Eryantis Protocol Documentation

Favento, Laini, Macaluso
Group 32

May 17, 2022

1 Messages

1.1 Acknowledgement

Message sent from the client to the server when a generic message has been acknowledged.

Arguments

This message has no arguments.

Possible responses

This message has no responses.

1.2 CommunicationMessage

Message sent from the server to the client to communicate a message. The message is selected from an enum.

Arguments

1. Message: string that can contain a request to the client, or a successful message.

Possible responses

1. Acknowledgement.

1.3 PlayerStatusMessage

Message sent from the server to the client to communicate the status of the player. The status is selected from an enum.

Arguments

1. PlayerStatus: string that contains the current status of the player.

Possible responses

1. Acknowledgement.

1.4 ErrorMessage

Message sent from the server to the client to communicate an error. The error is selected from an enum.

Arguments

1. Error: string that can contain an error.

Possible responses

1. Acknowledgement.

1.5 LoginMessage

Message sent from the client to the server after establishing a connection to create player session.

Arguments

1. Nickname: a string error that contains a nickname for the current player.

Possible responses

1. CommunicationMessage.
2. ErrorMessage.

1.6 SetGame

Message sent from the client to the server communicating the parameters for the creation of a new game.

Arguments

1. NumberOfPlayers: an integer to set the number of players in the game (2/3/4).
2. ExpertMode: a boolean that enables the Expert Mode. If FALSE, the gamemode is set to Default.

Possible responses

1. CommunicationMessage.
2. ErrorMessage.

1.7 JoinAlreadyExistingGame

Message sent from the server to the client communicating the join of an already started game.

Arguments

1. GameInfo: elements that contains the GameID, the number of waiting players, the number of total players and the Game Mode.

Possible responses

1. Acknowledgement.

1.8 WaitingForPlayers

Message sent from the server to the client when creating a new game. The server is waiting for other players to join the game.

Arguments

This message has no arguments.

Possible responses

1. Acknowledgement.

1.9 PlayerDisconnected

Message sent from the server to the client to communicate that a player has disconnected from the game.

Arguments

1. Nickname: nickname of the player disconnected.

Possible responses

1. Acknowledgement.

1.10 MatchStarted

Message sent from the server to the client to communicate the start of the game.

Arguments

1. GameInfo.

Possible responses

1. Acknowledgement.

1.11 AvailableTowers

This message is sent from the server to the client so that the client can see which tower colors are still available.

Arguments

1. AvailableTowers: map containing the number of towers still available for each color.

Possible responses

- ChosenTower.

1.12 ChosenTower

This message is sent from the client to the server to communicate the selected tower.

Arguments

1. Tower: The color of the selected tower.

Possible responses

- CommunicationMessage.
- ErrorMessage.

1.13 AvailableWizards

Message sent from the server to the client to inform the client about wizards still available.

Arguments

1. AvailableWizards: an ArrayList of wizards id.

Possible responses

1. ChosenWizard.

1.14 ChosenWizard

Message sent from the client to the server to comunicate the selected wizard.

Arguments

1. WizardID: an integer that identifies a wizard.

Possible responses

1. CommunicationMessage.
2. ErrorMessage.

1.15 UpdateBoard

Message sent from the server to the client to communicate the latest changes to the gameboard.

Arguments

1. PlayedCharacters: an ArrayList of characters with active effects.
2. Dashboard.

Possible responses

1. Acknowledgement.

1.16 StartofPlayerRound

Message sent from the server to the client to communicate the start of the round.

Arguments

1. RoundNumber: an integer that matches the current round.
2. Nickname: nickname of the active player.

Possible responses

1. Acknowledgement.

1.17 AvailableAssistants

Message sent from the server to the client to inform the player about Assistants still available and assistant played in this turn by other players.

Arguments

1. AvailableAssistants: an ArrayList of assistants of the current player still not played.

2. PlayedAssistants: a Map <Player, Assistant> of the assistant played in this turn by other players.

Possible responses

1. PlayAssistant.

1.18 PlayAssistant

Message sent from the client to the server to play a selected assistant.

Arguments

1. Assistant: the selected Assistant.

Possible responses

1. CommunicationMessage.
2. ErrorMessage.

1.19 MovableStudents

Message sent from the server to the client to give the list of all movable students.

Arguments

1. EntranceStudents: ArrayList of students containing the movable students from the entrance.

Possible responses

1. ChosenStudent.

1.20 ChosenStudent

Message sent from the client to the server to select the student from the list provided by the server.

Arguments

1. Student: student to move.

Possible responses

1. WhereToMove.
2. ErrorMessage.

1.21 WhereToMove

Message sent from the server to the client to choose where to move the student.

Arguments

1. DiningRoom: a boolean that enable the possibility to choose the DiningRoom as destination.
2. Islands: an integer that represents the number of islands. The player can choose a number between 1 and 'Islands'.

