Prova Finale (Progetto) di Ingegneria del Software- A.A. 2022/23 gruppo AM-25
Peer review n. 2
SEQUENCE DIAGRAM

Alfano Marco Boimah Divine Boisseau Arianna Cerutti Andrea

Meccanismo di connessione:

Il progetto verte sullo scambio tra client e server di oggetti chiamati “Message”, ne esistono varie sottoclassi accumunate in primis per origine del messaggio:

Message:

- ServerMessage
- ClientMessage

Successivamente altre sottoclassi vengono introdotte e rappresentano un’azione o un informazione da comunicare all’altra parte.

Gli oggetti Message conterranno eventualmente informazioni riguardanti l’evento, queste possono variare da singole stringhe fino a oggetti di bassa complessità (rappresentanti di informazioni del gioco).

Questi *Message* sono necessari per avvolgere più informazioni in modo efficacie nella comunicazione senza creare inutile separazioni di informazioni e introducendo ulteriori controlli.

Lo schema seguito è un server reattivo alle comunicazioni del client, che risponde alle richieste e restituisce il risultato delle operazioni sottoposte.

Ogni richiesta inviata al server contiene tutte le informazioni per essere eseguita lato server senza interruzioni, il risultato può essere successo o fallimento. In caso di fallimento viene generata un’eccezione che viene tradotta in un Message che quindi viene inviato in risposta al Client.

RMI e TCP socket:

In questo documento verrà illustrato con sequence diagram solo e soltanto esempi con protocollo RMI; per la scelta di design implementata è indifferente l’implementazione con la quale gli oggetti “Message” vengono trasmessi.

Entrambe le implementazioni si basano su un’oggetto **Server** padre di **TCPServer** e **RMIServer**, che gestiscono le comunicazioni per le relative tecnologie.

Le due modalità sono state implementate per RMI come due interfaccie (una client e una server) che possiedono dei metodi:

- receiveMessage(Message msg)

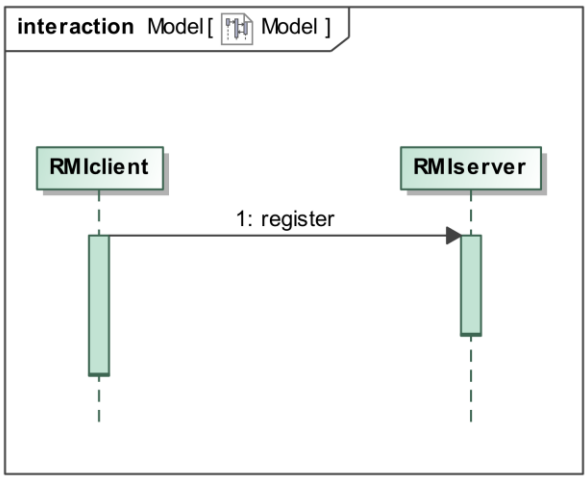
per TCP socket invece **TCPServer** accetta le richieste TCP e un altro oggetto **ClientHandler** che comunica con i Client. Le parti comunicanti contengono ognuno due stream, inputStream e outputStream che ricevono e mandano oggetti serializzati.

Fasi per la connessione e consuetudine di gioco:

Verranno ora illustrate come elenco puntato le varie fasi corrispondenti al protocollo di comunicazione a cui i client devono attenersi:

1. **Registrazione**, accettazione della comunicazione per TCP e semplice aggiunnta dell'oggetto remoto per RMI
2. **Richiesta delle Lobbies disponibili**, il server risponderà con un messaggio contenente tutte le lobbies
3. **Creazione di una lobby**
4. **Join di una Lobby esistente**
5. **Disconnessione da una partita**
6. **Riconnessione ad una partita**
7. **Invio di una mossa**
8. **Invio di un messaggio via chat**

