Eryantis Protocol Documentation

Francesco Arcuri, Giulia Prosio, Carlo Ronconi
Group 7

May 2, 2022

1 Messages

1.1 Handshake

Tells the server that a new client has connected successfully

Arguments

1. This message has no arguments.

Possible responses

- Acknowledgement: the message has been received

1.2 Acknowledgement

Tells the client that the massage has been received successfully.

Arguments

1. This message has no arguments.

Possible responses

- This message has no responses.

1.3 GetPreferences

Asks a player to set the preferences for the next game.

Arguments

1. This message has no arguments.

Possible responses

- SetPreferences: preferences chosen by the player (easy or hard game, number of players).

1.4 SetPreferences

Sets the preferences for the next game.

Arguments

1. NumOfPlayers: number of the players of the next game
2. GameMode: game mode for the next game

Possible responses

- This message has no responses.

1.5 GetNickname

Asks a player to choose a nickname.

Arguments

1. This message has no arguments.

Possible responses

- SetNickname: nickname of the player.

1.6 SetNickname

Sets the nickname of a player.

Arguments

1. Nickname: nickname of the player

Possible responses

- This message has no responses.

1.7 InvalidNickname

Tells the player that the nickname he chose is invalid becose someone else has already used it.

Arguments

1. This message has no arguments.

Possible responses

- SetNickname: new choice of nickname by the user.

1.8 GameState

This message is sent from the server to the client with the updated game board.

Arguments

1. Bag: current number of student pawns in the bag
2. ProfessorOwners: a map of the colors and their owners (available or the UUID of the current owner)
3. Clouds: adjourned list of the cloud UUID and its student pawns
4. Islands: adjourned list of the islands and their student pawns

5. Entrances: adjourned representation of each player's entrance
6. DiningRooms: adjourned representation of each player's dining room
7. AssistantDecks: map of players and their adjourned deck of assistant cards
8. CoinsMaps: maps of players and their adjourned stash of coins
9. CharacterCards: map of the three randomly chosen character cards and a boolean stating whether the card has already been played (its cost is incremented of one coin)
10. PlayedAssistantCards: set of player's UUID and his last assistant card played
11. MotherNaturePosition: UUID of the island on which mother nature is set
12. IslandOwners: map of the UUID of an island and the UUID of the player who owns it currently. The owner's UUID could be null, which means that no player is currently owning the island
13. Nicknames: list of the players' nicknames

Possible responses

- This message has no responses

1.9 GetAssistantCard

Asks a player to choose an assistant card.

Arguments

1. This message has no arguments.

Possible responses

- SetAssistantCard: the cloud that a player chose.

1.10 SetAssistantCard

Sets the assistant card one player has chosen in a turn.

Arguments

1. NumCard: number of the assistant card selected by the player

Possible responses

- This message has no responses.

1.11 InvalidAssistantCard

Tells a player that he has chosen an Assistant card that has been already played in that turn by another player.

Arguments

1. This message has no arguments.

Possible responses

- SetAssistantCard: the cloud that a player chose.

1.12 PlayerTurn

Tells the view who is the next player who has to play.

Arguments

1. Nickname: nickame of the player who has to play next

Possible responses

- This message has no responses.

1.13 ChooseCharacter

Asks a player to choose a character card.

Arguments

1. This message has no arguments.

Possible responses

- ChosenCharacter: the character that a player has chosen.

1.14 ChosenCharacter

This message is sent from the client to the server when a character card is chosen from the player.

Arguments

1. ChosenCharacter: the UUID of the chosen character card

Possible responses

- InvalidCharacterChoice: the message has been received but the choice is not valid

1.15 InvalidCharacterChoice

Tells that the player has chosen a character that costs more than the coins he has.

Arguments

1. This message has no arguments.

Possible responses

- ChosenCharacter: new choice of a character with a lower cost.

1.16 MoveStudent

Asks a player to choose the color of the student to be moved and where to move it.

Arguments

1. This message has no arguments.

Possible responses

- MovedStudent: where and which student to be moved.

1.17 MovedStudent

Tells what color of student a player wants to move to either the entrance or one of the islands.

Arguments

1. Color: color of the student to be moved
2. IslandId: if null the student has to be moved to the entrance, otherwise it contains the id of the island where the student has to be moved to

Possible responses

- This message has no responses.