Possible responses

1. ChosenDestination.

1.22 ChosenDestination

Message sent from the client to the server to select the destination from the list provided by the server.

Arguments

1. DiningRoom or Island.

Possible responses

1. CommunicationMessage.
2. ErrorMessage.

1.23 MotherNatureSteps

Message sent from the server to the client to ask for Mother Nature movement.

Arguments

1. MaxStepsAllowed: an integer between 1 and the number on the assistant card (can be incremented by some Characters).

Possible responses

1. ChosenSteps.

1.24 ChosenSteps

Message sent from the client to the server to select the number of steps from the options provided by the server.

Arguments

1. Steps: an integer with the chosen number of steps.

Possible responses

1. CommunicationMessage.
2. ErrorMessage.

1.25 IslandOwner

Message sent from the server to the all clients to provide the nickname of the island owner after calculating the influence.

Arguments

1. Island: the island with MotherNature.
2. Nickname: the nickname of the owner.

Possible responses

1. Acknowledgement.

1.26 SelectCloud

Message sent from the server to the client to choose between clouds.

Arguments

1. AvailableClouds: an ArrayList of Clouds containing the available clouds.

Possible responses

1. ChosenCloud.

1.27 ChosenCloud

Message sent from the client to the server to select the cloud from the list provided by the server.

Arguments

1. Cloud: the chosen cloud.

Possible responses

1. EndOfPlayerRound.
2. ErrorMessage.

1.28 AvailableCharacters

Message sent from the server to the client to show the available characters and their cost.

Arguments

1. Characters: an ArrayList containing the available characters.

Possible responses

This message has no responses.

1.29 UseCharacterEffect

Message sent from the client to the server to use the effect of a Character on the same turn.

Arguments

1. Character: the chosen character.

Possible responses

1. CommunicationMessage.
2. ErrorMessage.

1.30 EndOfPlayerRound

Message sent from the server to the client to communicate the end of the player actions for the current round.

Arguments

1. RoundNumber: an integer that matches the current round.
2. Nickname: nickname of the active player.

Possible responses

1. Acknowledgement.

1.31 EndOfRound

Message sent from the server to the client to communicate the end of the round for all the players

Arguments

1. RoundNumber: an integer that matches the ended round.

Possible responses

1. Acknowledgement.

1.32 CommunicateWinner

Message sent from the server to the client to communicate the winner of the the game.

Arguments

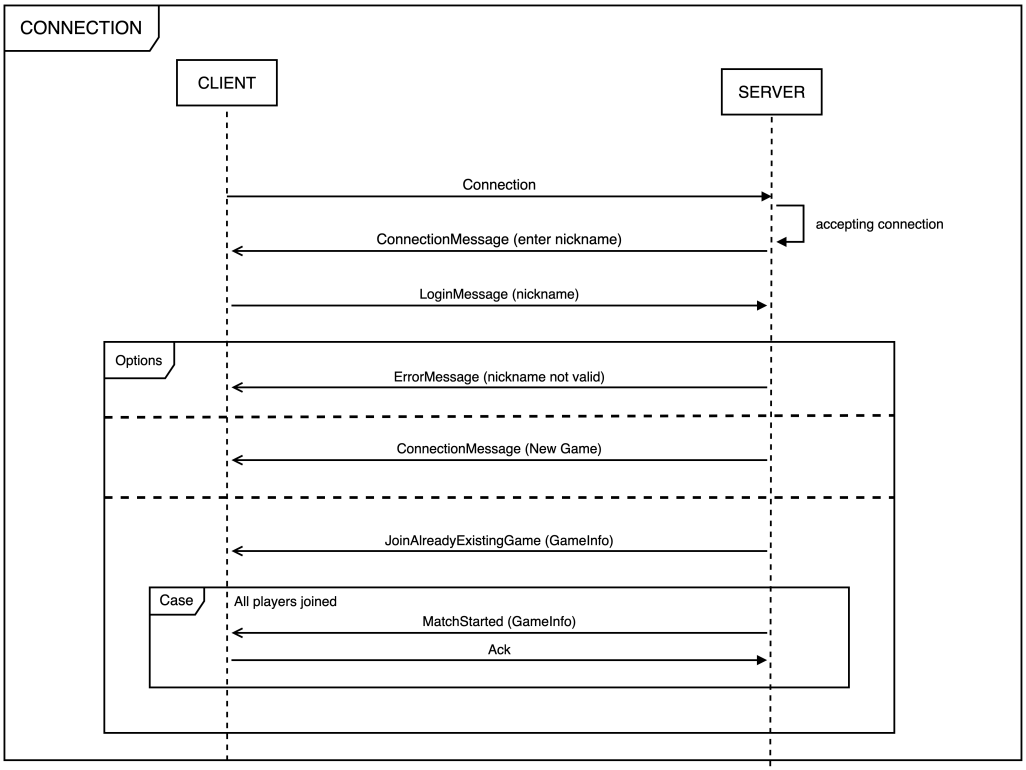
1. Nickname: the winner player nickname.
2. WinExplained: a short message to explain the reason why the player has won.

