Communication protocol

group AM 35

14/05/2019

We chose a 'fat model' approach, with a very skinny view. Therefore, the model is the only one that knows the logic of the game and determines what each player can do in a specific moment. The set of the selectable items (squares, players, commands, actions, ...) is sent to the client as lists of string. Once the login is performed (which is done on client initiative), the only interaction of the client is the selection and forwarding of one of these items (if any is available). The controller takes in charge the selection and makes the model update itself, then the new status and new selectables are sent to each player. Apart from selecting items, a client can choose to disconnect: in this case it sends a request to the server.

Once the match is started, players of the match are set permanently and remain until the end. When a player disconnects he becomes INACTIVE (can't do anything but he stays in the board and he can receive damages). Connection-loss, player choosing quitting the game and timeouts are all treated in this way.

When the client connects to the server, the server starts waiting for client's Events. The client sends a loginEvent which contains the chosen username and then starts waiting for server's Messages. Server and client listen for each other for the rest of the connection.

1 Messages and events

Information travel in form of Messages and Events. Message and Event are serializable objects, which both implement Visitable interface. Messages travel from server to client, while Events travel from client to server.

1.1 Type of messages

The server can send 6 types of messages:

- LoginMessage: states if the registration completed successfully
- ConnectionStateMessage: list of connected Players during login phase (before match starts)
- UpdateMessage: describes the situation of the game (by the point of view of one player). It contains:
  - all information about receiver player

  - partial information about other players
  - ammotiles and weapons on the board
  - state of the game
  - state of connection
  - all selectable lists of receiver player
- ResponseMessage: states if the client command is valid
- DisconnectionMessage: informs player that the connection is going to be closed
- GameOverMessage: contains the winner and each player's points.

1.2 Type of Events

The client can send 10 types of events:

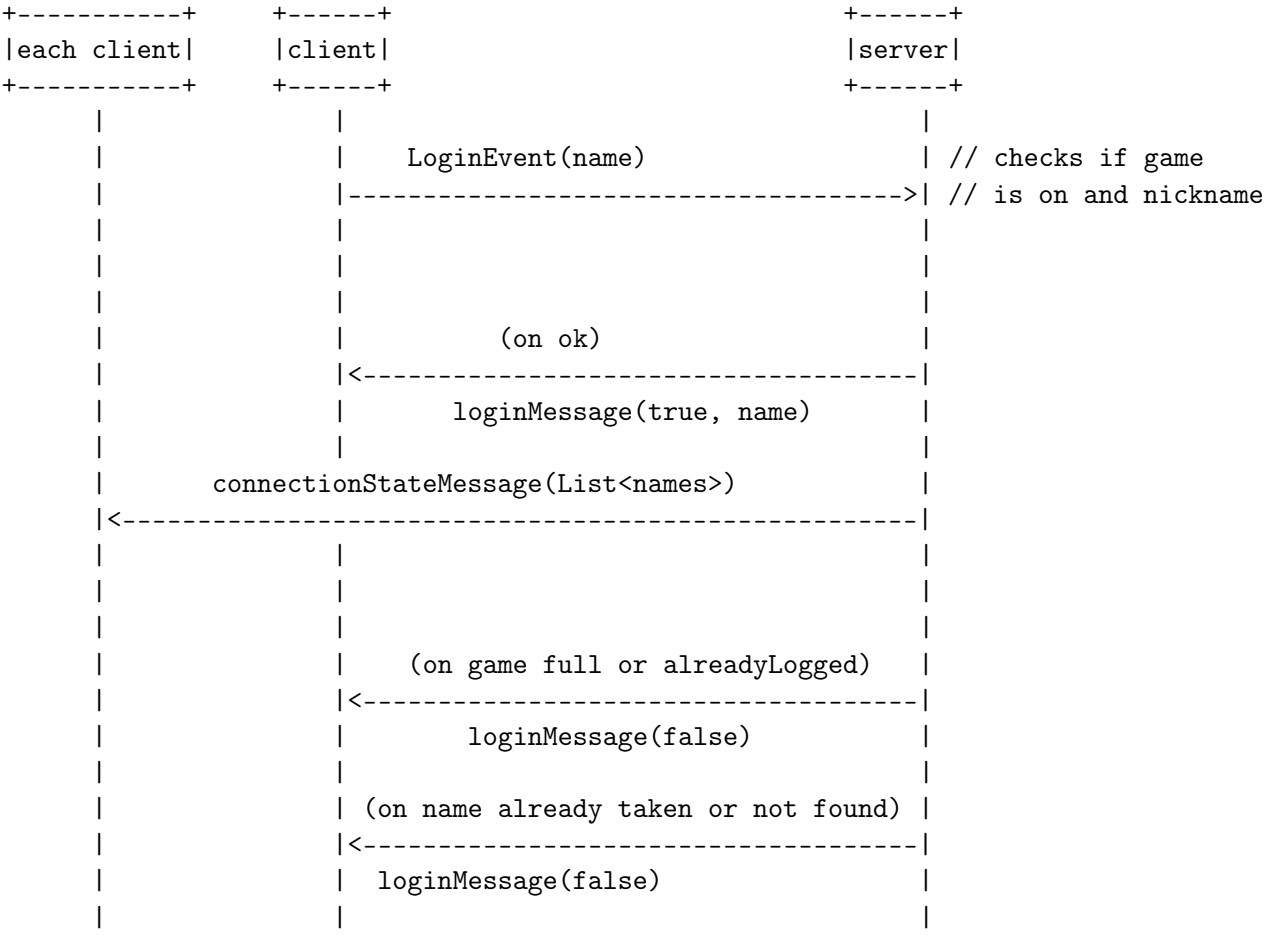
- loginEvent: contains the chosen nickname
- ActionSelectedEvent: contains the selected Action (as index of the selectable list)
- ColorSelectedEvent: contains the selected color (as index of the selectable list)
- CommandSelectedEvent: contains the selected command (as index of the selectable list)
- ModeSelectedEvent: contains the selected mode (as index of the selectable list)
- PlayerSelectedEvent: contains the selected player (as index of the selectable list)
- PowerUpSelectedEvent: contains the selected power up (as index of the selectable list)
- SquareSelectedEvent: contains the selected square (as index of the selectable list)
- WeaponSelectedEvent: contains the selected weapon (as index of the selectable list)
- DisconnectMeEvent: states that the client wants to disconnect

