Miner Signalled Consensus Upgrade

Aeternity Crypto Foundation

Juraj Hlísta

October 23, 2019

1 Abstract

This paper discusses what blockchain hard and soft forks are and why they are important. The paper describes how consensus protocol upgrades of the æternity network have been done and how we plan to upgrade the blockchain in the future. The focus of this paper is *miner signalled consensus upgrade*. We describe the motivation, challenges and possible implementations of this mechanism and provide a summary of what has been achieved.

2 Introduction

All the nodes that form a decentralized network need to follow certain rules used for validation of transactions and blocks in order to come an agreement on what the shared state of the blockchain is. These rules can be referred to as a *consensus* [24]. All the network participants following the same consensus results in a single blockchain. A fork can occur when a part of the network starts to follow a different set of rules while the other part of the network stays with the original consensus. There are several types of forks [25, 27]:

- *accidental forks* do not change consensus rules;
- *soft forks* tighten or add new consensus rules;
- *hard forks* loosen or eliminate the existing consensus rules.

2.1 Accidental forks

This type of fork can be quite common depending on the parameters of a blockchain. In case of a Proof-of-Work type of blockchain, the fork happens when multiple miners find a new block at nearly the same time. The chain that becomes longer, wins, and the shorter chain forks die off. The forks are more common when the average time between producing two blocks (*block time*) is shorter than it takes to propagate a new block within the whole network. The æternity blockchain uses Bitcoin-NG [28, 3] where the key block time is 3 minutes and the micro block time is 3 seconds. Compared to Bitcoin with a 10-minute block time [24], the accidental forks in the æternity blockchain are more common.

2.2 Soft forks

As opposed to the accidental forks, soft forks are permanent. It is a way of upgrading a blockchain that is backward compatible. It means that an old version of a node can still accept blocks produced by a newer version since all the validation pass. The blocks on a soft-forked blockchain, produced by nodes with a new version, follow both the old and the new consensus rules. In order to upgrade

the blockchain using a soft fork, not all the nodes are required to switch to the new version of a node. This prevents the blockchain from diverging [25].

There can be different kinds of soft forks [22]. For example, the Bitcoin network has used:

- *Miner activated soft fork* (MASF) [27] which relies on support from the miners to enforce the new consensus protocol;
- *User activated soft fork* (UASF) [26, 27] is mostly driven by the users that update their nodes to support the fork. The miners are generally supposed to follow the users in updating their nodes.

There has been no soft forks on the æternity blockchain.

2.3 Hard forks

Hard forks change the consensus rules in a way which is not backward compatible. An old version of a node will not be able to follow a new consensus protocol as it will reject blocks produced by new version of a node. All network participants are required to upgrade their nodes to the latest version, so they can follow the new consensus protocol. If the majority of the network supports the hard fork, but there are still some participants running the old version of a node, they end up on a chain fork, which typically will become abandoned [25, 27].

There are two types of hard forks (in Proof-of-Work type of blockchains) [25]:

- *Unconditional (planned) hard forks*;
- *Conditional (contentious) hard forks.*

2.3.1 Unconditional (planned) hard forks

This type of hard fork is planned and well communicated in advance. It includes consensus breaking features that the vast majority of developers and the community agree with [25].

The æternity blockchain has gone through a couple of scheduled hard forks:

- Minerva;
- Fortuna;
- Lima.

The Lima hard fork is the last planned hard fork on the æternity blockchain.

2.3.2 Conditional (contentious) hard forks

There might be a disagreement among the users, developers or the miners whether to upgrade to the new consensus protocol. The new consensus protocol becomes active only when a predefined condition is met. The new consensus protocol activation may depend on the miners. They can include a predefined signal in blocks within a predefined block interval, and once the number of blocks including the given signal is greater than a certain number, the new consensus protocol will be activated [25].

After the Lima hard fork, the æternity blockchain will use miner signalled consensus upgrades. This mechanism is described in the following text.

3 Motivation

Since the æternity is a complex platform that consists of components such as naming system, oracles, smart contracts, generalized accounts and state channels [2], the development of these components and the blockchain itself is likely to be consensus breaking in the future. Moreover, the Lima hard fork is the last scheduled hard fork, so planned hard forks must be replaced with another mechanism.

