Eryantis Communication Protocol Documentation

Camilla Gobbi, Alessandro Meroni, Marco Orsi

Gruppo 30

Messages

1. ACK

Sent as a response to a generic message or notification

Arguments

- ACK: no argument

Possible Responses

- Response: no response

2. NACK

Sent in response to a message if client and server are not aligned

Arguments

- NACK: no argument

Possible Responses

- Creation

3. Login

Sent from the client to the server to log in the game

Arguments

- Login: username of the player

Possible Responses

- LoginSucceeded: when the login has happened successfully
- LoginFailed: then the login failed

4. ChoosingGame

The client decides if it wants to start a new game (as first player), or join a game (as second, third of fourth player) or resume a paused (stored in the server memory) game

Arguments

- ChoosingGame: the player's choice between new game, join game, resume game

Possible Responses

- ACK: if selected new game, and selected the type of came (pro or base)
- NoGames: if selected "join game" there is no games started from another player or selected "resume game"
- ListOfGames: if selected "join game" and there are some matches waiting for some players to join
- ListOfStartedGames: if selected "resume game" and exists some games started with this usergame

5. GameSelected

The player chooses the game from the list

Arguments

- GameSelected: string

Possible Responses

- Creation

6. NumPlayers

Sent from client to the server to know the number of players of the game

Arguments

- NumPlayers: integer representing the number of the players

Possible Responses

- ACK
- NACK

7. Wizard

Sent from the server to make the player choose which wizard wants to play with

Arguments

- Wizard: available wizards

Possible Responses

- Choice: chosen wizard

8. Creation

The server sends the Board, Islands, Mother Nature, and the Clouds to the client

Arguments

- Creation: one board for every player, all the isles and clouds, all of them already initialized

Possible Responses

- ACK
- NACK

9. RefillClouds

The server sends the necessary students to refill the clouds

Arguments

- RefillClouds: list of students randomly extracted from the bag

Possible Responses

- ACK
- NACK

10. ChooseCard

The server asks to select an assistant card and the client sends the chosen card

Arguments

- ChooseCard: list of cards which haven’t been chosen yet

Possible Responses

- ChosenCard: the card chosen by the player

11. MoveStudents

The server requests a player to move the students from the entrance of his board

Arguments

- MoveStudents: string

Possible Responses

- Student1: Student and a string that contains if it is going to an island or in the dining room, if it's island, it also sends the id of the island where the student is sent
- Student2: same as Student1
- Student3: same as Student1

12. StepsMN

The client sends the number of steps that mother nature has to do

Arguments

- StepsMN: int

Possible Responses

- ACK

13. ChooseCloud

The server asks to select a cloud

Arguments

- ChooseCloud: a list of the clouds which haven't been chosen yet

Possible Responses

- Choice: the chosen cloud

14. NotifyChosenCard

The server sends which card is chosen by the other players

Arguments

- NotifyChosenCard: chosen card and his player

Possible Responses

- ACK
- NACK

15. NotifyMoveStudents

When a player moves some students from their entrance, the server notifies all the other players

Arguments

- Student1: Student and a string that contains if it is going to an island or in the dining room, if it's island, it also sends the id of the island where the student is sent
- Student2: same as Student1
- Student3: same as Student1

Possible Responses

- ACK
- NACK

16. NotifyMovementMN

The server sends the necessary stuff to notify all the changes derived by the movement of Mother Nature

Arguments

- NotifyMovementMN: new position of Mother Nature, list with all the updated lands

Possible Responses

- ACK
- NACK

17. NotifyProfessors

The server sends which professors must get in each board (only if there is a change)

Arguments

- NotifyProfessors: professors and players

Possible Responses

- ACK
- NACK

18. NotifyChosenCloud

The server sends the waiting players the choose (cloud) of the current player

Arguments

- NotifyChosenCloud: cloud chosen and his player

Possible Responses

- ACK
- NACK

19. NotifyTowers

The server sends the new position of the towers (only if there is a change)

Arguments

- NotifyTowes: towers and island or board

Possible Responses

- ACK
- NACK

20. EndGame

The server notifies the end of the game and the winner

Arguments

- EndGame: winning player and a string that explains why it ended and a panoramic of the ending set of the game

