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

UNIVERSITÉ DU LUXEMBOURG

RepuCoin: addressing the 51% attack

RepuCoin: addressing the 51% attack

IEEE Transactions on Computers 2019

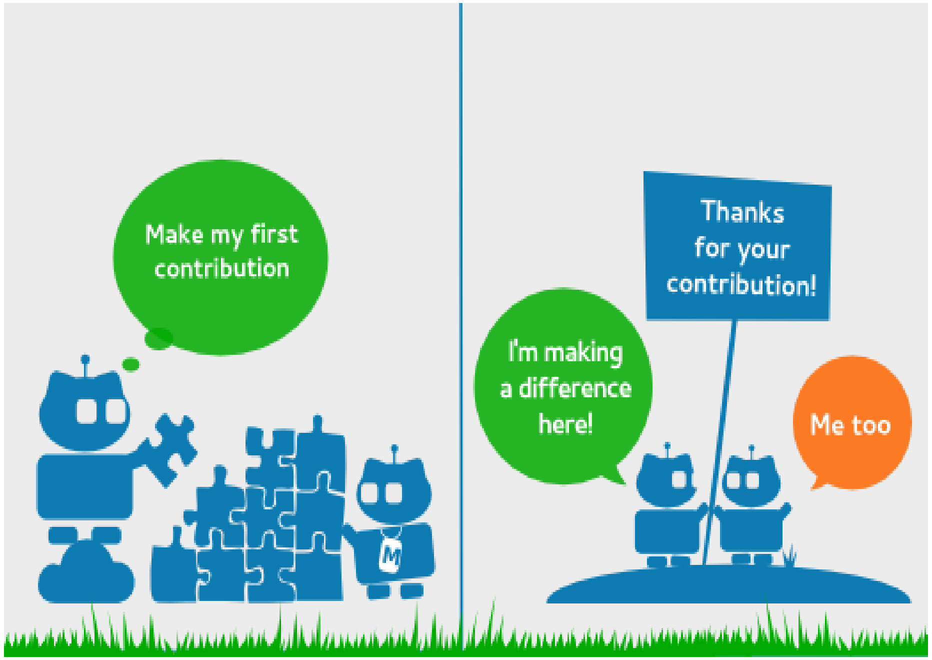
RepuCoin: Your Reputation is Your Power

Jiangshan Yu *, David Kozhaya†, Jeremie Decouchant*, and Paulo Esteves-Verissimo*
* SnT, University of Luxembourg, Luxembourg.
† 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 a condition happening is "unlikely". However, recent attack sophistication, e.g., where attackers can rent mining capacity to obtain a majority of computing power 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).

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:

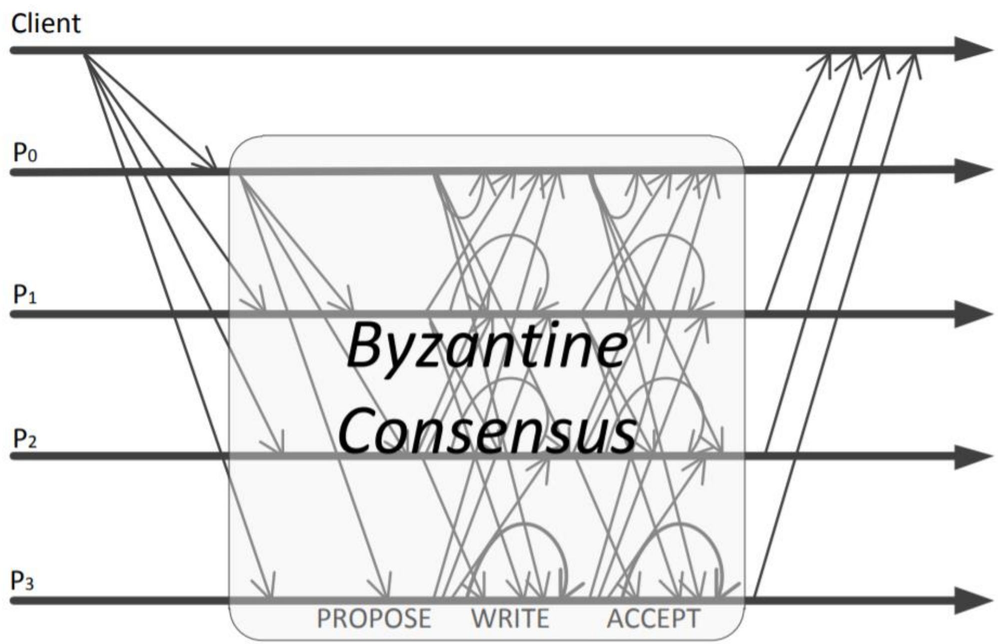
Joining time\ Target	1 week	1 month	3 months	6 months
1 month	infeasible	45%	30%	27%
3 months	infeasible	90%	45%	33%
6 months	infeasible	infeasible	68%	45%
9 months	infeasible	infeasible	90%	54%
12 months	infeasible	infeasible	infeasible	68%
18 months	infeasible	infeasible	infeasible	91%
20 months	infeasible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

Implementation: BFT-SMaRt (Java)

