Communication Protocol

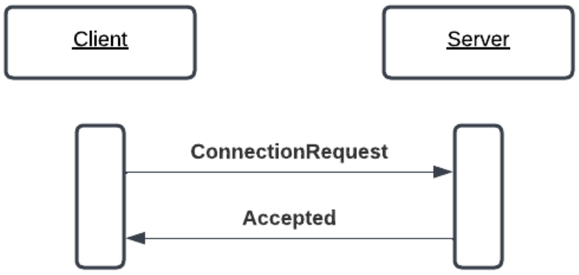
Chiara Colombo – Riccardo Corti – Stefano Da Silva

Group 32

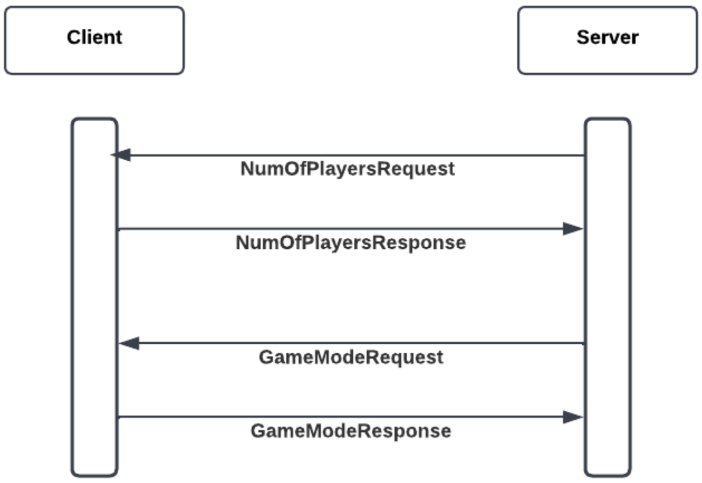
We decided to exchange messages between client and server using java objects native serialization trough serializable interface and the Visitor pattern to get rid of instance of methods when receiving the classes that needs to be sent. The communication protocol relies on TCP which guarantees the correct reception of messages, but we decided to implement a ping message system every 2/3s to notify the disconnection of a player or any other connection error.

Login Phase

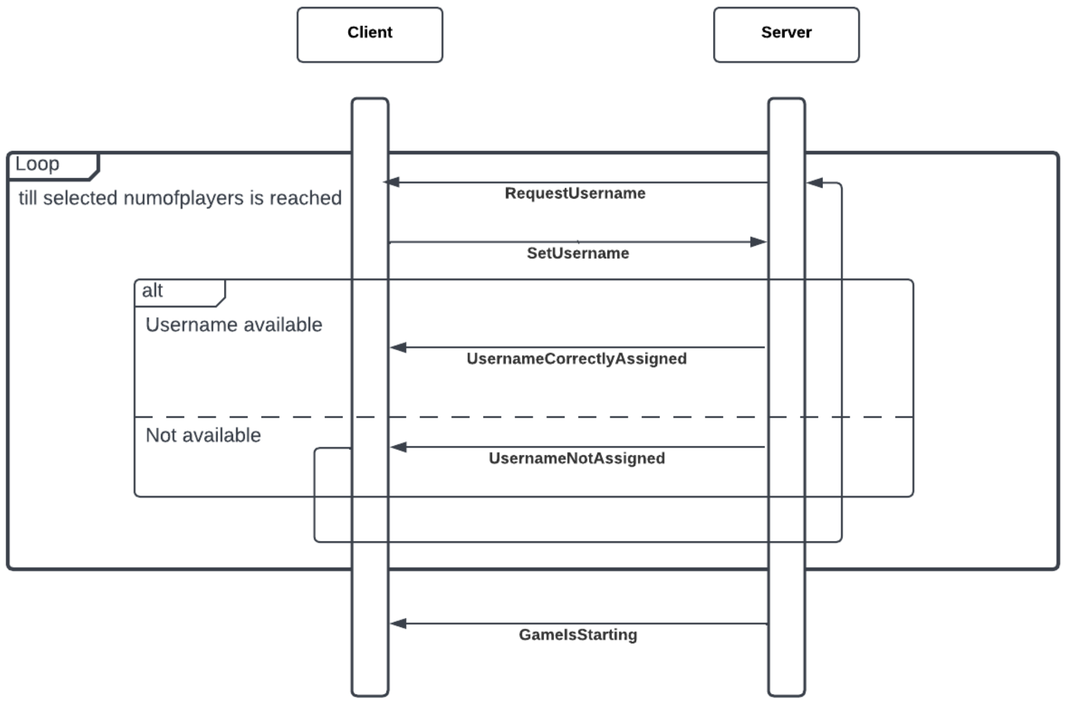
1. First, client needs to establish a connection with the server, it takes as a parameter the server IP and the chosen port. The choice between cli and gui is taken by the client before the connection.



