Relazione Progetto di programmazione concorrente e distribuita

Quaglio Davide
matricola 1026451

June 13, 2014

1 Abstract

Lo scopo di questa relazione è semplificare la lettura e la comprensione del codice del progetto di programmazione concorrente e distribuitaandando ad analizzare le varie scelte progettuali che hanno portato al completamento del progetto.

2 Parte Server

Per la realizzazione del server si è deciso di realizzare prima di tutto un' interfaccia RServer(remote server) che estende il tipo Remote, qui vengono definiti tutti i metodi che verranno invocati in modo remoto dai client o da altri server, per rendere possibile ciò la dichiarazione di ogni metodo indica il possibile sollevamento dell' eccezione *RemoteException*. Successivamente si è creato un oggetto Server il quale implementa RServer ed estende *java.rmi.server.UnicastRemoteObject* per supportare la comunicazione point-to-point usando il protocollo TCP. Tra i vari attributi di server si noti due oggetti di tipo Object, *lockserver* e *lockclient*, tali oggetti permettono ai vari metodi che operano con listaclient o listaserver, la correttezza sui dati su cui opereranno, in quanto prima di utilizzare questi due tipi di oggetti bisognerà prendere il lock sul rispettivo oggetto, andando ad assicurare l' impossibilità di effettuare letture sporche dei dati o altri problemi dati dai problemi di interferenza tra Thread.

2.1 Come mantenere la lista di Server connessi alla rete

Per fare ciò, si è creato un Thread all' interno di Server chiamato *connettiserver*, il quale andrà a prendere il lock su lockserver, pulirà listaserver e poi analizzerà l intero rmi salvandosi in listaserver tutti i server che trova, in questo modo ogni server non più presente non sarà salvato e verranno aggiunti quelli che si connetteranno successivamente al sistema.

2.2 Come mantenere la lista di Client connessi al server

In quanto non è possibile inserire in nessun registro RMI i riferimenti agli RClient, il server non fa altro che salvarsi in *listaclient* i riferimenti agli RClient(RemoteClient), tale lista verrà modificata quando:

- un client si disconnetterà(sia per aver premuto il tasto disconnetti che per aver chiuso la gui) invocando così il metodo disconnettiClient, il quale prendendo il lock su lockclient andrà a rimuovere da listaclient il client disconnesso;
- quando un server, tramite il metodo gotresource, invoca su ogni client il metodo haveresource per contollare se il client ha la risorsa cercata, ma se il client non è raggiungibile, al catch dell' eccezione si rimuove il client da listaclient;

2.3 Come mantenere l' elenco egli altri server che compongono il sistema

Per fare ciò si è creata una classe connettiserver che estende Thread, il metodo run di questa classe continua ad analizzare rmi tramite la chiamata a Naming.list, aggiornando ogni volta la lista dei server registrati al sistema, questo Thread prima di effettuare le operazioni prende il lock su lockserver per evitare letture sporche su tale variabile.

3 Parte Client

Per la realizzazione del client si è creata innanzitutto un' interfaccia RClient contenente tutti i metodi che verranno invocati in modo remoto e per questo estende Remote, tale interfaccia sarà implementata da tutti gli oggetti Client.

3.1 Tenere traccia delle proprie risorse e di quanti e quali client hanno scaricato le risorse

Per quanto riguarda questo aspetto, ogniqualvolta che il client scarica una risorsa, la aggiunge al suo vector di risorse completate che possiede, quando concede ad un altro client una risorsa, un messaggio di log viene inserito nella gui, in questo modo è possibile tener traccia di chi e quale risorsa sta scaricando.

4 Come viene gestito il download di una risorsa dalla parte client

1. L' utente inserisce la risorsa che sta cercando nell' apposito spazio, se tale risorsa non rispetta i criteri di ricerca viene fornito un messaggio d' errore, un' altro messaggio d' errore viene fornito qualora un utente cerchi di scaricare una risorsa che già possiede e non è possibile scaricare se si sta già scaricando.
2. se le precedenti condizioni sono soddisfatte, viene creato un oggetto di tipo Download, che estende Thread e fatto partire; Il metodo run di questo oggetto eseguirà le seguenti operazioni:
    - Se il riferimento al server non è null, chiederà al server la lista di client che possiedono tale risorsa;
    - ottenuta la lista di RClient da cui poter scaricare, si andrà a popolare un Vector di tipo ScaricaRisorsa chiamato downloads, tale vettore sarà popolato in modo che ogni oggetto del suddetto rappresenta un Thread pronto ad

essere eseguito che sarà incaricato di scaricare una parte di risorsa da un Client;

- Tramite l' utilizzo di alcuni oggetti, come partiscaricate che contiene il numero di partiscaricate correttamente, parti contenente il numero di parti da scaricare,download che è la capacità massima di download e downloadattivi che contiene il numero di download in esecuzione si gestirà il download delle varie risorse;
- si utilizza un ciclo while che permette di continuare a cercare d scaricare fino a che non si siano scaricate tutte le parti oppure fino a che non si hanno più client da cui scaricare e download attivi.
- dopo aver sincronizzato sull' oggetto attuale, si fanno partire tutti i download possibili, quindi non più delle parti che si devono scaricare e non più della capacità di download.
- al termine si stampa se si è riusciti a scaricare o meno la risorsa.

3. Ogni oggetto di tipo ScaricaRisorsa, avrà un riferimento al client che lo esegue, e un riferimento al client da cui scaricare, il suo metodo run si occupa di aggiornare la gui del client, successivamente tenta di effettuare il download invocando il metodo upload sul client da cui scaricare, se non ci sono stati problemi le parti scaricate vengono incrementate di uno e in quanto il download è finito con successo con questo client, si riaggiunge al vettore di download possibili un nuovo oggetto ScaricaRisorsa su tale client; sia che il download sia terminato con successo che con errori il numero di downloadattivi viene decrementato di 1. Queste operazioni vengono effettuate mantenendo la sincronizzazione con il client che esegue il download.

5 Come viene gestito il download di una risorsa dalla parte Server

Il server dovrà soltanto fornire al client che glielo chiede la lista di client iscritta alla rete che hanno la risorsa che cerca; per fare questo è stato creato il metodo cercarisorsa, questo metodo si sincronizza su lockserver, in quanto lavorerà sulla lista server; per ogni server della rete, anche se stesso, invocherà il metodo gotresource, tale metodo dopo essersi sincronizzato su lockclient invocherà su ogni client iscritto al server il metodo haveresource, il quale ritornerà il riferimento al client se tale client contiene la risorsa cercata. Una volta invocato il metodo su tutti i client ritornerà la lista di client con la risorsa al server che gliela ha chiesta.In questo modo il server che cerca i client con la risorsa otterrà la lista che verrà restituita al client.