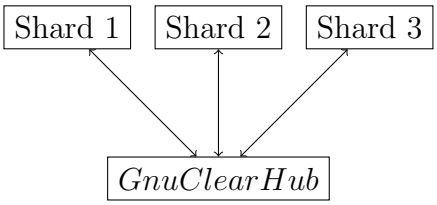
Struct *Node* **contains**

```
    Node* parent;
    int size;
    // other stuff
```

end



Ecclesia: A simple approach to self-organized public infrastructure

May 20, 2016

1 Introduction

The combined success of the open-source ecosystem, of decentralized file-sharing, and of public cryptocurrencies, has inspired an understanding that decentralized internet protocols can be used to radically improve socioeconomic infrastructure. Such protocols provide a means of participation which is accessible, meaningful, and rewarding, and a resulting service which is accountable and secure against malicious behaviour.

To date, however, networks implementing such protocols suffer from a number of drawbacks, including their gross inefficiency, poor performance and scalability, difficulty to work with, and the precarious legal condition of their access tokens.

Here, we present Ecclesia, a public network which simultanesouly addresses all of these problems. Ecclesia uses Tendermint, which provides a high-performance, safe, secure consensus engine that is easy to scale, and where strict accountability guarantees hold over the behaviour of malicious actors. At the application level, Ecclesia hosts a simple cryptocurrency and a formal governance mechanism enabling the network to adapt and upgrade. The economics are designed with organic growth and sustainability in mind, and specifically avoids a single, large, upfront crowdsale.

We intend for Ecclesia to start simple, and to upgrade and/or fork over time according to the will and needs of its users. That said, we conclude the paper with a potential roadmap.

2 Tendermint

In this section we describe the core consensus protocol, how it can be extended to an atomic broadcast protocol, and how to build applications that utilize the atomic broadcast.

2.1 Consensus

A fault tolerant consensus protocol enables a set of non-faulty processes to eventually agree on a value proposed by at least one of them. The problem is made more difficult by asynchronous network conditions, wherein messages may have arbitrarily long delays, and by Byzantine faults, wherein processes may exhibit arbitrary, possibly malicious, behavior. In particular, it is well known that deterministic consensus in asynchronous networks in impossible [**flp**], and that consensus protocols can tolerate strictly fewer Byzantine faults than crash faults ($1/3$ of processes, vs. $1/2$). The former results from the inability to distinguish crash failures from asynchronous message delay. The latter from the fact that three processes are not enough for a safe majority vote if one of them can lie (you need at least four).

In addition to providing optimal fault tolerance, a well designed consensus protocol should provide additional guarantees in the event that the tolerance capacity is exceeded and the consensus fails. This is especially necessary in public economic systems, where Byzantine behavior can have substantial financial reward. The most important such guarantee is a form of *accountability*, where the processes that caused the consensus to fail can be identified and punished according to the rules of the protocol, or, possibly, the legal system.

Tendermint is a mostly asynchronous, optimally Byzantine consensus protocol, notable for its simplicity, performance, and accountability. We say mostly asynchronous because Tendermint employs a deterministic leader-based proposal mechanism, thus incurring a weak synchrony assumption as faulty leaders must be detected and skipped.

security, accountability

2.2 Atomic Broadcast

consensus to atomic broadcast. why blocks. blockchain (this should be the only time we say blockchain!)

2.3 Application Interface

tmsp

3 Ecclesia

The previous section described the core Tendermint algorithm as it might be used in a traditional setting, to power, for instance, an internet service offered by a corporation. Using an atomic broadcast protocol in a public setting, however, requires additional considerations, most particularly with regard to spam prevention, incentive alignment, and governance. In a private consensus environment, access is easily permissioned and rate-limitted, and incentives are aligned by the formal structure of the legal entity maintaining the network (ie. a corporation). In a public setting, the only known way to manage spam and align incentives is with the introduction of an explicit economics.

In this section, we describe the structure of the Ecclesia application, which consists of two core modules: *basecoin*, a generic multi-asset cryptocurrency; and *governmint*, a general purpose governance system that additionally enables collective administration of the validator set and of software upgrades. Validators are required to post security deposits before they are eligible to be a validator, such that if they are found to be malicious, their security deposit can be destroyed, or otherwise re-allocated according to governmint.

3.1 Basecoin

describe basecoin

3.2 Governmint

voting with tokens changing validator sets upgrades forks

4 Distribution

problem/importance of distribution trouble with crowdsales training, etc. inflation

5 Road Map

shards, pegs, crypto integrations scalability

6 Conclusion