Possible responses

1. Acknowledgement.

2 Scenarios

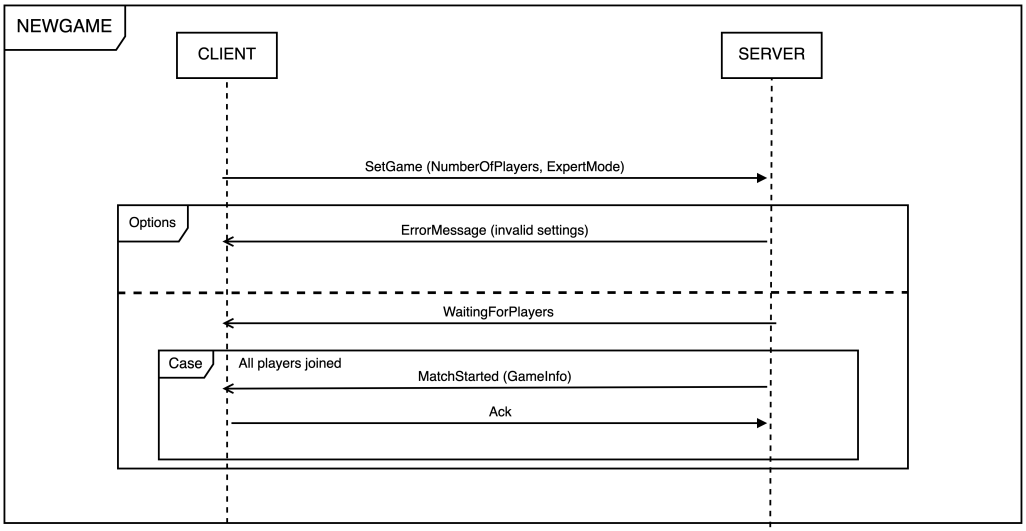
2.1 Connection



After the connection between client and server has been established, the server sends a ConnectionMessage to the client asking him to enter his desired nickname. After the client replies with a Login message containing his chosen nickname, there are 3 possible outcomes: if the nickname is invalid or has already been taken by another user, the server sends an ErrorMessage explaining why the nickname wasn't accepted, and the client will be prompted to enter another nickname.
If the nickname was valid the server saves it and, if the client is the first in the lobby he receives a new game message asking him to enter his game preferences, otherwise he will join the waiting list and will receive a JoinAlreadyExistingGame message containing the informations about the game (number of players, game mode, current players waiting).
As soon as the lobby is filled, the match can start and all the clients will receive a MatchStarted message, containing the number of players, the game

