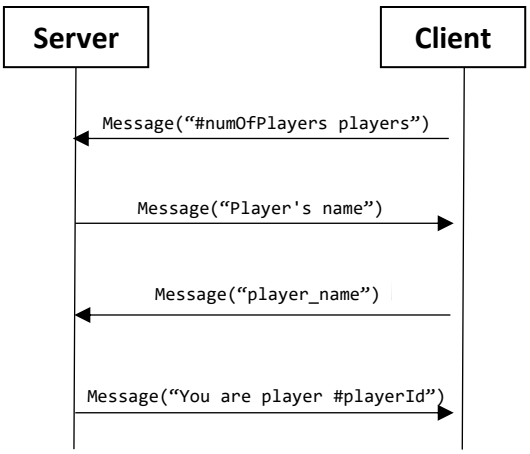
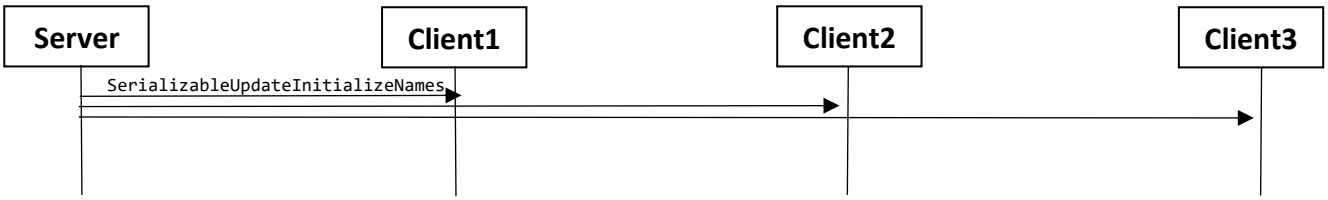
Inizializzazione connessione



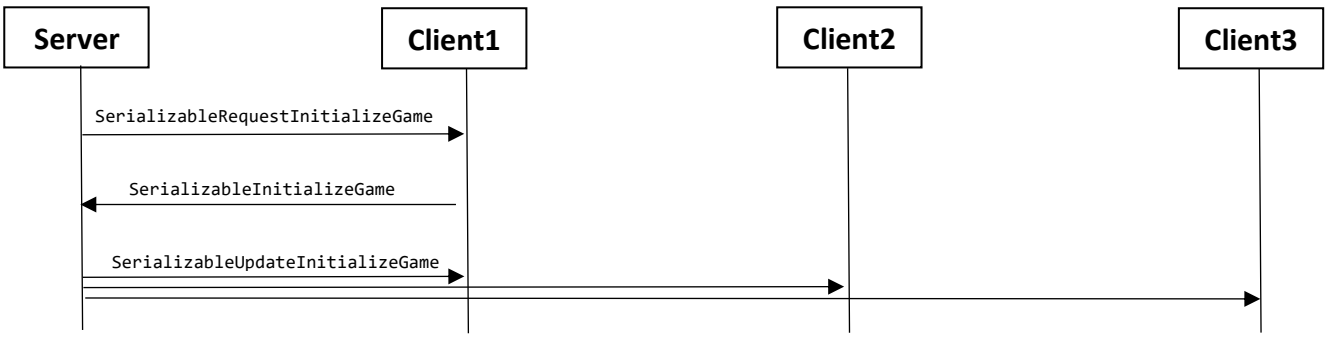
Client: comunica al Server il numero di giocatori con cui si vuole giocare (2/3).

Server: salva in lista d'attesa la socket del Client e rimane in attesa di altri giocatori che vogliano giocare ad una partita con lo stesso numero di giocatori; quando si sono connessi sufficienti client chiede il nickname dei giocatori e assegna loro un numero identificativo (1/2/3).

Inizializzazione partita



Server: invia ai client SerializableUpdateInitializeNames che riporta i nicknames di tutti i giocatori.

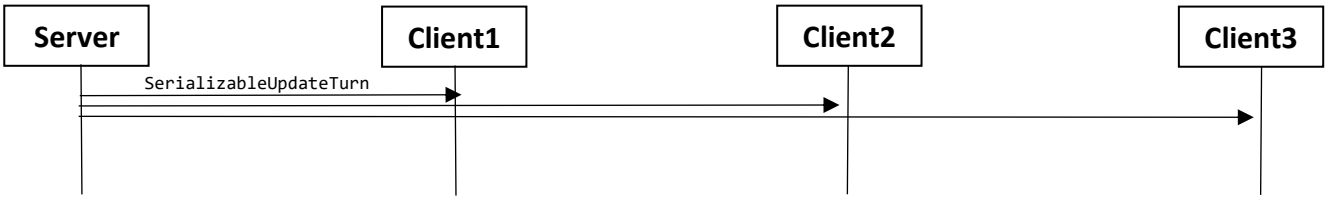


Server: invia un oggetto SerializableRequestInitializeGame ad ogni client in sequenza per chiedere ai giocatori di scegliere divinità e posizioni dei workers.

