Eriantys Protocol Documentation

Giorgio Miani, Alessia Porta, Andrea Verrone

Gruppo 31

Messages

Acknowledgment

This message is sent from the client to the server to acknowledge a change notification message previously sent from the server

Arguments

This message has no arguments.

Possible responses

This message has no responses.

ArgumentNotCorrectError

This message is sent from the server to the client when the argument of a request is not correct.

Arguments

This message has no arguments

Possible responses

This message has no responses

ImpossibleActionError

This message is sent from the server to the client to indicate that the action requested can't be executed in the current state of the game.

Arguments

This message has no arguments.

Possible responses

This message has no responses.

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

Possible responses

- GameCreated: the message has been received and the game has been created
- ArgumentNotCorrectError: if the specified number of players is unsupported

GameCreated

This message is sent from the server to the client when a new game has been created and a lobby is ready to accept players.

Arguments

- gameID: the identifier of the game created

Possible responses

- Acknowledgment: the message has been received

JoinGame

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

Possible responses

- PossibleGames: the message has been received

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

Possible responses

This message has no response

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

Possible responses

- GameEntered: if the user entered the game
- GameDontExistError: if the gameID don’t correspond to any available game
- GameFullError: if the game exists, but has reached the number of players requested
- NicknameAlreadyTakenError: if the nickname is already used by another user

GameEntered

This message is sent from the server to the client when a user entered a game.

Arguments

This message has no arguments

Possible responses

This message has no responses

GameDontExistError

This message is sent from the server to the client when a user tried to enter a not existing game.

Arguments

This message has no arguments

Possible responses

This message has no responses

GameFullError

This message is sent from the server to the client when a user tried to enter an existing game, but that game already has reached the maximum number of players requested.

Arguments

This message has no arguments

Possible responses

This message has no responses

NicknameAlreadyTakenError

This message is sent from the server to the client when a user tried to enter an existing game, but the nickname chosen is already in use of another user.

Arguments

This message has no arguments

Possible responses

This message has no responses

ExitFromGame

This message is sent from the client to the server when a user wants to exit from a starting game.

Arguments

- nickname: the nickname of the user

Possible responses

- ExitGameSuccess: if the user successfully exited the game
- ArgumentNotCorrectError: if there is no player in the game with that user
- ImpossibleActionErrorError: if the game is already started and the player can't leave

ExitGameSuccess

This message is sent from the server to the client when a user exited the game.

Arguments

This message has no arguments

Possible responses

This message has no responses

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

Possible responses

- Acknowledgment: the message has been received

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

- newNum: the new number of players

Possible responses

- NumPlayersChanged: if the number was successfully changed
- ArgumentNotCorrectError: if the number specified is not correct
- ImpossibleActionErrorError: if the game has started and the number of players can't be changed anymore

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

Possible responses

- Acknowledgment: the message has been received

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

Possible responses

- TowerChanged: if the tower successfully changed
- ArgumentNotCorrectError: if the tower chosen can't be assigned
- ImpossibleActionError: if the game has not started yet and the user can't choose a tower

TowerChanged

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

Arguments

- Tower: the new tower of the player

Possible responses

- Acknowledgment: the message has been received

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

Possible responses

- WizardChanged: if the wizard successfully changed
- ArgumentNotCorrectError: if the wizard chosen can't be assigned
- ImpossibleActionError: if the game has not started yet and the user can't choose a wizard

WizardChanged

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

Arguments

- wizard: the new wizard of the player

Possible responses

- Acknowledgment: the message has been received

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

Possible responses

- CurrentPlayerChanged: if the game changed state correctly
- ImpossibleActionError: if not all the conditions for changing state were met

GameStarted

This message is sent from the server to the client when all the players has finished the setup and the game has started.

Arguments

- table: the table of the game
- players: the players in this game

Possible responses

- Acknowledgment: the message has been received

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

Possible Responses

- AssistantUsed: the message has been received and the model has been updated
- AssistantNotExistsError: the message has been received, but the assistant passed as a parameter is not present in the deck of the current player
- AssistantCanNotBeUsedError: the message has been received, but the assistant passed as a parameter cannot be used
- ImpossibleActionError: if the client tries to use an assistant when it is not allowed to do that

AssistantUsed

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

Arguments

- Assistant: the assistant card played by the current player

Possible Responses

- Acknowledgement: if the message has been received

AssistantNotExistsError

This message is sent from the server to the client that has tried to use an assistant card that was not in his deck.

Arguments

This message has no arguments.

Possible Responses

This message has no responses.

AssistantCanNotBeUsedError

This message is sent from the server to the client that has tried to use an assistant card, but the assistant passed as a parameter cannot be used since a player cannot use an assistant card that has already been used by another player in the same turn (unless he has only already used cards)

Arguments

This message has no arguments.

Possible Responses

This message has no responses.

MoveStudentsToIsland

This message is sent from the client to the server after the current player moves a student from the entrance of his school board to an island.

Arguments

- Student: the student to be moved
- IslandID: the island on which put the student

Possible Responses

- StudentToIslandMoved: the message has been received and the model has been updated
- StudentInEntraceNotExistsError: the message has been received, but the student passed as a parameter is not present in the entrance of his school board
- IslandNotExistsError: the message has been received, but the island passed as a parameter is not present in the entrance of his school board
- ImpossibleActionError: if the client tries to move a student when it is not allowed to do that

StudentToIslandMoved

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 an island.

Arguments

- Student: the student that has been moved
- IslandID: the island on which the student has been placed

Possible Responses

- Acknowledgement: if the message has been received

ErrorStudentInEntraceNotExists

This message is sent from the server to the client that has tried to move a student from the entrance of his school board, but it is not present a student of that PawnType.

Arguments

This message has no arguments.

Possible Responses

This message has no responses.

IslandNotexistsError

This message is sent from the server to the client that has tried put a student on an island that does not exists.

Arguments

This message has no arguments.

Possible Responses

This message has no responses.

MoveStudentsToDiningRoom

This message is sent from the client to the server after the current player moves a student from the entrance of his school board to its dining room.

Arguments

- Student: the color of the student to move

Possible Responses

- StudentToDiningRoomMoved: the message has been received and the model has been updated
- StudentInEntraceNotExistsError: the message has been received, but the student passed as a parameter is not present in the entrance of his school board
- TableInDiningRoomFullError: the message has been received, but the table of the student passed as a parameter is full, therefore it can not be added another student of that color
- ImpossibleActionError: if the client tries to move a student when it is not allowed to do that

StudentToDiningRoomMoved

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.

Arguments

- Student: the student that has been moved

Possible Responses

This message has no responses.

ChangeProfessor

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
- ProfessorList: the new professor list of the player

Possible Responses

- Acknowledgement: if the message has been received

TableInDiningRoomFullError

This message is sent from the server to the client that has tried to move a student to its dining room, but the table of the student of that color is full.

Arguments

This message has no arguments.

Possible Responses

This message has no responses.

MoveMotherNature

This message is sent from a client to the server after the current player has moved mother nature.

Arguments

- Positions: number of islands mother nature moved on

Possible Responses

- MotherNatureMoved: the message has been received and the mother nature has been moved
- ArgumentNotCorrectError: the message has been received but the client has entered a number of positions over the number of movements allowed
- ImpossibleActionError: if current client tries to do another operation in this phase of the game

MotherNatureMoved

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

Arguments

- Positions: number of islands mother nature moved on

Possible Responses

- Acknowledgement: if the message has been received

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

Possible Responses

- Acknowledgement: if the message has been received

IslandsUnified

This message is sent from the server to all clients after two islands have been unified because of a tower that has changed on one of them.

Arguments

- IslandID1: ID of the island unified where the tower has been changed
- IslandID2: ID of the other island unified

Possible Responses

- Acknowledgement: if the message has been received

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

Possible Responses

- StudentsTakenFromCloud: the message has been received and the students on the chosen cloud have been moved on the current player's entrance
- CloudEmptyError: the cloud chosen is already empty
- CloudNotPresentError: the cloud chosen doesn't exist
- ImpossibleActionError: if the current client tries to do another operation in this phase of the game

StudentsTakenFromCloud

This message is sent from the server to all clients after all the students on a cloud have been moved to the current player's entrance.

