

University of Luxembourg

Multilingual. Personalised. Connected.

RepuCoin: Reputation-based Byzantine Consensus

Jeremie Decouchant,

Joint work with Jiangshan Yu, David Kozhaya, Paulo Esteves-Veríssimo

CritiX, SnT

jeremie.decouchant@uni.lu

RepuCoin: addressing the 51% attack



RepuCoin: addressing the 51% attack



IEEE Transactions on Computers 2019

RepuCoin: Your Reputation is Your Power

Jiangshan Yu*, David Kozhaya¹, Jeremie Decouchant*, and Paulo Esteves-Verissimo*
* SNT, University of Luxembourg, Luxembourg
¹ ABB Corporate Research, Switzerland

Abstract—Existing proof-of-work (PoW) cryptocurrencies cannot tolerate attackers controlling more than 50% of the network's computing power at any time, but assume that such work's computing power is "unlikely". However, recent attack conditions happening in "unlikely", where attackers can rent mining capacity sophisticatedly, e.g., where attackers can rent temporarily (flash attacks), render this assumption unrealistic.

This paper proposes RepuCoin, the first system to provide guarantees even when more than 50% of the system's computing power is temporarily dominated by an attacker. RepuCoin physically limits the rate of voting power growth of the entire system. In particular, RepuCoin defines a miner's power by its "reputation" as a function integrated over the entire blockchain, rather than through its sheer computing power which can be obtained relatively quickly and temporarily. As an example, after a single year of operation, RepuCoin can tolerate attacks compromising 51% of the network's computing resources, even if such power stays maliciously seized for almost a whole year. Moreover, RepuCoin provides better resilience to known attacks, compared to existing PoW systems, while achieving a high throughput of 10000 transactions per second (TPS).

1. Introduction

Bitcoin [47] is the most successful decentralized cryptocurrency to date. It has skyrocketed from almost no value to 7205 USD/BTC, as of Nov 2017. However, despite its enormous commercial success, many weaknesses have been associated with Bitcoin, including weak consistency, low transaction throughput and vulnerabilities to attacks, e.g., double spending attacks [30, 3], eclipse attacks [12], selfish mining attacks [20, 55, 49], and flash attacks [12]. Several promising existing solutions [39, 52, 16, 37, 21] targeted the low throughput problem of Bitcoin. Nevertheless, these solutions either provide only probabilistic guarantees about transactions (weak consistency) [21] or can provide strong consistency but suffer from liveness problems even when an attacker has a relatively small computing power [33]. Moreover, the resilience of such solutions against attacks, such as selfish mining attacks where an attacker has more than 25% of the computing power [3, 20, 55, 49], remains unsatisfactory. In addition, all existing contemporary proof-of-work (PoW) based variants of Bitcoin (e.g. [2, 51, 21, 33]) rely on the assumption that an attacker cannot have more than 33% or 50% of computing power at any time. However, with the sophistication of attacks mounted on Bitcoin, e.g., flash attacks

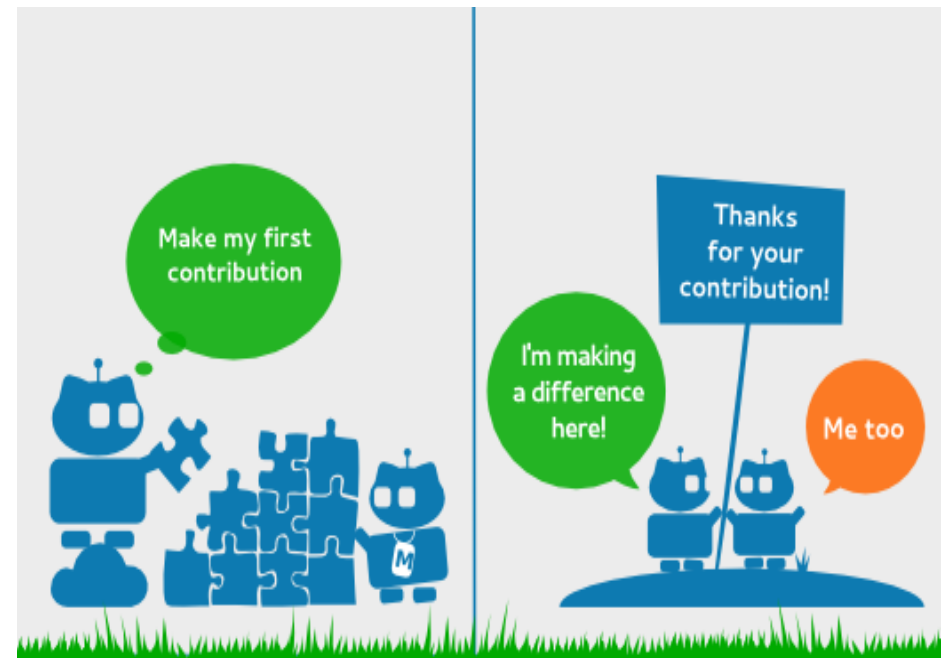
(a.k.a. bribery attacks), where an attacker can obtain a temporary majority (>50%) of computing power by renting existing capacity [12], all these systems would fail. In brief, existing solutions that address the weaknesses associated with Bitcoin still suffer from significant shortcomings—liveness of current high-throughput solutions, and vulnerability to attacks such as selfish mining and flash attacks. In particular, this paper addresses these shortcomings—liveness of current high-throughput solutions, and vulnerability to attacks such as selfish mining and flash attacks. In particular, we propose RepuCoin, the first system that can prevent attacks against an attacker who may possess more than 50% computing power of the entire network temporarily (e.g., a few weeks or even months). Our proof-of-concept implementation shows that while providing better security guarantees than predecessor protocols, RepuCoin also ensures a very high throughput (10000 transactions per second). In practice, Visa confirms a transaction within seconds, and processes 1.7k TPS on average [59]. This shows that RepuCoin satisfies the required throughput of real world applications.

Design principle. Our system addresses the aforementioned challenges by defining a new design principle, called *proof-of-reputation*. *Proof-of-reputation* is based on *proof-of-work*, but with two fundamental improvements. First, under *proof-of-reputation* a miner's decision power (i.e., the voting power for reaching consensus in the system) is given by what we call the miner's "instantaneous" power, i.e., the miner's computing power in a short time range, as in classic PoW. Instead, the reputation is computed based on both the total amount of valid work a miner has contributed to the system and the regularity of that work, over the entire period of time during which the system has been active. We call this the miner's "integrated work", when an attacker joins the system at time t , even if it has a very strong mining ability that is, high computational (i.e., instantaneous) power, it would have no integrated power at time t , or even shortly after, as it did not contribute to the system before t .

Second, when a miner deviates from the system specifications, RepuCoin lowers the miner's reputation, and hence its integrated power, in consequence of this negative contribution. This prevents a powerful malicious miner from attacking the system repeatedly without significant consequences. In contrast, classic PoW systems either do not support any feature for punishing miners that do not abide by system specifications, or they punish these miners by merely revoking their rewards — this does not prevent them from

RepuCoin – Intuition

1. Miners gain reputation by contributing to the blockchain
2. Only top reputed miners can vote through a BFT protocol (e.g., PBFT)
3. Mis-behaved miners will be punished, and they lose reputation
4. Leaders are randomly selected from top reputed miners to propose transactions



RepuCoin – Increased attack resilience

- For an adversary to build enough reputation takes time

Breaking the liveness property:

Joining time \ Target	1 week	1 month	3 months	6 months
1 month	infeasible	45%	30%	27%
3 months	infeasible	90%	45%	33%
6 months	infeasible	infeasible	68%	45%
9 months	infeasible	infeasible	90%	54%
12 months	infeasible	infeasible	infeasible	68%
18 months	infeasible	infeasible	infeasible	91%
20 months	infeasible	infeasible	infeasible	infeasible

RepuCoin – Increased attack resilience

- For an adversary to build enough reputation takes time

Breaking the liveness property:

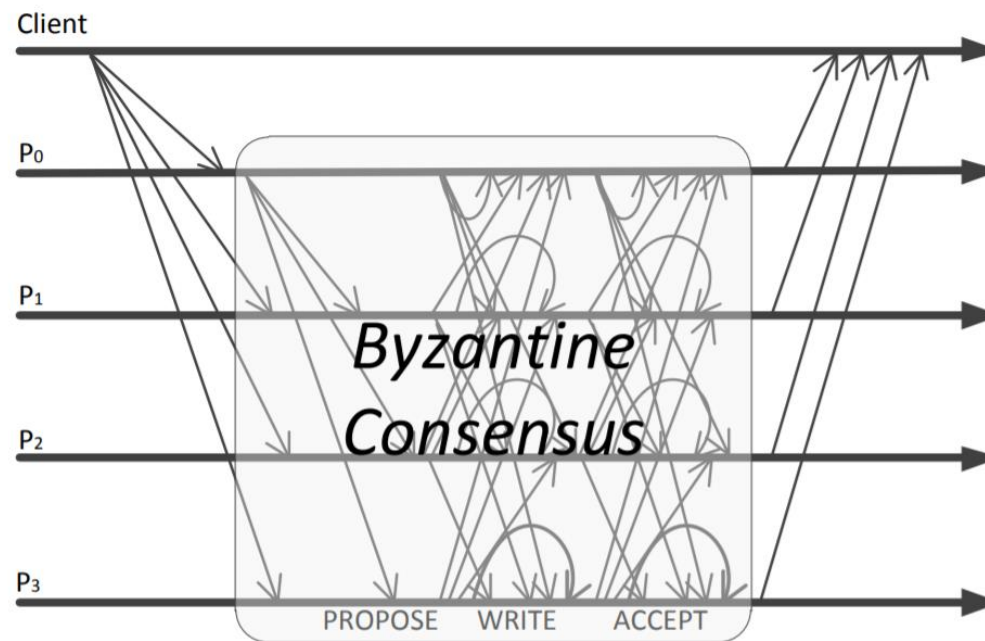
Joining time \ Target	1 week	1 month	3 months	6 months
1 month	infeasible	45%	30%	27%
3 months	infeasible	90%	45%	33%
6 months	infeasible	infeasible	68%	45%
9 months	infeasible	infeasible	90%	54%
12 months	infeasible	infeasible	infeasible	68%
18 months	infeasible	infeasible	infeasible	91%
20 months	infeasible	infeasible	infeasible	infeasible

Implementation: BFT-SMaRt (Java)

<https://github.com/bft-smart/library>

A block was discovered in the network

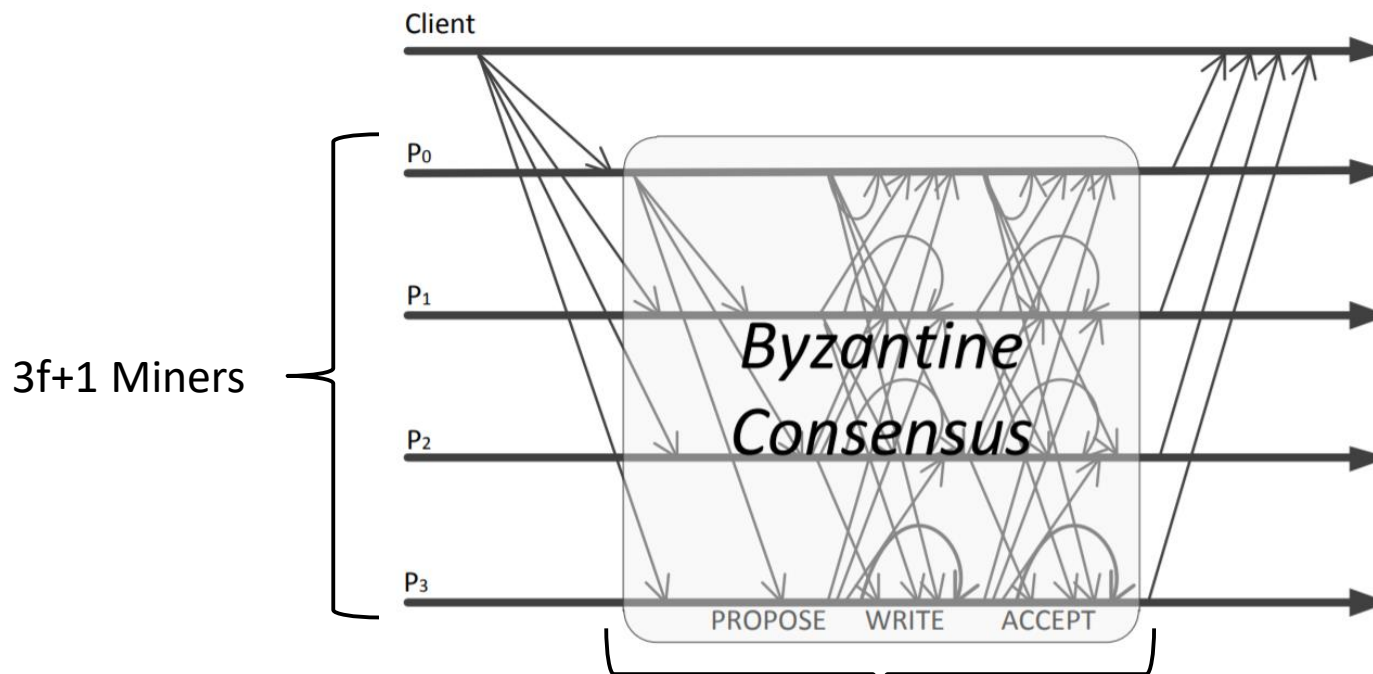
The block is added to the blockchain



Block ordering

Implementation: BFT-SMaRt (Java)

