Eriantys Protocol Documentation

Riccardo Milici, Riccardo Motta, Matteo Negro
Group 2

A.Y. 2021 - 2022

Contents

1 Messages

1.1 Ping

This message is sent from the server to each client in order to check if they're still online.

Request

```
{
  "type": "ping"
}
```

Response

```
{
  "type": "pong"
}
```

1.2 Error

This message is sent whenever an error occurs.

Request Parameters:

- `message`: a text which describes the cause of the error.

```
{
  "type": "error",
  "message": "Cause"
}
```

Response This message doesn't require a response.

1.3 Game creation

This message is sent from the client to the server in order to create a new game with the specified parameters.

Request Parameters:

- `playersNumber`: the total number of the players needed for the game;
- `expert`: a boolean parameter which indicates the difficulty level of the game.

```
{
  "type": "gameCreation",
  "playersNumber": 2,
  "expert": false
}
```

Response Parameters:

- `code`: the alphanumerical game identification code generated by the server.

```
{
 "type": "gameCreation",
 "code": "ABC12"
}
```

1.4 Enter game

This message is sent from the client to the server in order to search for a desired game and begin the login sequence.

Request Parameters:

- `code`: the alphanumerical game identification code generated by the server.

```
{
  "type": "enterGame",
  "code": "ABC12"
}
```

Responses Parameters:

- `found`: a boolean parameter which indicates if the searched game has been found.
- `expectedPlayers`: the number of the players needed for the searched game.
- `players`: a list of the players signed in the game, with their name and connection status.

```
{
  "type": "enterGame",
  "found": true,
  "expectedPlayers": 3,
  "players": [
    {
      "name": "Player 1",
      "online": true
    },
    {
      "name": "Player 2",
      "online": false
    }
  ]
}
```

```
{
  "type": "enterGame",
  "found": false
}
```

1.5 Waiting room

This message is periodically sent from the server to the client in order to update the waiting room players list.

Request Parameters:

- `expectedPlayers`: the total players number of the game.
- `players`: the players that are currently online.

```
{
  "type": "waitingRoomUpdate",
  "expectedPlayers": "3",
  "players": [
    {
      "name": "Player 1",
      "online": true
    },
    {
      "name": "Player 2",
      "online": false
    }
  ]
}
```

Response This message doesn't require a response.

1.6 Player login

This message is sent from the client to the server in order to log into a game, after the desired game has been found and a username has been selected.

Request Parameters:

- `name`: the name choosen by the player.

```
{
  "type": "login",
  "name": "Player 2"
}
```

Response Parameters:

- `success`: a boolean parameter which indicates if the login sequence has been completed successfully.

```
{
  "type": "login",
  "success": true
}
```

1.7 Player logout

This message is sent from the client to the server in order to log out from a game.

Request Parameters:

- `name`: the name choosen by the player.

```
{
  "type": "logout"
}
```

Response This message doesn't need a response.

1.8 Turn enable

This message is sent from the server to a specific client, every turn to enable or disable the game of the player that receives this message, in order to manage the game shifting.

Requests Parameters:

- `enable`: a boolean parameter which indicates whether the player can play.

```
{
  "type": "turnEnable",
  "enable": true
}
```

```
{
  "type": "turnEnable",
  "enable": false
}
```

Response This message doesn't require a response.

1.9 Game start

This message is sent from the server to all the clients, telling that the game is full and it's starting.

```
{
  "type": "gameStart"
}
```

Response This message doesn't require a response.

1.10 Commands

These messages are sent from the client to the server during the game, in order to tell the server how and what the player wants to play. After the server has updated the game model, it will notify the other player's clients about this move.

1.10.1 Play assistant

Request Parameters:

- `player`: the player that's performing the move.
- `assistant`: the identification number of the assistant card played.

```
{
  "type": "command",
  "subtype": "playAssistant",
  "player": "Player 1",
  "assistant": 10
}
```

Response This message doesn't require a response.