Arguments

- CloudID: ID of the cloud from where the students have been taken

Possible Responses

- Acknowledgement: if the message has been received

CloudFilled

This message is sent from the server to all clients after an empty cloud has been filled again.

Arguments

- CloudID: Id of the cloud filled
- Students: StudentList of the students moved on the cloud

Possible Responses

- Acknowledgement: if the message has been received

CloudEmptyError

This message is sent from the server to the client that has tried to take from an empty cloud

Arguments

This message has no arguments.

Possible Responses

This message has no responses

CloudNotPresentError

This message is sent from the server to the client that has tried to take from a cloud not present in the game table

Arguments

This message has no arguments.

Possible Responses

This message has no responses

Winner

This message is sent from the server to all clients to notify the winner at the end of the game

Arguments

Nickname: nickname of the winner

Possible Responses

- Acknowledgement: if the message has been received

Exit

This message is sent from a client to the server when the player logs out at the end of the game

Arguments

This message has no arguments.

Possible Responses

This message has no responses

CurrentPlayerChanged

This message is sent from the server to all clients if the current player has changed

Arguments

- NextPlayer: new current player

Possible Responses

- Acknowledgement: if the message has been received

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.

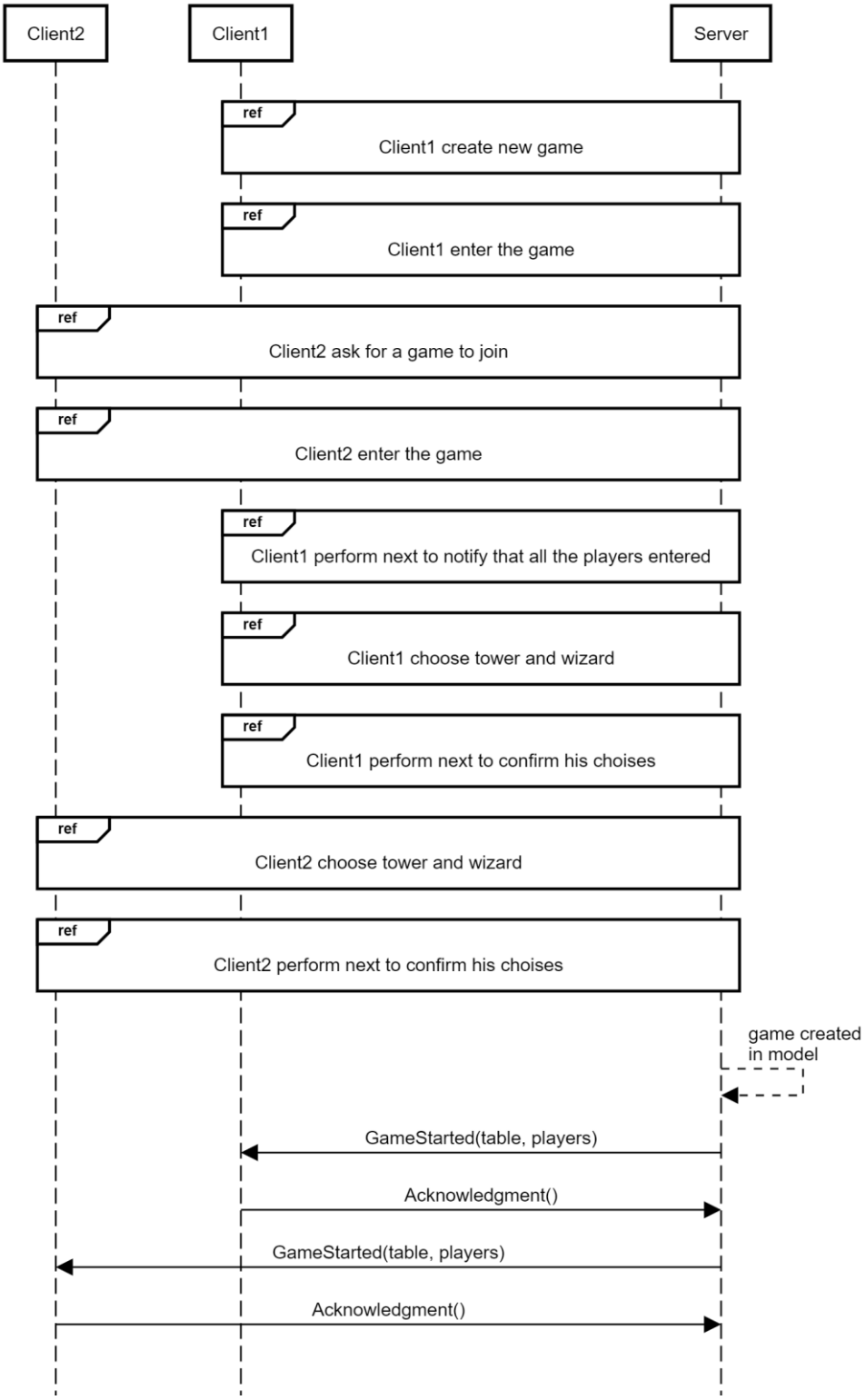
Possible Responses

- Acknowledgement: if the message has been received

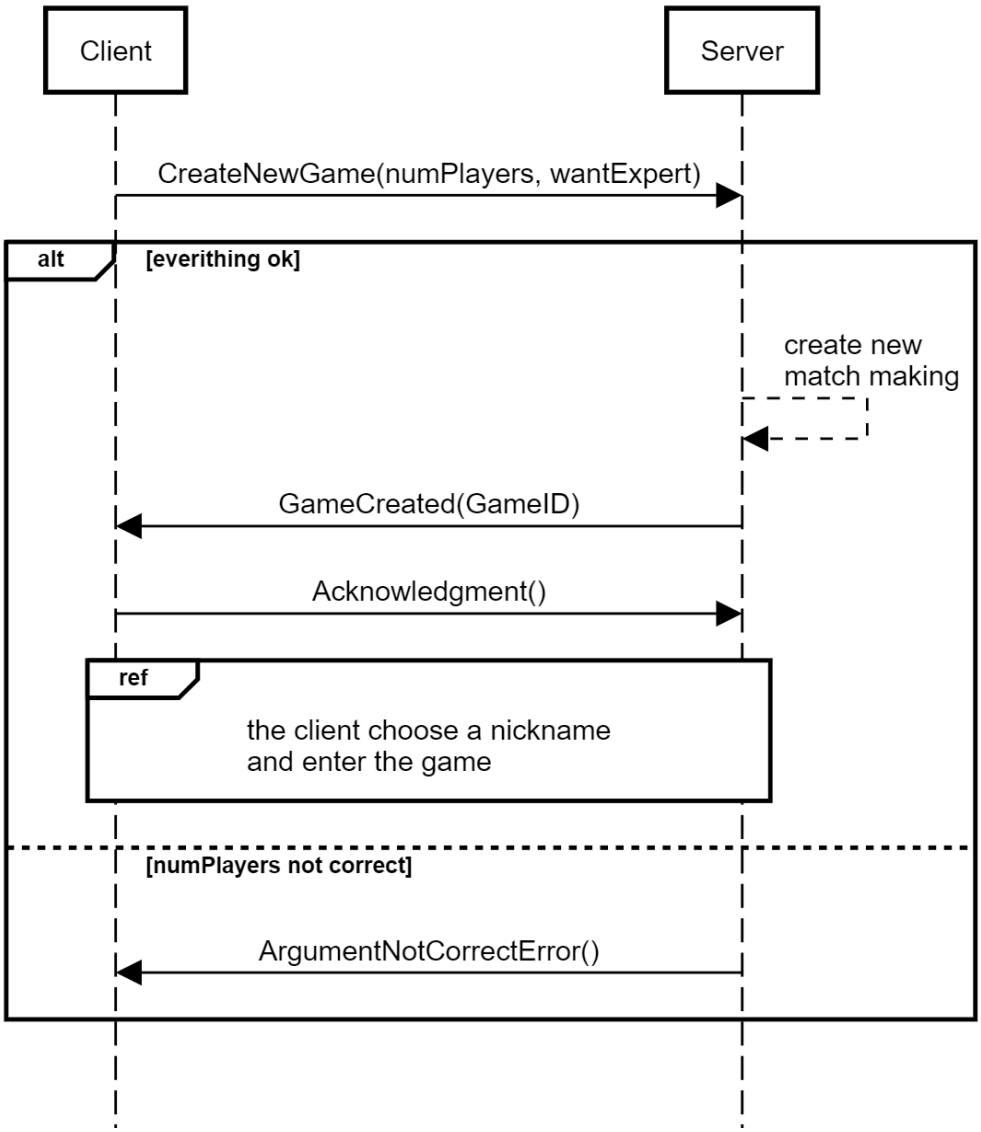
Scenarios

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 GameStarted message to the clients sending them the table and the list of players of this game.