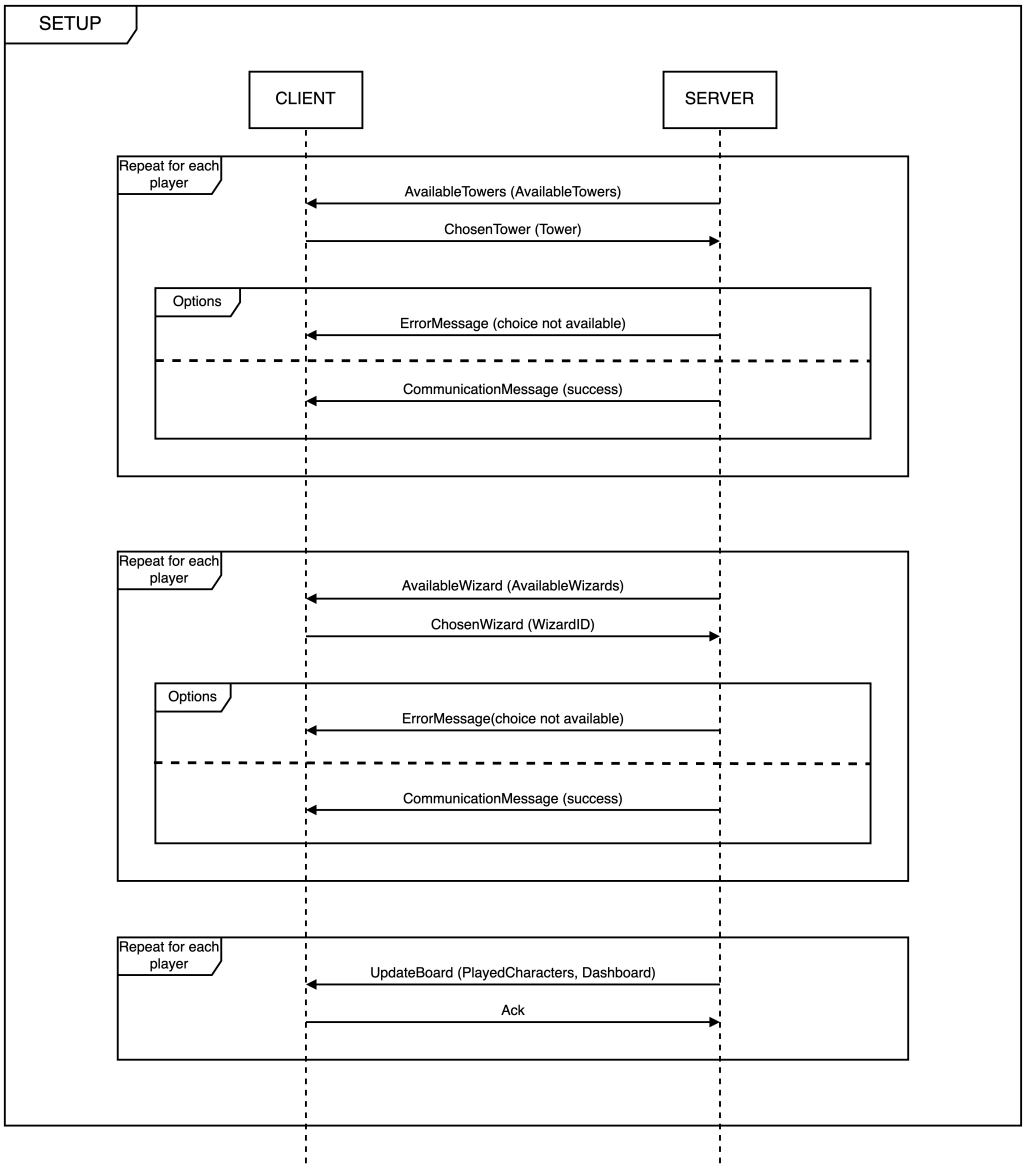
mode and the list of players.

2.2 New game



If the client was the first in the lobby, he has to set his preferences for the next starting game through a SetGame message containing the desired number of players and game mode. In case the proposed settings are not valid the server sends an ErrorMessage, otherwise the server sends a WaitingForPlayers message communicating that the options have been set and the server is waiting for more players to join the game. After there are enough players to start the game, the server sends a MatchStarted containing the list of the players, other than the game mode and the number of players previously set.

