Eriantys Protocol Documentation

Giorgio Miani, Alessia Porta, Andrea Verrone

Gruppo 31

Messages

Generic messages

These messages are sent both from client and server to the counterpart.

ResponseMessage

This message is sent after receiving a message to indicates if the action requested was handled successfully or not.

Arguments

- parentMessage: a reference to the message this response is for
- result: if the response indicates a success or a failure
- errorCode: the code of the error if this response indicates a failure

Possible responses

This message has no responses.

PingMessage

This message is sent to check if the connection between client and server is still alive.

Arguments

This message has no arguments

Possible responses

This message has no response

From client to server

These messages are sent from client to the server to indicate that the client wants to perform some action. All these messages receive as a response ResponseMessage with the result of the requested action and eventually an error code if the action was wrong.

CreateNewGame

This message is sent from the client to the server when a user wants to start a new game.

Arguments

- numOfPlayers: the number of players requested to participate in this game
- wantExpert: if the game needs to use the expert rules

GetGames

This message is sent from the client to the server when a user wants to see the existing games to join one.

Arguments

This message has no arguments

EnterGame

This message is sent from the client to the server when a new player wants to enter a game.

Arguments

- gameID: the identifier of the game in which the player wants to enter
- nickname: the nickname of the user

ResumeGame

This message is sent from client to server when a user wants to resume a game he was playing.

Arguments

This message has no arguments

SendUserIdentifier

This message is sent from client to server to send the unique identifier of a user.

Arguments

- Identifier: a unique identifier of the user

ChangeNumPlayers

This message is sent from the client to the server when a user wants to change the number of players requested for a game.

Arguments

- newNumPlayers: the new number of players

SetTower

This message is sent from the client to the server when a player wants to change his tower.

Arguments

- Tower: the new tower to be set

SetWizard

This message is sent from the client to the server when a player wants to change his wizard.

Arguments

- wizard: the new wizard to be set

NextPhase

This message is sent from the client to the server when a player wants to move the matchmaking to a new phase (i.e., from the lobby to the choose of towers and wizard or from the choose of a player to the next).

Arguments

This message has no arguments

UseAssistant

This message is sent from the client to the server when the player uses the assistant card.

Arguments

- assistant: the assistant card to be played by the player

ChooseStudentFromLocation
This message is sent from the client to server to choose a student to move.
Arguments
- student: the student chosen
- originPosition: the position in which the student was previously

ChooseDestination
This message is sent from the client to the server to choose a destination on which to operate.
Arguments
- destination: the chosen destination

MoveMotherNature
This message is sent from a client to the server to indicate that he wants to move mother nature.
Arguments
- movement: The number of movement mother nature need to do

TakeStudentsFromCloud
This message is sent from a client to the server after the client has chosen a cloud from where take all the students and put them in his entrance.
Arguments
- cloudID: ID of the cloud chosen by the client

UseCharacterCard
This message is sent from the client to the server to indicate that a user wants to use a character card.
Arguments
- card: the card chosen

QuitGame
This message is sent from the client to the server to indicate that a user wants to exit a game.
Arguments
This message has no arguments

From server to client
These messages are sent from the server to a client (or all the client associated to the same game) to send information that he asked or to indicate that a modification in the game occurred.
For all these messages a ResponseMessage with a success result is expected as a response to the message to indicate that the message has been received from the client.

PlayerOrStateChanged
This message is sent from the server to all clients connected to a game to indicate that the current player or the state of the game has changed.
Arguments
- currentPlayerNickname: the nickname of the current player
- currentState: the current state of the game

PlayerLeftGame
A message sent from server to all client connected to a game to notify that a player left the game.
Arguments
- nicknamePlayer: the nickname of the player that left

UpdateNicknameGameID
A message sent from server to a client that asked to resume a game to send him his nickname and ID of the game he was playing.
Arguments
- nickname: the nickname of the player
- gameID: the id of the game

GameCreated
This message is sent from the server to a client when the game he requested has been created and a lobby is ready to accept players.
Arguments
- gameID: the identifier of the game created

PossibleGames
This message is sent from the server to the client when a user asks to see the game that he can join.
Arguments
- gameIDs: the IDs of all the game that are currently created and not already started

GameEntered
This message is sent from the server to the client when the user enters a game.
Arguments
- playerLoginInfos: a list of all the players in the game
- numPlayers: the number of players requested to participate in this game
- isExpert: if this game uses the expert rules
- currentPlayer: the nickname of the current player

NumPlayersChanged
This message is sent from the server to the client when the number of players requested to join a game has been changed.
Arguments
- newNum: the new number of players

PlayersChanged
This message is sent from the server to the client when the players list of a game has changed.
Arguments
- players: the players currently in this game

ChoosePlayerParameters
A message sent from the server to all clients to indicate that an action is needed to choose the parameters of the current player in the matchmaking.

Arguments

- towersAvailable: a list of tower's type from which the current player can choose
- wizardsAvailable: a list of wizards from which the current player can choose

TowerSelected
This message is sent from the server to the client when the tower of a player has changed

Arguments

- nickname: the nickname of the player
- tower: the new tower of the player

WizardSelected
This message is sent from the server to the client when the wizard of a player has changed

Arguments

- nickname: the nickname of the player
- wizard: the new wizard of the player

TableCreated
This message is sent from the server to a client that connects to a game to send him all the information about the current state of the game.

Arguments

- table: a reduced version of the model of the game

AssistantUsed
This message is sent from the server to all clients after a player uses an assistant card.

Arguments

- nickname: the nickname of the player
- Assistant: the assistant card played by the current player

DeckChanged
A message sent from the server to a client to indicate that the assistants in his hand changed.

Arguments

- nickname: the nickname of the player
- assistantsList: a list of assistants that he has in his deck

StudentsOnEntranceChanged
This message is sent from the server to all clients after the students in the entrance of a player has changed.

Arguments

- Player: the nickname of the player in which the change happened
- studentList: the students that are now present in the entrance of that player

StudentsInDiningRoomChanged
This message is sent from the server to all clients after a change in the students in the dining room of the current player.

Arguments

- Player: the nickname of the player in which the change happened
- studentList: the students that are now present in the dining room of that player

StudentsOnIslandChanged
This message is sent from the server to all clients after the students on an island change.

Arguments

- StudentList: the new list of students on that island
- ID: the island on which the student has been placed

StudentsOnCloudChanged
This message is sent from the server to all clients after the students on a cloud have been changed (i.e., moved to the current player's entrance).

Arguments

- ID: ID of the cloud on which the students have been changed
- studentList: the new list of students on that cloud

ProfessorChanged
This message is sent from the server to all clients after the current player moves a student from the entrance of his school board to its dining room and the list of professors has been changed.

Arguments

- nickname: nickname of the player for which the professor list has been changed
- professors: the new professor list of the player

CoinOfPlayerChanged
This message is sent from server to all client connected to a game to indicate that the coins of a player has been changed.

Arguments

- player: the nickname of the player
- actualNumOfCoins: the number of coins that he has

CoinInBagChanged
This message is sent from server to all client connected to a game to indicate that the coins in the general bag has been changed.

Arguments

- numberOfCoinsInBag: the number of coins in the bag

MotherNatureMoved
This message is sent from the server to all clients after the current player has correctly moved mother nature and the model has been updated.

Arguments

- position: the new position of mother nature

TowerOnIslandChanged
This message is sent from the server to all clients after the tower on the island where mother nature is positioned has been changed. If not already present, a new tower has been positioned
Arguments
- newTower: color of the new tower positioned
- islandID: ID of the island where the tower has been changed

TowerNumberChanged
This message is sent from the server to all clients after the number of towers of a player has changed.
Arguments
- player: the nickname of the player
- actualNumOfTowers: the number of towers the player has

IslandsUnified
This message is sent from the server to all clients after two islands have been.
Arguments
- IDIslandToKeep: ID of the island unified that needs to remain on the table
- IDIslandRemoved: ID of the other island unified that needs to be removed from the table
- sizeIslandRemoved: the size of the island that needs to be removed

BanOnIslandChanged
This message is sent from server to all client connected to a game to indicate that the number of bans on an island has been changed.
Arguments
- islandID: the id of the island on which the change has happened
- actualNumOfBans: the number of bans on that island

CoinOnCardAdded
This message is sent from server to all client connected to a game to indicate that a coin was added on a character card.
Arguments
- card: the card where the coin was added

StudentsOnCardAdded
This message is sent from the server to all clients after the students on a character card change.
Arguments
- StudentList: the new list of students on that card
- card: the card on which the student has changed

LastRound
This message is sent from the server to all clients if the game is in its last round
Arguments
This message has no arguments.

