# Peer-Review 2: Protocollo di rete

Morelli, Morea, Neto Dell'Acqua

Gruppo 28

Valutazione del protocollo di rete del gruppo 27.

## Lati positivi

Il protocollo scelto dal gruppo 27 rientra tra quelli real-time request/response cycle, in cui il server richiede un'interazione al client, il quale provvede ad inoltrare nuovamente la risposta al mittente. Dal nostro punto di vista non solo è una scelta ottima, ma probabilmente era anche l'unica scelta possibile per un gioco turn-based con un'architettura thin client.
L'utilizzo di soli 5 messaggi parametrici, *RequestAction, ResponseAction, RequestValue, ResponseValue, sendInfo*, combinati all'utilizzo della libreria Gson, riduce di molto la complessità rispetto a quella che poteva essere un'implementazione che richiedesse la creazione di un messaggio custom per ogni tipo di interazione, che poi dovesse essere serializzato manualmente. Le scelte progettuali ci sembrano dunque molto ragionevoli e fondate su esigenze implementative concrete, ottimo lavoro.

Anche i sequence diagram sono ben articolati e dimostrano una chiara comprensione di tutti i possibili scenari di interazione client/server, inclusa la compatibilità con la funzionalità avanzata di persistenza della partita.

Interessante il meccanismo di handshake con il client che prevede l'inserimento da parte dell'utente di un numero intero che verrà poi utilizzato come seed pseudorandomico per determinare l'ordine di turno iniziale.

## Lati negativi

Non si sono riscontrate particolari criticità.

## Confronto tra le architetture

I parallelismi tra le due architetture sono molti:

- L'utilizzo della libreria GSON per serializzare e deserializzare i messaggi.
- L'architettura richiesta/risposta.

Per quanto riguarda i messaggi scambiati, noi siamo riusciti a scendere a soli 3 possibili classi di equivalenza di messaggi parametrici: *RequestToMe, RequestToOther, Response*.
*RequestToMe* : Un messaggio di richiesta dal server rivolto proprio a quel client
*RequestToOther* : Un messaggio di richiesta del server rivolto ad un altro client
*Response* : Un messaggio di risposta di un altro client al server
Le richieste che invia il server saranno richieste di interazione con il terminale ad un singolo client, tutti gli altri client vedono quella richiesta e capiscono se è rivolta a loro, se non è rivolta a loro gestiranno il tempo di attesa con una frase, ad esempio "Il giocatore xyz sta scegliendo abc".
Appena il client interpellato invia la risposta (i dati che ha scritto su terminale), anche tutti gli altri client vedono questo messaggio, e mostreranno a schermo un' interpretazione della risposta, ad esempio "il giocatore xyz ha scelto abc".

Sostanzialmente dopo la fase di connessione e creazione della partita il client diventa un *dumb terminal* che reagisce solo a queste 3 tipologie di messaggi e chiamerà uno di questi 3 metodi della view (parametrici rispetto al tipo di messaggio ricevuto):
void requestToMe(); void requestToOthers(); void response()