On Consensus in Public Blockchains

Wenbing Zhao
Cleveland State University
2121 Euclid Ave.
Cleveland, OH 44115
+1 216 523 7480
w.zhao1@csuohio.edu

Shunkun Yang
Beihang University
School of Reliability and Systems Engineering
37 Xueyuan Rd, Beijing, 100191
ysk@buaa.edu.cn

Xiong Luo
University of Science and Technology Beijing
Beijing Key Lab of Knowledge Engr. for Materials Sci.
xluo@ustb.edu.cn

ABSTRACT
In this paper, we formulate the consensus problem and its solutions as seen in public blockchains and contrast them to those of the traditional distributed consensus. The Proof of Work (PoW) algorithm introduced in Bitcoin provides the first elegant and practical solution for reaching distributed consensus in a large-scale untrusted environment. Due to the enormous energy cost of PoW, many alternative consensus algorithms have been proposed for public blockchains aiming at drastically reducing the energy consumption for reaching consensus. We examine four blockchain consensus algorithms, namely Proof of Work, Proof of Stake, Proof of Space, and Proof of Elapsed Time, with respect to the consensus model we have formulated and point out the challenges of adopting them in public blockchains.

CCS Concepts
• Computing methodologies → Distributed computing methodologies → Distributed algorithms

Keywords
Blockchain; distributed consensus; proof of work; proof of stake; proof of stake;

1. INTRODUCTION
Distributed Consensus has been under intense study since early 1980s [1]. Over the years, we have gained important insight on issues related to distributed consensus as demonstrated in several seminal works, including the Byzantine generals problem [2], the FLP impossibility result for asynchronous distributed systems [3], group communication systems [4-6], practical Byzantine fault tolerance [7], and the Paxos consensus algorithm [8,9]. In essence, a distributed consensus consists of two key steps: (1) a propose step, where one or more members would *propose* a value for consensus; and (2) a voting step, where members of the system would *choose* a value for consensus. Note that only a value that has been proposed by a member can be chosen to prevent trivial solutions for the consensus problem.

Common to all traditional solutions to the problem of distributed consensus is a logical structure of the membership where every member in the system knows its logical position. Furthermore, typically one of the members carries a (temporary) leadership role with additional responsibility, which is often referred to as the primary. This primary bears the responsibility of proposing a value, and it is initially designated based on the membership logical formation, and later elected in case the previous primary is suspected to have failed. We also note that which member is eligible to be elected is also based on its logical position.

ICBCT'19, March 15–18, 2019, Hawaii, HI, USA.

DOI: http://dx.doi.org/10.1145/3320154.3320162

The release of Bitcoin in early 2009 not only disrupted the financial sector as the first ever digital cryptocurrency, the blockchain technology underlying Bitcoin offers a non-traditional but practical way of achieving a probabilistic distributed consensus in a very large-scale and untrusted network. Doing so in such a scale has never been accomplished before.

The key to achieving a distributed consensus practically in a large-scale untrusted network as in Bitcoin is the Proof of Work (PoW) algorithm [10]. In this algorithm, the strict logical structure on the membership is not imposed, in particular, no member is designated as the primary simply because of its logical position in the membership. Instead, the distributed consensus problem is converted into a lottery process, where in most cases, there is only a single winner, and this winner gets to propose the new value (i.e., create the new block of transactions). It could occur that two or more winners are present, in which case, the conflict is resolved asynchronously, in a way similar to what has been done in optimistic replication [11-16]. That is, PoW consensus does not guarantee that there is at most one value to be chosen at any given time. However, should inconsistency occur, it will be resolved eventually.

To offer a fair playing ground to all members so that everyone has a chance to win the lottery and no one can cheat, PoW makes everyone solve a CPU-bound puzzle, and the first one who solves the puzzle becomes the lottery winner. This design is elegant and sound in security, however, it leads to a huge cost in energy consumption. If there are 1,000 members (referred to as miners in Bitcoin) competing to certify the new block, then 999 of them will waste their computation. This is equivalent to 0.1% energy efficiency (or 99.9% waste) for reaching a consensus!

To address this issue, several alternative blockchain consensus algorithms have been proposed, such as Proof of Stake [17], Proof of Space [18], Proof of Activity [19], Proof of Elapsed Time, Proof of Space Time, and Proof of Burn. Unfortunately, many of these are not well documented, and some have no publicly available implementation.