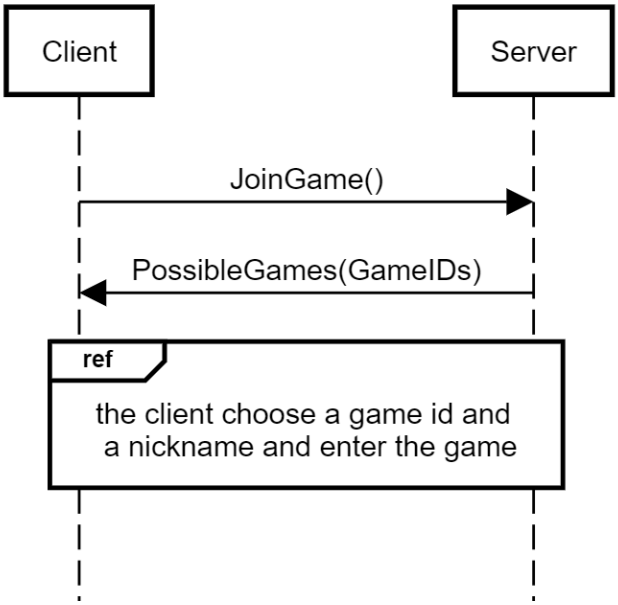


Creating a new game



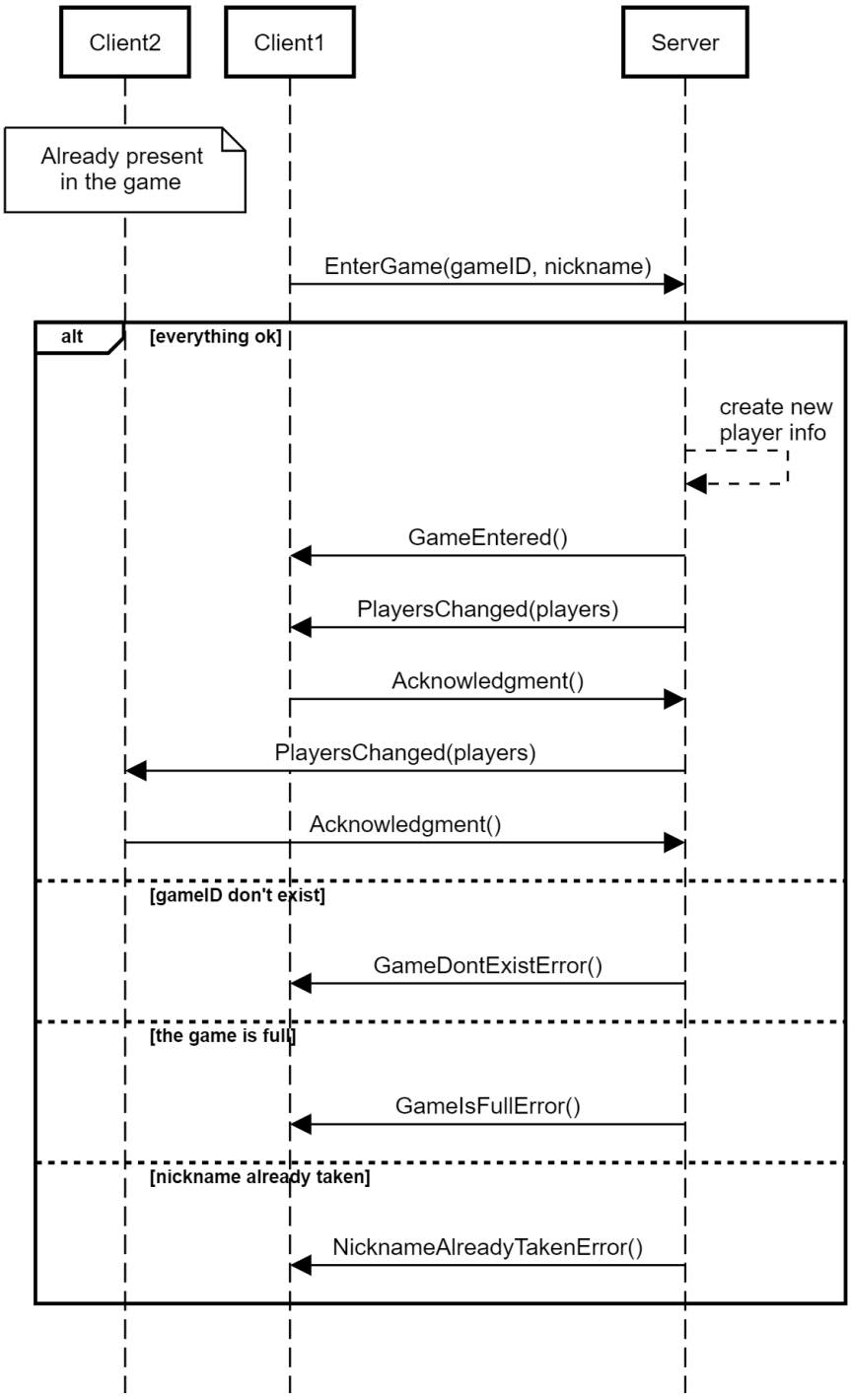
When a client opens the game, he can choose to create a new game and the CreateNewGame message will be sent with the number of requested players chosen by the user and if the game should use the expert mode. The number of players requested should be one of the supported value or the server will respond with an ArgumentNotCorrectError. If the number is correct instead it responds with GameCreated 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