1.10.2 Move professor

Requests Parameters:

- `player`: the player that's performing the move.
- `color`: the professor's colour.
- `from`: the player that owns the professor before the command.
- `to`: the player that will own the professor after the command.

```
{
  "type": "command",
  "subtype": "move",
  "pawn": "professor",
  "color": "RED",
  "from": null,
  "to": "Player 1"
}
```

```
{
  "type": "command",
  "subtype": "move",
  "pawn": "professor",
  "color": "RED",
  "from": "Player 2",
  "to": "Player 1"
}
```

Response This message doesn't require a response.

1.10.3 Move student

Requests Parameters:

- `player`: the player that's performing the move.
- `color`: the student's colour.
- `from`: where the student was before the movement.
- `to`: where the student will be after the movement.
- `fromId`: the identification number of the place where the student comes from.
- `toId`: the identification number of the place where the student has to move.

```
{
  "type": "command",
  "subtype": "move",
  "pawn": "student",
  "player": "Player 1",
  "color": "RED",
  "from": "entrance",
  "to": "diningRoom",
  "fromId": null,
  "toId": null
}
```

```
{
  "type": "command",
  "subtype": "move",
  "pawn": "student",
  "player": "Player 1",
  "color": "RED",
  "from": "entrance",
  "to": "island",
  "fromId": null,
  "toId": 0
}
```

```
{
  "type": "command",
  "subtype": "move",
  "pawn": "student",
  "player": "Player 1",
  "color": "RED",
  "from": "card",
  "to": "entrance",
  "fromId": 0,
  "toId": null
}
```

Response This message doesn't require a response.

1.10.4 Move mother nature

Request Parameters:

- `island`: the identification number of the island where mother nature has to move.

```
{
  "type": "command",
  "subtype": "motherNature",
  "island": 0
}
```

Response This message doesn't require a response.

1.10.5 Ban

Request Parameters:

- `island`: the identification number of the island that will be ban (special character's effect).

```
{
  "type": "command",
  "subtype": "ban",
  "island": 0
}
```

Response This message doesn't require a response.

1.10.6 Ignore color

Request Parameters:

- `color`: the color to ignore during an island resolution.

```
{
  "type": "command",
  "subtype": "ignore",
  "color": "BLUE"
}
```

Response This message doesn't require a response.

1.10.7 Return color

Request Parameters:

- `color`: the color of the students to move back from the dining room to the entrance.

```
{
  "type": "command",
  "subtype": "return",
  "color": "RED"
}
```

Response This message doesn't require a response.

1.10.8 Pay character

Request Parameters:

- `player`: the player that's performing the move.
- `character`: the identification number of the character that will be activated.

```
{
  "type": "command",
  "subtype": "pay",
  "player": "Player 1",
  "character": 1
}
```

Response This message doesn't require a response.

1.10.9 Refill cloud

This message is sent from a specific client to the server (which then redirects the same massage to all the other clients to update their views) when the player, at the end of his/her turn, takes the students from a cloud to refill the entrance.

Request Parameters:

- `player`: the name of the player that takes the students.
- `cloud`: the cloud from which the player takes the students.

```
{
  "type": "command",
  "subtype": "refill",
  "player": "Player 1",
  "cloud": 0
}
```

Response This message doesn't require a response.

1.11 Win

This message is sent from the server to each clients in order to indicate that the game is ended, carrying the winners list.

Request Parameters:

- `winners`: A list of the players that won the game.

```
{
  "type": "win",
  "winners": [
    "player1",
    "player3"
  ]
}
```

Response This message doesn't require a response.

1.12 Status

This message is sent from the server to the client in order to reload the game, when it satrtes or whene a player reconnects to the game.

