(048)

A Framework for Proof-of-Useful-Work Consensus Protocol

O. Nešković [1], P. Sekešan [1], D. Ramljak [2], B. Sharma [2], M. Todorović [3] and T. Davidović [3]
[1] University of Belgrade, Faculty of Mathematics, Studentski trg., 16, 11000 Belgrade, Serbia
[2] Penn State Great Valley, 30 East Swedesford Rd, 54321, 19355 Malvern, PA, USA
[3] Mathematical Institute, Serbian Academy of Science and Arts, Kneza Mihaila 36,
11000 Belgrade, Serbia
Tel.: + 381112630170, fax: +381112186105
E-mail: tanjad@mi.sanu.ac.rs

Summary: We propose a framework for the implementation of Proof-of-Useful-Work (PoUW) Blockchain (BC) Consensus Protocol (CP) preserving the main advantages related to improving the energy efficiency of BC systems. The idea in the development of this CP was to replace solving cryptographic puzzles by solving instances of some hard real-life combinatorial optimization problems. In such a way, the computations that are useful for some participants would be performed during the BC system maintenance. We described the components of the proposed CP and the implementation issues that we identified. Special attention is paid to the efficient utilization of the various types of resources owned by BC participants, incentives and rewarding schemes for BC participants and prevention of various types of fraud.

Keywords: Blockchain systems, Exploration of resources, Real-life optimization problems, Optimization methods, Rewarding schemes, Malicious behavior.

1. Introduction

Unsupervised maintenance of BlockChain (BC) systems includes the application of Consensus Protocols (CPs) which are the most computation-intensive parts of BC systems. The main issue related to the execution of CPs is that they require a huge amount of energy. There exist multiple ways to address this issue [5, 11]. We focus on Proof-of-Useful-Work (PoUW) CPs [1, 2, 4, 8, 10] which were envisioned as a way to overcome this problem. PoUW assumes that within mining process additional useful work is completed for the same number of resources (energy) used. In such a way, the computations that are useful for wider community, solving instances of some hard real-life Combinatorial Optimization (CO) problems, would be performed during the BC system maintenance.

The main issue in the implementation of PoUW-based CPs is to ensure the usefulness of the work performed by the miners without violating the security, consistency, reliability, fairness, and immutability of the considered BC system [2, 10]. We present the PoUW-based CP named Combinatorial Optimization-based Consensus Protocol (COCP), in which the useful work consists of solving real-life instances of various CO problems. In addition, we propose some modifications with respect to [2, 10].

We discuss the implementation of COCP that considers efficient use of energy in BC CP based on the PoUW concept because majority of the proposed PoUW approaches [1, 2, 4, 8, 10] do not adequately consider implementation challenges involved in having a functional CP. Studies that do address implementation either consider a private BC, or do not go into specific details in general case. The main implementation issues related to combining BC and CO, are security, fairness, sustainability, scalability, and consistency of maintaining the whole system. We discuss how to resolve them and ensure the benefits of the proposed COCP for all BC participants.

For resolving the security issues, COCP strongly relies on hashing, time-stamping, and proper signing of each contribution [10]. We investigate what needs to be done for COCP to be successfully implemented as the tool for maintenance of the BC systems. Even though our work is based on COCP we strongly believe that other PoUW-based CPs [1, 4, 8] could be implemented using the lessons we learned.

Our main contributions include the extension of the identified comprehensive list of challenges [10] that should be resolved in order to implement a useful, secure, and efficient PoUW consensus protocol, and resolution of some of those challenges. The main part of this paper is devoted to a detailed COCP description and a case study simulation. More precisely, we explain the main steps of the COCP, and provide a simulation using a part of the real-life BC network to illustrate the achieved benefits. Having in mind that this is still work in progress, we present the identified challenges that are still under consideration.

The remainder of this paper is organized in the following way. The main components of COCP and other PoUW-based CPs are described in the next section. Section 3 contains the implementation challenges that need to be resolved to ensure a secure, reliable, fair, and trustful BC system. In addition, case study simulation that illustrates the functioning of the large-scale BC system maintained by COCP is provided. Concluding remarks and directions for future work are given in Section 4.

2. Main Components of PoUW-based CPs

PoUW-based CP proposed in [2, 10] consists of several components (modules) executed by the BC participants when they take various roles (Fig. 1). Some of these modules are standard for any CP, like the submission of transactions (performed by the basic users), composing transactions into block (performed by miners), and verification of the announced block (performed by verifiers). However, some of the participants are required to perform the additional tasks to enable the completion of useful work, i.e., to solve CO problem instances. As it is illustrated in Fig. 1, a PoUW-based category of users (clients) are needed. These are the customers (organizations, companies) that face CO problems in their everyday activities. They are willing to offer a reward to miners who solve their instances. Their role involves submitting instances and retrieving the corresponding solutions.