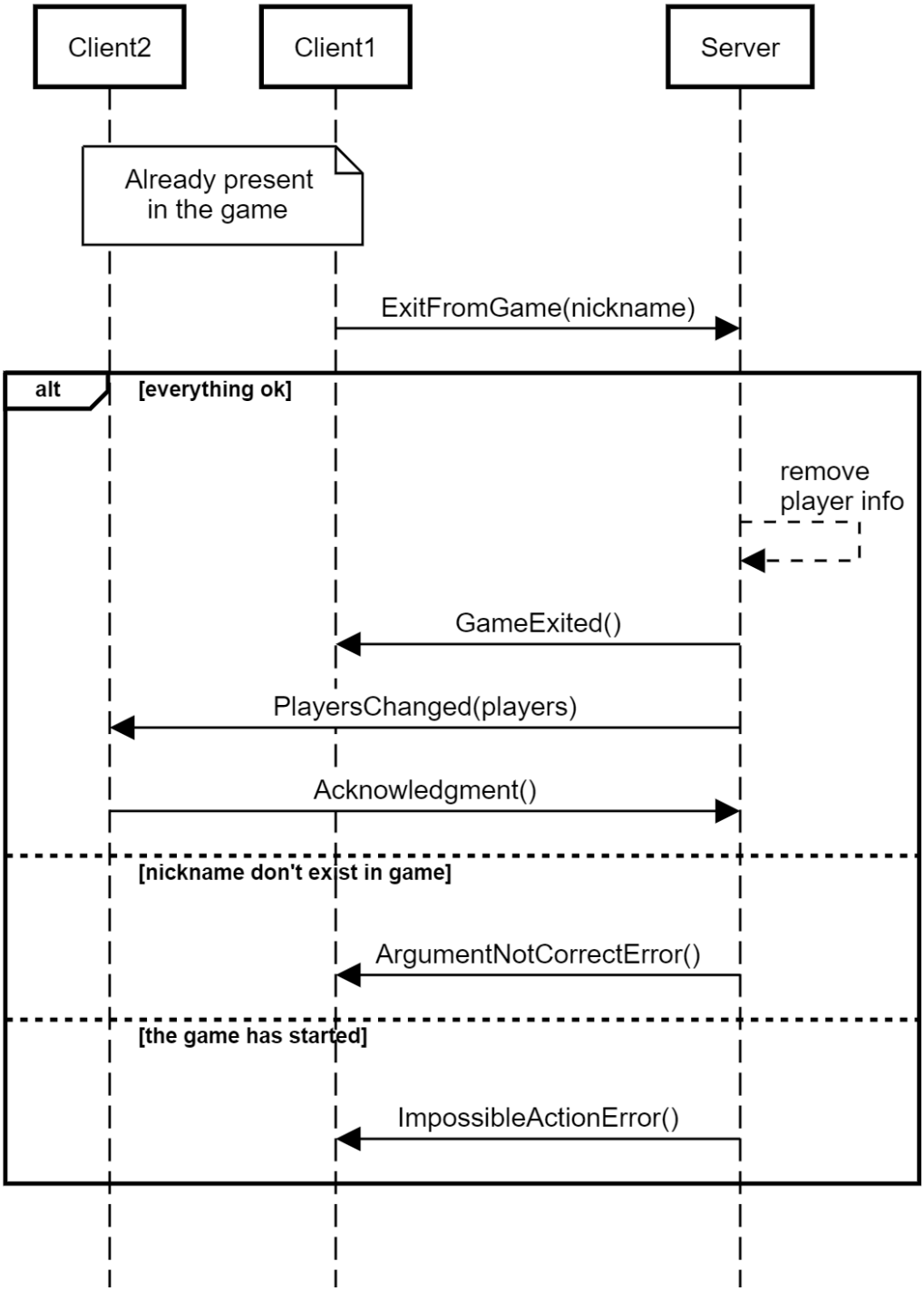
When a client opens the game, he can choose to join an existing game and the JoinGame message will be sent to the server. It will respond with PossibleGames giving the client the list of IDs of games that are in the initial phase of the matchmaking, where new players are allowed to join the game. After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



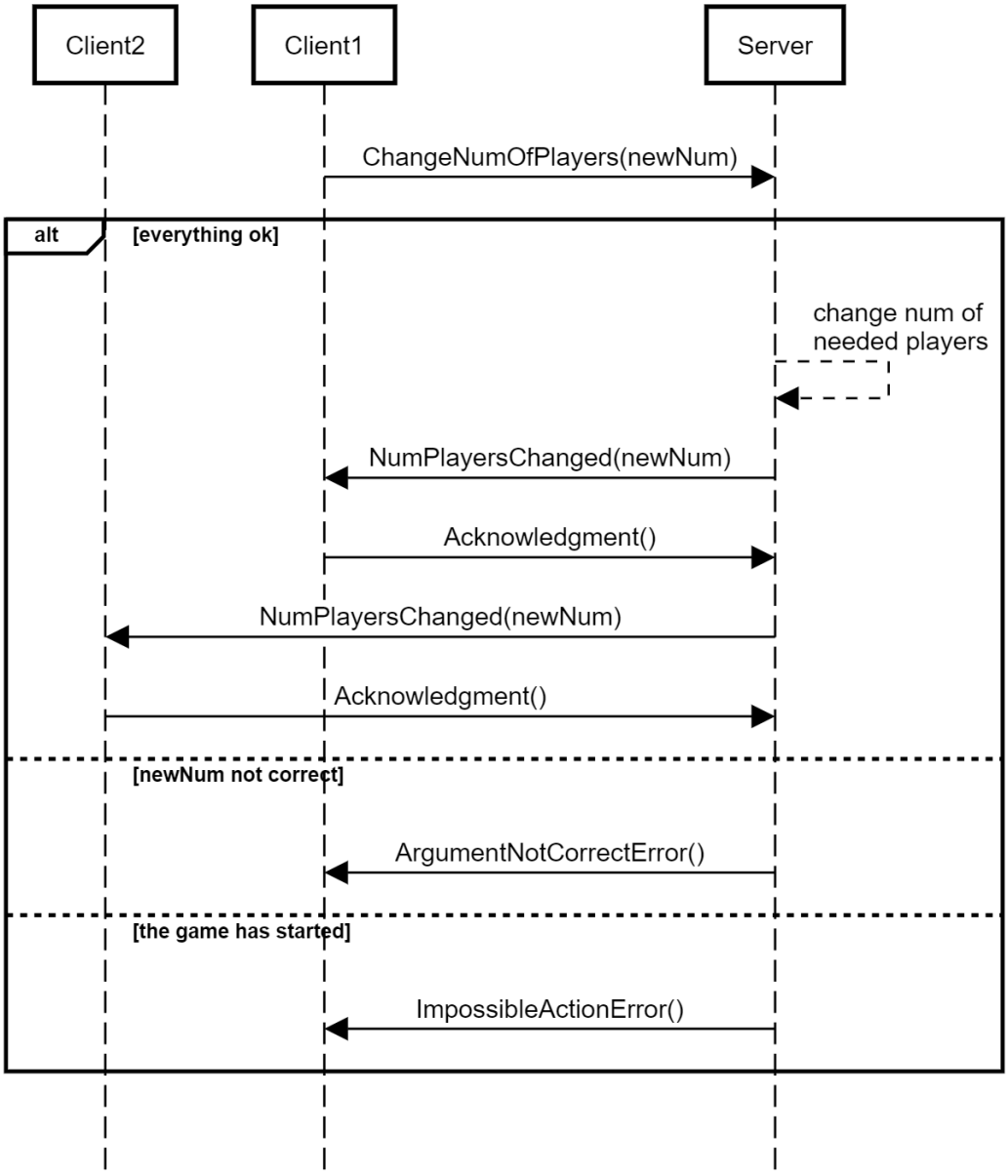
When a game is created, the server waits for clients to send EnterGame messages. Each time the server receives this message, it could respond with GameDontExistError if the gameID that he received doesn't correspond to any available game. If it finds a game corresponding the ID it received, it can send a GameFullError message if the game has reached the maximum number of players allowed and can't receive anymore, NicknameAlreadyTakenError if the game can accept new players but there is already a player with the same nickname, or GameEntered message if the player has successfully entered the game. 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 Acknowledgment to the server.

Exiting from a game



A client connected to a game can send a ExitFromGame message to communicate the server his intention to leave the game. When the server receives this message, it will check if players are allowed to leave the game in the current state, and if it's not the case it will send back an ImpossibleActionError. If the player is allowed to leave and there is a player in the game with the provided nickname, it will remove the player from the game, sending a GameExited message to the client that made the request and a PlayersChanged 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. If the previous condition were not met, the server will respond with ArgumentNotCorrectError, indicating that something went wrong and there is no player in the game with that nickname.

