Decentralized Open Source Sponsorship

share the love for open source projects

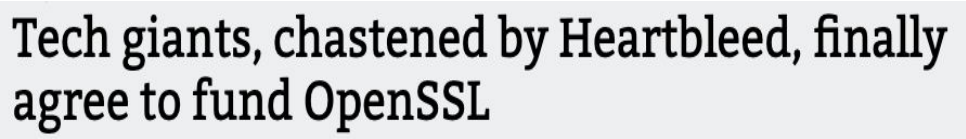
Domain Description / Motivation

Our Interest in the Domain, and Why This Problem Should Be Solved

- The open source community plays a vital role in supporting modern applications and technologies
- Often creators do not get the compensation that they deserve
- This impedes the development of open-sourced code, and reduces accessibility for the general population

Problems and Challenges to Address Within This Domain

- No incentives to keep creating Open-Source Projects
- Project contributions do not flow to major open-source contributors
- Centralized platforms required to manage sponsorships are not cost efficient


Tech giants, chastened by Heartbleed, finally agree to fund OpenSSL

"Open source is not about free software, it is about freedom of software"
- James Turner, Stack Overflow Blog

Mission

A **decentralized service** that allows sponsors to **distribute donations according to the dependency tree** of the open source project, so that projects that get used frequently in other projects will **receive collateral donations proportional to their popularity**.

Requirements

Functional Requirements

- The system **must** process donations via a smart contract
  - Two separate processes: The project sponsor donate funds, and the project owner redeems funds
- The system **must** generate a dependency tree of projects in order to determine donation distribution. Each node contains relevant information such as an address and donation ID
- The system **must** distribute donations according to the dependency tree generated
  - Donors are not able to choose how their donations get distributed
  - In order to achieve fairness and to provide project owners the credit they deserve, the system must follow a strict protocol to distribute donations

Non-Functional Requirements

- **Resource Utilization:** Current platforms take a cut for their service when donations are made. This software *should* provide a decentralized donation service that is more cost-efficient, effective and fair than existing infrastructure
- **Availability**: To provide an equitable service for all open-source project creators, the aim is that this system *should* be accessible and available for all.
- **Integrity**: It is essential that donations are not used for nefarious purposes. The system *must* ensure that donations are safely and securely provided to the project-owners, in the manner by which advertised.

