Eryantis Protocol Documentation

Alessio Buda, Dario Mazzola, Gabriele Munafo'

Group 21

1. Messages

1.1 Ack

This message is sent when a generic message has been acknowleged.

Arguments

This messsage has no arguments.

Possible responses

This messsage has no possible responses.

1.2 Nack

This message is sent from the server to the client when a generic message has not been acknowleged. If the client receives this message, it notifies the player of the error and asks him/her to make his/her choice again.

Arguments

- TypeOfError: specifies the possible errors

Possible responses

This messsage has no possible responses.

1.3 ConnectionRequest

This message is sent from the client to the server when a player wants to start a game

Arguments

- ip: the IP address of the server.
- port: the server port number

Possible responses

- Ack: when the connection has been established.
- Nack: when the connection has not been established

1.4 NumPlayers

This message is sent from the client to the server, when the client represents the first player connected, to communicate the number of players for this game.

Arguments

- NumPlayers: the number of players for this game.

Possible responses

- Ack: when the chosen numPlayers has been accepted.
- Nack: when the chosen numPlayer is not valid.

1.5 ExpertMode

This message is sent from the client to the server, when the client represents the first player connected, to communicate whether the current game will be played in expert mode. The controller on client will check that the given input is valid.

Arguments

- ExpertMode: true if the game mode chosen is "expert mode".

Possible responses

- Ack: when the expertMode choice has been accepted.
- Nack: when the expertMode choice is not valid.

1.6 Nickname

This message is sent from the client to the server to communicate the player's chosen nickname. Input validity must be checked by the server.

Arguments

- Nickname: the player's nickname.

Possible responses

- Ack: when the chosen nickname has been accepted.
- Nack: when the chosen nickname is already in use.

1.7 AllAvailableWizards

This message is sent from the client to the server to communicate all the available wizards. It indicates the end of the nickname selection phase and the beginning of the wizard selection phase.

Arguments

- Wizards: list of all the available wizards.

Possible responses

- Wizard: to communicate player's chosen wizard.

1.8 Wizard

This message is sent from the client to the server to communicate the player's chosen wizard.

Arguments

- Wizard: the player's wizard.

Possible responses

- Ack: player's choice has been accepted
- Nack: player's choice has not been accepted.

1.9 AllNicknames

This message is sent from the sever to the client when there are 4 players correctly connected. This message is used to manage the team creation phase and it indicates the end of the wizard selection phase the beginning of the team selection phase.

Arguments

- Nicknames: list of all players' nicknames.

Possible responses

- This message has no responses.

1.10 TeamTableUpdate

This message is sent from the server to the client during the team selection phase. It is used to update information about the forming teams.

Arguments

- Nickname: the nickname of the player to whom the other information refers.
- TeamChosen: the player chosen team.
- IsTeamLeader: true if the player is leader of the team.

Possible responses

- This message has no responses.

1.11 TeamSelection

This message is sent from the client to the server when there are 4 players. It is used to communicate to the server the player's chosen team if he/she wants to be the team leader.

Arguments

- SelectedTeam: the player's selected team.
- IsTeamLeader: true if the player wants to be the leader of the selected team.

Possible responses

- Ack: when the team selection choice has been accepted.
- Nack: when the team selection choice has not been accepted.

1.12 TowerColorUpdate

This message is sent from the server to the client to comunicate all the available tower colors. If there are more than 4 players it is only sent to the team leader.

Arguments

- Colors: list of the all the available tower colors.

Possible responses

- ColorTower: the player's selected tower color.

1.13 ColorTower

This message is sent from the client to the server to communicate the player's chosen tower color.

Arguments

- ColorTower: the player's tower color.

Possible responses

- Ack: the player's choice has been accepted.
- Nack: the player's choice has not been accepted.

1.14 GetAssistantCards

This message is sent from the client to the server to require the assistant cards in current player's deck. It can be used in the planning phase but also in other circumstances when there is the need to show current player's deck.

Arguments

This message has no arguments.

Possible responses

- AssistantCards: current player's deck.