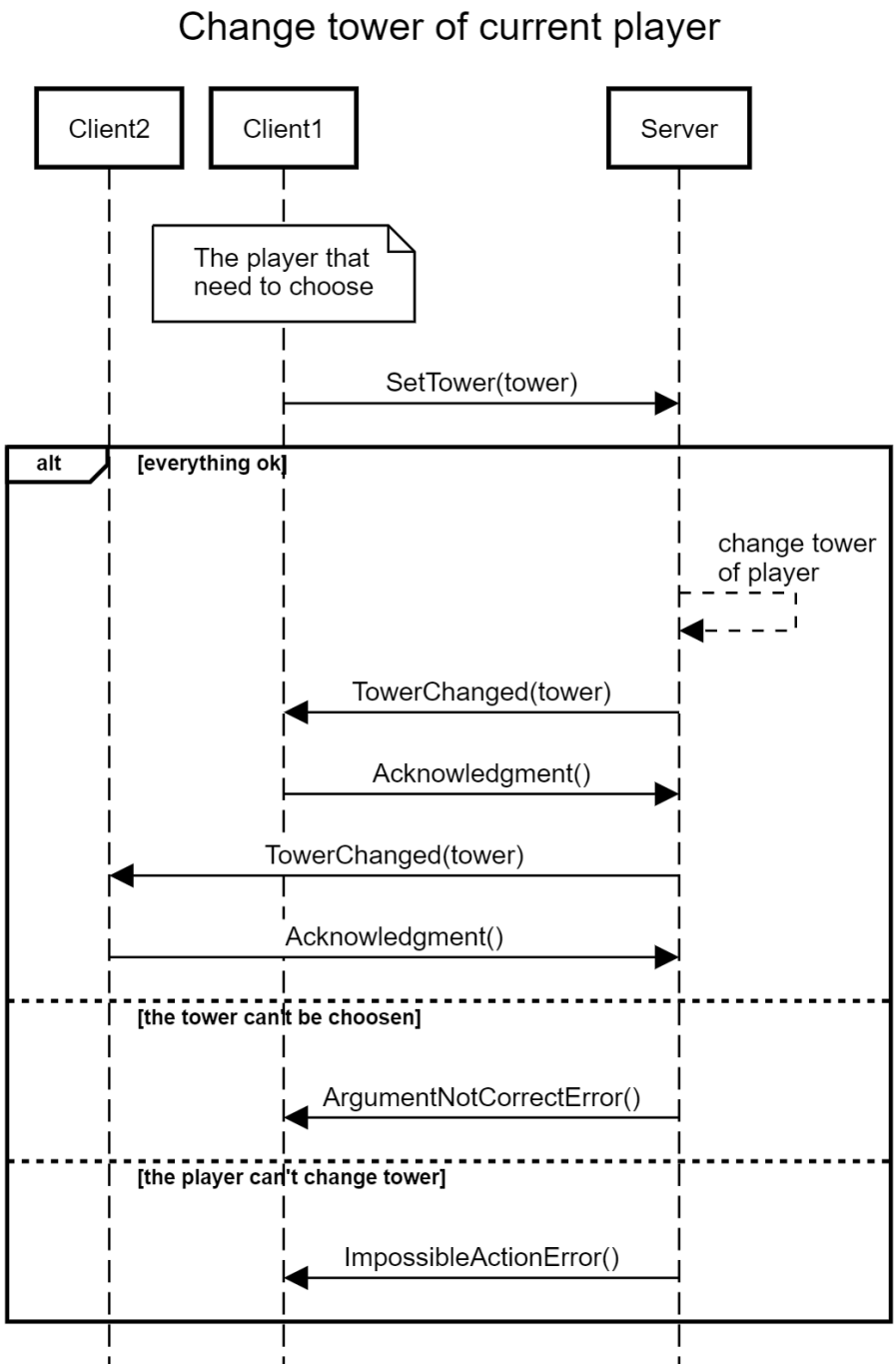
Changing the number of players



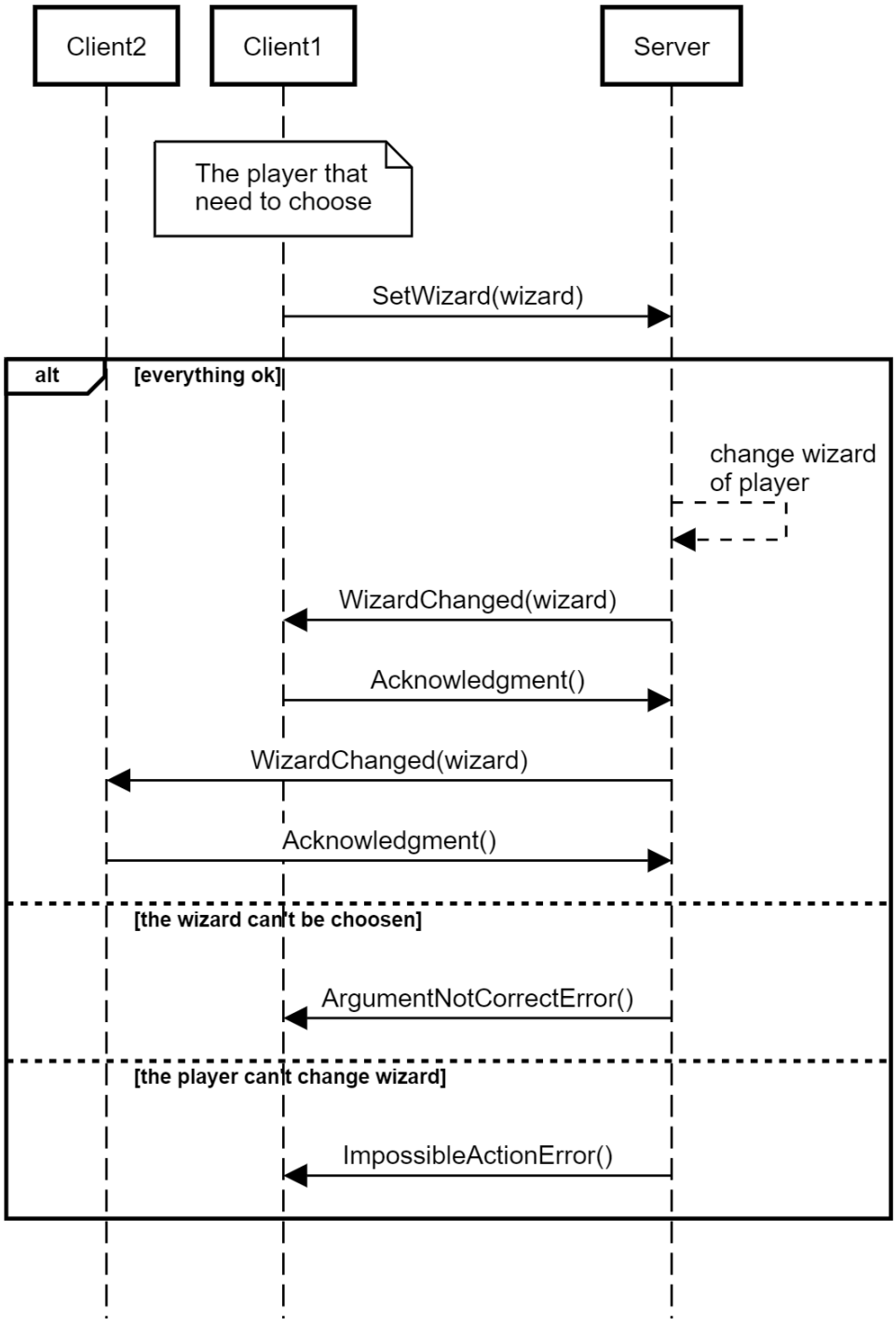
When in the first phase of the matchmaking, a player can send a ChangeNumPlayers message to the server to indicate his willingness to change the number of players needed in this game. If the number requested is not valid, for example if it's not one of the possible supported value or if it's less than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

