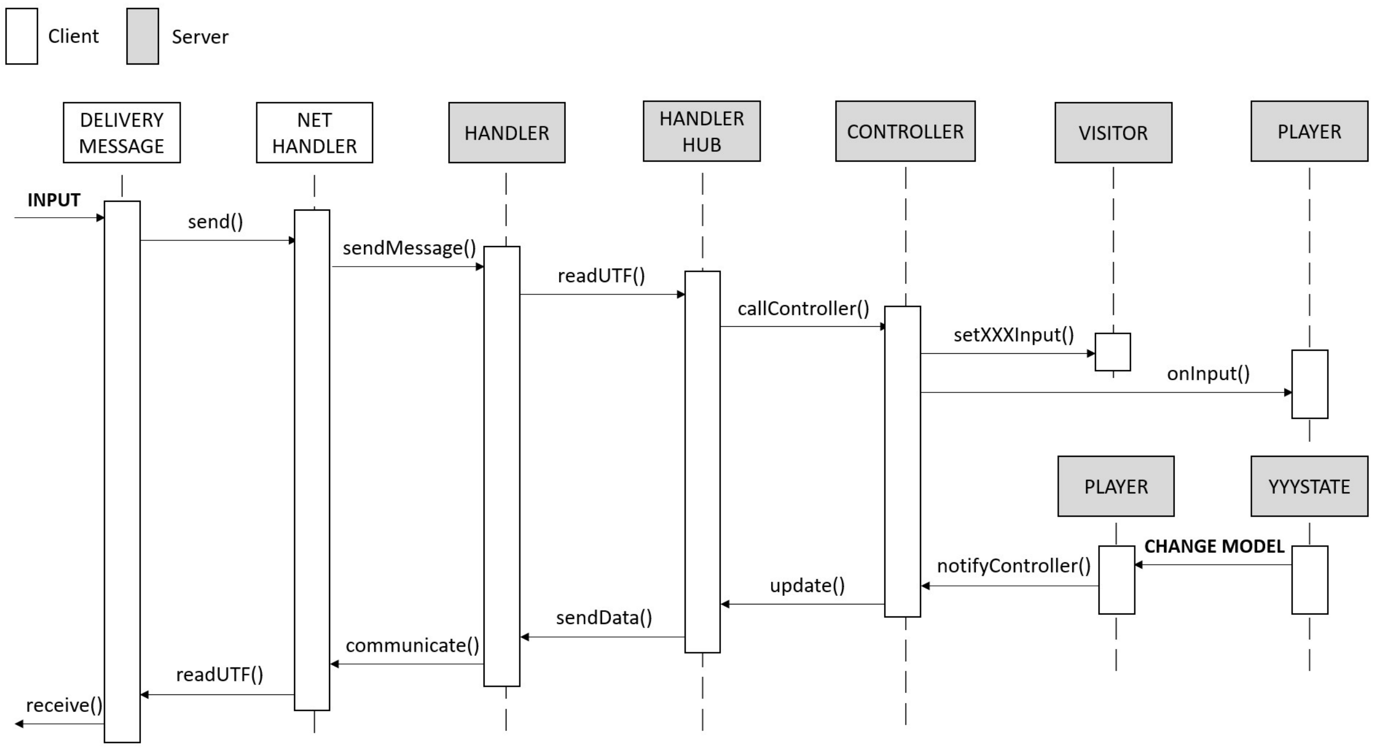
AM13 COMUNICATION PROTOCOL



Once the input is sent by the client, it is parsed in an XML message in DeliveryMessage and then sent to the server through NetHandler. DeliveryMessage insert a special code (between *ACTION_CODE*, *INT_CODE* and *STRING_CODE*) into the message to make easier the deparsing by the server.
Once that Handler's receiving thread receives a message it is sent, through HandlerHub, to the controller that deparses the message and set an input in the visitor and finally calls player.onInput() to execute the action.

When model changes it calls notifyController() in player that calls update() in the controller *. The controller get the LastChange by the player and builds a message. LastChange contains a code (between *UPDATE_TO_PRINT*, *UPDATE_CHOICE*, *UPDATE_GAME_FIELD* and *UPDATE_ENDGAME*) that manage how the message will be created and who will receive the message. The message is then sent to the client(s) through HandlerHub and Handler. If a message is marked as broadcast all the sockets in the HandlerHub will receive the message.
When the NetHandler's receiving thread receives a message it is sent to DeliveryMessage that deparses the message and then sends the info to Field (CLI) or GUI.

ACTION_CODE is used when there's been an action such as MOVE or BUILD (-> visitor.setWorkerSpaceCouple());
INT_CODE is used when the input is an integer, like the number of players in a game (-> visitor.setIntInput());
STRING_CODE is used in all other cases (-> visitor.setStringInput()).

UPDATE_CHOICE is used when a player has to choose, such as a god or a starting position for a worker (NO BROADCAST);
UPDATE_GAME_FIELD is used to send to all players where another player moved or built (BROADCAST);
UPDATE_ENDGAME is used at the end of a game and contains the last message to a player (NO BROADCAST);
UPDATE_TO_PRINT is used in all other cases such as error messages, somebodyHasLost messages… and it's both broadcast and no broadcast.

If the controller, while deparsing the message, finds a special word (between *HELP*, *GOD*, *GOD****, *QUIT* and *WHAT_TO_DO*), the message will be seen as a special request and the response will start from the controller and not from the model.

HELP will give you a list of possible operations you can always require;
GOD will give you every God with associated power;
*GOD**** (where *** is the name of a God) will give the power of the written God;
WHAT_TO_DO will give you the last significant message, which contains indication for what you have to do;
QUIT will finish the game (if the player is in game) or will close the connection with the server only.

*: pattern Observer