Communication Protocol

Client and server communicate via JSON messages.
The use of JSON messages allows the client to get any kind of implementation, as long as it is respectful of the protocol. Subsequently, the clients has not language constraints and the and serialization/deserialization occurs manually.
Each client request/action sent to the server is followed by a response to each client if it is a broadcast message (e.g. event on the map) or to the original sender (e.g. “illegal action”).
The server sends messages to request setup data and possible actions like move or build.

Sample messages:

Client moving builder from (0,0) to (0,1)
 	
{
 	
“type”: “move”,
 	
“src”: {“i”: 0, “j”: 0 },
 	
“dst”: { “i”: 0, “j”: 1 }
 	
}
	Server notifies all clients (if there was no problem):
	
	{
	“type”: “move”, “name”: “player1”, “src”: { “i”: 0, “j”: 0 },
	
	“dst”: { “i”: 0, “j”: 1 },
	
	“result”: true }

After each move or build event the server sends a message containing the list of the coordinates of every cell in which the current player can move or build during next step. In this way the client knows which options are available. Before sending the move/build request, the model looks at the possible next steps for the player and, if it’s more than one, sends a request to choose between them (move, build or end).

The server will manage any request received. If the action is accepted, the server notifies all the clients in order to set map changes, otherwise only the current client is notified that the action has been refused.

There are two main different phases: setup and game.
Setup phase asks for: number of players, nickname and date, match godCards, single player godCard, color of builders and where to place builders at first for each player.

After the setup phase there is the game phase, where players send to the server actions like move or build until the server replies with an end game, which means that there is a winner or, in the case of a 3 players match, with a lossUpdate to notify the first of them who lost.

In the next pages is shown an example of messages sent during an entire game

Setup number of players

Server → Client

AskNumberOfPlayers {"type":"askNumberOfPlayers"}

Client → Server
SetNumberOfPlayers
{“numberOfPlayers”:"2",type":"setNumberOfPlayers"}

Setup nickname and date

Server → Client

AskNickAndDate (broadcast) {"type":"askNickAndDate"}

Client → Server
AddPlayer

{"date":"1998.04.11","name":"pitty","type":"addPlayer"}

Server → Client

NickAndDate reply

{"result":true,"name":"pitty","type":"playerAdded"}

Setup Cards

Server → Client
MatchGodCards request

{“numberOfPlayers”:”2”, "godDescriptions":{"Atlas":"Your Worker may build a dome at any level”, "Apollo":"Your Worker may move into an opponent Worker space", …}, “type":"chooseMatchGodCards"}

Client → Server
SetMatchGodCards
{“godCardNames”:{“Minotaur”, “Zeus”},”name”: “challengerName”, “type”:”setMatchGodCards"}

Server → Client
MatchGodCards Reply

{”result”:”true”,"godCardNames":{“Minotaur”, “Zeus”},”type":"setMatchGodCards"}

AskGod request
{"godDescriptions":{"Atlas":"Your Worker may build a dome at any level","Zeus":"Your worker may build a block under itself", …},”chosenGodCards”:{},"type":"askGod"}

Client → Server
SetGodCard
{"godCard":"Minotaur","type":"setGodCard"}

Server → Client
Server Reply
{"godCard":"Minotaur","result":true,"name":"pitty","type":"godCardAssigned"}

Setup StartPlayer

Server → Client
StartPlayer request

{“players”:{“player1”, “player2”},”type”:”chooseStartPlayer”}

Client → Server
SetStartPlayer

{"name":"player1","type":"setStartPlayer"}

Server → Client ColorUpdate reply

{"result":true,"name":"player1","type":"setStartPlayer"}

Setup Color

Server → Client
AskColor request

{"chosenColors":[],"type":"askColor"}

Client → Server
ColorUpdate

{"color":"MAGENTA","type":"colorUpdate"}

Server → Client Color
ColorUpdate reply

{"result":true,"color":"MAGENTA","name":"pitty","type":"colorUpdate"}

Setup Builders

Server → Client
AskBuilders request

{"type":"askBuilders"}

Client → Server
BuildersPlacement

{"name":"pitty","positions":[{"i":1,"j":1},
{"i":2,"j":2}],"type":"buildersPlacement"}