Client: risponde al Server con un oggetto SerializableInitializeGame contenente divinità e posizioni dei lavoratori scelte dal player.

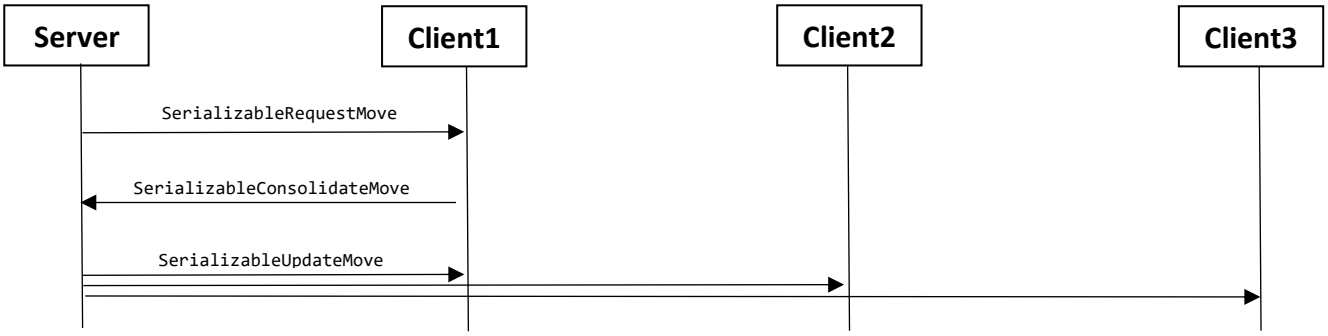
Server: tramite un oggetto SerializableUpdateInitializeGame notifica tutti i client dell'aggiunta dei lavoratori avversari alla plancia di gioco e della scelta di una divinità da parte di un giocatore.

Cambio turno



Server: invia ai client SerializableUpdateTurn che riporta il giocatore di turno corrente; deve essere lanciato all'inizio di ogni turno.

Move

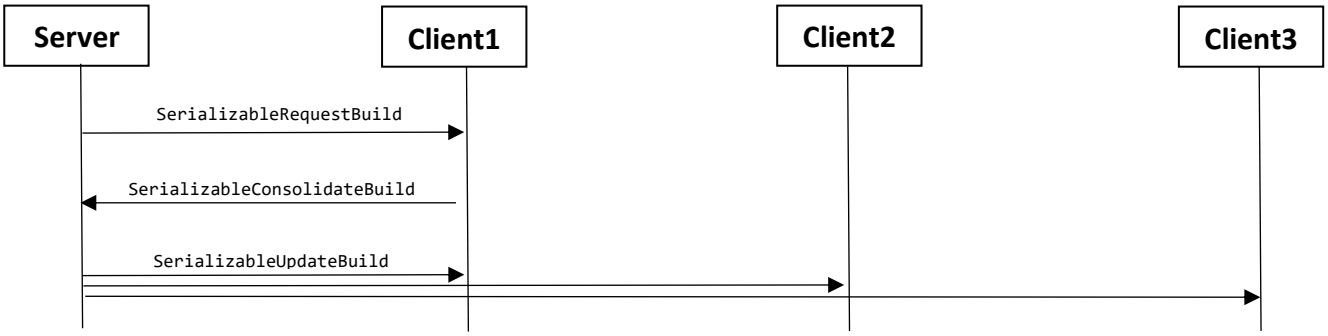


Server: invia al player di turno SerializableRequestMove contenente le posizioni in cui i workers possono spostarsi.

Client: invia al server SerializableConsolidateMove che contiene il lavoratore e la posizione di destinazione.

Server: tramite un oggetto SerializableUpdateMove notifica tutti i client della mossa effettuata dal lavoratore avversario.