1.18 MoveMotherNature

Asks a player to choose how may steps to move mother nature.

Arguments

1. This message has no arguments.

Possible responses

- MovedMotherNature: the number of steps the player has chosen.

1.19 MovedMotherNature

Tells how many steps of mother nature the player has chosen.

Arguments

1. MotherNatureSteps: number of steps of mother nature

Possible responses

- This message has no responses.

1.20 InvalidMotherNatureMove

Tells a player that he has asked to move mother nature more than it is allowed by the card he has played in the planning phase.

Arguments

1. This message has no arguments.

Possible responses

- MovedMotherNature: new choice of how may steps mother nature has to be moved.

1.21 ChooseCloud

Asks a player to choose a cloud.

Arguments

1. This message has no arguments.

Possible responses

- ChosenCloud: the cloud that a player has chosen.

1.22 ChosenCloud

This message is sent from the client to the server when a player chooses a cloud at the end of his action phase.

Arguments

1. Cloud: the UUID of the chosen cloud

Possible responses

- This message has no response

1.23 GameOver

This message is sent from the server to the client when one of the possible end of game conditions is met.

Arguments

1. Winner: the UUID of the winner

Possible responses

- this message has no responses

1.24 AskPlayAgain

Asks a player if he wants to play again.

Arguments

1. This message has no arguments.

Possible responses

- SetPlayAgain: true if a player wants to play again

1.25 SetPlayAgain

Tells if the player wants to play another game.

Arguments

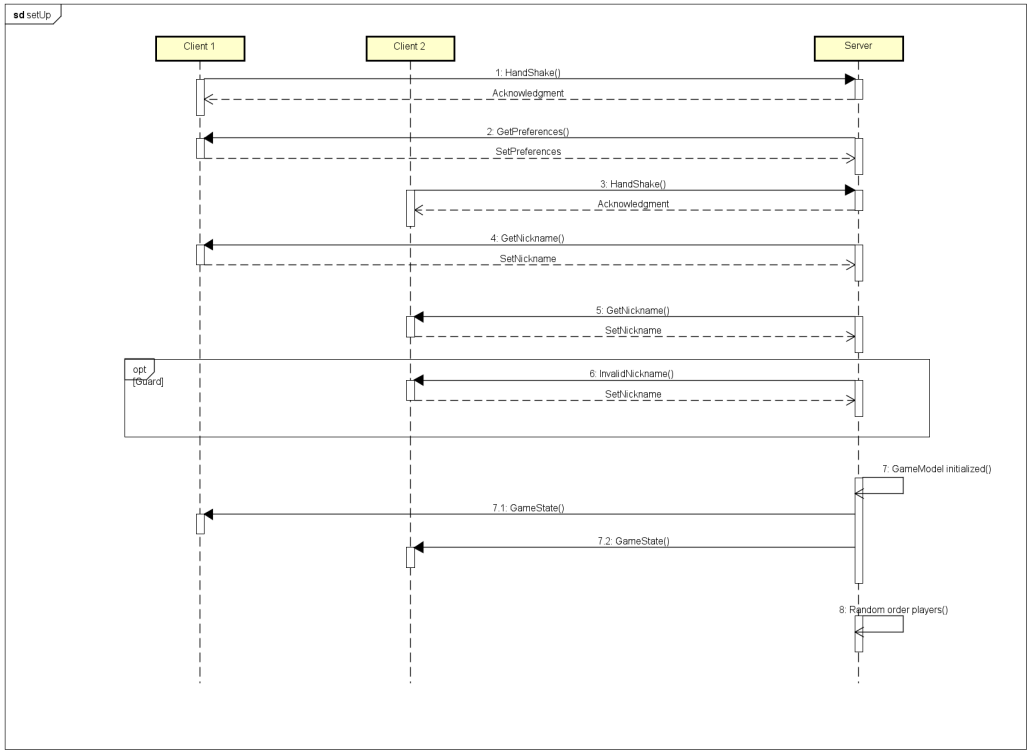
1. PlayAgain: true if the player wants to play again

Possible responses

- This message has no responses.