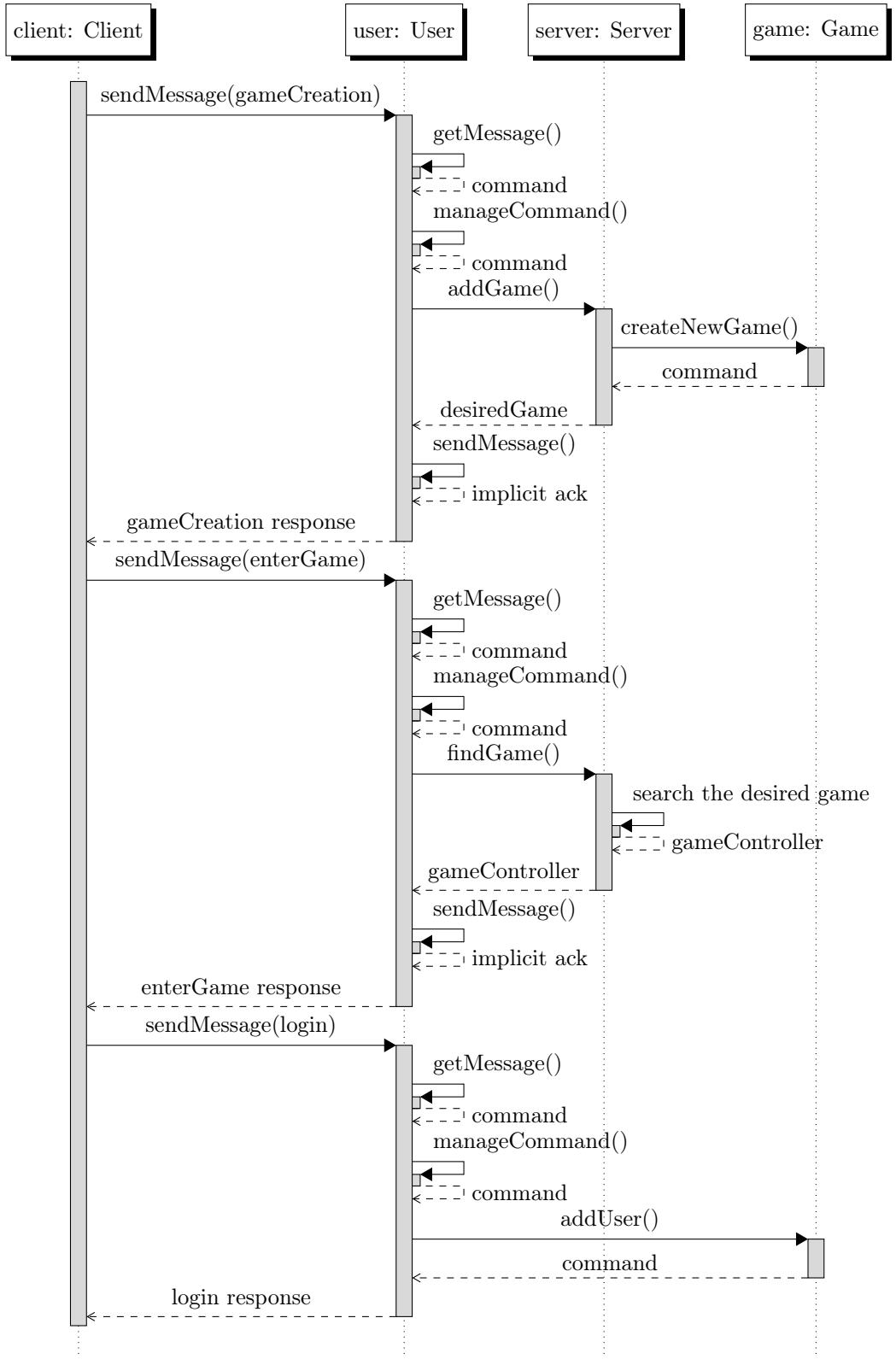
Request Parameters: all the parameters that are required to restart the game client-side.

```
{
  "type": "state",
  "players": [
    {
      "name": "Player 1",
      "wizardType": "GREEN",
      "coins": 0,
      "assistants": [
        0,
        10
      ],
      "schoolBoard": {
        "towerType": "WHITE",
        "towersNumber": 0,
        "entrance": {
          "BLUE": 1,
          "FUCHSIA": 0,
          "GREEN": 1,
          "RED": 0,
          "YELLOW": 1
        },
        "diningRoom": {
          "BLUE": 1,
          "FUCHSIA": 0,
          "GREEN": 1,
          "RED": 0,
          "YELLOW": 1
        }
      }
    }
  ],
  "board": {
    "bag": {
      "BLUE": 0,
      "FUCHSIA": 0,
      "GREEN": 0,
      "RED": 0,
      "YELLOW": 0
    },
    "clouds": [
      {
        "BLUE": 0,
        "FUCHSIA": 0,
        "GREEN": 0,
        "RED": 0,
        "YELLOW": 0
      }
    ],
    "islands": [
      {
        "size": 1,
        "tower": "BLACK",
        "ban": false,
        "students": {
          "BLUE": 0,
          "FUCHSIA": 0,
          "GREEN": 0,
          "RED": 0,
          "YELLOW": 0
        }
      }
    ],
    "playedAssistants": [
      {
        "player": "Player 1",
        "assistant": 10
      }
    ],
    "characters": [
      {
        "id": 1,
        "effectCost": 0,
        "alreadyPaid": false,
        "paidInRound": false,
        "active": false
      }
    ],
    "professors": {
      "BLUE": null,
      "FUCHSIA": null,
      "GREEN": null,
      "RED": null,
      "YELLOW": null
    },
    "motherNatureIsland": 0
  }
}
```

