BloHeS Consensus Mechanism – Introduction and Performance Evaluation

Jovan Karamachoski(✉) and Liljana Gavrilovska

Ss. Cyril and Methodius University in Skopje, Skopje, Republic of North Macedonia
jovankaramac@yahoo.com, liljana@feit.ukim.edu.mk

Abstract. Consensus mechanisms are important instruments of Blockchain based systems. The consensus mechanism performance depends on its capability to balance the security, scalability and decentralization of the network. The Proof-of-work and Proof-of-stake are the most accepted consensus mechanisms. However, despite the highest level of protection they are struggling to scale with the increased transaction demand. Comparably, the Tendermint consensus mechanism has better scaling property, but decreased protection capabilities. This paper introduces the BloHeS consensus mechanism that is based on the Tendermint consensus mechanism. The BloHeS is capable to reduce the message complexity, still keeping the protection capabilities on par with the Tendermint consensus mechanism.

Keywords: Consensus mechanism · Tendermint · BloHeS · Message count · Protection capacity

1 Introduction

The problem of scaling the Blockchain networks in terms of transaction throughput, latency, user space and storage is more or less present in all Blockchain technologies. The Blockchain technologies are not one solution to fit all the applications and in the same time obtain optimal operation processes. Certain improvements and optimization can be achieved depending on the implementation scenario. Generally, the scaling problem comes from the requirement for all nodes to have knowledge of common truth and participate in the consensus mechanism. This imposes that the nodes are getting familiar with all transactions that are traversing the network, which generates pressure on the network throughput. The peer-to-peer nature of the Blockchain technologies generates high redundancy of the packages due to the requirement for sharing the common truth between the participants. The package redundancy fills up the network bandwidth especially at the router points.

There are plenty of consensus mechanisms found in the literature. Every consensus mechanism has its strong and weak characteristics. Generally, the consensus mechanism is a trade-off between the transaction throughput, the system security and the network segmentation, which optimization falls under the Blockchain trilemma problem, initially introduced by Vitalik Buterin. As described in [1] the trilemma is not formally defined

D. Perakovic and L. Knapcikova (Eds.): FABULOUS 2022, LNICST 445, pp. 80–94, 2022.
https://doi.org/10.1007/978-3-031-15101-9_6

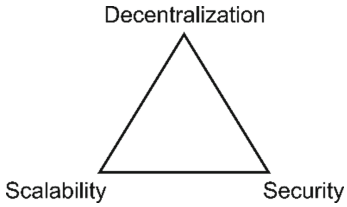

Fig. 1. Blockchain trilemma

in the literature but defines the challenge problem to obtain network's decentralization, scalability and security (see Fig. 1).

The most accepted Blockchain technologies, Bitcoin [2] and Ethereum [3], are implementing Proof-of-work (PoW) [4] consensus mechanism that offers a strong security aspect in a highly decentralized network, but lacks scalability potential for massive adoption. On top of that, the PoW consensus mechanism has compute-intensive algorithm to manage the consensus among the peers in the network, which consumes a lot of electricity, making the PoW an expensive consensus mechanism for Blockchain-based applications.

The main alternative to the PoW is the Proof-of-stake (PoS) [5] consensus mechanism that mimics the PoW consensus achievement, but requires less computation power and consumes less electricity. The PoS consensus mechanism requires the peers to stake a certain amount of assets in order to participate in the process of transaction validation. The whole process of transaction validation with the PoS consensus mechanism finishes faster, compared to the PoW, and gives room for more transactions to pass the validation process. This makes the PoS more scalable than the PoW regarding the transactions throughput. Both, the PoW and PoS are having strong protection capabilities. The networks implementing PoW and PoS can withhold 1/2 of the participants to act maliciously, and still achieve correct consensus over the data circulating in the network.

Another alternative to the PoW and PoS consensus mechanism is the Practical Byzantine Fault Tolerant (PBFT) [6] based consensus mechanisms. The PBFT is the first consensus mechanism that implements the Byzantine Fault Tolerant [7] algorithm in practice. The most prominent algorithms of this family of consensus mechanisms are PBFT and Tendermint [8]. The PBFT-based algorithms are having short voting rounds, which are finalizing in 1–3 s, resulting in faster transaction validation, thus obtaining faster transaction throughput. In the context of the Blockchain trilemma triangle, the PBFT-based consensus mechanisms are inclined toward enhanced scalability, but are lacking decentralization factor and are having reduced security. The protection capability of the PBFT and Tendermint consensus mechanisms are having reduced protection factor compared to the PoW and PoS. Systems implementing PBFT or Tendermint can withhold 1/3 malicious participants in the network. To improve the protection from malicious participants, an additional security measure, like user authentication, may enhance the overall security.

