Peer to Peer Systems and Blockchains
Academic Year 2018/2019

Final Term
Smart Auctions: Dutch, English and Vickery Auctions on the Ethereum blockchain
Deadline 17-06-2019

1 Overview

The project requires to implement two types of auction systems using Ethereum. One is Vickery and the other one should be chosen between the Dutch and the English auction. All these type of auctions are described in the following. A common feature of all the auctions is the "grace period", corresponding to the period when the auction exists, but is not yet active. This period is measured as the number of blocks corresponding to ~ 5 min.

2 Smart Auctions

2.1 Dutch Auction

price descends until some bidder is willing to pay it

A Dutch auction, also called *open-bid descending-price* auction or clock auction, is a type of auction in which the price of the good is initially set to a very high value and then gradually lowered. The first bidder to make a bid instantly wins the good at the current price. There may be a non-zero reserve price representing the minimum price the seller is willing to sell for. The good can never be sold for less than the reserve price, which prevents the auction from being won at a price that is lower than what the good's owner is willing to accept. Consider a Dutch auction starting with the auctioneer asking for 1000 € . If there are no bidders, the auctioneer might then ask for 900 € and continue to lower by 100 € every few moments. Eventually, (say for a price of 500 €), a bidder will accept and obtain the good for this last announced price of 500 €. If the reserve price is 100 € and no one has accepted the offer before, the auction stops.

Implement a Dutch auction using Ethereum. The smart contract takes the following parameters at the time of creation:

- the reserve price (which may be zero)
- the initial price the auctioner asks for
- the number of blocks the auction will be open for, including the block in which the auction is created. That is, the auction starts immediately.

- a contract implementing an interface. The contract contains a method to compute the decrease of the price (for instance the decrease may be linear, exponential and so on with respect to the elapsed time, measured as number of blocks).

Once created, anybody can submit a bid by calling this contract. When a bid is received, the contract calculates the current price by querying the current block number and applying the specified rate of decline in price to the original price. The first bid which sends a quantity greater than or equal to the current price is the winner; the money should be immediately transferred to the seller and the auction contract terminated. Invalid bids should be refunded immediately.

2.2 English Auction with temporary buyout

new bids increase the price until no new bid has been posted for a fixed number of blocks. The good may be bought before the auction starts

An open-bid ascending-price auction, also called an English auction, is the classic auction house format: the auctioneer starts with an initial good price (the reserve price) and asks for an opening bid. Once some individual has placed a bid at or above this price, others are free to offer a higher bid within a short time interval. This is usually where the auctioneer will say "Going once, going twice..." before declaring the good sold to the last bidder.

An English Auction with temporary buyout is an English Auction with a buyout price which is defined by the seller. If the buyer agrees to buy at the buyout price before the auction starts, the good is sold immediately and the auction ends. If a bid comes first, the buyout price disappears and the auction continues as an English auction. Hence, the buyout price is temporary. This type of auction is implemented in eBay's "Buy It Now price" version.

Implement an Ethereum smart contract that supports this kind of auction. In addition to the reserve price interpreted exactly as before, your English auction contract will take 3 additional parameters:

- the number of blocks a bid must be unchallenged before it is considered the winner, including the block in which the bid is posted.
- the minimum bid increment
- the buyout price.

Once created, the contract should accept bids from anybody with a value greater than or equal to the current winning bid plus the minimum bid increment. The initial bid must be the reserve price or higher. When a successful bid is placed, the money should be held by the contract and the value of the previous bid returned to the previous bidder. At the end, when a sufficient number of blocks passes with no new bids, the auction should stop accepting new bids and wait for the *finalize()* function which may be called by the buyer or by the seller to perform the movement of money.

2.3 Vickery Auction with withdrawal

Bidders submit sealed bid commitments and later reveal them. Highest revealed bid wins but pays only the price of the second highest revealed bid. Bidders who don't reveal forfeit a deposit.Bidder can withdrow their bid.

This is called a sealed-bid second-price auction or Vickrey auction. While it is named for economist William Vickrey who first formally described it in 1961 and helped popularize it, this format has been used in practice since at least the 19th century. An offline Vickrey auction proceeds as follows: all participants submit their bid in a sealed envelope. The auctioneer then opens all of the envelopes, and the highest bidder obtains the good but only pays the price specified by the second-highest bidder (hence the term second-price auction). This is the price the highest bidder would have needed to pay (perhaps plus a small increment) to outbid their closest competitor in an English auction. Bidders can withdraw their bid before envelopes are opened.

