# Hard Deadline Communication

Edited by:

Andrea Alfieri
Andrea Ceruti
Amedeo Carrioli

AM29

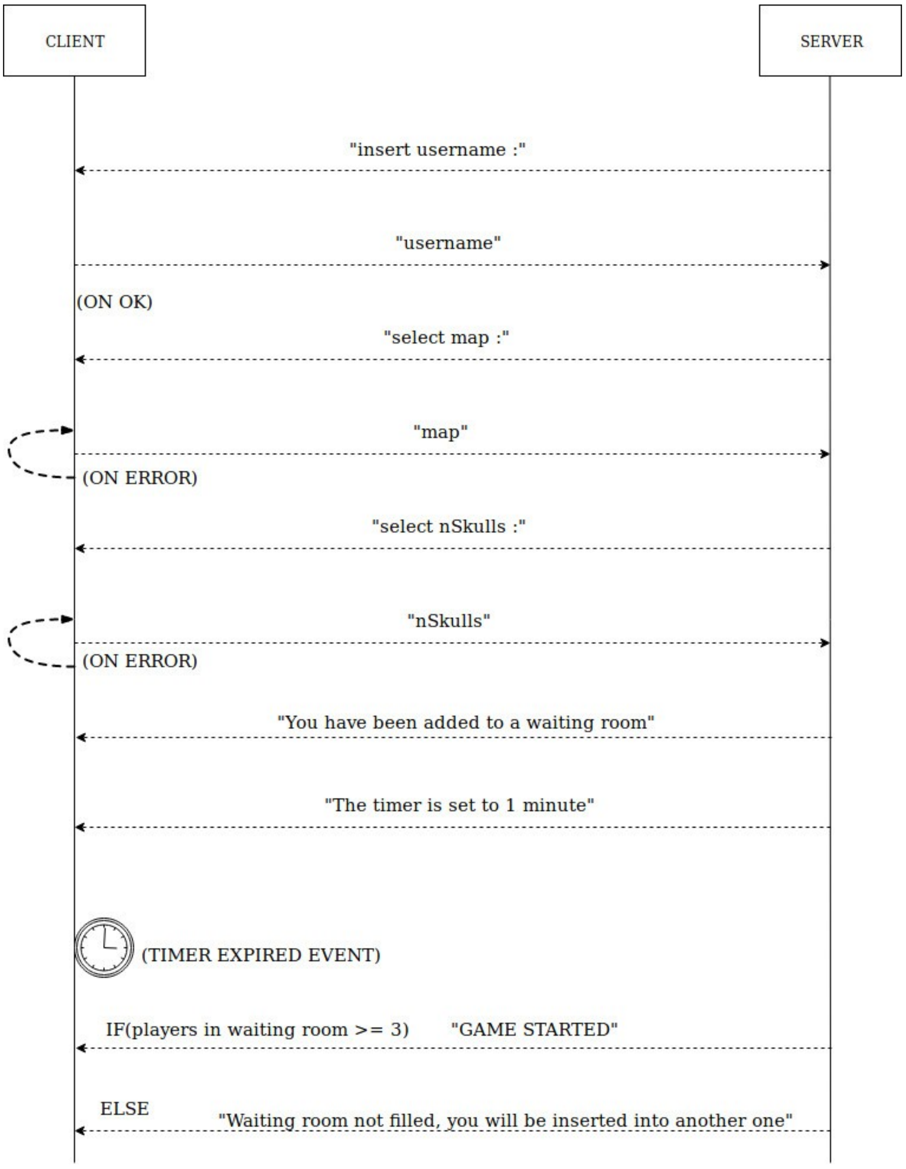
Academic Year 2018-2019

# Communication Protocol

## Login

The login phase makes use of a WaitingRoom object, whose scope is to hold all the new players until the beginning of the new game.
At any time, our server is able to accept new connections by any client.
Once connected, the new client is asked for his username:
if the username was already registered for an active game or a waiting room, it is sent to the correct place. Otherwise, it is asked for a vote on which map to play on and on how many skulls to play with.
If the votes are correct, the player is added to a waiting room.