In the same performance range of the PBFT and Tendermint are the Directed Acyclic Graph (DAG) based consensus mechanisms [9]. The DAG-based consensus mechanisms

are managing the consensus over a mesh-chain structure by endorsing past transactions using directional gossiping. The transaction throughput and the protection capabilities are in the range of the PBFT and Tendermint consensus mechanisms. The most prominent Blockchain technologies using the DAG-based consensus achievement are IOTA [10] and Hashgraph [11].

2 BloHeS Consensus Mechanism

The Tendermint consensus mechanism, per definition [8], progresses in rounds, where each round consists of three stages: prevote, precommit and commit. After a successful consensus over a certain voting round, the mechanism increases the height for the next block. Contrary, the unsuccessful voting round will keep the same block height and will enter a new voting round. The Tendermint consensus mechanism diagram in Fig. 2 shows the transition between the stages of the voting round [12]. This simple voting mechanism offers fast block finalization and high transaction throughput, as proven by [13, 14 12]. The Tendermint consensus mechanism can protect the network from 1/3 faulty nodes, which in turn requires the number of validators in the network to be $n = 3 * f + 1$ where f is the maximum number of faulty nodes for a given number of validators [15].

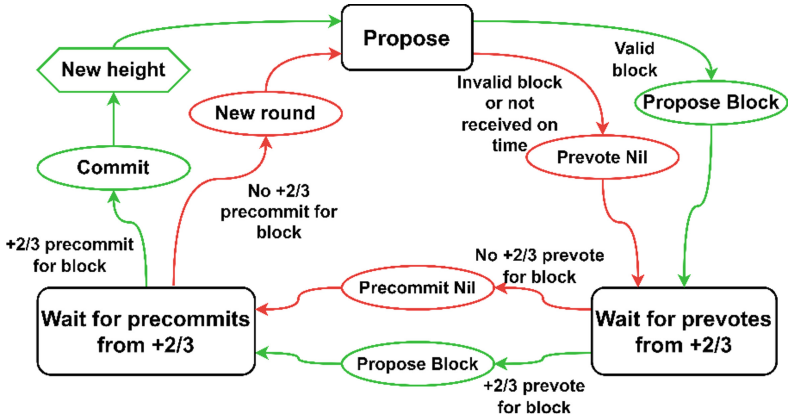

Fig. 2. Tendermint consensus mechanism diagram

The validators in the network are exchanging voting messages extensively, in order to achieve consensus, which creates a message storm between the validators in the network. The validators are voting positively for a block that is correct, and the vote messages are sent to other validators in the networks. Even if the block is incorrect or delayed, the validators will vote with nil votes, which again generates a message storm between the validators. The number of exchanged messages increases exponentially with the increasing number of validators. This affects the network bandwidth and is significant in scenarios with large numbers of validators. The papers [16, 17] are pointing out the problem of the Tendermint consensus mechanism with a large number of validators in the network. Generally, the problem in the scenarios with large numbers of validators is

a result from the limited processing power of the validators, the limited network bandwidth and the increased complexity of the consensus achievement due to the increased number of the exchanged messages among the validators. Because of these limitations, the Tendermint network recommended maximum number of validators is 100, where this number can vary depending on the validators' performance, network bandwidth and transaction load. The recommended maximum number of validators is an approximative value, experimentally determined. It can be more precisely determined according the implementation scenario and participants' performance.

The single-layer architecture of the Tendermint network obviously will not scale accordingly, regarding the number of validators in a large-scale network. In order to make the network to scale, a new dimension for scaling has to be introduced. The BloHeS architecture [18] introduces a multilayer hierarchy that enables the extensive network's scaling for large-scale scenarios. This paper introduces the BloHeS consensus mechanism for the multilayer BloHeS architecture. Figure 3 shows the BloHeS consensus mechanism diagram.

