Eriantys Protocol Documentation

Riccardo Inghilleri, Daniela Maftei, Manuela Merlo
Gruppo 35

To implement the communication protocol we decided to use Messages, objects that implement the serializable interface, and the stream of the latter through the established connections between clients and server.

1 Main Classes

To manage the communication we have provided the following classes:

1.1 Server side

- **SERVER:** class whose aims are to:
  1. Accept connections from client;
  2. Enter clients in different queues depending on the game mode chosen
     - Queue #1: 2 players normal mode
     - Queue #2: 2 players expert mode
     - Queue #3: 3 players normal mode
     - Queue #4: 3 players expert mode
  3. Create Games once reached enough clients;
     NB: We decided to implement the server so that it can manage more games **simultaneously**.
  4. Remove a game from the list of active Games once it is finished.
- **GAMEHANDLER:** class that rapresents the actual Game. Its tasks are:
  1. Notify all clients that the game has started;
  2. Manage the setting of the game parameters (nickname, color and magician) for each player, according to their order connection to the server;
  3. Fill the clouds automatically and ask to each player the Assistant card during the Round Planning phase;
  4. Send the correct messages to the View via VirtualView and manage the client responses through the Controller during the Round Action phase;
  5. Manage the end of the Game, notifying all players the winner and the closure of the connection.
- **VIRTUAL VIEW:** class that communicates with the ClientConnecton through the socket created by the server. The VirtualView forwards messages based on the value of an *inGame* attribute:
  - **False:** the client has not been assigned to a Game yet. The only message it receives in this phase contains the settings (player numbers and game mode) to be communicated to the **Server** so that the latter can insert the client in the correct waiting queue.
  - **True:** the client is in a Game, the messages are forwarded to the **GameHandler**.

1.2 Client side

- **VIEW:** Cli and Gui are an implemetation of this interface with which the client interacts.
- **CLIENTCONNECTION:** Client-side connection that forwards through the message network from the View.

2 Message Management

Messages are handled differently depending on whether they are received on the client side or on the server side:

- **Client:** Messages implement the forward(View view) method of a *ServerMessage* interface. The function of the latter is to call the correct method of the View interface.
- **Server:** Messages from the client contain an *Action* attribute that corresponds to a phase of the controller and which tells the GameHandler which method to call.

3 The Creation of a Game

The flow for creating a Game is the following:

1. Each client connects to the server;
2. Each client is asked to choose the game settings: number of players and game mode;
3. The view checks that the parameters are correct before sending the message through the network;
4. The VirtualView notifies the choice to the server;
5. The server inserts the VirtualView into the correct queue;
6. For each addition, the Server checks whether the number of players is sufficient for the creation of a Game.
   If so the Server creates a GameHandler to which he passes the VirtualView by removing them from the waiting queue. At this point the *inGame* parameter is set to true.

