Simulazione di un sistema P2P di scambio risorse

Programmazione Concorrente e Distribuita A.s. 2013/2014

Relazione di Progetto

Alberto Garbui – Matricola: 561226

1 Introduzione

L'intero progetto è stato realizzato utilizzando il design pattern MVC che prevede quindi la suddivisione logica dei dati (model) dal codice principale che svolge le attività di base (controller) e dall'interfaccia grafica dell'utente (view). Questo permette di modularizzare il codice oltre che a renderlo più chiaro, mantenibile e facilmente riutilizzabile.

2 Packages

Sono presenti tre packages: server, client e common.

2.1 Package Server

Il package Server contiene l'interfaccia IServer.java (contenente la dichiarazione di tutti i metodi remoti del server) e la struttura MVC relativa alla parte server del progetto comprendente il modello (ServerModel.java), la vista (ServerView.java) ed il controller (ServerController.java) che implementa i metodi remoti dell'interfaccia IServer.

2.2 Package Client

Il package Client, analogamente al package server, contiene l'interfaccia IClient.java (contenente la dichiarazione di tutti i metodi remoti del client) e la struttura MVC relativa alla parte client del progetto comprendente il modello (ClientModel.java), la vista (ClientView.java) ed il controller (ClientController.java) che implementa i metodi remoti dell'interfaccia IClient.

2.3 Package Common

Il package Common contiene classi che definiscono oggetti comuni ai due package precedenti server e client. Essi sono DeviceClient, DeviceServer e Resource. DeviceClient e DeviceServer definiscono rispettivamente oggetti client ed oggetti server mentre la classe Resource definisce l'oggetto risorsa.
Tutti questi oggetti comuni contengono campi dati propri secondo le specifiche del progetto e vari metodi per accedere ad essi.

3 Funzionalità Client

Ogni client dopo essere stato avviato, inizializza il proprio modello MVC ed avvia un thread in background (*CheckConnectionsThread*) che tenta di connettersi automaticamente al proprio server, in caso negativo o per problemi di rete attende la disponibilità del server. Una volta connesso al proprio server il thread *CheckConnectionsThread* controlla inoltre a determinati intervalli di tempo la presenza del proprio server e gestisce eventuali disconnessioni dello stesso e/o problemi di rete comunicandolo all'utente.

Il pulsante connetti/disconnetti permette all'utente forzare la disconnessione o connessione al proprio server mentre il pulsante ricerca e scarica permette di inviare una richiesta di ricerca risorsa al server ed una volta ricevuta una risposta positiva inserisce la risorsa cercata in coda download.

Il thread *DownloadManagerThread* gestisce la coda download, il suo compito principale è tentare di svuotare la coda limitatamente ai limiti imposti (capacità di download e clients disponibili che possiedono la risorsa). A sua volta ogni singolo thread *DownloadResourcePartThread* gestisce il download di una singola parte di risorsa presente in coda download.

Una volta terminato il download la risorsa completa viene spostata nella lista di risorse complete, mentre in caso di fallimento il download viene interrotto non permettendo un successivo riavvio.

In *ClientController.java* è possibile settare il tempo di simulazione del download (DOWNLOAD_TIMEOUT) settato di default a 7 secondi.

Ogni client che riceve una richiesta di upload apre un relativo thread di upload (*UploadResourcePartThread*) che simula l'upload della risorsa mediante uno sleep() e successivamente richiama un metodo remoto al client che ha eseguito la richiesta per confermare la terminazione dell'operazione.

4 Funzionalità Server

Ogni server dopo essere stato avviato ed aver inizializzato il proprio modello MVC, cerca di comunicare con il registro RMI per pubblicare il proprio nome e prelevare la lista di server attualmente connessi utilizzando una determinata path (rif. RMITAG su ServerController.java). Una volta prelevata la lista di server, tenta la connessione con tutti i server online.
Da questo momento in poi il server resta in attesa di richieste di connessione/disconnessione/ricerca da possibili clients mentre un thread in background (*CheckConnectionsThread*) controlla tutti i dispositivi connessi sia server che client per gestire eventuali disconnessioni, problemi di rete o del registro RMI e notificarlo all'utente.
In caso di problemi al registro RMI vista la gravità dell'errore il server verrà terminato automaticamete allo scadere di un determinato timeout impostabile a piacere.

Le richieste di ricerca di una risorsa vengono gestire concorrentemente aprendo un thread per ogni ricerca richiesta sia dai client sia inoltrate da altri server.