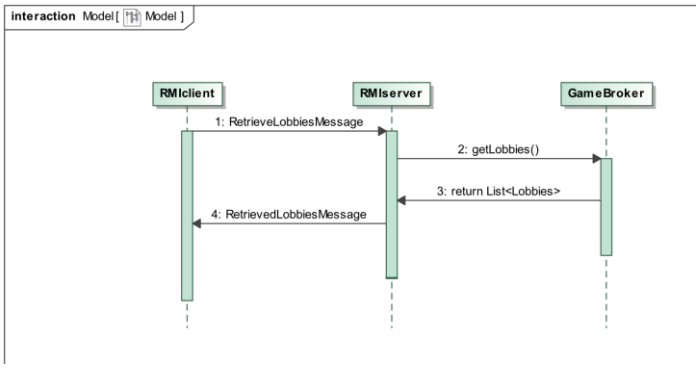
1. Registrazione



Per RMI una semplice invocazione del metodo register(RMIClientInterface) è sufficiente ad iscriversi al server.

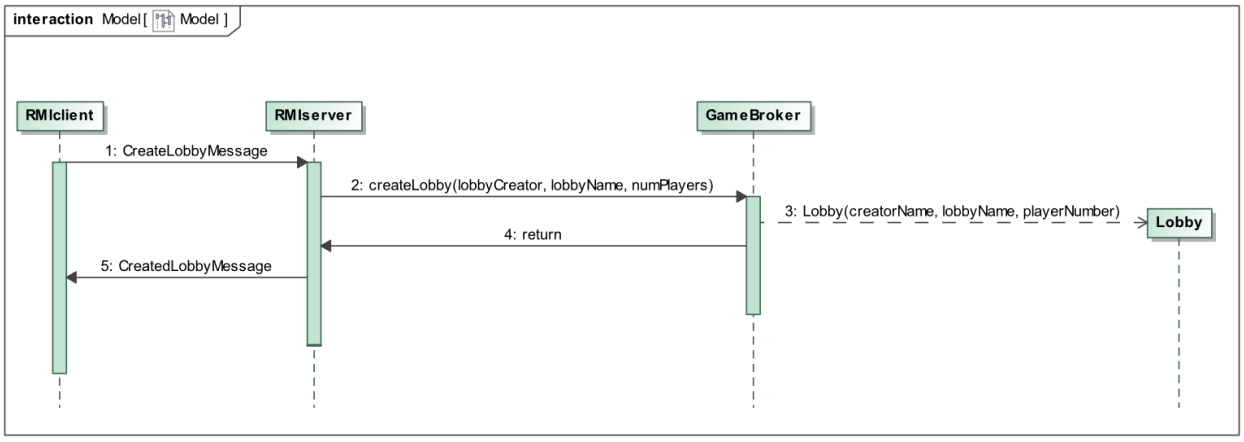
Viene passata per parametro l'oggetto remoto che rappresenta il client stesso.

2. Richiesta delle Lobbies disponibili



Un oggetto RetrieveLobbiesMessage è inviato al server, che inoltra al **Client** il risultato del metodo getLobbies() di **GameBroker** (l'oggetto che gestisce le lobby) attraverso un messaggio contenente una lista di **Stringhe** contenente i nomi delle lobby disponibili.

3. Creazione di una Lobby o join di una esistente



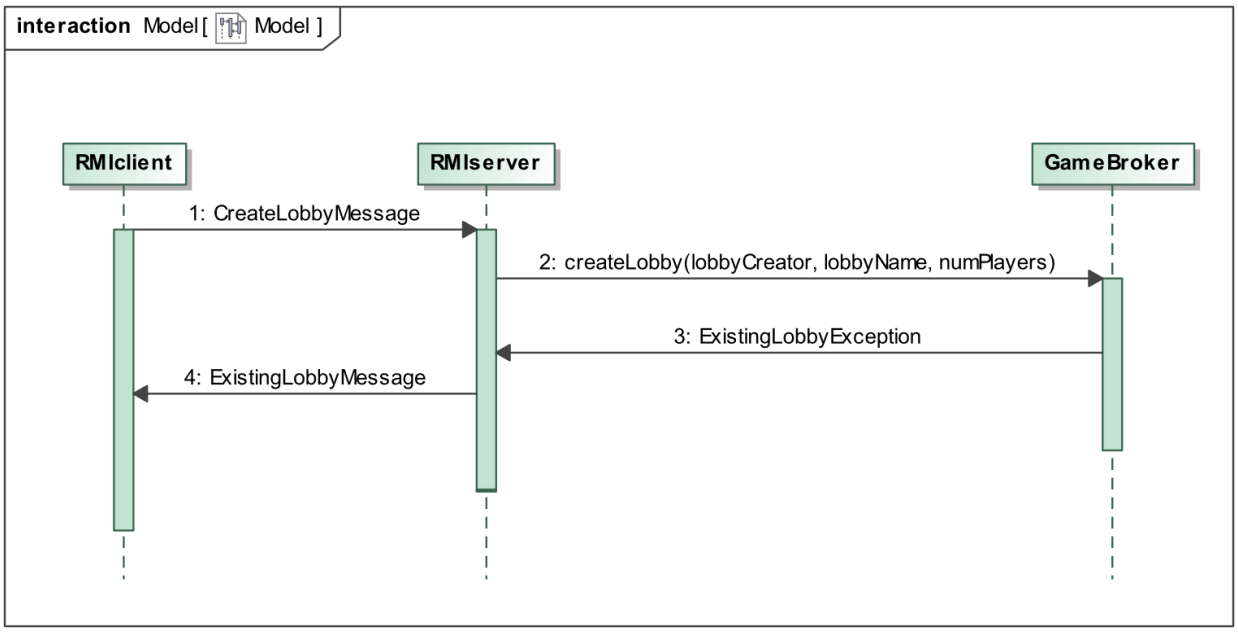
Il Client invia un oggetto message "CreateLobbyMessage" che contiene gli elementi per creare una lobby e indentificarla unicamente:

- Creatore
- Nome della lobby
- Numero di giocatori per la partita

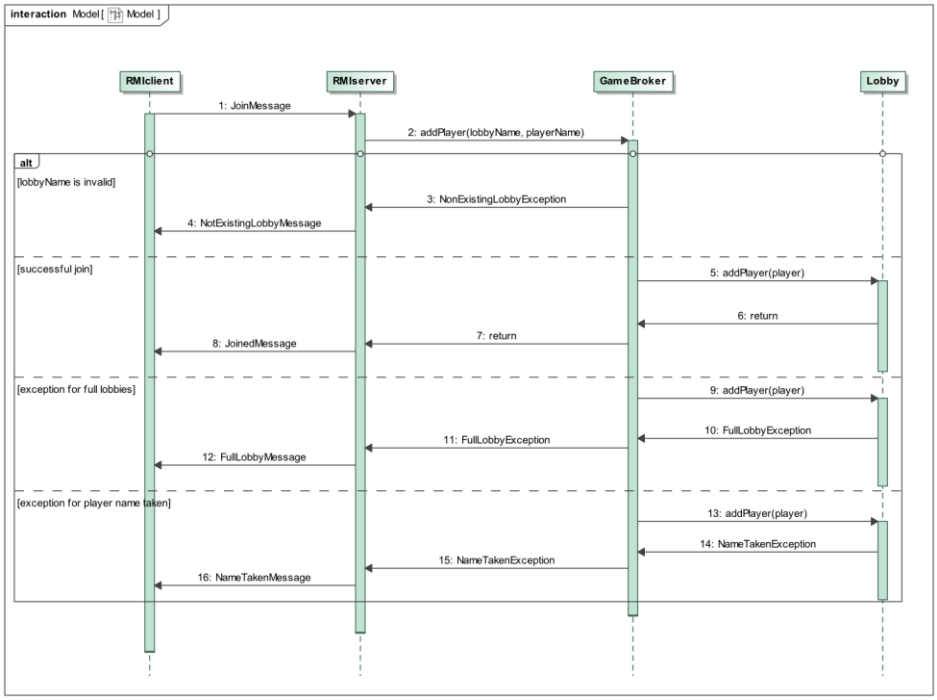
Se la creazione della lobby ha successo il server invia al Client un oggetto messaggio "CreatedLobbyMessage"

Una volta raggiunto il numero di giocatori necessari per l'inizio della partita il gioco viene istanziato.+

Potrebbe arrivare una richiesta di creazione di una lobby con un nome Lobby già esistente, in queste casistiche il server risponde al client con un messaggio di eccezione.



4. Join di una Lobby esistente

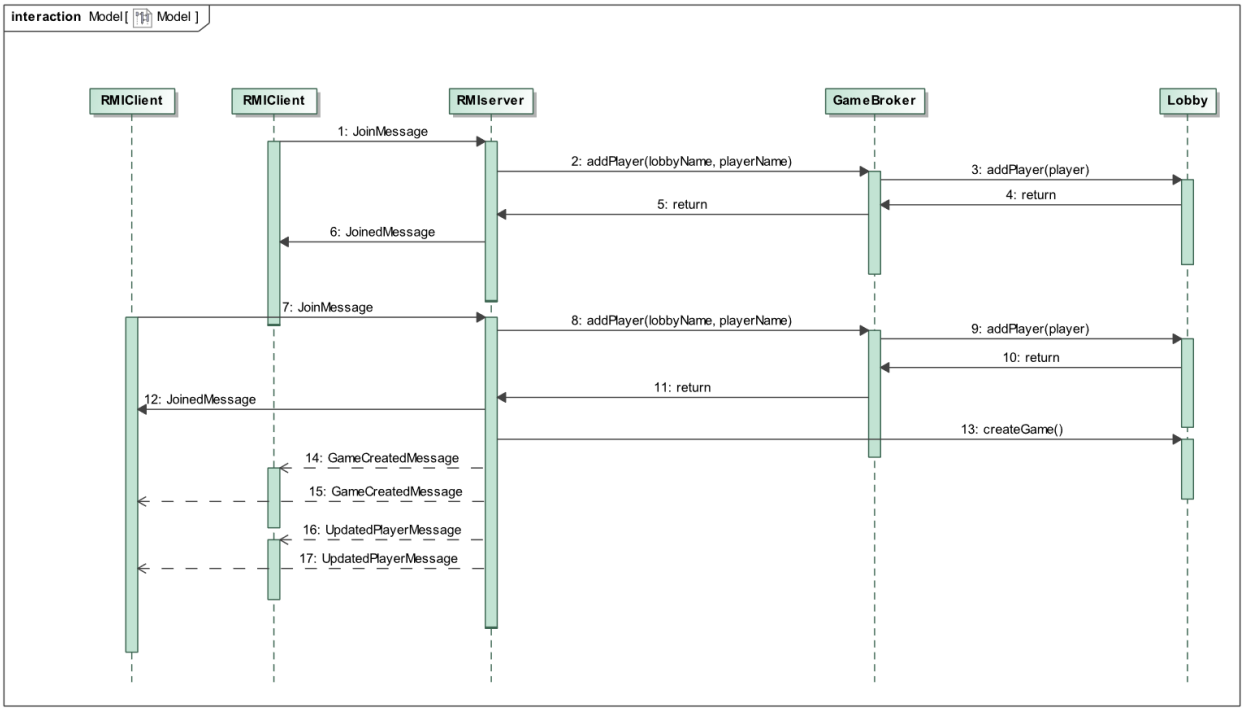


Il Client invia un oggetto JoinMessage contenente il nome della lobby dove si vuole unire e Nome Player sotto il quale ci si vuole iscrivere. Il server riceve il messaggio e inoltra la richiesta al Gamebroker.