1.15 AllAssistantCards

This message is sent from the server to the client to communicate the assistant cards in current player's deck.

Arguments

- PlayersDeck: current player's deck.

Possible responses

- AssistantCard: when the player choses the assistant card to use in this round.

1.16 AssistantCard

This message is sent from the client to the server to communicate player's selected assistant card for this round. Input validity can be checked by the controller on client.

Arguments

- AssistanCard: the player's selected assisant card.

Possible response

- Ack: the choice has been accepted.
- Nack: the choice has not been accepted.

1.17 UsedAssistantCards

This message is sent from the server to the client to communicate the assistant cards used by other players in this ruond. With this information the client controller is able to check whether the card selected by the player can or cannot be used.

Arguments

- UsedCards: assistant cards used in this round by other players.

Possible responses

- AssistantCard: when the player selects the assistant card for this round.

1.18 NextPhase

This message is sent from the server to the client to communicate the end of the current round phase and the beginning of next phase

Arguments

This message has no arguments.

Possible responses

- Ack: the message has been received.

1.19 StudentsInEntrance

This message is sent from the server to the client to communicate the students in current player's entrance.

Arguments

- StudentsInEntrance: the students currently in player's entrance.

Possible responses

- MoveStudentsToDiningHall: when player wants to move a student from the entrance to the dining hall.
- MoveStudentsToIsland: when the player wants to move a student from the entrance to an island.

1.20 MoveStudentToIsland

This message is sent from the client to the server to communicate the student the player wants to move from his/hers dining hall to an island.

Arguments

- House: the house of the student to move.
- Island: the island where the student will be added.

Possible responses

- Ack: when the move has been accepted.
- Nack: when the move has not been accepted.

1.21 MoveStudentToDiningHall

This message is sent from the client to the server to communicate the student to move from the player's entrance to his/hers dining hall.

Arguments

- House: the player's student to move.

Possible responses

- Ack: when the move has been accepted.
- Nack: when the move has not been accepted.

1.22 PlayerMaxMoves

This message is sent from the server to the client to communicate the maximum number of steps of mother nature for the current player in this round.

Arguments

- MaxMoves: the maximum number of mother nature steps.

Possible responses

- MoveMother: when the player choses of how many steps to move mother nature.

1.23 MoveMother

This message is sent from the client to the server to communicate the number mother nature steps selected by the player.

Arguments

- MotherSteps: number of steps of which mother nature should be moved.

Possible responses

- Ack: when the move has been accepted.
- Nack: when the move has not been accepted.

1.24 GetCharacterCard

This message is sent from the client to the server to request the character cards in use in this game.

Arguments

This message has no arguments

Possible responses

This messagge has no responses.

1.25 AllCharacterCards

This message is sent from the server to the client to communicate all the character cards in use in this game.

Arguments

- CharacterCards: list off character cards in use.

Possible responses

- SelectCharacterCard: the player selects the desidered character card.

1.26 SelectCharacterCard

This message is sent from the client to the server to select the character card the player wants to use.

Arguments

- CharacterCard: the character chosen chard.

Possible responses

- Ack: when the character card choice has been accepted.
- Nack: when the character card choice has not been accepted.

1.27 SendMapCharacterCard

This message is sent from the client to the server to communicate the parameters needed to use the selected character card.

Arguments

- Parameters: map containing the parameters needed to use the selected character card.

Possible responses

- Ack: when the map has been accepted.
- Nack: when the map has not been accepted.

1.28 AvailableClouds

This message is sent from the server to the client to communicate the available clouds and the students on them.

Arguments

- AvailableClouds: map containing for each cloud the number of sudents of each house.

Possible responses

This message has no responses

1.29 SelectCloud

This message is sent from the server to the client to communicate the player's selected cloud.

Arguments

- Cloud: cloud selected by the player.

Possible responses

- Ack: when the move has been accepted.
- Nack: when the move has not been accepted.

1.30 SendWinner

This message is sent from the server to the client to communicate the winning player.

Arguments

- Winner: winning player:

Possible responses

- Ack: the message has been received.