Response This message doesn't require a response.

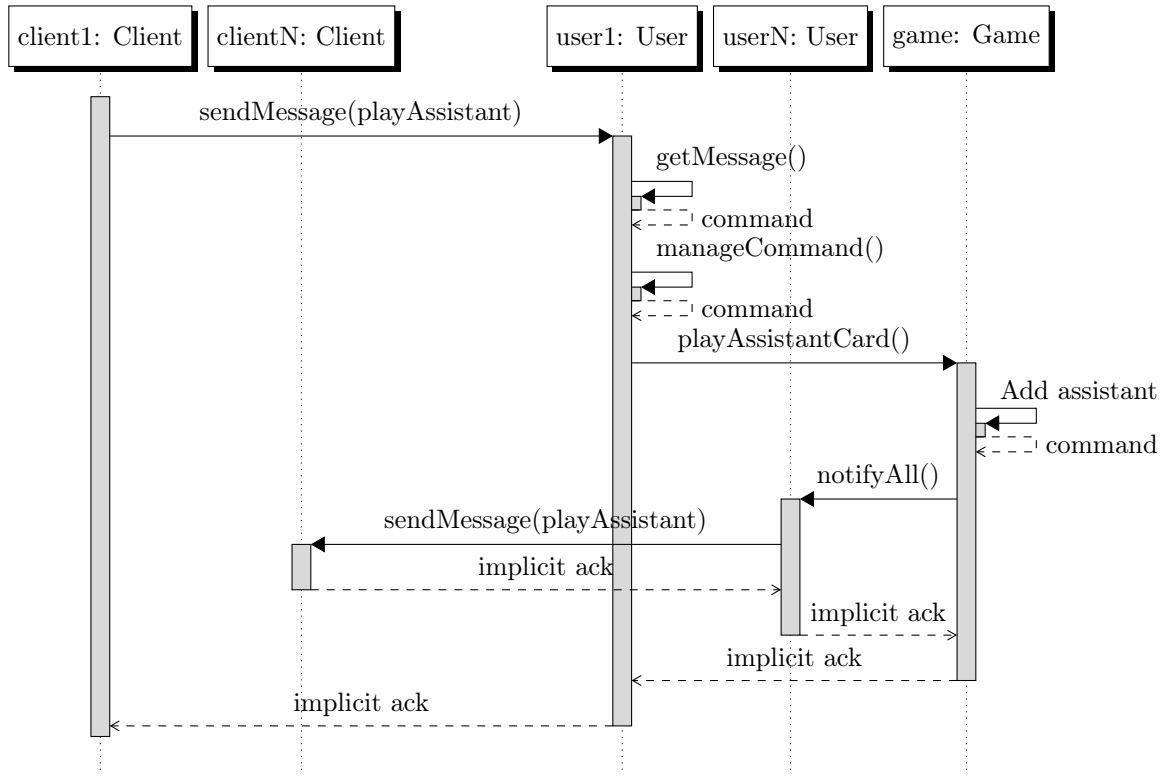
2 Scenarios

2.1 Connection to a game



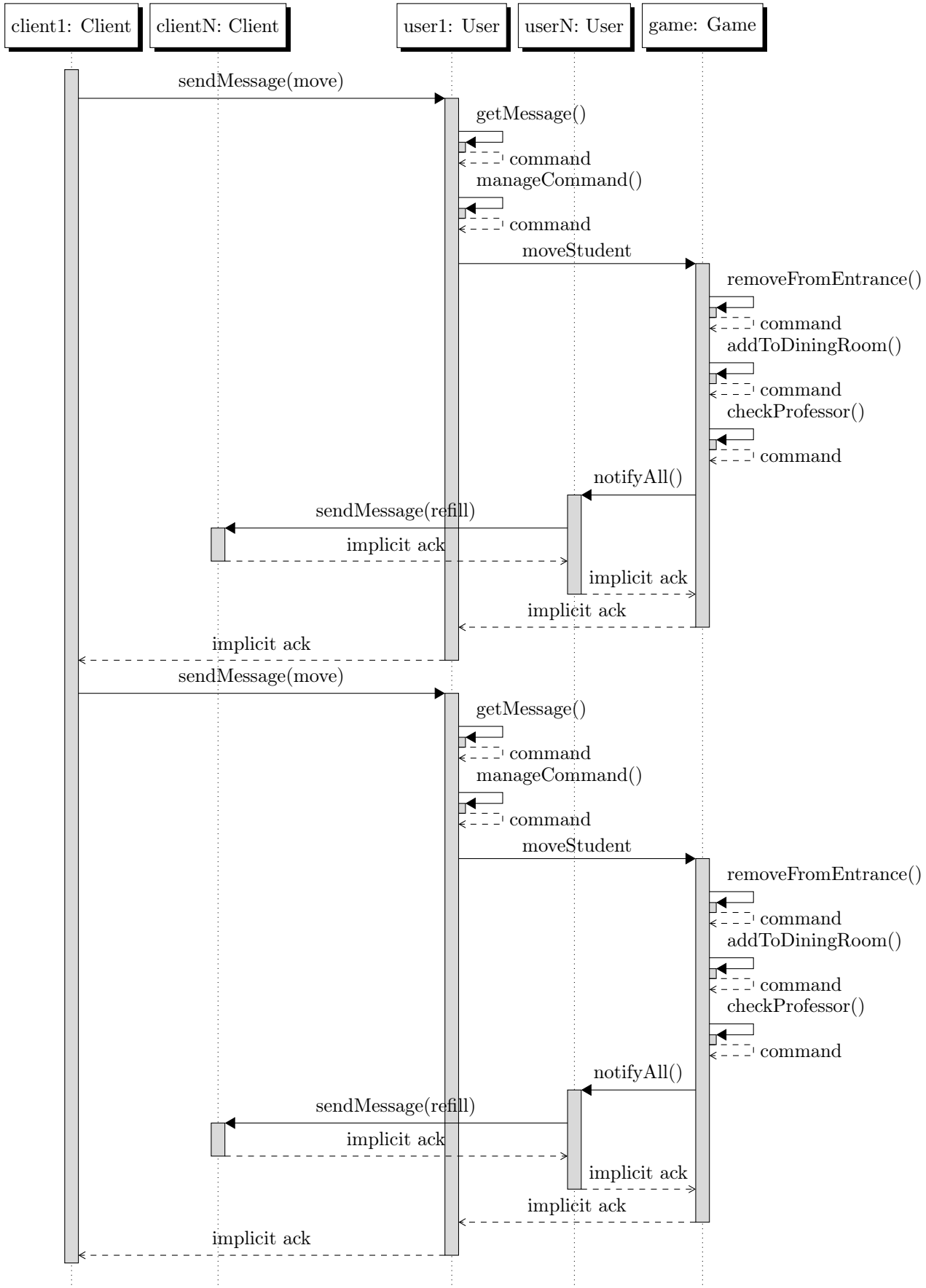
This scenario represents the game creation phase. The user connects to the game server, and, if he/she wants, sends a game creation request, building a new game. After this optional passage, the client enters the alphanumerical code of the game he/she wants to access. The server replies with a message containing information about the game searched (if it exists), and then the user chooses a name and asks to log into the game with a "player entrance message". Finally, the server adds the new player to the game.

