Proof-of-Clash

A fast, fair, provably (p)BFT (cross-chain) consensus algorithm

v0.5

Bearreth Bender `@BullrethB`

Alfi Atron `@ReignOfAtron`

Kirk Kareleos `@Careleossness`

Vladimir Vabtcarius `@VladdiBit`

Neal Nabchels `@NealNoSteal`

June 18, 2019

Abstract

In [10], Satoshi Nakamoto first introduced the concept of consensus in a distributed system in order to achieve finality of data. We present an improved version of Bitcoin's Proof-of-Work (PoW) consensus by introducing Proof-of-Clash (PoC) which solves pretty much all the issues that PoW has. We also propose the application of PoC to create consensus between mutliple blockchain networks (or other DLTs) in order to ensure cross-chain communication under a common consensus layer.

1 Introduction

Ever since the first introduction of Bitcoin and the Proof-of-Work consensus mechanism in 2008, researchers, technologicians and economists have been pursuing improvements over the way of reaching consensus through application of computing power. These took shape in various new, supposedly improved consensus mechanisms:

- Proof-of-Stake (PoS), as proposed for Ethereum's Casper, or in Cardano
- Delegated-Proof-of-Stake, as first introduced with BitShares and later applied in Steem, EOS, Lisk and Tron
- Other, graph-based Byzantine Fault Tolerant consensus mechanisms as in IOTA, Stellar or Hashgraph

- And many, many other

While many of the abovementioned proposals had advantages over the original PoW, e.g. in

- Scalability
- Decentralization
- Cost
- Usage of natural resources
- Security

none of them has actually managed to take a lead across multiple use cases. Ethereum with its PoW basis is still the number 1 DLT network because of its security and resilience, while EOS or tron mostly have applications in Gambling or so-called "high risk", mainly Ponzi schemes.

We're here to change that. We propose Proof-of-Clash (PoC), which is a consensus layer being introduced in ClashChain (tbc), but is generally open-source. It provides a scalable, fair and provably Byzantine Fault Tolerant (BFT) algorithm to reach consensus in a permissionless distributed public data structure, e.g. a database or a blockchain.

2 Core concepts

We define the following terms and concepts:

- *Transactions* - any member can create a signed transaction at any time. All members get a copy of it, and the community reaches Byzantine agreement on the order of those transactions
- *Blocks* - a set of transactions that the majority of the network has agreed are correct, consistent and do not include double spends
- *Fairness* - it should be difficult for a small group of attackers to unfairly influence the order of transactions that is chosen as the consensus
- *Clash* - a challenge between two participants to figure out who is fitter for a given task, in our case finding a block
- *Clash Blocks* - in order to ensure consistency between transactions and clashes in a given block, hashed data about all clashes for that block is stored in the block itself

- *Mining* - the act of contributing to the security of the network by performing challenges (i.e. clashing)
- *(Block) Rewards* - incentives for the participants who successfully engage in mining upon every block
- *Faction* - a group of network participants who work together in an honest fashion to secure the network
- *Clash Power* - the potency of mining, similar to hash rate in PoW. Clash power can be delegated to factions
- *Sybil attack* - In a Sybil attack, the attacker subverts the reputation system of a peer-to-peer network by creating a large number of pseudonymous identities and uses them to gain a disproportionately large influence. A reputation system's vulnerability to a Sybil attack depends on how cheaply identities can be generated, the degree to which the reputation system accepts inputs from entities that do not have a chain of trust linking them to a trusted entity, and whether the reputation system treats all entities identically.
- *Famous Witness* - The community could put a list of n transactions into order by running separate Byzantine agreement protocols on $O(nlogn)$ different yes/no questions of the form "did event x come before event y?" A much faster approach is to pick just a few participants, to be called witnesses, and define a witness to be famous if it has specific properties to represent a faction better than most other participants. Then it's sufficient to run the Byzantine agreement protocol only for witnesses, deciding for each witness the single question "is this witness famous?" Once Byzantine agreement is reached on the exact set of famous witnesses, it is easy to derive from the network topology a fair total order for all transactions

3 The PoC algorithm

The Proof-of-Clash algorithm is based on *Clashes*, challenges between participants or factions of participants. Instead of all participants in the network having to participate in a given challenge, clashes only occur between two specific participants or factions. After a number c of clashes a winner can be derived based on their performance in the clashes.

