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: String

Possible Responses

- Response: no response

2. NACK

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

Arguments

- NACK: String

Possible Responses

- The previous message
- 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. ListOfGames

The server sends the client all the matches which are not full yet and the stored ones which contain the player's username

Possible arguments

- JionGames: the list of matches which the player could join in
- ResumeGames: the stored matches which contain the player's username

Possible answers

- ChoosingGame

5. ChoosingGame

The client decides if they want 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

Possible Arguments

- NewGame: the player wants to start a new game, it contains a string, the number of players and a Boolean variable specifying if it is a pro or abase match
- JoinGame: the player wants to join an opened game, it contains a string with the name of the creator
- ResumeGame: the player wants to resume a game previously started and not already finished maybe for a server failure, is contains a string

Possible Responses

- ACK
- NACK

6. 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

7. 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

8. 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

9. 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

10. 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
- (Student 4: same as the other students but sent only if there are three players in the match)

11. StepsMN

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

Arguments

- StepsMN: int

Possible Responses

- ACK

12. ChooseCloud

The server asks to select a cloud

Arguments

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

Possible Responses

- ChoiceCloud: the chosen cloud

13. ChoiceCloud

The client sends che chosen cloud to the server

Arguments

- ChosenCloud: the player’s choice

Possible Responses

- ACK
- NACK

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 (id)/(board)

When a player moves some students from their entrance, the server notifies all the other players.
(id) -> when a player places a student on an island
(board) -> when a player places a student on their board

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
- (Student4: same as the other students but sent only when there are three players in the match)
- Id: the id of the island where the student has been placed on

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 (land)/(board)

The server sends the new position of the towers (only if there is a change).
(land) -> the tower(s) built on an island (group) after the last Mother Nature's movement
(board) -> the tower(s) returned to a board when an island (group) is conquered

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
- No responses

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

Arguments
- NoMoreStudents: string

Possible Responses
- No responses

23. NotifyThreeArchipelagos
The server notifies the client when there are only three groups of islands in the sky

Arguments
- NotifyThreeArchipelagos: sting

24. Ch
The server sends the client the request to choose a character card

Arguments
- Ch: string

Possible responses
- ChChosen: the player’s choice
- No_Ch: if the player hasn’t played any character card

25. ChChosen
The client sends the chosen character card to the server

Possible Arguments
- Ch_1: if the player chose the first character card. It contains a student and the island where it has been placed
- Ch_2: if the player chose the second character card.
- Ch_4: if the player chose the fourth character card.

- Ch_5: if the player chose the fifth character card. It contains the land where the no-entry tile has been placed on
- Ch_8: if the player chose the eighth character card.
- Ch_10: if the player chose the tenth character card. It contains the list of changed students and the tables (list of Type_Student) the students have been changed with
- Ch_11: if the player chose the eleventh character card. It contains the student which has been placed in the player's entrance
- Ch_12: if the player chose the twelfth character card. It contains the type (Type_Student) which the players have to return three students to the bag from their dining room

Possible Responses

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

26. NotifyCh

The server sends the client the changes happened after a player draw a character card

Possible Arguments

- NotifyCh_1: notifies that a player has drawn the first character card. It contains the island where the student has been placed on, the updated list of students on the card, the moved student and the player who drew the card
- NotifyCh_2: notifies that a player has drawn the second character card. It contains the updated professors of the match and the player who drew the card
- NotifyCh_4: notifies that a player has drawn the fourth character card. It contains the player who drew the card
- NotifyCh_5: notifies that a player has drawn the fifth character card. It contains the island where the no-entry tile has been placed on and the player who drew the card
- NotifyCh_8: notifies that a player has drawn the eighth character card. It contains the player who drew the card
- NotifyCh_10: notifies that a player has drawn the tenth character card. It contains the player who drew the card, the list of changed students and the tables
- NotifyCh_11: notifies that a player has drawn the eleventh character card. It contains the updated list of the students on the card, the player who drew the card and the moved student
- NotifyCh_12: notifies that a player has drawn the twelfth character card. It contains the type of students to return to the bag and the player who drew the card

Possible Responses

- ACK

27. NextTurn

The server notifies the beginning of the turn of a player

Arguments

- NextTurn: sends the player that has to start their turn and the name (sting) of the phase

Possible Responses

- ACK
- NACK

28. Ping

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

Arguments

- Ping: String

Possible Responses

- Pong: String

29. GenericError

When an error occurs in the server it is notified to the clients connected and the match ends