GameEnded

This message is sent from the server to all clients to notify that the game has ended and tell the winner of the game

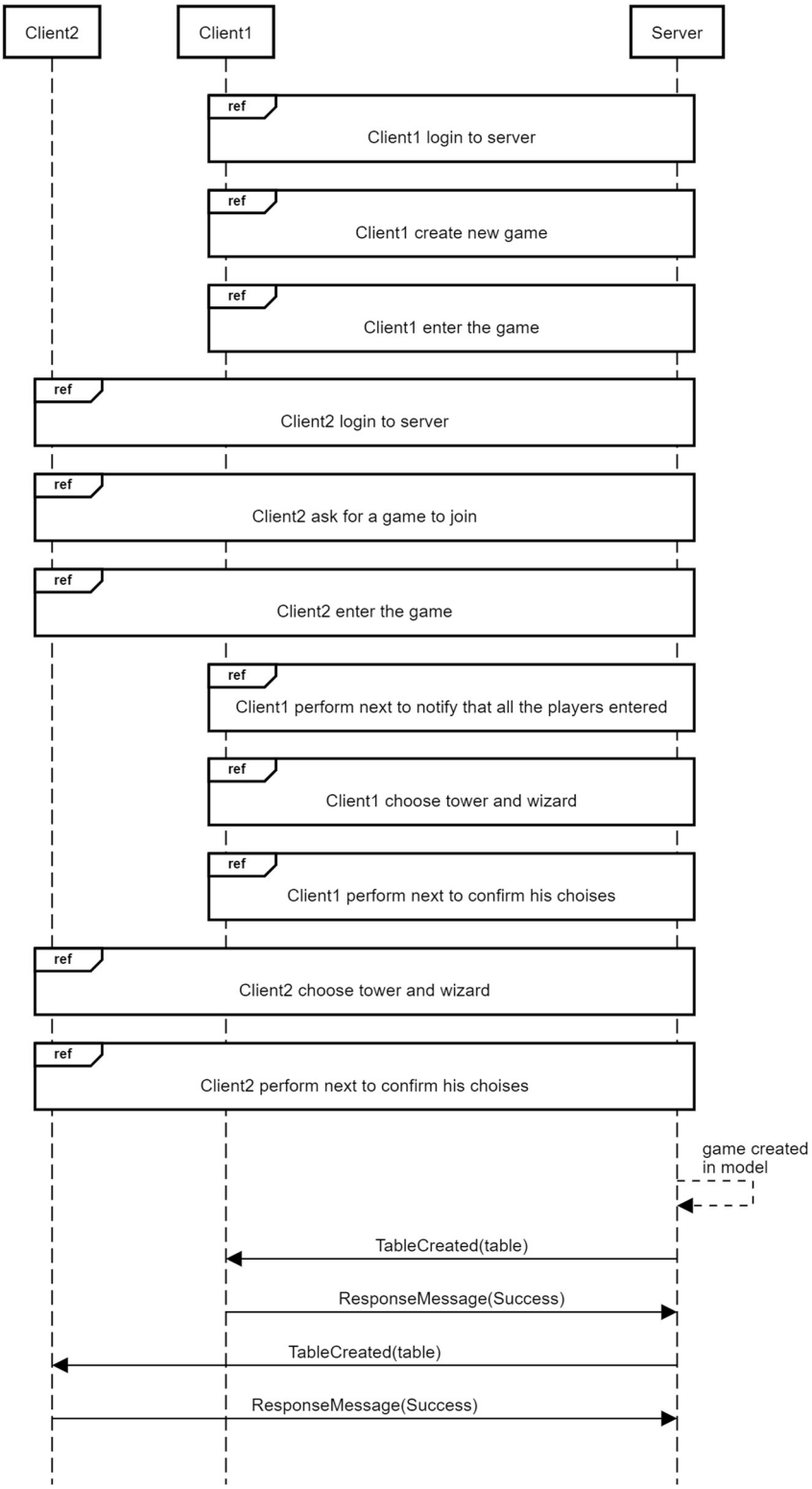
Arguments

- winners: a list of the nickname of the winner/s

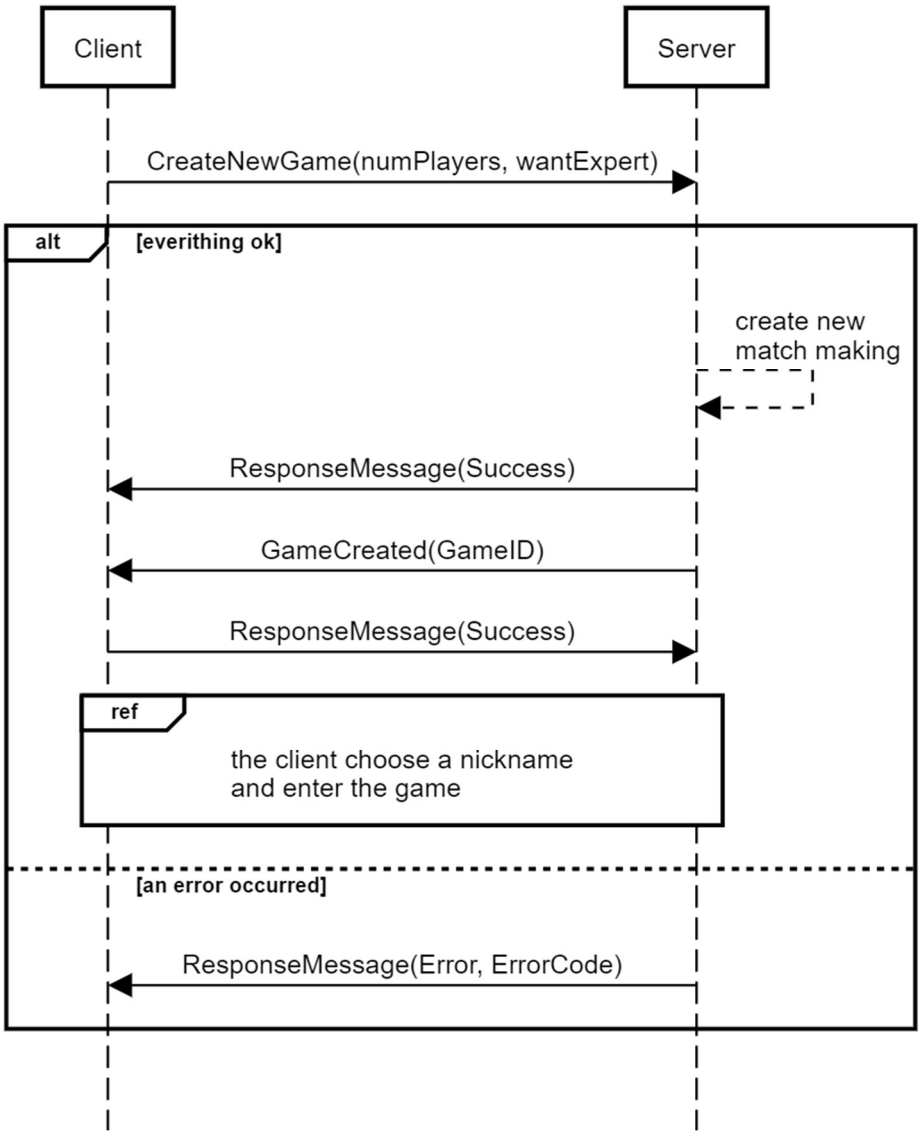
Scenarios

Creation and set ups of the game

Here is the sequence diagram simulating the creation of a game between two players. All the ref actions are described more precisely later. After all the set ups are done, the server creates the game in the model and send a TableCreated message to the clients sending them all the information of this game.