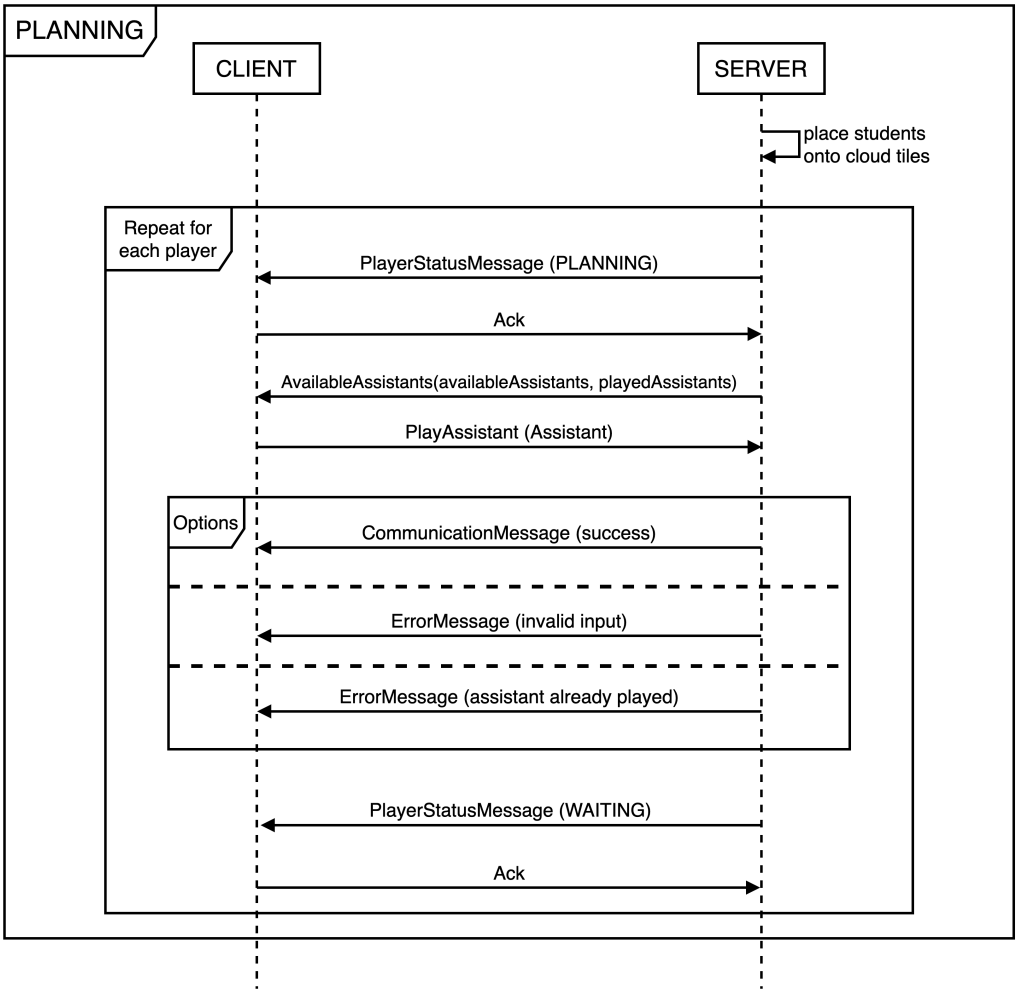
2.3 Setup



After the match has been initialized the server sets up the table according to the number of players. After the basic elements have been placed on the table, the server sends all players (one at a time) an AvailableTowers message containing a map with the number of places left for every tower color. Each client will send a ChosenTower message containing the chosen

tower color, after which the server will send either an ErrorMessage if the choice wasn't available or a CommunicationMessage stating that the choice has been registered.
The same process applies when the clients have to choose the wizard id: the server proceeds once one at a time sending an AvailableWizards message to all clients containing a list of the available wizards, after which the client replies with a ChosenWizard message with the id of the chosen wizard. Even in this case, the server can send an ErrorMessage or a CommunicationMessage confirming the operation.
At last, when the server completes the setting up of the board, it will send to each client an UpdateBoard containing the resulting dashboard, which will receive an Acknowledgement message as an answer. Later, the server will send a StartOfPlayerRound message, containing the number of the round (1 in this case) and the nickname of the first player (chosen randomly), which will need an Acknowledgement message as an answer.

