FinalProject Peer2Peer Systems and blockchains

Development of a DAPP for Smart Auctions

Lorenzo Bellomo, 531423

Project Structure

The project is organized as follows:

- *Root folder*: It contains only the report file and a folder DAPP, all the code is in this folder.
- *DAPP folder*: It contains the configurations files needed (*bs-config.json* for lite-server, *package.json* for node, *truffle-config.json* for truffle). It additionaly holds file ropsten.json, which contains the private credentials for ropsten deployment. Then other folders are explained just in the next points.
- *contracts, migrations, test and build*: Truffle related folders. They are compulsory when dealing with truffle projects. Also, at this level, the build folder is generated.
- *node_modules and src*: Folders related to node and the front-end part. The *src* folder contains some *.css* files (bootstrap).
- *index.html*: It only contains the view for the top navbar in the web page, and the two buttons used to choose the auction to render.
- *views folder*: It contains *vickrey.html* and *dutch.html*, which are the files needed to render the auctions.
- *js/app.js*: This is the file that contains the logic for the initialization of web3, the rendering of the two auctions, the handling of all the callbacks and manages the interface towards the contracts.

Contracts ChangeLog

This section contains all the modifications applied to the delivered Final Term smart contracts. All the modifications have a motivation assigned.

- *Vickrey - Removed file Util.sol*: the only method that was provided in this file (computation of the nonce and amount hash for the Vickrey auction) has been moved to the file Vickrey auction.
- *Both - Added auctioneer role*: the text of the project explicitely asks for an auctioneer role. The owner decides which account is the auctioneer when he calls the newly added method createAuction. If he does not decide an auctioneer, he becomes the one to take the role. As a note, in the Vickrey auction, the role of the auctioneer doesn't have a clear purpose. The way it will be used during the tests is to take care of the phase switches (at his expenses).
- *Both - Moved start time*: Both auctions, in the provided FinalTerm implementation, started in the moment the constructor was called. This was split in two phases. Phase one starts when the constructor is called, and it is the phase in which the auction is on the blockchain, but is not yet active (note, grace period is considered active). The secondPhase starts when the owner

calls the method createAucion. In this method, the grace period begins and an event is emitted.

- *Dutch - Added auctionPhase*: In order to better implement the previous point, the boolean variable ended has been changed to a phase enum in the Dutch Auction. The phases are NEW, ALIVE and ENDED.
- *Dutch - Changed assert to require*: In order to better respect the require-assert definition, the assert in the constructor has been changed to a require.
- *Both - Added some getters*: In order to improve clarity and respect the previous changes in the contracts, some getters have been added.

Main Project Choices

Below are listed the main project choices, together with an explanation regarding the reasoning behind those ones.

- *One HTML view*: The main choice is the one to collapse the three views (auctioneer, owner and bidder) in one HTML page. Technically this is incorrect (because of course every role should have its own customized web view), but this strongly simplifies testing and gives a broader view of the auction as a whole. This choice was taken mainly due to the fact that the setup code is identical for all the roles.
- *address management*: The first choice (collapsing the html views) strongly implies that it is not possible to fetch the MetaMask address while in initialization phase, and then use the same one for the whole interaction. This means that, for each operation that needs to have an address specified, it has to be fetched from MetaMask, resulting in a (very low) delay. The operations affected are only the main contract ones, as getter methods do not need to specify the address they come from.
- *HTML view organization*: The web index prompts to the choice between the two implemented auctions (Dutch and Vickrey). After deciding which one to view, the web3 initialization process begins, which ends with the rendering of the appropriate view for the two auctions. The web page is organized in three areas:
  - *Top area*: Provides all the operations for the three roles of the auction.
  - *Bottom Left area*: Provides an interface to call some getter methods from the contract and shows the result of both the main methods from the top area and the getters.
  - *Bottom Right area*: Area with some live information about the auction, together with all the events generated from the smart contract.