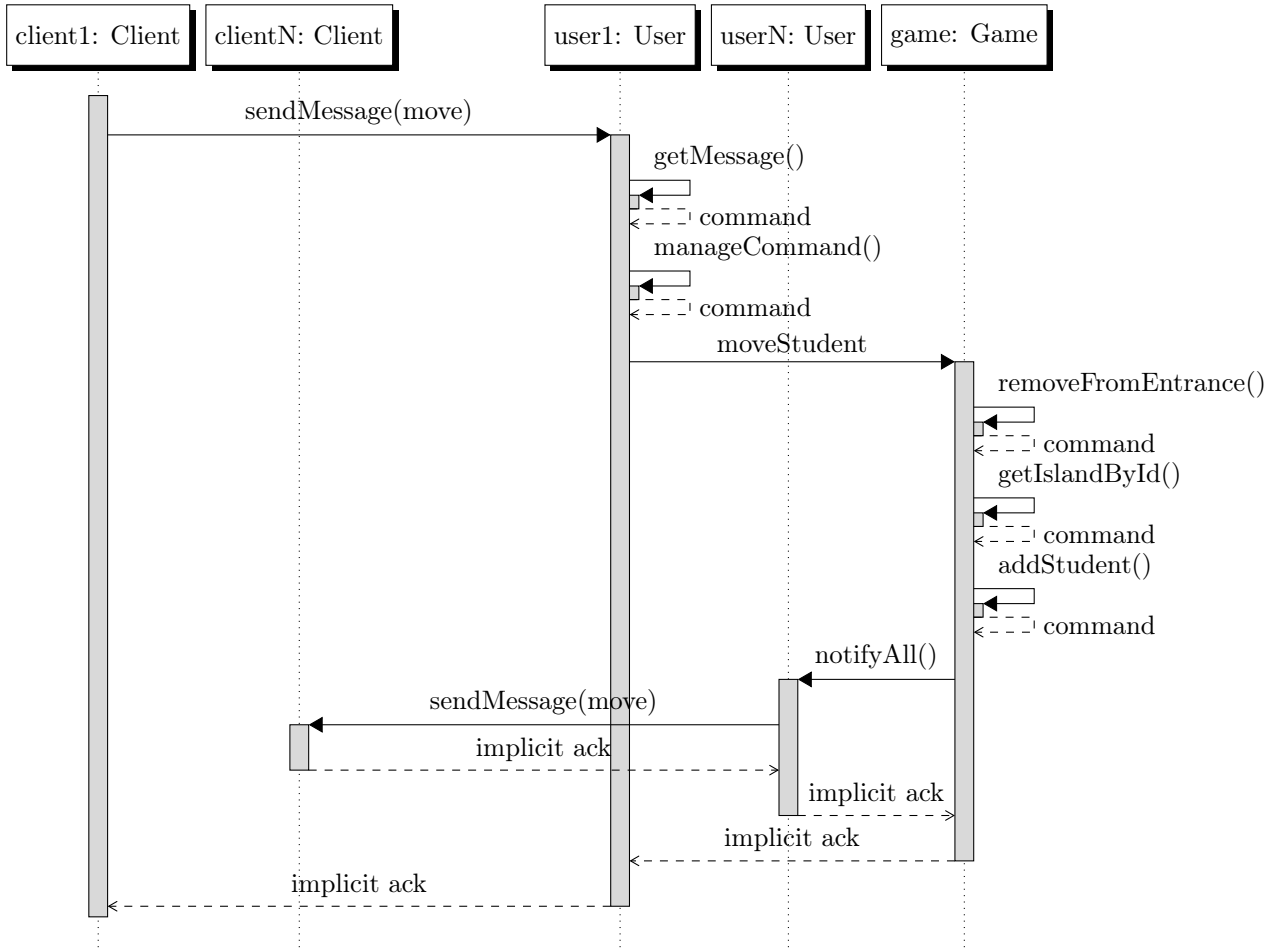
2.2 Planning phase (assistant card play)



This scenario represents the first game phase, during which each player has to choose and play an assistant card from his/her hand; this is achieved sending a `playAssistant` message to the server. The server handles the command adding the corresponding played card into the model status and then it notifies the other users about this move, sending a message through each client's TCP tunnel (the current player's client doesn't need to be updated about this action).