2. The first player entering the game is the one that choose the game mode (whether with the expert mode or normal) and numbers of players. The others player that connects will join first player lobby.

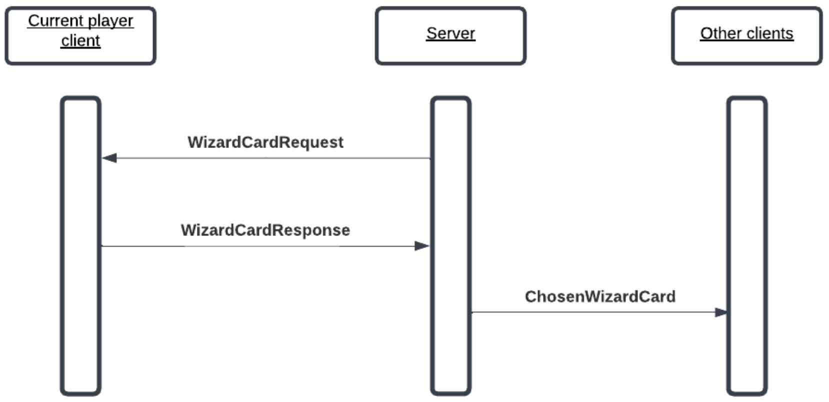


3. Server sends a message **RequestUsername** to the client which replies with **SetUsername** message that contains the specific username. If the given username is not already used by someone else the procedure is successful, if not server sends back a message **RequestUsername**. The procedure is looped until the numofplayers connected equals the number selected by the first player.

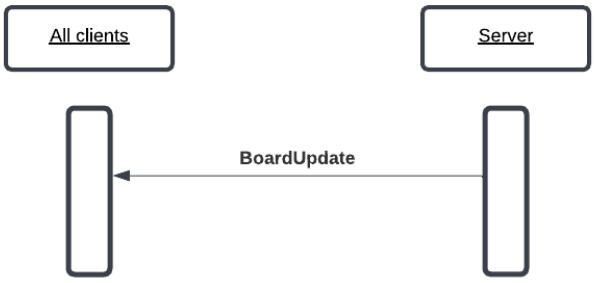


Setup Phase

1. The Server asks each player to select one wizard and then broadcast to all other clients the choice that was made by that particular client.