5 Concorrenza

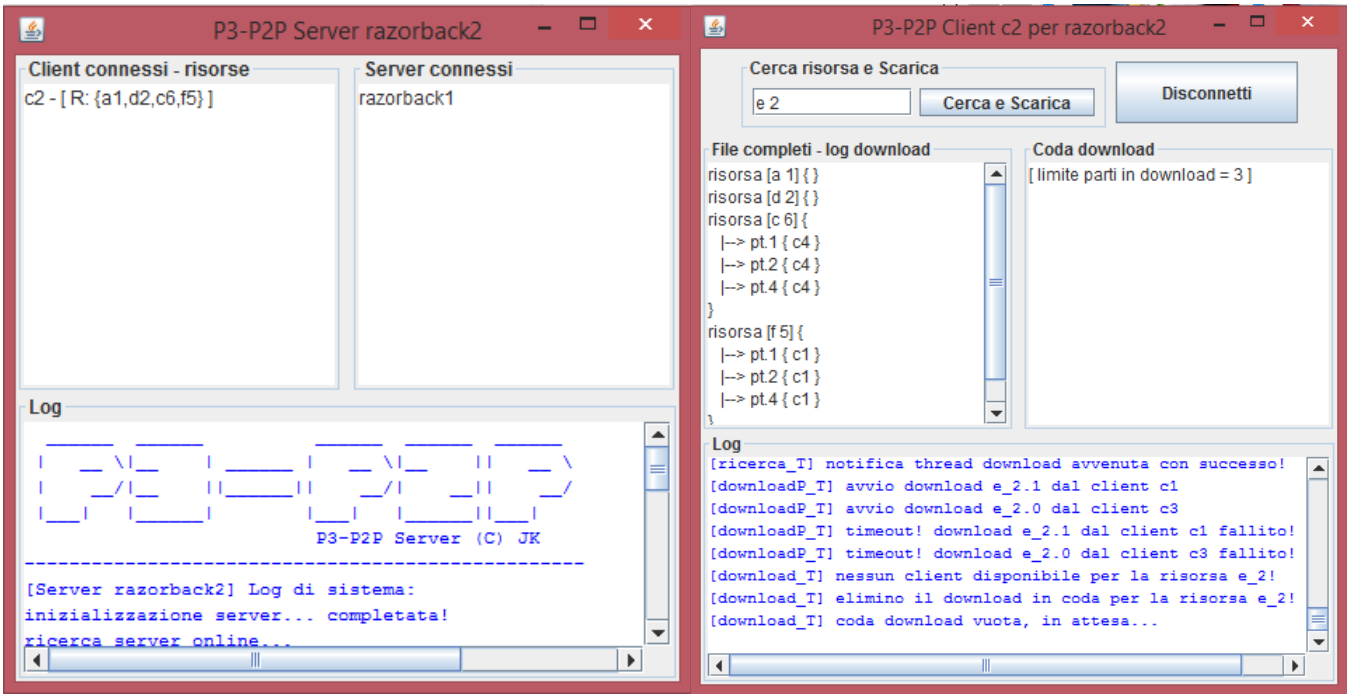
L'utilizzo di thread diversi per ogni diversa operazione sia lato client che lato server permette di effettuare ricerche, download, upload, connessione e disconnessione in modo concorrente. Per garantire la robustezza del codice sono stati utilizzati lock espliciti nelle sezioni critiche del codice dove più thread possono avere accesso ai dati per leggerli e/o modificarli ovvero la parte model del pattern MVC (ServerModel.java e ClientModel.java).

Inoltre è stata usata particolare attenzione nell'evitare problemi di concorrenza relative al thread di download utilizzando dei lock espliciti nella sezione critica dove vengono lette le risorse in coda di download e scelti i clients dai quali scaricare la parte di risorsa.

6 Interfaccia grafica

L'interfaccia grafica del server presenta una lista di server connessi nella colonna di destra mentre una lista di client e relativa lista di risorse nella colonna di sinistra.

L'interfaccia grafica del client invece presenta nella colonna di destra la coda di risorse in download con lo stato delle varie parti della risorsa mentre nella colonna di sinistra sono presenti le proprie risorse complete, per ogni parte della risorsa è presente una lista di client che hanno richiesto quella parte di risorsa.



Nelle immagini successive si possono vedere alcuni download in corso e le varie limitazioni relative alla capacità di download nell'immagine di sinistra ed al numero di client limitato nell'immagine di destra: