MASTER OF SCIENCE
IN ENGINEERING

Haute Ecole Spécialisée
de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences and Arts
Western Switzerland

Master of Science HES-SO in Engineering
Av. de Provence 6
CH-1007 Lausanne

Master of Science HES-SO in Engineering

Orientation: Information and Communication Technologies (ICT)

IMPROVING COSTS, SAFETY AND LIVENESS IN BLOCKCHAIN SYSTEMS

Author:
Nicolas Huguenin

Under the direction of:
Prof. Dr. Marcelo Pasin
HE-Arc

Neuchatel, HES-SO//Master, January 12, 2019

Information about this report

Contact information

Author:	Nicolas Huguenin
	MSE Student
	HES-SO//Master
	Switzerland
Email:	*nicolas.huguenin@master.hes-so.ch*

Declaration of honor

Validation

Abstract

TODO: add abstract

Keywords: Algorithms; Blockchain; Distributed Systems; Sharing Economy

Contents

1 Introduction

1.1 Context

The Ethereum blockchain aims to move from the current PoW consensus protocol to a PoS one. Multiple teams are currently researching ways to describe, and implement such a protocol. One of these teams, lead by Vlad Zamfir, is working on a protocol family called Correct by Construction (CBC) Casper. CBC-Casper describes an abstract set of protocols that can achieve consensus between any kind of value, for example an integer, a vote, or a blockchain. This project aims to find and compare block publishing strategies for a CBC-Casper blockchain consensus.

1.2 Objectives

As the CBC-Casper paper only describes an abstract way of consctructing PoS consensus protocols, and does not make any assumptions on synchrony, one of the main challenges of the actual implementation is to find incentive mechanisms and strategies telling the nodes when to produce blocks. The main goals of this project are:

- to propose multiple block producing strategies;
- to create a model that allows one to easily compare said strategies;
- to discuss the advantages and disadvantages of each strategy.

1.3 Contents

TODO: explain what is in this

2 | Background

2.1 PoW and PoS

In Ethereum, PoS is aiming at replacing PoW as a distributed consensus algorithm. This section succintly describe both methods as well as the main differences between them, and then explains which problems arise when you replace PoW with PoS.

2.1.1 What is PoW?

PoW is the current consensus protocol used to decide on a blockchain in Ethereum. In order To create a new valid block, a node has to solve a cryptographic puzzle and include its solution in the newly created block. The difficulty of the puzzle is parametrized in order to have -on average- a block created at a set interval. A reward is given to the creator of each block. The consensus rule states that the chain with the greatest total difficulty is to be considered the main one. Miners are therefore incentivised to build on the main chain if they want to get rewards for their work. The fact that the difficulty changes to keep a certain interval between blocks means that said work is a proxy for timing; a miner cannot create an arbitrary large number of blocks in a short time because it's inherent to the protocol.

2.1.2 What is PoS?

PoS, on the other hand, selects a new block creator according to its weight (or stake). This weight can be the node's age, wealth, etc. In this report, we will mainly discuss a specific PoS protocol, CBC-Casper.

2.1.3 CBC-Casper

CBC-Casper TODO: cite somewhere is an abstract consensus protocol family which is PoS-ready. Nodes, called validators in this context, send messages to each other, acknowledging they saw other messages by including them in a *justification*, that is attached to each message. Based on its justification as well as a weighted list of validators, each message defines an *estimate*, which is the consensus value proposed by the sender of the message. In the case of a blockchain, messages each point to one older message as their estimate and form a *block-Directed Acyclic Graph (DAG)*. Running a slightly modified version of the Greedy Heaviest Observed Sub-Tree (GHOST) algorithm on the DAG returns a blockchain.

TODO: change structure; make comparison not between POS and POW but between POW and CBC

2.1.4 POS vs POW

TODO: talk about differences between pos and pow

	PoW	PoS
Block production	Miners can publish blocks if the can prove they worked for it	Nodes can publish blocks at any time
Timing assumption	Work is a proxy for timing	None
Spam	Work removes the possibility to spam	Negligible computationnal costs to produce blocks imply potential spam
Economic majority	Work is a proxy for economic majority	Economic majority
Building strategy	Nodes are incentivised to build on the longest chain because they have to work for to build blocks	No clear incentive to build on the longest chain

Table 2.1 Summary of key differences between PoW and PoS

2.1.5 key differences

2.1.6 new problems

TODO: talk about key differences and what this project is about

2.2 CBC casper

TODO: describe the protocol in my own words

TODO: describe liveness, safety

2.3 `core_cbc`

`core_cbc` a Rust implementation of the CBC-Casper, made by TrueLevel. It implements the consensus algorithms proposed in the paper and offers an abstract structure that can be used to create consensus on any value.

2.4 the other casper?

2.5 Parity Ethereum

Parity is a Rust Ethereum client. It includes a *Pluggable Consensus* module that allows one to easily add new consensus protocols by implementing an interface. At first, the goal of this project was to implement a small bridge between the Parity module and the core cbc implementation to test block creation strategies in a pseudo real-like manner. The implementation of the bridge was not as straight forward as planned so it has been decided to cut it out and test strategies without mimicking the network and client settings.

2.5.1 Background work

During the early stages of this project, a clear objective was set: to be able to run a Casper *testnet* with Parity custom nodes. Some work has been done for the implementation of the `core-cbc` library into Parity, but that was left to people that had a better understanding of the underlying library. Furthermore, TODO: rephrase "a lot of" a lot of effort had been injected in the creation of a Docker infrastructure in order to easily deploy, connect and monitor multiple custom Parity nodes on a single machine in order to experiment with different message building strategies. The choice of using Docker was made because there was a possibility to work on the UniNE clusters, which happen to work well with containerized software. After seeing that the Parity implementation would take too much time to be completed, and therefore might be unusable for this thesis, it has been decided to evaluate strategies inside the `core-cbc` library instead of the more real-life-like setting that is a Parity testnet. The main disadvantage of doing the experimentations in the library is that the whole network latencies and topology are not taken into account. However, a non-negligible advantage of implementing strategies in the core library is that it will be easier to test them on consensus values that are not only blockchains.

3 Strategies Evaluation

3.1 Modelisation

3.2 Strategies

The following strategies were proposed in order to visit the entierty of the trade-off triangle:

- randomness;
- round-robin;
- double round-robin;
- overhead.

These strategies should allow one to visit the whole triangle and to discuss their respective strength and weaknesses. The following sections describe the strategies as well as their expected locations in the triangle.

3.2.1 Round-robin

The first strategy that comes to mind is a simple round-robin. Nodes send messages one after the other, in a fixed order.

3.2.2 Randomness

The next strategy is the simplest to think of: complete randomness. Using fixed probability density functions, nodes chose when to create messages and to which other validator to send them.

3.2.3 Double Round-robin

In this setting, two nodes send messages at the same time, in a fixed order. If the two nodes that send messages at the same step are at opposite places in the set of validators TODO: explain better, the latency to finality is supposedly half as much as the simple round-robin strategy. The overhead is however doubled.

3.2.4 Maximal Overhead

This strategy is the most expensive in terms of bandwidth; at each step, each validator sends a message to the others. This example strategy should give a baseline value for the maximum overhead that is reachable in the tradeoff triangle.

3.3 Experimentations

Over the duration of this thesis, the `core-cbc` library has included a test framework called *proptest*. The testing framework that has been implemented includes ways to simulate the behavior of the Casper protocol over multiple nodes and thousands TODO: numbers of blocks. At the time of the writing, the simulations do not include networking latencies.

TODO: schema with what to measure TODO: how the measurements take place in the code

3.4 Visualization

3.5 Analysis

4 Conclusion

TODO: insert conclusion

List of Figures

List of Tables

Glossary

CBC	Correct by Construction. 1, 3, 4
DAG	Directed Acyclic Graph. 3
GHOST	Greedy Heaviest Observed Sub-Tree. 3
PoS	Proof-of-Stake. vii, 1, 3, 4, 13
PoW	Proof-of-Work. vii, 1, 3, 4, 13