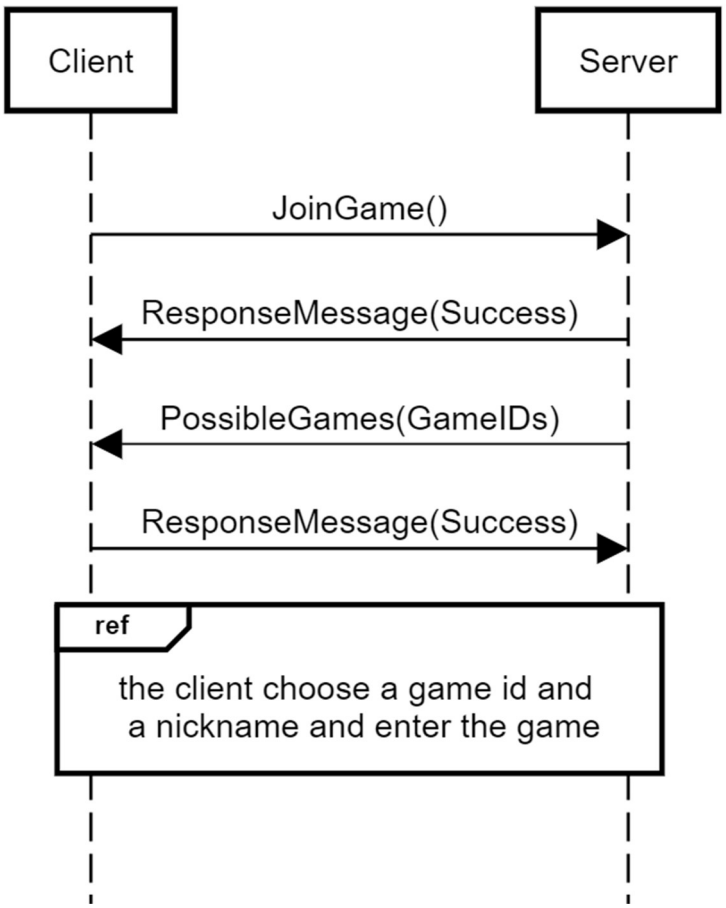


Creating a new game



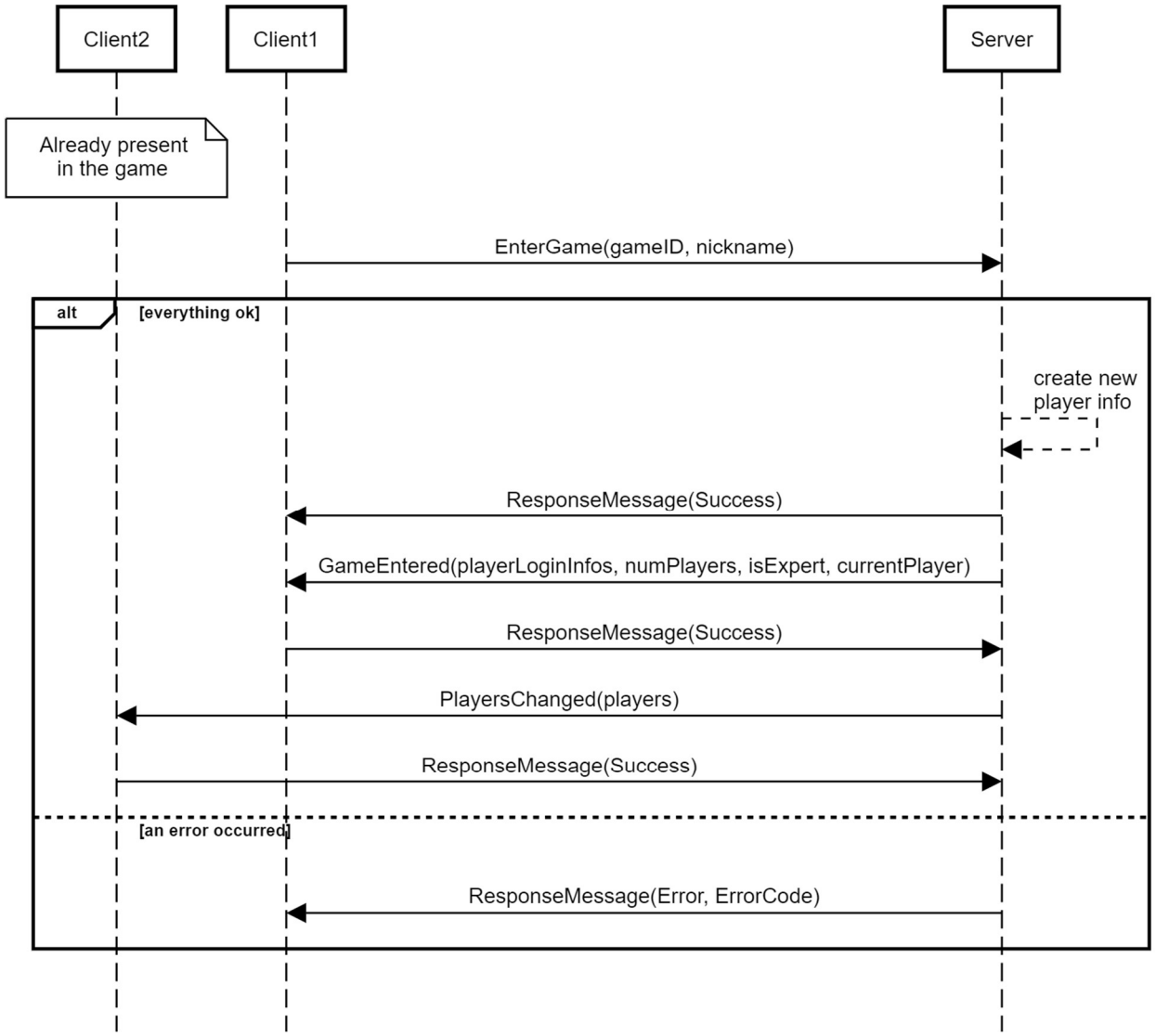
When a client opens the game, he can choose to create a new game and the CreateNewGame message will be sent with the number of requested players chosen by the user and if the game should use the expert mode. The number of players requested should be one of the supported value or the server will respond with an Error ResponseMessage. If the number is correct instead it responds with a success and send a GameCreated message giving the client the id of the game created. After this the user will be prompted to choose a nickname and this with the id given by the server will be used to make the client join the game he created.

Joining an existing game



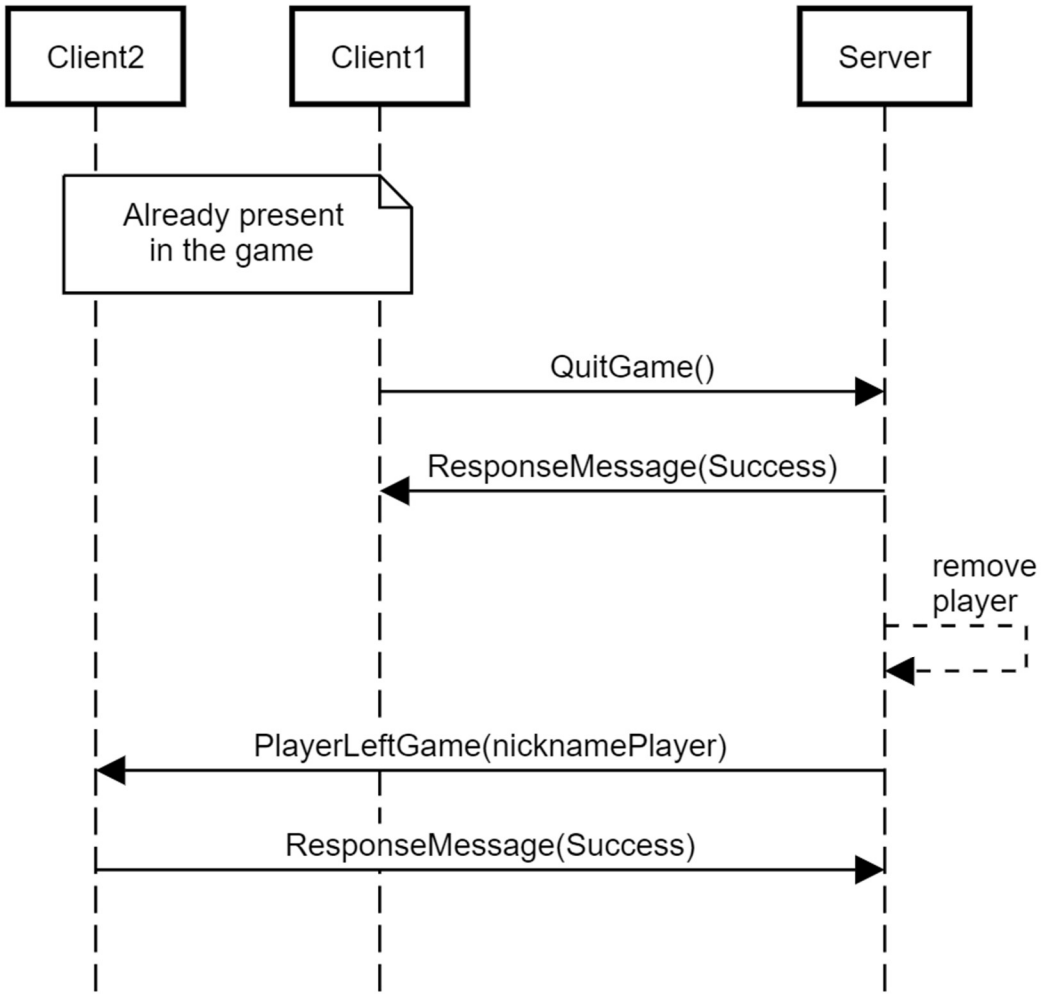
When a client opens the game, he can choose to join an existing game and the JoinGame message will be sent to the server. It will respond with PossibleGames giving the client the list of IDs of games that are in the initial phase of the matchmaking, where new players are allowed to join the game. After this the user will choose a game to join and a nickname and these two elements will be used to make the client join the game he selected.