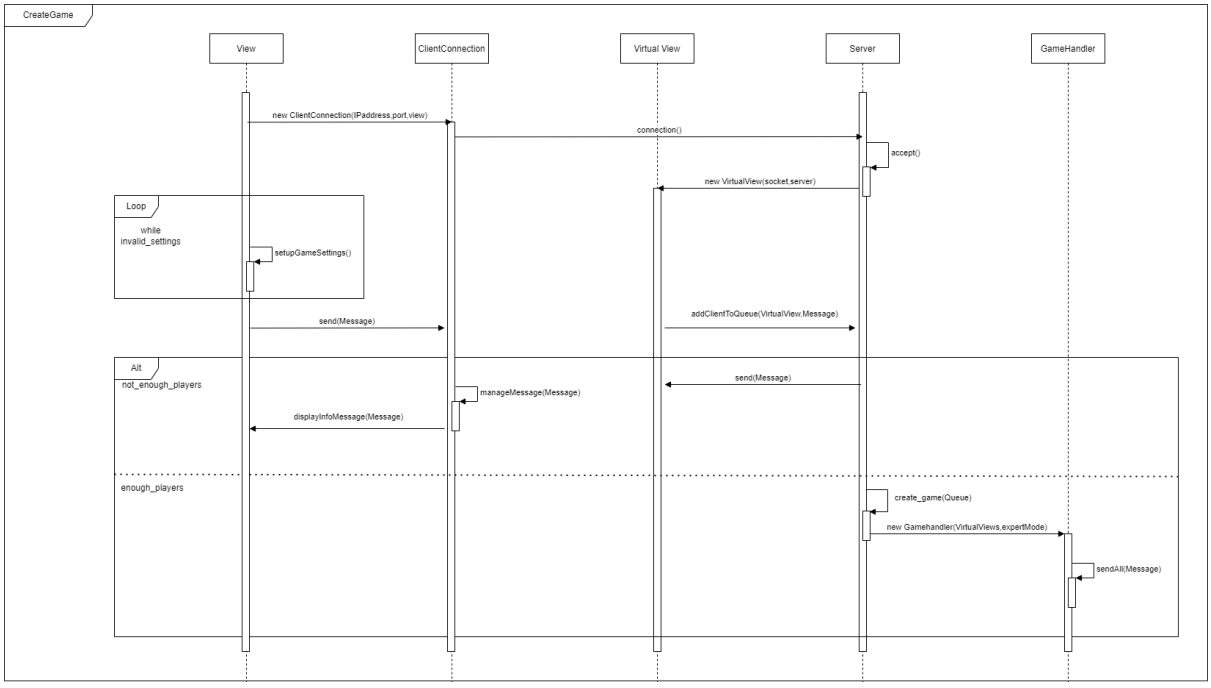

Figure 1: CreateGame

Messages from the Client:

1. SettingMessage : contains the game's settings: playersNumber and mode.

Messages from the Server:

1. InfoMessage: contains the message to be displayed by the view. In this case, the beginning of the Game.

4 InGame Phase: The Server creates the Game

The server has a list of Gamehandlers that each manages a different game. At the beginninng of the Game, each client is notified. The constructor of the GameHandler needs among the parameters:

1. The VirtualView of clients
2. Game mode

NB:

1. Unsolicited messages are ignored
2. The validity of the data is checked:
   - By the View: if the choice concerns the state of the game.
     `Example:` choosing a color of a student from the Entrance which is not actually an option.
   - By the GamHandler: if the data is linked to other players' choices.
     `Example:` the choice of an AssistantCard
3. When a request is sent, a timer is startedand it is stopped when receiving an answer. If the response is delayed, the client is notified and disconnected.
4. The connection is always tested by a pinger. Ping messages are ignored.

5 GameHandler SetUp Phase

The GameHandler has several phases:

SETUP_NICKNAME: The GameHandler manages the players nicknames choice according to the order they connected. If a nickname has already been chosen by another player, the GameHandler asks the client to chose again.

SETUP_COLOR: The Gamehandler manages the players color choice according to the order they connected by providing them a message with the colors still available.

- IF 2 PLAYERS: The color is chosen only by the first player.
- IF 3 PLAYERS: The color is chosen by the first and second player, while the third is notified of the automatic assignment by the GameHandler.

SETUP_WIZARD: The GameHandler manages the players magician choice according to the order they connected by providing them a message with the magicians still available.

PLANNING: The GameHandler manages cloud filling and the selection of Assistant Cards.

ACTION: The GameHandler manages the actual game round.

NB: The choice of the nickname, color and wizard is atomic for each client, so they are asked all together.
NB: At the end of the wizards setup, the GameHandler creates the Board to storage all the data.