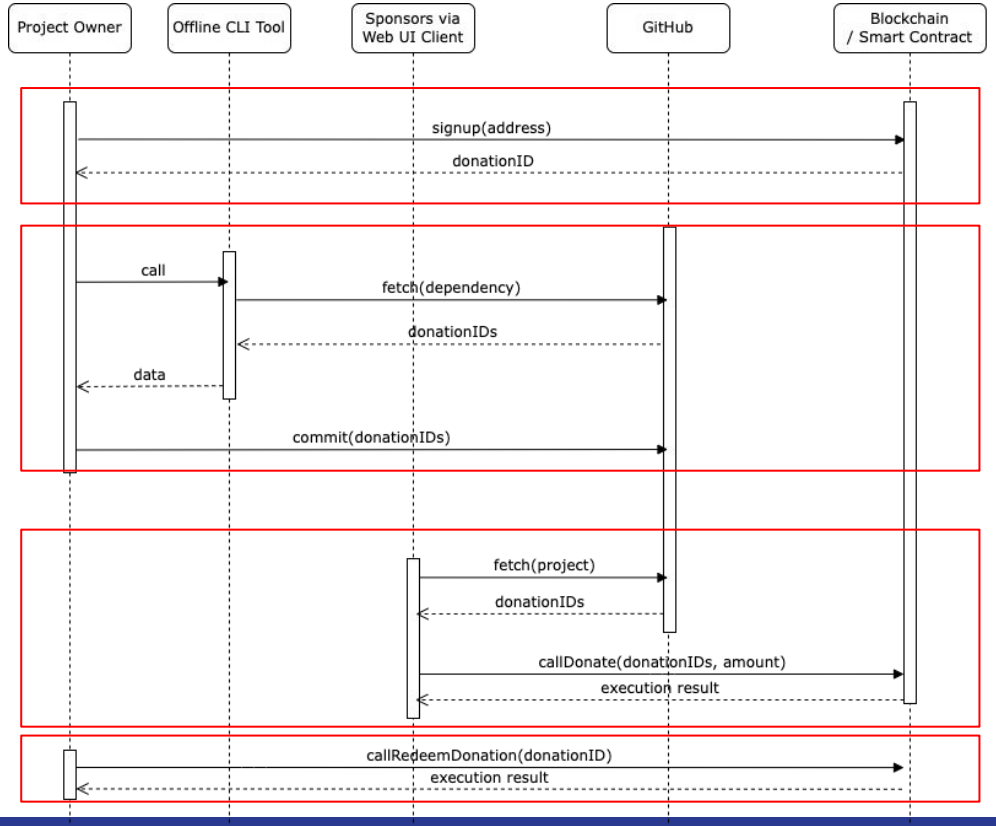
Suitability Analysis

- Multi-party
  - Project owners
  - Project sponsors
- Trusted authority
  - GitHub
  - Payment platforms/banks can be replaced
- Decentralised operation
  - Involve individuals around the world, without a central payment authority
- Immutable, transparent data
  - Auditable transactions
- High performance not required

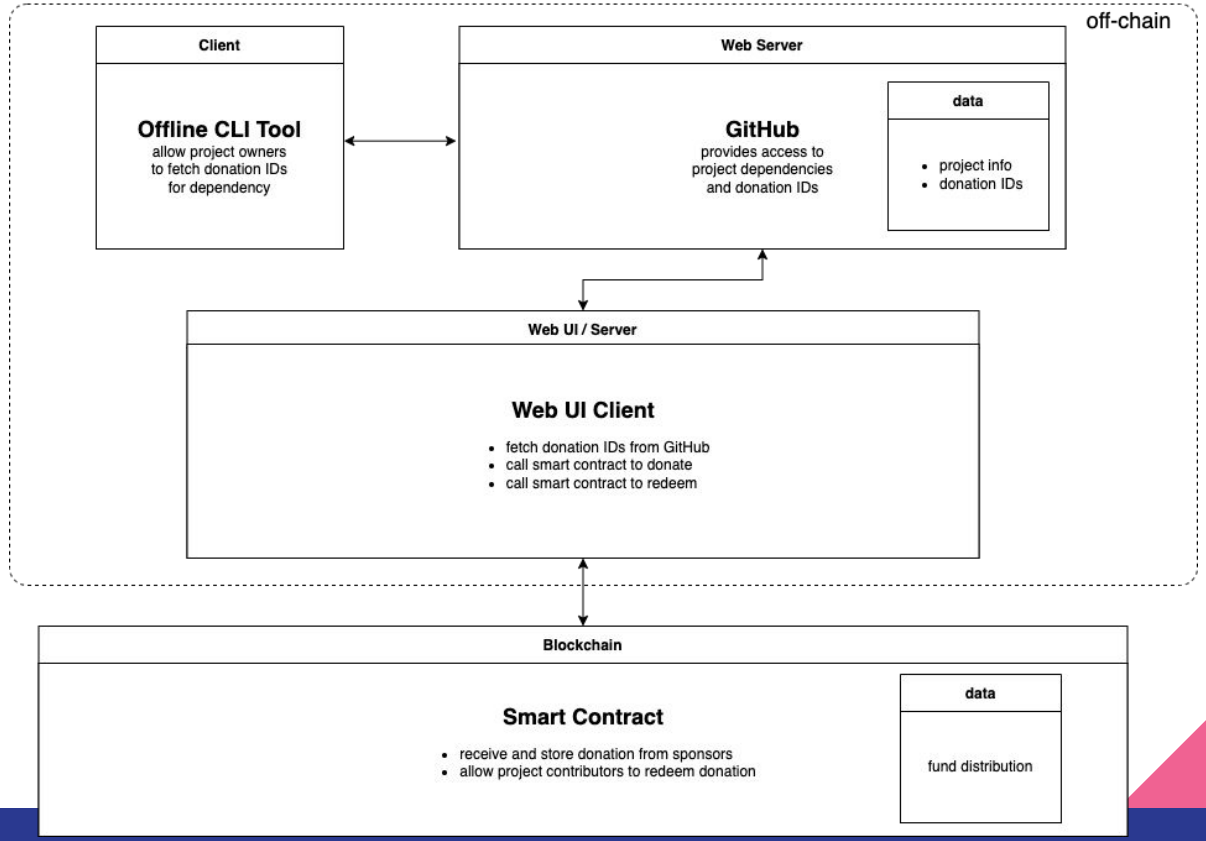
- Permissionless, public blockchain
  - Involves individuals around the world
- Not cross-organization operations
- Smart Contract functionality
  - Distribute donation