3.1 Basic clash

We define a clash between two ordinary participants in the network to be a *basic clash*. Basic clashes are deterministic, light-weight computations of a value

with a certain *seed*. The participant that calculates the better result based on the participant's properties and the seed wins the clash and receives a certain amount of points p. Similar to the Elo point system in chess, the number of points a participant A can win or lose in a clash against B depends on

- the number of points A has
- the number of points B has

The larger $p_A - p_B$, the fewer Δp A will receive and the fewer points B will lose in case A wins.

Once the network has run enough (c) clashes, the participant W with the highest p wins and is allowed to mine the current block.

Basic information about the clashes that have elected the winning participant is put into the block in order for other participants to be able to validate correctness of the block.

3.2 Determining c

The number c of minimum clashes needed to determine a winner obviously needs to correspond with the number n of mining participants in the network. However, as the network topology typically does not change very frequently, it is sufficient to update c every u blocks.

c can therefore be a parameter that differs in networks applying PoC; however, we recommend something similar to:

$c = k * log(n)$,

whereas k is a number that can be set by the network.

3.3 Faction clash

Rather than clashing on their own, participants in the network can form factions to perform *faction clashes* in order to have a higher chance of winning sufficient clashes to mine a block. This mechanism can be compared to mining pools in PoW, however, it is baked into the protocol.

Network participants can keep on clashing individually; if they delegate their *clash power* to a faction, their results will count towards the results of the faction instead. That way, factions with the most potent individual participants have

the best chances of mining; therefore, the best suited faction will be the one creating the block and the block rewards will be distributed across the delegating participants.

Within factions, the node actually validating transactions and forming the block is selected randomly; this is meant to incentivize creating smaller, potent, factions as otherwise the likelihood of actually mining the block by a specific node will be extremely low.

4 Proofs of Clash

In order to prove legitimacy of a clash and a clash result, we need a few proofs in order to ensure correctness and fairness of the protocol.

We start with proving that a clash happened.

4.1 Clashing

In order to prove that two participants have indeed clashed, we use simple Secure Multiparty Computation (SMPC) methods as described in [12], [5], [4]. Both participating nodes need to securely compute their part of the hash individually, which can then be validated by any other node without actually knowing the inputs of the two computations. The complexity of such an operation is very limited,

$$T(A) = O(C(f)log\frac{1}{\epsilon\delta})), \tag{1}$$

whereas $T(A)$ is the number of bits exchanged between the two participants.

This provides a simple, verifiable proof for a clash.

4.2 Clash results

To prove the clash results themselves we can follow a very similar approach, only with slightly more expensive operations, as we also need to prove the result of a clash. We cannot trust any single node to correctly report (broadcast) a clash result; therefore, the network will be able to verify the results by receiving the results of both computations f and g of the respective clashing parties as well as the seed; and without knowing the individual private variables, they can verify that both had computed the values for the clash correctly.

4.3 Turned-down clashes

Clashes that one of the participants does not accept are also easily comprehended by missing computational values of the respective clash function f or g with the seed of this specific clash. Nodes that consistently turn down clashes, e.g. because they are busy picking their noses or caught up in a lot of different clashes, will be penalized in their overall score; thereby restoring balance of power by the "masses" in case one node got overly powerful. Weak nodes can clash against strong nodes a lot in order to do that.

Formal verification is possible through methods shown in [3].

Figure 1: A clash is taking place invisibly on the network, but we could imagine it looks pretty much like this.

4.4 Merit / Fairness

Nodes that win clashes frequently will have a higher score and therefore have a higher chance of mining (or the factions they delegated to). This provides an order of merit for the participants of a network; thereby a node that has contributed to the network's stability and success will be rewarded more frequently with block rewards than nodes that, although they perform well, don't have a track record in providing legitimate service to the network.