Build

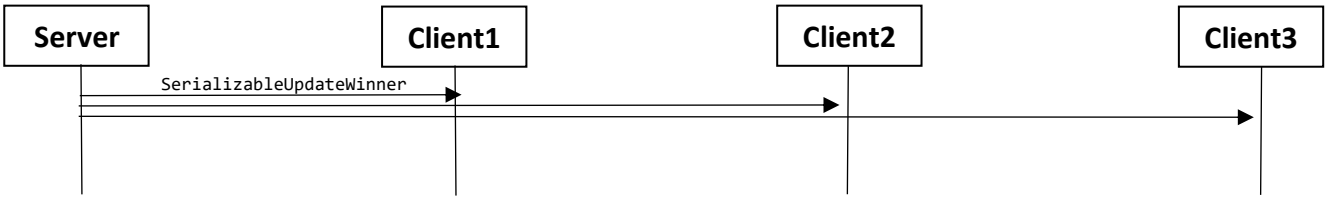


Server: invia al player di turno SerializableRequestBuild contenente le posizioni in cui i workers possono costruire e un boolean che indica se il player abbia diritto a forzare la costruzione di un dome.

Client: invia al Server SerializableConsolidateBuild che contiene il lavoratore, la cella in cui si desidera costruire e un boolean, risultante true se e solo se un player avente diritto di forzare la costruzione di un dome mette in atto tale potere.

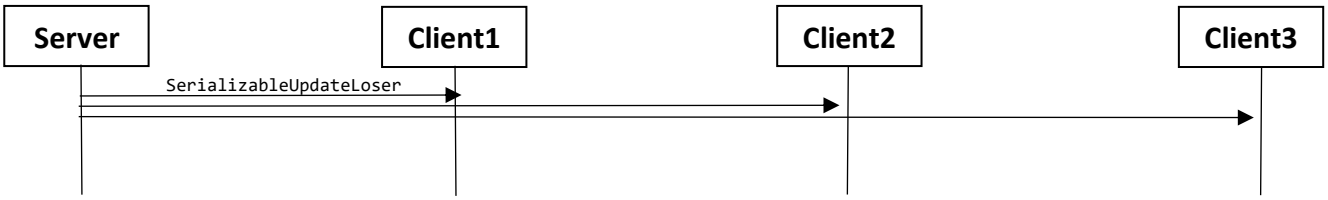
Server: tramite un oggetto SerializableUpdateBuild notifica tutti i client della build effettuata dal lavoratore avversario. Contiene un valore boolean risultante true se e solo se è stato costruito un dome.

Vittoria



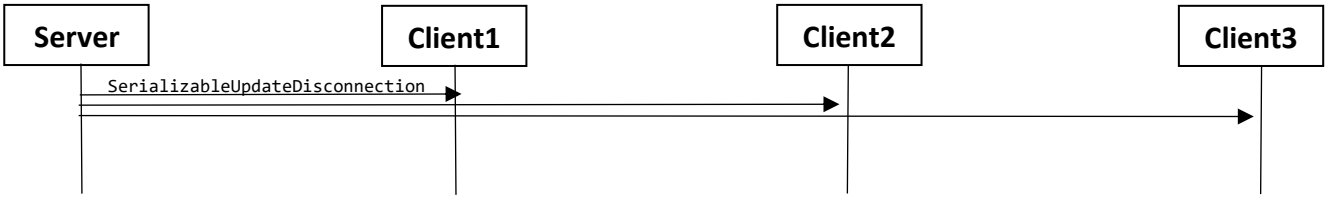
Server: invia a tutti i client SerializableUpdateWinner contenente il numero identificativo del player che ha vinto. L'invio di questo oggetto termina la partita, i client ancora attivi si disconnettono e il server chiude il relativo thread.

Sconfitta



Server: invia a tutti i client SerializableUpdateLoser contenente l'identificativo del player che ha perso. In una partita con 2 giocatori rimasti questo messaggio sarà seguito da un SerializableUpdateWinner per decretare il vincitore, in una partita con 3 giocatori rimasti il Server invierà un SerializableUpdateTurn e avrà inizio il turno successivo.

Disconnessione



Server: invia a tutti i client SerializableUpdateDisconnection contenente il numero identificativo del player che si è disconnesso. L'invio di questo oggetto termina la partita, i client ancora attivi si disconnettono e il server chiude il relativo thread.

Azione facoltativa

Server: invia a tutti i client SerializableRequestOptional, contenente le possibili moves e/o builds, un boolean risultante true se sono le moves ad essere facoltative e le builds obbligatorie (false viceversa), un altro boolean risultante true se il Client può rifiutare l'azione facoltativa terminando il turno. In presenza di un'ultima azione facoltativa nell'oggetto sono presenti solo azioni di una delle due categorie (move/build) e il secondo boolean è true.

Client: risponde con un normale oggetto SerializableConsolidateMove/SerializableConsolidateBuild in base alla decisione del player, secondo la normale interazione move/build, oppure, se ne ha il permesso, può rifiutare l'azione facoltativa mediante oggetto SerializableDeclineLastOptional, terminando così il proprio turno.