Changing the attributes of a player

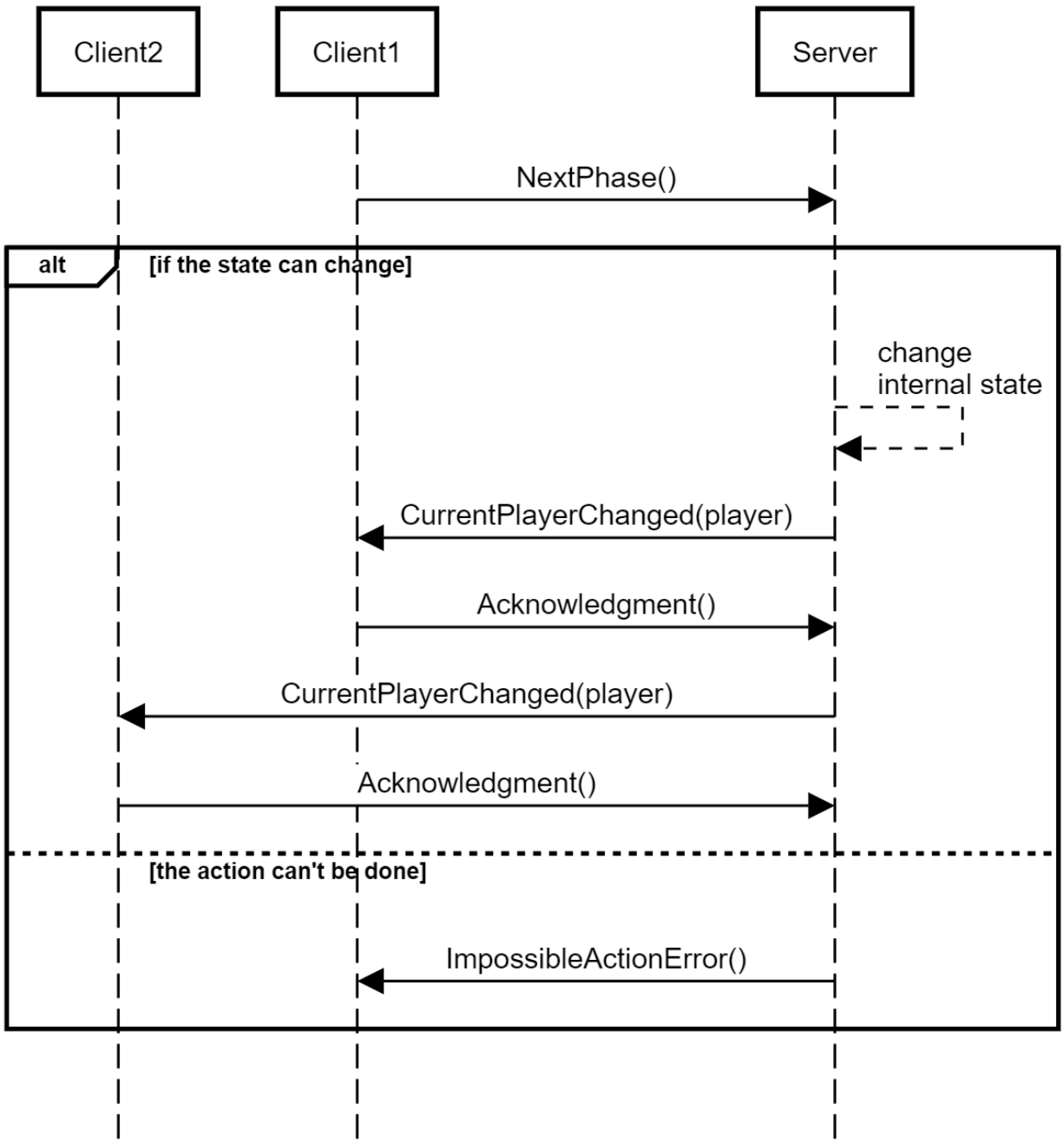
After all the needed players has joined a game, one by one they will be asked to choose a color for the tower and a wizard to use during the game. These two actions can be done using the SetTower and SetWizard messages, in any order and multiple times in the same turn. If any of these messages arrives to the server in an invalid state, for example if the game is in the first phase where new players are expected to join, it will respond with an ImpossibleActionError message. If it's not the case, the server can respond with ArgumentNotCorrectError 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. If everything is fine, the server will change the tower/wizard of the current player and notify all the player of the change with TowerChanged/WizardChanged.



Change wizard of current player

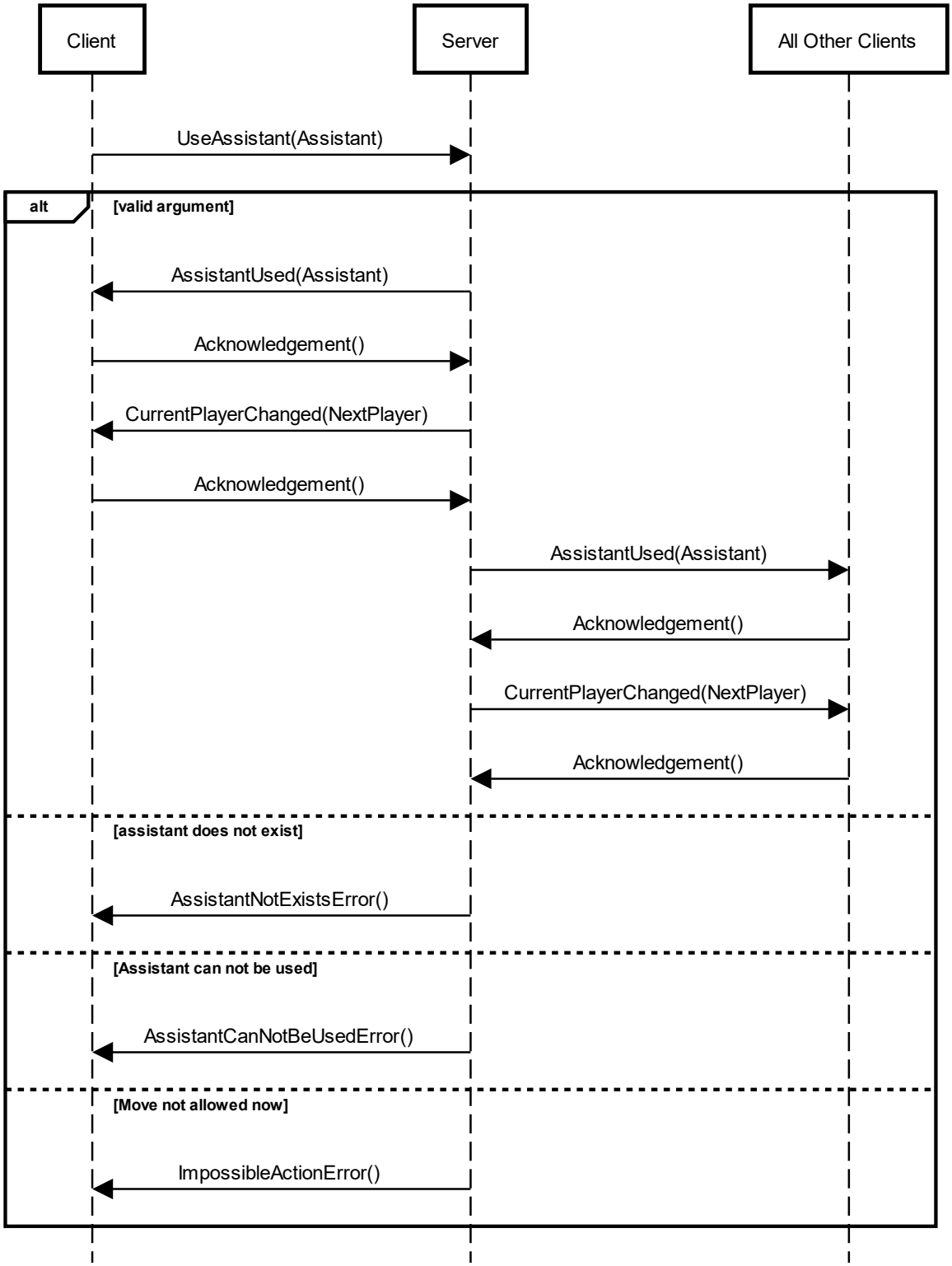


Changing state of matchmaking



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 CurrentPlayerChanged, 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 ImpossibleActionError to indicate that.