Server → Client
BuildersPlacement reply

{"result":true,"name":"pitty","positions":[{"i":1,"j":1},
{"i":2,"j":2}],"type":"buildersPlacement"}

Move

Server → Client
Possible Move destinations

{"possibleDst":[[{"i":0,"j":1},{"i":1,"j":0}],[{"i":3,"j":3},{"i":3,"j":4},
{"i":4,"j":3}]],"type":"possibleMoveDestinations"}

Client → Server
Move

{"dst":{"i":0,"j":1},"src":{"i":0,"j":0},"name":"pitty","type":"move"}

Server → Client
Move reply

{"result":true,"dst":{"i":0,"j":1},"src":
{"i":0,"j":0},"name":"pitty","type":"move"}

Build

Server → Client
Possible Build destinations

{"possibleDst":[[{"i":0,"j":0},{"i":1,"j":2},{"i":0,"j":2},{"i":1,"j":0}],
[{"i":3,"j":3}, {"i":3,"j":4},{"i":4,"j":3}],[],
[]],"type":"possibleBuildDestinations"}

Client → Server
Build

{"dst":{"i":1,"j":2},"src":
{"i":0,"j":1},"buildDome":false,"name":"pitty","type":"build"}

Server → Client
Build reply

{"result":true,"dst":{"i":1,"j":2},"src":
{"i":0,"j":1},"buildDome":false,"name":"pitty","type":"build"}

Ask Step

Server → Client
AskStep

{“type”:"askStep", “possibleSteps”: {“BUILD”, “END"}}

Client → Server
SetStepChoice

{“name”: “Pitty”, “type”:”setStepChoice”, “stepChoice”: “BUILD”}

Server → Client
Possible Build destinations

{"possibleDst":[[{"i":0,"j":1},{"i":1,"j":0}],[{"i":3,"j":3},{"i":3,"j":4},
{"i":4,"j":3}]],"type":"possibleBuildDestinations"}

Turn Update

Server → Client (broadcast)
TurnUpdate

{“name”: “currentPlayer”,“type”:”turnUpdate”)

State Update

Server → Client (broadcast)
StateUpdate

{“state”: “currentState”,“type”:”stateUpdate”)

Loss

Server → Client
LossUpdate

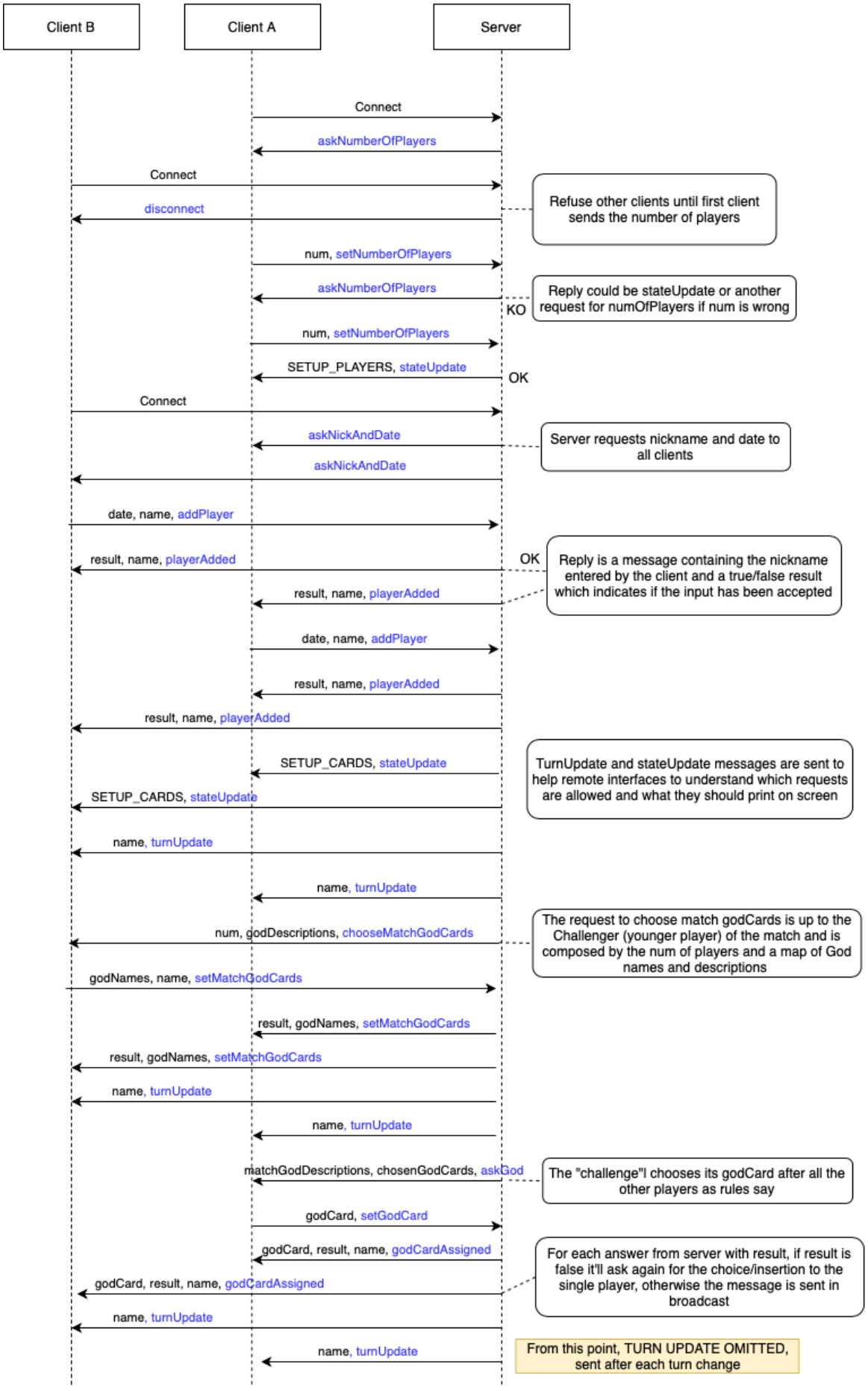
{“type”:”lossUpdate”, “name”: “loserPlayer”)