Arguments

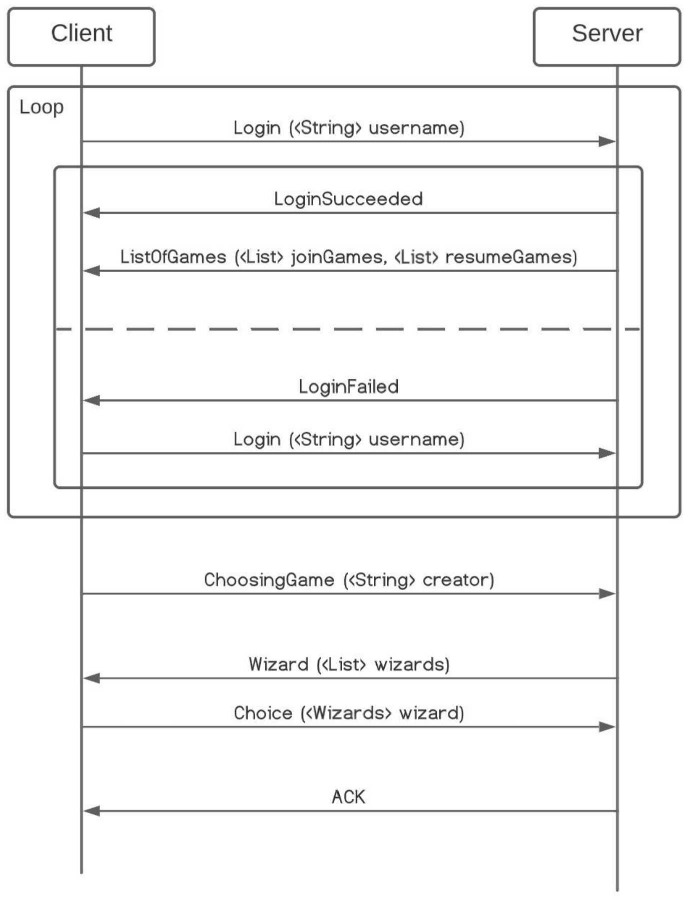
- GenericError: the message of the thrown exception

Possible Responses

- ACK

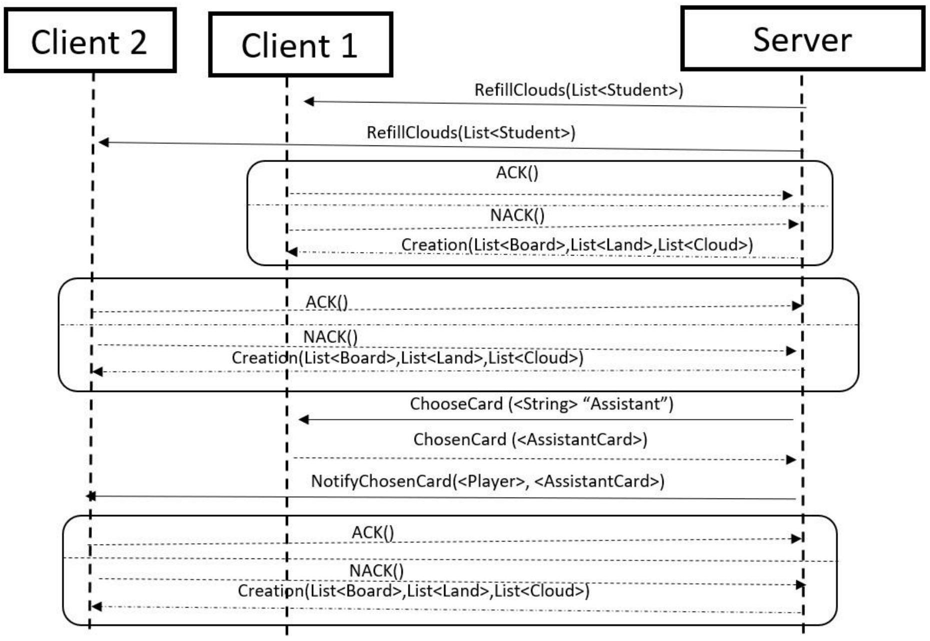
Scenarios

1. Login


Client
Server
Loop
Login (<String> username)
LoginSucceeded
ListOfGames (<List> joinGames, <List> resumeGames)
LoginFailed
Login (<String> username)
ChoosingGame (<String> creator)
Wizard (<List> wizards)
Choice (<Wizards> wizard)
ACK

