Semantic Interpretation

Solution to the codebase consensus conflict in context of the Bismuth Blockchain

Jan Kučera, July 19 2018

The Problem

In 2017, a new type of a 51% attack became prolific. It was not of technical, but of governance (political) nature. The resilience of cryptocurrencies is supposed to lie within decentralization of the network, but the code repository is a single point of failure which became the target for this new type of attack - forking and network hijacking, a type of consensus rule attack.

Ethereum proved that it is possible to fork a cryptocurrency network, arbitrarily change its contents and win majority of the participants through a public campaign. This opened the door to a new way of thinking - that it is no longer required to fork a codebase to start your own network, but instead it is possible and profitable to hijack an existing network and take over its participants by promotion. Once the forked codebase wins the majority, it is de facto the winner.

Existing Proposals

A few solutions have been proposed by new projects, most prominent of them is Tezos. Tezos markets itself as the self-amending ledger, where participants can vote on proposals that are eventually integrated into the system. This is an improvement over the existing situation, which in my opinion does not address the problem in an ideal way.

Semantic Interpretation Solution

I hereby introduce the semantic interpretation approach to the singular codebase problem. In this model, the core system remains untouched and is governed by consensus. The semantic interpretation level exists above the core layer.

There, both the secondary layer codebase storage and the way it is interpreted are subject to users. The main advantage is that it can endlessly fork, users can pick which code authors they want to express without influencing others and mainly without having any impact on the primary level codebase. The main consensus remains the same and the semantic layer is not bound by shared state or enforced using traditional consensus rules.

Implementation of Secondary Layer

Level 1: Passive Data

For a system like this, we need a blockchain which enables storage of arbitrary data. The data do not represent a contract of any type, only passive input. This is the first level of interpretation as all users can store data. They can be referred to as service providers.

Level 2: Active Interpretation Engines

On the side of the individual users, we need code that interprets the passive data and turns it into meaningful output. An application which analyzes blocks, looking for code the user is interested in. Service providers can distribute such applications, doing this task for the users, so they reach the desired level of communication between users and the service. Interpretation engines can then store output on the blockchain again.

This output can be recycled as passive data if desired. Interpretation engines can be thought of as individual contract executors and can be run by **one or many** nodes, depending on design and the need for particular **sub-consensus**.

Level 3: Conflict Resolution

Level 1 conflict is the basis for this system to work. Users do not strive for consensus, instead they choose which services they individually want to use, just like on the internet they pick which websites they want to visit. Therefore, nobody has any power over which data anyone else has to use or execute. They use the shared blockchain database for common storage and availability.

Level 2 conflict is similar to a standard blockchain fork. If someone disagrees with an interpretation engine of of a provider and that engine is available openly, they can change it and offer it for the users, becoming a provider of a new solution. Both engines can continue running on the underlying blockchain without any security risks, even if users of a particular engine drop to 0 for any length of time.

Practical Example

This scenario deals with a one node example for simplicity. Many node scenario includes handling of user resources whose interpretation does not require shared state consensus, such as tokens.

Part 1: Service Provider Emerges

User with address A decides to become a service provider of a newspaper. He starts a service by submitting news to the blockchain on daily basis, leveraging the availability advantages of such a solution.

Part 2: Data Storage and Interpretation Engine Creation

The provider stores data to the blockchain in a particular form. In the interpretation process, the main trigger is his **address** and a specific command in the **operation** field. The news data is stored in the **data** field. He also creates an application which reads the previously stored data.

Data for Interpretation Engine 1:
address: 123, operation: newspaper, data: banks bailout in 2019

Interpretation Engine 1 pseudocode:
Select all data from blockchain where the address is 123, operation is "newspaper" and block time ranges between last monday and today. Display data from data field to the user using a web interface available through the browser.

Part 3: Choice as Consensus Conflict

Users A and B decide to download the application and use it. Users C and D ignore it, governance consensus is already split into two groups, one deeming the app worthy, the other one not. If the application was run from a traditional smart contract, this would result in a disruption of the consensus. No such scenario occurs with the Interpretation Engine.

After some time, a new provider decides to copy the Interpretation Engine and issue news in a different language. They change the source address to their own, find new users for it.
address: 234, operation: newspaper_cz, data: finanční pomoc bankám v r. 2019 (Interpretation Engine 2)

A third provider chooses to edit this Interpretation Engine to only read news on Monday, utilizing the first data source, but changing the application.
address: 123, operation: newspaper, data: banks bailout in 2019 (Interpretation Engine 3)

Summary

Interpretation Engines provide a viable solution to the governance problem and consensus centralization in the world of blockchain. Instead of looking for consensus, they intentionally break it without any consequences for the core levels where shared state is critical for operation. It is possible to use them to transfer data and value, run them from one or many nodes, introduce advanced trigger solutions or use one of the existing Bismuth applications as a guideline.

Bibliography

Goodman, L.M (2014). Tezos Whitepaper. https://tezos.com/static/papers/white_paper.pdf
2018 bitcoincash.org. Bitcoin Cash. https://www.bitcoincash.org