Peer-Review 2: Communication Protocol

Andrea Sanguineti, Emanuele Santoro, Andrea Piras
Gruppo GC-45

4 Maggio 2022

Descrizione del protocollo di comunicazione usato.

Serializzazione e classi

Il protocollo di comunicazione utilizzato per far comunicare client e server sfrutta la serializzazione e la deserializzazione di Java, tramite l'utilizzo di due interfacce ***ToClientMessage*** e ***ToServerMessage***.

Esempio: ogni volta che il client deve comunicare con il server, crea un'istanza di una delle classi che implementano ***ToServerMessage*** specifica dell'operazione contenente i dati relativi alla richiesta, quest'ultima viene poi serializzata e inviata al server.
Il server deserializza tramite l'interfaccia ***ToServerMessage*** capendo così il tipo dinamico del messaggio e chiama l'***execute()*** interna al messaggio.

L'***Execute()*** interna al ***ToServerMessage*** richiede come parametro un ***ClientHandler*** e chiama le funzioni del ***ClientHandler*** o del ***Controller*** adatte. Il server invece risponde con ***ErrorException*** nel caso di errore e di
La classe ***ClientHandler*** è il thread che gestisce la connessione e la comunicazione di un player che gioca.
Per aggiornare invece il client sullo stato del gioco, abbiamo un ***ToClientMessage*** apposito che contiene un ***GameDelta*** (***DeltaUpdate***). Il ***GameDelta*** è un oggetto che viene modificato ogni volta che una qualsiasi funzione del Game modifica i suoi attributi. Quando il ***GameDelta*** cambia, vengono notificati tutti i listener (ovvero i ***ClientHandler***) che inviano un ***ToClientMessage*** contenente il ***GameDelta***. Quando arriva al client, lo deserializza e andrà a modificare il suo model di conseguenza.
Il client e il server inoltre appena viene aperta la connessione iniziano a scambiarci un messaggio di tipo *PING* che permette in caso di disconnessioni di rete di accorgersene e di agire di conseguenza. Vedi figura 2

Tipi Di Messaggi

To Client Messages	
DeltaUpdate	Contiene il GameDelta. Tramite la execute del Client aggiornerà la view
Phase	Contiene la gamePhase che si sta giocando
MatchInfo	Inviato a inizio del gioco, contiene il tipo di gioco
PlayerJoined	Inviato prima che il gioco si inizia quando un giocatore si connette
EndGame	Contiene un array list dei team vincitori o null se c'è stata una disconnessione
ErrorException	Contiene il messaggio di errore dato da una qualsiasi eccezione
Ok	Mesaggio di OK (azione eseguita senza eccezioni)
TextMessageSC	Messaggio di chat ricevuto da un qualsiasi altro player
To Server Messages	
NickName	Contiene il nickname scelto dal giocatore
CreateMatch	Contiene il MatchType della partita da creare
JoinMatchById	Contiene l'id della partita a cui il client si vuole unire
JoinMatchByType	Contiene il MatchType della partita a cui voglio unirmi
PlayCard	Contiene l'id della carta assistente che si vuole giocare
Move	Contiene il colore dello studente e l'id della destinazione di quest'ultimo
ChooseCharacter	Contiene l'id della carta personaggio selezionata
SetCharacterInput	Contiene input richiesto dalla carta personaggio
PlayCharacter	Richiede di giocare la carta personaggio selezionata
MoveMotherNature	Contiene il numero di mosse che si vogliono effettuare
MoveFromCloud	Contiene l'id della nuvola dalla quale pescare studenti
Quit	Il client richiede di disconnettersi, termina il thread
TextMessageCS	Contiene il messaggio che si vuole mandare agli altri giocatori