Possible Responses

- ACK
- NACK

21. LastTower

The server notifies the clients when a player builds his last tower

Arguments

- LastTower: the player who built their last tower

Possible Responses

- ACK
- NACK

22. NoMoreStudents

The server notifies the clients when the bag has no more students inside

Arguments

- NoMoreStudents: string

Possible Responses

- ACK

23. ChChosen

The client sends the chosen character card to the server

Arguments

- ChChosen: chosen character card and player

Possible Responses

- Changes: the changes happened in the model that modifies the view

24. Ping

The server regularly sends each player this message to check if the connection is still working

Arguments

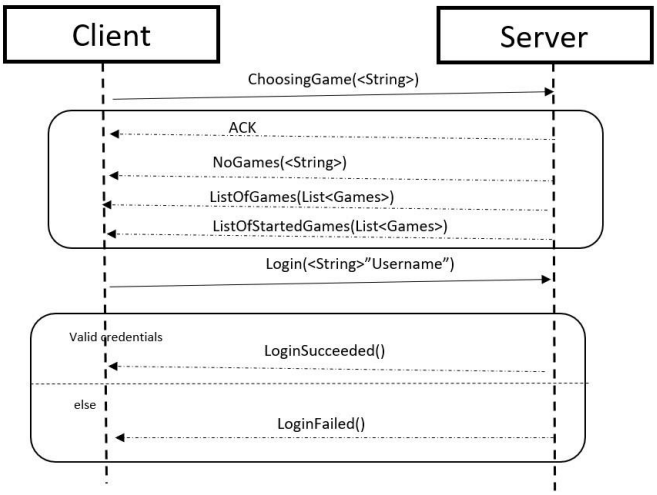
- Ping: no argument

Possible Responses

- ACK

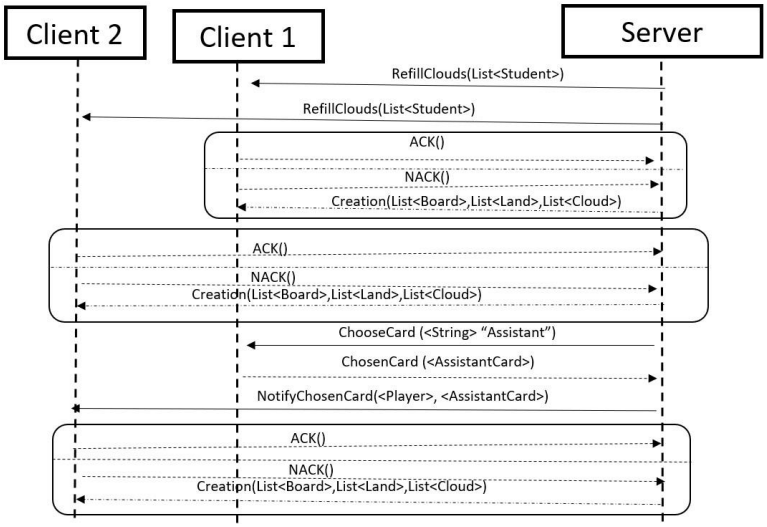
Scenarios

1. Login



The server waits to a ChoosingGame message from the client, then he send the answer between ACK, NoGames,ListOfGames and ListOfStartedGames. After that the server waits the client to send Login message. Then if the Username is correct the server sends a LoginSuccessed message, otherwise sends a LoginFailed message and the method is repeated another time until the Username received by the server is correct and the server sends a LoginSuccessed message

