Eryantis Protocol Documentation

Moretti Lorenzo, Mormile Matteo, Orofino Antonino
Group 6

May 3, 2022

Contents

1 Messages

The messages are classified depending on the recipients and the sender. The possible kinds are the following:

- `CVMessage`: sent by the controller to the view;
- `CVMessage`: sent by the view to the controller;
- `SYSMessage`: generic information message sent by controller to the client view.

1.1 Ack

This message is sent from the server to the client when a generic message has been acknowledged. It is a `SYSMessage`

Arguments

This message has no arguments.

Possible responses

This message has no responses.

1.2 AskGameSetting

This message belongs to `CVMessage` category. It is only sent to the first player in order to ask information about the game configuration such as the number of players and the game mode.

Arguments

This message has no arguments.

Possible responses

- `SetGameSettings`: message containing the game settings.

1.3 SetGameSettings

This message, which belongs to the `VCMessages` class, is sent by the first player to provide the game configuration.

Arguments

1. `playerID`: the ID of the message's sender;
2. `numPlayers`: the number of players (2..3);
3. `gameMode`: the game mode (Basic, Expert).

Possible responses

This message has no responses.

1.4 AskNickname

This message is sent from the server to the client (`CVMessage`) to ask the player to provide a nickname.

Arguments

1. `playerID`: the player ID assigned by the server to the client itself.

Possible responses

- `SetPlayerNickname`: message sent to set the player nickname.

1.5 SetNickname

`VCMessage` sent to the server with the chosen nickname.

Arguments

1. `playerID`: the ID of the message's sender;
2. `nickname`: the nickname chosen by the player.

Possible responses

- `Ack`: condition in which the nickname provide is available and it was assigned to the player;
- `AskNickname`: the message has been received but the nickname was unavailable and another choice should be sent.

1.6 AskTowerColor

This message is sent from the server to the client (`CVMessage`) to ask the player to choose the color of the towers.

Arguments

1. `possibleColors`: the available colors.

Possible responses

- `SetTowerColor`: message sent to set the color.

1.7 SetTowerColor

`VCMessage` sent to the server with the chosen color.

Arguments

1. `playerID`: the ID of the message's sender;
2. `color`: the color chosen by the player.

Possible responses

- `Ack`: condition in which the color provide is available and it was assigned to the player;
- `AskTowerColor`: the message has been received but the color was unavailable and another choice should be sent.

1.8 GameStart

This message is a `CVMessage` and it is sent to every client to inform that the game is ready to start after two or three players have been matched. Moreover it specifies the first player chosen randomly.

Arguments

1. `mapInfo`: the current state of the game board;
2. `firstPlayerNickname`: the nickname of the first player.

Possible responses

This message has no responses.

1.9 AskAssistantCard

This message is sent from the server to the client (`CVMessage`) to ask the player to choose an assistant card.

Arguments

1. `possibleCards`: list of possible cards that can be chosen by the player.

Possible responses

- `SetAssistantCard`: message sent to set the chosen card.

1.10 SetAssistantCard

`VCMessage` sent to the server to set the assistant card.

Arguments

1. `playerID`: the ID of the message's sender;
2. `assistantCard`: the card chosen by the player.

Possible responses

This message has no responses.

1.11 UpdateGameBoard

This message is sent from the server (`CVMessage`) to the client when the state of the game's board has been updated.

Arguments

1. `mapInfo`: current state of the game board.

Possible responses

This message has no responses.

1.12 UpdateCurrentPlayer

This message is sent from the server (`CVMessage`) to the client when the current player of the game has been updated.

Arguments

1. `playerNickname`: current player of the game.

Possible responses

This message has no responses.

1.13 AskAction

`CVMessage` sent to the current player to ask to perform an action.

Arguments

1. `possibleActions`: list of possible cards that can be chosen by the player.

Possible responses

- `SetAction`: the message has been received and an action is sent.

1.14 SetAction

This message is sent from the client to the server (`VCMessage`) when an action is performed by the player.

Arguments

1. `playerID`: the ID of the message's sender;
2. `action`: move of the player.

Possible responses

- `Ack`: the message has been received and the action was performed;
- `AskAction`: he message has been received but the action was impossible and another choice should be sent.

1.15 ShowLastRound

This message is sent from the server to each client (`CVMessage`) when either a player ends his cards or the students' bag is empty. The game will finish at the end of the current round.

Arguments

This message has no arguments.

Possible responses

This message has no responses.

1.16 ShowEndGame

`CVMessage` sent to each client to report the end of the game and the nickname of the winner.

Arguments

1. `winnerNickname`: nickname of the winner.

Possible responses

This message has no responses.

1.17 ShowDisconnection

This message is sent from the server (`SYSMessage`) to every client, but the disconnected one, to report that the game must end due to the disconnection of a player.