2 Scenarios

2.1 Setup



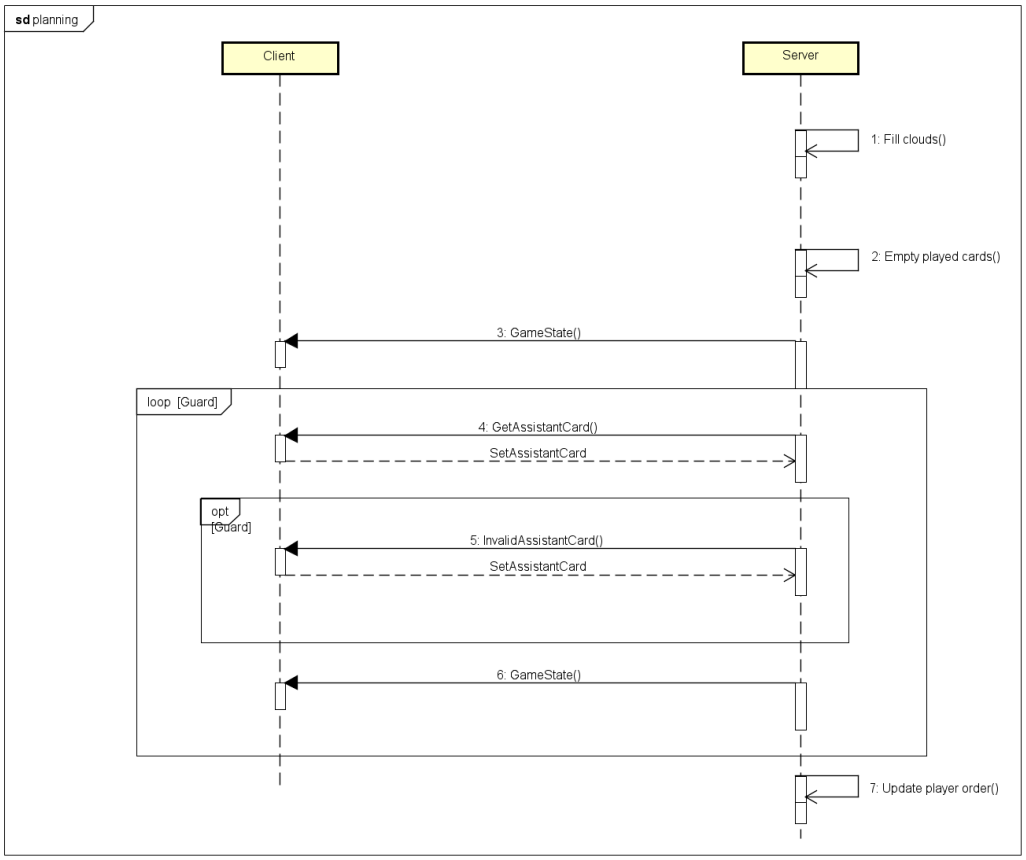
First, the Handshake message is sent from the client to the server, meaning that it is ready to be connected. If the server answers to this message with an Acknowledgement one, this means that the connection has been established and the server is now ready to accept game-related messages.

Once the first client is connected to the server, the latter sends to the client a GetPreferences message, asking the client (first player) the game general settings, in particular the number of players (two or three) and the variant of the game (easy or expert).

After that the server establishes the connection with the other clients, following the same handshake-acknowledgement messages path.If the number of connections created is equal to the number of players chosen by the first player, the server sends a GetNickname message for each client and waits for the SetNickname message that signals the nickname configuration of the player in question.

In the event that a player's nickname had already been chosen by a previous one, the server sends an InvalidNickname message to the client in question, waiting for a response provided by the SetNickname message. Finally, having all the information needed, the controller initializes the game and sends a GameState message to all the players, containing the initialized Board and, for each client, their School Board. According to the rules of the game, the player who starts the game will be chosen randomly.

2.2 Planning phase



At the beginning of the Planning Phase the server fills all the clouds in the model by randomly drawing students from the bag and clears the played cards map, which keeps track of the cards each player has played in the last turn. After that, a GameState message is sent to each client with the updated Board.