Entering a game



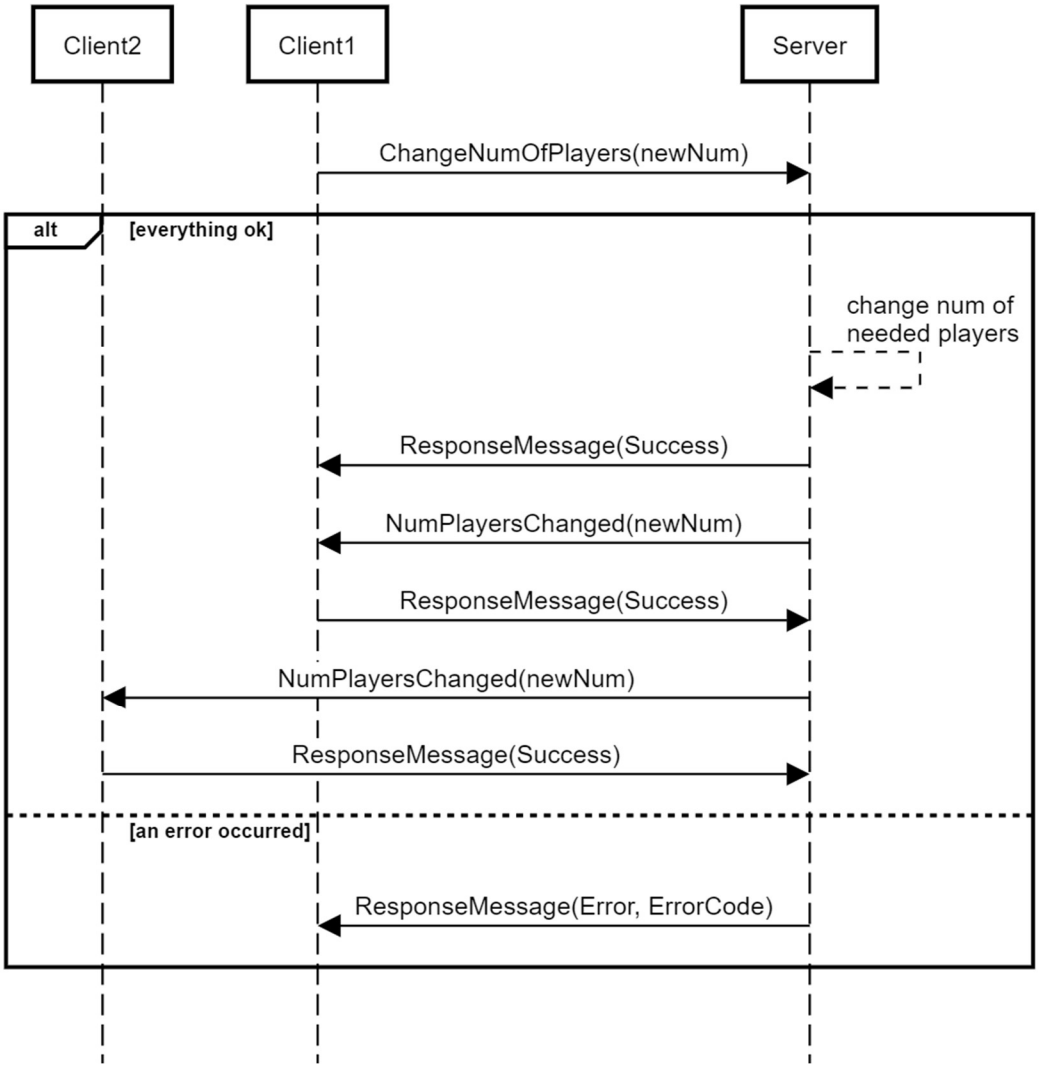
When a game is created, the server waits for clients to send EnterGame messages. Each time the server receives this message, it could respond with an error ResponseMessage if the gameID that he received doesn't correspond to any available game or if there is already a player with the same nickname. If everything goes fine instead the server will send a ResponseMessage indicating the success and a GameEntered. In the last case the server will create a new player in its internal state and send a PlayersChanged message to all clients connected with that game, notifying that the players in the lobby are changed. Each client will respond to that message with an ResponseMessage to the server.

Exiting from a game



A client connected to a game can send a QuitGame message to communicate the server his intention to leave the game. When the server receives this message, it will remove the player from the game, sending a successful ResponseMessage message to the client that made the request and a PlayerLeftGame message to any other client related to that game, if any. If there are no more players left in the game, the server will also delete the game.