Arguments

1. `disconnectedNickname`: nickname of the player who disconnected from the match.

Possible responses

This message has no responses.

1.18 Heartbeat

This message is a `SYSMessage` and it is sent to keep alive the connection between two sockets.

Arguments

This message has no arguments.

Possible responses

This message has no responses.

2 Scenarios

2.1 Game setup

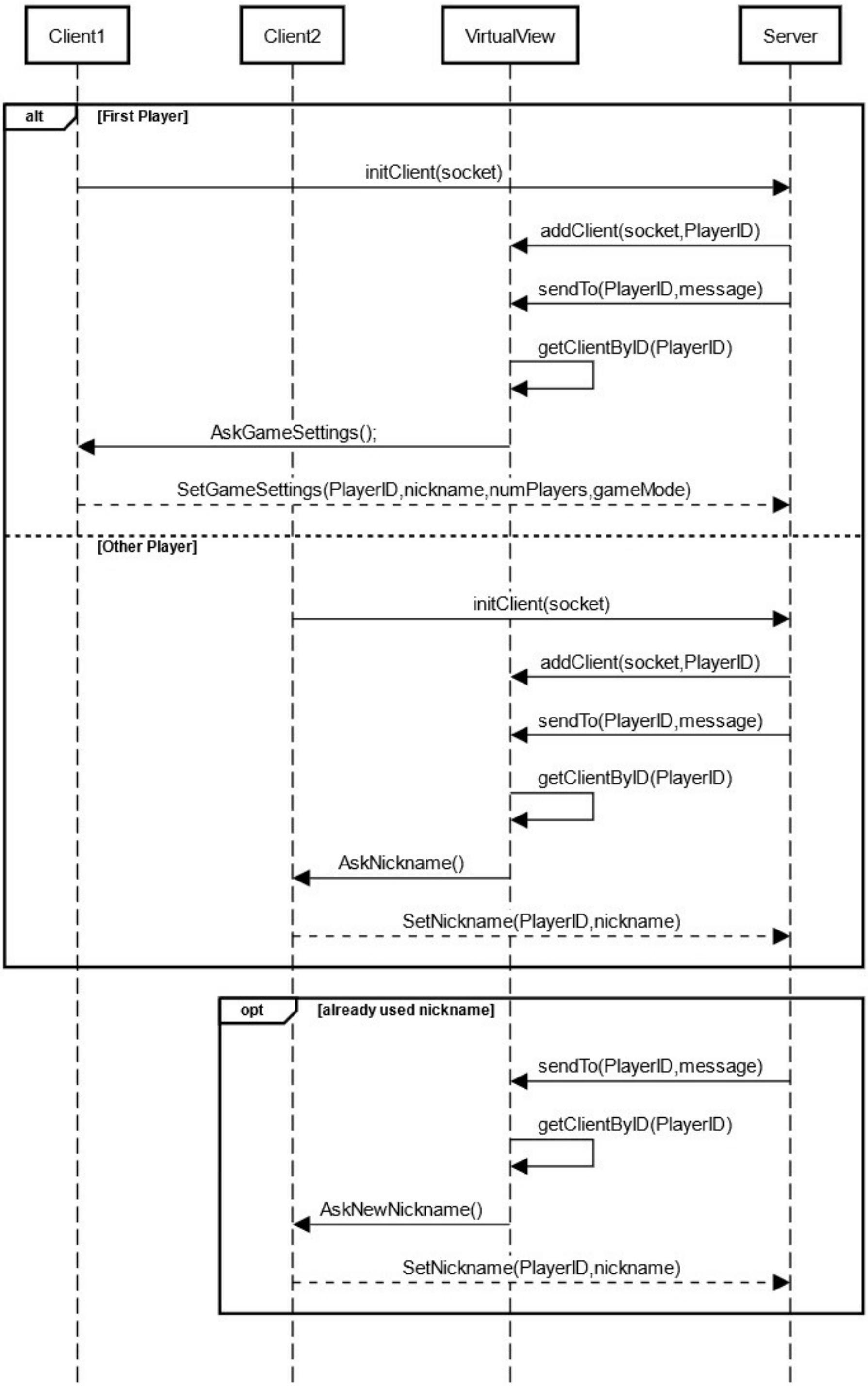

Figure 1: Game setup diagram

2.2 Planning phase

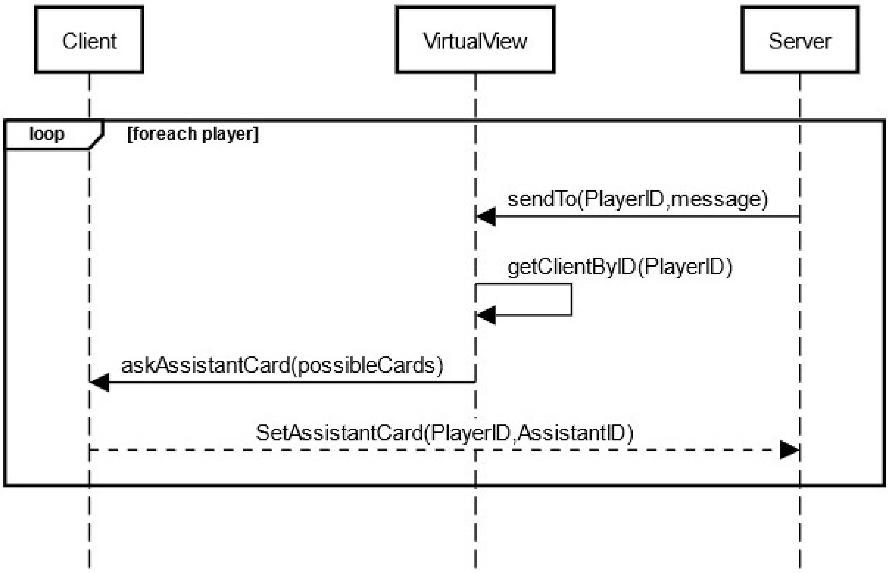

Figure 2: Planning phase diagram

2.3 Action phase

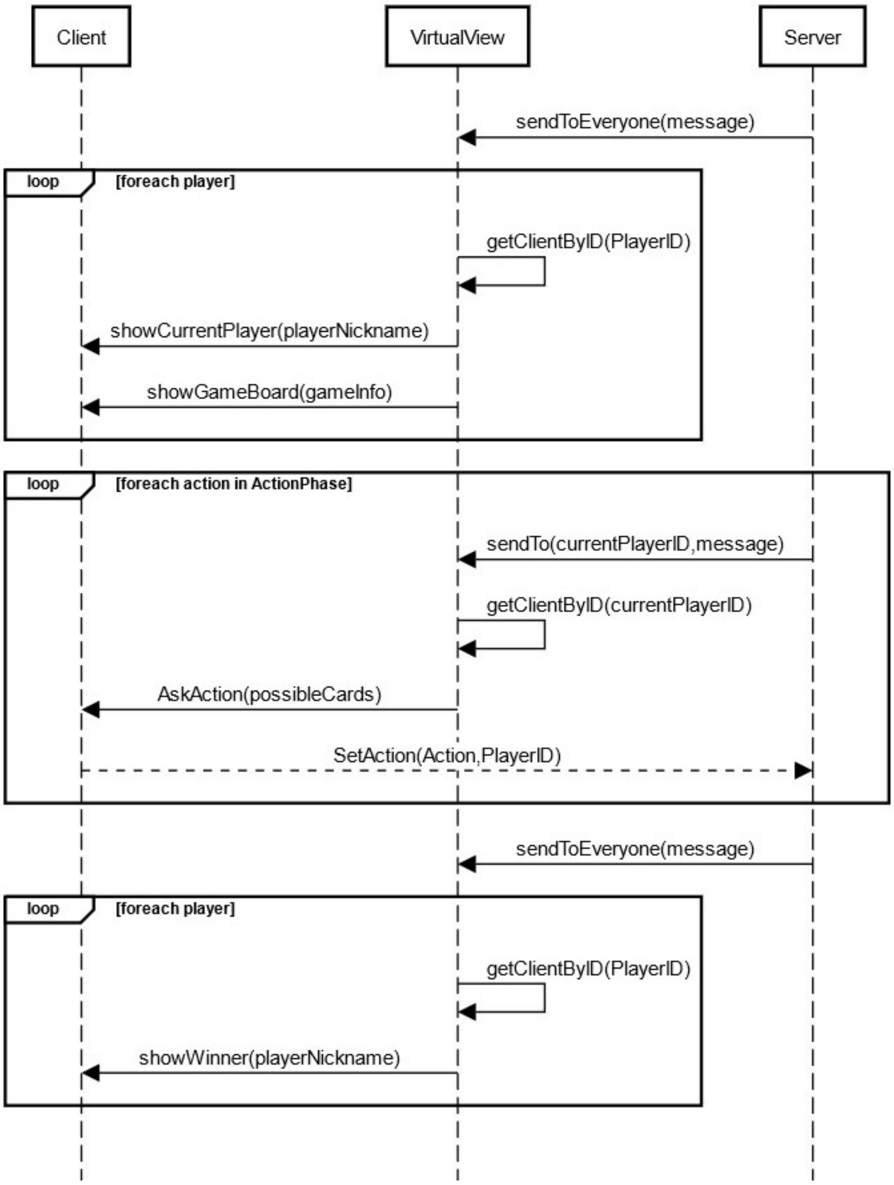

Figure 3: Action phase diagram