This solves one of the largest shortcomings of Delegated-Proof-of-Stake protocols, in which delegates (also witnesses / block producers) simply depend on their likability in the community; votes for specific delegates are often being bought with a share of the block rewards, but also similar systems have failed to provide satisfactory results as could be seen in the 2016 US presidential election. There, the presidential electors had to vote for the candidate who had the fewer total number of votes; and in addition, this candidate was Donald Trump.

In addition, such a system is supremely fair.

No accumulation of wealth: Nodes which are rich don't become richer just by being rich in the first place.

Equal chances: In the beginning, every node has equal chances as long as they contribute to the stability of the network and clash.

Cost efficiency: Running a node is extremely low cost, enabling nodes to be run on pretty much any kind of hardware by anyone having sufficient network bandwidth.

5 Proof of practical Byzantine fault tolerance

This section will soon be released.

6 Discussion

In this section we want to compare various traditional DLT consensus mechanism with PoC and distill the benefits of the latter.

Proof-of-Work: This is super obvious. Why would PoC not be better than PoW if it was designed specifically to be that - better than PoW? It's faster, more lightweight, fairer, allows for shorter confirmation times and higher throughput. On top of that, it is equally secure because of the supreme security of the PoC protocol as shown in 5. This also counts for Proof-of-Activity as suggested in [2]

Proof-of-Stake: PoC combines the advantages of PoS with even more advantages. PoS typically has a problem with fair distribution and accumulation of wealth (thereby increasing the likelihood of accumulating even more wealth). We have shown in 4.4 that PoC does not suffer from these issues. Also compare [7]

Delegated-Proof-of-Stake: Voting for delegates is difficult. Nobody votes. This is being discussed in all the DPoS communities: *how do we get participants to vote?* There you go - you don't. You let them clash. If they don't want to, they lose.

Directed Acyclic Graph: Haha, funny. This is not a consensus mechanism, but a data strcuture. PoC can very well be applied to non-linear data structures. We will demonstrate that in an upcoming paper. Stay tuned.

Hashgraph: Interesting. Hashgraph is great, except there is no open source implementation of it that has proven to work. Also, Hashgraph suffers a lot from changing network topology. PoC doesn't.

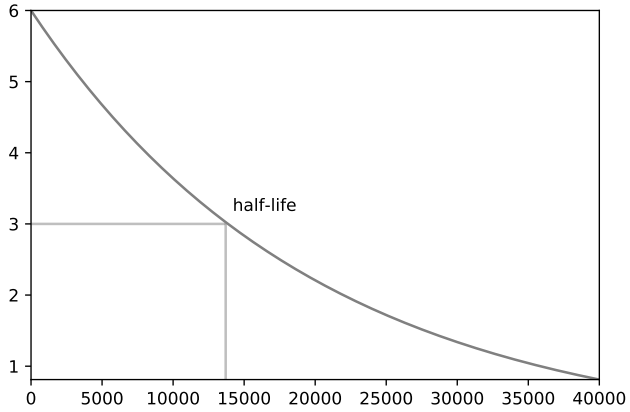

Figure 2: The figure shows the average half-life of any other consensus mechanism once Proof-of-Clash has been launched. The x axis shows the number of blocks mined by PoC, y represents the number of consensus mechanisms that the crypto community believes in.

7 Applications

With the supreme Proof-of-Clash approach we can imagine almost unlimited decentralized applications. However, we present a few specifics.

7.1 Clash Chain

The first real-world implementation of Proof-of-Clash is *Clash Chain*, a new, highly scalable blockchain unleashing the power of *Clash*. It can handle up to 10000 c/s (clashes per second) at the current stage (unoptimized). [9]

7.2 Meta chains

As Ethereum has been suffering from scaling issues, one of the most promising proposals to solve that is sharding ([6], [13], [11], [1]). However, sharding is non-trivial, so the actual implementation of it into the Ethereum network takes time. One of the proposals towards sharding is to re-synchronize individual sharded chains through a master chain which provides a consensus layer for these shards.