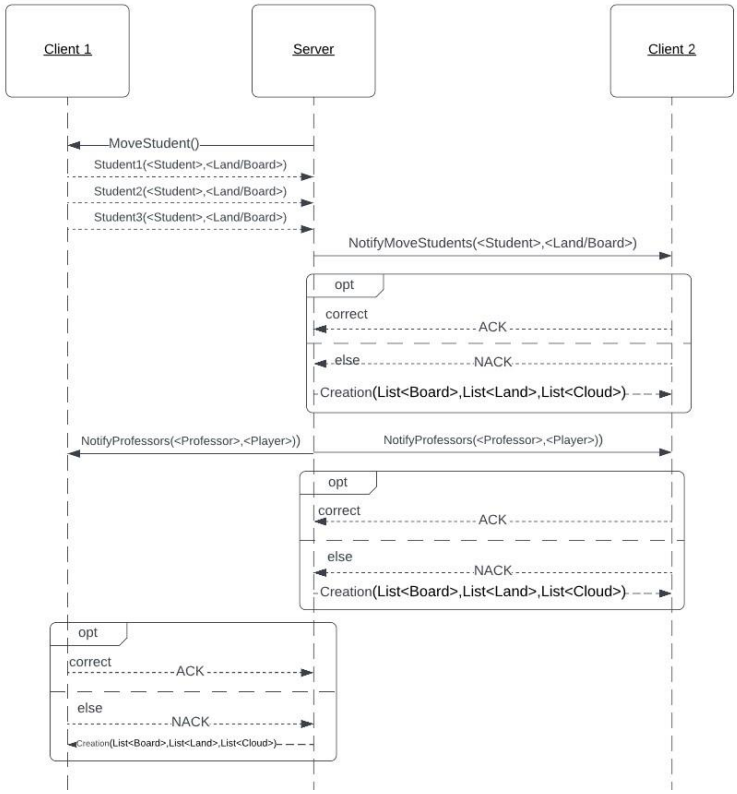
2. Planning



The server sends to all the clients the message RefillClouds with the Students to put into the Clouds, if the message is correctly received, they send an ACK message. If the message is not correct the server sends a NACK message and a Creation massage in order to clarify the situation. After that the server sends to the client of the current turn a ChooseCard message and he responds with a ChosenCard message with an AssistantCard. If the server receives this message correctly it notifies to all the other clients the card chosen during the planning method with a

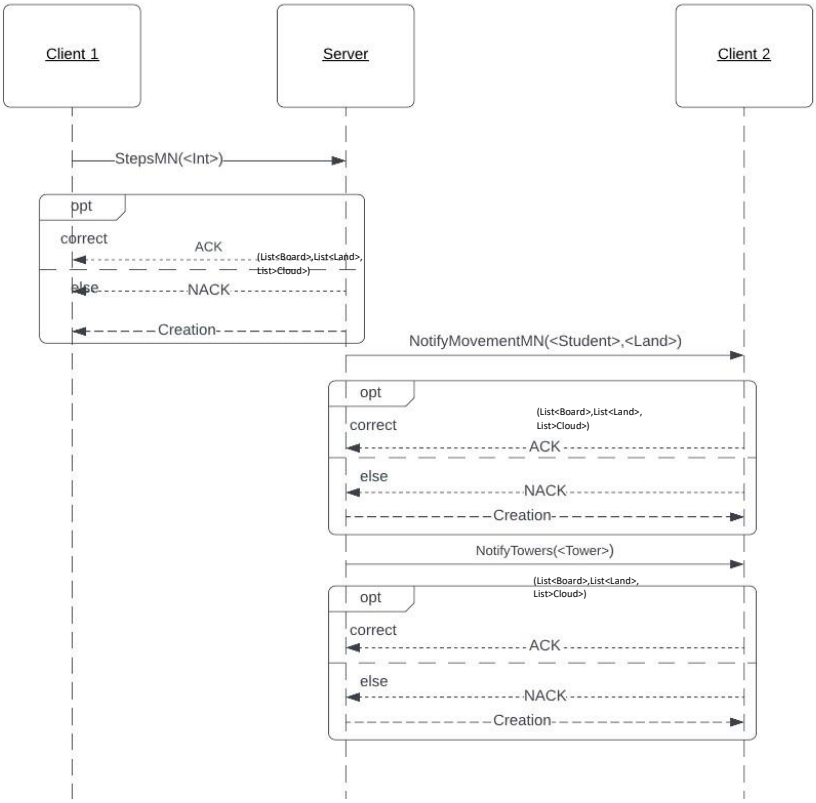
NotifyChoosenCard message. When the clients receive this message respond with an ACK message.