<https://github.com/bft-smart/library>



Consensus reached when:

- $2f+1$ miners agree on a block
- They represent more than two 3rd of the group reputation

Performance evaluation

Measure

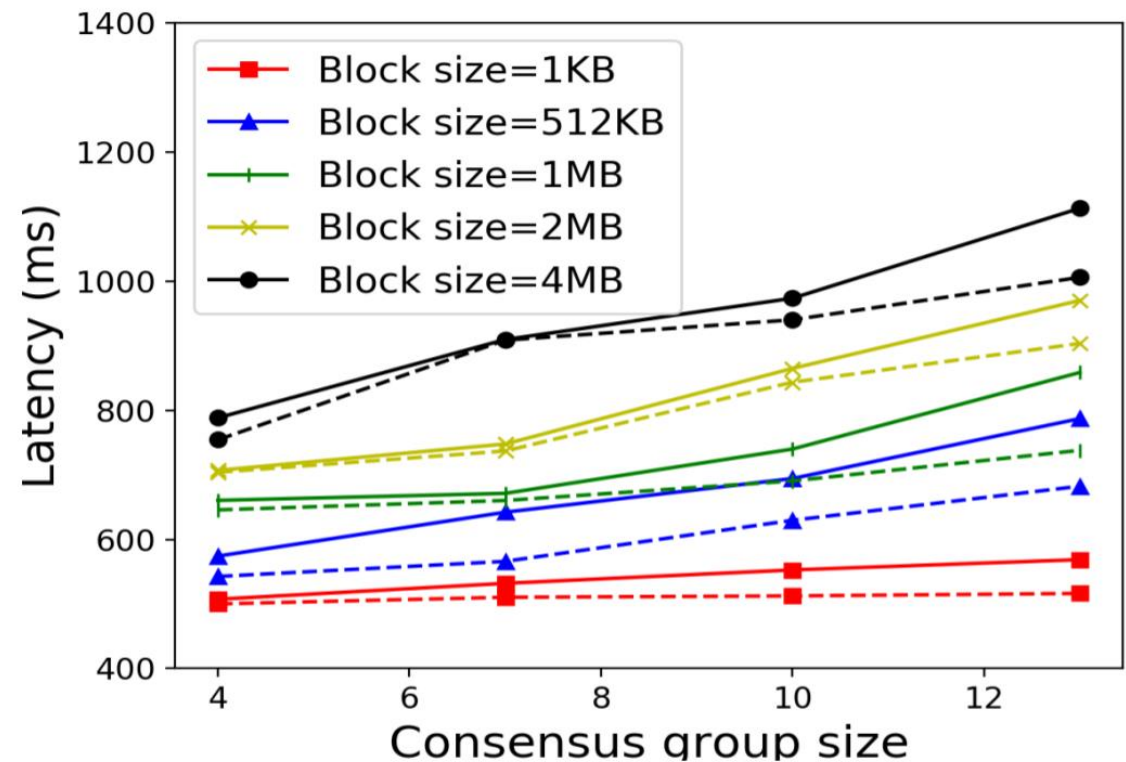
- Latency
- Throughput

Depending on

- Consensus group size
- Block size

Settings (in the code)

- Limit bandwidth
- Impose network latency



1. Find a set of machines on a single cluster
2. Create an interactive job and connect to it

```
$ oarsub -I -l nodes=13,walltime=1:0:0  
$ oarsub -C 12345
```

3. Edit BFT-SMaRt's config files (machines to use and port)

```
$ cat $OAR_NODEFILE
```

4. Bash: script to run the throughput/latency benchmark

- Kill any java application on the machines
- Launch replicas
- Launch clients

```
$ oarsh -f ${ip_addr} "cd bftsmart.repucoin; ./runscript > /dev/null 2>&1 &" &
```

5. Python: collect the results (output file) and plot

- Search for the right code basis
 - Your life will be much easier
- Automate everything
 - You always think you won't need to repeat the experiments: wrong!
 - The initial additional work is quickly amortized
- Latency vs. throughput experiments are tricky
 - The throughput should increase with the load up to a certain point, where the latency starts increasing
 - But too many requests make the applications crash (message queues)
 - Find the right number of clients

- Estimate the time it takes for your experiment and double it
 - Plan ahead
- It is difficult to be on a completely controlled environment
 - Change the machines → Change your performance
 - Are there a lot of jobs ongoing?
- Performance is sometimes difficult to understand
 - Example: The performance with 8MB blocks is lower than with 4MB
 - I spent a day repeating the experiments and got the same result: I still don't explain it