The client sends the message Login and the server answers LoginSucceeded or LoginFailed (the log fails if another client is connected with the specified username). This goes on until the client receives LoginSucceeded.
Then, the server sends ListOfGames and the client chooses between joining, resuming or creating a match and sends the choice to the server.
At this point the server asks the client to choose a wizard between the available ones (the ones which haven't been chosen by any other player in the match yet) and the client answer with Choice.
In the end the server sends ACK to confirm the choice

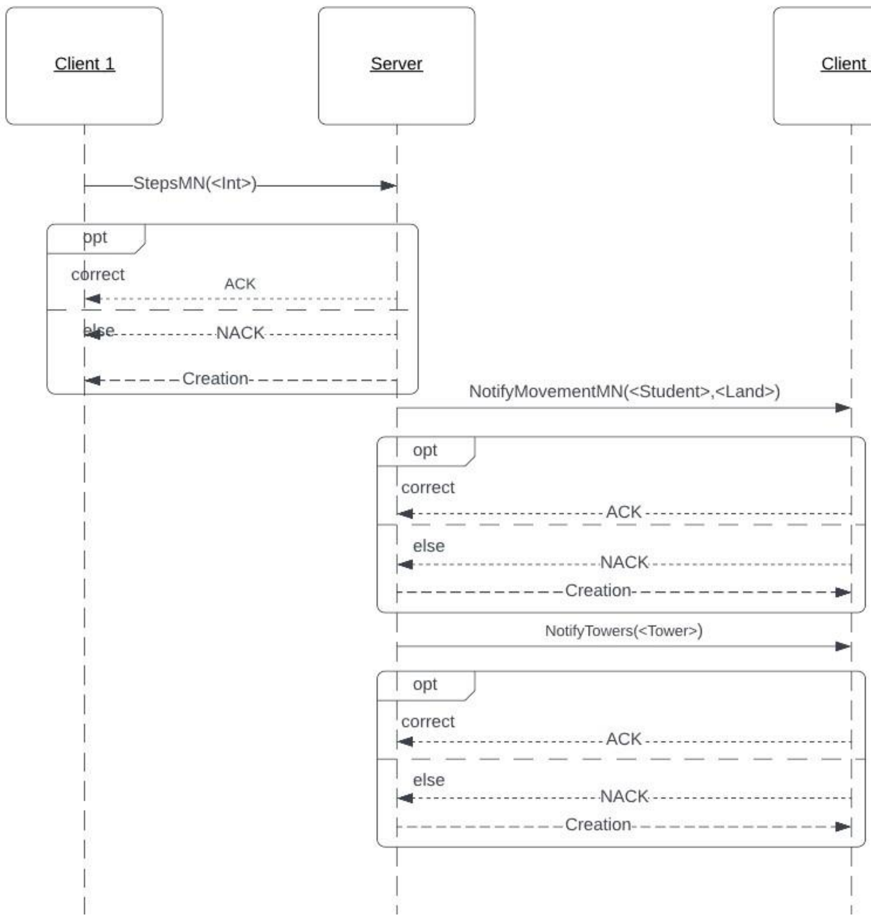
2. Planning



Client 2
Client 1
Server
RefillClouds(List<Student>)
RefillClouds(List<Student>)
ACK()
NACK()
Creation(List<Board>,List<Land>,List<Cloud>)
ACK()
NACK()
Creation(List<Board>,List<Land>,List<Cloud>)
ChooseCard (<String> "Assistant")
ChosenCard (<AssistantCard>)
NotifyChosenCard(<Player>, <AssistantCard>)
ACK()
NACK()
Creation(List<Board>,List<Land>,List<Cloud>)