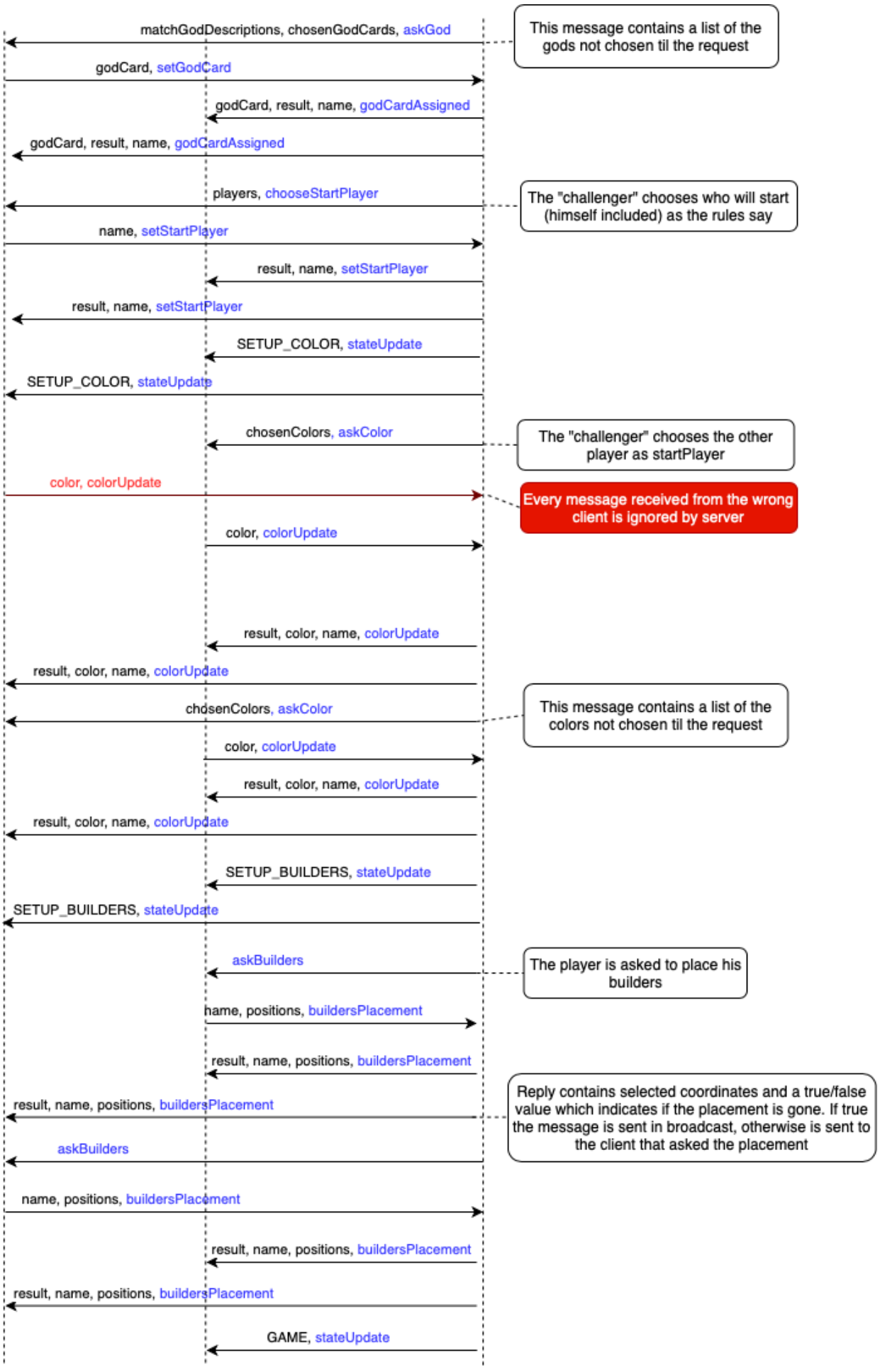
Game Over

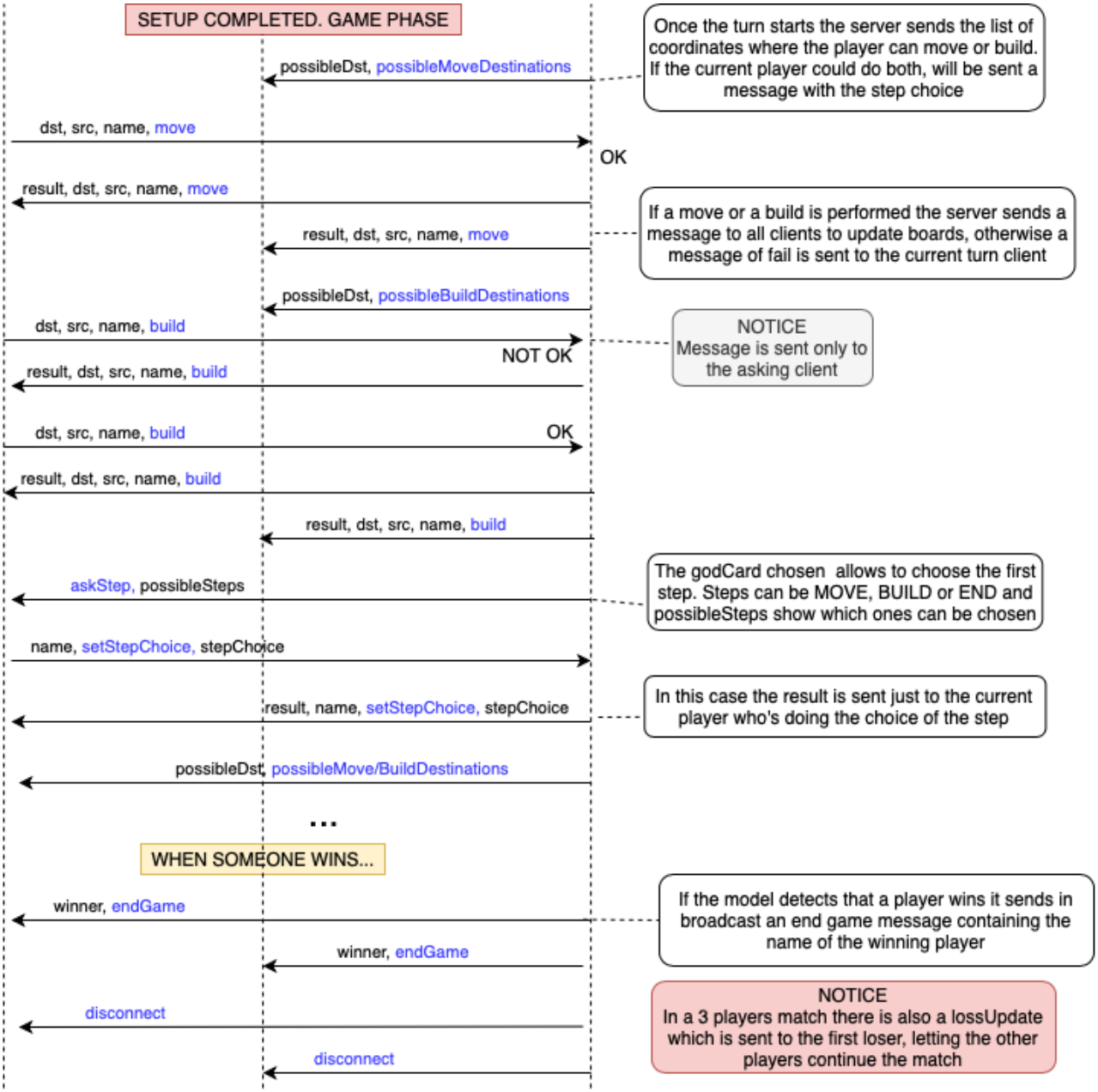
Server → Client
EndGame

{“type”:”endGame”,”winner”:”pitty”}

We suppose that client B is younger than client A, so he'll be the challenger
The blue writings represents the message type




matchGodDescriptions, chosenGodCards, askGod
This message contains a list of the gods not chosen til the request
godCard, setGodCard
godCard, result, name, godCardAssigned
godCard, result, name, godCardAssigned
players, chooseStartPlayer
The "challenger" chooses who will start (himself included) as the rules say
name, setStartPlayer
result, name, setStartPlayer
result, name, setStartPlayer
SETUP_COLOR, stateUpdate
SETUP_COLOR, stateUpdate
chosenColors, askColor
The "challenger" chooses the other player as startPlayer
color, colorUpdate
Every message received from the wrong client is ignored by server
color, colorUpdate
result, color, name, colorUpdate
result, color, name, colorUpdate
chosenColors, askColor
This message contains a list of the colors not chosen til the request