When the timer of the waiting room expires, the game starts and a “GAME STARTED” message is sent to all players.



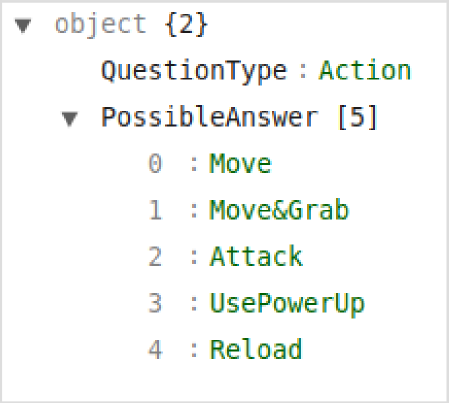
CLIENT
SERVER
"insert username :"
"username"
(ON OK)
"select map :"
"map"
(ON ERROR)
"select nSkulls :"
"nSkulls"
(ON ERROR)
"You have been added to a waiting room"
"The timer is set to 1 minute"
(TIMER EXPIRED EVENT)
IF(players in waiting room >= 3)
"GAME STARTED"
ELSE
"Waiting room not filled, you will be inserted into another one"

## Game messages

The communication during the game is based on two types of messages: “Question” and “Answer”.
Both of these messages are JSON files and they can only travel in one direction: the Question always goes from the server to the client, while Answer only moves the opposite way.

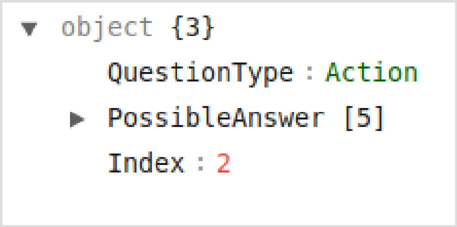
### Question

This type of message has 2 attributes: QuestionType and PossibleAnswer.
QuestionType defines what question the server is asking the player.
PossibleAnswer is the list of acceptable answers, so the client only has to reply with the index in this array of its choice.
Each QuestionType defines how the PossibleAnswer should be structured:
for example, if the QuestionType is “Action”, the PossibleAnswer is an array of String that, depending on the game state, might look like ["Move”, “Move&Grab”, “Attack”, “UsePowerUp”, “Reload”].



```
▼ object {2}
      QuestionType : Action
    ▼ PossibleAnswer [5]
          0  : Move
          1  : Move&Grab
          2  : Attack
          3  : UsePowerUp
          4  : Reload
```

### Answer



```
▼ object {3}
      QuestionType : Action
    ▶ PossibleAnswer [5]
      Index : 2
```

This JSON file contains the same attributes of the Question message but also contains an Index attribute which defines the player’s choice as the index of the answer in the PossibleAnswer array. This way, when the server receives a new message, it is possible to modify the Model only looking at the QuestionType attribute and the players status.