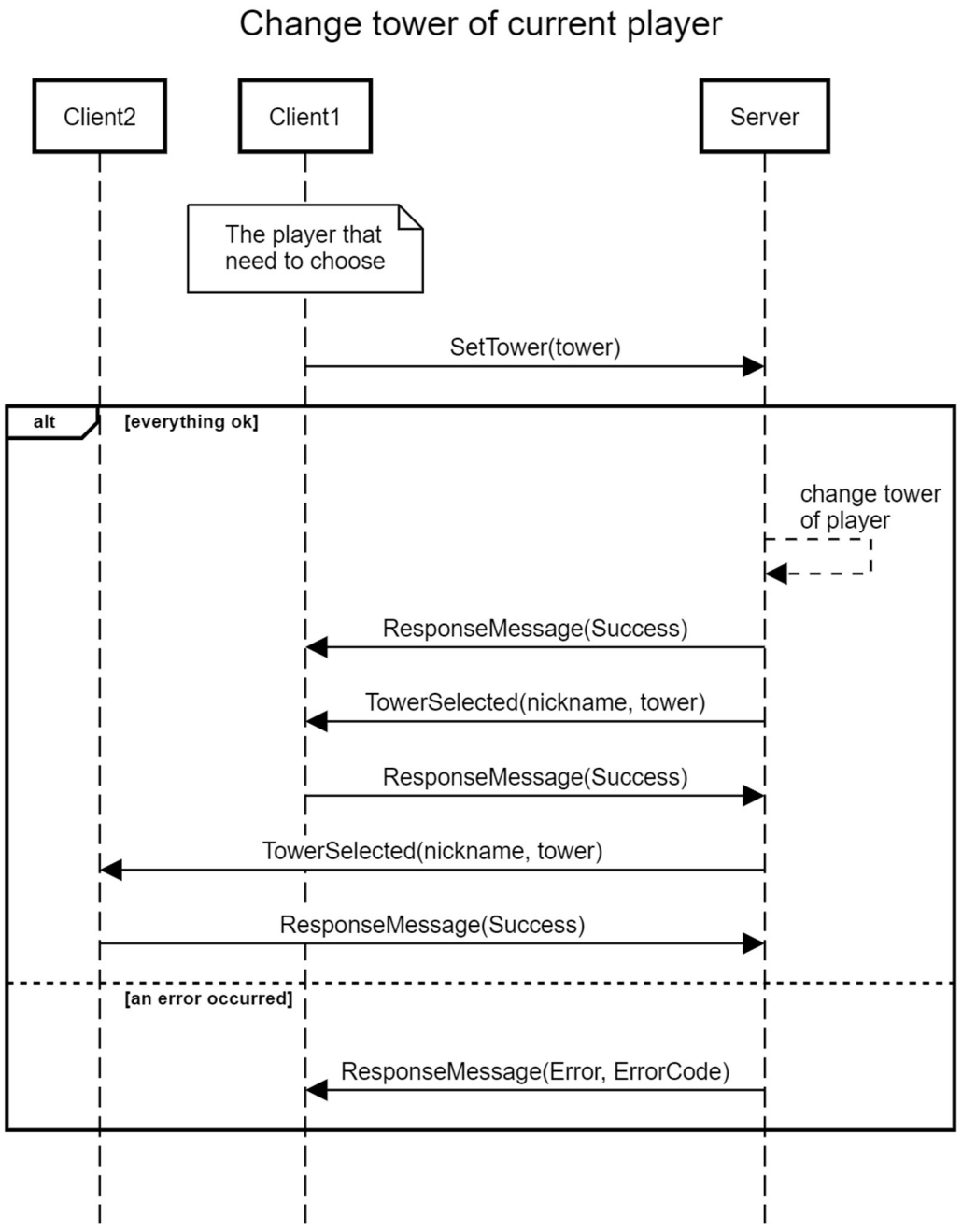
Changing the number of players



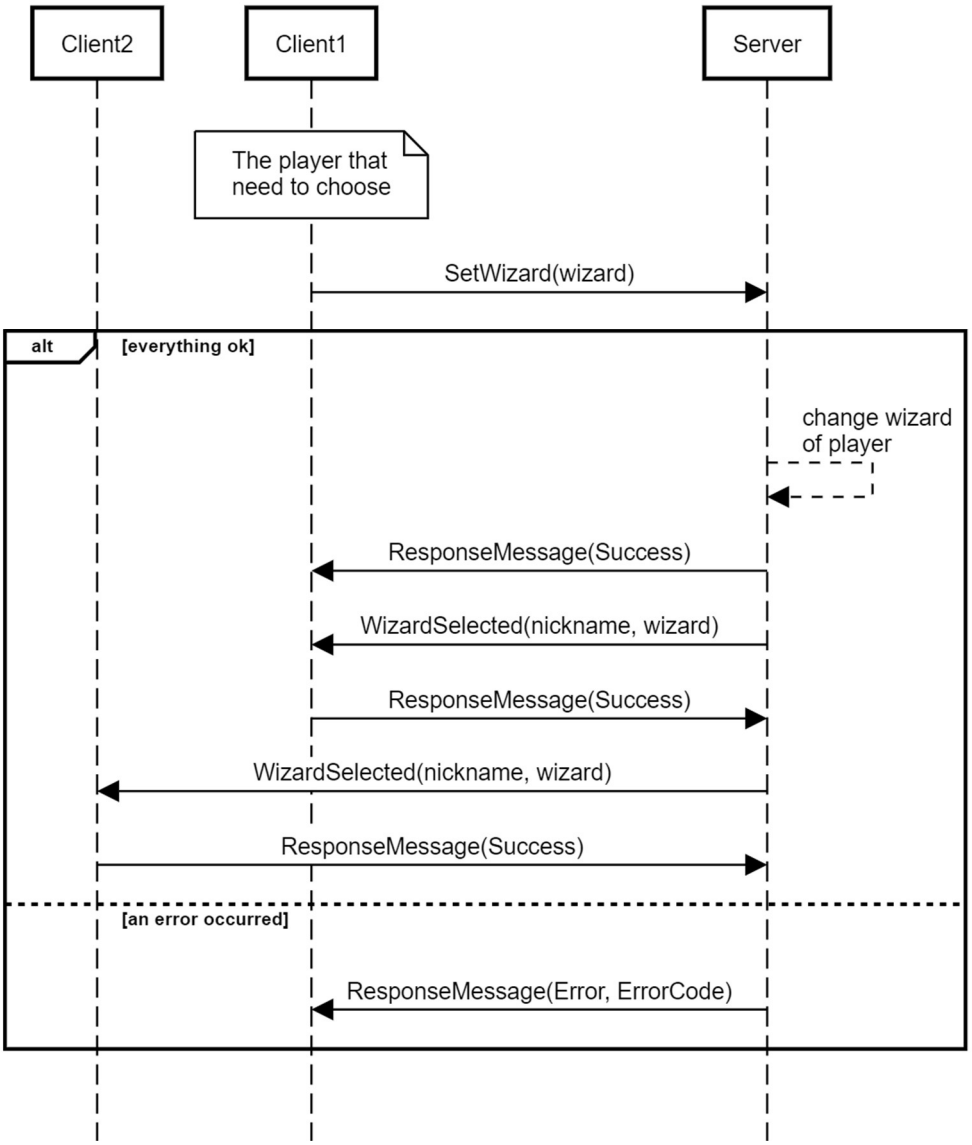
When in the first phase of the matchmaking, a player can send a ChangeNumPlayers message to the server to indicate his willingness to change the number of players needed in this game. If the number requested is not valid, for example if it's not one of the possible supported value or if it's less than the number of players currently in the game, the server respond with an error ResponseMessage. If the value is correct, the server changes the number and notifies all the clients connected to that game with NumPlayersChanged.

Changing the attributes of a player

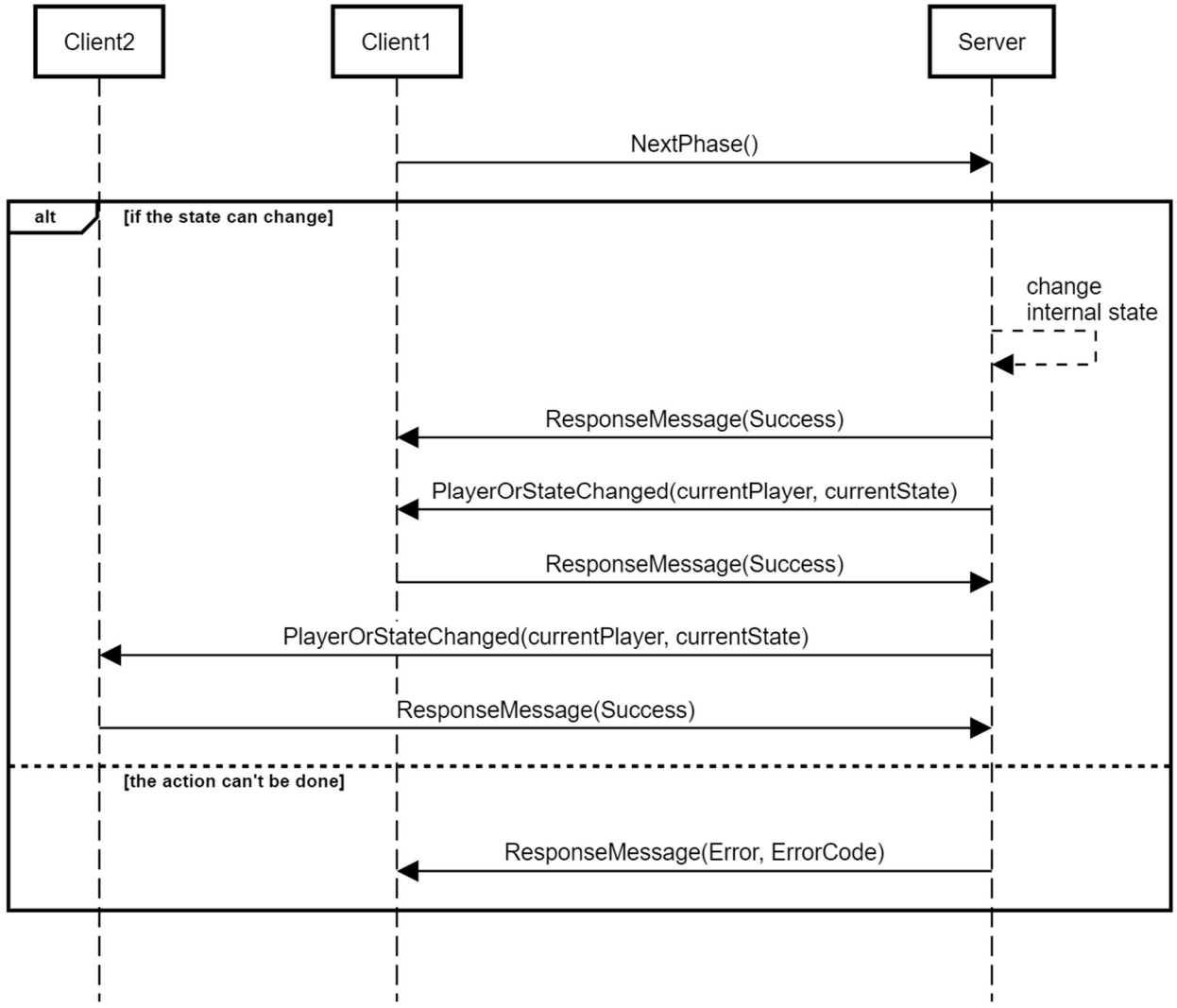
After all the needed players has joined a game, one by one they will be asked to choose a color for the tower and a wizard to use during the game. These two actions can be done using the SetTower and SetWizard messages, in any order and multiple times in the same turn. If any of these messages arrives to the server in an invalid state, for example if the game is in the first phase where new players are expected to join, or if the corresponding argument (a tower for SetTower and a wizard for SetWizard) cannot be chosen, for example because they're already chosen by another player, the server will respond with an error message. If everything is fine, the server will change the tower/wizard of the current player and notify all the player of the change with TowerSelected/WizardSelected.



