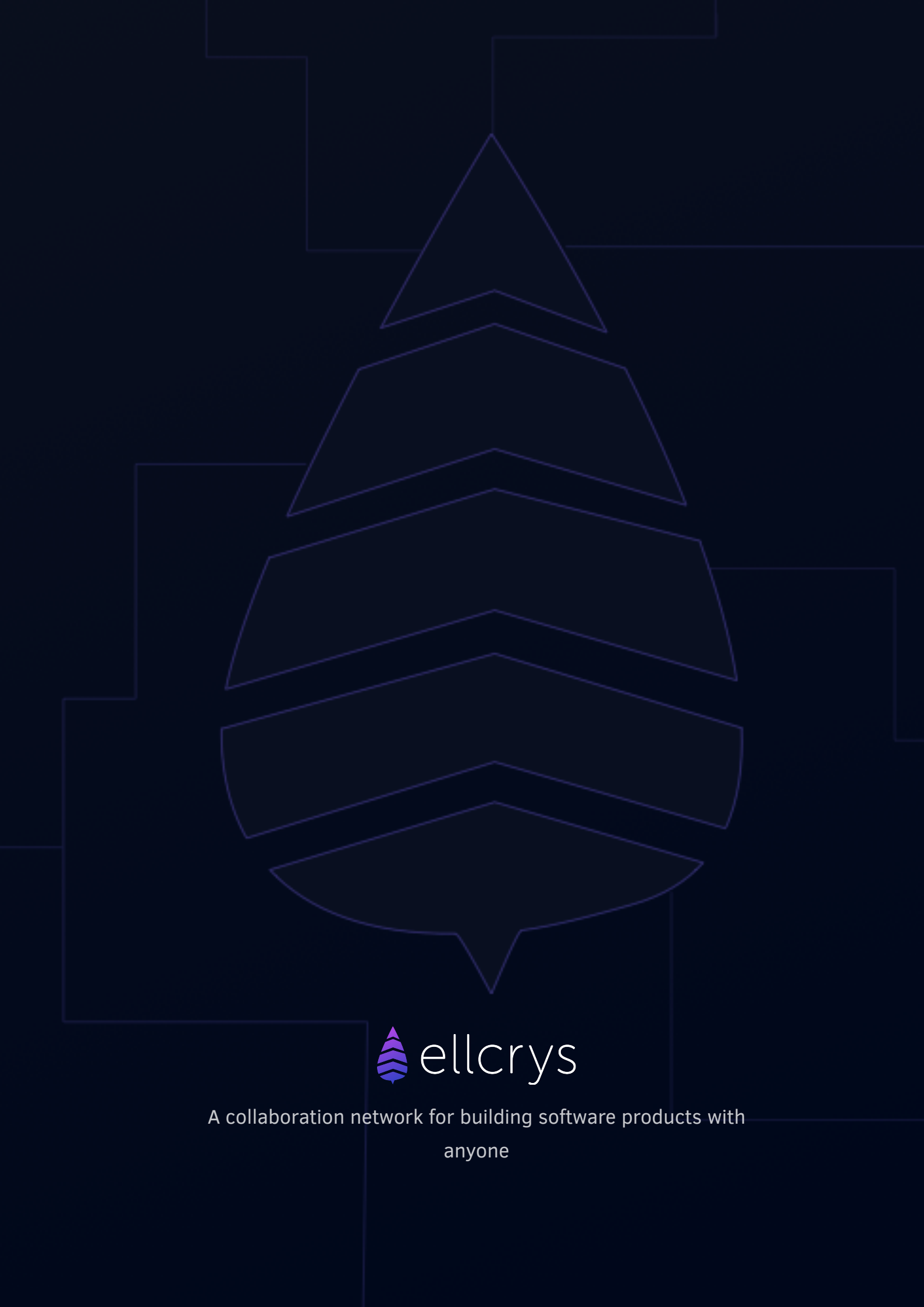


A collaboration network for building software products with anyone

Ellcrys: A blockchain for building software products

Kennedy Idialu

September 7, 2017

v1.1

Abstract

In this paper, we present Ellcrys, a network for creating and managing open, decentralized organizations comprised of globally distributed collaborators working together to build software products and services. Ellcrys offers a blockchain network that integrates Git[1] to allow collaborator build software collectively and from anywhere. Anyone can create or join a project, contribute and get compensated by the network with coins. We refer to this organization as a *community contract*. A community contract is composed of (i) an address for sending and receiving payments and transactions, (ii) a git repository for version control and collaboration, (iii) a proposal and voting system for approving predefined actions put forward by collaborators and (v) a task-based system for distributing work and receiving contributions. Community contracts can contain code for a website or web service, packages, smart contract etc. Ellcrys provides an off-chain deployment protocol to allow community contract run services on centralized cloud hosting providers while denying access to executing processes. The rest of this paper will refer to organizations on the network as "community contract" and basic understanding of blockchain and consensus protocols are assumed.