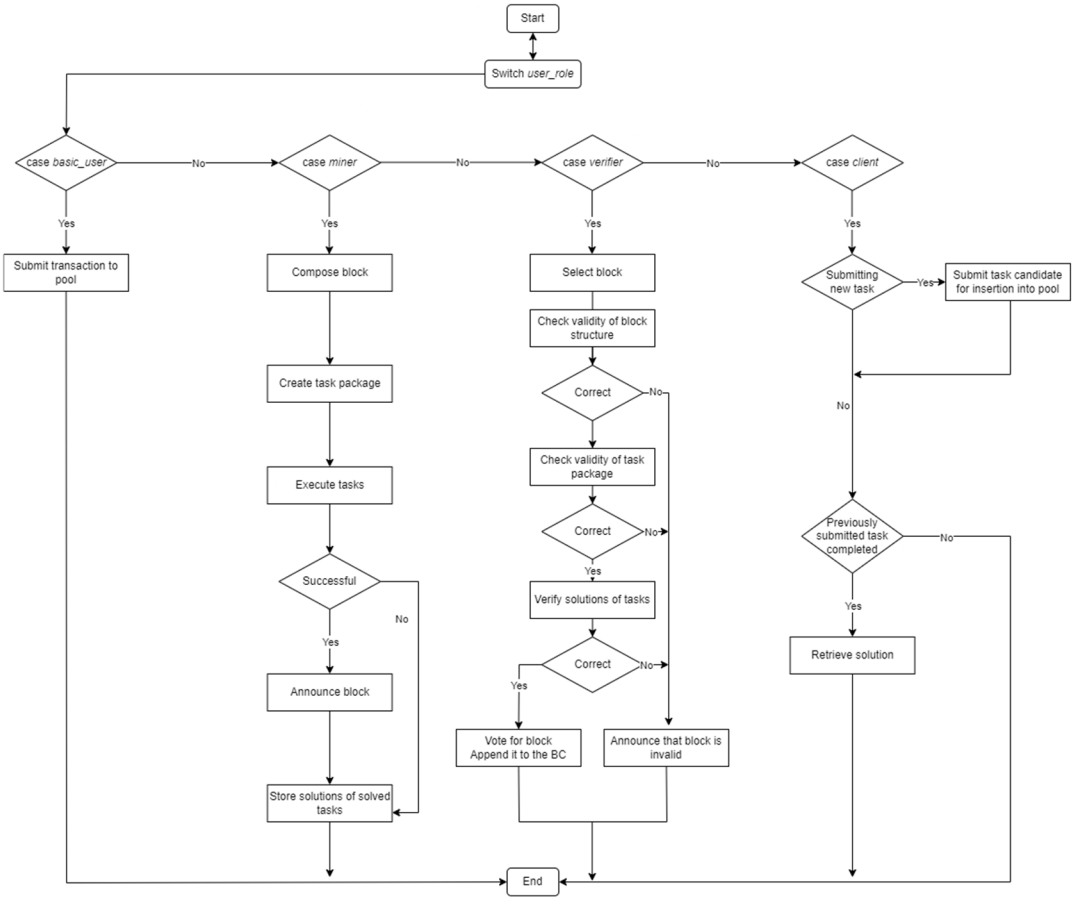

Fig. 1. Activities of BC users in PoUW-based CP.

To ensure the balancing of work for miners and in such a way provide them with similar chances to include a new block into BC system, CO problem instances should be grouped into packages containing instances of various difficulty [6]. After composing a block, miners have to create a package of instances that corresponds to that particular block and to solve all instances before announcing the block. Creation of the package has to be performed in non-biased and verifiable way.

Even if the miner does not succeed in announcing the new block, they can provide solutions for some of the instances and potentially get rewarded for the completed useful work. In such a way, the waste of resource can be reduced [2, 10].

There are some additional tasks for verifiers. Beside validity of the transactions, they need to validate the content of the instance package and solutions generated by the miner. Moreover, checking the validity of solutions not corresponding to the announced blocks is another task that should be performed by verifiers.

3. Implementation Challenges

To implement all the above-mentioned modules in a trustful and reliable way, one should strongly rely on hashing, time stamping, and proper signing each part of provided results [10].

The proposed framework of the implementation is under development. To avoid implementing the whole BC system from scratch, we decided to select an existing BC platform and modify the CP, transactions,

block headers, and verification procedures. COCP requires a platform that supports both standard transactions and smart contracts that are necessary for distributing rewards (collecting payments from customers and rewarding miners that solved the required problem and/or created a new block). Smart contracts might also be a useful feature in future development [3]. This narrows our choice to BC platforms that already support smart contracts [7, 9]. Considering market cap, existing community size, and project maturity, we decided to base the implementation of COCP on Ethereum source code.