In this paper, we formulate a model for this type of large-scale distributed consensus problem as we have learned from PoW implemented in Bitcoin. This type of blockchains is often referred to as public or permissionless blockchain because anyone could join the network with sufficient equipment and minimum

credential. This is in contrast to private blockchains, whose membership is tightly controlled by one or few institutions and the scale of the network typically is much smaller. The concerns and requirements for reaching consensus in private blockchains are totally different from those for public blockchains.

Furthermore, we examine four blockchain consensus algorithms, namely Proof of Work, Proof of Stake, Proof of Space, and Proof of Elapsed Time, with respect to the consensus model we have formulated and point out the challenges of adopting them in public blockchains.

2. BACKGROUND AND RELATED WORK

In blockchain consensus, we see the integration of two important research fields: (1) distributed systems, and (2) computer/network security. As we have elaborated in the previous section, distributed consensus is a central issue in distributed systems research and has been studied for many years. Yet, there lacked a practical solution for reaching consensus in large-scale networks. In the meantime, the security community has been exploring ways to protect shared resources. The basic idea is to ask whomever wishes to access the shared resource to dedicate a non-trivial amount of previous resources of its own before the access is granted. The first such algorithm is the PoW proposed by Dwork and Naor in 1992 to reduce email spams where every email must carry the solution to a puzzle that is hard to compute but easy to verify [20]. Later on, similar algorithms have also been proposed. One such example is Proof of Space [18].

The integration of Proof-of-* line of research for the purpose of reaching distributed consensus is in fact not trivial, as we will elaborate in the next section. Nevertheless, it is exciting to see the integration of the research outputs from two very different research communities.

3. PUBLIC BLOCKCHAIN CONSENSUS

Public blockchains require building consensus in a large-scale untrusted network. Traditional distributed consensus approaches are not appropriate in this context because they all inevitably impose a strict logical structure on the members of the system, which is not a practical requirement for public blockchains. Based on the experiences we learned from PoW-based consensus in Bitcoin, we formulate the requirements for reaching consensus in public blockchains.

- *New block creation*: Typically, there is only a single member that could manage to create a new block.
- *New block verification and propagation*: The new block can be efficiently verified, and only valid new blocks are propagated throughout the network.
- *Conflict resolution*: In the rare case where two or more competing blocks are proposed, which would cause the fork of the blockchain, only one of the branches will eventually be selected as the correct chain.

The first requirement is necessary because members independently select transactions to form each new block. If there are two or more members that could form new blocks, the new blocks are likely to be different. The second requirement is essential to ensure the integrity and viability of operating a large-scale network. While the creation of a new block can be expensive, the verification of the new block must be efficient. One essential requirement for verification is that the verification must be *non-interactive*, that is, a member who wishes to verify the validity of the new block must not have to explicitly challenge the member who created the block because doing so would make the system incredibly non-scalable.

Because it is not practical to use traditional conservative consensus algorithms in public blockchains, the creation of two or more competing blocks is inevitable, even though it should occur rarely. This conflict (i.e., inconsistency) must be resolved eventually so that the network choses only a single block as the successor for the blockchain.

Furthermore, a public blockchain consensus algorithm must also ensure the following properties on new block creation:

- *Freshness*: Regardless of what algorithm is used, a member must dedicate precious resources in order to participate the new block creation process. To ensure a fair competition on new block creation, the resources for new block creation must be fresh for each new block, otherwise, the cost of participating in the process would be significantly reduced in the long run if resources can be reused.
- *Unpredictability*: No one can predict which member would solve the puzzle for the next new block. This *unpredictability* is essential for the integrity of the network because if it is known a member could be the one who decides on the next new block, then this member could be corrupted by bribes or be threatened by others to include questionable transactions that would lead to double-spending attacks, or to exclude legible transactions that would lead to denial of service attacks.

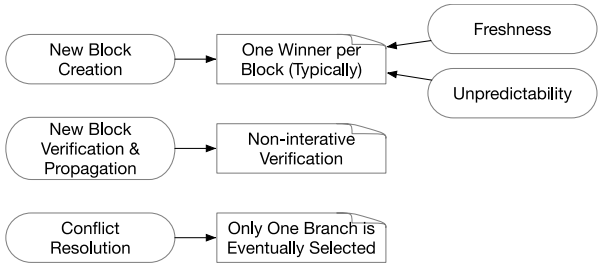

Figure 1. Requirements for public blockchain consensus.