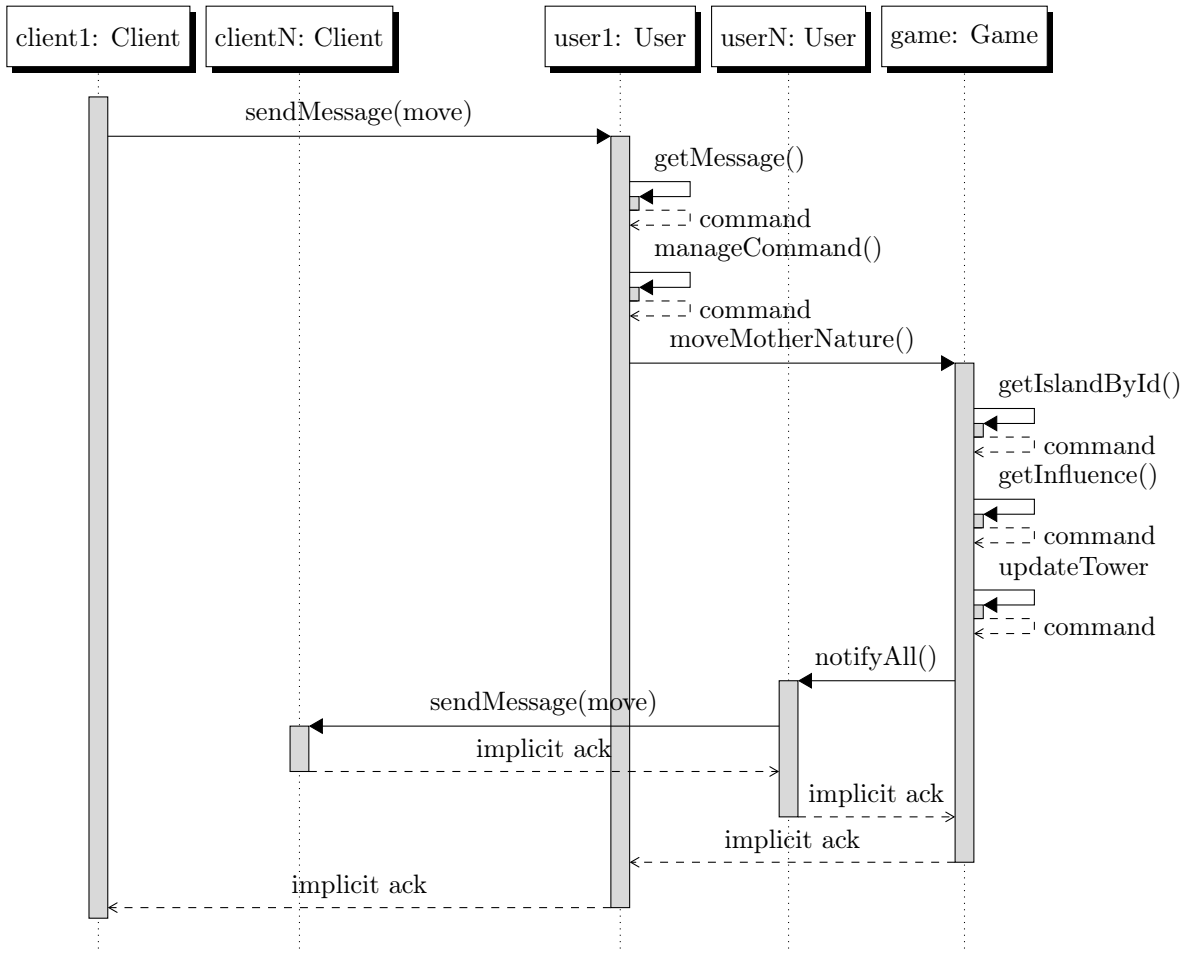
2.3 Action phase, move #1





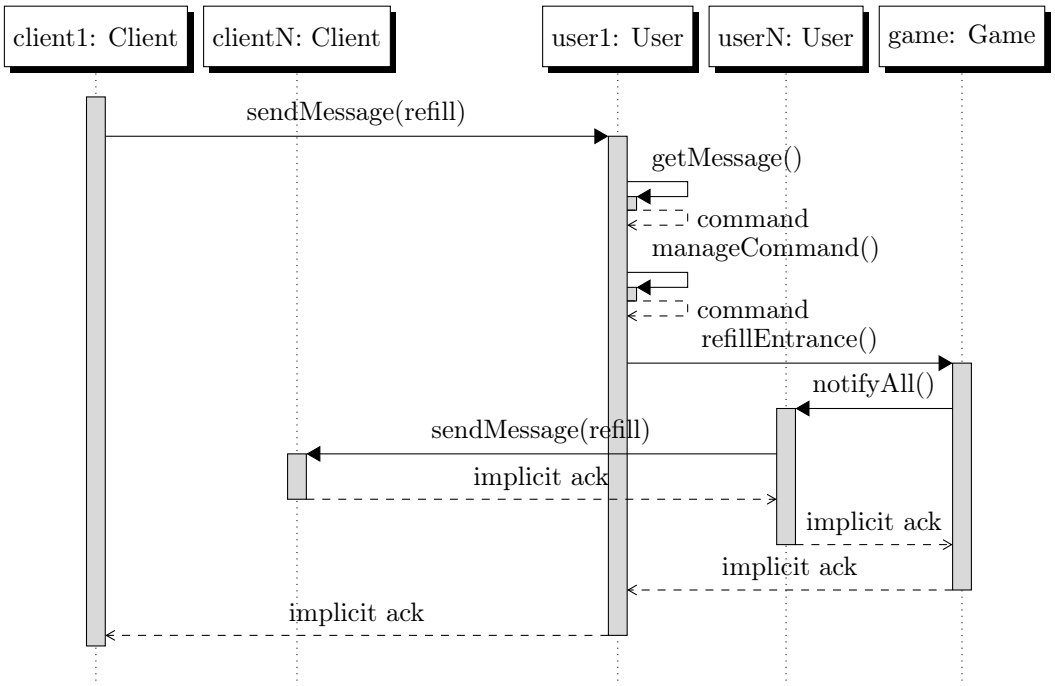
This scenario represents the first action the player can do during his/her turn. The client behind that player send a message telling the server that s/he would like to move a student from his/her school entrance to his/her dining room or an island. The server, once receive the message, sends it back to every other client (here generically represented as `clientN`) in order to let them update their views. The message can carry information about special movements link to the character in the expert mode.

2.4 Action phase, move #2



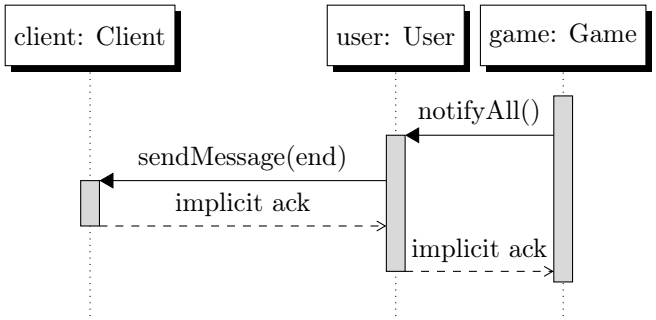
This scenario represents the second action the player can do during his/her turn. The client behind that player send a message telling the server that s/he would like to move mother nature to a specific island, that starts a sequence of call to update the state of the game model (influence, tower...). The server, once receive the message, sends it back to every other client (here generically represented as `clientN`) in order to let them update their views.

2.5 Action phase, move #3



This scenario represents the last action the player can do during his/her turn. The client behind that player send a message telling the server that s/he took the students from a specific cloud to refill his/her school entrance. The server, once receive the message, sends it back to every other client (here generically represented as `clientN`) in order to let them update their views.

2.6 Game end



This scenarios represents the notification that the game has reached a point where it's concluded. In this case, the server notifies every client that the game has ended and if there is a winner, tells every client who is, otherwise says that the game ended tie.