1 Introduction

In this first part, we will discuss the concept of a community contract, why they are important, their applications and our philosophy.

1.1 What is a Community Contract

A community contract describes an entity that allows one or multiple collaborators build open source organizations offering software products and services such as a website, API, smart contract, packages, frameworks etc. Collaborators push and pull code to and from the network, perform governance functions by creating, approving proposals and receive compensation from the network. A community contract can be deployed to centralized cloud hosting platform if it is a web app/service or compiled and loaded on the network as a smart contract for processing transactions.

1.2 Importance

Ellcrys is on a mission to create an ecosystem where open source collaboration goes beyond offering contributions during free or leisure periods to one where collaborators can build open source enterprises that can be profitable, thus, enabling them to spend more time building great products and improving open source ecosystem. It is our believe that collaborators should be able to work on open source projects full-time, earn and live comfortably. To achieve this, we offer a platform where contributing to open source is natively rewarded and various contribution

models to allow open source contributors create revenue generating and sustainable projects. An ecosystem where open source is profitable to contributors will ultimately improve the quality of contributions and products.

1.3 Philosophy

Ellcrys' goal as a network is not to become another decentralized network with the purpose of making centralisation irrelevant — Ellcrys is not anti-centralisation. Ellcrys leverages on blockchain technology to provide and enforce transparency between collaborators, node owners and the Ellcrys organization. Contributors can collaborate to build products meant for centralized systems or ones meant to be executed decentrally. Ellcrys will actively work with centralized technologies and entities while enabling openness and transparency through blockchain technology.

1.4 Community Contract Governance

Ellcrys' communities need a mechanism to reach agreement on topics or issues surrounding operations. The model employed by Ellcrys is a Proposal/Approval Voting scheme where contributors create proposals and solicit approvals from other contributors. A proposal describes a type of predefined action that must be approved to by $^{2}/_{3}$ of contributors. Ellcrys will support weighted membership where members of a community contract gain more voting powers based on the amount of stake they own.

1.5 Accessibility

Community contracts building software products intended to be accessed from a browser or as a web service may implement and structure their project to be compiled and executed by any centralized cloud hosting platforms that support Ellcrys' buildpacks and host protocol. Ellcrys buildpack will provide interfaces for services running on centralized platforms to have access to official and third-party services. Projects launched with the buildpack will not be directly accessible to developers once deployed.

1.5.1 Ellcrys Host Protocol

With the understanding that hosting a community contract may not be free, Ellcrys Host Protocol provides a mechanism for cloud providers to charge community contracts with the consent of the contract collaborators. The protocol will also support subscriptions-based payment. It is built on the collaborators agreement mechanism that allows members to create and approve a proposal type known as **HostProposal**. Below is a sketch of how the HostProposal looks like:

<table>
<tr><th colspan="2">ELLCRYS HOST PROTOCOL SKETCH</th></tr>
<tr>
<td>On Client (Proposer):

1. Include contract.yaml file in source root and merge to master branch
(a) Create new branch
(b) Create contract.yaml and add config specific to host provider
(c) Commit & Push to network
(d) Create MasterMergeProposal
(e) Send approval vote for MasterMergeProposal

2. Create one or more HostProposal</sub> for each requested service listed in “contract.yaml”
(a) Send approval vote for HostProposal</sub></td>
<td>On Client:
For each MasterMergeProposal proposal:
1. Pull & review proposed branch
2. Send approval vote
3. Checkout master, merge branch (only possible if proposal has 2/3 approval votes) and push.

For each HostProposal proposal:
1. Review proposal
2. Send approval vote</td>
</tr>
<tr>
<td>On Host:
On community contract update:
1. Check contract.yaml for any change to requested services.
2. For every new requested service:
(a) Get HostProposal</sub> from network
(b) if approved, deliver service
(c) Construct charge-proof using HostProposal</sub> and send to network to effect a charge on the community contract balance.</td>
<td></td>
</tr>
</table>

1.6 Contribution Model

There are several ways to contribute work to a community contract. Some contribution type may result in some compensation.