Change wizard of current player

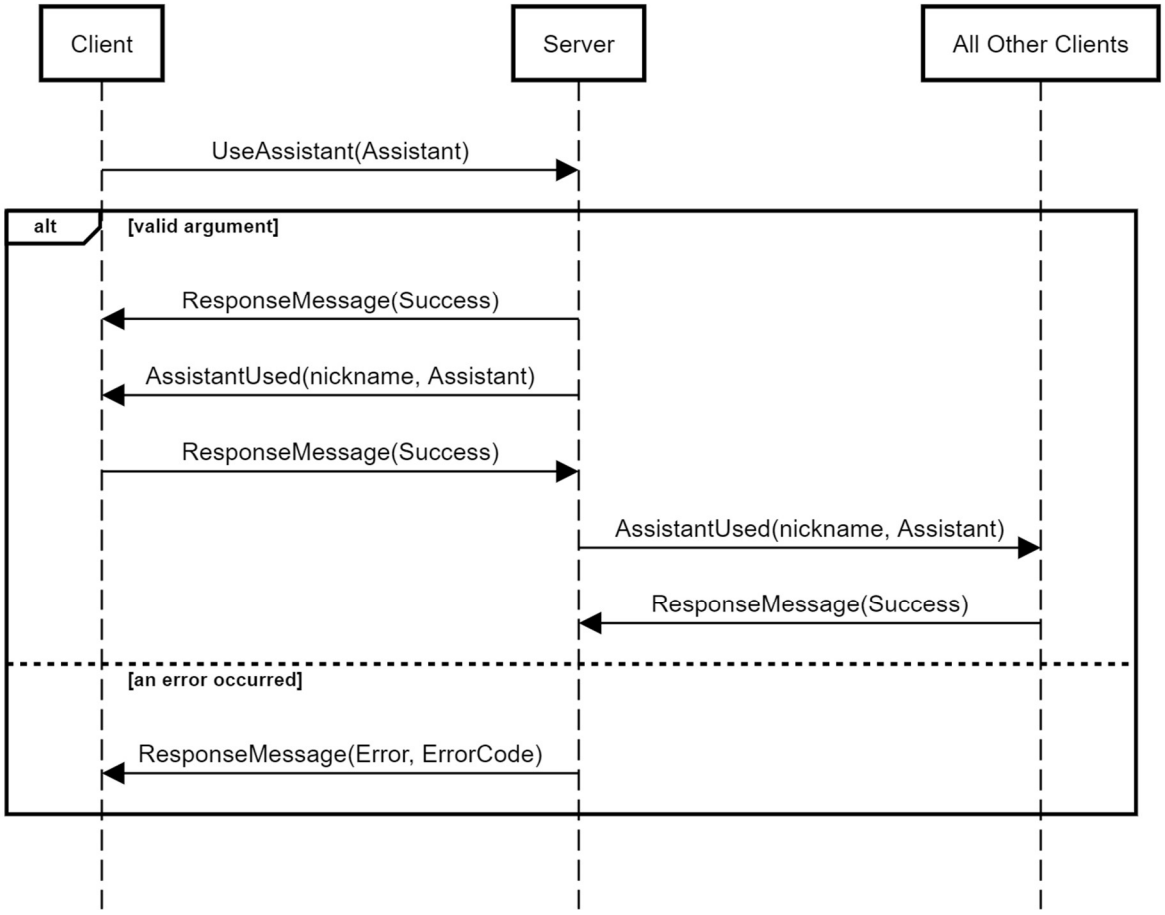


Changing state of matchmaking



A player can send a NextPhase message during the matchmaking. This message is used to move the game to the next phase, for example when all the players entered the game this is used to notify that to the server and move to the step where the players need to choose a tower and wizard. If this message is sent at the right time, the server changes the state of the game accordingly and notify all the players of the change with PlayerOrStateChanged, also sending the new player that need to take and action in this phase. Instead, if the message is sent when not all the conditions for changing state are met, the server will send an error message to indicate that.

Planning Phase



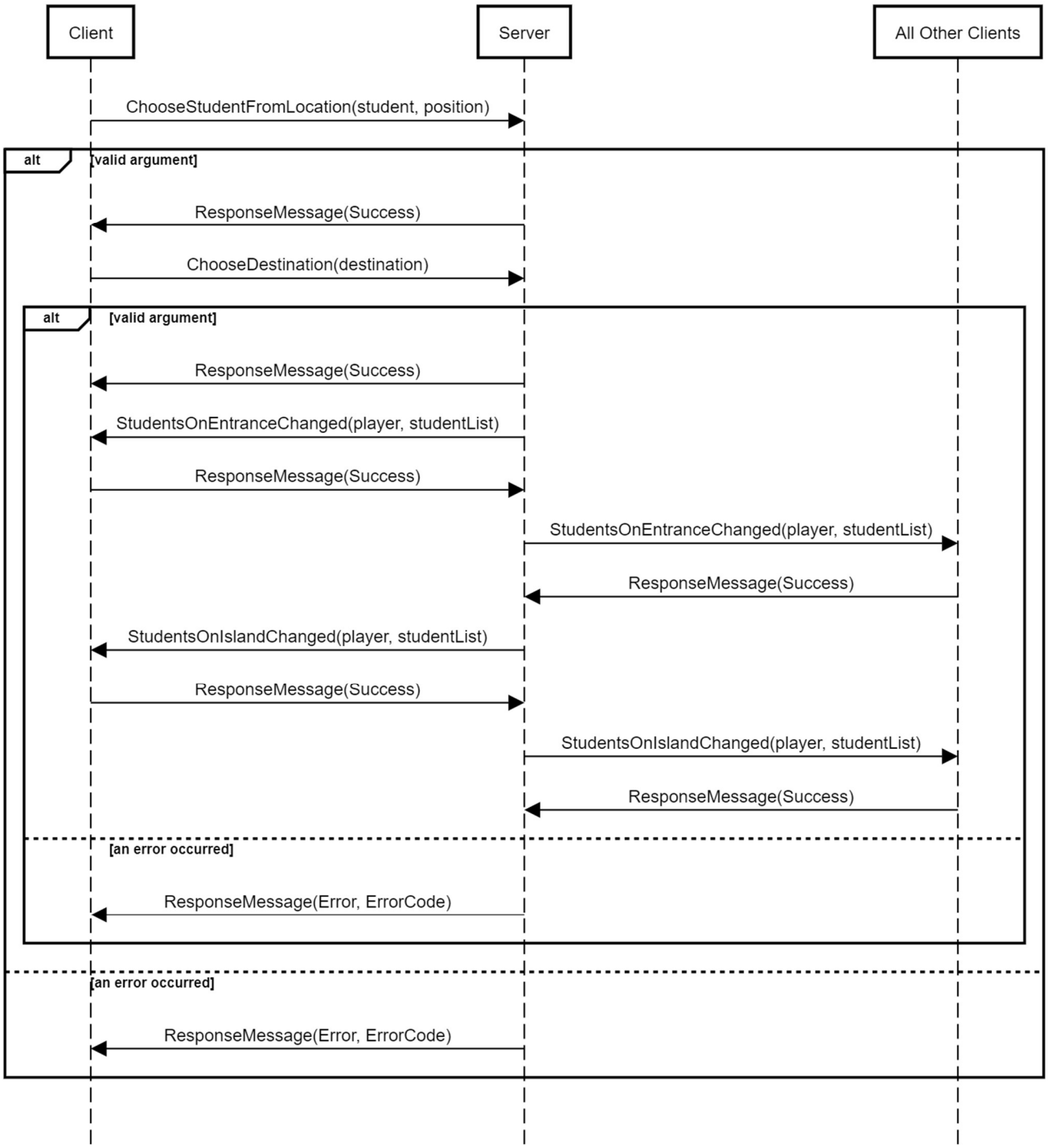
During the planning phase the client sends to the server a message to use an assistant.
The server will send an AssistantUsed message to all clients if everything is fine and then it will notify that the current player has changed with a CurrentPlayerChanged message, otherwise it will send back an error kind of message to the client that is different based on the specific error.

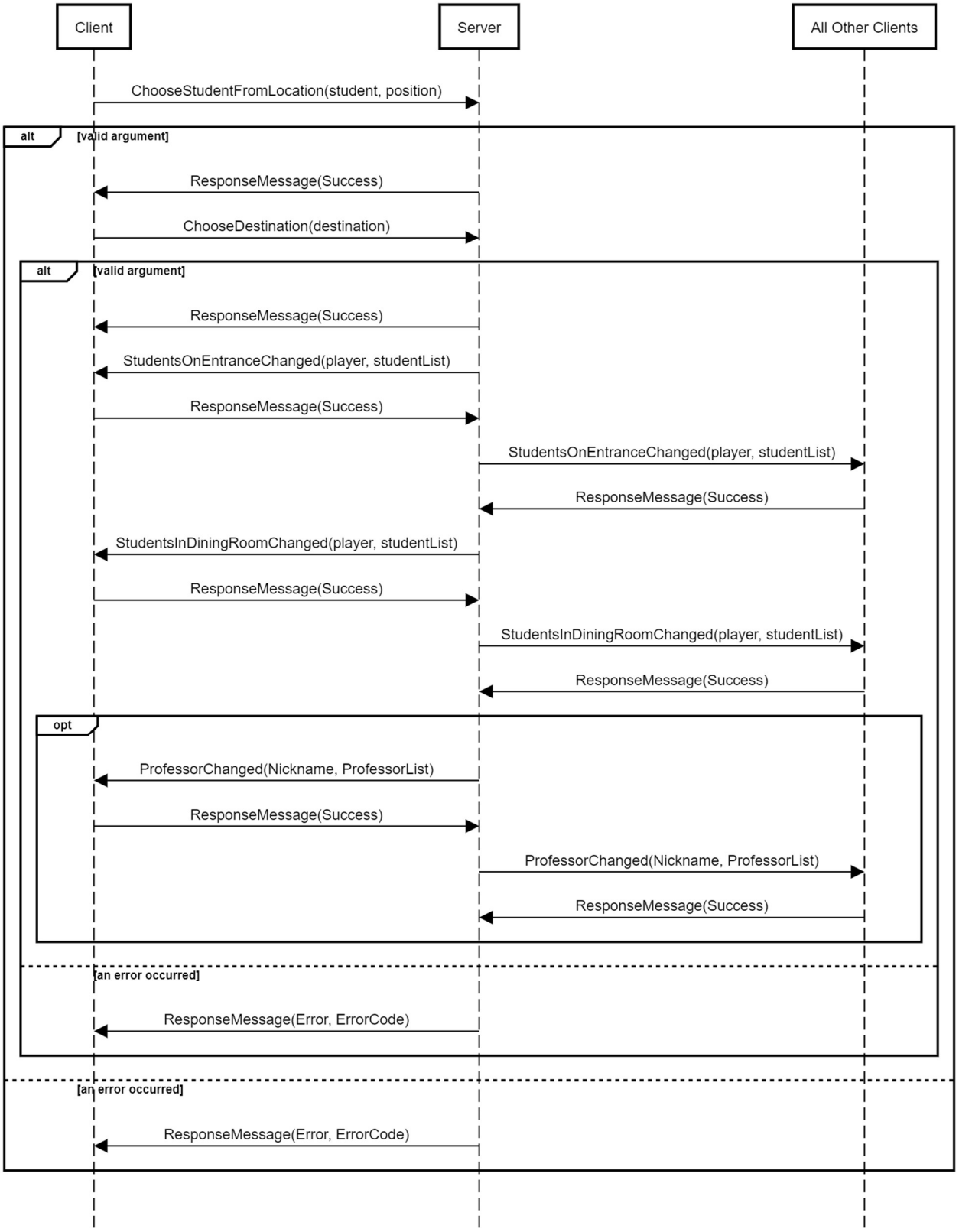
Action phase: move students

During the move student stage of the action phase, the client can send two types of messages to the server: a message to select one student from the entrance and a message to indicate where to put it.