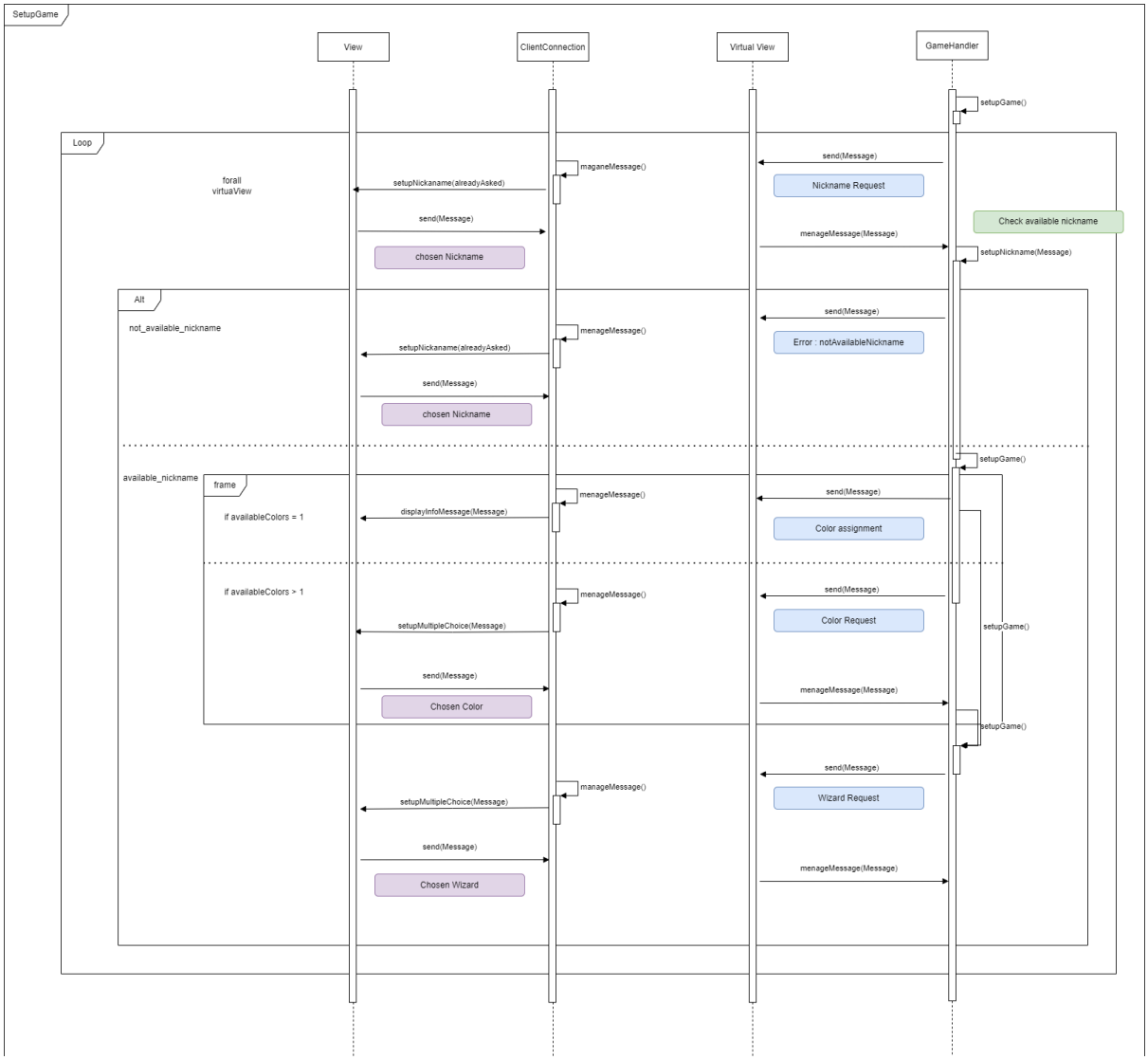

Figure 2: SetUpGame

Messages from the Client:

1. SetupMessage: contains a string-type attribute to which the is assigned
   - The nickname in the first exchange;
   - The color in the second exchange;
   - The magician in the third exchange.

Messages from the Server:

1. InfoMessage: contains the message to be displayed by the view. In this case, the unavailability of the nickname or the assigned color.
2. NicknameMessage: contains a boolean: `alreadyAsked`. Based on this parameter, the view changes the request to the player.
3. MultipleChoiceMessage: contains a List of parameters with the available colors or magicians.