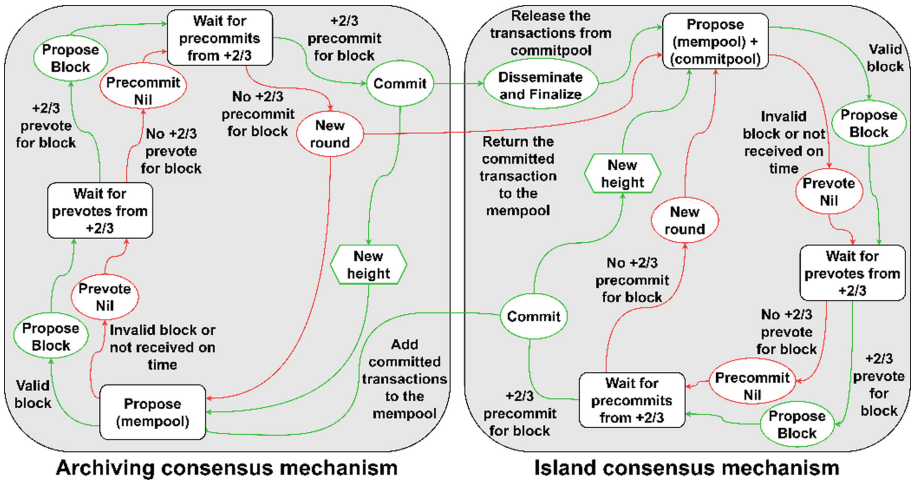

Fig. 3. BloHeS consensus mechanism diagram

The BloHeS consensus mechanisms design is based on the Tendermint consensus mechanism. It considers layered architecture and consists of two interrelated and modified Tendermint consensus mechanisms that are orchestrating the Archiving and Island domains of the BloHeS network. The network organization is presented in Fig. 4. The Island domain is the layer in the BloHeS network built from multiple Islands. An Island is a cluster of validators orchestrating independent Island consensus mechanism, which submits validated blocks for archiving to the validators in the Archiving domain. The Archiving domain is a single cluster of validators in the BloHeS network managing the Archiving consensus mechanism, which is ordering and verifying the records submitted by the Islands.

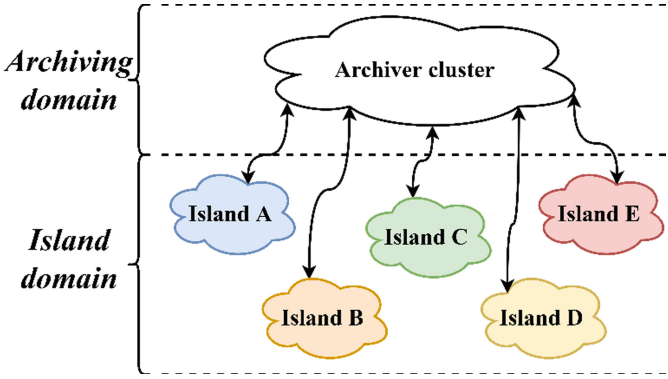

Fig. 4. Network organization of the BloHeS system

In addition to the mempool memory, [19], defined by the Tendermint consensus mechanism, the BloHeS consensus mechanism introduces a commitpool memory. Every Island manages a separate commitpool where the validators are storing the committed blocks until they are committed by the Archiving cluster. The independent consensus mechanisms of the Archiving cluster and the individual Islands are interconnected with forwarding links. The Islands' committed blocks are sent to the Archiving cluster validators, where the block is stored in the mempool of the Archiving cluster validators and will be integrated into the proposal of the next block. The Archiving cluster has two forwarding links to the Islands, depending on the outcome of the validation process of the transactions. If the block in the Archiving consensus mechanism is committed, a notification is forwarded to the corresponding Islands' proposers in order to clear up the transactions from the commitpool and accordingly the mempool of the validators of the Island. If the Archiving cluster validators cannot commit the block, the Archiving cluster proposer informs the corresponding Islands' proposers of the outcome of the validation process and the transactions are cleared from the commitpool of the Islands.

A new stage in the Island consensus mechanism is the dissemination and finalization stage, which occurs after the Islands receive a commit from the Archiving cluster. During this stage, the Island proposers are informing the Island participants to finalize the validation process by inserting the transactions in the Blockchain record and remove the transactions from the commitpool and mempool of the Island validators.

Figure 5 presents the time line of the BloHeS consensus mechanism. It shows that all committed packets from the independent Island are collected in the mempool of the Archiving cluster, in order to be integrated in the next block.

The BloHeS design and the appropriate BloHeS consensus mechanism can be implemented as a particular use case in a public healthcare system. In such case the Archiving cluster consists of dedicated validator nodes, which are under governmental management. Other independent bodies may also participate in this cluster. The validators from the Island domain are the actual healthcare practitioners from the healthcare network. They are major content creators, creating medical records for every visit from the patients, under the patients' consent.