1.31 ModelUpdate

This message is sent from the server to the controller to trigger an update of the view.

Arguments

This message has no arguments.

Possible responses

This message has no possible responses.

1.32 Ping

This message is sent from the server to the client to check if it is still connected.

Arguments

This message has no argumets.

Possible responses

- Pong: to signal to the server that it is still connected.

1.33 Pong

This message is sent from the client to the server to signal that the connection is sitll alive.

Argumemts

This message has no arguments.

Possible responses

This message has no possible resposes.

1.34 EndGameDisconnection

This message is sent from the server to the client to signal that the game has ended due to a client disconnection.

Argumemts

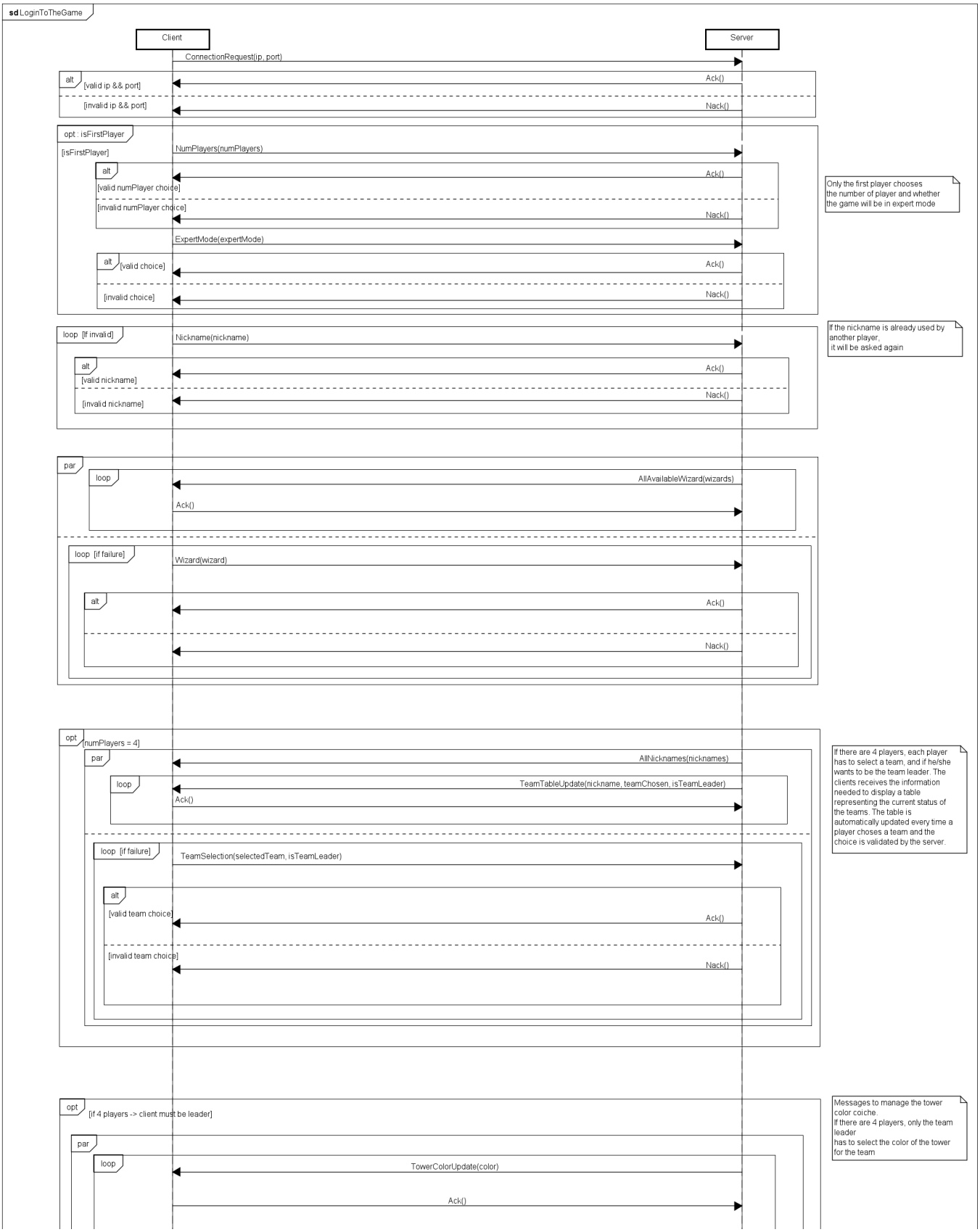
This message has no arguments.

