Politecnico di Milano

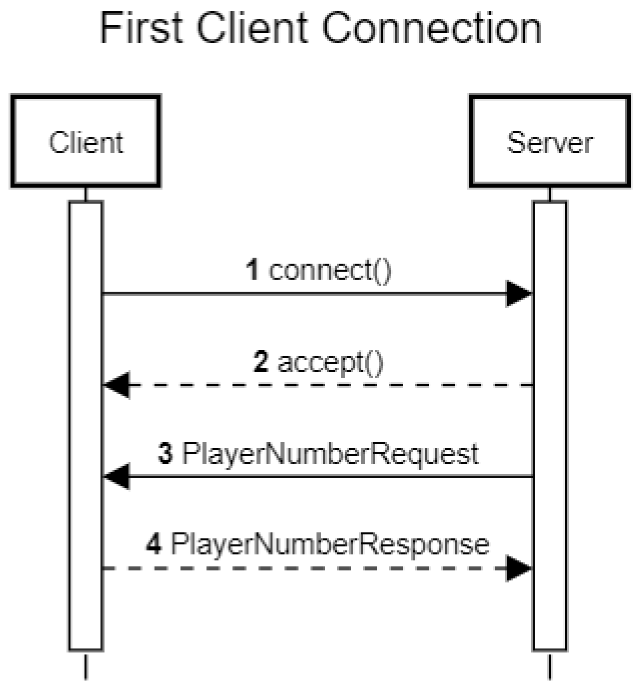
Laura Amabili – Roberto Capolungo - Walter Catalfamo

Client-Server Interaction

Contents

First Client Connection

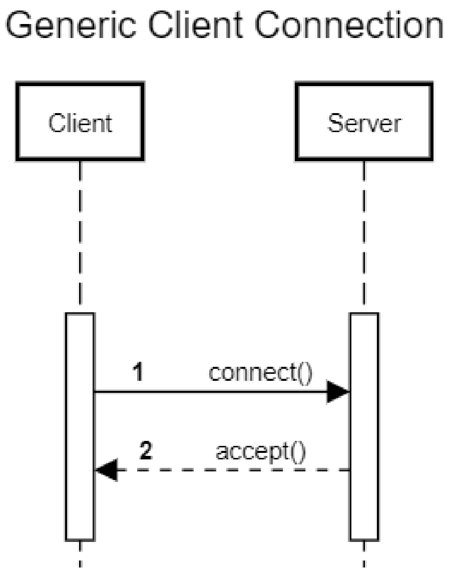
The first client requests a connection with the server. The server accepts the connection and requests the number of players.



Message	Attributes
Connect()	
Accept()	
PlayerNumberRequest	
PlayerNumberResponse	[“numberOfPlayers”]

Generic Client Connection

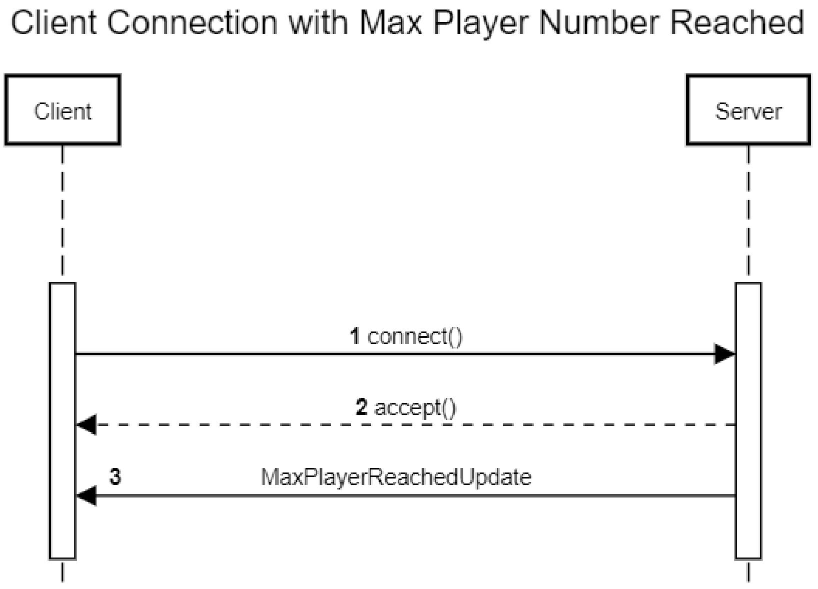
The first client requests a connection with the server. The server accepts the connection.