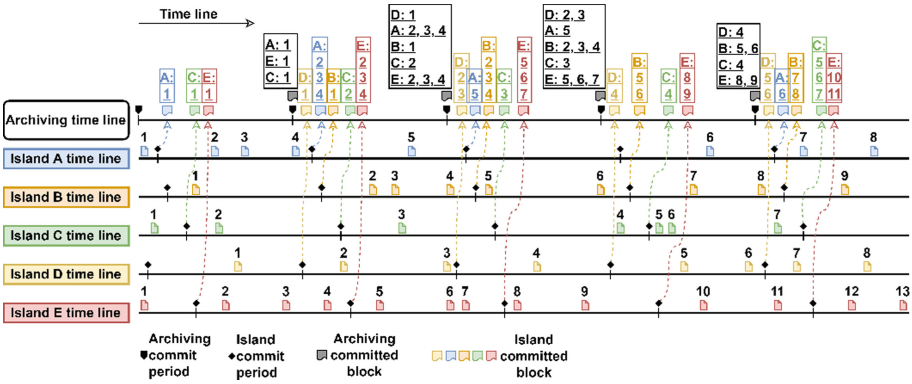

Fig. 5. Time line of BloHeS consensus mechanism

The process of transaction generation and finalization (or archiving of the transaction) starts at the healthcare practitioner's office. The patient visits the healthcare provider and gives permission to the healthcare practitioner to insert new medical record in his medical file. After the record is generated and signed by the healthcare practitioner, the transaction is sent to the Island members for validation. Actually, the validators from the Island are checking the correctness of the transactions through the Island consensus mechanism. After the transaction is validated, proof for transaction validity and its address are inserted in a consolidated transaction that is forwarded to the Archiving cluster. The validators from the archiving cluster are conducting Archiving consensus mechanism to vote for the correctness of the transactions. The output of the Archiving consensus mechanism is a consolidated list of addresses for the transactions in the system in a single consensus cycle. The address of the consolidated list of addresses for the transactions is sent to the appropriate Islands in order to finalize the transactions. The healthcare practitioner receives the finalization signal for the actual transaction and informs the patient.

3 Tendermint and BloHeS Message Exchange Diagrams

For a successful validation process of the Tendermint consensus mechanism, the number of active validators n has to be in the range $[2 * f + 1, 3 * f + 1]$, which results that the number of faulty nodes are in range $[0, f]$, [15]. All active validators contribute with their votes to achieve consensus and finalize the block during the process of consensus achievement. The number of exchanged messages between the validators in every stage of the voting process is called *message count*. The total message count for block finalization in a single round may range between minimum message count M_{min} and maximum message count M_{max}, respectively to the scenarios of f and 0 faulty nodes. The message exchange diagram of the Tendermint consensus mechanism is shown in Fig. 6.

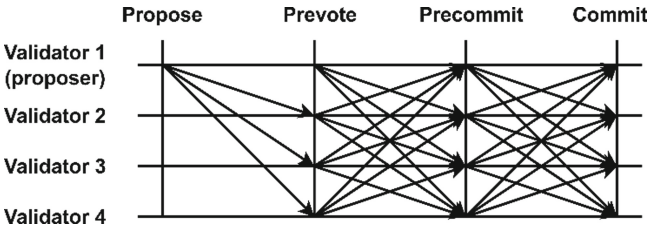

Fig. 6. Tendermint consensus mechanism message exchange

The total message count for a single block finalization in a Tendermint consensus mechanism is the sum of the message count values for three stages of the consensus mechanism. During the propose stage the message count is:

$$M_{propose} = n - 1 = 3f + 1 - 1 = 3f \tag{1}$$

where n is the number of validators in the network and f is the maximum number of the faulty nodes for the given n.

The capability of the consensus mechanism to protect the network from f faulty nodes, gives the possibility to manage correct consensus when the number of correct validators in the network is between $n - f$ and n. Accordingly, the message count for the prevote and precommit phase may range between their minimum and maximum values depending on the number of correct validators in the network. The message count for the prevote and precommit phases are calculated as:

$$M^{min}_{prevote} = M^{min}_{precommit} = (n - 1) * (n - f) = 3f * (2f + 1) \tag{2}$$

$$M^{max}_{prevote} = M^{max}_{precommit} = (n - 1) * n = 3f * (3f + 1). \tag{3}$$

Respectively the formulas for the total minimum message count M_{min} and maximum message count M_{max} are:

$$M_{min} = M_{propose} + M^{min}_{prevote} + M^{min}_{precommit} \tag{4}$$

$$M_{max} = M_{propose} + M^{max}_{prevote} + M^{max}_{precommit}. \tag{5}$$

From the analysis of the message exchange diagram of the BloHeS consensus mechanism (see Fig. 7), it is obvious that the message count consists of messages exchanged inside the Islands or intracluster messages, and messages exchanged between the Islands and the Archiving cluster or intercluster messages.

