Peer-Review 2: Rete e Protocollo

Emanuele Paci, Lorenzo Paleari, Thomas Puppinato
Gruppo 12

3 maggio 2022

Descrizione dell'architettura di rete e del protocollo.

1 Descrizione UML

In allegato alla mail trovate gli UML dettagliati di ogni package, oltre ad un UML completo nel quale si possono vedere le interazioni tra le diverse classi dei package.

1.1 Model

Rispetto all'ultima peer review il model non ha subito sostanziali modifiche. Abbiamo incluso nella rappresentazioni alcuni collegamenti con il controller per facilitarne la comprensione.

1.2 Controller

Il controller è formato da una omonima classe principale. I comandi richiedibili dall'utente sono suddivisi nei vari handler in base a cosa andranno a modificare. Prima di eseguire le azioni viene sempre fatto un controllo per verificare la correttezza e i permessi della richiesta (tramite turnController), ad esempio se non è il tuo turno, non hai abbastanza monete, ...
Nei controller abbiamo anche aggiunto degli strategy pattern, raggruppati nei subpackage che cambiano a runtime le regole del gioco, se necessario, in base alla carta personaggio utilizzata.

1.3 Client

La classe principale Client permette all'utente di scegliere se vuole utilizzare CLI o GUI, entrambe le quali implementano l'interface View in modo da poter utilizzare gli stessi metodi in override.
Dopo aver scelto il tipo di grafica, viene creato il serverHandler che si occuperà della comunicazione con il server.
La classe ackControl manda dei ping ogni 5 secondi al server per verificare che sia ancora attivo.
Abbiamo anche delle classi di appoggio che contengono le informazioni del game serializzate per poter essere trasmesse dal server al client: quando avviene un'azione il game viene convertito nella classe GameInfo, quindi serializzato e inviato al client per essere visualizzato.

1.4 Server

La classe principale Server si occupa di accettare le connessioni al server, passando poi la gestione al LobbyHandler per la scelta della partita a cui partecipare (abbiamo deciso di implementare le partite multiple). Per ogni client connesso verrà creato un ClientHandler, che si occuperà della comunicazione con il client. Tra il controller e il ClientHandler abbiamo la classe VirtualView che simula una view a livello di controller, occupandosi di inviare i comandi da ClientHandler a controller e viceversa.

1.5 Network

Il network si occupa della comunicazione tra client e server, tramite varie classi messaggio. Queste ultime implementano dei tipi di messaggio specifici (LobbyMessage, ViewControllerMessage, ControllerViewMessage, ModelViewMessage).
Abbiamo inoltre un'interfaccia GenericMessage che è estesa dai quattro tipi di messaggio elencati precedentemente (i tipi di messaggio ereditano i metodi di GenericMessage). Questo ci permette di utilizzare la stessa chiamata per tutti i tipi di messaggio sfruttando override e overload degli stessi.
Nell'UML dettagliato allegato mancano tutte le frecce dalle classi messaggio alle interfacce messaggio per una semplice questione di ordine e chiarezza.
Nell'UML completo sono invece presenti e potete osservare quali messaggi implementano ogni specifica interfaccia.

2 Descrizione del protocollo di rete

Per facilitare la descrizione del protocollo di rete da noi adottato, faremo riferimento ai due sequence diagram allegati. Partendo dal caso standard (allegato Eriantys protocollo), all'avvio del client viene fatto scegliere se utilizzare CLI o GUI. Vengono quindi creati View e ServerHandler, viene richiesto l'IP del server e viene instaurata la connessione.
Viene avviata la classe AckControl che invia ogni 5 secondi un messaggio di ping per verificare la connessione al server. Nel caso in cui sia caduta, viene notificato al client tale evento. Contemporaneamente a questo, il server, una volta accettata la connessione, chiederà al client il nickname (tramite messaggio). Nel caso non fosse univoco, verrà richiesto nuovamente. A questo punto viene chiesto al client se vuole iniziare una nuova partita o se vuole connettersi ad una lobby (vengono passate tramite messaggio tutte le lobby disponibili): se non è presente alcuna lobby viene creata una nuova partita, altrimenti l'utente può scegliere in quale entrare potendo vedere gli utenti già collegati ad ogni lobby.
Continuando sullo stesso sequence diagram, dopo aver scelto di creare una nuova partita verrà chiesto il numero di giocatore e la modalità di gioco e infine il colore delle torri tra quelli ancora disponibili.

Passando invece al caso in cui si scelga di entrare in una lobby (allegato Eriantys protocollo lobby, da messaggio 11 in avanti), possono succedere due cose: la lobby scelta è ancora disponibile e sta attendendo nuovi giocatori e quindi accetta il client, oppure nel mentre si è riempita e quindi notifica di sceglierne un'altra.

Altre funzioni implementate sono la refresh, invocabile dal client in fase di scelta della lobby per ricevere la lista di lobby aggiornata. La lista di lobby non mostrerà le lobby piene e quelle che non hanno ancora impostato il numero di giocatori.