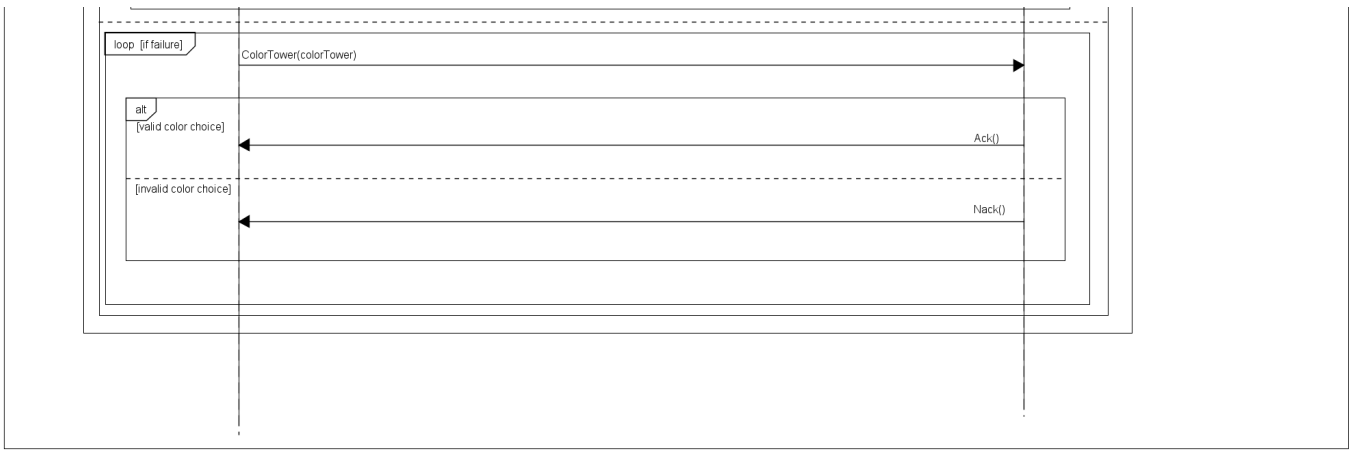
Possible responses

- Ack: the message has been received.