6 InGame Phase

6.1 The rounds

Each round is divided in 2 phases: Planning and Action.

6.1.1 Planning

1. Filling of all the clouds.
2. Every player chooses an assistant card to define the play order.

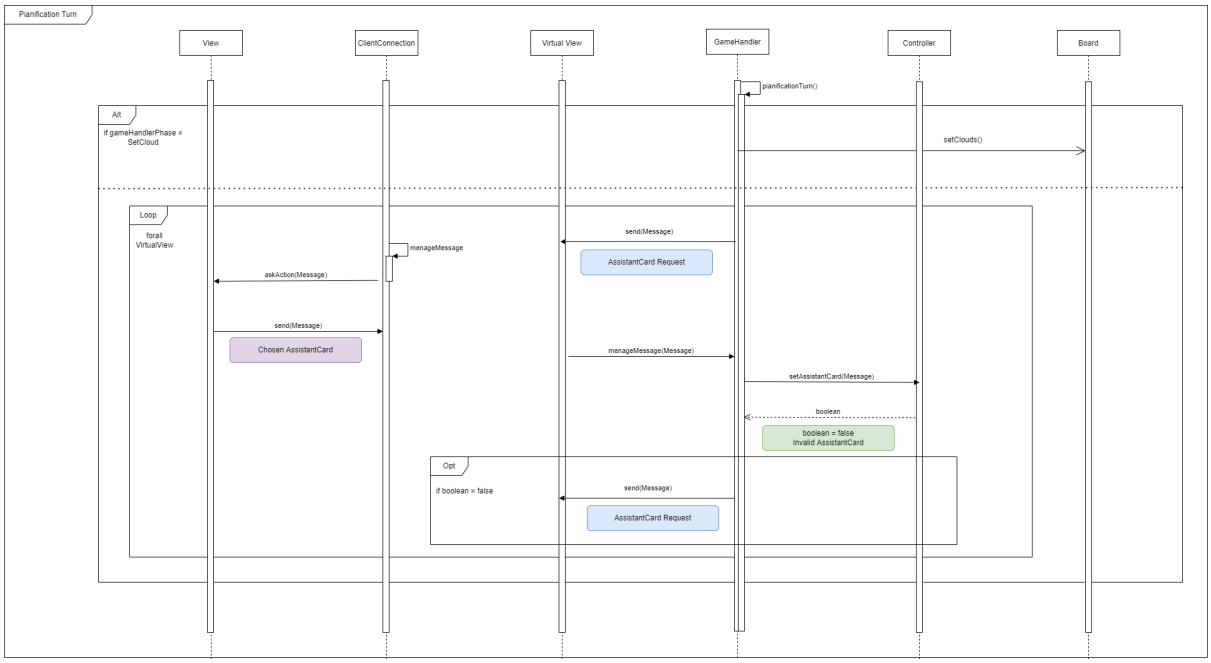

Figure 3: Planning Phase

Messages from the Client:

1. ActionMessage: contains the priority of the chosen AssistantCard.

Messages from the Server:

1. InfoMessage: contains the message to be displayed by the view. In this case, the unavailability of the AssistantCard.
2. AskActionMessage: contains
   - the action-type : for example, `CHOOSE_ASSISTANT_CARD`;
   - the player's available AssistantCards.

6.1.2 Action

It depends on the game mode.