color, colorUpdate
result, color, name, colorUpdate
result, color, name, colorUpdate
SETUP_BUILDERS, stateUpdate
SETUP_BUILDERS, stateUpdate
askBuilders
The player is asked to place his builders
name, positions, buildersPlacement
result, name, positions, buildersPlacement
result, name, positions, buildersPlacement
Reply contains selected coordinates and a true/false value which indicates if the placement is gone. If true the message is sent in broadcast, otherwise is sent to the client that asked the placement
askBuilders
name, positions, buildersPlacement
result, name, positions, buildersPlacement
result, name, positions, buildersPlacement
GAME, stateUpdate


SETUP COMPLETED. GAME PHASE
possibleDst, possibleMoveDestinations
Once the turn starts the server sends the list of coordinates where the player can move or build. If the current player could do both, will be sent a message with the step choice
dst, src, name, move
OK
result, dst, src, name, move
If a move or a build is performed the server sends a message to all clients to update boards, otherwise a message of fail is sent to the current turn client
result, dst, src, name, move
possibleDst, possibleBuildDestinations
dst, src, name, build
NOTICE
Message is sent only to the asking client
NOT OK
result, dst, src, name, build
dst, src, name, build
OK
result, dst, src, name, build
result, dst, src, name, build
askStep, possibleSteps
The godCard chosen allows to choose the first step. Steps can be MOVE, BUILD or END and possibleSteps show which ones can be chosen
name, setStepChoice, stepChoice
result, name, setStepChoice, stepChoice
In this case the result is sent just to the current player who's doing the choice of the step
possibleDst, possibleMove/BuildDestinations
...
WHEN SOMEONE WINS...
winner, endGame
If the model detects that a player wins it sends in broadcast an end game message containing the name of the winning player
winner, endGame
disconnect
NOTICE
In a 3 players match there is also a lossUpdate which is sent to the first loser, letting the other players continue the match
disconnect