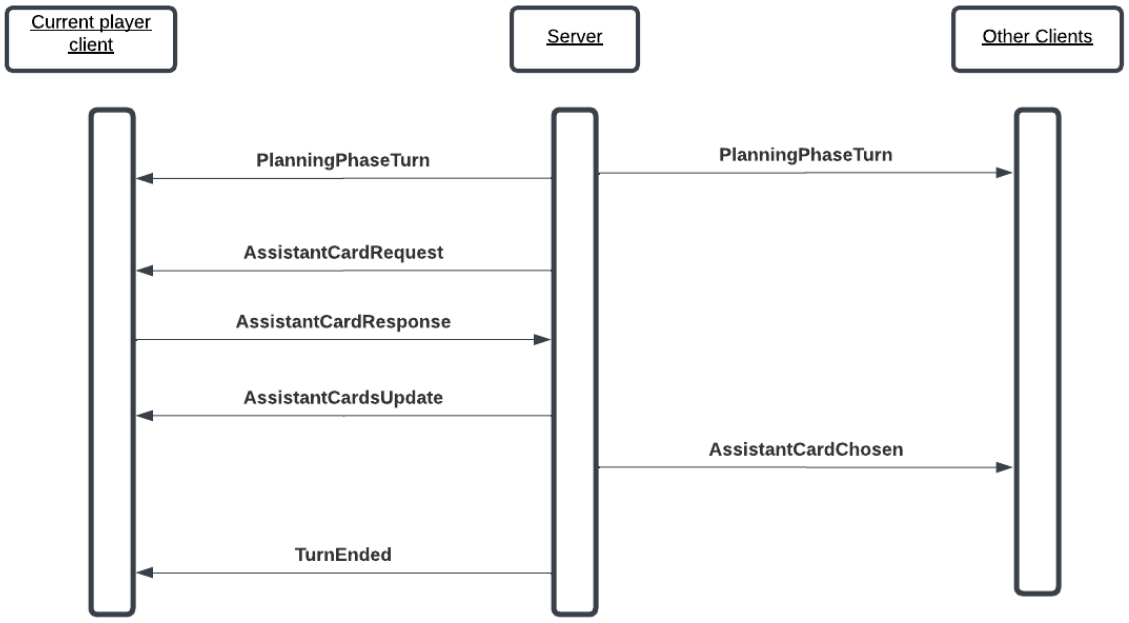


2. Once this procedure is completed by every player Server broadcast a message **BoardUpdate** which contains information about the board, such as mother nature position, students on island, students on cloud, available professors and Character Cards (available only if the expert mode is selected).