Planning Phase



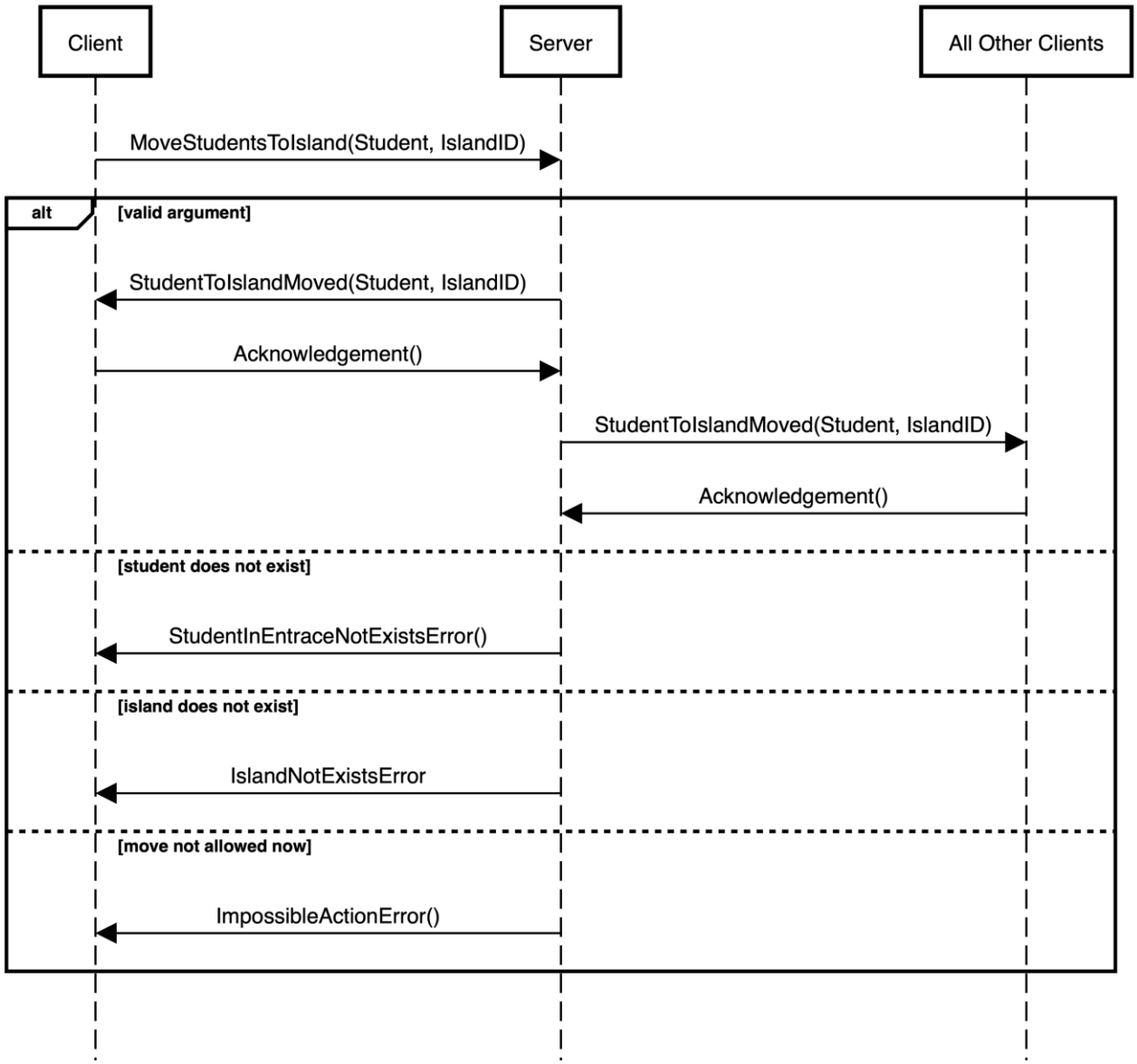
During the planning phase the client sends to the server a message to use an assistant.
The server will send an AssistantUsed message to all clients if everything is fine and then it will notify that the current player has changed with a CurrentPlayerChanged message, otherwise it will send back an error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

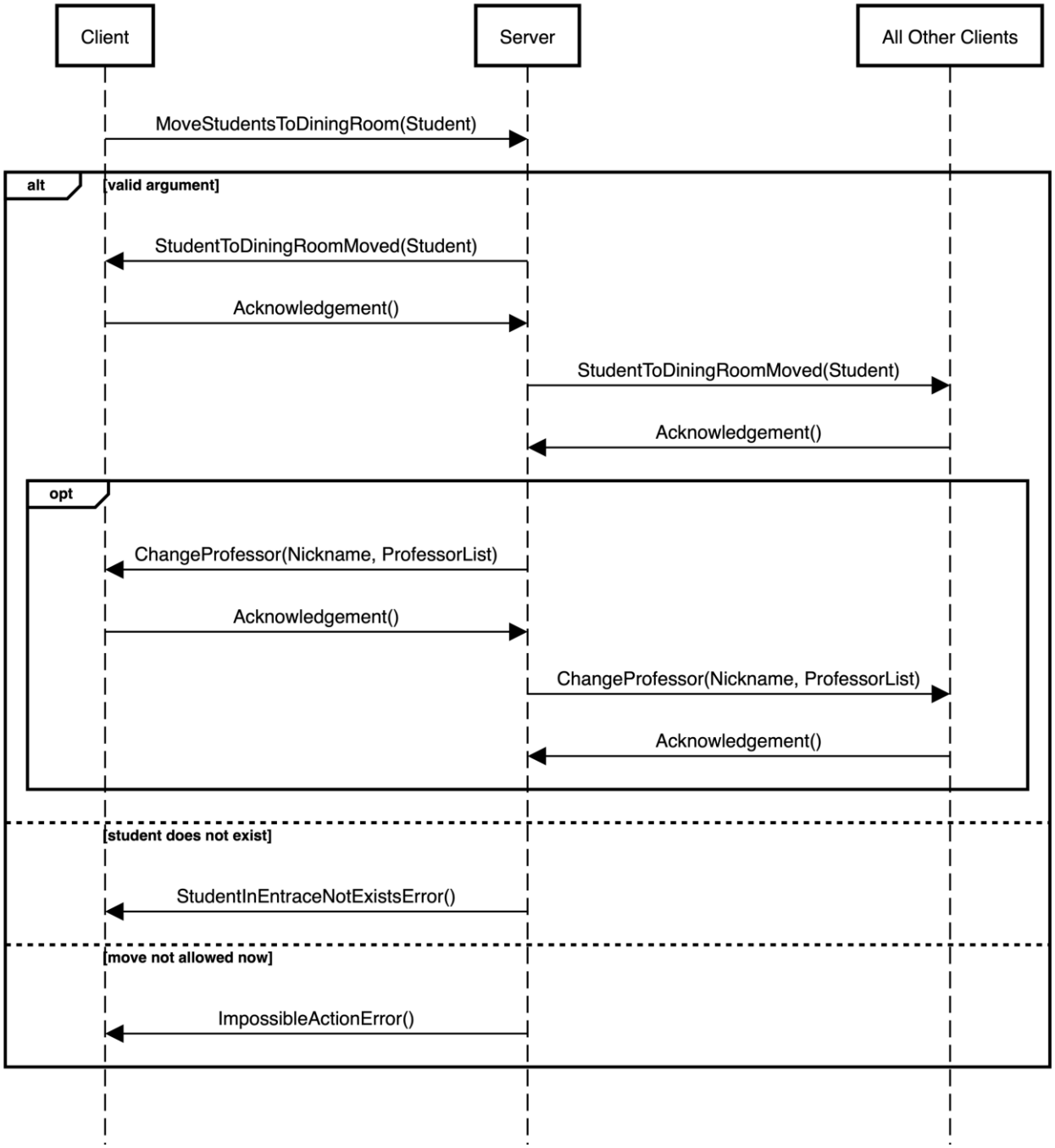
Action phase: move students

During the move student stage of the action phase, the client can send two types of messages to the server: a message to move one student from entrance to island and a message to move one student from entrance to dining room.

If the client sends a MoveStudentsToIsland, the server will send to all clients a StudentToIslandMoved message if everything is fine.
Instead, if client sends a MoveStudentsToDiningRoom message, if everything is fine, the server will send a StudentToDiningRoomMoved message and if is the case it will send also a ChangeProfessor message.

