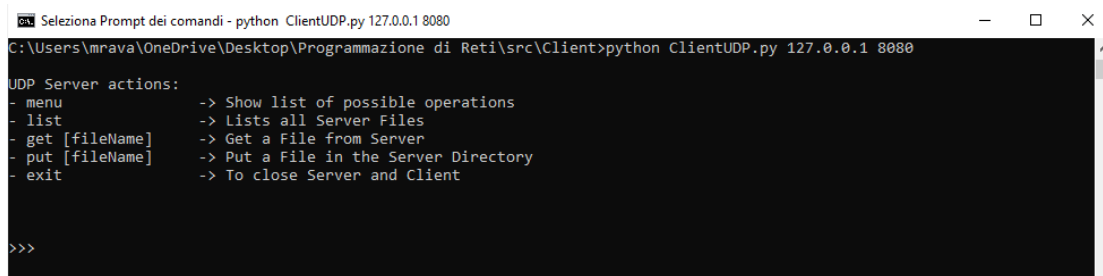
La porta scelta deve essere di valore compreso tra 1025 e 65535, nel caso

contrario sarà lo script a notificare l'errore.

Per il corretto funzionamento bisogna avviare prima il Server, altrimenti il Client segnalerà un errore nella connessione a quest'ultimo.

Il corretto avvio del Server è accompagnato da un messaggio di conferma nel terminale.

Non appena il Client effettuerà la connessione verranno visualizzate tutte le funzionalità disponibili.



```
Seleziona Prompt dei comandi - python ClientUDP.py 127.0.0.1 8080
C:\Users\mrava\OneDrive\Desktop\Programmazione di Reti\src\Client>python ClientUDP.py 127.0.0.1 8080

UDP Server actions:
- menu          -> Show list of possible operations
- list          -> Lists all Server Files
- get [fileName] -> Get a File from Server
- put [fileName] -> Put a File in the Server Directory
- exit          -> To close Server and Client

>>>
```

Il programma verifica che le istruzioni impartite dall'Utente siano scritte correttamente o altrimenti viene notificato.

Con le istruzioni Get e Put è richiesto l'inserimento del nome del file che verrà inviato o ricevuto.

L'elaborato garantisce il trasferimento di file in ogni formato dato che il trasferimento avviene in binario, fino ad una grandezza di 1024Kb, senza escludere il corretto funzionamento anche con file di dimensione superiore.

E' facile intuire che il tempo necessario al trasferimento di file è direttamente proporzionale alla grandezza del file stesso.

L'esito della funzione list è la file list del Server.

Il comando exit permette la terminazione della comunicazione.

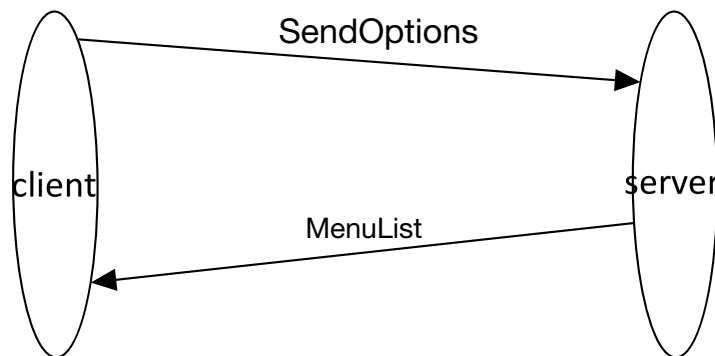
Dettagli Progettuali

Ogni messaggio scambiato ha un header di verifica composto da un numero e da dei caratteri di spaziatura("*****"),

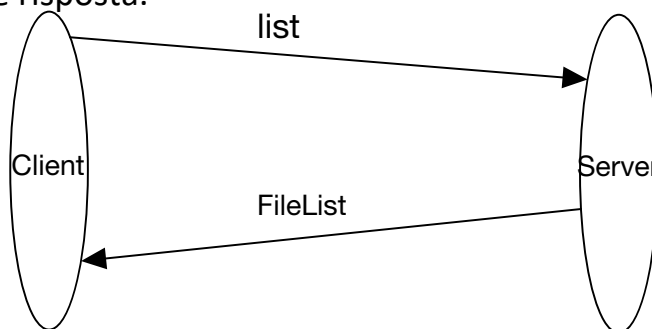
il client invia richieste che iniziano col numero 0 ed il server risponderà con messaggi aventi come primo carattere un 1.

l'architettura nel livello trasporto usa il protocollo UDP che garantisce rapidità a discapito dell'affidabilità, in quanto connection-less e priva di controlli presenti in protocolli affidabili come TCP.

Il client come primo messaggio invia una richiesta delle azioni disponibili, il server risponderà con una stringa contenente il menu come osservabile nell'immagine sopra. Questo sarà possibile consultarlo nuovamente inviando il comando "menu".