In the following subsections, we examine several blockchain consensus algorithms with respect to the requirements for public blockchain consensus as illustrated in Fig. 1.

3.1 Proof of Work

Two major cryptocurrencies, Bitcoin and Ethereum use PoW to achieve consensus on new block creation. The basic idea of PoW is to let all members compete to solve a CPU-bound puzzle and whomever solves the puzzle first gets to create the next new block, which is later verified by all other members. This is sort of like winning the lottery due to the unpredictability of the puzzle design.

The puzzle in Bitcoin PoW is entirely based on the hash of the block header, which makes it possible to use application-specific integrated circuit to drastically speed up the puzzle solving process. To avoid this problem, Ethereum made the puzzle both CPU-bound and memory-bound by randomly selecting nodes from a pre-prepared directed acyclic graph into the hash materials (https://github.com/ethereum/wiki/wiki/Ethash).

The ensure the *freshness* of the puzzle, in both Bitcoin and Ethereum, members are asked to find a random number N such that the secure hash of the new block where N is part of the block header is smaller than a predefined value. Because the block must consist of new transactions, this prevents any member from starting the computation sooner than others, which ensures a fair playing ground for all members.

The use of secure hash function in PoW ensures the *unpredictability* of the puzzle solving. Although the probability of becoming the winner increases with the computation power, the member who has the most computation power does not necessary win in every new block creation due to the nature of secure hash functions.

In Bitcoin, the difficulty of the puzzle is adjusted so that on average one can solve the puzzle in about 10 minutes. Ethereum is much more aggressive and sets this average time to 12 seconds (according to https://github.com/ethereum/wiki/wiki/Mining). Hence, the probability of two members solving the puzzle concurrently in one second in Bitcoin is 1/(600x600)=1/360,000, and the probably in Ethereum is 1/(12x12)=1/144. Conversely, there will be a collision in Bitcoin every 360,000 seconds (or about 4.2 days) and a collision in Ethereum every 144 seconds (or 2.4 minutes).

In Bitcoin, conflict resolution is done asynchronously. Due to the randomness of the PoW competition, sooner or later one of the competing branches will be longer (or contains the greater cumulative difficulty to be more precise) than others, and this branch will be chosen as the main chain. In its current implementation, Ethereum adopts a Proof o Stake algorithm to resolve any conflicts in a separate finality step [19], which we will elaborate in the next subsection.

Verification in PoW is highly efficient due to the design of hash-based puzzle. Any member can apply the hash on the block header to see if the hash value achieves the designated difficulty level.

3.2 Proof of Stake

Proof of stake (PoS) was proposed as a way to reduce the huge energy cost of solving CPU-bound puzzles in Proof of Work. The basic idea is to select members based on their stake in the cryptocurrency. Those who have more stake presumably have incentive to play fair and do not cause harm to the system. The Proof-of-Stake FAQ for Ethereum (https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQs) has argued for the benefits beyond energy cost savings. However, its actual implementation in blockchains is very limited because of the difficulty in supporting new block proposal. In Ethereum, PoS is used only in the finality stage where a few members (referred to as validators) who are selected according to their stake would vote which branch should join the main chain.

According to the Proof of Stake FAQ in Ethereum, "*the algorithm pseudo-randomly selects a validator during each time slot, and assigns that validator the right to create a single block*." (note that this is not implemented in Ethereum) Other than the lack of details (such as how exactly a particular validator is selected), this inevitably imposes a strict logical structure on the membership of the network, which is by itself not a practical requirement. Furthermore, it assumes a strongly synchronous system model, which is not only not suitable for a large-scale network where members' clocks are not synchronized, but very dangerous from the security perspective because denial of service attacks may be used to delay members from sending and/or receiving messages. The lack of synchronization between members and the presence of asynchrony in the network would inevitably lead to two or more members all think they should be the proposers of the next new block.

Another tradeoff of using PoS is that a member could vote for two or more branches concurrently due to the removal of CPU-bound puzzle solving requirement. Accordingly, Ethereum PoS designed appropriate conditions on slashing the stake invested by members who violated the conditions [21].