Message	Attributes
Connect()	
Accept()	

Client Connection with Max Player Number Reached

The first client requests a connection with the server. The server accepts the connection. However, the sever sends a message saying that the lobby is full.



Message	Attributes
Connect()	
Accept()	
MaxPlayerReachedUpdate	

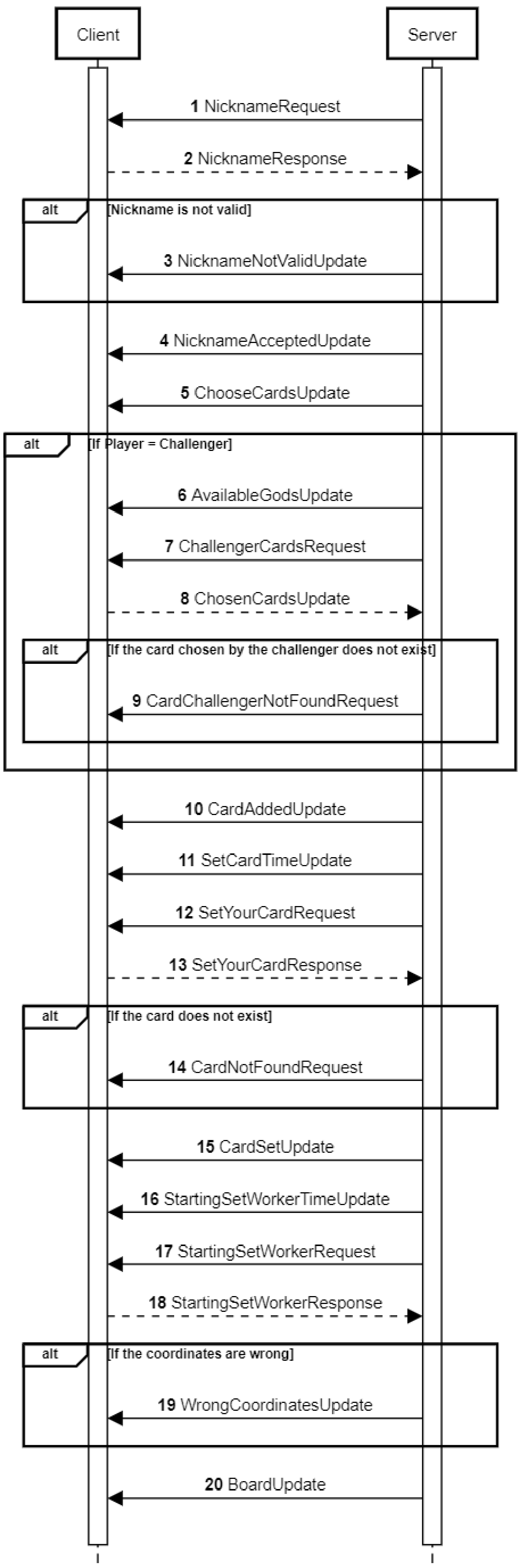
Setting Up the Lobby

The lobby is created. The god and the workers of each player are set.