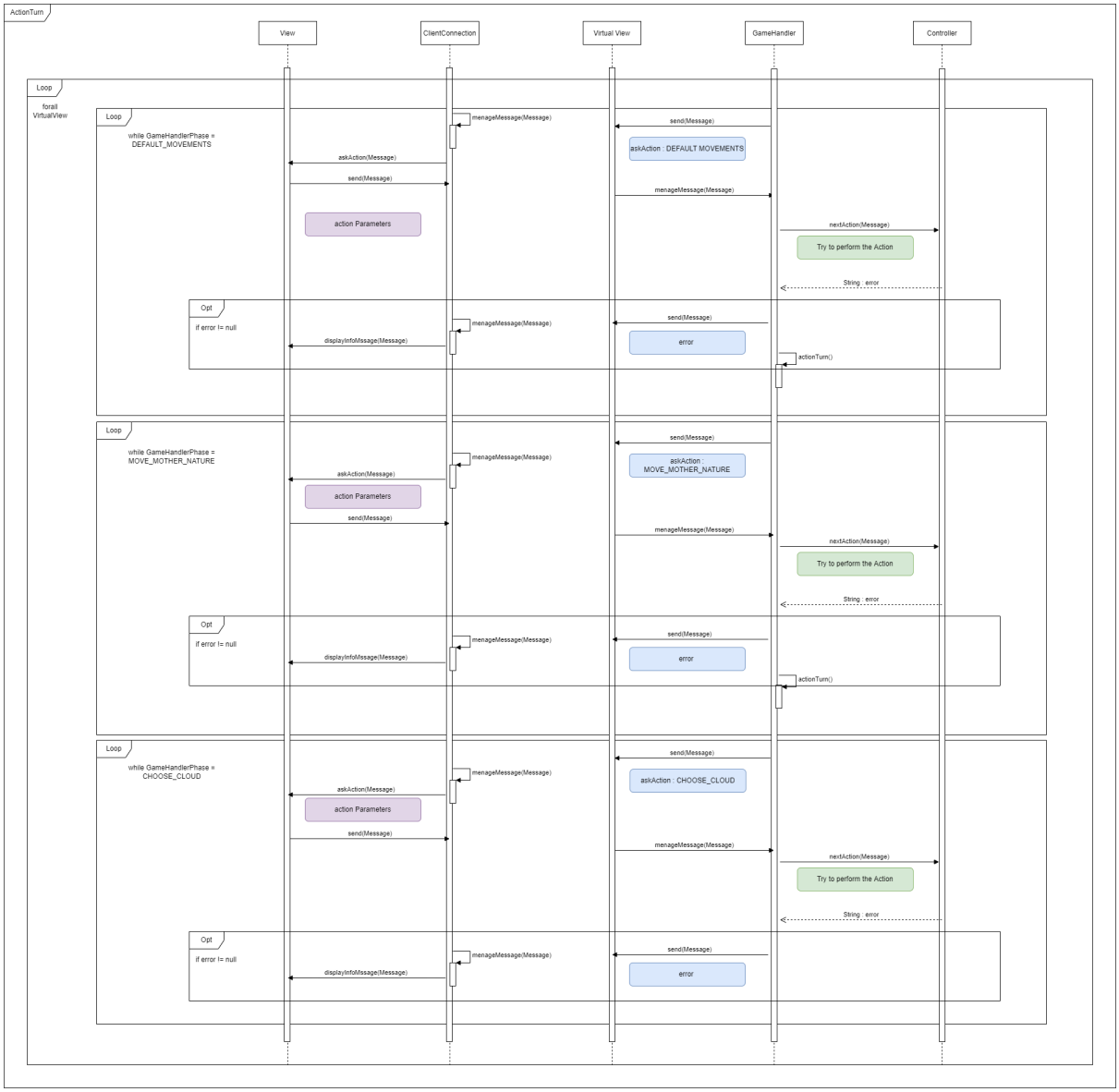

Figure 4: Action Phase

Messages from the Client:

1. ActionMessage: contains the parameters for the action
    - the action-type : for example, `MOVE_MOTHER_NATURE`;
    - the chosen mother nature steps;
    - the color of the students;
    - the island where to move the student from the Entrance;
    - the cloud with which to fill the Entrance;

Messages from the Server:

1. AskActionMessage: contains the parameters to make a request
    - the action-type;

- the available choices: for example, the available students in the Entrance or the remaining islands.

NORMAL MODE: the available actions are:

1. DEFAULT_MOVEMENTS: It allows the movement of a student from the Entrance to the DiningRoom or to an island
2. MOVE_MOTHER_NATURE: It allows to move MotherNature to another island.
3. CHOOSE_CLOUD: It allows to choose the cloud from which to take students to be added to the Entrance.