3.3 Proof of Space

Proof of Space (PoSp) was proposed as an alternative to PoW prior to the creation of the Bitcoin, the first cryptocurrency [11]. Similar to the pre-Bitcoin Proof-of-* design, which all concern how to make sure someone has dedicated sufficient resources before the access is granted. These algorithms describe a protocol between a prover and a verifier where the verifier challenges the prover to make sure it indeed has the resources claimed by the prover. For PoSp, the resource is disk space. Hence, PoSp has been under discussion as a candidate for replacing PoW to save energy cost in cryptocurrencies. However, we see enormous obstacles for doing this.

PoSp asks the prover to demonstrate that it indeed dedicated sufficient disk space to store the necessary information as requested by the verifier. This requires an initialization phase where both the prover and the verifier could deploy the data structure (typically a hard-to-pebble graph), and a challenge-response phase, where the verifier would challenge the prover on the accessibility of certain information in the graph.

The biggest challenge of adapting pre-Bitcoin Proof-of-* algorithms for blockchain consensus is that it is insufficient to ask a member to pay (i.e., dedicate precious resources, such as CPU, disk, or stake) to participate in the system. While this requirement undoubtedly enhances the difficulty of a malicious players to disrupt the operation of the system, it does not directly solve the distributed consensus problem. PoW happens to be a great fit for distributed consensus because the puzzle can be made so difficult that it is highly likely that only one member could solve the puzzle at a time for each new block. Other Proof-of-* algorithms lack this characteristic, which is the case for PoSp as well as PoS.

Besides the difficulty of creating a new block, even if we employ PoSp only in the finality stage, there are still serious challenges. First, to ensure the freshness of each round, new data structures must be deployed at every validator. How to do this efficiently is unclear. Second, verification in PoSp is interactive in that the prover must respond to every challenge. Considering that scale of the public blockchain networks, this is not practical either.

3.4 Proof of Elapsed Time

Proof of elapsed time (PoET) is based on the secure execution hardware platform offered by Intel. Other than being a hardware-specific solution, PoET is actually a very elegant algorithm to achieve blockchain consensus with much less energy consumption than PoW. The detailed specification of the algorithm is available at the hyperledger website (https://www.hyperledger.org/).

The competition to become the one who gets to create the next block is determined randomly by the wait time. Every member asks the trusted execution environment (referred to as an enclave) to wait for a random amount time that follows a random distribution with a predefined mean. The member who has the least wait time becomes the winner. This design satisfies both the

freshness and the unpredictability requirements for new block creation.

The verification is done via attestation, again offered by the Intel trusted execution environment, which is non-interactive and efficient. Here attestation means that the member will provide sufficient information (in particular a wait certificate) so that others can verify that the certificate was created within the enclave and that the member has waited the allotted time.

The PoET specification does not mention how to resolve conflict. Presumably, conflicts can be resolved in a similar fashion as that in PoW.

It is worth noting that PoET has careful design on the mean wait time and the detection of members who has been winning the election too frequently. The mean wait time is calculated based on the membership size and the target wait time. The target wait time is determined based on the network characteristics so that it is much larger than the message propagation time. The mean wait time is the product of the target wait time and the membership size. Furthermore, z-test is used to determine if a member wins the election too frequently to help detect cheaters.

4. CONCLUSION

In this paper, we presented our view of the consensus problem in public blockchains. We formulated a set of essential requirements for blockchain consensus algorithms, which include the freshness and unpredictability of new block creation, new block verification and conflict resolution in case of forks. Then we examined four blockchain consensus algorithms and point out the challenges of implementing some of them in public blockchains.

We should note that blockchain is a disruptive technology that builds up consensus and trust in a large-scale untrusted environment. It is much more significant than being the enabling technology for cryptocurrency. Blockchains are being used in almost all sectors for the next generation killer apps [22], from supply chain management to healthcare, from clinical trial to Internet-based voting [23]. A low-cost, yet trustworthy consensus mechanism will help make the blockchain technology reach its full potential.

5. ACKNOWLEDGMENTS

This work was supported in part by the National Natural Science Foundation of China [grant 61672080], JWYY [grant 41402020502], and NASFC [grant 2016ZD51031].

6. REFERENCES

[1] Zhao, W. 2014. *Building dependable distributed systems*. John Wiley & Sons.

[2] Lamport, L., Shostak, R., & Pease, M. 1982. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems* 4, 3 (1982), 382-401.

[3] Fischer, M. J., Lynch, N. A., & Paterson, M. S. 1982. Impossibility of distributed consensus with one faulty process (No. MIT/LCS/TR-282). Massachusetts Inst of Tech Cambridge Lab for Computer Science.