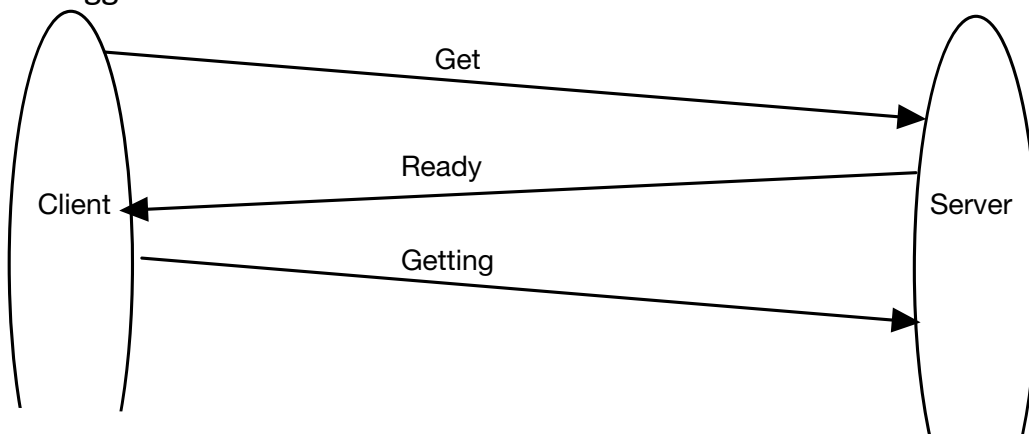


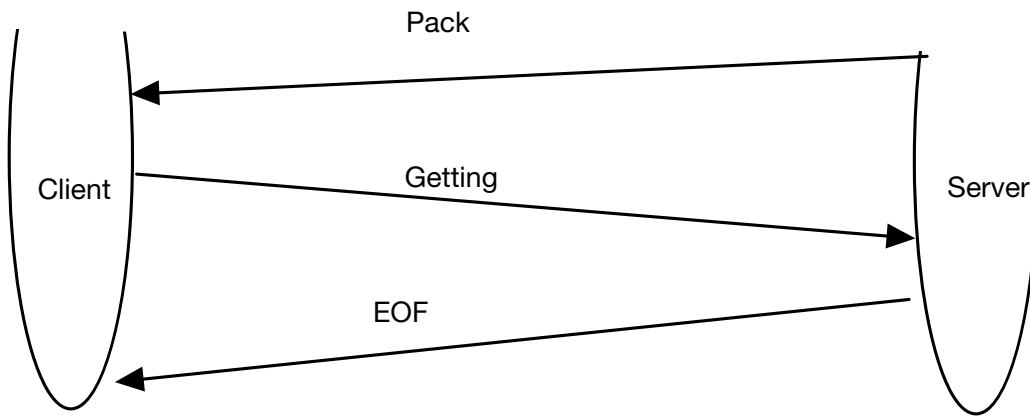
Dopo che il comando "list" è stato inviato da Client a Server, questo prepara la file list con un ciclo for, andando a scrivere in una lista i nomi dei file e la rispedisce al client come risposta.



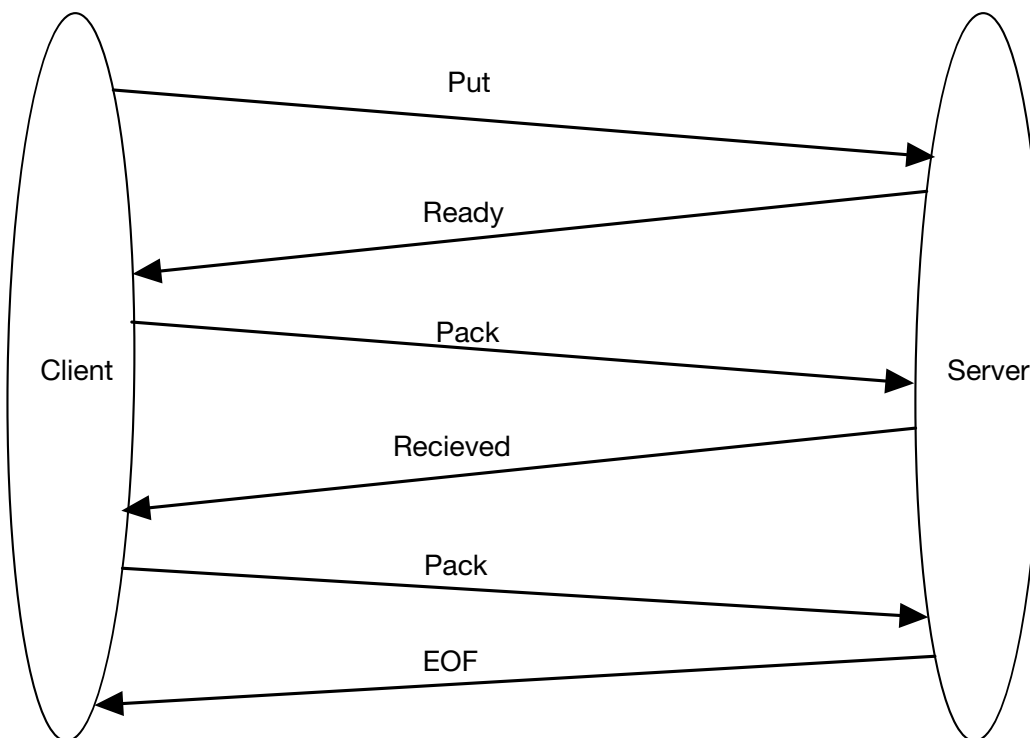
Per il comando "get" come nelle altre funzioni nel primo messaggio scambiato viene preparato il server all'invio di un file, dopo aver ricevuto la conferma, il Client manda il nome del file richiesto, qualora non esistesse viene notificato al mittente, altrimenti manda la conferma.

Il Client poi invierà un messaggio per la ricezione di ciascun pacchetto che riceverà in risposta finché non ottiene 'EOF'. In conclusione all'operazione viene scambiato un messaggio di corretto funzionamento.





Il comando "put" analogamente al "get" avvia l'invio di pacchetti al server previa verifica che il file esista nella directory del Client.
Questa volta saranno i messaggi del mittente che invieranno pacchetti ed il Server risponderà con l'esito dello scambio. Anche in questo caso c'è uno scambio finale riguardante l'esito positivo dell'intera operazione.



Ricevuto il comando "exit" il Server procede con la chiusura del socket e la corretta uscita dagli script da entrambi i lati.