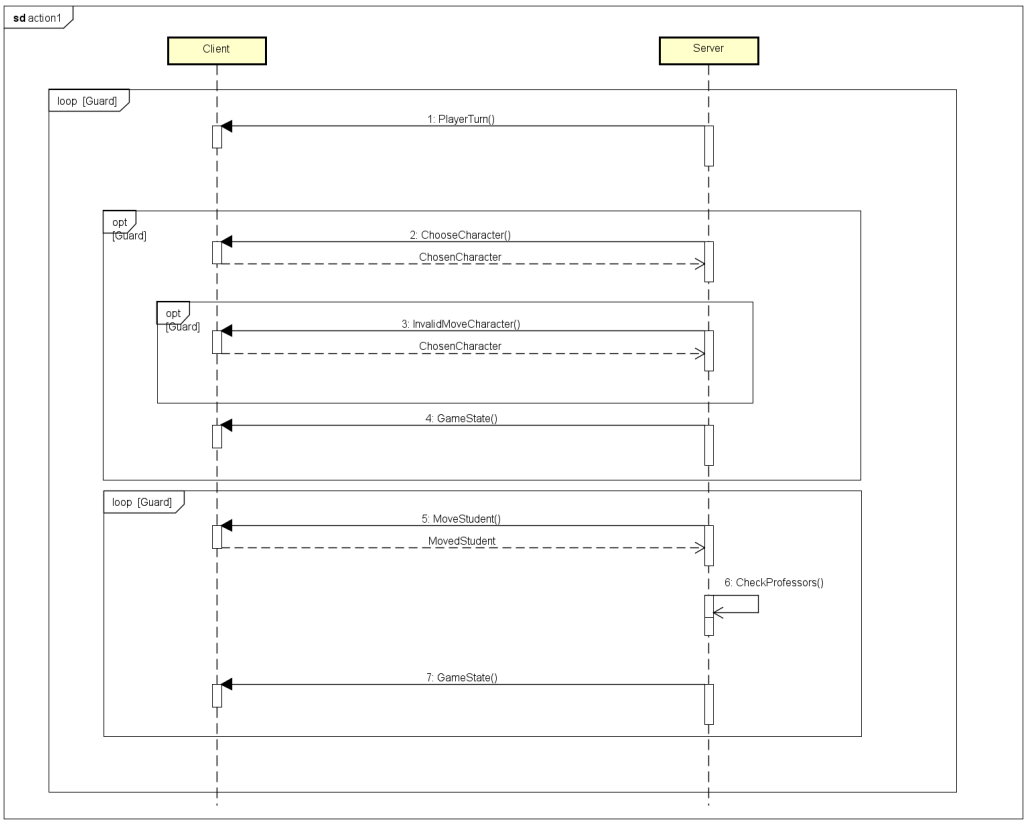
At this point, we start looping on each client connected, asking one by one what Assistant Card they want to play (with GetAssistantCard and SetAssistantCard messages). The order of the loop is decided by the results of the previous turn. For the first turn, the first player is chosen randomly by the server and the following ones are called in clockwise order.

If the card drawn by a player has already been played by another client

in the same turn, the server sends an InvalidAssistantCard message to the interested client, to which the client responds with a new SetAssistantCard message. After each players choice has been stored by the server, a GameState message is sent to all the players, displaying the new elements.

After all the players have chosen their cards, we can finally update the player order for the action phase and for the next planning phase.

2.3 Action phase (1 of 2)



Note: for graphical reasons, we had to split the action phase diagram in two parts, but they are intended as a single scenario.

