Peer-Review 2: Network Protocol

Samuele Allegranza, Matteo Arrigo, Lorenzo Battini, Federico Bulloni
Gruppo AM13

4 maggio 2024

Valutazione del protocollo di rete delle classi del gruppo AM22.

1 Lati positivi

La scelta di usare JSON per trasmettere i messaggi in rete è valida e flessibile, per quanto possa risultare più complessa da implementare rispetto a utilizzare la serializzazione di java. Il "problema" risiede nella gestione della serializzazione/deserializzazione usando librerie come `Jackson`, per le quali potrebbe essere necessaria particolare cura sulle regole legate al parsing. A parte questo dettaglio, lo scambio di messaggi JSON è più elegante, avendo pieno controllo sulle informazioni da trasmettere, e flessibile, dato che si potrebbe reimplementare il codice del client o del server in altro linguaggio senza cambiare protocollo. Inoltre le informazioni trasmesse sono in chiaro, utile per eventuali debug.
Per quanto riguarda i messaggi del protocollo, sembrano coprire tutte le varie casistiche per una corretta gestione del flusso di gioco. Ci troviamo in accordo con una buona parte delle scelte implementative su questo fronte.
Anche le catene di chiamate sul server, che si innescano una volta ricevuta una richiesta dal client, sembrano rispettare il modello architetturale composto da controller e model.

2 Lati negativi

Elenchiamo gli aspetti che riteniamo possano essere migliorati riguardo al protocollo di rete:

- Manca un messaggio di richiesta dei `game id` attualmente presenti. Così il client che si vuole unire ad un gioco (che non ha ancora raggiunto il numero di giocatori fissato) non può sapere quale sia il suo `game id`.
- Sembra che non venga mandato il messaggio di risposta che indica che un giocatore si è unito alla partita (nell'UML sono indicate solo le risposte in caso di errore). In questo modo gli altri giocatori non saprebbero il numero attuale di client connessi fino a quando non provano ad inviare il messaggio di *start game setup*.
- Metodi come `SetStartingCard()` e `SetCommonGoals()` sono in realtà un mix tra getter e setter. Inoltre, non è chiaro cosa succeda se un giocatore chiama `SetCommonGoals()` ma le `GoalCard` comuni sono già state settate. Poiché le `GoalCard` comuni sono stabilite una volta sola potrebbe essere conveniente che questo processo sia svolto in automatico dal server.

- Dall'UML non si capisce se i messaggi di risposta alle azioni dei giocatori siano mandati a tutti o solo al giocatore in questione. Se così non fosse, i singoli giocatori potrebbero sapere lo stato di avanzamento del gioco solo tramite refresh manuali (per esempio con il messaggio *want to see other player game*).

- Alcuni metodi restituiscono true/false mentre sarebbe più opportuno utilizzare delle Exception.

- Viene utilizzato un messaggio distinto per le varie informazioni da inviare all'inizio del gioco come `startingCard` e `commonGoals`, anche se di fatto i messaggi sono inviati uno dopo l'altro senza un'effettiva interazione con i giocatori.

- Non sono mostrati casi in cui una richiesta non va a buon fine, e di conseguenza non è chiaro come e da parte di chi vengano gestiti gli errori (per esempio il caso in cui venga giocata una carta in una posizione sbagliata).

- Nel sequence diagram non è stato mostrato un flusso completo di gioco ma solo (nel diagramma a destra) alcune possibili chiamate a metodo sconnesse tra loro.

- Dal secondo UML sembra che non venga salvato nulla in locale (lato client) ma vengano solamente mostrati i risultati dei messaggi ricevuti. Questo complica la gestione dello stato di gioco dal lato di client e aumenta il numero di messaggi mandati in rete.

3 Confronto tra le implementazioni

La struttura e il numero di classi che gestiscono la comunicazione tramite socket è molto simile alla nostra.
In generale ritroviamo in questo sequence diagram alcune scelte implementative prese in considerazione anche da noi, anche se poi le abbiamo scartate, per esempio l'uso del formato JSON per i messaggi da inviare in rete.
Una differenza risiede nella gestione della fase antecedente all'inizio della partita di gioco. Nel nostro progetto infatti:

- La scelta del colore della pedina avviene in fase di creazione/entrata nella *stanza di gioco* assieme alla scelta del nickname.

- la partita inizia automaticamente se e solo se viene raggiunto il numero necessario di player nella *stanza di gioco*.