To prevent the blockchain from a split, we chose to use the consensus upgrade based on

the miner signalling outcome. The vast majority (90% or more) of mining power must be supporting the new consensus protocol in order to upgrade the blockchain. The minority can still follow the old consensus, but since its mining power will be very low, this fork is generally supposed to become abandoned.

4 Specification

On a node configured for a consensus upgrade based on miner signalling outcome, the new consensus protocol becomes active at a determined height if at least a determined majority of the blocks between two determined block heights (*signalling interval*) contains a predefined *signal*. Such consensus protocol upgrade happens per each (accidental - see Subsection 2.1) fork as shown in Figure 1.

In order for the miner signalled consensus protocol upgrade to succeed, each node participating in this consensus upgrade must have the same set of parameters. Let us assume that *C1* is the current protocol version, and *H1* the height at which it was first effective. The parameters are:

- *new consensus protocol* is a protocol version which becomes active if the miner signalling is successful. It must be strictly greater than C1;
- *signal* is a part of a key block header and its value must be unique;
- *signalling interval start (HS)* is the height at which the signalling interval starts (inclusive), it should cater for the time required by most willing miners to adopt a version of the node able to signal and able to follow the new consensus protocol e.g. at least two weeks. HS must be strictly greater than H1;
- *signalling interval end (HE)* is the height at which the signalling interval ends (exclusive). HE must be strictly greater than HS. The difference between HE and HS is recommended to be approximately a week i.e. 3360 key blocks;
- *new consensus protocol height (fork height)* is the height at which the new consensus protocol may be activated. It is equal to HE;
- *majority* is the number of key blocks in the signalling interval containing a signal that is meant to activate the new consensus protocol. It is recommended to be 90% of the difference between HE and HS. So it is 3024 (90% of 3360).

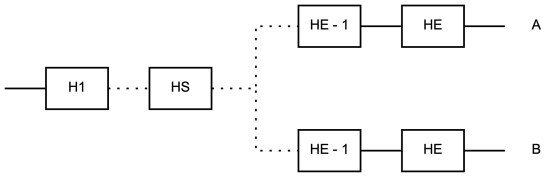

Figure 1: A fork within the signalling interval.

5 Implementation

The æternity network is composed of nodes, some of which are mining and produce new blocks, and the others are validating the produced blocks. Currently, there is one implementation of such a node which is written in Erlang programming language [1].

5.1 Configuration

After users and miners updated to the latest node version, which is able to signal and follow the new consensus protocol, they still can opt-out from the new consensus protocol by changing the configuration of a node. The only option is whether they want to follow the outcome of the miner signalling or stay with the current consensus protocol. Modification of the parameters from Section 4 is

not allowed to prevent the participants of the possible future consensus upgrade from having distinct miner signalling configuration parameters.

5.2 Code refactoring

We had to identify places in the code where we tried to get a protocol version from a block height. Since there were just planned hard forks, there was no ambiguity what protocol version was supposed to be effective at a certain height.

However, this changed with conditional forks as it is not possible to determine which protocol is effective at a given height (after Lima), because it depends on the miner signalling outcome. Therefore, we had to refactor code that assumed a protocol version from height [4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18]. This led us to a conclusion that the miner signalling outcome is required to be known only for a block at height HE whenever:

- a block (both key and micro block) candidate, eventually leading to a trusted block, is prepared;
- an untrusted block is being inserted into the storage. This includes the cases of a sync, gossip and HTTP API workers trying to insert a new key block into the storage;
- a trusted block is being inserted into the storage. This includes the cases of locally mined or signed key block or a block submitted by Stratum server.

5.3 Computation of miner signalling outcome

At first, we assumed that the miner signalling configuration will be possible to change. Therefore, we implemented a solution using asynchronous Erlang processes. For each key block at height HE − 1 there was created a new Erlang process which traversed the blockchain backwards until in reached the key block at height HS counting the key blocks containing a predefined signal. Stopping a node while the height of its top key block was within the signalling interval, changing its miner signalling configuration and starting the node again did not cause any issue with regard to the miner signalling outcome. The outcome always reflected the current node configuration [23].