2. Scenarios

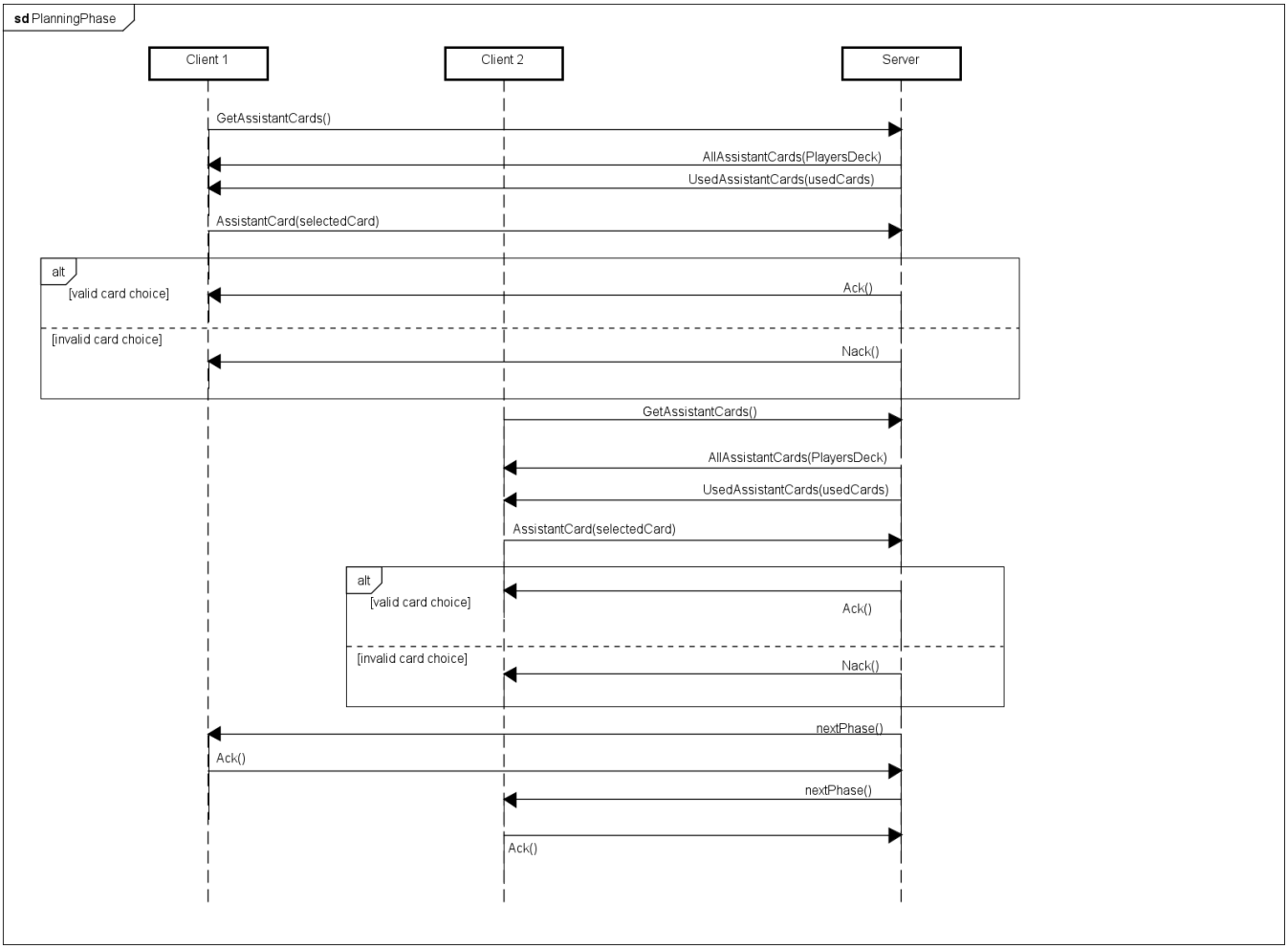
2.1 Login to the game





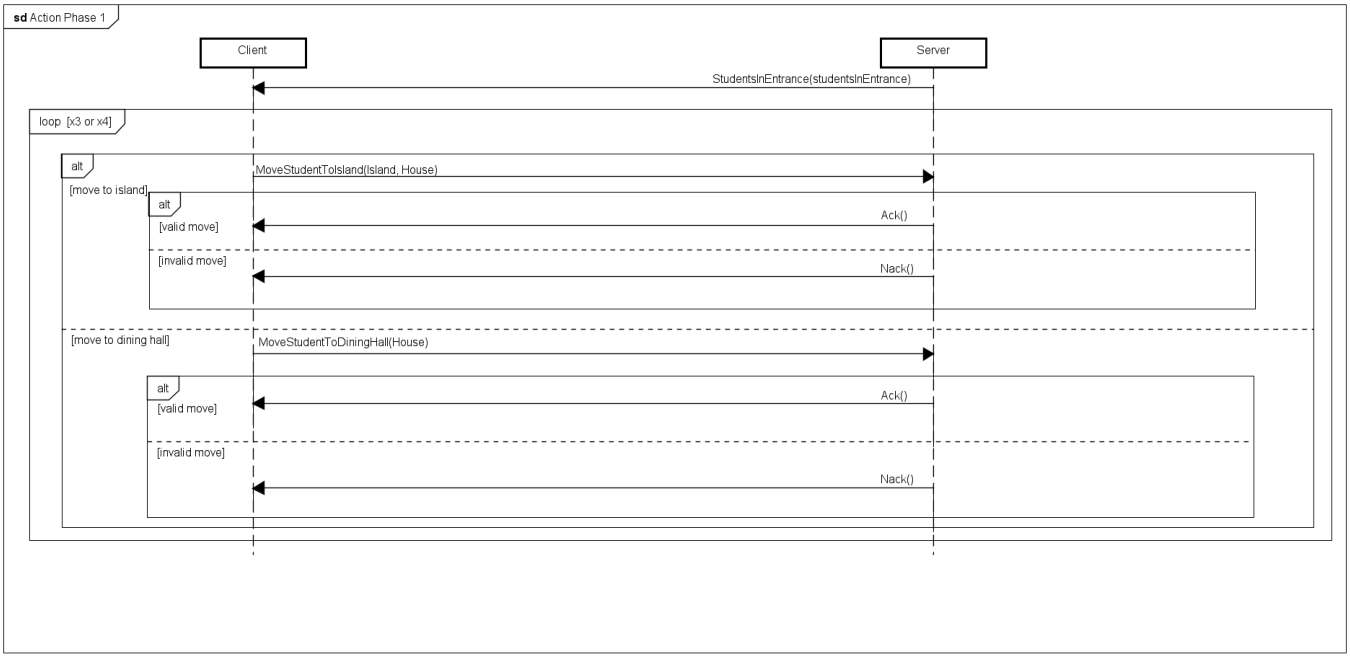
Firstly, the number of players is sent from the client to the server, who responds with an ack. Then, if the player is the first logging in, he will also comunicate which mode he wants to play, either normal or expert, and recives an ack as a response. Then, the nickname of the player is sent and the server can respond with either a success or a failure message. In the second case, the nickname selection phase starts again until the name is univoque to the player. Now the server sends all players the list containing all the wizards. Every player has to select one of them and communicate it to the server, which will respond with a success or a failure, for example if the client selects a wizard who is not present in the list given. If the wizard is not accepted, the wizard selection phase starts again until the wizard is univoqu to the player. If the wizard is accepted, the server updates every client with the new list of available wizard they can choose from. If the number of players is four, that means it is a team game. The server sends every client the list of the players' nicknames, so that they can build a table, and recives an ack. Every player has to select the number of team they want to play in and whether or not they want to be the leader of that team. The server responds with either a success or a failure, for example if the team is already full or already has a leader. In the second case, the team selection phase starts again until an acceptable team and decision on the team leader is given. If the team and decusion on the team leader are accepted, the server updates every client with a map containing the nickname, team and decision on the team leader of that player, so that they can update the table row. If it is not a team game or the player is the team leader, the server sends the list of all the tower colors. Every player has to select a tower color and the server responds with either a success or a failure, for example if the color is already taken. In the second case, the tower color selection phase starts again until every player or team has an univoque tower color.