[4] Birman, K. P., & van Renesse, R. (Eds.). 1994. *Reliable distributed computing with the Isis toolkit*. Los Alamitos: IEEE Computer society press.

[5] Moser, L. E., Melliar-Smith, P. M., Agarwal, D. A., Budhia, R. K., & Lingley-Papadopoulos, C. A. 1996. Totem: A fault-tolerant multicast group communication system. *Communications of the ACM*, 39, 4 (1996), 54-63.

[6] Zhao, W., Melliar-Smith, P. M., & Moser, L. E. 2012. Low latency fault tolerance system. *The Computer Journal*, 56, 6 (June 2012), 716-740.

[7] Castro, M., & Liskov, B. 2002. Practical Byzantine fault tolerance and proactive recovery. *ACM Transactions on Computer Systems* 20, 4 (Nov. 2002), 398-461.

[8] Lamport, L. 2001. Paxos made simple. *ACM Sigact News*, 32, 4 (2001), 18-25.

[9] Zhao, W., Zhang, H., & Chai, H. 2009. A lightweight fault tolerance framework for web services. *Web Intelligence and Agent Systems: An International Journal*, 7, 3 (2009), 255-268.

[10] Nakamoto, S. Bitcoin: A peer-to-peer electronic cash system. http://www.bitcoin.org.

[11] Zhao, W. 2016. Optimistic byzantine fault tolerance. *International Journal of Parallel, Emergent and Distributed Systems* 31, 3 (2016), 254-267.

[12] Zhao, W. 2014. Application-aware byzantine fault tolerance. In *Proceedings of the IEEE 12th International Conference on Dependable, Autonomic and Secure Computing* (Dalian, China, August 24-27, 2014). IEEE, 45-50.

[13] Babi, M., & Zhao, W. 2017. Towards Trustworthy Collaborative Editing. *Computers* 6, 2 (March 2017), 13.

[14] Zhao, W. 2016. Performance optimization for state machine replication based on application semantics: a review. *Journal of Systems and Software*, 112 (February 2012), 96-109.

[15] Zhang, H., Chai, H., Zhao, W., Melliar-Smith, P. M., & Moser, L. E. 2012. Trustworthy coordination of Web services atomic transactions. *IEEE Transactions on Parallel and Distributed Systems*, 23, 8, (2012), 1551-1565.

[16] Chai, H., Zhang, H., Zhao, W., Melliar-Smith, P. M., & Moser, L. E. 2013. Toward trustworthy coordination of Web services business activities. *IEEE Transactions on Services Computing*, 6, 2, (2013), 276-288.

[17] User "QuantumMechanic" et al. 2011. Proof of stake instead of proof of work. Bitcoin forum thread. https://bitcointalk.org/index.php?topic=27787.0.

[18] Dwork, C., & Naor, M. 1992. Pricing via processing or combatting junk mail. In *Proceedings of the Annual Cryptology Conference* (Santa Barbara, CA, USA, August 16-20, 1992). Springer, Berlin, Heidelberg, 139-147.

[19] Dziembowski, S., Faust, S., Kolmogorov, V., & Pietrzak, K. 2015. Proofs of space. In *Proceedings of the Annual Cryptology Conference* (Santa Barbara, CA, USA, August 16-20, 2015), Springer, Berlin, Heidelberg, 585-605.

[20] Bentov, I., Lee, C., Mizrahi, A., & Rosenfeld, M. 2014. proof of activity: Extending bitcoin's proof of work via proof of stake [extended abstract]. *ACM SIGMETRICS Performance Evaluation Review*, 42, 3 (2014), 34-37.

[21] Buterin, V., & Griffith, V. 2017. Casper the friendly finality gadget. arXiv preprint arXiv:1710.09437.

[22] Andoni, M., Robu, V., Flynn, D., Abram, S., Geach, D., Jenkins, D., McCallum, P. & Peacock, A. 2019. Blockchain technology in the energy sector: A systematic review of

challenges and opportunities. *Renewable and Sustainable Energy Reviews*, 100, (February 2019), 143-174.

[23] Akbari, E., Wu, Q., Zhao, W., Arabnia, H. R., & Yang, M. Q. 2017. From Blockchain to Internet-Based Voting. In *Proceedings of the International Conference on Computational Science and Computational Intelligence* (Las Vegas, NV, USA, December 14-16, 2017). IEEE, 218-221.