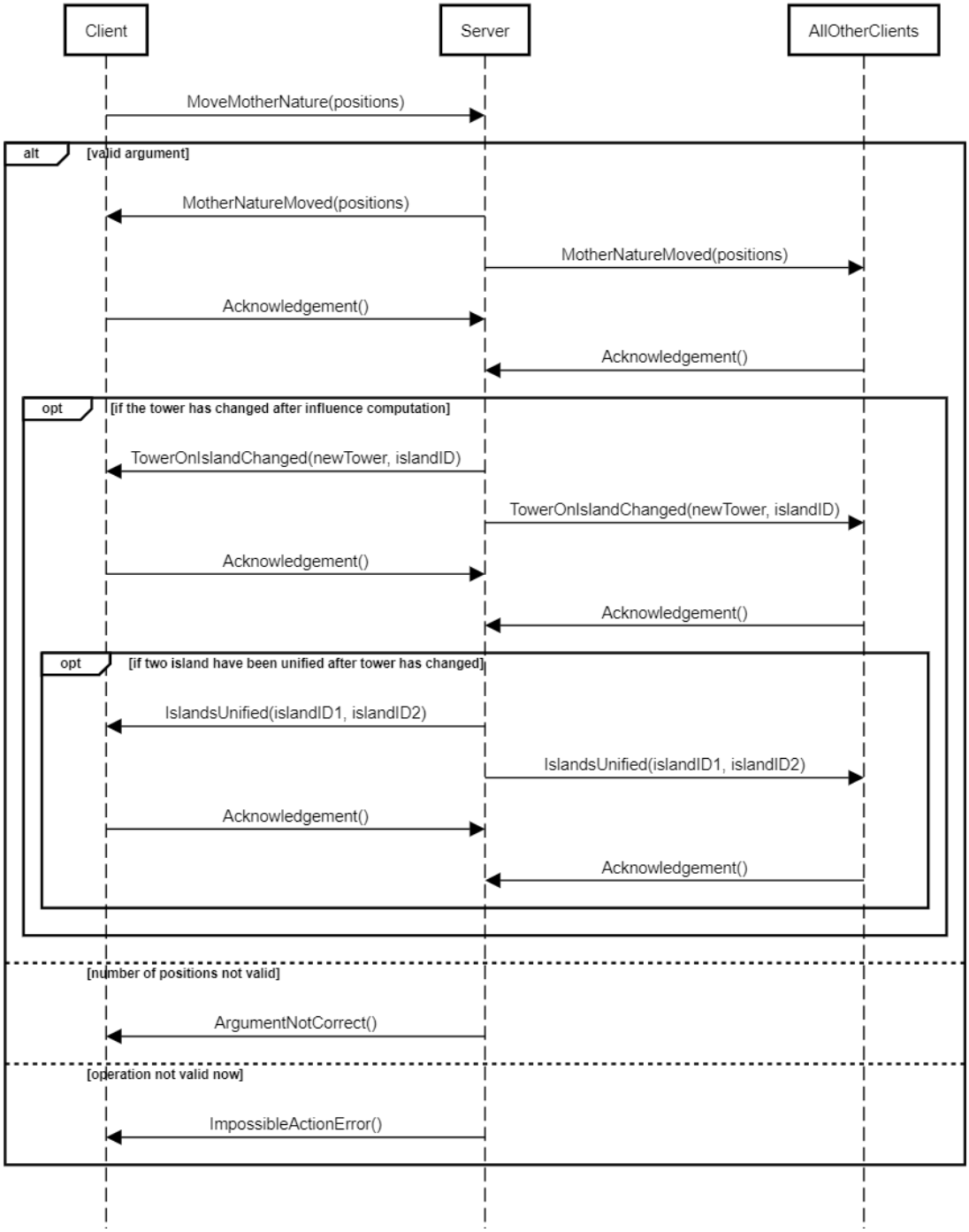
In both cases, in case of errors, the server will send back to the client an error kind of message that is different based on the specific error.




Client
Server
All Other Clients
MoveStudentsToDiningRoom(Student)
alt
[valid argument]
StudentToDiningRoomMoved(Student)
Acknowledgement()
StudentToDiningRoomMoved(Student)
Acknowledgement()
opt
ChangeProfessor(Nickname, ProfessorList)
Acknowledgement()
ChangeProfessor(Nickname, ProfessorList)
Acknowledgement()
[student does not exist]
StudentInEntraceNotExistsError()
[move not allowed now]
ImpossibleActionError()

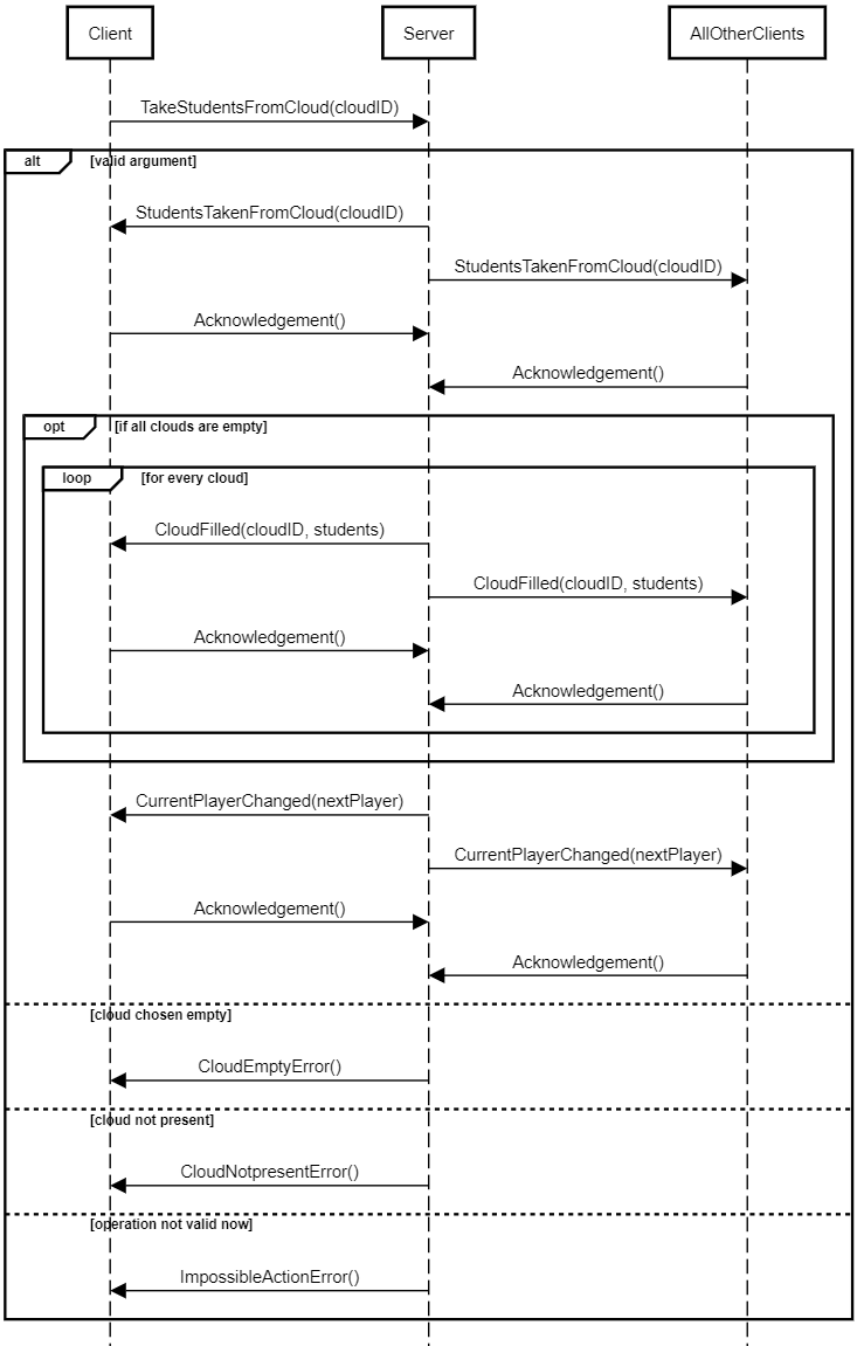
Action phase: move mother nature

In this scenario the current player sends a MoveMotherNature message to the server to move mother nature. The server sends ArgumentNotCorrect 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 acknowledgements. 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 waits for the acknowledgements. If the current player tries to do other operations in this phase it sends ImpossibleActionError 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 StudentsTakenFromCloud message and waits for the acknowledgements. 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 CloudFilled message and waits for the acknowledgements for each cloud. Finally, it notifies all clients that the current player has changed with CurrentPlayerChanged message and waits for the acknowledgements. If the client sends the ID of an already empty cloud, the server sends to this client the CloudEmptyError message and if the clients send the ID of a cloud that doesn't exist it sends the CloudNotPresentError message. Finally, if the current player tries to do a move not allowed in this phase the server sends the ImpossibleActionError 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 Winner message to all clients with the nickname of the winner and waits for acknowledgements. 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 Winner message without continuing the game, then it waits for acknowledgements.