Auction Smart Contract Report

Pruvendo Team

07/26/21

Executive Summary

This document describes the auction smart contract system developed by Pruvendo. This flexible system supports a wide variety of auctions allowing, at the same time, to implement other scenarios such as a cascaded auction. All the auctions are thoroughly tested and accompanied by the depots that allow to deal with auctions in the interactive mode.

Table of Contents

Source Data

The source code of the smart contracts that have been verified is available at: https://github.com/Pruvendo/freeton-auctions, branch master, commit hash : fad8be512a305e345d8ee59d5c5a622a40b6b0bc

Deployed smart contracts

All the smart contracts were deployed to the developers network with the following address for the root 0:2fff952ec5b83553dc502ac86890b12b56387131181562b086c376714edf4cc0.

Description of the smart contract system

Overview

The proposed solution provides the implementations for the following types of auctions:

- English[1]
- English first price
- English reverse[2]
- First price reverse
- Dutch[3] straight
- Dutch reverse

The first price options for the Dutch auctions are not considered as they are technically equivalent to the corresponding English options. The provided framework provides support for another types of auctions, such as second price or cascaded, but they are not immediately implemented as it was not required in the present contest.

The following types of bids are supported:

- Bids in TONs (native currency)

[1] English auction states the most convenient type of auction where the person that provides some price (usually, highest) wins. The most known example of English auction is Salisbury one
[2] In contrast to the straight auctions where the highest price wins the reverse ones are where the lowest price wins. The simplest example is a tender where the participants offer the lowest price for their service
[3] Dutch auction initially sets a very high price (or low, in case of reverse one) and gradually decreases it. When somebody finds the price applicable he pushes a button and wins an auction. This type of auction is widely used for flower trading (especially, in the Netherlands, the top seller of tulips)

- Bids in TONs bound to DePools[4]
- Bids in TIP-3 tokens

The core part of the framework is accompanied with tests and debots for simpler usage.

High-level technical description

The smart contract system is implemented in Solidity, where tests are implemented using TS4 framework.

The system includes the following key smart contracts:

- *AuctionRoot* - initiates an auction and settles the transfer of tokens from *IGiver* (seller) to *IBid* (bidder). The *AuctionRoot* is moved to a separate entity from the *IAction* entities to allow to implement cascaded auctions in future
- *IAuction* - contains the properties of the specific auction
- *IBid* - contains the properties of a bid
- *IGiver* - contains the properties of a sale proposal

While *IBid* can be either a plain native currency, a TIP-3 token or depooled native currency, the IGiver can be arbitrary (the good example is NFT-token, however its nature is known by the auction).

Business Logic Description

A root contract is deployed by the host of the auction. Once deployed, it can implement Auction contracts with the desired parameters. All the activity can be managed through depots.

Users can deploy sale proposals (Givers) or bids (Receivers). Commonly such proposals will be submitted via depots (Receivers only, the Givers are supposed to be handled manually until their origin is clearly identified).

When the auction's time has run out, the system determines the winner (its meaning depends on a particular kind of an auction). Funds are then transferred to the winner, and, if the winning bidder/Receiver had sent a sum in excess of the final auction price, the excess sum is returned to them. The details for TIP-3 and DePools are provided below.

