P2P Systems and Blockchains
Final Project
Academic Year 2019/2020

BitCollect:
A decentralized crowdfunding platform

1 Goal of the Project

The goal of the project is to develop a (set of) smart contract(s) to implement a crowdfunding platform satisfying the requirements listed in the next section. Due to an emergency many institutions and small entities, like organizations and groups of people, decide to open a crowdfunding campaign to contribute. Some of these actors use existing centralized platforms like GoFundMe or banks; others use cryptocurrency networks. In the latter case the actors have the possibility to deploy a smart contract that allows them to collect donations and, as soon the campaign is over, send these funds to the beneficiaries. A crowdfunding based on a smart contract benefits from transparency and immutability. The bytecode of the smart contracts can be easily visualized and, if the smart contract code is available, it is possible to prove its authenticity by comparing the compilation result with the one stored on the blockchain. Moreover, any user can check if the amount collected by the smart contract has been sent to the correct recipients. The smart contract code cannot be modified in a second moment, therefore granting to the campaign a safety measure.

2 Requirements

The crowdfunding campaign has three main actors: the **organizers**, those who open the campaign; the **donors**, those who donate to the campaign; the **beneficiaries**, the receivers of the funds collected during the campaign.
The crowdfunding smart contracts need to follow specific requirements. Table 1 lists the requirements that MUST be implemented by the smart contracts; Table 2 lists the requirements that SHOULD be implemented. Each requirement has an **ID** and a **description**. For each description the subject is the smart contract.

ID	Description
RQ-CREATE	Create a crowdfunding campaign collecting donations from Etherum accounts.
RQ-PARAMS	Specify the organizers, beneficiaries and the campaign deadline. Both the number of organizers and that of beneficiaries is greater or equal than 1.
RQ-INIT	Collect an initial donation from each organizer before the campaign starts.
RQ-DONS	Accept a donation from any Ethereum account.
RQ-DONS2	Allow a donor to choose how to distribute their donation among the beneficiaries
RQ-HIST	Keep the history of the donations for each donor.
RQ-REJECT	Reject any donation after the deadline.
RQ-WITHDR	Allow the beneficiaries to withdraw their expected amounts after the deadline of the campaign has been reached.
RQ-CLOSE	Allow the organizers to deactivate the contract after all of its funds have been withdrawn.
RQ-EVENT	Log any important activity that has been executed.

Table 1: Must requirements

Clarifications for Requirements in Table 1:

- RQ-CREATE: As soon as the smart contract has been deployed, no new organizers nor beneficiaries can be included.
- RQ-INIT: This means that opening a campaign is not free for the organizers. The organizers choose the amount to donate.
- RQ-DONS2: Assuming a user donates 1 ether, he/she can choose to devolve 0.7 ethers to beneficiary 1, 0.3 ethers to beneficiary 2 and 0.0 to beneficiary 3. (`Humble-Bundle` like)
- RQ-CLOSE: Needs to adapt to the chosen "Should" requirements.
- RQ-EVENT: This holds also for the "Should" requirements.

Clarifications for Requirements in Table 2:

- RQ-REWARD: Choose any model suits you to represent the prizes (strings, flags etc..).
- RQ-REPORT: Reporting a campaign is not free to avoid abuses. You choose how many users and how much they need to invest in order to declare a campaign as "fraudulent".

ID	Description
RQ-REWARD	Support a rewarding system. The organizers set a series of rewards each paired to an amount of ether. A donor donating a certain amount of ether X can claim all the rewards that have paired an amount less or equal to X. (`Kickstarter` like)
RQ-REPORT	Support a reporting system. A user can report an hypothetical fraud campaign by investing ether. If a certain number of users report the campaign as fraudulent, the smart contract stops accepting new donations, and all the donors (and the reporters as well) are allowed to withdraw the ether donated so far; the initial donations of the organizers (RQ-INIT) will be subdivided equally among all the reporters. Otherwise, if the campaign terminates successfully all the ether collected by the reports will be subdivided equally among all the beneficiaries.
RQ-MILEST	Support a milestone system. The creators of the campaign set a series of milestones each characterized by an amount of ether that the campaign can reach during its life. Every time the campaign reaches a milestone the deadline is prolonged, and the smart contract is able to withdraw from a third-party smart contract an amount as a reward for its success.
RQ-FREE	A free of choice functionality.

Table 2: Should requirements

- RQ-MILEST: The third-party smart contract may be a service rewarding successful campaigns.
- RQ-FREE: You can think and implement any kind feature you think it is interesting to have in a crowdfunding platform. This should not be too simple nor too complicated.

3 Tasks

The student is asked to implement the mentioned smart contract. It can be a single Solidity smart contract, or more than one if required. The tasks are the following:

1. The smart contracts need to satisfy ALL the "Must" requirements and TWO "Should" requirements.
2. The smart contract should not have known security issues.
3. The student should describe the smart contract(s), its functionalities and should provide an example of workflow with *2 organizers and 3 beneficiaries*. In order to do that the student can either provide a script executing the functions, sequence diagrams or any other way they feel more confident with.
4. The student has to develop, at their choice, *one of the two* following extras:
   (a) A Frontend application with Web3:
      - It can either be a CLI or a GUI interface (student's choice);
      - It should build an interface to all the functionalities provided by the smart contract(s);

   (b) An integration with IPFS:
      - Choose one of the two implemented "Should" requirements. Add extra data to fulfill that requirement that it is too bulky to store on a smart contract (e.g. if you choose the RQ-REWARD requirement, then a reward may contain textual information and an image);
      - Store this data on IPFS instead;
      - Provide commands (CLI or GUI) *only* for the functionality of the chosen requirement such that it stores/fetches data both on/from IPFS and from Ethereum. For instance, if you choose the RQ-REWARD requirement, you can provide two commands: "view_rewards" that shows the list of rewards in that campaign (with all their bulky information), and "claim_reward" that, after the previous smart contract transaction, shows that you have that reward.

4 Project Submission Rules

The project must be developed individually. The material to be submitted for the evaluation is the following one:

- a report (pdf document) describing the main features of the project. The report should include: a brief summary of the project choices and implementation, an evaluation of the gas costs of the smart contracts.
- a pdf document reporting the code of all the smart contracts defined to set up the simulation.

The report and the code must be submitted electronically, through the Moodle. The project will be discussed a week after its submission. The discussion of the project consists in the presentation of a short demo, which can be run on the personal laptop and a general discussion of the choices made in the implementation of the system.

The oral examination (if required) will regard a review of the topics presented in the course. I recall that the oral examination is waived for the the students who have passed the Mid and Final Term.
Do not hesitate to contact us (laura.ricci@unipi.it, andrealisi.12lj@gmail.com)by e-mail, we will fix a meeting in the Meet room of the course.