The total message count M of a single round of the BloHeS consensus mechanism is:

$$M = M_{intracluster} + M_{intercluster}. \tag{6}$$

The *intracluster message count*, $M_{intracluster}$, is:

$$M_{intracluster} = n_i * M_{island} + M_{archiving\ cluster} \tag{7}$$

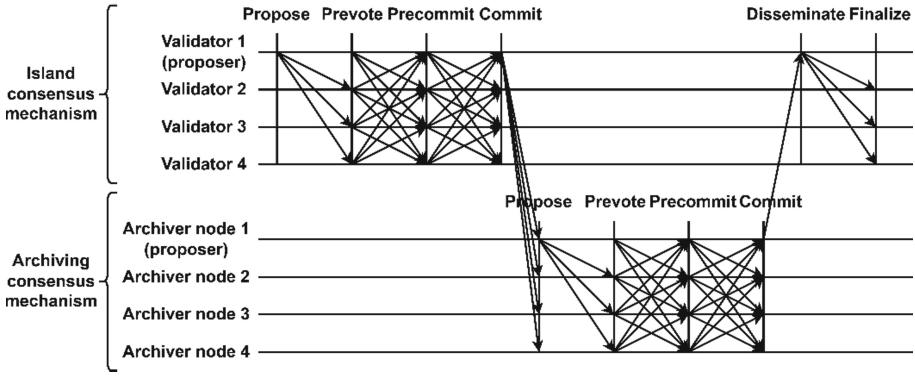

Fig. 7. BloHeS consensus mechanism message exchange

and the *intercluster message count*, $M_{intercluster}$, is:

$$M_{intercluster} = M_{request} + M_{response} = n_a * n_i + n_i. \tag{8}$$

The variable n_i represents the number of Islands in the network and n_a is the number of validators in the Archiving cluster.

The intercluster communication in the Islands has an additional communication stage represented by the dissemination phase, so the message count inside the Island is:

$$M_{island} = M^{island}_{propose} + M^{island}_{prevote} + M^{island}_{precommit} + M_{disseminate}. \tag{9}$$

Further, the message exchange pattern inside the Archiving cluster is the same as the traditional Tendermint consensus mechanism:

$$M_{archiving\ cluster} = M^{arch}_{propose} + M^{arch}_{prevote} + M^{arch}_{precommit} \tag{10}$$

where the actual message count is equal to (4) and (5) respectively, for the minimum message count and maximum message count inside the Archiving cluster.

The message count of the propose, prevote and precommit phases are following the Tendermint consensus mechanism pattern and are calculated as:

$$M_{propose} = n - 1 = 3f + 1 - 1 = 3f \tag{11}$$

$$M^{min}_{prevote} = M^{min}_{precommit} = (n-1) * (n-f) = 3f * (2f+1) \tag{12}$$

$$M^{max}_{prevote} = M^{max}_{precommit} = (n-1) * n = 3f * (3f+1) \tag{13}$$

and the dissemination phase message count is:

$$M_{disseminate} = n - 1 = 3f + 1 - 1 = 3f. \tag{14}$$

The message count of the prevote and precommit phases is also related to the number of correct validators in the consensus mechanism. Accordingly, there are minimum

and maximum message count values. Therefore, the consensus mechanism inside the Archiving cluster and inside the Islands can manage correct consensus between the validators if the number of correct validators ranges between $n - f$ and n, this means that the total message count may range between minimum message count M_{min} and maximum message count M_{max}, respectively. So, the total minimum message count M_{min} and maximum message count M_{max} are:

$$M_{min} = M_{intercluster} + n_i * \left(M_{propose} + M_{prevote}^{min} + M_{precommit}^{min} + M_{disseminate}\right) + M_{archivingcluster} \tag{15}$$

$$M_{max} = M_{intercluster} + n_i * \left(M_{propose} + M_{prevote}^{max} + M_{precommit}^{max} + M_{disseminate}\right) + M_{archivingcluster} \tag{16}$$

where n_i is the number of Islands in the network.

4 Protection Capacity

The novel parameter *protection capacity* P determines the capability of the consensus mechanism to protect the consensus achievement in presence of faulty nodes. The protection capacity is related to the number of faulty nodes that can be endured by the consensus mechanism, while keeping the correctness of the consensus in the system. It is calculated as a sum of the maximum number of faulty nodes that can be tolerated by every cluster in the network:

$$P = \sum_{k=1}^{n_i} f_k \tag{17}$$

where the f_k is the maximum number of faulty nodes that can be tolerated by every Island and n_i is the number of Islands in the network.