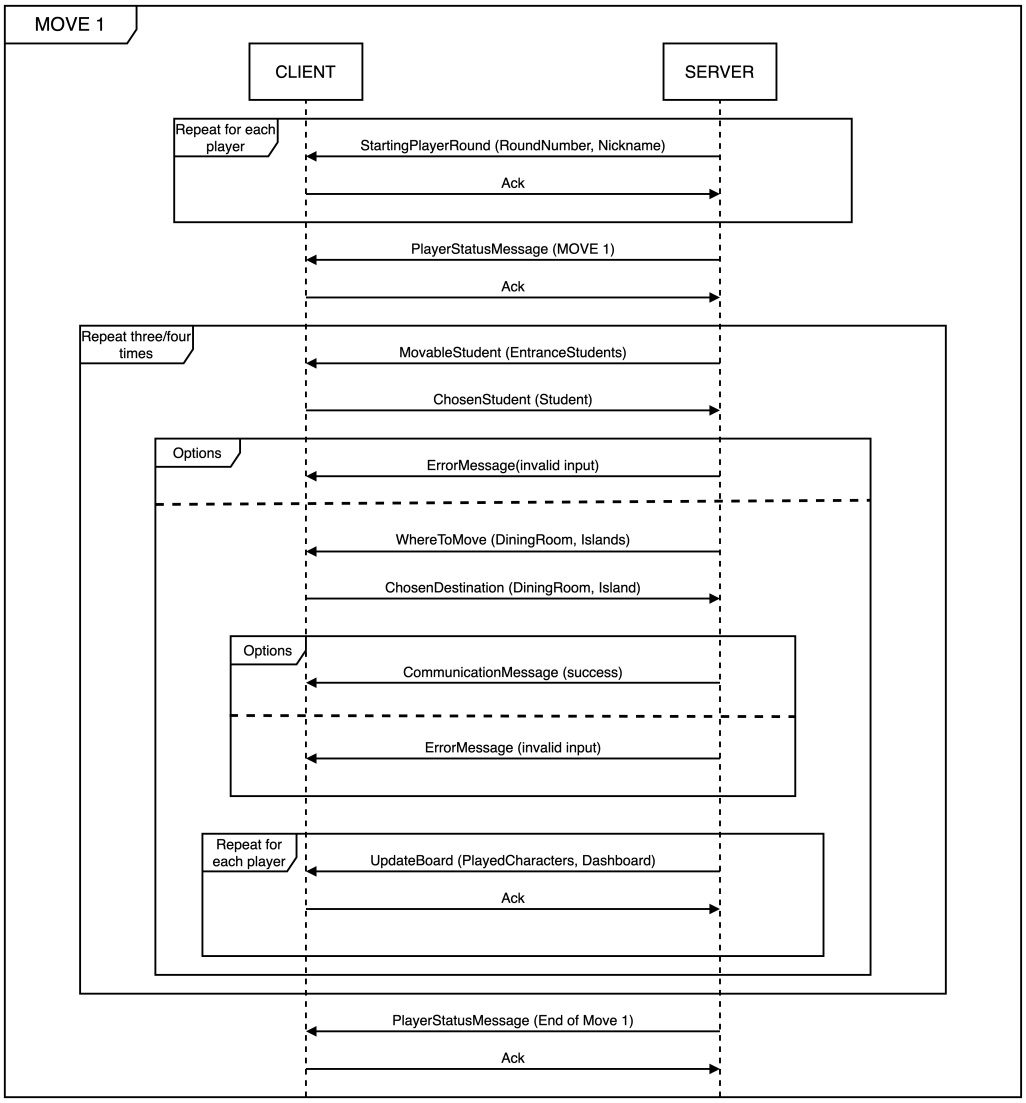
2.4 Planning Phase



When the Planning Phase starts, at first the server will add the students onto the cloud tiles, completing the first part of the Phase. After that, the server will repeat the following procedure for each player in the game, mantaining the order of the previous turn (for the first planning phase of the game the order is random): the status of the player is set to 'Planning', receiving an Acknowledgement message as answer; than the server sends the list of available assistant and a map in which every played assistant is followed by the nickname of the corresponding player. Client replies with a message containing the selected assistant. In case of error, the server can send an

error message to inform the player that the selected assistant has already been played, or a generic 'invalid input' message. If everything is ok the server provides a 'Success' message. At the end the server sets the status of the player to 'Waiting', receiving an Acknowledgment message as answer. Now the Planning Phase is over. The server calculates the order of the turn and then moves to the Move1.