If the client sends a ChooseDestination for an island, the server will send to all clients a StudentsOnIslandChanged message if everything is fine.
Instead, if client sends a ChooseDestination message for the dining room, if everything is fine, the server will send a StudentsInDiningRoomChanged and, if is the case, it will send also a ProfessorChanged message.

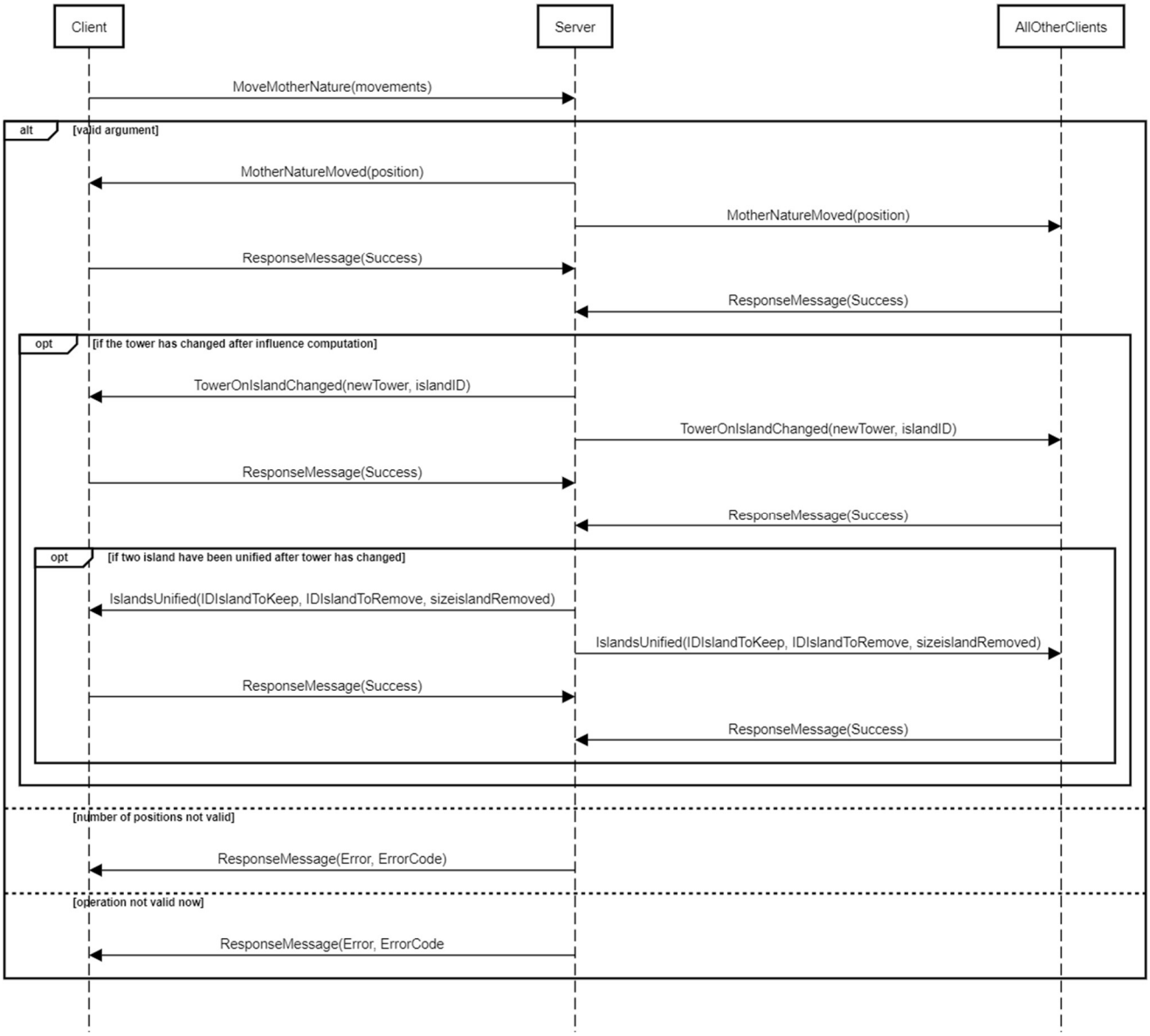
In both cases, in case of errors, the server will send back to the client an error kind of message that is different based on the specific error.


Client
Server
All Other Clients
ChooseStudentFromLocation(student, position)
alt
[valid argument]
ResponseMessage(Success)
ChooseDestination(destination)
alt
[valid argument]
ResponseMessage(Success)
StudentsOnEntranceChanged(player, studentList)
ResponseMessage(Success)
StudentsOnEntranceChanged(player, studentList)
ResponseMessage(Success)
StudentsOnIslandChanged(player, studentList)
ResponseMessage(Success)
StudentsOnIslandChanged(player, studentList)
ResponseMessage(Success)
[an error occurred]
ResponseMessage(Error, ErrorCode)
[an error occurred]
ResponseMessage(Error, ErrorCode)


Client
Server
All Other Clients
ChooseStudentFromLocation(student, position)
alt
[valid argument]
ResponseMessage(Success)
ChooseDestination(destination)
alt
[valid argument]
ResponseMessage(Success)
StudentsOnEntranceChanged(player, studentList)
ResponseMessage(Success)
StudentsOnEntranceChanged(player, studentList)
ResponseMessage(Success)
StudentsInDiningRoomChanged(player, studentList)
ResponseMessage(Success)
StudentsInDiningRoomChanged(player, studentList)
ResponseMessage(Success)
opt
ProfessorChanged(Nickname, ProfessorList)
ResponseMessage(Success)
ProfessorChanged(Nickname, ProfessorList)
ResponseMessage(Success)
[an error occurred]
ResponseMessage(Error, ErrorCode)
[an error occurred]
ResponseMessage(Error, ErrorCode)

Action phase: move mother nature

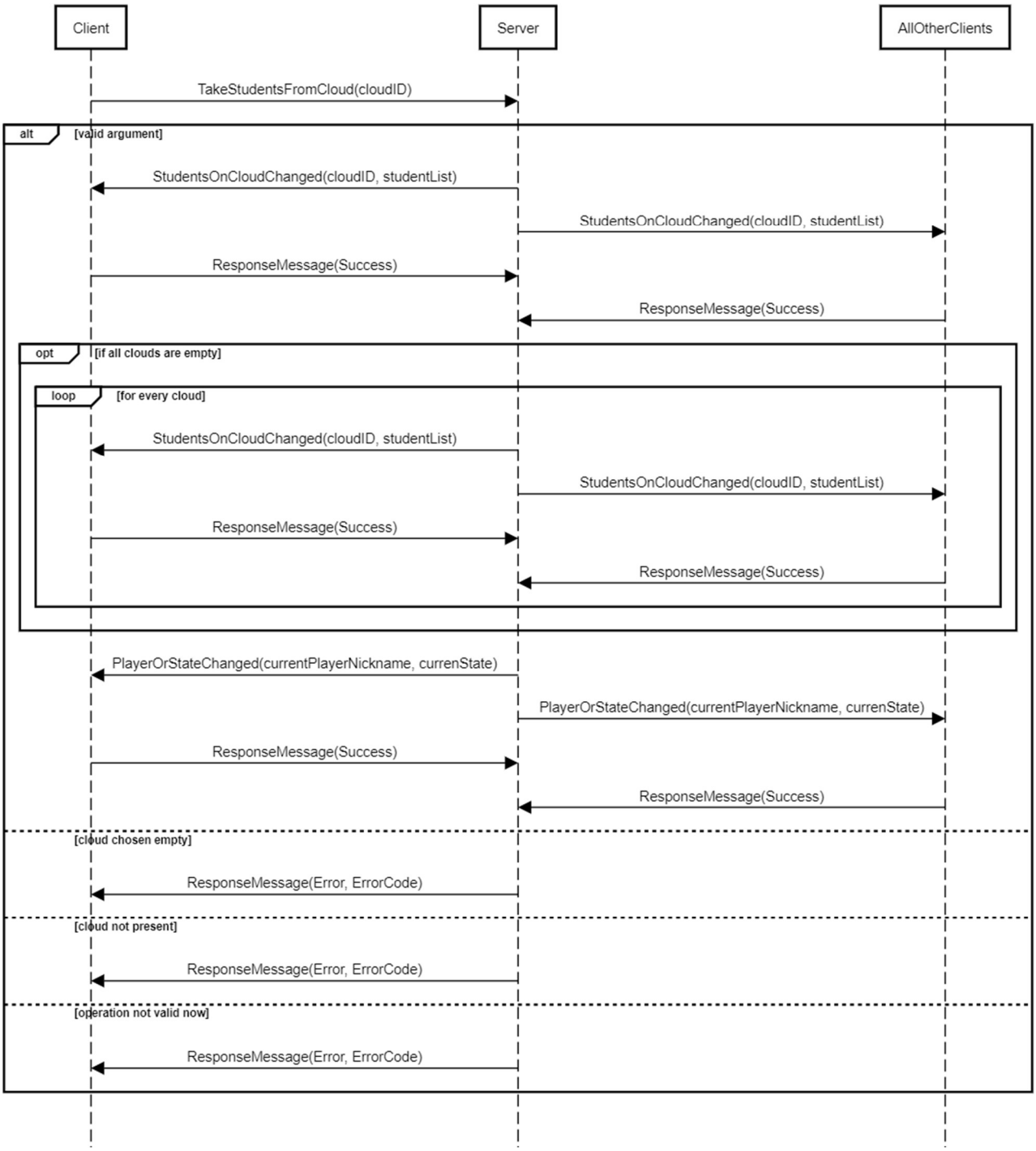
In this scenario the current player sends a MoveMotherNature message to the server to move mother nature. The server sends ResponseMessage(Error, ErrorCode) message if the number of positions is greater than the movements allowed. If the argument is valid, it updates the model, sends a response to all the clients and waits for the responses. After that it calculates the influence on the island where mother nature is positioned, if the tower on it has changed, the server sends TowerOnIslandChanged message to all clients and waits for the acknowledgements. Moreover, if two islands have been unified, it sends IslandsUnified message to all clients with the islands IDs and the size of the island removed and waits for the responses. If the current player tries to do other operations in this phase it sends ResponseMessage(Error, ErrorCode) message.