Implement an Ethereum smart contract for Vickrey auction which takes, as before, a reserve price and also

- a bid commitment phase length measured in blocks.
- a bid withdrawal phase length measured in blocks.
- a bid opening phase length measured in blocks.
- a bid deposit requirement. This deposit is a deterrent against uncorrect behaviours of the bidders, like sending an amount which is lower than their bid.

The auction will have three phases:

- during the bid commitment phase (which lasts commitment phase length) anyone can submit a bid. To preserve secrecy, bidders submit a commitment to their bid, rather than the bid itself. Specifically, this commitment should be a SHA-3 hash of a 32 byte nonce and their bid value (formatted as a 256-bit buffer). Bidders must also send the value specified by the bid deposit requirement. This money is to ensure they submit a well-formed bid and eventually open their bid. Bidders will get their deposit back in the second phase after revealing their bid. Note that in this phase, bidders send actual money as deposit, while they only write in the bid the amount they are going to invest, without actually sending the money.
- during the bid withdrawal phase (which lasts withdrawal phase length) the bidders can withdrawn their bids. In this case they receive one half of the amount of their deposit back.
- in the bid opening phase (which lasts the bid opening length, starting from the block after the last block in which bids can be submitted) all bidders should reveal their bid by sending the nonce used in their bid commitment. They

must also send the precise amount of funds to the contract to pay for their bid if they win. Any attempts to open a bid incorrectly (e.g. by sending an incorrect nonce or failing to send the correct funding value) must be rejected. Every valid bid opening should receive the bid deposit back.

Finally, after the bid opening period has ended (at the specified time), the winner is the entity which sent in the highest bid. The winning price is the second highest bid value which was revealed (or the reserve price if only one valid bid was opened). After the reveal deadline has passed, the $finalize()$ function will need to be called. Of course, if called too early it should not close the auction.

All losing bidders (who successfully opened their bids) need to get their bid amounts refunded (in addition to their bid deposits). It is possible to do this all in one pass at the time the auction closes, or you may optimize by doing this during the bid opening phase. If a higher bid has already been opened, losing bidders can be immediately refunded (or need not send in their actual bid amount at all). Losing bidders must be refunded in any case. The winner likely also needs a refund, since they sent in the full amount of their bid but only pay the amount of the second-highest bid. This should happen when *finalize()* is called.

Note that in practice, the bid deposit should be quite high (on the order of the expected price of the good). If it is too low, the winning bidder might try to bribe the losers not to reveal their bids, lowering the price they ultimately pay. The bid deposit should be high enough that losers are always strongly incentivized to reveal their bids. The contract may be left with surplus funds at the end due to the deposit bids associated to unopened bids or uncorrect fund transfer. You may burn this money, or send it to a designated charity address, but not to the winning bidder: otherwise, they can use it to bribe other bidders not to open their bids.

3 Requirements

The student must:

- write the smart contracts in *Solidity*, and compile/deploy them on *Remix* [1]. Security issues should be taken into account when considering the interactions between the bidder and the seller with the smart contract.
- log important events generated by contracts notifying the outcome of a set of operations. The student may choose which operations to consider.
- provide a list of operations (contracts deployment, transactions sent, etc...), temporally ordered, enabling to run a meaningful simulation of the system.
- provide an estimation of the *gas* consumed by *non trivial functions*, (i.e. not completed by a single contract funcion call) implemented by the system. Use the functionalities of *Remix* for gas evaluation.
- define the parameters of the system and discuss their choice, when critical.

The assignment must be done individually and its deadline is 17 June 2019. The assignment can be submitted even if the mid term has not been submitted/is not sufficient. If the evaluation of both the mid and of this final term will be positive, the student will be relieved from the oral exam. The assignment is not mandatory, if it is not presented, the student will be required to pass the oral exam on the second part of the course. The assignment requires the submission of:

- the code of the smart contracts. Code should be adequately commented.
- a brief report describing the main project choices and the evaluations performed (gas,...).

Submit the assignment through Moodle and bring a paper copy of the relation and of the code at the reception of the Depratment of Computer Science. The evaluation of the assignment will be notified through Moodle. Post doubts/clarification requests on the Moodle page of the course.

References

[1] *Remix - Solidity IDE* https://remix.ethereum.org/.