1.6.1 Branch Contribution

Branch contributions are work done on a branch and pushed to the community contract. They may include new features or fix for issues. Anyone can create a branch or push code to a branch. Contributors will receive network compensation for their contributions. The amount of coin will be determined by factors such as the contributors network rank, project ranks, rank of community members involved in reviewing the contribution and more.

1.6.2 Task Contribution

A task is an explicit request for implementation of a new feature. Community contract maintainers create tasks and include compensation to be awarded to anyone who implements the task. Compensation can be a one-time fee or a stake in the community contract. A contributor that has implemented a task must create proposal that includes the branch where their implementation was pushed to. Compensation is sent if the community accepts it.

1.6.3 Issue Contribution

Issues are faults, suggestions or feature requests created by stakeholders (users or contributors) of a community contract. They may carry a reward payable to anyone who fixes the faults or implements a requested feature. Maintainers can earn rewards for fixing issues or implementing requested features. All implementations of an issue are subjected to review by the community before reward is allocated. Issues can be

addressed to everyone or to one or several contributors. Issues can also be “up-voted” with additional reward to increase priority and likelihood of attracting attention.

1.6.4 Paid Branch Contribution

A paid branch contribution describes a branch that attracts a fee to fetch its contents. This can be used by maintainers to sell varying versions of the community contract (or software) for a specific fees. Open source software maintainers will be able to use this feature to offer paid versions of their software. On purchase, a receipt will be issued which allows the branch buyer to fetch the branch in the future without further charge and to use as a proof-of-payment to access off-chain support. Paid branches can be contributed to by multiple contributors with a specific fee sharing formula.

1.6.5 Delayed Branch Contribution

Similar to paid branches except the branch becomes free after a period of time. This contribution type allows teams developing open source software to release their work as a premium product for a limited time. When this period elapses, the branch becomes free for anyone to fetch.

2 Network

In this section, I will be describing the Ellcrys blockchain and our design goals. Please be aware that details here are not final and are very likely to change significantly.

2.1 Distributed Ledger

Existing distributed ledger technologies like Bitcoin & Ethereum perform identical operations and store the same state on every node leading to a system that does not scale. These systems become increasingly inefficient as new participants join the network and the time it takes to process transactions increases. The size of the blockchain also increases; Bitcoin, the leading distributed ledger has a blockchain size of 150GB and grows daily by over 144MB. For a decentralized Git repository

like Ellcrys, where Git objects are stored on the network, it will be catastrophic to implement similar ledger design as the blockchain size will be orders of magnitude larger.

2.2 Sharding

To solve the scaling problem, Ellcrys must research and develop a distributed ledger whose capacity scales horizontally as new participants join the network. This can be achieved through sharding[2]. Each ledger process fractions of total network transactions enabling the network to scale out and process more transactions per seconds as more nodes join the network. Ellcrys will be building on OmniLedger[3], the first distributed ledger architecture that provides "scale-out" transaction processing capacity competitive with centralized payment processing systems such as Visa.

2.3 Consensus Method

To support fast, efficient and more democratic consensus model, we will be leveraging on Delegated Proof of Stake (DPOS)[4]. DPOS relies on approval votes of stakeholders to determine a set of delegates responsible for forging blocks. DPOS was first implemented on BitShare[4] and allows transaction confirmation time of 1 second.

3 Git

In this sections, we describe how Git objects are handled on the blockchain.

3.1 Introduction

Git is one of the most used tools for many collaborators across the world today. It is designed for distributed collaboration, allowing multiple people to work together seamlessly. Interestingly, Git is a content-addressable system which allows files to be addressed by their content and updates handled efficiently without duplications. Ellcrys leverages the content-addressable, snapshot capabilities to index objects on the blockchain and sync with collaborators globally. Additionally, we make use of

the Interplanetary Filesystem (IPFS)[2] to immutably store and access Git objects within the network.