2 Communication scenarios

The whole comminication can be described through 5 scenarios:

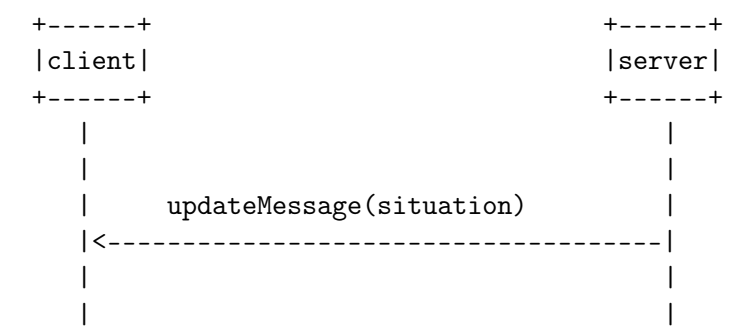
2.1 Login/reconnection

Login and reconnection are in fact the same event from the client. They are treated differently according to the state of the match. The client tries to login until it receives a positive loginMessage.

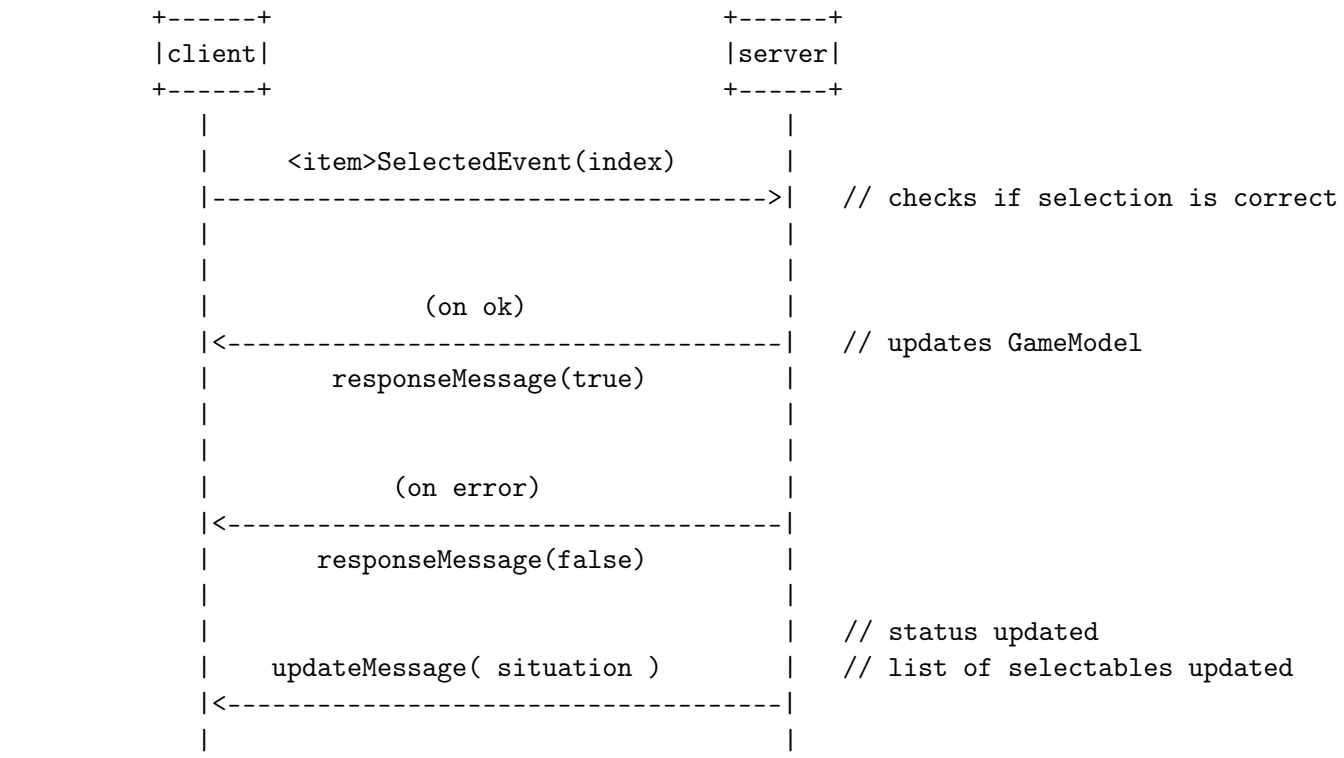


2.2 Game start (for each client)

```
// The server checks if there is a saved game with the same players.
// If yes it restores it, otherwise it starts a new match
```



2.3 Player selects something

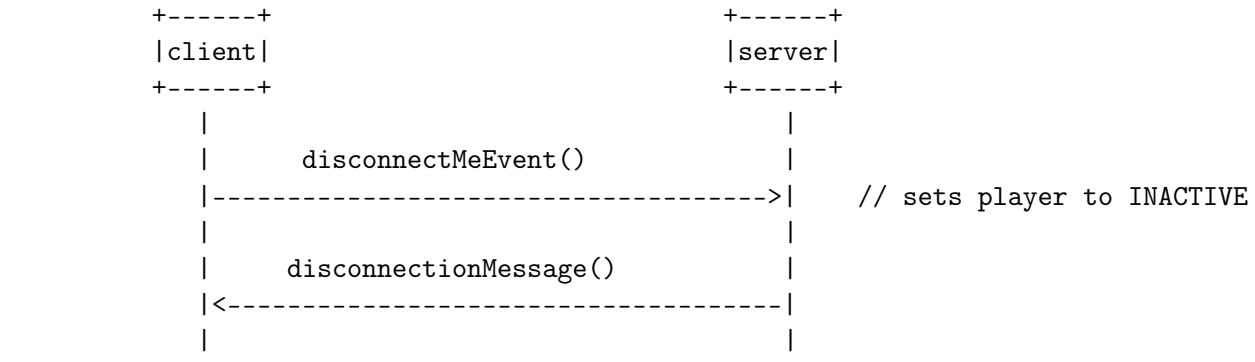