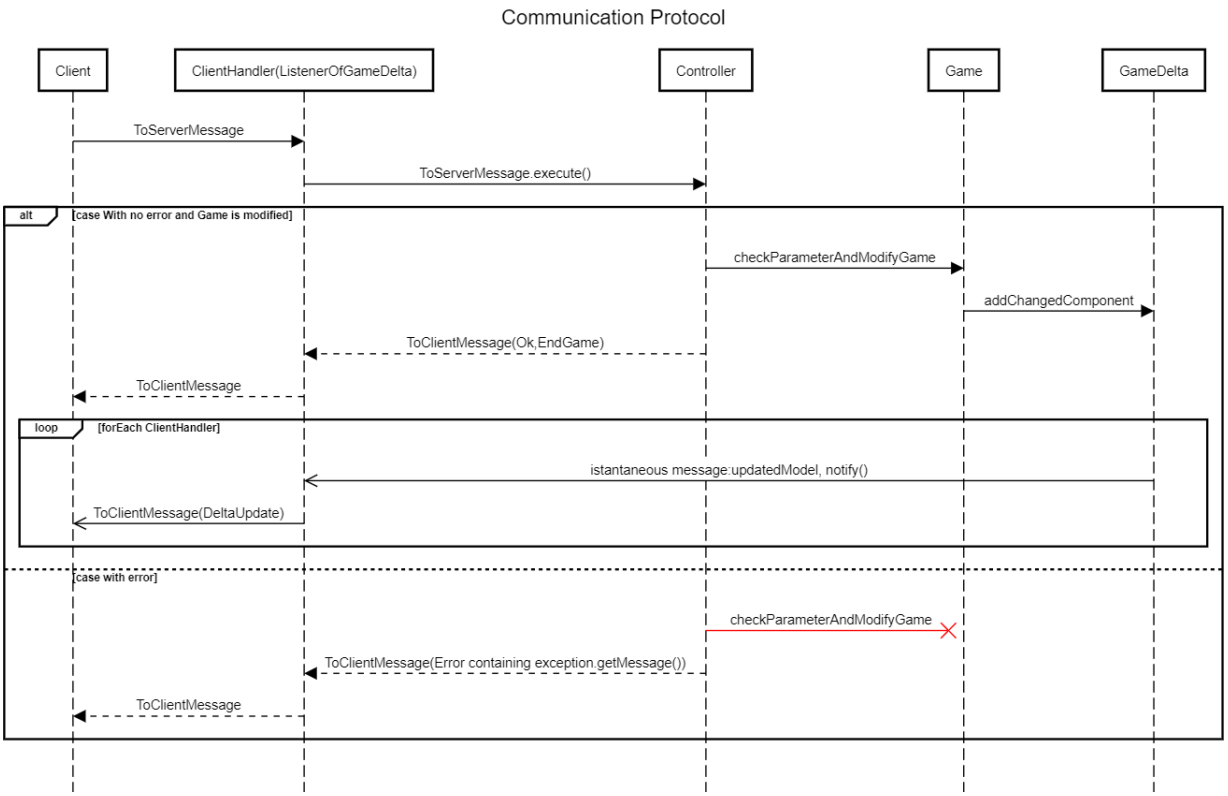

Figura 1: Sequence diagram

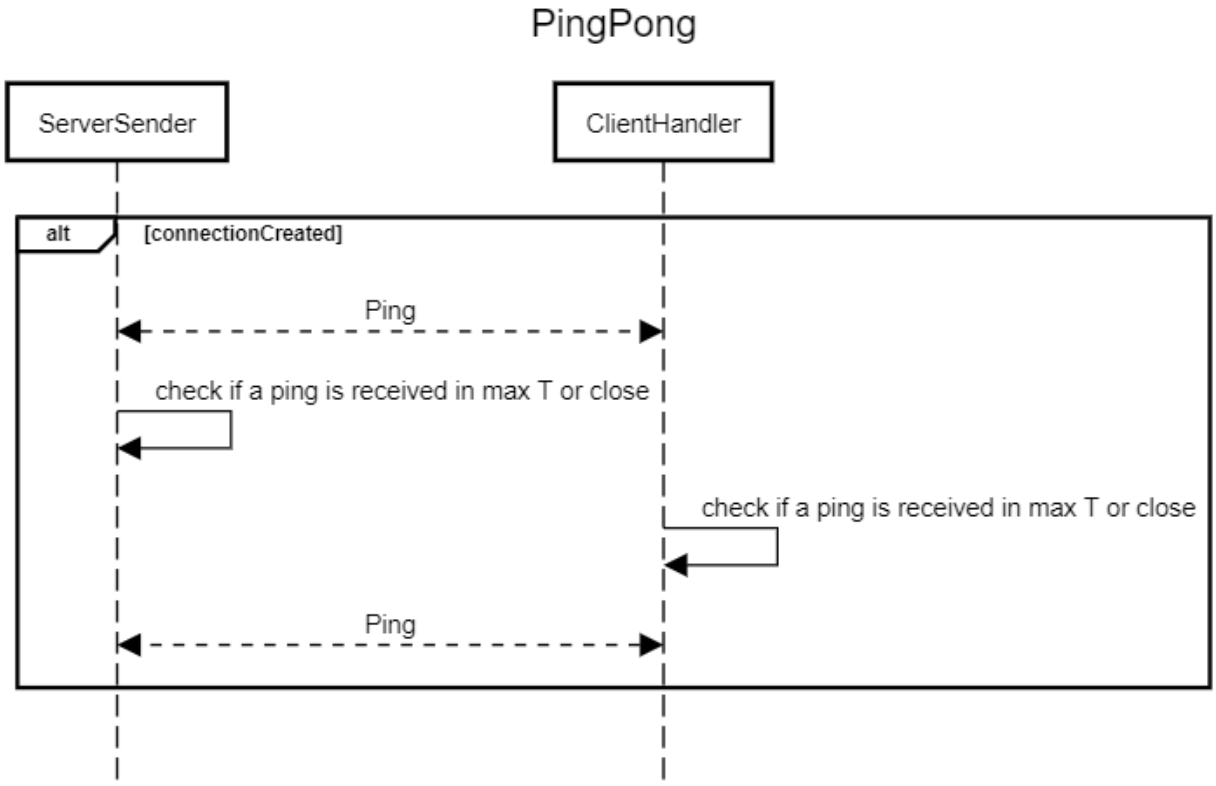

Figura 2: Sequence diagram