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

A block was discovered in the network

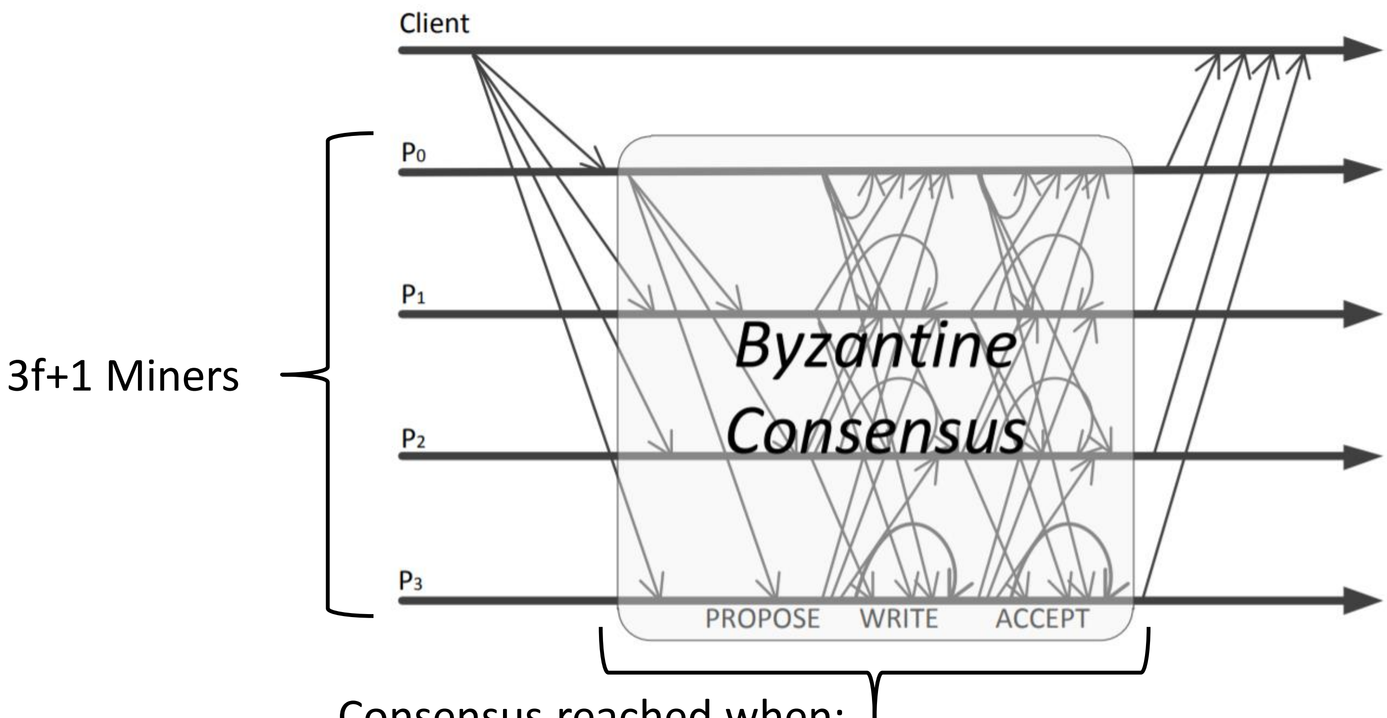
The block is added to the blockchain



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

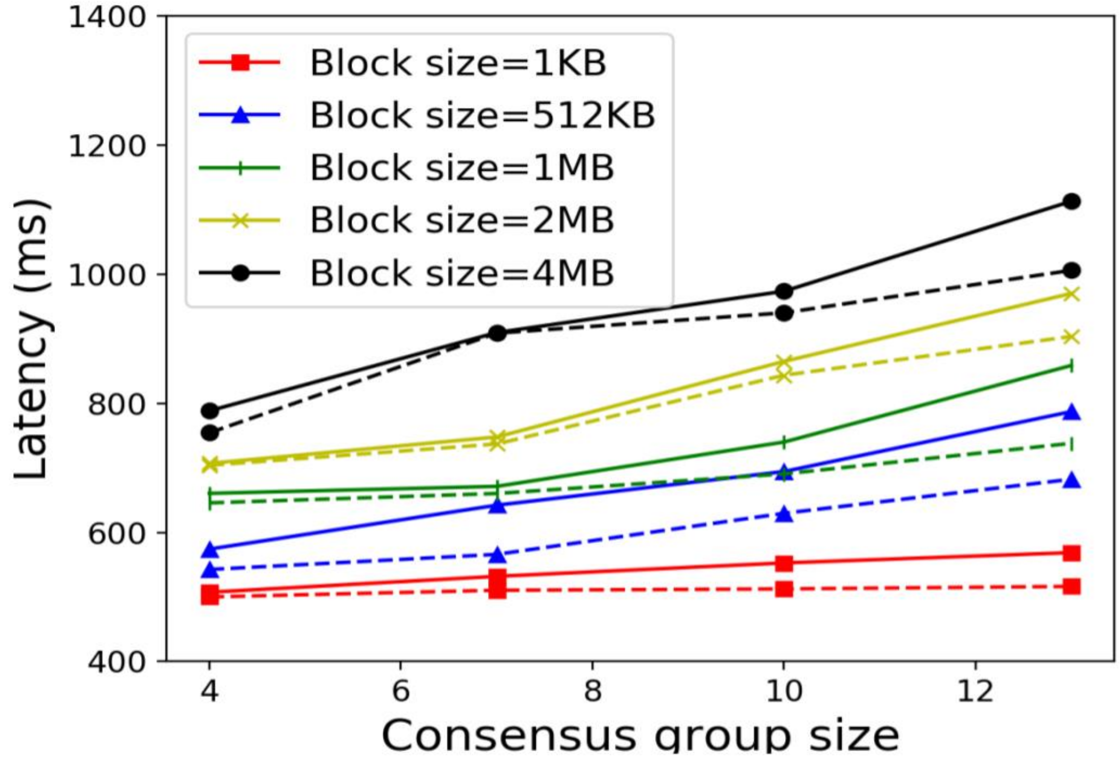
- Latency
- Throughput

Depending on

- Consensus group size
- Block size

Settings (in the code)

- Limit bandwidth
- Impose network latency



HPC workflow

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

Best practices

- 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

Lessons learned

- 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