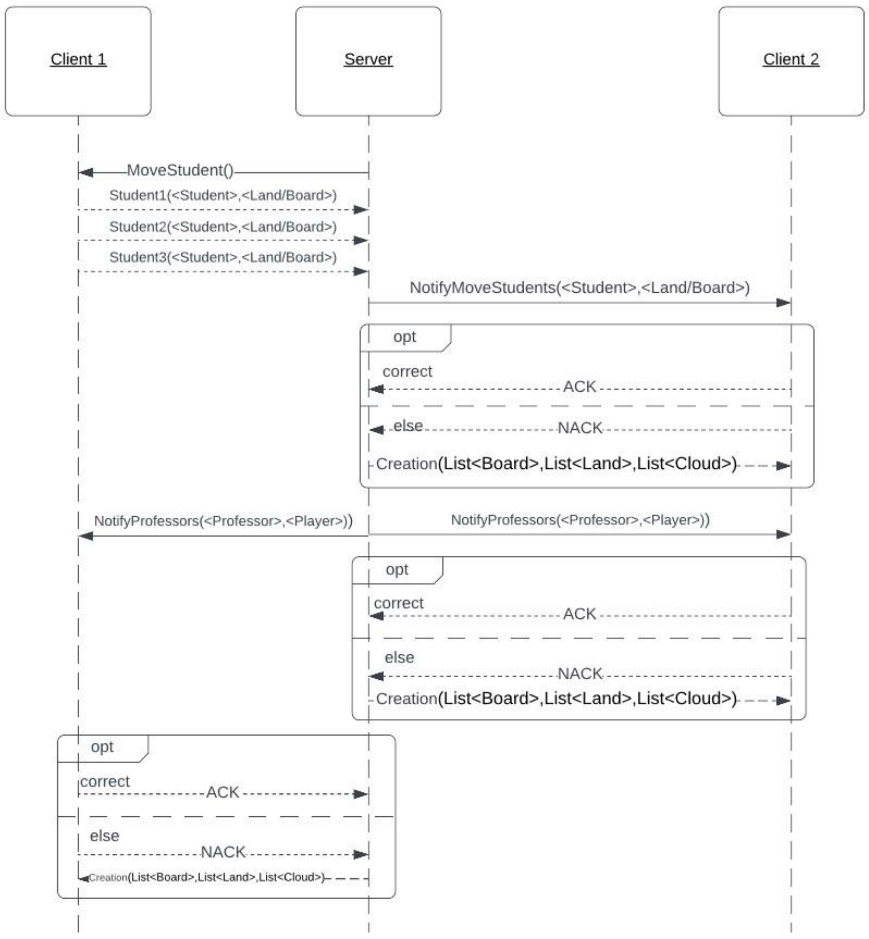
The server sends to all the clients the message RefillClouds with the Students to put into the Clouds, if the message is correctly received, they send an ACK message. If the message is not correct the server sends a NACK message and a Creation massage in order to clarify the situation. After that the 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



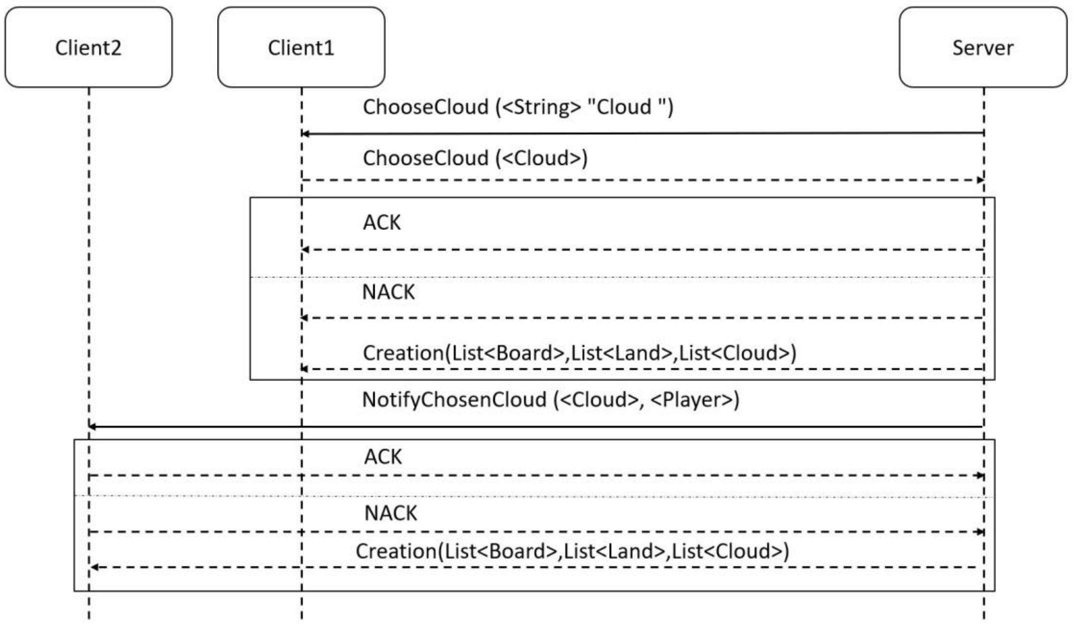
The server sends the client of the current turn (Client 1) the message MoveStudent to let him the permission of move three students, and then the Client 1 responds with the messages Student1, Student2, Student3 to communicate the choice of the player. After that the server notifies to all the other clients 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