Although, as described in Subsection 5.1, we disallowed changes to the miner signalling configuration parameters. It resulted into implementation of another solution, which is going to be used by the æternity blockchain. It uses a dedicated database table that stores aggregated signal count for each key block within the signalling interval. At the end of the signalling interval, the node is able to make a decision whether to activate the new consensus protocol based on the aggregated signal count related to a block at the height HE − 1 [19, 20, 21].

5.4 Backward compatibility

The signal value is stored in the field `info`, which is a part of a key block header. The meaning of this field is not under consensus.

The node's storage is backward compatible as there were no modifications done to the existing tables.

5.5 Possible future extensions

The `info` field used for storing the signal value offers room for accommodating simultaneous consensus proposals by interpreting the `info` field as a bit mask rather than a value. This mechanism could also be extended to support miner activated soft forks as described in Subsection 2.2.

6 Conclusion

The activation of the new consensus protocol based on the miner signalling outcome is meant to be a temporary measure during the time interval when the new protocol may be activated on the network while the node is running. Once the network reasonably settles on whether to activate the new consensus protocol, the user is meant to enforce such decision on the eventual node restart either by configuration or by updating the node to a new version.

References

[1] Aeternity node. https://github.com/aeternity/aeternity/.

[2] Aeternity protocol. https://github.com/aeternity/protocol/.

[3] Bitcoin-NG for Aeternity. https://github.com/aeternity/protocol/blob/master/consensus/bitcoin-ng.md.

[4] Pull request #2678. https://github.com/aeternity/aeternity/pull/2678.

[5] Pull request #2686. https://github.com/aeternity/aeternity/pull/2686.

[6] Pull request #2697. https://github.com/aeternity/aeternity/pull/2697.

[7] Pull request #2705. https://github.com/aeternity/aeternity/pull/2705.

[8] Pull request #2760. https://github.com/aeternity/aeternity/pull/2760.

[9] Pull request #2763. https://github.com/aeternity/aeternity/pull/2763.

[10] Pull request #2769. https://github.com/aeternity/aeternity/pull/2769.

[11] Pull request #2774. https://github.com/aeternity/aeternity/pull/2774.

[12] Pull request #2778. https://github.com/aeternity/aeternity/pull/2778.

[13] Pull request #2793. https://github.com/aeternity/aeternity/pull/2793.

[14] Pull request #2800. https://github.com/aeternity/aeternity/pull/2800.

[15] Pull request #2802. https://github.com/aeternity/aeternity/pull/2802.

[16] Pull request #2804. https://github.com/aeternity/aeternity/pull/2804.

[17] Pull request #2837. https://github.com/aeternity/aeternity/pull/2837.

[18] Pull request #2847. https://github.com/aeternity/aeternity/pull/2847.

[19] Pull request #2868. https://github.com/aeternity/aeternity/pull/2868.

[20] Pull request #2910. https://github.com/aeternity/aeternity/pull/2910.

[21] Pull request #2913. https://github.com/aeternity/aeternity/pull/2913.

[22] Soft Fork. https://en.bitcoin.it/wiki/Softfork.

[23] The computation of miner signalling outcome using asynchronous processes. https://github.com/aeternity/aeternity/tree/PT-166642114-hard-fork-signalling.

[24] Andreas M. Antonopoulos. *Mastering Bitcoin*. 2 edition, 7 2017. https://github.com/bitcoinbook/bitcoinbook.

[25] Bisade Asolo. Blockchain Soft Fork and Hard Fork Explained, 11 2018. https://www.mycryptopedia.com/hard-fork-soft-fork-explained/.

[26] Bitcoin Exchange Guide News Team. User activated soft fork. https://bitcoinexchangeguide.com/uasf/.

[27] Eric Lombrozo. Forks, Signalling, and Activation. https://medium.com/@elombrozo/forks-signaling-and-activation-d60b6abda49a.

[28] Emin Gün Sirer Robbert van Renesse Ittay Eyal, Adem Efe Gencer. Bitcoin-NG: A Scalable Blockchain Protocol. 3 2016. https://www.usenix.org/system/files/conference/nsdi16/nsdi16-paper-eyal.pdf.