Peer-Review 2: Protocollo di Comunicazione

Longo, Molè, Negro.

Gruppo 3

Valutazione del protocollo di comunicazione del gruppo 2.

Lati positivi

- Il `Ping` è un'ottimo modo per controllare che la connessione sia persistente.
- Il format degli eventi è modulare, ad esempio `1.7.3` permette di rappresentare una moltitudine di eventi.
- Il messaggio di `Status` è un modo appropriato per ripristinare la sessione nel caso in cui il Player perda la connessione.

Lati negativi

- Non è chiaro se il messaggio di `Error` possa essere inviato sia dal Server al Client che viceversa e se diversi tipi di errori prevedono azioni differenti.
- `Game creation` dovrebbe far entrare implicitamente nel game il Player che lo ha creato, senza il bisogno di scambiare altri messaggi tra Client e Server.
- Il `login` del Player dovrebbe avvenire prima, subito dopo l'avvenuta connessione, in modo da poter anche partecipare a più partite con lo stesso nickname.
- `Turn enable` potrebbe essere un messaggio mandato in broadcast che comunica a ogni Player di chi è il turno.
- `Move professor` dovrebbe essere fatto automaticamente dal server, e non richiesto dal Client, a maggior ragione trattandosi di un messaggio senza risposta.
- `Move tower` dovrebbe essere un'azione del Server.
- Non vi è esplicita validazione dei comandi del Client, cosa necessaria soprattutto nel caso in cui compia un'azione non consentita.
- In `End game` non è possibile comunicare la vittoria di più giocatori, come nel caso di partita a squadre.
- Manca la rappresentazione dei possibili scenari in cui il Client seleziona una Character Card.

Confronto tra le architetture

- A differenza del gruppo 2, abbiamo preferito rendere espliciti gli `Ack`.
- C'è una corrispondenza tra il loro `Error` e il nostro `Nack`, ma quest'ultimo è più specifico e inviato solo dal Server al Client e mai viceversa.
- Potremmo ispirarci al loro `Ping` per le connessioni persistenti.
- Abbiamo preferito far fare all'utente un unico login, per non costringerlo ad autenticarsi ogni qual volta decida di accedere ad una partita.

- Nel nostro protocollo i `Commands` inviati dal Client sono distinti dalla risposta del Server, che aggiorna il Model per tutti i Client contemporaneamente. Apprezziamo la loro scelta di evitare ridondanze di messaggi, omologando il modo in cui il Client fa le richieste e il Server vi risponde.