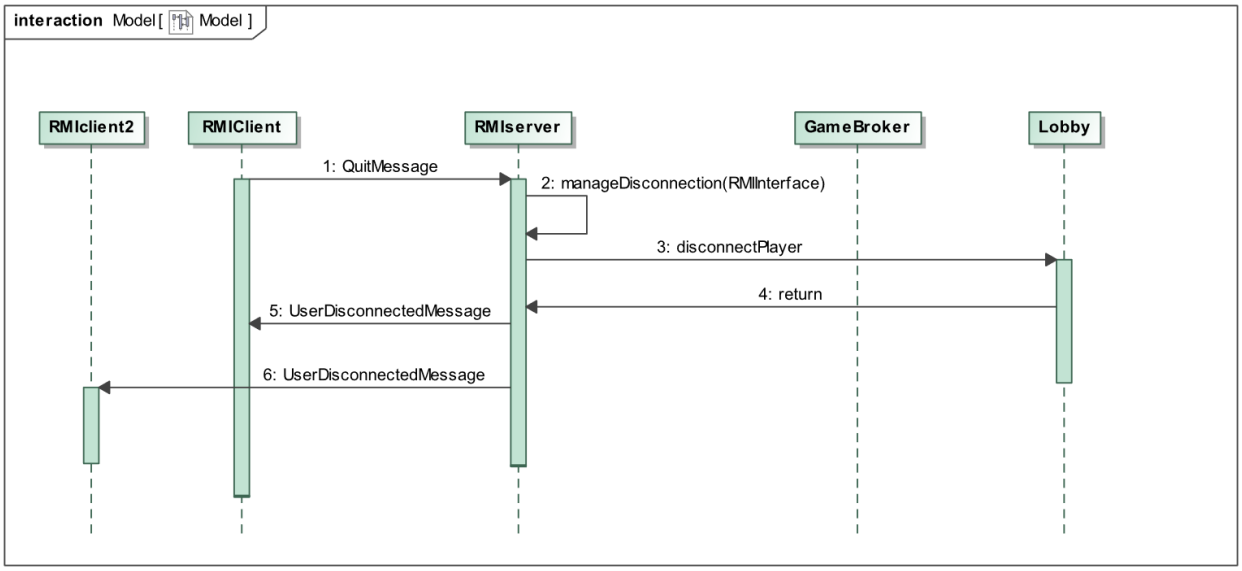
Il GameBroker può rispondere in vari modi:

1. Il nome della lobby inviato non è valido, non esiste alcuna lobby con quel nome e viene inviato un messaggio di eccezione;
2. Il client si è unito correttamente ad una lobby, il messaggio inviato al client è di successo **JoinedMessage;**
3. La lobby a cui si è fatta richiesta è piena, viene inviato al client un messaggio di eccezione;
4. Il nome sotto il quale ci si vuole iscrivere alla partita è già stato preso, viene inviato al client un messaggio di eccezione.

Il caso sotto illustrato si verifica quando l'ultimo player si unisce alla lobby, quindi il gioco viene istanziato e le prime informazioni riguardanti alla "scacchiera" e al giocatore che deve giocare per primo vengono inviate a TUTTI i client iscritti alla lobby attraverso GameCreatedMessage e UpdatedPlayerMessage.

