Eriantys Protocol Documentation

Federica Tommasini, Giuseppe Bonanno, Angelo Zagami
Group 18

May 3, 2022

The communication between client and server is implemented through the usage of sockets, which allows the exchange of messages over the network using the TCP protocol. The messages are defined as implementations of three interfaces: "ClientToServerMessage", "ServerToClientMessage" and "SetUpMessage" that, respectively, represent the information that have to be sent from client to server and vice versa and the messages needed for the setup of the match. The messages are sent over the network as serialized objects by the "ClientHandler" and, once they are received by the server, they are parsed by the "GameHandler" that will identify the action that has to be made accordingly to the type of message.

1 Messages

1.1 RequestNickname

Once the connection with the client is created, the server asks the client to choose a nickname. The class representing this message has a static attribute that contains the string "Choose a nickname".

Possible responses

- ChooseNickname: in order to start the game the client has always to answer with this message.

1.2 ChooseNickname

The client sends a message containing the string chosen as a nickname.

Arguments

1. nickname: string representing the player's nickname.

Possible responses

- Error: an error of the type "DUPLICATENICKNAME" is sent to the client if the nickname is already associated with another player connected to the server.
- Error: an error of the type "INVALIDNICKNAME" is sent to the client if the nickname contains invalid symbols.
- ActionValid: a message, containing the string "The nickname has been registered successfully", is sent to the client in all the other cases.

1.3 ChooseMatch

The server sends a message to the client, asking to choose a match to join. The class representing this message has a static attribute that contains the string "Select a match to join or type 0 to create new match > ".

Arguments

1. matches: a list of strings that represent the matches that are available on the server.

Possible responses

- SelectMatch: this response is always needed to select the match and proceed with the game.

1.4 SelectMatch

The client sends a message to inform the server of which match the player wants to join.

Arguments

1. match: a int value greater than zero, which represent the ID of an already existing match that the user wants to join, or it is equal to zero if the user wants to create a new match.

Possible responses

- ActionValid: a message, containing the string "Match selected with success" is sent to the client if the user select a valid value among the ones previously indicated by the server.
- Error: an error of the type "CHOICENOTVALID" is sent to the client if the integer value inserted by the user is not associated with an available match.

1.5 RequestSetUp

The server sends a message to the client, asking to choose the number of players and the mode of the match. The class representing this message has a static attribute that contains the string "Select the number of players and the mode of the match".

Possible responses

- SelectModeAndPlayers: a message is sent to the server, if the player insert a valid value for the number of players and a valid option for the mode.

1.6 SelectModeAndPlayers

The player who created a new match has to indicate in a message the number of players for the match and whether the mode is "expert" or not.

Arguments

1. numberOfPlayers: int value that represents the number of players chosen for the match.

2. mode: a boolean value that is true if the player chooses the expert mode or false otherwise.

Possible responses

- ActionValid: a message, containing the string "Match created with success" is sent to the client if the user select a valid value for the number of players and the mode.
- Error: an error of the type "CHOICENOTVALID" is sent to the client if the values inserted by the user are not valid.

1.7 WaitForOtherPlayer

The server sends a message to the client to inform the user that the match will start as soon as the other players join. The class representing this message has a static attribute that contains the string "Wait for other players".

1.8 SelectWizard

The server sends a message to the client asking him to select one of the wizards' names available. The class representing this message has a static attribute that contains the string "Select a wizard among the ones available".

Arguments

1. availableWizards: a list of strings containing the names of the wizards available.

Possible responses

- ChooseWizard: the client sends a message to the server to communicate the wizard chosen by the player.

1.9 ChooseWizard

The client sends a message to the server containing the name of the wizard chosen by the player.

Arguments

1. wizard: an enum value that represent the wizard name chosen by the player.

Possible responses

- ActionValid: a message, containing the string "You have been successfully associated with the wizard that you chose" is sent to the client if the user select a valid value for the number of players and the mode.
- Error: an error of the type "CHOICENOTVALID" is sent to the client if the value inserted by the user is not valid.

1.10 PlayersInfo

A message is sent to the client to pass the information about the others players.

Arguments

1. playerNicknames: a list of strings containing the nicknames of all the players connected to the match.
2. numberOfPlayers: an integer value corresponding to the number of players in the match.
3. expertMode: a boolean value that is true if the player chooses the expert mode or false otherwise.
4. mapPlayerWizard: an hashmap containing the mapping between the players' nicknames and the wizards' name to which they are associated.
5. mapTowerToPlayer: an hashmap containing the mapping between the players' nicknames and towers that have been associated to them.