TIP-3

[4] TONLabs implementation of DePool is used as a reference

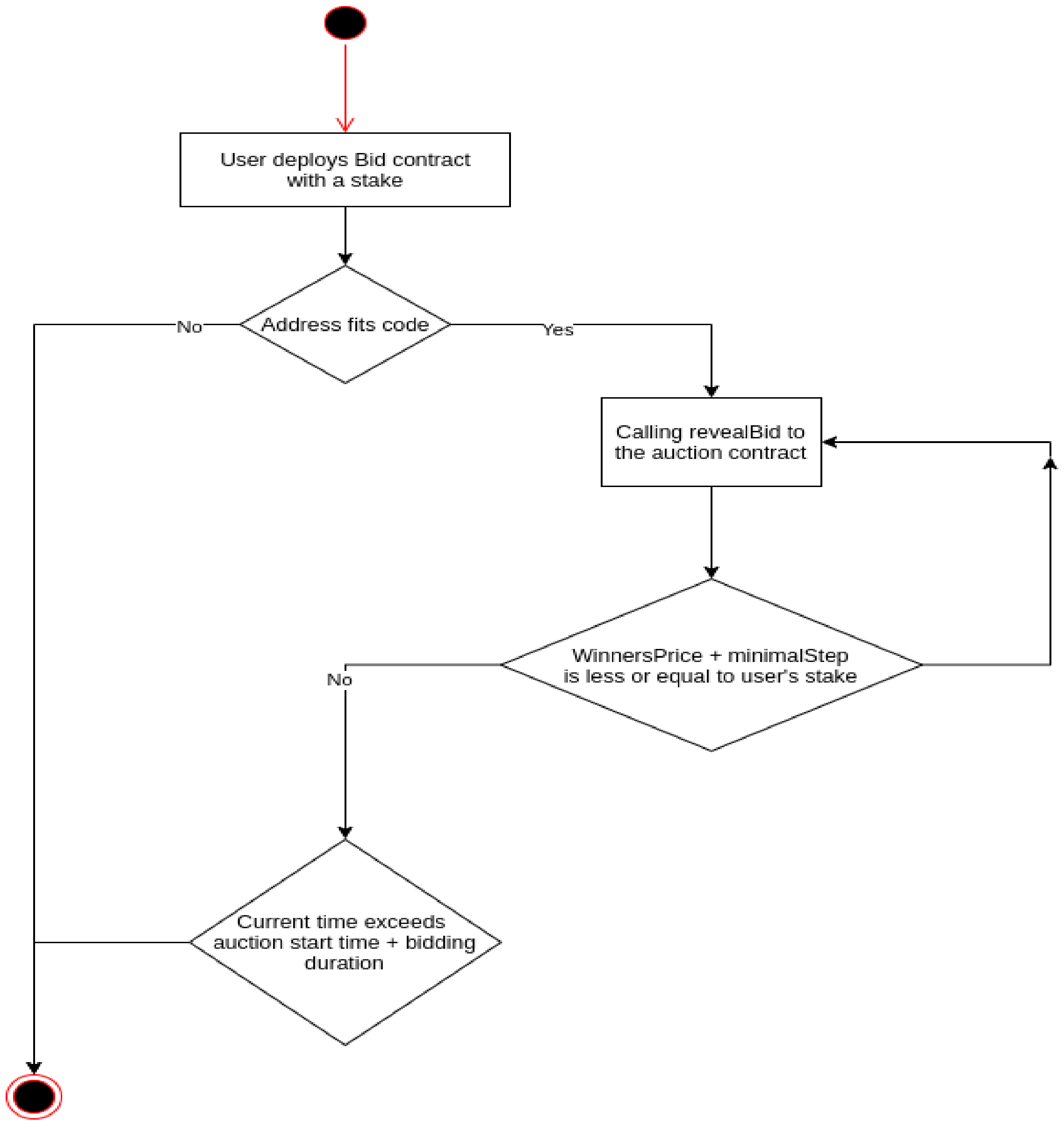
TIP-3 supported by the presented auction system is compatible with Flex TIP-3. This means that, in addition to the basic TIP-3 support, it's assumed that such an operation as lending is supported. So the TIP-3 tokens set as a deposit are not transferred to the *IBid* instance, but rather lent to the bid until the auction is over. This approach is much safer than a direct transfer and eliminates the risk of deposit stalling due to a software bug.