3. Action 1



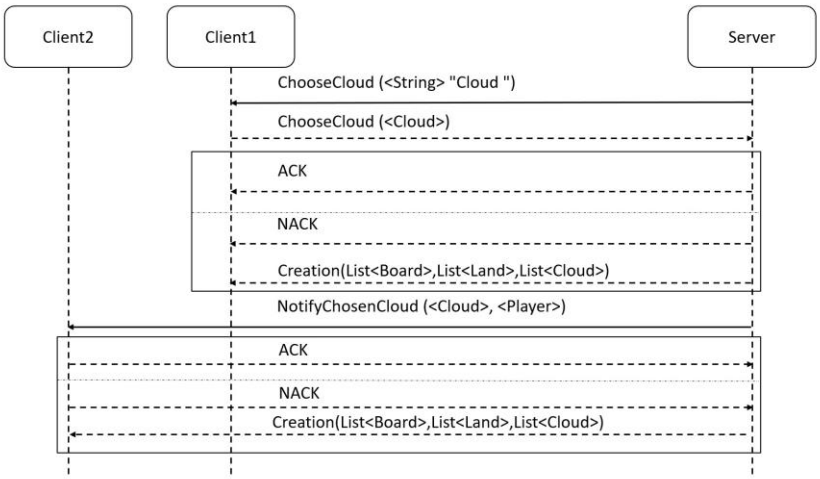
The server sends the client of the current turn (Client 1) the message MoveStudent to let him the permission of move three students, and then the Client 1 responds with the messages Student1, Student2, Student3 to communicate the choice of the player. After that the server notifies to all the other clients the choices of Client 1 with the message NotifyMoveStudents, and if the message is correctly received they respond with an ACK message. If the message is not correct the client sends a NACK message and the server sends a Creation massage in order to clarify the situation. At the end the server notifies to all the clients the new position of the professors in the game with the message NotifyProfessors, and if the message is correctly received they respond with an ACK message. If the message is not correct the clients send a NACK message and the server sends a Creation massage in order to clarify the situation.

3. Action 2



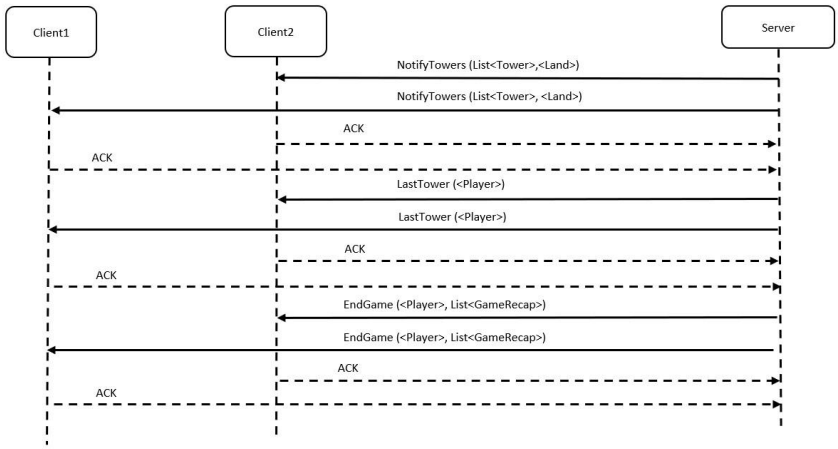
The client that is playing in the current turn notifies to the server his choice about the movement of Mother Nature with the message StepsMN and the server responds with an ACK message. After that the server notifies to all the other client the choice of the movement of Mother Nature with the message NotifyMovementMN, and if the message is correctly received they respond with an ACK message. If the message is not correct the server sends a NACK message and the server sends a Creation massage in order to clarify the situation.
At the end the server notifies to the client that is not playing during this turn and owns the towers the position of his towers (in the case there are effectively some changes), and if the message is correctly received it responds with an ACK message. If the message is not correct the client sends a NACK message and the server sends a Creation massage in order to clarify the situation.

3. Action 3



First the server sends the message ChooseCloud plus the list of the available clouds to the right player (the one currently playing) then the remote controller answers with a cloud and the server notifies the correct reception with an ACK. If the message is not correct the server sends a NACK message and the server sends a Creation massage in order to clarify the situation. In the end the server sends all the other players the message NotifyChosenCloud with the cloud and the player who just chose it and then is repeated the module ACK/NACK/Creation.

4. End



When a player builds their last tower, the server sends all the other players the message NotifyTowers to refresh their View and the message LastTower with the player who built it (the clients answer the notifications with an ACK). In the end the server sends EndGame which is a notification containing the username of the winner and a list of GameRecap and the remote controllers answer with an ACK (even here is the ACK/NACK/Creation module but we didn't put it in the scheme to simplify the visualization). GameRecap is a class containing a player username, the number of built towers and which professors they control when the game ends. Thus, for each player List<GameRecap> contains a recap of their status when the match ends.