# Question Types

## Regular Types

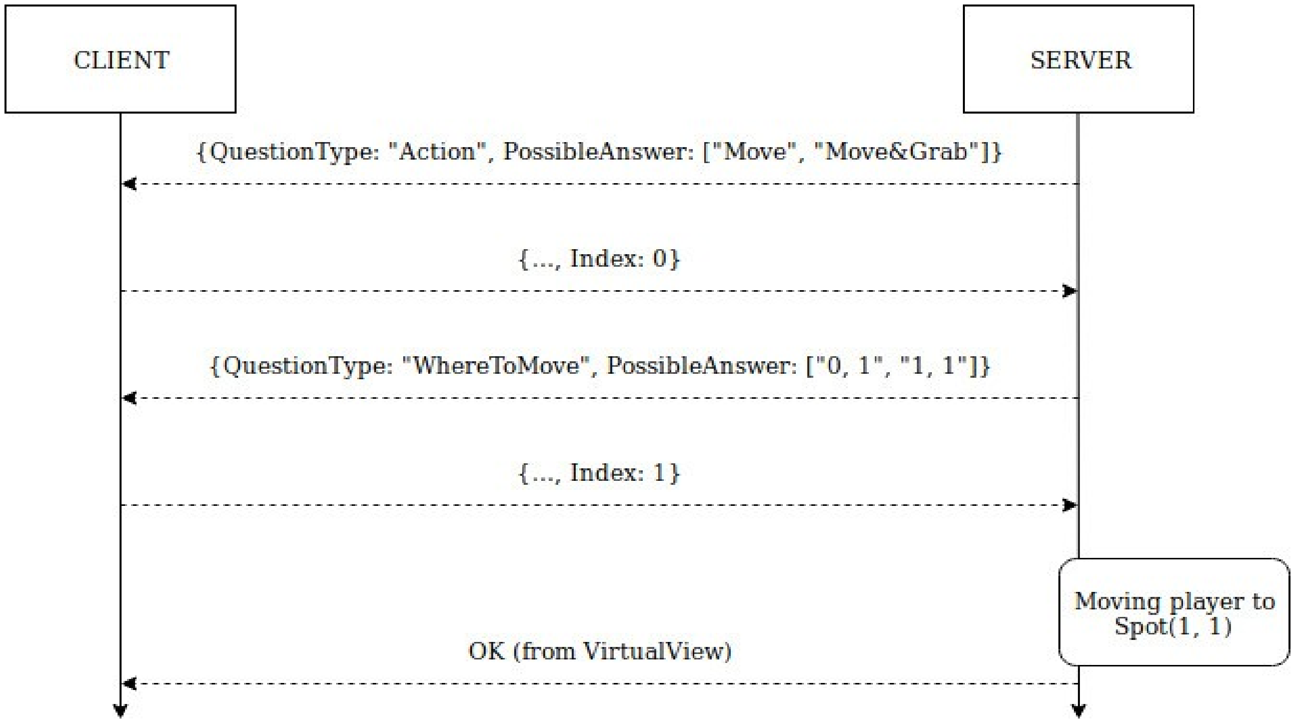
- **Action**: asks the player for the next action he wants to make (Move, Attack, …)
- **WhereToMove**: asks the player where he wants to move next.
  The PossibleAnswer strings must look like “0, 2”, which are the coordinates of the spot where the player wants to move.
- **WhereToMoveAndGrab**: asks the player where he wants to move and grab. PossibleAnswer is structured exactly like WhereToMove.
- **ChoosePowerUpToDiscard**: asks the player which power up he wants to discard. This works for choosing a power up to respawn and to choose a powerup to pay.
- **ChoosePowerUpToAttack**: asks the player which power up to attack with
- **ChooseWeaponToAttack**: asks the player which weapon he wants to use to attack
- **ChooseWeaponToPick**: asks the player which weapon he wants to pick from the spawn spot he’s on.
- **ChooseWeaponToSwitch**: asks the player which weapon he wants to put back on the board when picking up a new weapon from a spawn spot.
- **ChooseWeaponToReload**: asks the player which weapon he wants to reload
- **PayWith**: Asks the player how he wants to pay