However, to be able to complete the implementation, it is necessary to resolve several issues: preventing various types of fraud, ensuring fair balancing of miners' work, defining suitable rewarding schemes, etc. Therefore, challenges that are under consideration are: Instance submission scheme, Instance package creation, Verification of provided instance solutions, and Reward transfer schemes.

To implement a secure Instance submission scheme, clients are required to pay an adequate fee to obtain high-quality solutions for their CO problem instances. However, the reward offered to the miner who solved an instance is less than the provided fee. In such a way malicious users are discouraged in trying to submit instances for which they possibly have valid solutions. In addition, we need to ensure the adequate, fair and secure transfer of reward from the client to the miner for each solved instance. We propose two ways to resolve it: Without smart contracts and Via a smart contract.

Without smart contracts. The instance to be solved is submitted as a single transaction. The transaction has to contain the problem definition (type of problem, address of input data, constraints) in the data field. In addition to the standard fees, the transaction should now include an instance fee. The client sets the instance fee proportional to the estimated difficulty of the submitted instance [6]. If this fee does not exceed the minimal fee required for instance of that difficulty, the transaction is considered invalid and will not be included.

Via a smart contract. An instance would be submitted by sending the transaction to a special smart contract. The consistency of data, especially if the value of the fee is larger than the pre-specified minimum, is automatically checked by the smart contract.

The miner collects transactions into a block in the standard way for the Ethereum protocol. Based on the state of the instance pool (containing unsolved instances that have been submitted) and the structure of the composed block, the difficulty estimation algorithm determines which CO problem instances are to be solved for announcing that block. The hash value of the composed block is used as a seed for the random selection of instances from the pool. This enables easy verification of the proper correspondence between the block structure and the content of the instance package. The number of instances is pre-specified and adjusted periodically to preserve the block mining difficulty. During the mining process, the miner solves the given CO problem instances in accordance with the specified constraints [10]. For the obtained solutions, the hash values are computed and added to the block header (together with the hash value of input data). The miner saves the solutions outside the BC and includes the paths (links, addresses) to this data in the block header. A block is now successfully mined and can be announced for inclusion in the BC. When creating a block, the miner can add their own transaction to claim the rewards for solved instances. It is the responsibility of the block verifiers to check that the transaction added by the miner is valid, i.e., that the miner claimed rewards only for instances that they have solved. Once the block is accepted for the BC, the miner is rewarded for both block creation and solving problems. At this point the miner will be rewarded for all CO problem instances solved during the block creation.

To verify a block in COCP, besides the verification of transactions required by the Ethereum platform, the following must also be checked:

1. If the hash of the data addressed in the blockheader corresponds to the valid solutions of CO problem instances. This is to verify that there is no fraud related to the solution data tampering. More precisely, it should be confirmed that the miner has not changed data after submitting the block or that some other users have not altered data for the miner.

2. If the miner solved the adequate CO problem instances and if the given solutions satisfy the required constraints. For the first part, verifier computes the hash value of the block and (using it as a seed and knowing the state of the instance pool at the time of block creation) generates a sequence of random numbers to determine the set of instances corresponding to the composed block. From the paths and hash values stored in the block header, the verifier can match CO problem instances and their solutions for verification. For completing the second part, verifier should execute the evaluation algorithm on solution data [8] and compare the obtained results with the constraint provided by clients.

An important issue related to the verification concerns the solutions for instances that are solved during the mining of blocks that have not been included in the BC system. As we already mentioned, to reduce the waste of resources, we need to acknowledge as many solutions as possible. The unsuccessful miner may solve several instances from the package and these solutions should be included in the competition for rewarding. This could be realized in such a way that, for each solved instance, the miner creates a transaction containing all data relevant for retrieving and verifying its solution. Once the transaction is verified as a content of an accepted block, the miner could be rewarded for the corresponding work. There are several issues that should be resolved in this process. The main refers to the fact that several miners can provide valid solutions for the same instance and claim the reward. Only the one that offers the best solution should be rewarded.

The second important issue is related to preventing malicious users from collecting reward for solutions that are obtained in some irregular way, for example, that are "stolen" from other users. These issues are still unresolved and are under consideration.

Simulation, which you can find at the following link https://github.com/oneskovic/cocp_simulations, could potentially help understand some of the resolved issues and challenges under consideration. The simulator is a simple environment that simulates strategies of miners on the platform related to setting the difficulty distribution, and to setting the appropriate fee for solving the instances. It allows scalable testing of the implementation without the need to involve any participants.