Action phase: choose a cloud tile

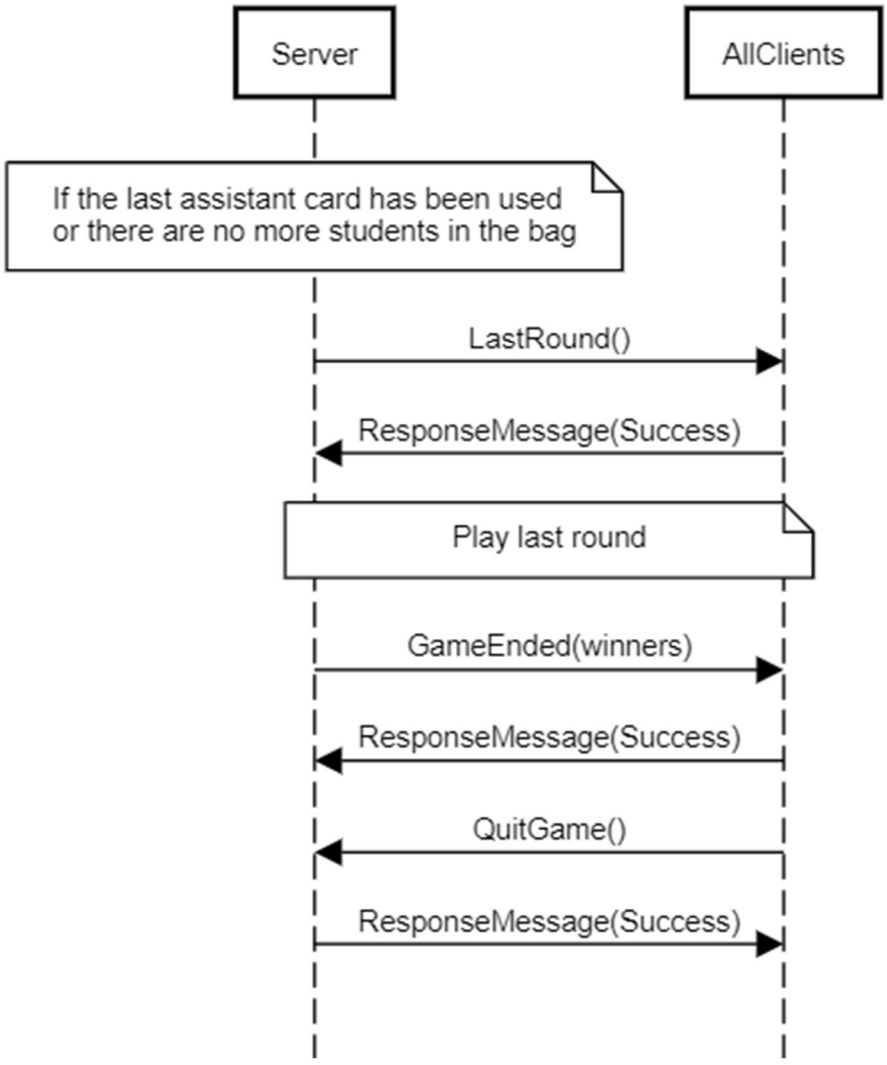
In this phase the current player sends a message to the server to take all the students from the cloud with the ID given as a parameter, the server updates the model and notifies all clients with the StudentsOnCloudChanged message and waits for the responses. If all clouds are empty, therefore if every player has played his turn, the server fills every cloud with students, notifies all

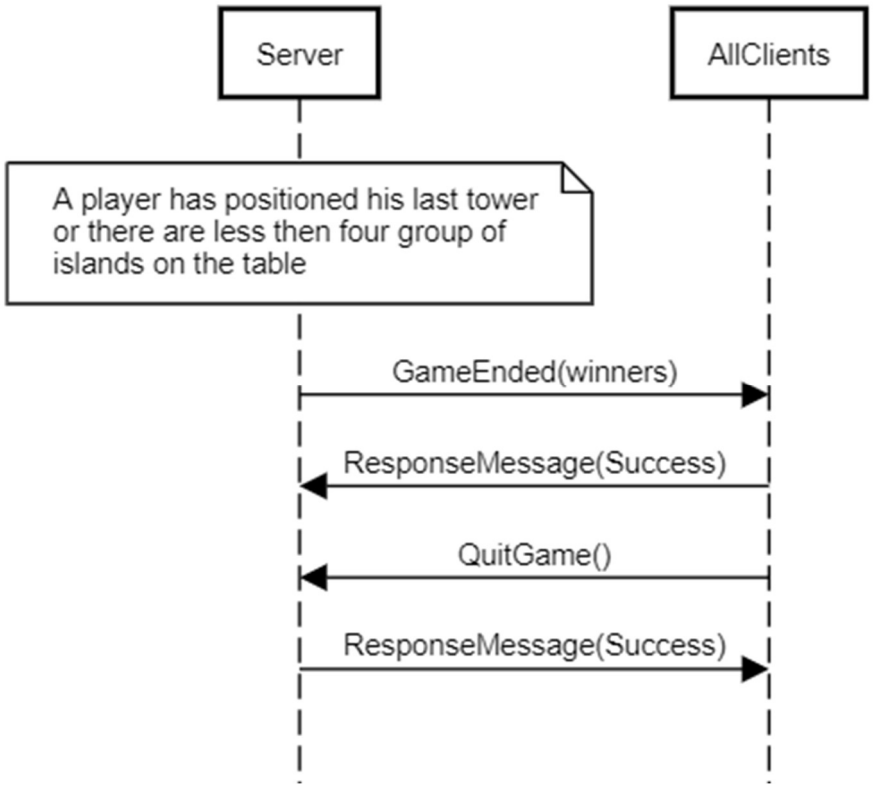
clients with the StudentsOnCloudChanged message with the new students and waits for the responses for each cloud. Finally, it notifies all clients that the current player and the current state havechanged with PlayerOrStateChanged message and waits for the responses. If the client sends the ID of an already empty cloud, the server sends to this client the ResponseMessage(Error, ErrorCode) message and if the clients send the ID of a cloud that doesn't exist it sends the ResponseMessage(Error, ErrorCode) message. Finally, if the current player tries to do a move not allowed in this phase the server sends the ResponseMessage(Error, ErrorCode) error message.



End of the game

When the last assistant card has been used or there are not enough students in the bag to fill all the clouds, the server notifies all clients with the LastRound message and waits for acknowledgements. Then the last round is played and at the end the server sends GameEnded message to all clients with the nicknames of the winners and waits for responses. At this point the clients can exit by sending a message to the server.
If a player has positioned his last tower or there are less than four group of islands on the table, the server sends directly the GameEnded message without continuing the game, then it waits for responses. That sends the QuitGame message to exit the game and waits for response.




Server
AllClients
A player has positioned his last tower or there are less then four group of islands on the table
GameEnded(winners)
ResponseMessage(Success)
QuitGame()
ResponseMessage(Success)