## Shooting Types

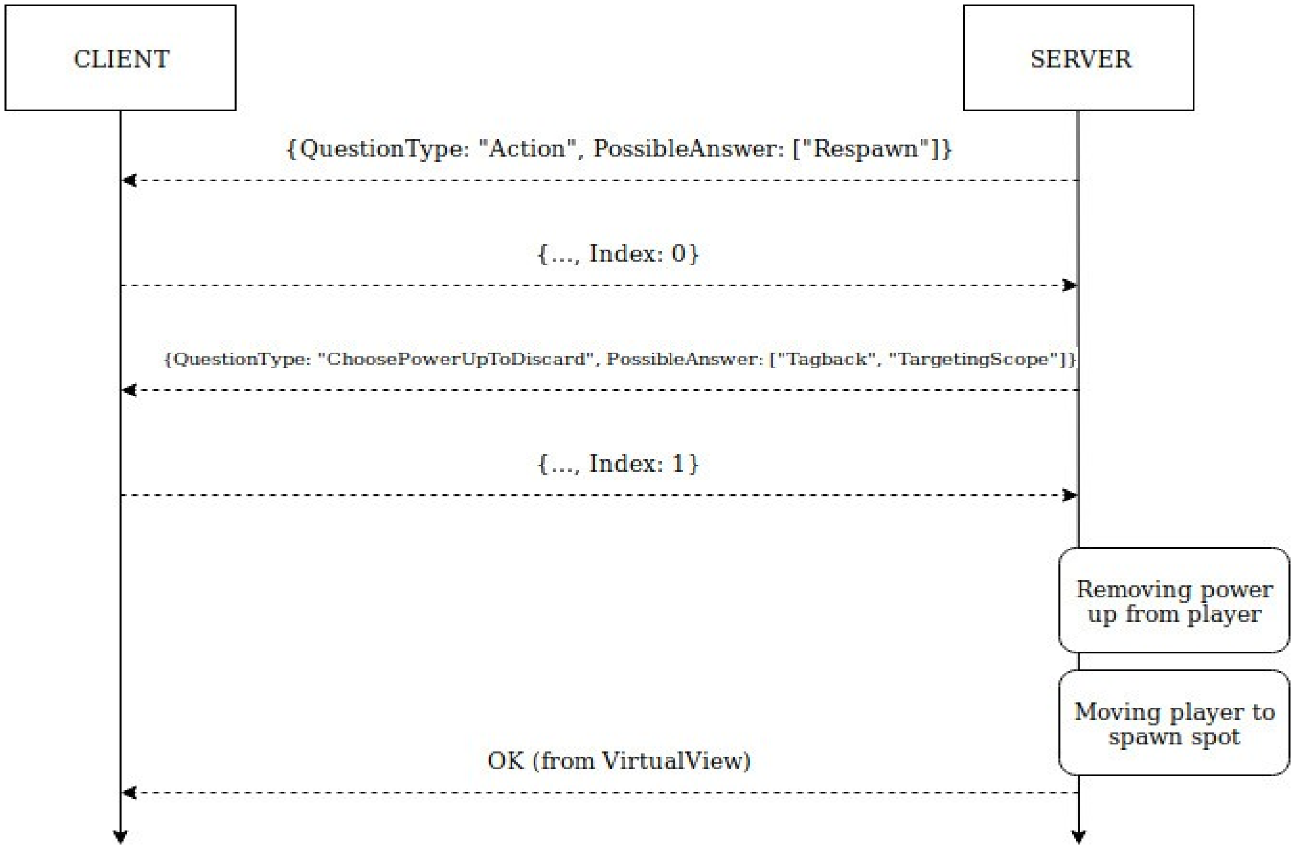
- **AddDefender**: asks the player which defender he wants to add to the defender list. The server keeps asking until the player selects the STOP answer
- **AddCoords**: works like the previous type, but asks for coordinates.
- **AddMover**: works like the previous type, but asks for which players to move
- **ChooseOrder**: asks in which order the player wants to attack