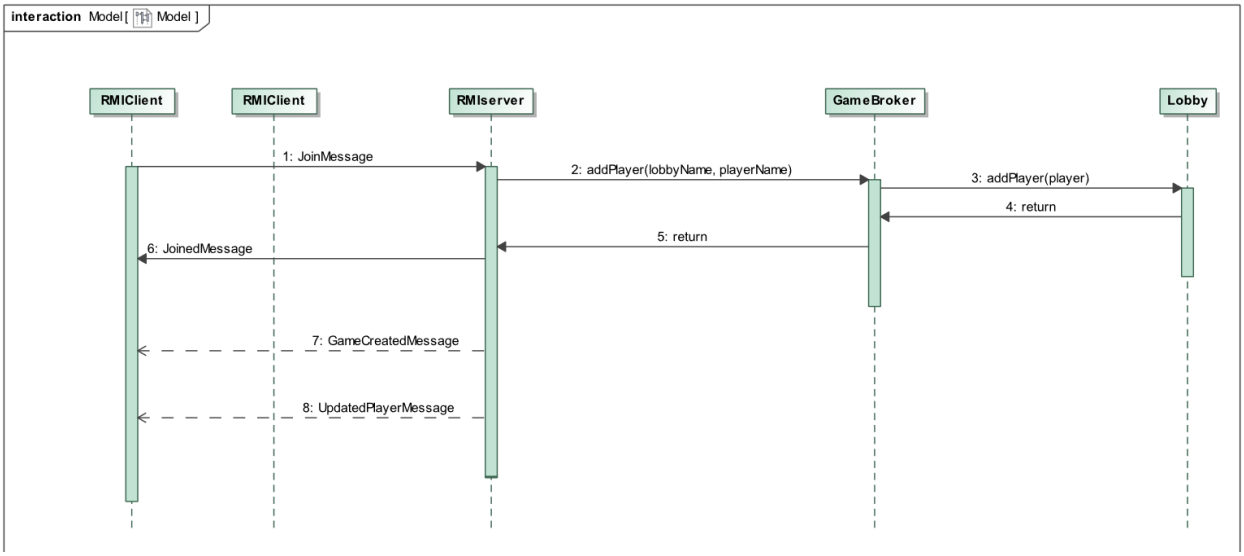


5. Disconnessione da una partita



La disconnessione dal gioco di un client viene inoltrata alla lobby che provvede ad informare il model e successivamente il server invia messaggi che notificano tutti i Client della disconnessione del giocatore.

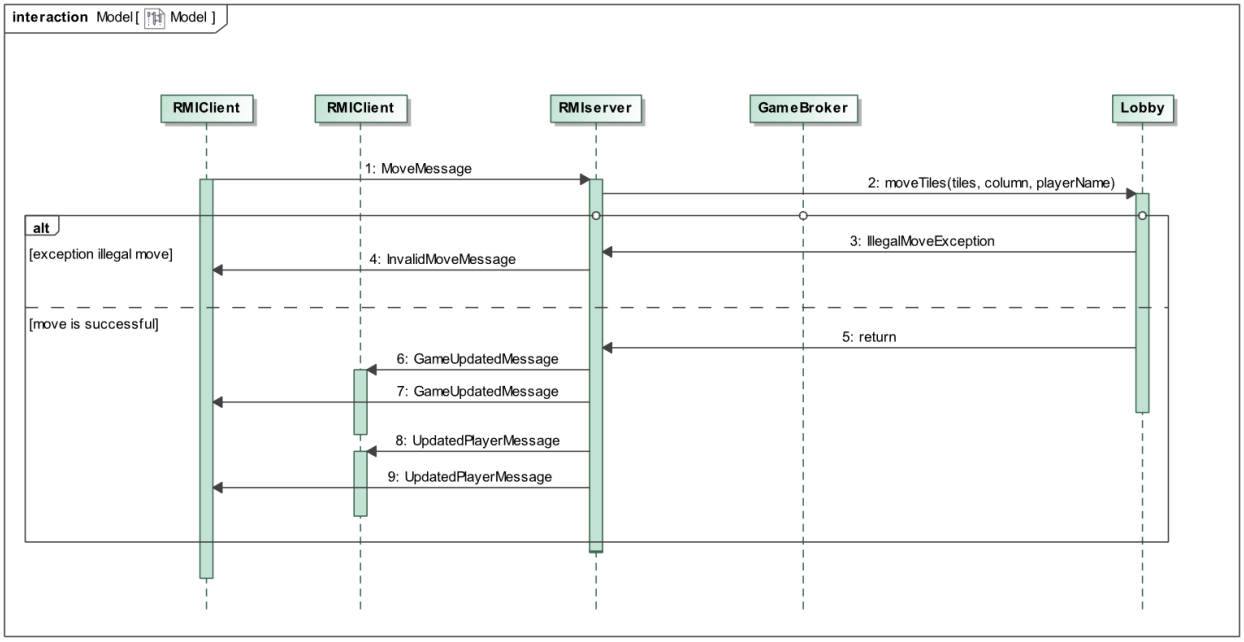
6. Riconnessione ad una partita



Nel caso di una riconnessione da parte di un client, questa deve essere eseguita tramite un JoinMessage contenenti le stesse informazioni del precendentemente usato per la prima iscrizione. Quindi alla stessa lobby sotto lo stesso nome.