The algorithm steps are the following:

1. Set nickname of all players
2. The challenger has to choose the cards
3. Each player chooses their card
4. Each player chooses the location of the workers
5. Board is printed

Setting Up the Lobby



Message	**Attributes**
NicknameRequest	
NicknameResponse	[“nickname”]
NicknameNotValidUpdate	
NicknameAcceptedUpdate	
ChooseCardsUpdate	[“challenger”]
AvailableGodsUpdate	[“cards”]
ChallengerCardsRequest	
ChosenCardsUpdate	[“chosenCard”]
CardChallengerNotFoundRequest	
CardAddedUpdate	[“addedGods”]
SetCardTimeUpdate	[“currentPlayer”]
SetYourCardRequest	[“chosenGods”]
SetYourCardResponse	[“chosenGod”]
CardNotFoundRequest	
CardSetUpdate	[“currentPlayer”, “godName”]
StartingSetWorkerTimeUpdate	[“currentPlayer”]
StartingSetWorkerRequest	[“worker”]
StartingSetWorkerResponse	[“row”,”col”,“worker”]
WrongCoordinatesUpdate	[“worker”]
BoardUpdate	[“board”]

Game

The game starts. It's a player's turn to play.

At the end of the diagram the sequence comes back to the first message and the turn is updated with the next player. Each turn is made by two steps:

1. Move
2. Build

There are two possibilities of win of a player:

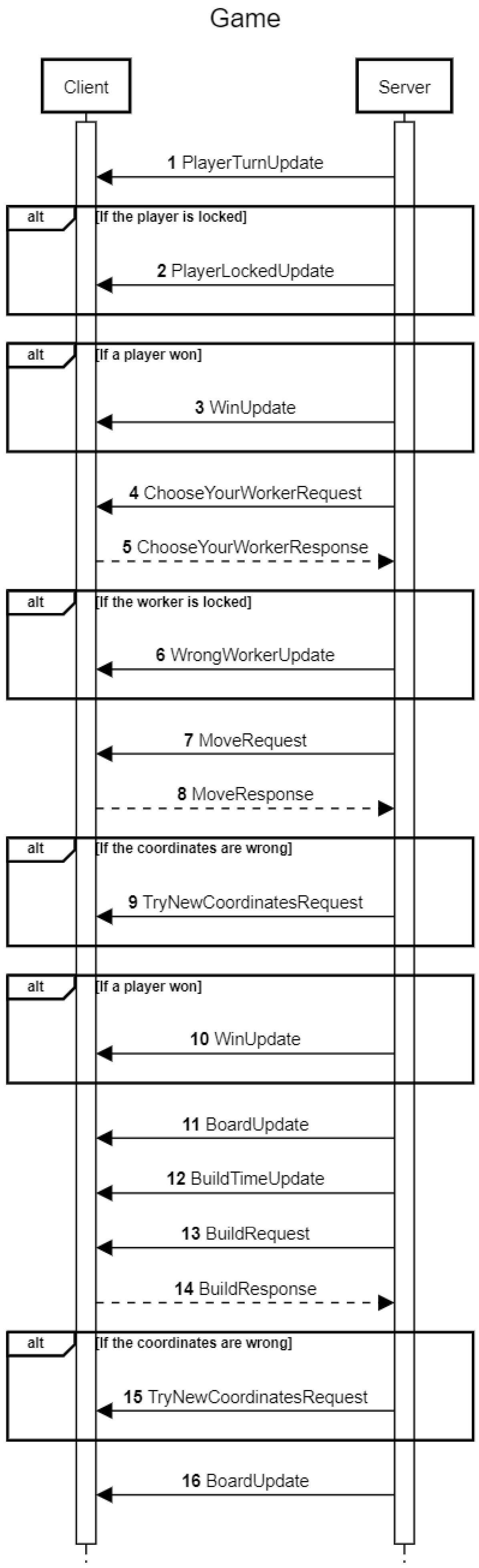
1. All the players are locked
2. A move from level 2 to level 3 is performed

Each one of this conditions is checked in this context.

The board is updated twice, each one after the actions (move, build) performed by the player.