1.11 MatchCreated

The server sends a message to the client containing the information about the initial position of mother nature and the students initially placed on the islands.

Arguments

1. motherNaturePosition: an integer value that indicates the initial position of mother nature.
2. mapStudentIsland: an hashmap containing the mapping between the integer value that represent the island and the color of the single student that is placed on such island at the beginning of the match.

1.12 SetUpSchoolStudent

The server sends a message to the client to communicate the initial configuration of each school entrance associated with a player. This message will be sent for each player.

Arguments

1. playerNickname: a string that represents the nickname of the player.
2. entranceStudents: an hashmap containing the mapping between the number of and the color of the students initially placed in the entrance of the player's school.

1.13 SetUpCharacterCard

If the match is created in expert mode, the server sends a message to the client communicating the three characters card randomly chosen between the twelve available.

Arguments

1. characterCards: an array of strings containing the names of the three cards selected.

1.14 YourTurn

The server sends a message to the client to inform the player that it is his turn. The class representing this message has a static attribute that contains the string "it's your turn".

1.15 IsTurnOfPlayer

The server sends a message to the client to inform that it is the turn of another player.

Arguments

1. playerNickname: a string containing the nickname of the player whose turn is.

1.16 SelectAssistantCard

The server sends a message to the client asksing to select an assistant card to play when is the turn of the player associated with that client.

Arguments

1. availableCards: a list of strings that represent the names of the assistant cards still available in the player's deck.

Possible responses

- PlayAssistantCard: the player sends a message selecting the card he wants to play.

1.17 PlayAssistantCard

The client sends a message to the server containing the information about the assistant car that the player wants to use.

Arguments

1. cardValue: integer value that correspond to the value of the assistant card that the player wants to use.

Possible responses

- ActionValid: a message, containing the string "You played the assistant card selected" is sent to the client if the user select a valid value for the card.

- Error: an error of the type "NOTYOURTURN" is sent to the client if the message is sent during the turn of another player.

1.18 MoveStudent

The client sends a message to the server containing the information about which student he wants to move and where to.

Arguments

1. MoveTo: an enum value that indicates whether the student has to be moved to an island or to a school.
2. Color: an enum value that indicates the color of the student that has to be moved
3. islandPosition: an integer value that represents the island position to which the student has to be moved

Possible responses

- ActionValid: a message, containing the string "You have successfully moved the student" is sent to the client if the move is valid.
- Error: an error of the type "NOTYOURTURN" is sent to the client if the message is sent during the turn of another player.

1.19 MoveMotherNature

The Client sends a message to the server containing the information about the steps of which the player wants to move mother nature.

Arguments

1. steps: an integer value that represents the number of positions to be crossed by mother nature.

Possible responses

- ActionValid: a message, containing the string "You have successfully moved mother nature" is sent to the client if the move is valid.
- Error: an error of the type "NOTYOURTURN" is sent to the client if the message is sent during the turn of another player.

1.20 UpdateMessage

The server sends a message to the clients, to inform all the players of the changes applied during the turn of one of them. The message contains information about such changes, that can either be movements of students and mother nature, the conquest of an island or the merging of two islands.

Arguments

1. change: an object of the class BoardChange, which implements serializable and has different constructors accordingly to the different kinds of updates that may be needed.

1.21 UseCharacterCard

The client sends a message at any time during the action phase of the player's turn, indicating which character card he wants to play. The message will contain other attributes accordingly to the specific character card which is used.

Arguments

1. asset: a string, which is the name of the character card that the player chooses to use.

Possible responses

- ActionValid: a message, containing the string "You have successfully used the card" is sent to the client if it is its turn.
- Error: an error of the type "NOTYOURTURN" is sent to the client if the message is sent during the turn of another player.

1.22 GenericMessage

The server sends a message to the client containing a string that it is set accordingly to the message that the server wants to communicate.

Arguments

1. msg: a string that represents a message.

1.23 YouWin

The server sends a message to the client to communicate that he won. The class representing this message has a static attribute that contains the string "Congratulations, you won!".