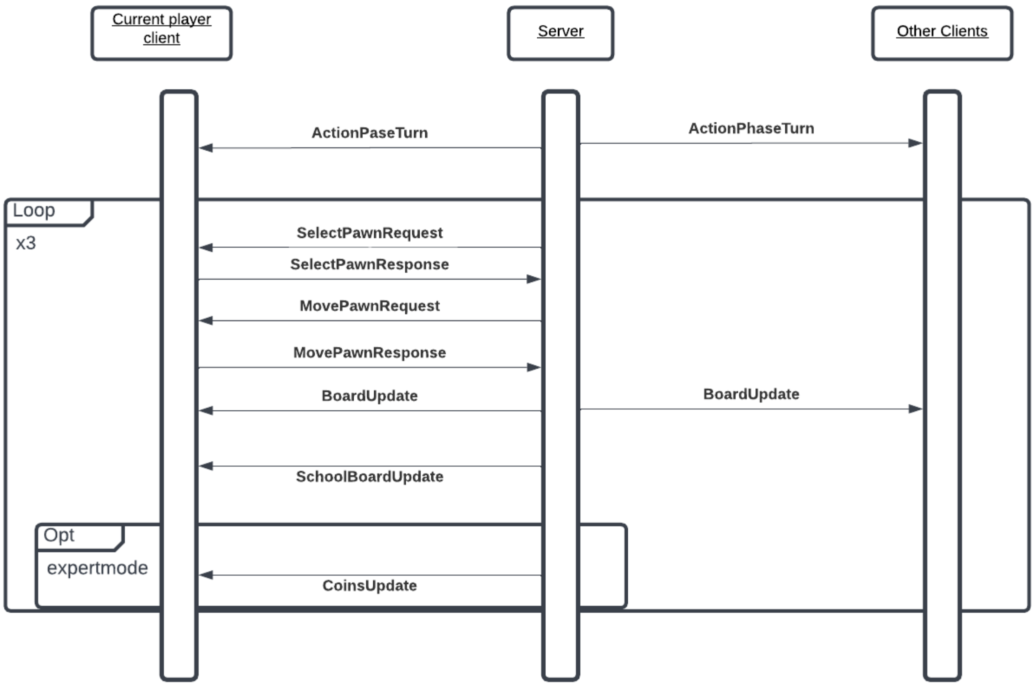


Game Phase

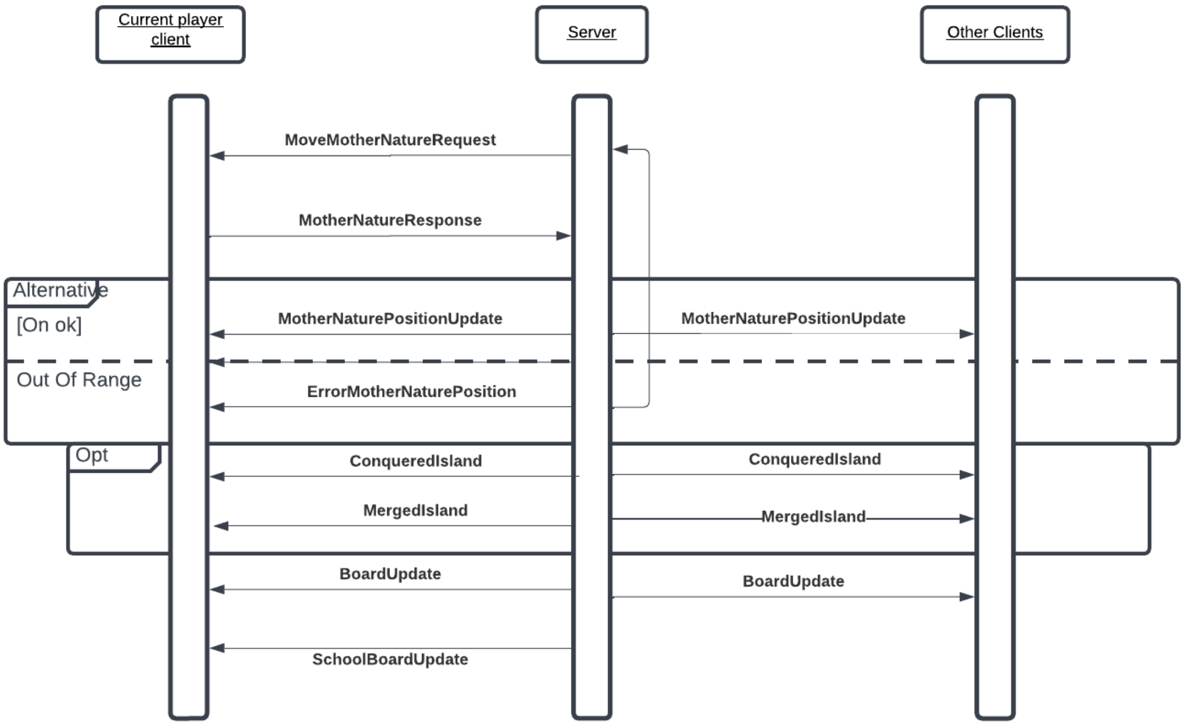
1. Now that the board is settled and every player has a deck of cards, the game enters in his core, the game phase, which is divided in a planning phase and an action phase. The server informs all the clients about the beginning of the planning phase using the **PlanningPhaseTurn** message. This message tells the clients who is playing and sends an **AssistantCardRequest** message to the first player. The **AssistantCardRequest** Message contains the drawable assistant cards. The client answers with a **AssistantCardResponse** that contains the Assistant Card chosen by the current Player. This message leads to an update of the available cards and a broadcast message about the chosen card via **AssistantCardChosen**. Now the Current Player receives a **TurnEnded** message that notifies the end of the Planning Phase. This procedure is repeated for every player.



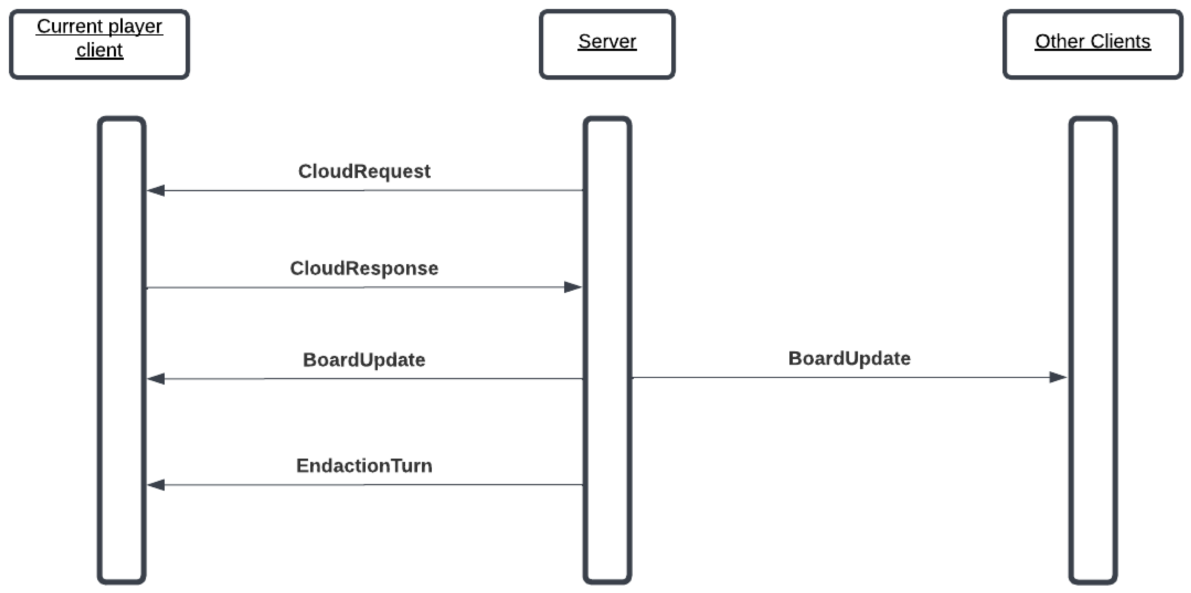
2. When every client has completed the planning phase action phase takes places, starting from the player that picked the assistant card with the lowest value. The mechanism is similar. The phase starts with an **ActionPhaseTurn** message. This message tells which player is going to start. After that a loop (repeated for 3 times) starts. The loop begins with a **SelectPawnRequest** that contains the possible pawns that a player can move followed by a **MovePawnRequest** that contains the available positions where to put the selected pawn. Current Player, now, answers with a **MovePawnResponse** message where are sent the coordinates of the position chosen for the pawn. After that the server sends broadcast a **BoardUpdate** message. Then, the server sends a **SchoolBoardUpdate** message where are contained info about the students on the entrance, the students in dining room and the eventually owned professors. If the game mode is the experts one, the server sends a **CoinsUpdate** message that is about the earned coins in the round.