Depool

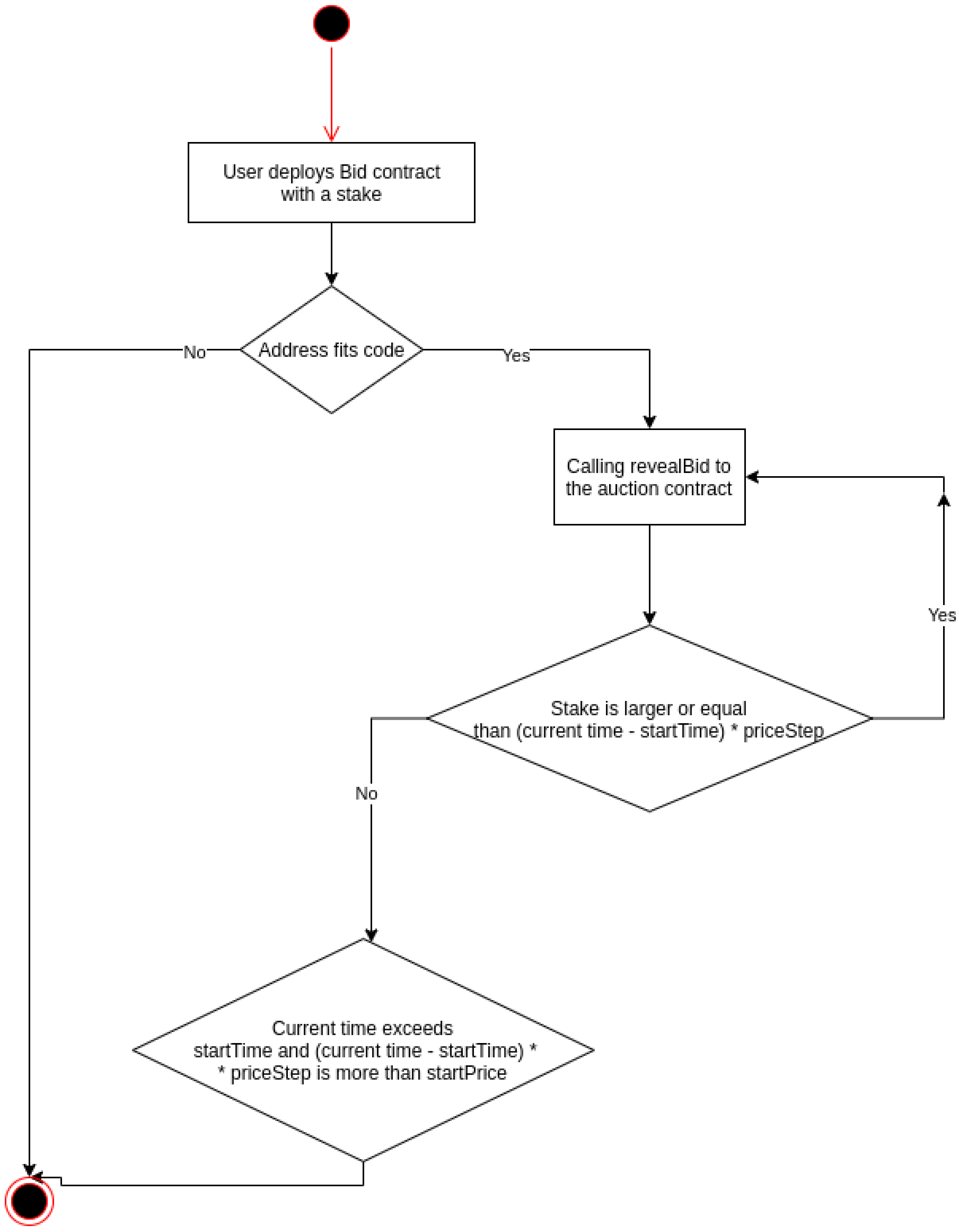
Depool depositing is based on the *transferStake* method. The main trick here is that all the depool deposits can be at work (sent to *Elector*) upon the time of the auction's end. To ensure all the assets are correctly distributed between the giver and receiver, the distribution is postponed until the *onRoundComplete* message is received that indicates that all the money is unlocker and can be freely distributed. This approach looks as know-how as the depooling was never correctly handled in the previous contracts.