1.24 OtherPlayerWins

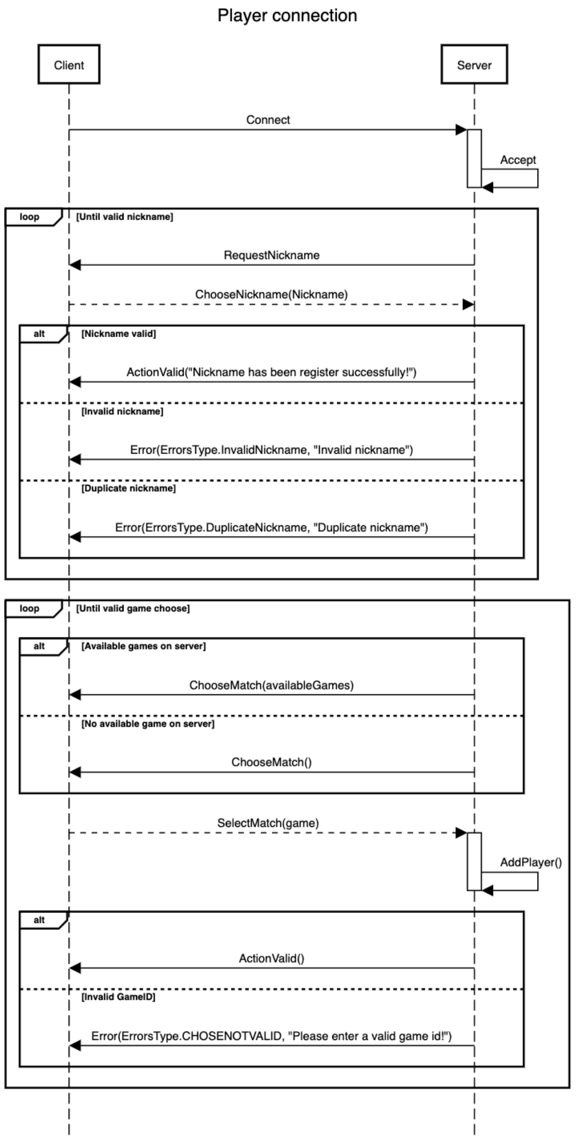
The server sends a message to the client communicating the winner of the match.

Arguments

1. player: a string that represents the nickname of the winner.