Client 1
Server
Client 2
MoveStudent()
Student1(<Student>,<Land/Board>)
Student2(<Student>,<Land/Board>)
Student3(<Student>,<Land/Board>)
NotifyMoveStudents(<Student>,<Land/Board>)
opt
correct
ACK
else
NACK
Creation(List<Board>,List<Land>,List<Cloud>)
NotifyProfessors(<Professor>,<Player>))
NotifyProfessors(<Professor>,<Player>))
opt
correct
ACK
else
NACK
Creation(List<Board>,List<Land>,List<Cloud>)
opt
correct
ACK
else
NACK
Creation(List<Board>,List<Land>,List<Cloud>)

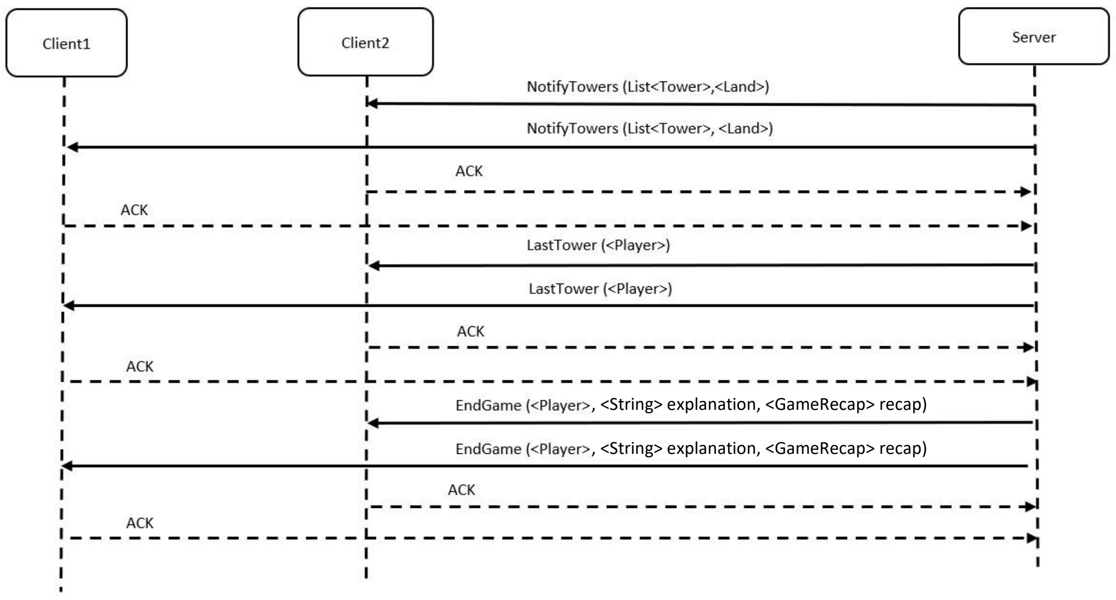
The client that is playing in the current turn notifies to the server his choice about the movement of Mother Nature with the message StepsMN and the server responds with an ACK message. After that the server notifies to all the other client the choice of the movement of Mother Nature with the message 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


Client2
Client1
Server
ChooseCloud (<String> "Cloud ")
ChooseCloud (<Cloud>)
ACK
NACK
Creation(List<Board>,List<Land>,List<Cloud>)
NotifyChosenCloud (<Cloud>, <Player>)
ACK
NACK
Creation(List<Board>,List<Land>,List<Cloud>)

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


Client1
Client2
Server
NotifyTowers (List<Tower>,<Land>)
NotifyTowers (List<Tower>, <Land>)
ACK
ACK
LastTower (<Player>)
LastTower (<Player>)
ACK
ACK
EndGame (<Player>, <String> explanation, <GameRecap> recap)
EndGame (<Player>, <String> explanation, <GameRecap> recap)
ACK
ACK

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.