3.2 Bundle Workflow

Git offers several workflows for moving Git objects between collaborators. Two of these include remote and bundle-based workflows. Remote workflow allow users to fetch/push objects to a remote server over HTTP/S, SSH, and Git protocols while the bundle-based workflow allows users to package their work in a binary file and transmit via email, flash drive etc. Ellcrys supports and automate bundle-based workflow. With bundle-based workflow, network nodes do not need to run Git servers. They simply store chucks of bundles and maintain their order.

3.3 Bundle Storage

Git bundle storage add to the storage complexity of the network as each node is expected to store all bundles to ensure high availability. We leverage Interplanetary File System (IPFS)[5] to immutably store and access bundle objects. The blockchain need only store references to the IPFS objects. In future protocol upgrades, we will implement improved replication model that efficiently distributes objects across small groups of nodes.

3.4 Fees

Committing bundles to the blockchain requires the committer to pay a fee to the network. This fee serves as a measure to protect the network from misuse. Additionally, the amount paid as fee is dependent on the size of the bundle. A 1KB sized bundle, could cost 1000 *chakras* (Note: not finalised yet) and 2KB will cost 2 000 *chakras.*

4 Economy

This section describes various economic concepts of the Ellcrys network.

4.1 Coin

There will be an initial supply of 10 000 000 000 (ten billion) coins divisible up to 8 decimal places. A single coin will be referred to as an *"ell"* (plural: "*ellies",* ticker: ELL). It is also represented by the latin small è (unicode = \u0205). The smallest unit (0.00000001) be referred to as *"chakra"*.

4.2 Minting (a.k.a Mining)

Generation of new *ell* will occur through an event referred to as StackMint — StackMint allows anyone in the world to convert national banknotes to the ellies by taking a photo of their banknotes and sending them to the network for processing and validation. This offers a fair distribution model for future generated coins where there is no need for powerful, expensive machine only available to a few. The StackMint mining process involve the collaboration of validator nodes equipped with image analysis capability and users working together to detect and approve valid banknotes. Below is a protocol sketch:

STACKMINE PROTOCOL SKETCH

On Mobile Client (Banknote Owner):

1. Take a photo of any supported banknote
2. Create a ***ConvertBanknote*** transaction and send to network.

On Mobile Client (Client Validator):

On receiving for ***CheckBanknote***:

1. Review banknote image
2. Include result in new *****BanknoteChecked*** transaction.

On Node:

On receiving block:

For each ***ConvertBanknote***:

1. Run banknote through image analyser to detect banknote and construct unique fingerprint.
2. If analysis is successful, Check if fingerprint already exist
   (a). If yes, commit ***InvalidBanknote*** as tx output with appropriate message
   (b) If no, commit ***CheckBanknote*** as tx output

For each ***BanknoteChecked***:

1. If n ***BanknoteChecked*** transactions have been received, calculate the result.
   (a) if majority accepted the banknote, execute **GenCoin**(***BanknoteChecked***) to generate new coin
   (b) Award 15% of generated coins to validator nodes, 15% to client validators and 70% to banknote owner.
   (b) Share losers bond among winning client validators

* ***BanknoteChecked*** transactions requires the sending account to deposit some "Ell" as a *proof of honesty.* In the event that the sending address does not agree with the majority on the validity of a banknote, it will forfeit the bond. The forfeited bond is shared among the winning majority.

4.2 Quantum Token Signature

As of today, on-chain transactions on most distributed ledger are not very fast and can be expensive. Stakeholders of popular blockchains like Ethereum and Bitcoin are studying and developing off-chain transaction capabilities to speed up transaction through-put and reduce cost.

In line with this endeavours, Ellcrys will be implementing an off-chain transaction scheme inspired by the Quantum Tokens for Digital Signatures[6] proposed by Shalev Ben-David and Or Sattath. Digital money based on QTDS require an authority (like a bank). On Ellcrys, the authority is a community contract comprising of a smart contract and a centrally deployed applications. QTDS allows for traceless money transfer through off-chain channels like Email, SMS, QR-Code etc.

A high-level description of the scheme; Alice creates a quantum state by sending some Ellies to the authority (community contract). She associates a new public key or her existing address to the quantum state for transaction verification. Alice can send all or some of the Ellies held in the quantum state to Bob by:

- Constructing a quantum token and sending it to Bob.
- Signing a transaction that permits the authority to give Bob some Ellies.