Inherent assumptions that allow the testing and scalability include the following: I) Each miner follows a strategy consisting of 1) Choose the acceptable difficulty, 2) Find and solve the instance package with that difficulty, 3) Stop when the block insertion time expires, 4) Claim the rewards if any 5) Restart the process from step 2. II) Each miner has a perfect knowledge of difficulties. III) Instance fee is a function of the difficulties only. IV) The probability of miners solving instance packages is proportional to their compute power.

Parameters of the simulation that could be adjusted include the number of instances in the instance pool, the number of instances in the package, miners' computing power, minimum and maximum difficulty that miners will consider solving, and distributions of instances' difficulties (implicitly the instance fees).

One of the most noticeable findings was that distributions of instances' difficulties and block insertion times are the same and that each miner will be able to solve instances according to their computing power. It is important to note that the distribution of block insertion times was not among the parameters of our simulation. This finding indicates that, under the listed assumptions, we can achieve fair mining and control the block insertion time by controlling the distribution of difficulty.

Additional output of the simulation, a pure-strategy equilibrium that results in a "fair" reward distribution is unlikely to exist for large packet sizes. Therefore, prompting a further investigation into mixed strategies.

4. Conclusions

We described the main components of the PoUW-based CP framework, identified the main challenges that should be resolved prior to its implementation. To model user behavior, we simulated an artificial BC system. Apart from the implementation, our future work should involve the extension of the CO problems palette that can be treated within the proposed framework.

Acknowledgements

This work has been funded by the by the Science Fund of Republic of Serbia, under the project AI4TrustBC and Serbian Ministry of Science, Technological Development and Innovations, Agreement No. 451-03-9/2021-14/200029 and supported by Penn State Great Valley Big Data Lab.

References

[1]. M. Ball, A. Rosen, M. Sabin, P. N. Vasudevan, Proofs of Useful Work, *IACR Cryptology ePrint Archive*, https://eprint.iacr.org/2017/203.pdf

[2]. T. Davidović, M. Todorović, D. Ramljak, T. Jakšić-Krüger, L. Matijević, Đ. Jovanović, D. Urošević, COCP: Blockchain Proof-of-Useful-Work Leveraging Real-Life Applications, in *Proceedings of the International Conference on Blockchain Computing and Applications (BCCA'22)*, San Antonio, Texas, USA, 2022, pp. 107-110.

[3]. T. Davidović, M. Todorović, B. Sharma, D. Ramljak, Exploring Arbitrary Real-life Problems in Proof-of-Useful-Work: Myth busting? in *Proceedings of the 5th International Conference on Blockchain Computing and Applications (BCCA'23)*, Kuwait City, Kuwait, 2023, accepted.

[4]. M. Fitzi, A. Kiayias, G. Panagiotakos, A. Russell, Ofelimos: Combinatorial Optimization via Proof-of-Useful-Work A Provably Secure Blockchain Protocol, *IACR Cryptology ePrint Archive*, https://eprint.iacr.org/2021/1379.pdf

[5]. L. Ismail, H. Materwala, A review of blockchain architecture and consensus protocols: Use cases, challenges, and solutions, *Symmetry*, Vol. 11, Issue 10, 2019, 1198.

[6]. U. Maleš, D. Ramljak, T. Jakšić-Krüger, T. Davidović, D. Ostojić, A. Haridas, Controlling the Difficulty of Combinatorial Optimization Problems for Fair Proof-of-Useful-Work-based Blockchain Consensus Protocol, *Symmetry*, Vol. 15, Issue 1, 2023, 140.

[7]. K. Nelaturu, A. Mavridou, E. Stachtiari, A. Veneris, A. Laszka, Correct-by-design interacting smart contracts and a systematic approach for verifying ERC20 and ERC721 contracts with VeriSolid, *IEEE Transactions on Dependable and Secure Computing*, Vol. 20, Issue 4, 2022, pp. 3110-3127.

[8]. N. Shibata, Proof-of-Search: Combining Blockchain Consensus Formation with Solving Optimization Problems, *IEEE Access*, Vol. 7, 2019, pp. 172994-173006.

[9]. H. Taherdoost, Smart Contracts in Blockchain Technology: A Critical Review, *Information*, Vol. 14, Issue 2, 2023, 117.

[10]. M. Todorović, L. Matijević, D. Ramljak, T. Davidović, D. Urošević, T. Jakšić-Krüger, Đ. Jovanović, Proof-of-Useful-Work: Blockchain Mining by Solving Real-Life Optimization Problems, *Symmetry*, Vol. 14, Issue 9, 2022, 1831.

[11]. J. Xu, C. Wang, X. Jia, A survey of blockchain consensus protocols, *ACM Computing Surveys*, Vol. 55, Issue 13s, 2023, 278.