ethereum

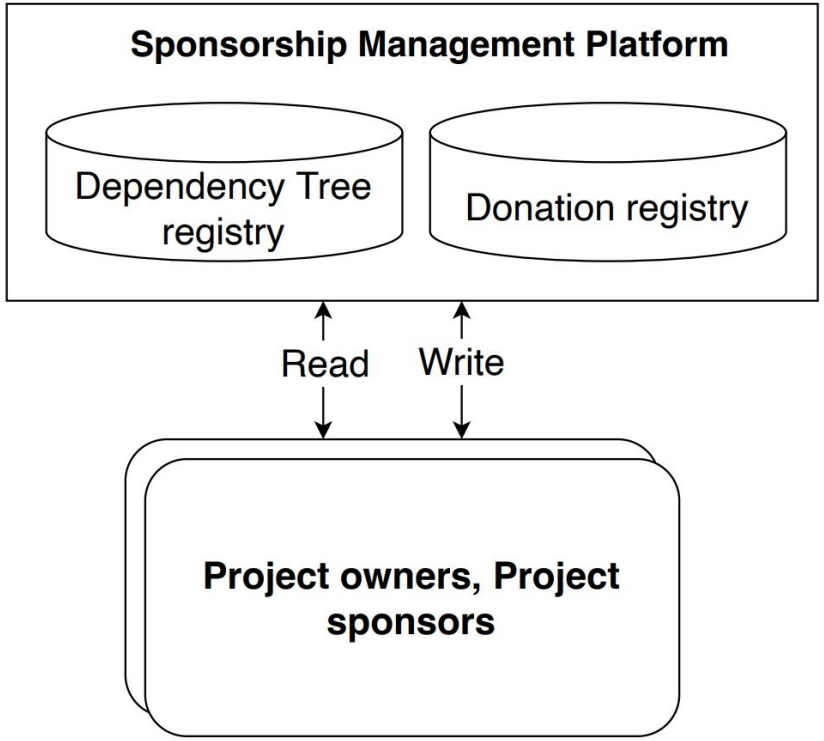
Sequence Diagram



Architecture



Alternative Design



Comparison

	Decentralised Design	Centralised Design
Integrity	- Transactions are validated by all nodes in the blockchain network - All nodes in the network hold a local copy of the blockchain, which defends against data manipulation	- Transactions are indefensible to error and malicious manipulation
Availability	- The use of blockchain increases data redundancy, thus improve data availability - Project owners can more easily retrieve/redeem available fundings - Project sponsors can more easily issue fundings or view retrieve dependency tree	- The platform is hosted in centralised location, which is prone to single point of failure for dependency tree availability for data consumers
Resource utilisation	- Project owners and project sponsors both pay parts of service fee when redeeming or issuing a funding - Less computing resources needed to generate project dependency trees	- Sponsors pay the full amount associated with funding processing

Development Plan

- Week 6
  - Smart contract
  - Offline cli tool
- Week 7
  - Web UI Client
- Week 8
  - Web UI Client
  - Testing
- Week 9
  - Testing
  - Demo for Presentation