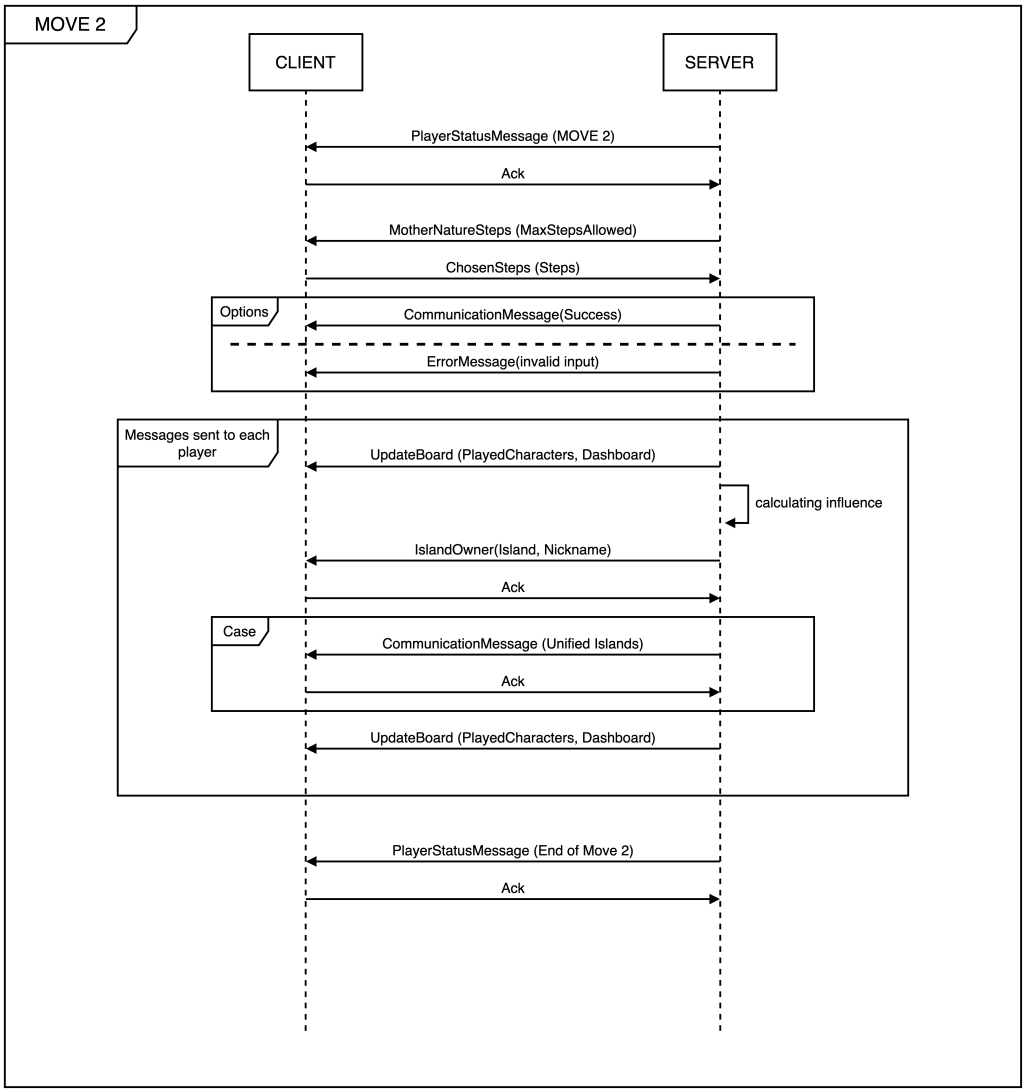
2.5 Move Phase, Move1



When the action phase starts the server communicates to every client the first player, based on the minor assistant value, and the round number. Then the server sets the first player status to 'Move 1', receiving an Acknowledgment message as answer. For three or four times in a row, based on the number of players in the game, the server follows this pattern: at first sends to the client the list of the students placed in the entrance of the player schoolboard. The client provides the choice sending back the selected

student. If there's no error in the client response,the server sends back a 'Success' message, and then asks where to move the student, providing the number of remaining islands and the option for the dining room. The client provides a response with the player choice. If there's no error and after the 'Success' message, the server updates the board for every player to show the movement of the student. At the end of the three times repetition, the server sets the player status to 'End of Move 1', receiving an Acknowledgment as answer. Now it's time for the second move.

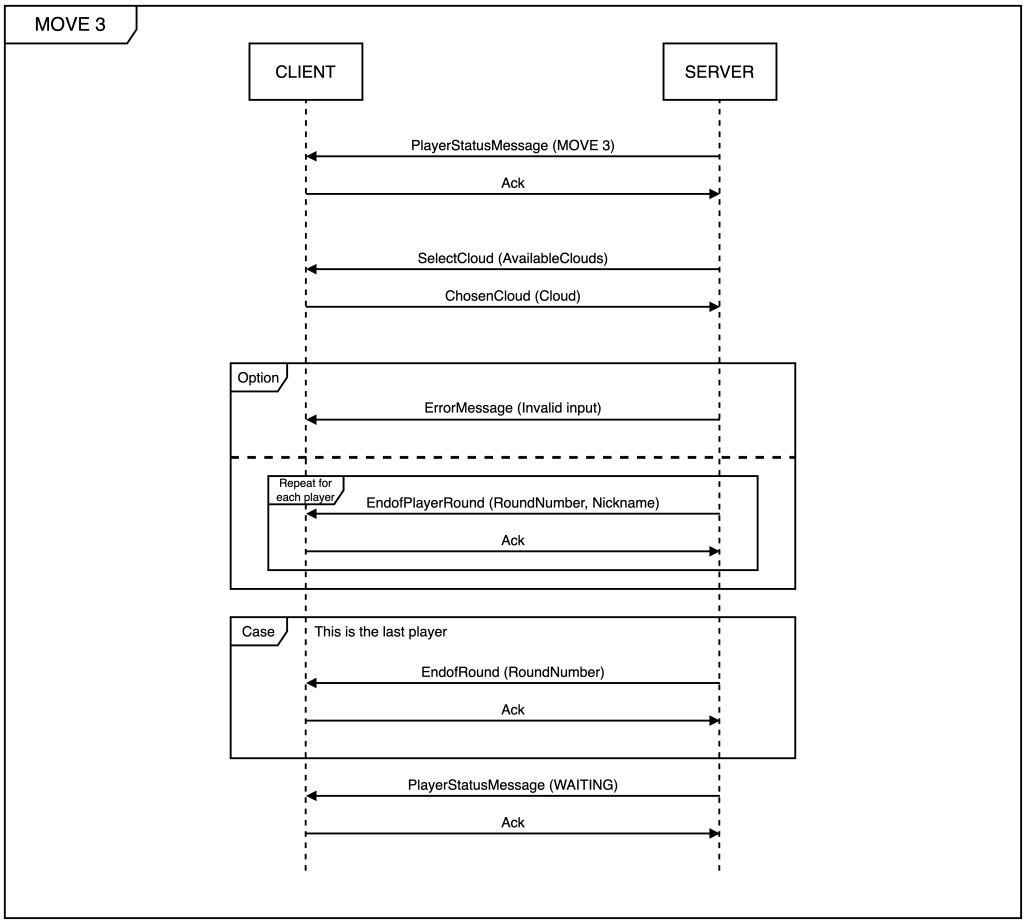
2.6 Move Phase, Move2



The server sends a PlayerStatusMessage to the client that just completed the first move, informing him that the 'Move 2' is about to begin, and expecting an Acknowledgement message as an answer. After this, the server sends a MotherNatureSteps message containing an integer representing the maximum number of steps allowed for mother nature to move. The client will subsequently reply with a ChosenSteps message containing an integer between 1 and the number previously communicated by the server. If the