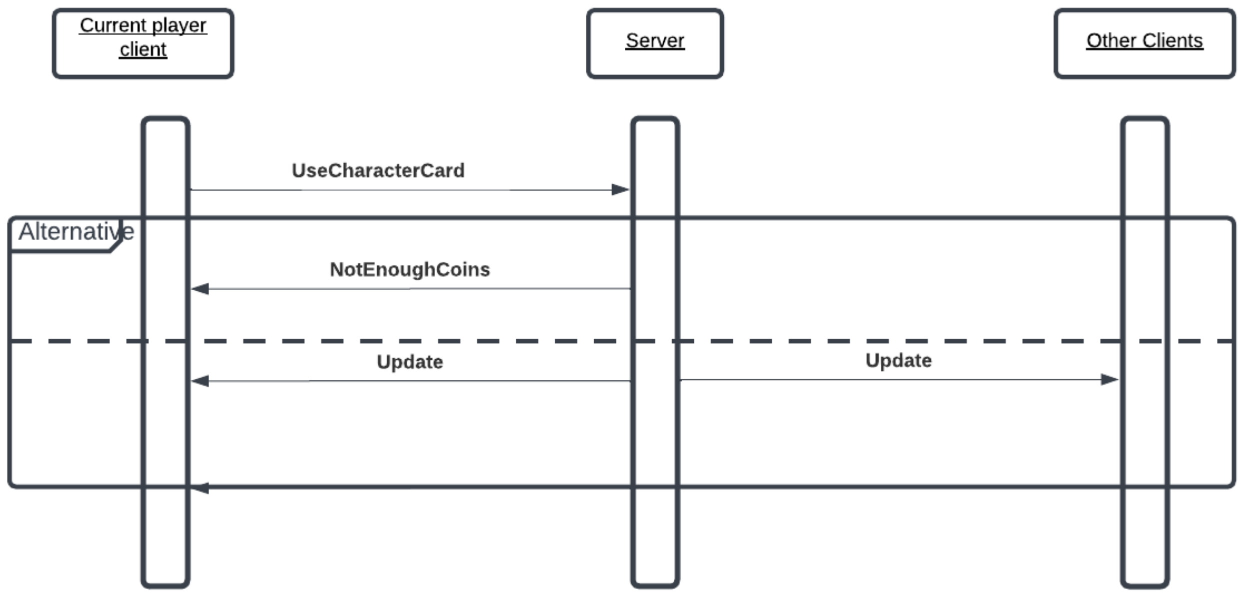
3. The action phase continues with the next part, where player choose where to place mother nature pawn. **MovemotherNatureRequest** contains the possible values that can be chosen. The player answers with a **MotherNatureResponse** that contains the position of Mother Nature. Now there are two options, the first one is that the position is accepted and its consequence is a **MotherNaturePositionUpdate**, the second one is that the position chosen by the client is out of range, therefore the server sends and **ErrorMotherNaturePosition** message and then sends again a **MovemotherNatureRequest**. If an island is conquered or a group of islands is merged, it's notified by two messages broadcasted: **ConqueredIsland** and **MergeIsland**. This part end with two messages: **BoardUpdate** and **SchoolBoardUpdate**.