The algorithm steps are the following:

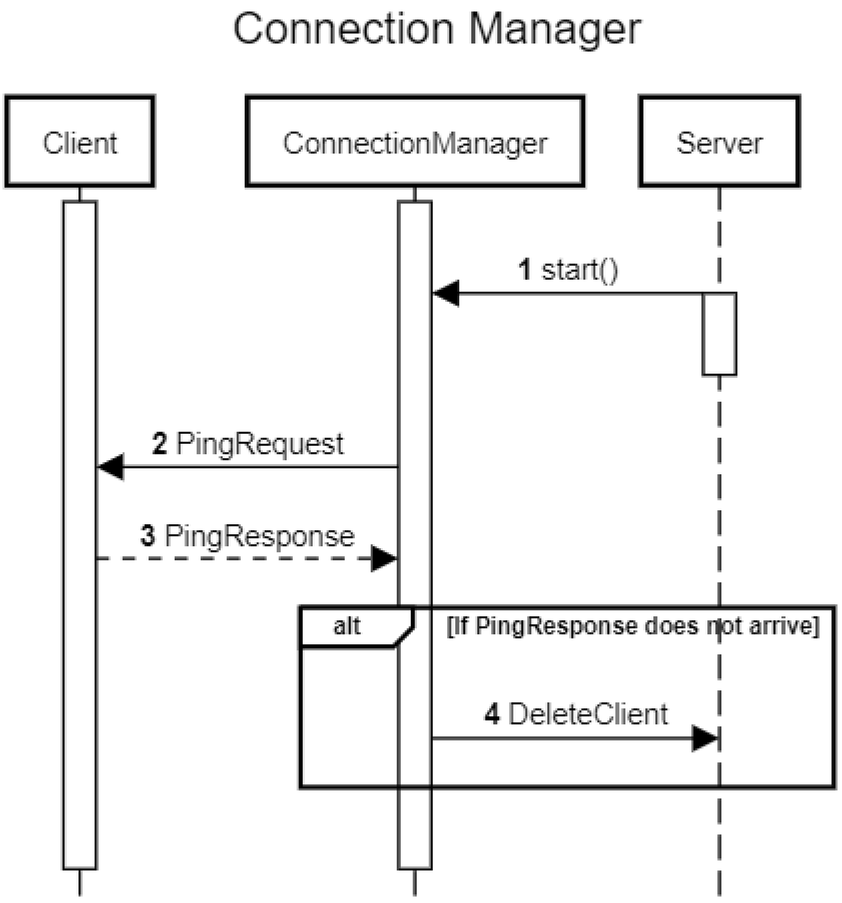
1. The current player's turn is sent
2. It's checked if someone is locked
3. The player chooses which worker wants to move and performs the move
4. The board is updated
5. The player performs the build
6. The board is updated again



Message	**Attributes**
PlayerTurnUpdate	["nickname"]
PlayerLockedUpdate	["nickname"]
WinUpdate	["winnerName"]
ChooseYourWorkerRequest	
ChooseYourWorkerResponse	["worker"]
WrongWorkerUpdate	["worker"]
MoveRequest	["worker"]
MoveResponse	["row","col","worker"]
TryNewCoordinatesRequest	["worker"]
WinUpdate	["winnerName"]
BoardUpdate	["board"]
BuildTimeUpdate	["currentPlayer"]
BuildRequest	["worker"]
BuildResponse	["row","col","worker"]
TryNewCoordinatesRequest	["worker"]
BoardUpdate	["board"]

Connection Manager

The connection manager manages the possible problems of the client connection. A pingRequest message is sent to all connected clients. A timer is started. In the meantime, the clients receive the request and sends a pingResponse as answer. If the answer arrives before that the timer ends, then the connection is working correctly. Otherwise the clients is no more connected and it's deleted from the server.



Message	Attributes
start()	
PingRequest	["Id"]
PingResponse	["Id"]
DeleteClient	