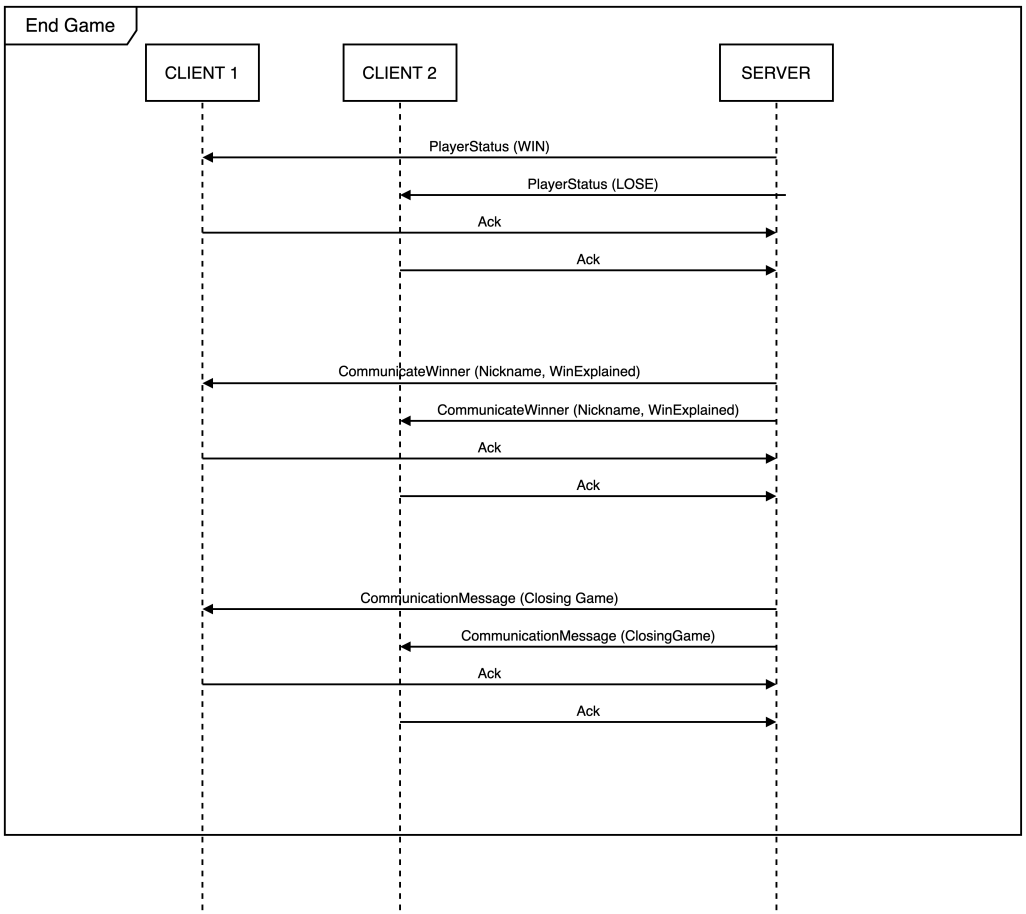
client provides a valid input, he will receive a 'Success' message, otherwise if the input is not valid an ErrorMessage will be sent. Right after this, the server will communicate to all the clients about the state of the board through an UpdateBoard message, and will then calculate the influence over the island over which mother nature landed. The server will then send an IslandOwner message communicating the owner of the island where the influence was calculated, and will expect an Acknowledgement as an answer. If the influence calculation caused a unification of the islands, every client will receive a CommunicationMessage that will inform about this event, and will reply with an Acknowledgement message. The server thus will send an UpdateBoard message to all clients with the new state of the board. This concludes the Move 2 hence the server will inform the current player about the end of the Move 2 through a CommunicationMessage, and will receive an Acknowledgement message as a confirmation of the correct receipt.

2.7 Move Phase, Move3



The third move phase starts with the server setting the player status to 'Move 3', receiving an Acknowledgement message. The server provides a list of available clouds to the client, so the player can choose the cloud seeing the student on it. The client communicates the choice sending the selected cloud back to the server, which verifies the absence of errors in the answer. At this point the server informs every player about the end of the current player round. If this player was the last for this turn, the server sends a message to every client notifying the end of the round. In every case, at the end of the third move, the server sets the current player status to 'Waiting'.

2.8 EndGame Phase



The server sends to each player a PlayerStatus message, containing the status of each player (WIN or LOSE). Every player responds with an Acknowledgement message. The server then communicates to each player the nickname of the winner and the reason why that player won the game. Every player responds with an Acknowledgement message. Finally, the server sends to each player a message that communicates the finish of the current game.