2.2 Planning phase



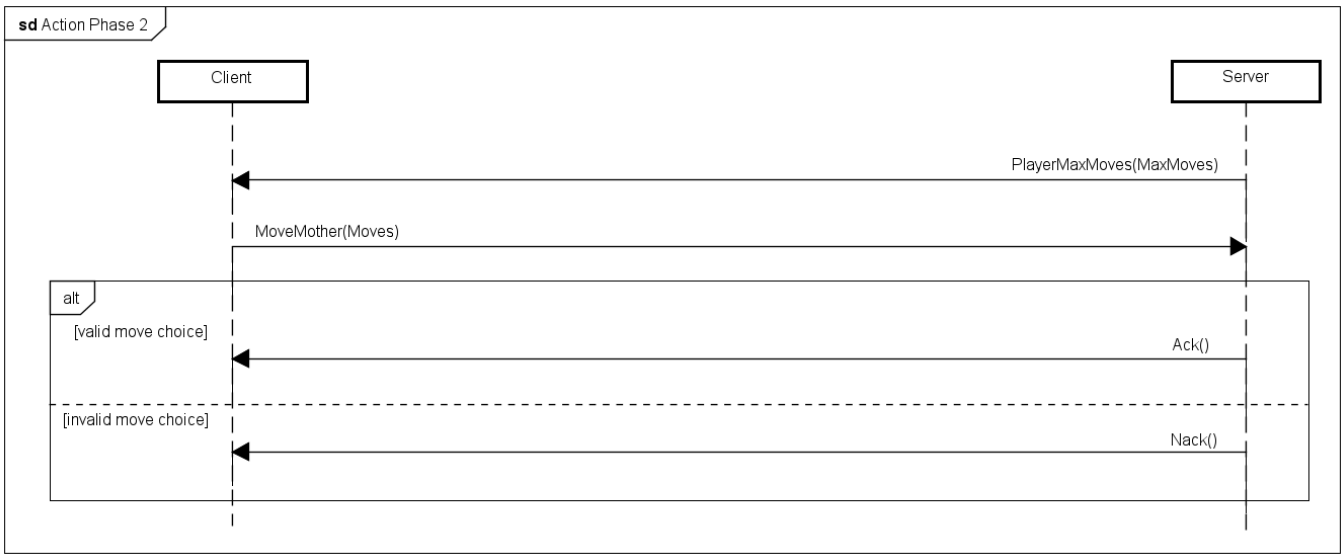
The sequence diagram shows the interaction between 2 clients and the server in the choice of the Assitant Cards. The choice is sequential, therefore the clients will wait their turn to use a card that respects the constraints imposed by the rules. The controller on the client, before showing the cards to the player, sends a message to the server asking to send the list of cards that the current player has in his deck. The controller responds by sending the player's deck and a list with the cards used by the other players that turn. The same scenario is repeated for a second client and can be extended if 3 or 4 clients are connected.

2.3 Action phase 1



The sequence diagram shows the interaction between a client and the server in the first move of the action phase. In this phase the player must move 3 or 4 students and can choose to move them in any order to an island or their DiningHall. The loop will be performed 3 or 4 times, based on the number of players. The controller on the client will send, depending on the player's choice, a `MoveStudentToIsland` or `MoveStudentToDiningHall` message. The controller can respond in both cases with failure or success messages to indicate to the client to repeat the choice or continue with the next move.

