Peer-Review 2: Sequence Diagrams

Alessandro Amandonico, Francesco Buccoliero,
Kaixi Matteo Chen, Lorenzo Cavallero
Gruppo 10

2 maggio 2023

Valutazione dei sequence diagrams del gruppo `IS23AM19`.

1 Lati positivi

Il diagramma di rete del gruppo 19 è formalmente corretto e astrae correttamente l'interfaccia di rete tra client e server. Il gruppo ha scelto di implementare le funzioni aggiuntive della multi-partita e della resilienza alle disconnessioni. Non è stata riscontrata la presenza di chat o di persistenza. La seguente review considererà le due funzioni come ignorate.

Un lato positivo dell'architettura di rete è la presenza di un meccanismo di login/signup basato su ID e password, che potrebbe essere utile per evitare accessi indesiderati. Non si può non notare tuttavia il fatto che in assenza di persistenza ad ogni avvio del gioco occorre registrarsi nuovamente, rendendo poco utile la funzionalità.

Le funzionalità di creazione di una nuova partita o di join a una partita esistente sembrano gestite correttamente a livello di rete; si nota l'interessante aggiunta del parametro `searchID` che richiede l'identificativo di un giocatore già in partita, per unirsi alla stessa partita. Questa aggiunta, tuttavia, richiede quindi di mantenere l'univocità dell'username fra più partite.

Si può notare come il server invia al client in modo differenziale gli oggetti aggiornati (e.g. `sendPlayersList, sendMainBoard...`) velocizzando così le operazioni, tuttavia, tale approccio può portare a possibile inconsistenza del model.

Si segnala inoltre una corretta gestione dei controlli, replicati su server e su client, per evitare di accettare mosse illegali o da parte di un giocatore che non potrebbe fare mosse.

2 Lati negativi

Sebbene ad alto livello l'architettura sia completa e funzionale ci sono alcune note negative da segnalare: la prima e più corposa riguarda l'approccio usato per la suddivisione dei task lato server: il gruppo ha un entrypoint `LobbyServer` che gestisce la parte di login e creazione/join delle partite e poi un `Server` che viene evocato per ogni singola partita, su porte diverse. Il `LobbyServer` ritorna infatti all'utente una porta (o due se implementata la doppia connessione) a cui connettersi. Questo approccio risulta poco scalabile oltre che inutilmente contorto.

Un commento è sollevato anche sulla gestione delle eccezioni, che dal diagramma non risulta di facile comprensione: non è infatti chiaro se le exception vengono lanciate dal server (che quindi crasha?) o se viene gestita e redirezionata al client (che quindi crasha? come la gestisce?).

Infine nella parte di gioco, ci preme segnalare la scelta a nostro avviso sbagliata di delegare la gestione dello stato del model, quindi del gioco, al client, che si porta dietro anche i problemi di "consistency" evidenziati dallo stesso gruppo (*"the server will consider the data on client obsolete so it will send new data to all clients"*). Una soluzione più semplice può essere la gestione dello stato del gioco fatta esclusivamente dal Server, unica "truth source", che si occupa di inviare lo stato del gioco aggiornato ai client. Anche il comando `notifyActivePlayer` potrebbe quindi essere eliminato, con eventuali mosse da player non autorizzati già coperte dai controlli.

3 Confronto tra le architetture

Nel complesso le architetture dei due gruppi risultano simili nella fase di gestione del gioco, con la differenza che il nostro gruppo invia lo stato del gioco nella sua interezza, mentre il gruppo 19 preferisce inviare i singoli *model items*.

La fase di login e connessione, così come la gestione delle partite, è invece molto differente. Il nostro approccio prevede un server unico che gestisce più

partite e più client contemporaneamente, sulle stesse due porte (differenti per RMI e socket), indipendentemente da chi gioca o meno in quale partita. A nostro avviso la flessibilità e la pulizia del codice di questo approccio sono nettamente migliori rispetto a quelle proposte dal gruppo 19.

Si segnalano alcuni spunti di riflessione interessanti, come l'autenticazione via password, che però non intendiamo implementare per via dei motivi sopra elencati, non implementando la persistenza, risulterebbe poco utile aggiungere una funzione di questo tipo.