The Tendermint network is a single cluster network, which implies the number of Islands to be $n_i = 1$. The Tendermint consensus mechanism by definition can protect the consensus in the system from f faulty nodes that is calculated as:

$$f = \frac{n-1}{3} \tag{18}$$

where n is the number of validators in the Tendermint network [15]. By implementation of this relation in (17), the protection capacity for the Tendermint consensus mechanism will be:

$$P_T = f = \frac{n-1}{3} \tag{19}$$

where P_T is the protection capacity of the Tendermint consensus mechanism.

The network organization of the BloHeS system is multi-clustered. The clusters, also known as Islands, implement the Island consensus mechanism, which is based on Tendermint consensus mechanism so they can achieve the same level of protection on

an Island level. The distinct Islands can provide different protections from faulty nodes due to different sizes. The protection against faulty nodes on an Island level is given with (18). The validators are participating in the Islands by autonomous decision, which creates non-uniform, and random distribution of the validators among the Islands, so the number of faulty node protection on the Island level will be different for different Islands. The non-uniform validator distribution makes the definition of the protection capacity hard. The protection capacity of the BloHeS system is calculated under assumption of uniform validator distribution, i.e.:

- All Island in the network are of the same size;
- The number of faulty nodes in every Island is the same.

Implementing these assumptions in the formula for protection capacity gives:

$$P_B = P_A + n_i * P_i \tag{20}$$

where P_B is the protection capacity of the BloHeS consensus mechanism, P_A is the protection capacity of the Archiver cluster, n_i is the number of Islands in the network, and P_i is the protection capacity of the Island.

5 Performance Evaluation

To determine the improvement of the BloHeS consensus mechanism over the Tendermint consensus mechanism, a comparison of the message count and protection capacity between the two consensus mechanisms is conducted using MATLAB [20] software.

5.1 Evaluation Scenarios

The performance is evaluated over the same-sized networks of validators, respecting the structural difference between two networks implementing Tendermint and BloHeS consensus mechanisms. The number of validators in the network ranges between $V = [1, 1000]$.

Tendermint network is a single cluster network, so the number of Islands equals $n_i = 1$ and the number of validators in the Island is equal to the number of validators in the network $n = V$. The number of validators in the Tendermint scenario forms a discrete set of validators following the relation:

$$n = 3 * f + 1, for \quad f = \left(\frac{n-1}{3}\right) \in \boldsymbol{N} \tag{21}$$

where n is the number of validators in a cluster, f is the maximum number of faulty validators in the network for a given n and $\boldsymbol{N}$ is the set of natural numbers.

In the simulation scenarios for the BloHeS network the Archiving cluster is part of the calculations as a single cluster of 4 validators, $n_a = 1$, and $f = 1$ where $n = 3 * f + 1 = 4$ assuming a fully functional cluster of validators. Furthermore, the distribution of validators in the BloHeS scenario is following the uniformity criterion (see Sect. 4).

Accordingly, the number of Islands n_i in the simulation increases with the increment of the number of validators in the network in discrete steps as:

$$n_i = \frac{V-4}{n} \in \boldsymbol{N} \tag{22}$$

where $n = 3 * f + 1$ is the number of validators in a single Island, V is the number of validators in the network and $\boldsymbol{N}$ is the set of natural numbers.

The maximum number of faulty nodes per Island is $f = \{1, 3, 7, 21\}$ and it determines the evaluated scenarios for the BloHeS consensus mechanism. The number of validators in a single Island is $n = \{4, 10, 22, 64\}$, according (21).

5.2 Evaluation Results

The simulation results for the message count, comparing the Tendermint network and BloHeS network scenarios, are presented in Fig. 8. There is obvious improvement and significant message count reduction in the scenarios where the same numbers of validators in both of the networks are participating in the consensus achievement. The results for the message count of the Tendermint consensus mechanism are showing a faster increment rate than the message count of the BloHeS consensus mechanism, which becomes significant as the number of validators in the network increases. The most significant reduction in message count is achieved in the scenario where the number of validators in the Island is $n = 4$. The message count reduction is less significant as the number of validators in the Island increases. For scenarios of $n = 64$ validators in the Islands, the improvement brought by the BloHeS consensus mechanism is small when there is small number of Islands in the network and the improvement is gaining significance as the number of Islands increases.