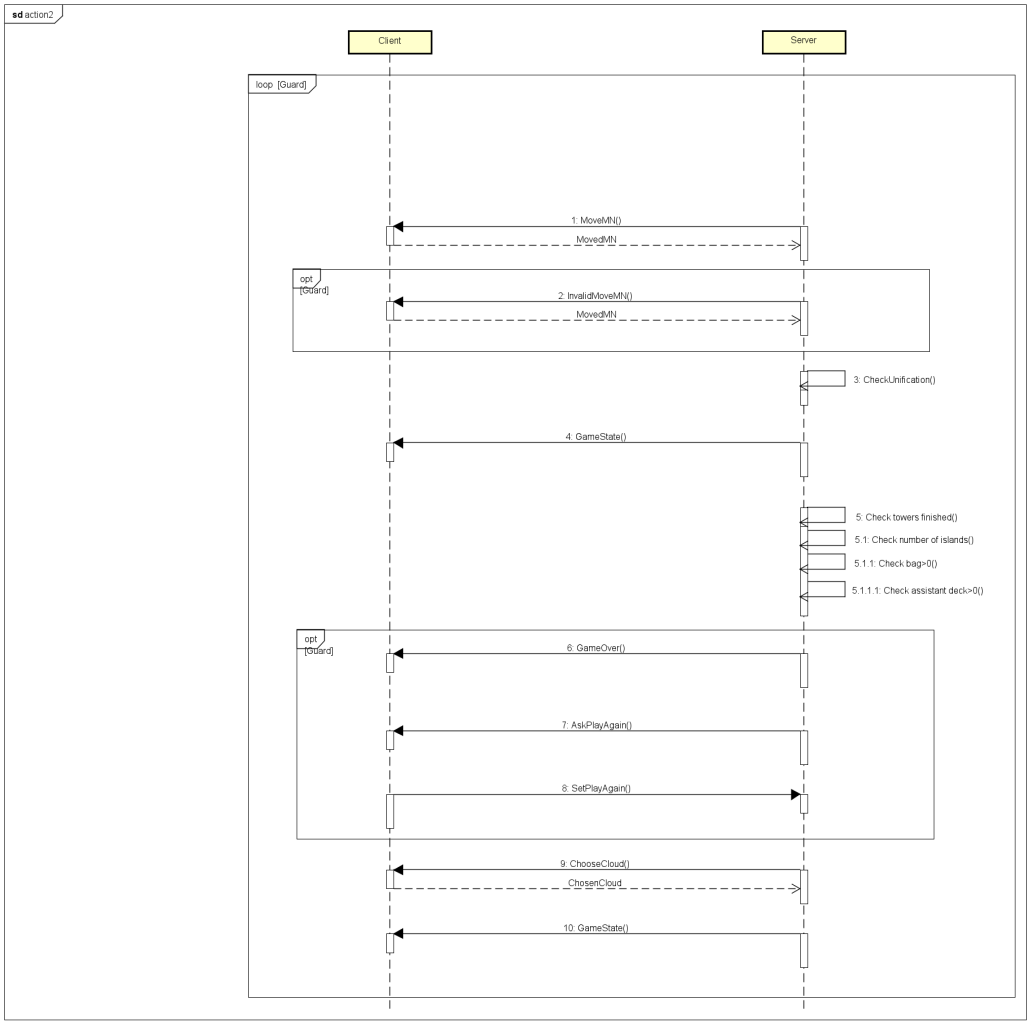
In the action phase, we again loop for each client/player following the order we evaluated in the planning phase. Firstly, a PlayerTurn message is sent to all clients to inform whose turn it is.

After that, if the game mode selected during the setup is hard, we check if the player has any coins: if so, the server asks to choose a character card (ChooseCharacter and ChosenCharacter messages, the latter can also tell if the player doesn't want to purchase a character). If the character card chosen by the player costs more than he can afford, the server sends an InvalidMoveCharacter message to the client, who then sends a different choice with another ChosenCharacter message. Then, we sent the updated GameS-

tate to all the clients.

Now, we need to ask the player to choose which students he wants to move and where he wants them moved. So, looping for three times (four if there are three players), the server sends a MoveStudent message and the client responds with a MovedStudent message. The server updates accordingly the professor owners and sends to the clients the updated GameState.

2.4 Action phase (2 of 2)



The second diagram depicts the remaining part of a player's Action Phase turn. After having placed the student pawns from the Entrance of his School Board to the Dining Room and/or the Island tiles, the player is now asked to move Mother Nature.

The possible number of steps, which represents the player's choice, goes from one to a maximum number fixed by the Assistant Card played. If the number of steps input by the player exceeds this bounds, an InvalidMN-Move message is sent by the controller, which requires as a response a new MovedMN message. A new GameState message is sent to all the clients, depicting the new situation.

Now the controller checks if a condition to end the game is met. If not, the server starts the Action Phase of the second player, repeating the path shown.

If it is, a GameOver message is sent, signaling the winner, and the server asks to all the clients if they would like to play again. If the answer is negative, the connection between server and client is closed; else the game starts again and a GetPreferences message is sent.