Peer-Review 2: NETWORK

Giulia Rossi, Andrea Rondi, Xin Ye, Samuele Russo

Gruppo GC41

Valutazione del diagramma UML MVC-Network del gruppo GC31.

Lati positivi

I lati positivi che abbiamo riscontrato nell'UML del gruppo 31 sono:

- La gestione della connessione tramite socket, chiara e ben strutturata, che dimostra una buona conoscenza dei principi fondamentali della programmazione di rete.

- La predisposizione di una lista di messaggi sul ConcreteServerRMI, che permette di gestirli sequenzialmente e di recuperarli se il client si disconnette.

Lati negativi

I lati negativi che abbiamo riscontrato nell'UML del gruppo 31 sono:

- L'assenza di un'interfaccia remota del client, che permetta al server di chiamarne i metodi. Così facendo il server deve chiamare setMessage() e notificare al client la presenza di un messaggio, che poi deve recuperarsi con il metodo getMyMessage(). Riteniamo sia meglio permettere al server di inviare direttamente il messaggio al destinatario.

- La carenza di MessageType per gestire eventi quali ping, ack, disconnessioni e tutto ciò che riguarda la connessione. Consigliamo di prendere in considerazione una struttura dei messaggi più complessa, per consentire di trasmettere informazioni rilevanti e difficilmente riconducibili al solo messageType.

- La completa scissione tra server RMI e socket, che condividendo grand parte dei propri metodi potrebbero implementarli da un unico server in grado di gestire tutti i client indipendentemente dal tipo di connessione scelta.

- La presenza di numerosi metodi nell'interfaccia Server_RMI. Nella progettazione di un'interfaccia remota RMI è generalmente una buona pratica definire un numero limitato di metodi che rappresentano le funzionalità principali che il server mette a disposizione per i client. Si potrebbero fornire tramite l'interfaccia i soli metodi necessari allo scambio di messaggi e spostare la complessità all'interno di handler finalizzati alla gestione degli stessi.

Confronto tra le architetture

Dal confronto con l'UML MVC-Network del gruppo GC31 risulta evidente la differenza tra la loro struttura e le nostre scelte implementative. Abbiamo infatti pensato di separare i client rmi e

socket in due classi distinte, entrambi ereditanti la stessa classe astratta Client, e di gestirne le connessioni con un unico server, che si appoggia alle due classi ServerRMI e ServerSocket.
Abbiamo inoltre utilizzato un'interfaccia remota per permettere al server di chiamare i metodi del client e abbiamo fatto ruotare il tutto attorno allo scambio di messaggi più complessi, ricevuti da apposite classi "Connection" e da queste inoltrati a degli handlers.