UML Diagrams

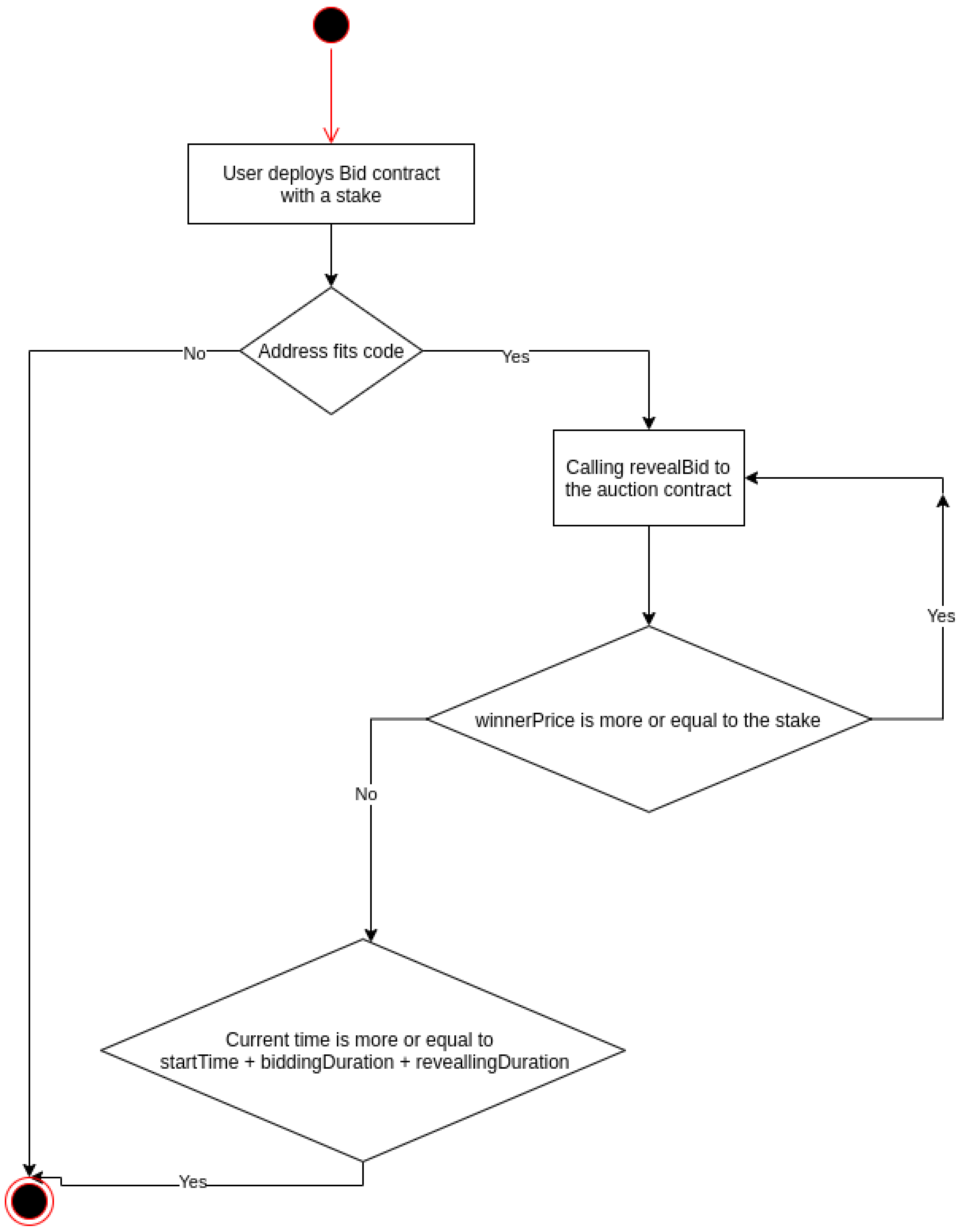
Direct English auction


User deploys Bid contract with a stake
Address fits code
No
Yes
Calling revealBid to the auction contract
WinnersPrice + minimalStep is less or equal to user's stake
No
Current time exceeds auction start time + bidding duration

Direct Dutch auction


User deploys Bid contract with a stake
Address fits code
No
Yes
Calling revealBid to the auction contract
Yes
Stake is larger or equal than (current time - startTime) * priceStep
No
Current time exceeds startTime and (current time - startTime) * * priceStep is more than startPrice

Direct First Price auction


User deploys Bid contract with a stake
Address fits code
No
Yes
Calling revealBid to the auction contract
winnerPrice is more or equal to the stake
Yes
No
Current time is more or equal to startTime + biddingDuration + reveallingDuration
Yes