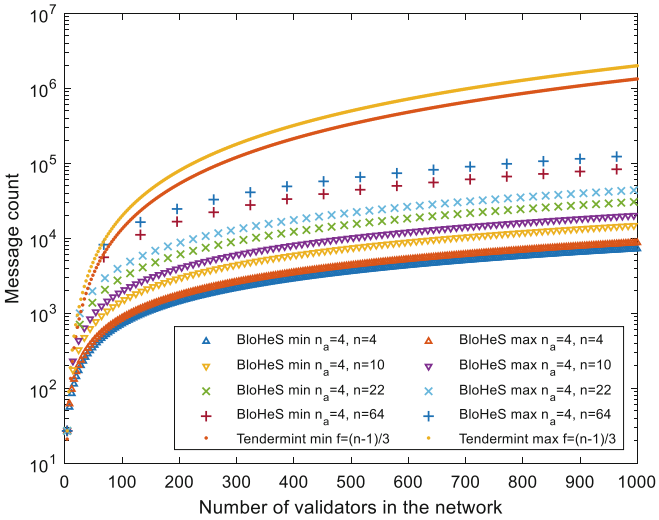

Fig. 8. Message count comparison between Tendermint and BloHeS consensus mechanism

The better performance of the BloHeS consensus mechanism regarding the message count is result of the increased segmentation of the communication in the cluster. The message exchange between the Islands and the Archiving cluster is significantly smaller than the message exchange pattern inside the Islands.

Regarding the simulation of the protection capacity for the Tendermint consensus mechanism the discrete set of the values for number of validators in the network satisfies the formula:

$$P_T = f = \frac{n-1}{3} \in \boldsymbol{N} \tag{23}$$

where f is a maximum number of faulty validators, n is the number of validators in the network and $\boldsymbol{N}$ is the set of natural numbers.

The simulation for the BloHeS consensus mechanism assumes protection capacity of the Archiver cluster $P_A = 1$, protection capacity of the Islands $P_i = \{1, 3, 7, 21\}$, size of the Islands $n = \{4, 10, 21, 64\}$ and discrete set of Islands in the network calculated according to (22).

Figure 9 shows the protection capacity of the BloHeS and Tendermint consensus mechanism. The results for the protection capacity are showing decreased protection against faulty nodes in the networks implementing BloHeS consensus mechanism compared to the Tendermint consensus mechanism. It is important to note, that the protection capacity rapidly increases in the simulation scenarios where the number of validators in the Islands increases. This means that the BloHeS network participants should tend to self-organize in bigger Islands, with an upper limit of around 100 validators per Island, in order to achieve better protection capacity and reduced total message count. The Fig. 9 shows that the protective property of the Tendermint consensus mechanism is better than the BloHeS consensus mechanism, except for marginal values in the scenarios with bigger Islands in the network.

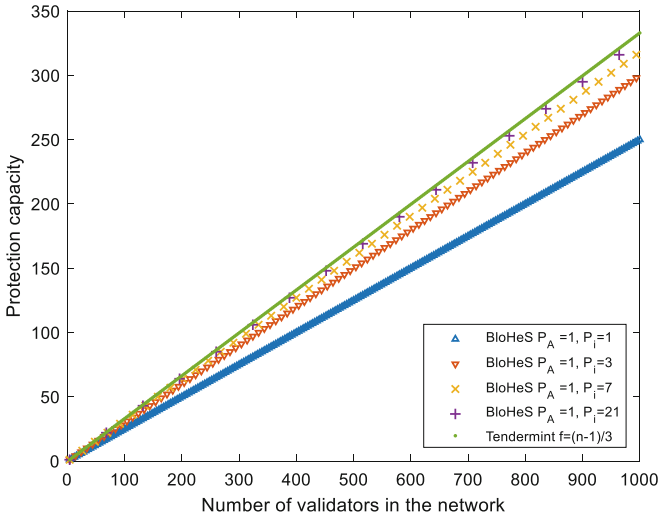

Fig. 9. Protection capacity comparison between Tendermint and BloHeS consensus mechanism

A novel parameter *ratio between the message count and the protection capacity* is determined to provide a sense if the protection gain is sufficient compared to the increased communication complexity of the Tendermint consensus mechanism. Figure 10 presents the results of the ratio between the message count and the protection capacity, where it is obvious that the Tendermint consensus mechanism has a linear increasing factor. This means that the Tendermint's communication complexity increases faster compared to the protection capacity of the Tendermint consensus mechanism when the number of validators in the network increases. The constant ratio between the message count and protection capacity of the BloHeS consensus mechanism shows that there is no increment in the message complexity over the protection capacity as the validators' number increases.

The results in Fig. 9 show that the BloHeS consensus mechanism with bigger Islands gains similar protection capacity with the Tendermint network. Moreover, the ratio between the message count and protection capacity shows that the BloHeS consensus mechanism obtains similar protection capacity to the Tendermint consensus mechanism for less complex message communication.