```
//  checks if action triggers the end of game. See game over scenario.
```

2.4 Disconnection

A disconnection can always trigger the end of the game (if there are less than three players). See Game-over scenario. The disconnection can occur for three different reasons:

2.4.1 The player pauses the game

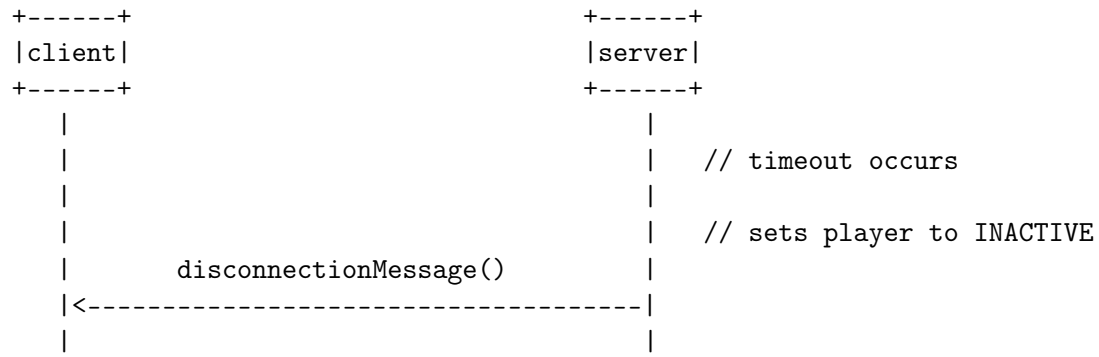



```
 // The client restarts from login phase
```

2.4.2 The connection fails

```
// The client restarts from login phase.
// The server sets corrisponding player to INACTIVE
```

2.4.3 Timeout




```
// The client restarts from login phase
```

2.5 Game over