Peer-Review 2: Network Protocol

Airoldi, Confalonieri, Ettore

Gruppo **AM03**

Valutazione del protocollo di comunicazione del gruppo **AM33**

Lati Positivi

Abbiamo apprezzato la presenza di numerosi messaggi, specifici e atomici, che permettono maggiore scalabilità nella comunicazione tra server e client.

Il client risulta inoltre particolarmente leggero dato che è il server ad occuparsi della gestione dei messaggi e non c'è quindi bisogno di un gestore degli stati che si dedichi alla scelta delle azioni da inviare.

Anche voi avete utilizzato la libreria json, che rende immediate la serializzazione e la deserializzazione delle classi in stringhe. In particolare, il funzionamento della deserializzazione è simile, se non uguale, al nostro.

Buono anche il keep-alive incaricato del controllo delle disconnessioni lato server.

Lati Negativi

Sarebbe comodo aggiungere il keep-alive anche lato client, dato che una sua mancata implementazione non permetterebbe all'utente di essere a conoscenza di eventuali disconnessioni con il server.

Avendo utilizzato un approccio di tipo "thin" il client risulta poco indipendente, infatti necessita di un server che richieda delle azioni, inoltre i messaggi saranno molteplici e di tipo diverso (divisione tra request e action).

Confronto tra le architetture

Buono l'utilizzo del pattern action, infatti anche noi inizialmente avevamo pensato di usare tale approccio, successivamente però abbiamo deciso di ridurre il numero dei messaggi tramite l'implementazione di un controller, incaricato di chiamare dei servizi esterni alle action in modo che, queste ultime, possano essere utilizzate e condivise, sia dal server che dal client.

Abbiamo implementato in modo equivalente i metodi di serializzazione e deserializzazione, utilizzando una enumeration contenente il tipo di messaggio, in questo modo viene immediato risalire alla classe corretta.

Simile anche l'idea del keep-alive, con l'unica differenza di averla pensata sia lato server che lato client.