# Examples

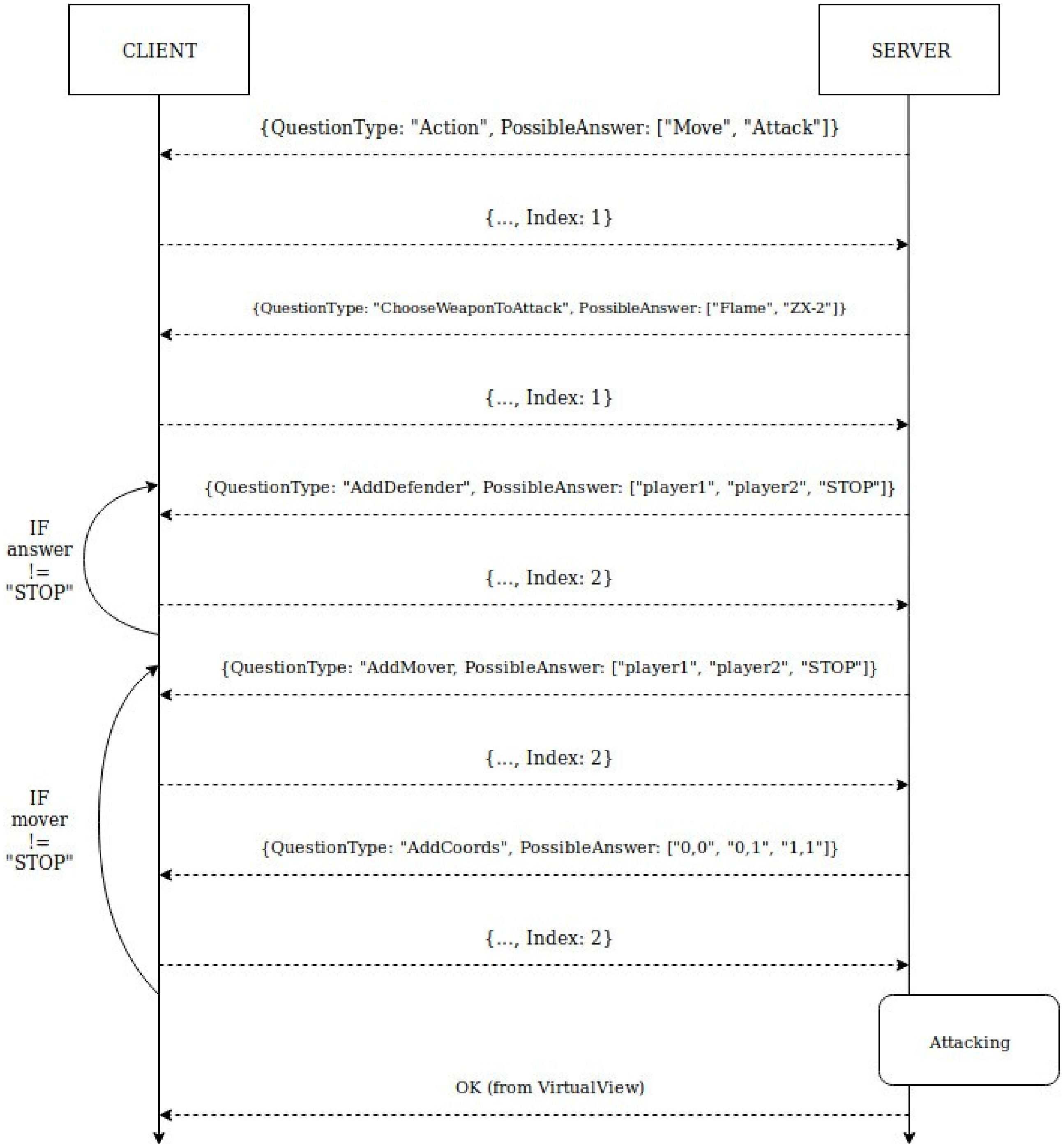
## Moving


CLIENT
SERVER
{QuestionType: "Action", PossibleAnswer: ["Move", "Move&Grab"]}
{..., Index: 0}
{QuestionType: "WhereToMove", PossibleAnswer: ["0, 1", "1, 1"]}
{..., Index: 1}
Moving player to Spot(1, 1)
OK (from VirtualView)

## Respawning


CLIENT
SERVER
{QuestionType: "Action", PossibleAnswer: ["Respawn"]}
{..., Index: 0}
{QuestionType: "ChoosePowerUpToDiscard", PossibleAnswer: ["Tagback", "TargetingScope"]}
{..., Index: 1}
Removing power up from player
Moving player to spawn spot
OK (from VirtualView)

# Attacking


CLIENT
SERVER
{QuestionType: "Action", PossibleAnswer: ["Move", "Attack"]}
{..., Index: 1}
{QuestionType: "ChooseWeaponToAttack", PossibleAnswer: ["Flame", "ZX-2"]}
{..., Index: 1}
{QuestionType: "AddDefender", PossibleAnswer: ["player1", "player2", "STOP"]}
IF answer != "STOP"
{..., Index: 2}
{QuestionType: "AddMover, PossibleAnswer: ["player1", "player2", "STOP"]}
IF mover != "STOP"
{..., Index: 2}
{QuestionType: "AddCoords", PossibleAnswer: ["0,0", "0,1", "1,1"]}
{..., Index: 2}
Attacking
OK (from VirtualView)