In this game mode, the player is guided in the choice of actions by the GameHandler since the order is default.

EXPERTMODE: In addition to the predefined actions, there are also: CHOOSE_CHARACTER_CARD and USE_CHARACTER_CARD. The player is partially guided in the choice of actions: before each default action the client is asked if he wants to use a character card in case he hasn't already done it.

NB: Any default action is atomic, therefore a player can choose to use the Character Card only before or afterhavng moved all the students from the Entrance.

7 The Winner

Both the GameHandler and the Controller have to check if a termination condition has occured:

1. 10 rounds have been played;
2. There are no students left on the Board;
3. A player has finished all his towers;
4. There are only 3 remaining islands.

When the `endGame()` method is called, the gamehandler accesses the winner, defined by the board, and:

1. Sends a message to all players, announcing the winner;
2. Notifies each player of the disconnection;
3. Notifies the Server which removes it from the list of active Games.

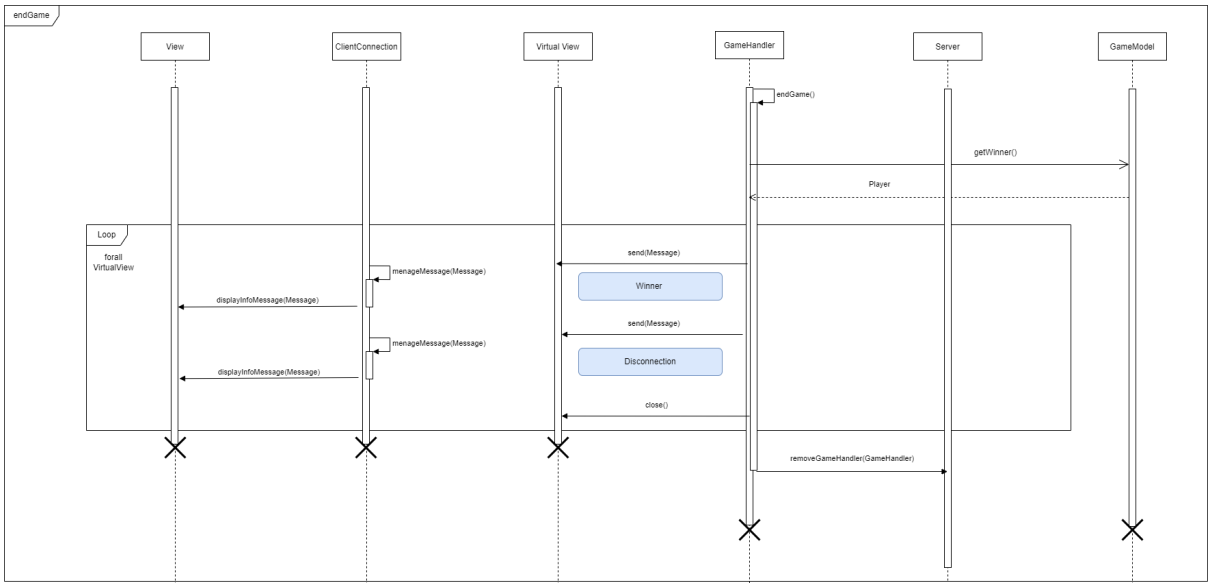

Figure 5: EndGame

Messages from the Server:

1. InfoMessage: contains the message to be displayed by the view. In this case, the winner and the disconnection message.

8 Client Disconnection

In the following cases, a player must be disconnected:

1. The timer ended before the player has answered;
2. The game ended because a termination condition occured.

In the first case the end of the game is forced and the other players are notified and finally all the connections are closed.Whereas in the second case all players are notified and all connections are closed. The gameHandler, in order to disconnect a client, calls the *closeConnection()* method from the Virtual Viwe, which in the following order:

1. Stops the timer if it is still active;
2. Stops the thread that that is responsible for testing the connection;
3. closes the InputStream, OutputStream and the clientSocket.