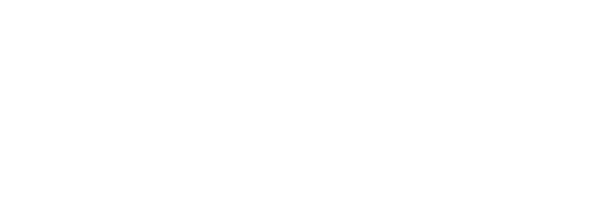
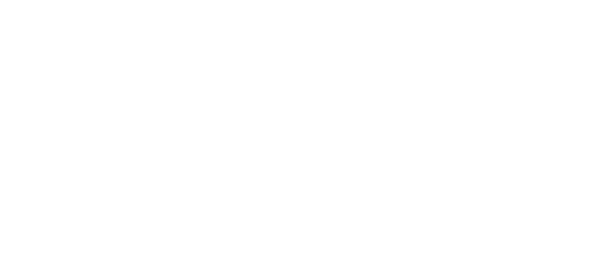


RELAZIONE ELABORATO



PROGRAMMAZIONE DI RETI



Studente: Buzi Sajmir

Traccia nr.2

INDICE

1. Analisi del problema…………………………………………...
2. Design …………………………………………………………
3. Guida d’uso……………………………………………………

1.Analisi del problema.

Il trasferimento di file in UDP è un semplice software che in grado di scambiare attraverso il protocollo UDP dei file attraverso un’architettura Client-Server.

I comandi principali sono del server sono:

-Put: fa l’upload del file nel server.

-Get: fa il download del file da server a client.

-List: fa vedere la lista dei file del server.

Ovviamente ci sarà bisogno di un collegamento tra server e client attraverso una porta prescelta.

Saranno necessari, inoltre, dei messaggi di interazione tra server e client nel momento in cui viene eseguita un’operazione e nel caso di errore mandare in uscita dei messaggi di errore oppure la terminazione del programma in alcuni casi.

2.Design.

L’applicazione come detto in precedenza usa il protocollo UDP, ed è designato in maniera tale che nel caso si possono aggiungere nuove funzionalità nel caso fosse necessario

La richiesta del client viene fatta con un semplice funzione chiamata

-**sendMessage (message, sAddr)**

Alla funzione passiamo il messaggio che nel programma saranno i messaggi provenienti da list, get, put, menu ed exit ed il socket address.

Subito dopo bisogna aprire la generazione del socket attraverso il comando:

- **sock = sk.socket(sk.AF\_INET, sk.SOCK\_DGRAM)**

Subito dopo avviene l’invio del messaggio tramite il socket con il seguente comando:

- **sock.sendto(msgEn, sAddr)**

Il socket aspetterà di recevere una risposta,questo aspetto verrà gestito tramite un timeout preimpostato;

-**sock.settimeout(6)**

infatti, se non avverrà nessuna risposta entro un certo range temporale scatta un’eccezione nel programma che ci fa uscire dal programma con la scritta in output

-**print("\nAn Error Occurred with the Server connection").**

La funzione **checkArgv(arg)** controlla il numero di argomenti che deve essere 3, mentre la funzione **checkPort(num)** controlla la porta di ascolto che deve essere in un certo range sempre preimpostato.

La funzione get , **get(fName, sAddr)** prende come dati il nome del file ed il suo server address ed attraverso la funzione **sendMessagge** avviene il download del file da server, ovviamente prima di tutto bisogna aprire il file con la funzione **open** e infine chiuderlo con la funzione **close.**

Nella funzione put **put(fName, sAddr),** avviene la stessa cosa solamente che il file viene aperto ,poi letto e uploadato nel server sempre attraverso **sendMessagge** infine chiudiamo il file.

Con la funzione **ServerList()** aggiungiamo i file in lista ,se il file non è l primo lo aggiungiamo in coda con il comando **lists.append(file).**

3.Guida d’uso

Per prima cosa bisogna aprire 2 terminali uno per il server ed uno per i client Immagine che contiene testo, screenshot, elettronico, schermo

Descrizione generata automaticamente

Utilizzare il comando python sia per il client che per server per avviare il programma, successivamente il programma ci chiederà di immettere il la porta e l’host, noi useremo il nostro indirizzo IP oppure in maniera più semplice possiamo scrivere localhost , infatti di default avverrà il collegamento con l’indirizzo IP del pc.

Appena lanciato il programma nei due terminali la prima schermata visualizzata sarà il menu gestionale per gestire l’inserimento la visualizzazione ecc..

Immagine che contiene testo

Descrizione generata automaticamente

Ad ogni comando, dalla parte del server ci sarà una risposta contenente il nome del comando ed un controllo di sicurezza del server.

Un esempio è raffigurato in figura:

Immagine che contiene testo

Descrizione generata automaticamente

A questo punto dopo aver visualizzato il menu principale possiamo passare all’utilizzo dei comandi del programma.

Digitando il comando **list** visualizzeremo in output tutti i file in caso contrario altrimenti non visualizzeremo niente.

Nel mio caso digitando il comando list possiamo vedere i seguenti file casuali che ho creato per la prova del programma.



Attraverso l’utilizzo di questi file di prova sono riuscito a fare le verifiche necessarie.

Nel caso volessimo gestire il **put** dobbiamo scrivere l’apposito comando nel terminale seguito dal nome di un file presente nella lista dei file (in questo caso il file prova1.txt), dalla parte del server come sempre ci sarà una risposta dove visualizzeremo il comando che è stato eseguito.



Nel caso del **get** avverrà la stessa cosa ma totalmente al contrario ovvero il file verrà spostato nel client, in questo caso prendiamo in considerazione il file prova1.txt che deve essere presente nella lista altrimenti il programma ci avviserà con un messaggio in output che il file non esiste.



Infine, l’ultimo comando rimasto è quello dell’uscita dall’programma attraverso il comando **exit**. Attraverso questo comando il programma termina da tutti i due terminali.

Questo era il funzionamento del programma Client-Server UDP attraverso poche righe di codice si riesce a gestire molti file di dimensione variabile con grande velocità a discapito però dell’affidabilità.