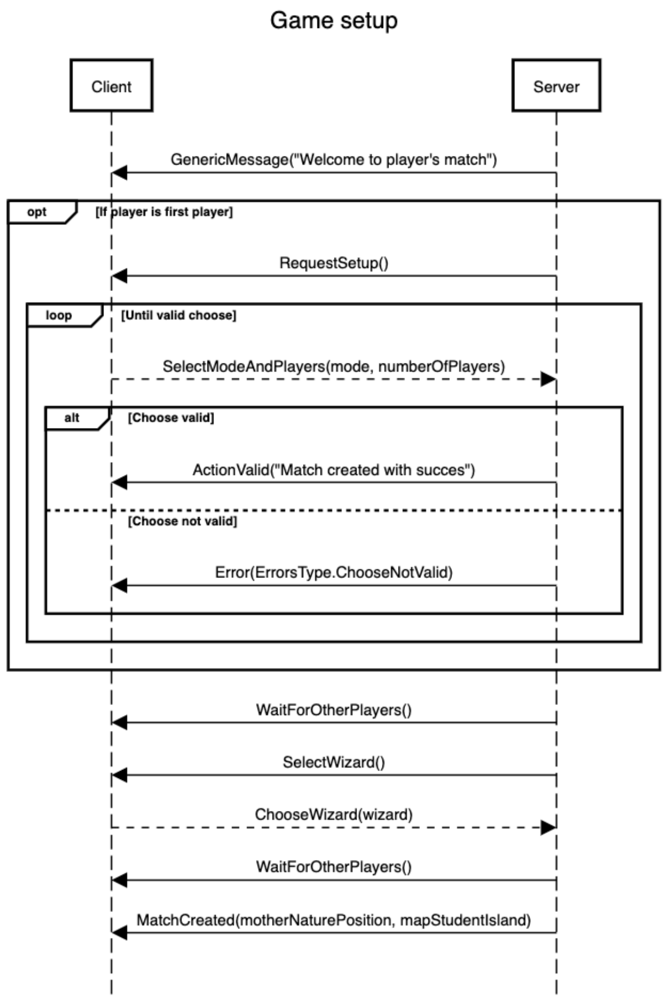
2 Scenarios

2.1 Player connection


Player connection
Client
Server
Connect
Accept
loop
[Until valid nickname]
RequestNickname
ChooseNickname(Nickname)
alt
[Nickname valid]
ActionValid("Nickname has been register successfully!")
[Invalid nickname]
Error(ErrorsType.InvalidNickname, "Invalid nickname")
[Duplicate nickname]
Error(ErrorsType.DuplicateNickname, "Duplicate nickname")
loop
[Until valid game choose]
alt
[Available games on server]
ChooseMatch(availableGames)
[No available game on server]
ChooseMatch()
SelectMatch(game)
AddPlayer()
alt
ActionValid()
[Invalid GameID]
Error(ErrorsType.CHOSENOTVALID, "Please enter a valid game id!")

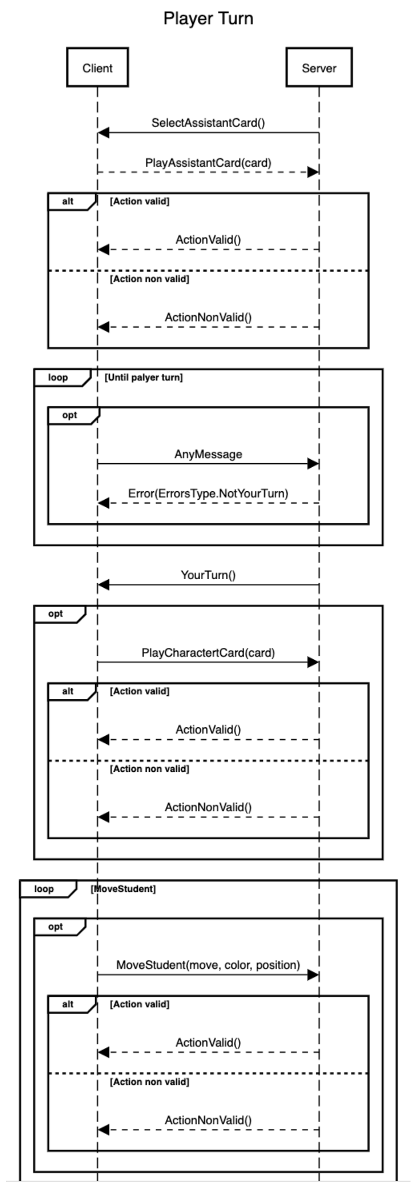
In the player connection scenario, the client requests the connection to the server, which accepts and sends a RequestNickname message. The class representing this message has a static attribute that contains the string "Choose a nickname". The Client replies with a ChooseNickname message which is a class with a String attribute that contains the chosen nickname. Once received the nickname chosen by the client, the server processes the request and responds successfully if the nickname respects the name constraints, and is not duplicated. Otherwise it responds with the InvalidNickname or DuplicateNickname error. The nickname selection operation is in a loop as it will be repeated until a valid nickname is obtained. Once the client has been successfully registered, the server sends a message of the type ChooseMatch, which presents a list of strings with all matches that can be started on the server, and it allows you to create a new game. If there is no bootable match list then the message is sent without the list and the client is asked to create a new match. The client chooses the game by sending a message of the type SelectMatch and the server at this point adds the client if everything goes correctly otherwise it sends an error of the type CHOICENOTVALID. The match choice is also in a loop as it will be required until a valid game is selected.