The PoC mechanism can be utilized to create consensus between multiple, arbitrary chains via clashes between these chains. Chains which are successful in clashing will also be able to mine blocks on a cross-chain meta consensus layer (CCMCL). Implementations of CCMCL will emerge quickly once implementation of the generic protocol (e.g. 7.1) have proven its viability.

7.3 Gaming

Actually, we've come up with that concept in order to familiarize you, dear reader, with the general idea of clashing between multiple blockchains and their tokens in a gamified setting. If you want to learn more, check out www.chainclash.com ([8].

8 Clashing

The following is a sample listing of the clash code.

```
function clash(_otherParticipantID) {
  int r1 = ranking(myID)
  int r2 = ranking(_otherParticipantID)
  clash = Clash{
    attacker = _otherParticipantID;
    defender = _otherParticipantID;
    r1 = r1;
    r2 = r2;
    blockNumber  = currentBlockNumber;
```

```
  }
  enqueue(clash)
}

function reveal(clashID) {
  Clash clash = getClash(clashID);
  require(!clash.wasRevealed);
  require(currentBlockNumber > clash.blockNumber);
  determineWinner(clash);
}
```

References

[1] Douglas Adams. *The Hitchhiker's Guide to the Galaxy*. 1995.

[2] Iddo Bentov et al. "Proof of Activity: Extending Bitcoin's Proof of Work via Proof of Stake [Extended Abstract]y." In: *SIGMETRICS Performance Evaluation Review* 42.3 (2014), pp. 34–37. URL: http://dblp.uni-trier.de/db/journals/sigmetrics/sigmetrics42.html#BentovLMR14.

[3] Stephan Merz Bernadette Charron-Bost. "Formal Verification of a Consensus Algorithm in the Heard-Of Model". In: (2009). Ed. by xx.

[4] Ronald Cramer, Ivan Damgard, and Jesper Buus Nielsen. *Secure Multiparty Computation and Secret Sharing - An Information Theoretic Appoach*. 2012. URL: http://cs.au.dk/~ivan/.

[5] Ronald Cramer, Ivan Damgård, and Jesper Buus Nielsen. *Multiparty Computation, an Introduction*. Lecture note, 2009. URL: http://www.daimi.au.dk/~ivan/.

[6] Sébastien Forestier. "Blockclique: scaling blockchains through transaction sharding in a multithreaded block graph." In: *CoRR* abs/1803.09029 (2018). URL: http://dblp.uni-trier.de/db/journals/corr/corr1803.html#abs-1803-09029.

[7] Peter Gazi, Aggelos Kiayias, and Alexander Russell. "Stake-Bleeding Attacks on Proof-of-Stake Blockchains." In: *IACR Cryptology ePrint Archive* 2018 (2018), p. 248. URL: http://dblp.uni-trier.de/db/journals/iacr/iacr2018.html#GaziKR18.

[8] CHAINWISE Group. *Chain Clash: Battle of the Crypto Clans*. URL: https://chainclash.com.

[9] CHAINWISE Group. *Clash Chain: A next-generation blockchain implementation*. URL: https://clashchain.com.

[10] Satoshi Nakamoto. *Bitcoin: A peer-to-peer electronic cash system*. 2009. URL: http://www.bitcoin.org/bitcoin.pdf.

[11] Nadi Sarrar. "On transaction parallelizability in Ethereum." In: *CoRR* abs/1901.09942 (2019). URL: http://dblp.uni-trier.de/db/journals/corr/corr1901.html#abs-1901-09942.

[12] Andrew C. Yao. "Protocols for Secure Computations]". In: 322.10 (1905), pp. 891–921. DOI: http://dx.doi.org/10.1002/andp.19053221004.

[13] Mahdi Zamani, Mahnush Movahedi, and Mariana Raykova. "RapidChain: A Fast Blockchain Protocol via Full Sharding." In: *IACR Cryptology ePrint Archive* 2018 (2018), p. 460. URL: http://dblp.uni-trier.de/db/journals/iacr/iacr2018.html#ZamaniMR18.