Programmazione di Reti
2021/2022
Relazione progetto

Traccia n.2: Architettura client-server UDP per trasferimento file

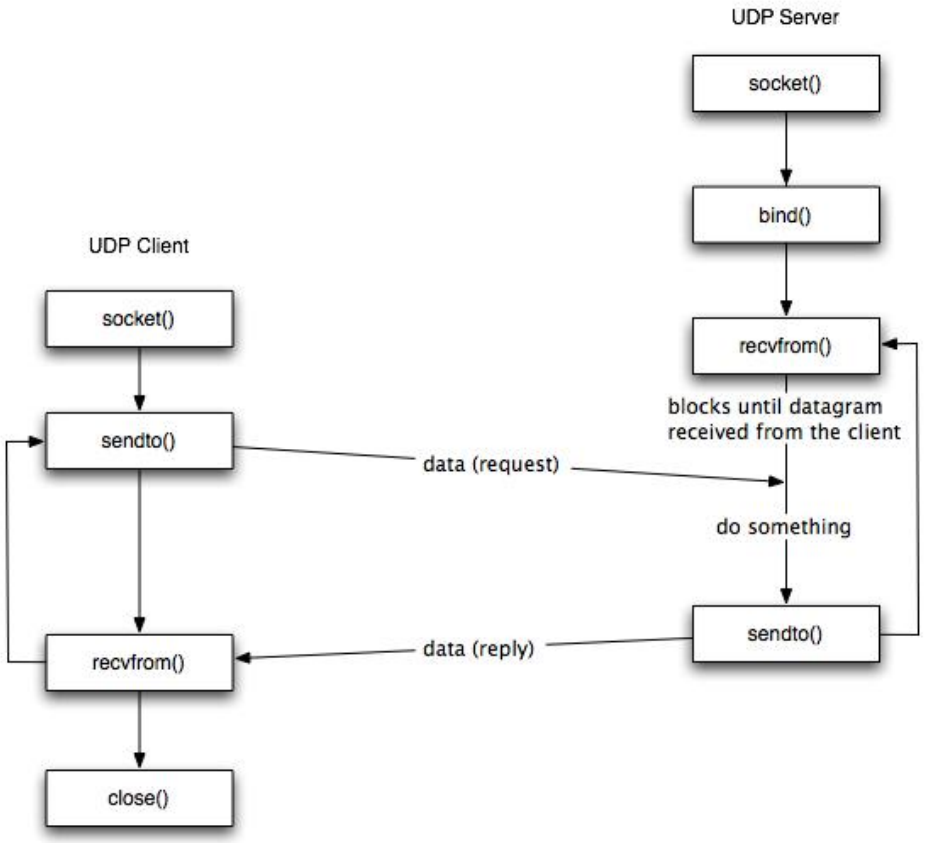
Studente: Davide Tonelli

Indirizzo email: davide.tonelli8@studio.unibo.it

Matricola: 0000970464

Descrizione:

Architettura:



La realizzazione di questo progetto si basa su due figure:

- Server
- Client

Per questo motivo la struttura è composta da due directory, chiamate client e server.
Ogni directory è composta da:

- Un modulo, contenente le implementazioni dei metodi principali utilizzati
- Uno script

Si è scelto di implementare i metodi in un modulo separato dallo script per mantenere una buona modularità, nell'ottica di un progetto futuro che va mantenuto ed aggiornato.

Funzionamento:

Client e server comunicano attraverso una serie di messaggi. Tali messaggi presentano un layout realizzato appositamente per il progetto.

es. lato client
Viene ottenuta la seguente richiesta:

get file1.txt, file2.txt

Il client la traduce rispettando lo standard sopracitato e la invia al server:

GET.!.file1.txt,file2.txt

Dove .!. viene chiamato SEPARATOR.
Questo ha permesso di trattare ogni richiesta in modo efficace, ottenendo immediatamente l'operazione richiesta (GET), una "lista" degli argomenti e soprattutto lasciando la possibilità di aggiungere in futuro nuovi comandi e varianti di ogni comando.

Per ogni richiesta del client, il server crea un thread apposito che la analizza e svolge. Ogni volta che un thread viene creato si richiama join().

Descrizione metodi:

list

Viene ottenuta la seguente richiesta dal client:

list

Tradotta dal client in:

LIST.!.

Ed inviata al server che prima controlla che sia valida, aggiungendo come prefisso della risposta un ‘NOTDONE.!.’ + un messaggio descrivente l’errore per un esito negativo e ‘DONE.!.’ per uno positivo.
Nel caso positivo, richiama **os.listdir(**posizione corrente**)**, salvandone il risultato in una stringa, composta dal nome di ogni file separato da ‘\n’.
Il server invia la risposta al client.
Il client come prima cosa analizza l’esito della risposta, in caso negativo stampa un avviso di errore, in caso positivo il risultato.

get

Viene ottenuta la seguente richiesta dal client:

get file1.txt, file2.txt

Tradotta dal client in:

GET.!.file1.txt,file2.txt

Il client scompone tale richiesta in sotto-richieste e le invia al server:

GET.!.file1.txt

GET.!.file2.txt

Per ogni argomento, si verifica che sia presente e sia un file (sempre con invio di un messaggio di errore per esito negativo) ed in seguito si inviano al client tanti “pezzi” del file di grandezza BUFFER_SIZE (una “costante” predefinita) fino alla sua fine, in seguito invia un SEPARATOR per avvisare il client del termine del file.
Il client quindi mostra all’utente l’esito dell’operazione.

put

Viene ottenuta la seguente richiesta dal client:

put file1.txt, file2.txt

Tradotta dal client in:

PUT.!.file1.txt,file2.txt

Il client scompone tale richiesta in sotto-richieste, controlla la validità di ogni argomento (con stampa di un messaggio di errore per esito negativo) e le invia al server:

PUT.!.file1.txt

PUT.!.file2.txt

Il server è pronto a ricevere dal client tanti “pezzi”, di grandezza BUFFER_SIZE (una “costante” predefinita), del file fino alla sua fine. Il client in seguito invia un SEPARATOR per avvisare il server del termine del file.
Il client quindi mostra all’utente l’esito dell’operazione.

quit

Comando utilizzato per terminare l’esecuzione dello script.
Presenta due varianti:

- **quit**, chiude il socket del client e ne termina l’esecuzione dello script.
- **quit all**, ordina al server di notificare al client la sua terminazione, chiude il socket e termina l’esecuzione dello script. Infine chiude il socket del client e ne termina l’esecuzione dello script.

Ipotesi di progetto:

- I file possono avere dimensione arbitraria
- Il trasferimento di file (get, put) è valido solo per file testuali (no audio, no video, no immagini)
- Un comando può avere un numero di argomenti arbitrario
- L’esecuzione dei due script (client.py e server.py) riguarda solo le rispettive directory correnti