If Bob gets a quantum token, he immediately exchanges it for a new quantum state/ token associated to his own public & private keys. If Bob gets a signed transaction, he can claim the signed amount of Ellies. Bob can instruct the authority to send his claimed Ellies to his blockchain address. This off-chain scheme can process thousands of transactions per second since a community contract partly implements the authority functionalities as a service hosted on a centralized, trusted hosting service. Users who do not trust the authority can choose not to use the off-chain transaction scheme.

4.3 Universal Node Reward (UNR)

Universal node reward is a standard reward paid to all participants of the network. Every month, the reward is paid to validators based on their level of availability. We measure availability in percentage for a given month. The table below shows the reward:

Uptime (Nines)	Reward (ѐ)	Downtime/month
90%	1	3 days
99%	2	7 hours
99.9%	3	43 minutes
99.99%	4	4 minutes
99.999%	5	25 seconds

Nodes that do not meet the uptime requirements of the network will not be eligible for UNR.

5 Smart Contract

Ellcrys offers an environment for a community contract to execute arbitrary blockchain logic against transactions addressed to it.

5.1 Contract Type

Ellcrys smart contracts are upgradable, versioned codes loaded from a repository. They are repository-addressable; This means originating transactions must reference the community contract as opposed to directly referencing the smart contract. Addressing a smart contract by its community contract address allows collaborators to upgrade it and still be able to access it with the same address. Originating transactions can force a previous version of a contract to be loaded and used by providing a reference to specific versions. We think the ability for smart contracts to be upgradable, especially in situations where a new feature needs to be included or a bug/vulnerability needs to be fixed without breaking existing, dependent services is very useful.

5.2 Code & Execution

Ellcrys aims to allow the use of modern languages such as Javascript, Go, Ruby & Python for writing smart contracts. Contracts are executed in an isolated environment with a capped execution time — This means contracts are expected to

produce an output before the execution time elapses or risk forced termination. Applications may choose to implement expensive operations in an externally hosted community contract or break the operations into multiple transactions. If the execution limit is insufficient, the protocol can be updated. The amount of fee incurred by a transaction will not be determined by computation but storage utilised and execution time.

5.3 Storage

The cost of storage for smart contract data is equivalent to the cost of storing repository objects. Like Ethereum, senders of transactions must include sufficient fees to cover the cost of sending and processing. When excessive fees are sent, the balance will be returned to the sending account.

5.4 Transaction Fee

All transaction fees received by the network are paid to the the Universal Node Reward account. Transaction fees are considered to be the network's earnings.

6 Use Cases

- **Decentralized, Highly Available Git Host:** Ellcrys can serve as an alternative Git host to existing Git hosting services. A decentralized Git repository is relevant for projects that require censorship-resistance and unrestricted access.

- **Open Source Software:** Developers around the world will be able to create open source softwares and tools hosted on a blockchain with the ability to earn revenue from paid or delayed branches, paid issues and tasks.

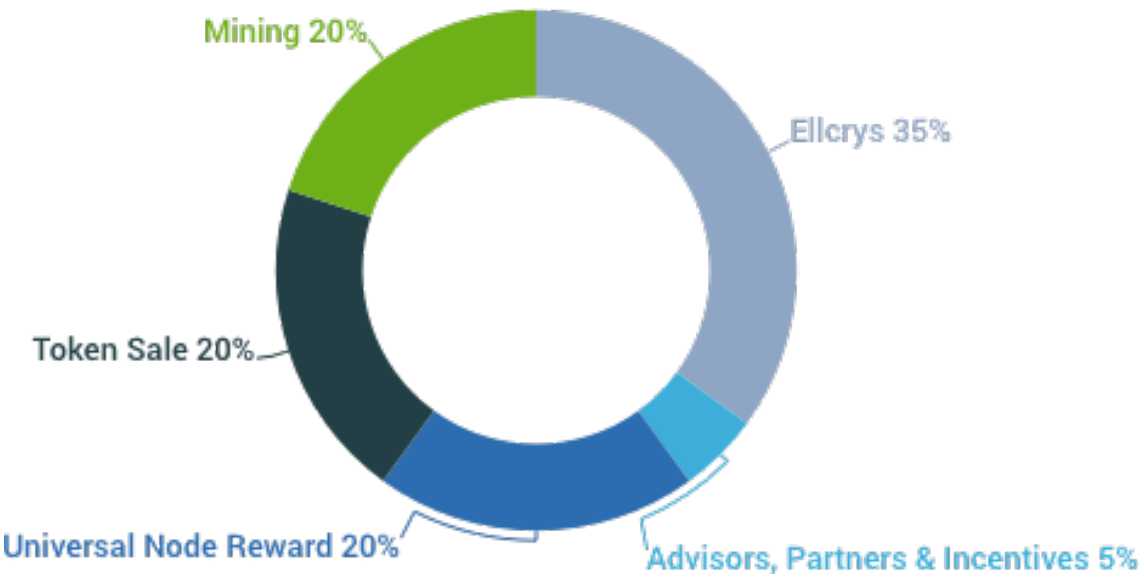
- **IT Outsource:** Companies/Individuals can create community contracts with the purpose of outsourcing tasks to independent developers and designers. Outsourcing parties retain 100% control of their project files and create and set price tasks.

