Peer-Review 2: Protocollo di Comunicazione

Lorenzo Franzè, Valentina Moretti, Lara Nonino
Gruppo 22

Valutazione della documentazione del protocollo di comunicazione del gruppo 21.

Lati positivi

- I messaggi scambiati sono molto specifici: questo permette un maggiore e più frequente controllo del gioco da parte del server. La connessione, per esempio, non viene effettuata con un unico messaggio che specifica il numero di giocatori, la modalità di gioco ed il nickname, ma da più messaggi singoli. In questo modo, nel caso dovesse sorgere un qualsiasi tipo di errore, è possibile avvisare il giocatore dopo ogni messaggio ricevuto.
- I giocatori hanno molte possibilità di scelta nella fase di "inizializzazione" del gioco. Ad ogni utente è resa possibile la scelta del proprio team, del colore delle proprie torri e dei maghi. Poiché queste caratteristiche potevano essere "inizializzate" lato server e quindi assegnate ai giocatori, apprezziamo la scelta fatta. Inoltre, questa decisione rende ancora più complesso lo scambio dei messaggi che però ci sembra stato svolto correttamente.
- Gli aggiornamenti dal server contengono solo le informazioni riguardanti un cambiamento e non l'intero stato di gioco. Per esempio, nel momento in cui un utente deve scegliere una nuvola, gli vengono inviate solo le nuvole rimanenti e non l'intero stato di gioco. Questo rende i messaggi più leggeri e di facile comprensione. Se si fosse optato per la scelta alternativa però si sarebbero potute risparmiare molte classi messaggio per l'aggiornamento dei client.
- I sequence diagrams presentati sono completi ed esaustivi. Per ogni scenario vengono riportati tutti i possibili messaggi scambiati tra gli attori in tutte le casistiche. Non ci è però chiaro quando e come viene scambiato il messaggio di Update. Probabilmente è stato dimenticato il sequence diagram relativo ad esso (vedi lati negativi).

.

Lati negativi

- ConnectionRequest: gli argomenti del messaggio risultano inutili in quanto il server non ha alcun bisogno di ricevere sia il proprio IP che la propria porta, inoltre se è possibile mandare il messaggio allora tali elementi sono già noti a priori sia al client che al server.
- PlayerMaxMoves: il messaggio può essere evitato permettendo al client di salvare tale informazione quando sceglie la carta personaggio e in caso di risposta positiva dal server. Inoltre, il giocatore avendo giocato la carta dovrebbe già sapere quanti passi può fare madre natura, è necessaria solo una verifica lato server nel momento in cui il client comunica al server i passi.
- Non è chiaro l'uso di ModelUpdate: non compare nei sequence diagrams.
  Se supponiamo che serve solo per notificare se ci sono stati cambiamenti, siccome non passa altri parametri, allora perché non usare già gli altri messaggi per notificare al client che sono avvenuti dei cambiamenti? Ad esempio, quando un giocatore sceglie una tessera nuvola tutti i giocatori ricevono già il messaggio SelectCloud da cui "scoprono" sia che qualcosa è cambiato sia cosa è cambiato.

Consigliamo inoltre di mandare tutto lo stato del gioco a ogni update e non solo ciò che cambia, questo ha in vantaggio di semplificare la visualizzazione lato client permettendo di usare un unico metodo ogni volta per mostrare lo stato del gioco, inoltre sempre lato client si evita di ricercare e aggiornare i singoli cambiamenti del gioco ogni volta che avviene un cambiamento. Sebbene i messaggi scambiati risultino più pesanti, occupando maggiormente la banda di gioco, la gestione degli update risulta più semplice.

- Come illustrato dei sequence diagrams è molto spesso il server a iniziare la comunicazione per chiedere all'utente di fare una mossa, il ché semplifica la CLI ma complica la GUI in quanto il client non può effettuare delle mosse di propria volontà ma deve aspettare l'autorizzazione dal server.
- Non è chiaro il motivo per cui il client debba inviare anche un messaggio di ACK alla ricezione di ogni messaggio: la ricezione è garantita grazie al protocollo TCP, inoltre la disconnessione viene già rilevata attraverso i messaggi di PING e PONG.
- Gli ACK ricevuti non hanno nessuna caratteristica che permette di differenziarli gli uni dagli altri. In caso di concorrenza questo potrebbe rappresentare un problema: non saprei a che messaggio si riferisce l'ACK appena ricevuto.

Confronto

- Rispetto al nostro gioco, il gruppo 22 presenta al giocatore più possibilità di scelta e di personalizzazione della partita. Nelle scelte di progetto si era deciso di assegnare i maghi e il colore delle torri in maniera casuale ai giocatori, essendo questi aspetti non rilevanti nelle logiche di gioco. Tuttavia, visto che molti giochi attualmente in commercio rendono disponibili scelte del genere in quanto possono in alcuni casi aumentare il livello di gradimento del gioco da parte dei giocatori, riteniamo una buona scelta rendere possibile questa interazione. Valutiamo, dunque, se aggiungere anche nel nostro gioco la possibilità di scelta dei maghi e del colore delle torri, tenendo conto però che questo aumenta gli scambi di comunicazione tra client e server nei casi in cui le scelte non siano scelte valide o non più disponibili perché prese da altri giocatori.

- Una differenza sostanziale esiste tra la nostra gerarchia di messaggi e quella del gurppo 22: la nostra squadra ha raggruppato diversi tipi di messaggi in una stessa classe di messaggio fornendo una via di distinzione tramite un campo "MessageType". In questo modo abbiamo un numero di classi di messaggio ridotto. Questo è utile per esempio nei messaggi che esprimono una scelta di mossa del giocatore (questa può essere la scelta di una carta assistente, di una tessera nuvola, degli spostamenti di madre natura o degli spostamenti degli studenti). Abbiamo ritenuto comodo e più facile da gestire questo raggruppamento di messaggi e ciò ci ha inoltre permesso di scrivere il codice in maniera più pulita e ordinata. Riteniamo ugualmente valida la scelta del gruppo 22 di creare una classe di messaggio di tipo diverso per ogni tipologia diversa di messaggio. Abbiamo preferito tuttavia mantenere la nostra struttura di messaggi in quanto riteniamo inutile creare classi di tipo di messaggio diverso quando queste hanno gli stessi attributi e non sono sostanzialmente diverse nella logica del gioco. Ad esempio riteniamo comunque sia meglio distinguere i messaggi di gioco da quelli di errore, di ping e di disconnessione. Inoltre, in un'ottica di possibile estendibilità del codice, la nostra scelta ci sembra migliore.

- Nel nostro gioco la verifica della validità della scelta di alcune mosse si trova più sul server rispetto che sul client. Prendiamo come esempio il sequence diagram della scelta della carta assistente. Il nostro gioco verifica se la carta è già stata scelta da un altro giocatre sul

server, quello del gruppo 22 invece sul client. Questo aspetto è evidenziato dallo scambio del messaggio “UsedAssistantCards” nel sequence diagram del gruppo 22. A nostro avviso, le scelte dei due gruppi sono egualmente valide. Per prendere una decisione questi gli aspetti da valutare sono i seguenti: più spostiamo i controlli sul client, più alleggeriamo il lavoro del server, tuttavia più rendiamo i client pesanti e complessi, più gli aggiornamenti e i cambiamenti futuri sono difficili da gestire.