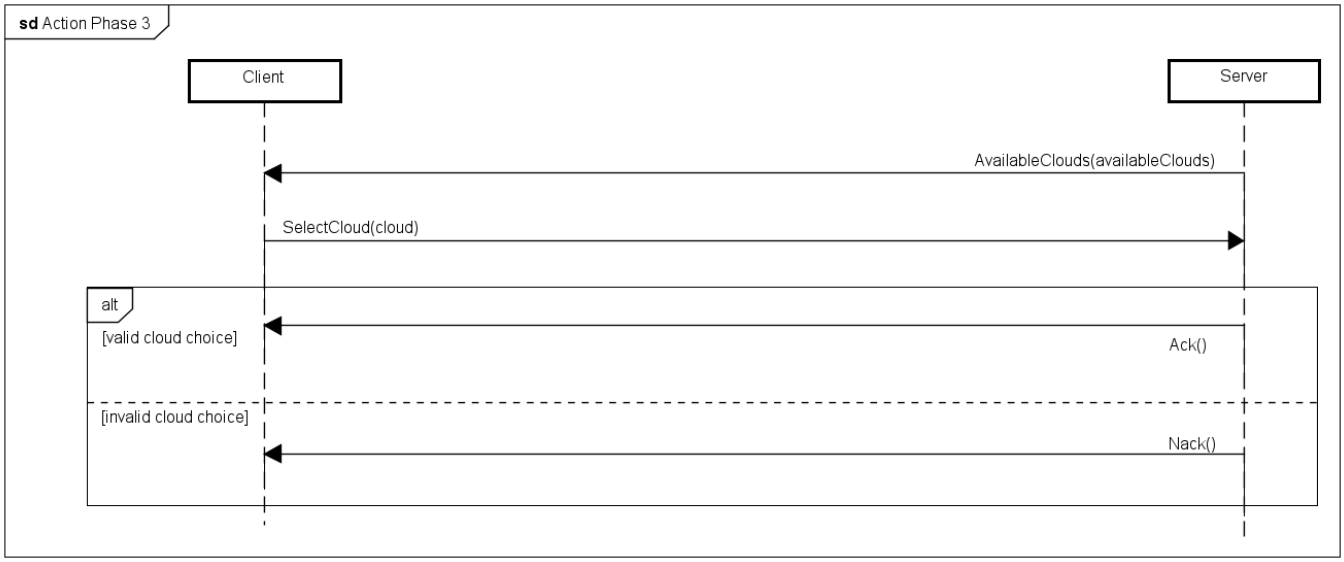
2.4 Action phase 2



This diagram shows the interaction between the client and the server when the player is chosing of how many steps to move mother nature. The server sends the client the maximum possible number of steps as indicated in the player's selected assistant card (potentially modified according to the effect of character cards, in case the game is in expert mode). The player responds with the desired number of steps. Input validity is checked

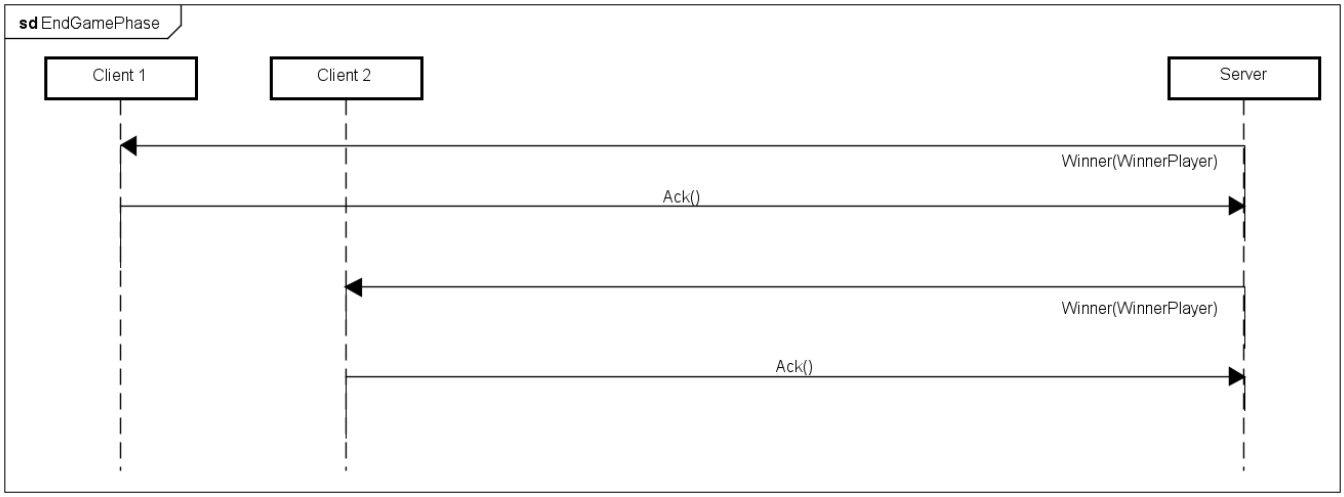
both on the client and the server. If the input is correct, the server responds with a message of success, wheras when the given input is not valid the server responds signaling a failure thus asking the client to send a new message with valid input.

2.5 Action phase 3



This diagram shows the interaction between the client and the server when the player is chosing the cloud from which to refill the entrance of his/hers dashboard. The server sends to the client all the availbale clouds and the students on them. The player choses a cloud and communicates his/hers choice to the server. Input validity is checked both on the client and the server. If the input is correct, the server responds with a message of success, wheras when the given input is not valid the server responds signaling a failure thus asking the client to send a new message with valid input.

2.6 End game phase



This diagram shows the interaction between the client and the server when the server is signaling the winning player and thus the end of the game. The client respondes whith an ack to let the server know that the message has been correctly received.