- **Decentralized Startups/Organizations:** Todays internet services require one individual to hold the passwords of an organization and add new employees — This kind of structure makes it hard for people from around the world to come together to build startups that solve real world problems like Uber, Airbnb etc. Ellcrys makes it possible for this kind of multi-owner to be created and managed transparently.

7 Token Sale & Budget Plan

Researching and developing the ideas above will require money. Although, the current team has contributed extensive time and money into research and development of prototypes for various components, we will need additional funding to continue. To this end, we will be hosting several token sale rounds. During the Pre-Sale, tokens will be issued on our own wallet system. After the main token sale, We will issue ERC20 tokens which will be swapped for the main network tokens when it is launched. This will be a 1:1 swap. Below are the details of the tokens distribution, Pre-Sale and Token Sale:

7.1 Distribution



A total of Two Billion (2,000,000,000) tokens will be sold during the Pre-Sale and Token Sale events.

7.2 Pre-Sale

The Pre-Sale will be available to the public and will last for 4 weeks. It is targeted at early backers of the Ellcrys projects and includes a maximum of 30% bonus.

- Pre-Sale Launch Date: February 1st, 2018
- Duration: 4 Weeks
- ELL available: 800,000,000
- Price: $0.08 / ELL

Tokens not sold will be available for purchase in the main token sale. Below is the bonus structure for the Pre-Sale.

- Week 1: 30%
- Week 2: 20%
- Week 3: 10%
- Week 4: 5%

7.3 Token Sale (ICO)

The Pre-Sale will be available to the public and will last for 4 weeks and distribution of tokens will occur 2 weeks after ICO. Purchases will include a bonus of up to 30%.

- Token Sale Launch Date: March 30th, 2018
- Duration: 4 Weeks
- ELL available: 900,000,000
- Price: $0.1 / ELL

Tokens not sold will be equally shared to the mining and universal node reward supply. Below is the bonus structure for the Token Sale.

- Week 1: 30%

- Week 2: 20%
- Week 3: 10%
- Week 4: 5%

7.4 Budget Plan

The Ellcrys team plans to spend the funds raised from the Token sales in the following manner.

- **Product Development and Staffing: 60%**
  This allocation will be used for hiring top talent within the Silicon Valley area and around the world. Our current team is composed of talented software, mobile developers, user experience/design and management personnels. The addition of new talent will greatly strengthen the team and speed up product delivery time.

- **Marketing: 20%**
  Our marketing goal is to grow adoption by reaching developer, designers, makers, testers.

- **Security: 10%**
  The importance of security cannot be overstated. This allocation will go towards hiring and consulting with security experts to investigate our algorithms and systems as well has sponsoring bug/security bounties.

- **Legal: 10%**
  At a time where the legal implications for blockchain technology is unclear, this portion of the budget will be used to hire outside legal counsel to help us figure out all legalities relating to Ellcrys.

8 Conclusion

While our protocol is evolving, we are excited about our mission to create a platform that helps open source collaborators continue to build great software and organisations with the power to sustain themselves.

9 References

[1] Git - https://en.wikipedia.org/wiki/Git

[2] Spanner: Google's Globally Distributed Database - https://dl.acm.org/citation.cfm?doid=2491245

[3] OmniLedger: A Secure, Scale-Out, Decentralized Ledger - https://eprint.iacr.org/2017/406.pdf

[4] Delegated Proof-of-Stake Consensus - https://bitshares.org/technology/delegated-proof-of-stake-consensus/

[5] InterPlanetary File System - https://ipfs.io

[6] Quantum Tokens for Digital Signatures - https://arxiv.org/abs/1609.09047