4. In the end, the player selects one cloud and collect the pawns available. There is again a **BoardUpdate**. The action turn ends with an **EndActionTurnMessage** sent to the current turn player. Sections (2), (3), (4) are repeated for each player.

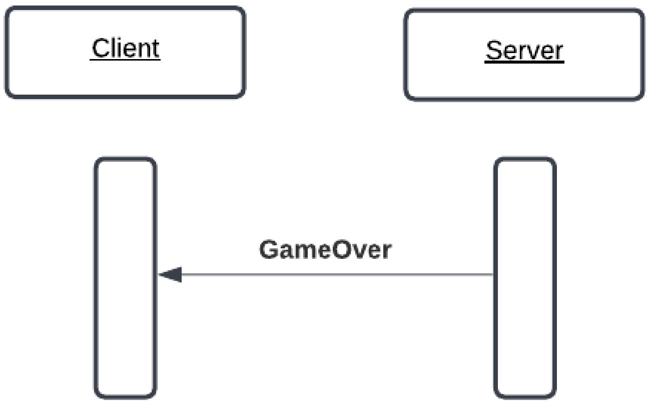


5. In the expert mode each player can activate a character card during the in-game phase. The card effects on the game are variable depending on the card used. **Update** is a general message representing the change on the specific class that the character effects refer to (for example: Board, Game, SchoolBoard etc)

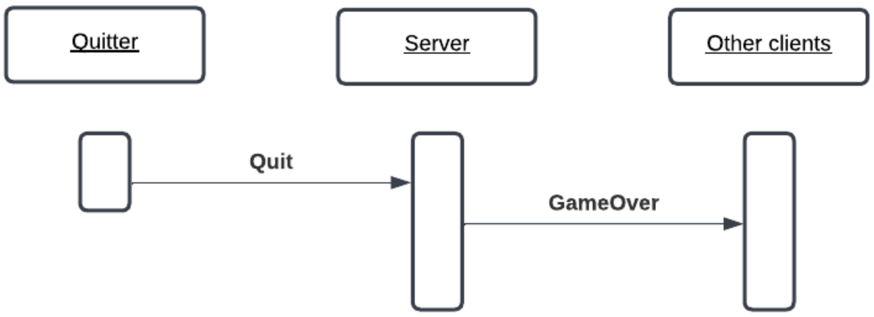


End Of Game Messages

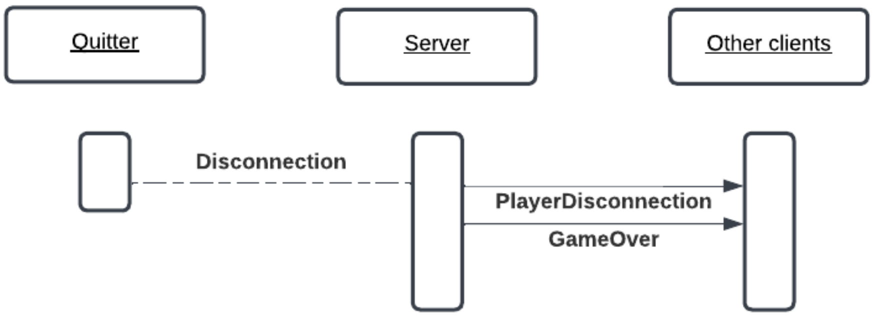
1. A player meets the win condition: a message **GameOver** is broadcasted to every client, it contains the name of the player who won and the reason



2. One player quits: the client sends a disconnection message and the server notifies the other clients that the game is over with a **GameOver** message, the winner and the reason.



3. One player disconnects: a client doesn't respond after the PingTimeout and the server notifies to the others that one client has left the game, and a **GameOver** message to notify the winner and the reason.


Quitter
Server
Other clients
Disconnection
PlayerDisconnection
GameOver