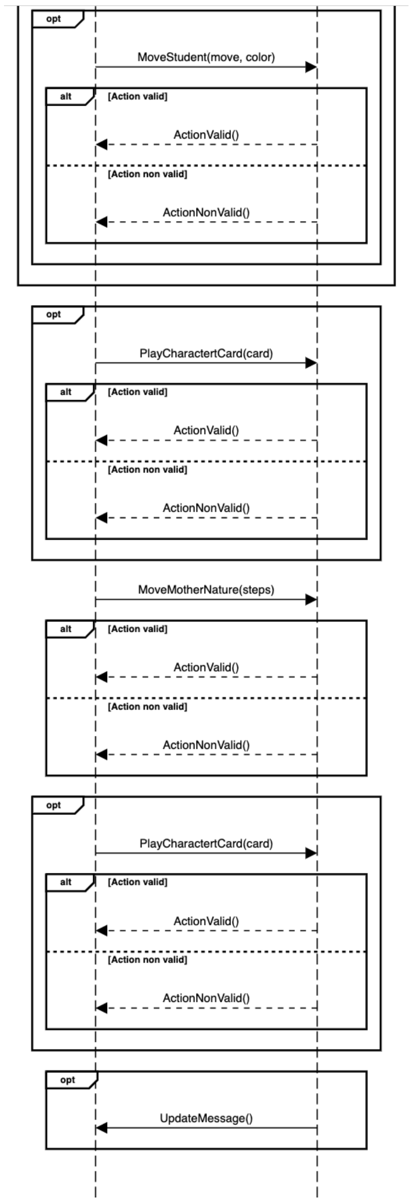
2.2 Game set up


Game setup
Client
Server
GenericMessage("Welcome to player's match")
opt
[If player is first player]
RequestSetup()
loop
[Until valid choose]
SelectModeAndPlayers(mode, numberOfPlayers)
alt
[Choose valid]
ActionValid("Match created with succes")
[Choose not valid]
Error(ErrorsType.ChooseNotValid)
WaitForOtherPlayers()
SelectWizard()
ChooseWizard(wizard)
WaitForOtherPlayers()
MatchCreated(motherNaturePosition, mapStudentIsland)

In the game set up scenario, the server sends a GenericMessage message welcoming the client to the game selected in the player connection scenario. If the client has decided to create the game it will be the first player, then the server will send a message of the type RequestSetup to ask the client to choose the game mode and the number of players. The client will reply with a message of the type SelectModeAndPlayers specifying the required parameters. In the event that the player has decided to join a game in progress this part will not be requested and the server will send the client two messages, one to inform that it is waiting for new players, therefore a WaitForOtherPlayers message, and another for the choice of the wizard, through a message of the type SelectWizard. The client will choose the wizard and will send a ChooseWizard type message to the server. After choosing the wizard the server will inform the client that it is still waiting for other players, when all the players have joined the game the server will send the client a MatchCreated message.

2.3 Player turn

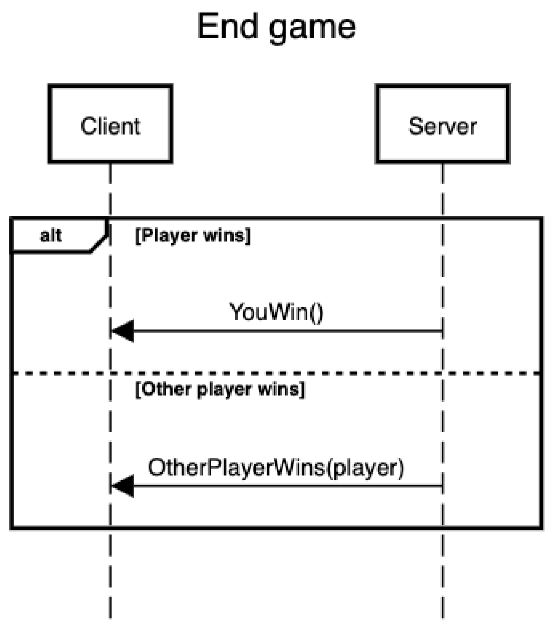




In this scenario we are in the middle of the game and the server will ask the client to choose an assistant card via the SelectAssistantCard message. The client will choose the card to play and will send a message like PlayAssistanCard with the chosen card Once the game turn has been established, the client will only be able to send messages during his turn, so if he tries to send any message during another client's turn, the server will respond with a

NotYourTurn error message. During the whole action phase of his turn the player can play the character card therefore in this scenario the possibility of sending a message of the type PlayCharacterCard specifying the card to play has been inserted before any possible action. The server responds to this request with an Action Valid message, confirming the client's action, or with an ActionNonValid message, invalidating the action Each player in their turn can decide to move a certain number of students to the island or the game board, so a loop has been made to perform the action for each student. The client requests the server to move the student to the island with the MoveStudent (move, color, position) message or to the game board with the MoveStudent (move, color). The server will respond to these requests with a confirmation, therefore an ActionValid message or in case of problems with an ActionNonValid message, invalidating the action. Once all the students have been placed, it is time to move mother nature, the client requests via the message MoveMotherNatureStep (steps). Also in this case the server will be able to validate or invalidate the action. At the end of the round, the server sends the game status update message to the client.

2.4 Endgame



In this scenario the server communicates the end of the game to the client, based on the outcome of the match the messages that the server will send to the client can be YouWin or OtherPlayerWins (player) with the nickname of the winner.