Al Client appena riconnesso gli vengono inviati i dati del gioco aggiornati tramite 2 messaggi.

7. Invio di una mossa

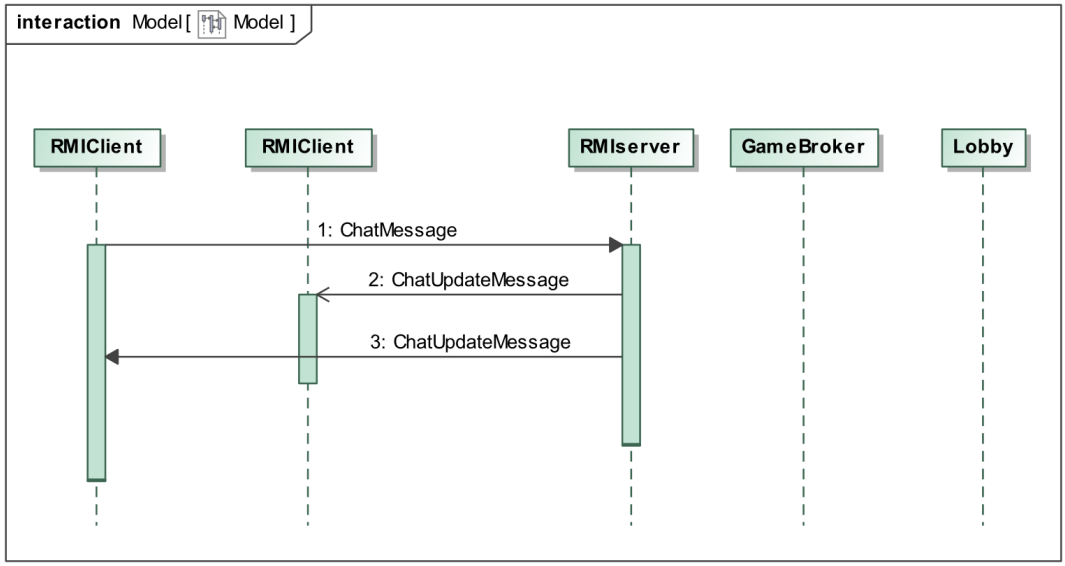


L'invio da parte di un Client di una mossa segue sempre lo stesso schema di invio di un oggetto MoveMessage.

In caso di mossa invalida una risposta di eccezione "InvalidMoveMessage" viene generata e inviata.

Nel caso di corretto invio di una mossa, quindi legale per le regole di gioco e corretta per la priorità dei turni, viene inviato un messaggio a TUTTI i Client, contenente le informazioni di gioco aggiornate alla mossa appena eseguita.

8. Invio di un messaggio via Chat



La chat è stata implementata legata alla Lobby, quindi questa viene resa disponibile all'iscrizione ad una partita e rimane durante la partita, il client invia dei ChatMessage contenenti il messaggio e il mittente; il server provvede a generare un timestap e inoltra a tutti il messaggio tramite dei ChatUpdateMessage.

Note riguardo alle disconnessioni forzate o ping

Sono state introdotte funzionalità di timeout sia lato server che lato client, che permettono al fine partita di essere puramente un evento lato client, con una susseguente eliminazione della lobby solo in caso di disconnessione forzata o non di TUTTI i client.

La disconnessione forzata è un meccanismo a timeout lato server, che per ogni giocatore disconnette il player in caso il dispositivo di questo non risponda per un intervallo prestabilito