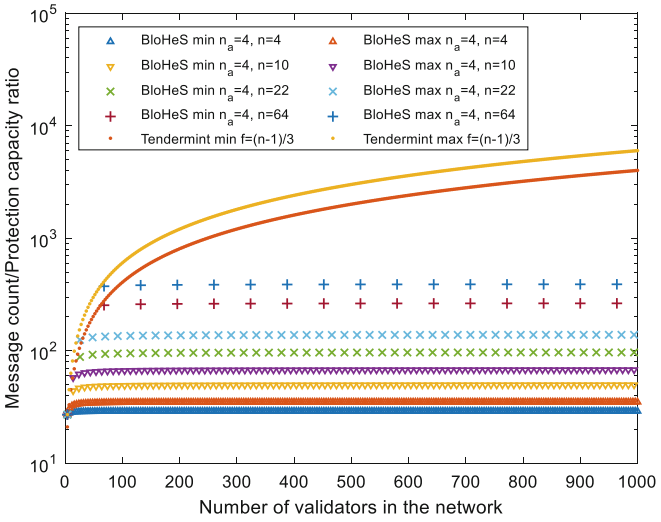

Fig. 10. Message count/Protection capacity ratio comparison between Tendermint and BloHeS consensus mechanism

6 Advantages and Disadvantages of the BloHeS Consensus Mechanism

The main advantage of the BloHeS consensus mechanism is the *reduced message complexity* in the process of consensus achievement. The addition of the new dimension, more precisely the introduction of *multi-layered structure* of the BloHeS consensus mechanism, cuts down the requirements for message exchange with all participants in

the network. This advantage is achieved with the price of a *slight reduction of consensus mechanism protection*, which asymptotically approaches the protection achieved by the Tendermint consensus mechanism, when the number of the validators in the Islands increases.

A main disadvantage of the BloHeS consensus mechanism is the *extended time required for consensus finalization*, due to the requirement for consensus finalization in the two stages of the independent consensus cycles. In the best case scenario the time required for consensus finalization of the BloHeS consensus mechanism will be approximately two-times longer than the Tendermint consensus mechanism.

7 Conclusion

The flat architecture of the general Blockchain technologies shows a scaling problem. Independent of the implemented technology, the requirement for peer-to-peer communication creates a burden for the network expansion. An improvement of the network scaling can be achieved by addition of a new dimension. Accordingly, the transaction throughput of the Tendermint consensus mechanism and the scaling performances can be enhanced.

The novel BloHeS consensus mechanism implements the Tendermint consensus mechanism as a base for the segmented consensus mechanisms in each individual cluster in the network. The two-way forwarding links between the higher-layer Archiving cluster and the lower-layer Islands in the network significantly reduce the need for message exchange for consensus achievement. The communication pattern of the Tendermint consensus mechanism shows an exponential increase in the message count as the number of validators increase. The BloHeS consensus mechanism has a linear increment in the message count achieving significant difference for a high number of validators in the network.

The Tendermint consensus mechanism is capable to protect the consensus achievement process from 1/3 malicious nodes. Due to clustered architecture of the BloHeS consensus mechanism the determination of the protection capabilities is hard to be obtained due to the non-uniform distribution of the validators in the Islands. The determination of the protection capabilities is easier under assumption for uniformity in the validator distribution in the Islands. The protection capacity is introduced, to compare the protection capabilities. The analysis shows smaller protection capacity of the BloHeS consensus mechanism in small Island scenarios. However, for bigger Island scenarios of the BloHeS consensus mechanism asymptotically approaches the Tendermint's protection capacity performance.

The ratio of the message count and the protection capacity gives the relation between the message complexity and the protection capabilities of the consensus mechanism. The linearly increasing ratio of the message count and protection capacity by the Tendermint consensus mechanism shows faster increasing of the message complexity compared to the steady protection capacity increment. This performance of the Tendermint consensus mechanism in large networks will create an unusable network due to extreme message storm generated by the validators. On the other side, the constant ratio of the message count and the protection capacity of the BloHeS consensus mechanism shows

the capability to obtain constant increment of the message complexity by the constant increment of the protection capacity of the BloHeS consensus mechanism. Comparison of the same-sized Tendermint and BloHeS networks (under assumption of big Islands and uniformity for the BloHeS network) shows that both networks will perform with protection capacity of almost the same level, but the Tendermint network will have more complex message pattern and bigger message count than the BloHeS network. That allows the BloHeS network to scale better than the Tendermint network.

References

1. Del Monte, G., Pennino, D., Pizzonia, M.: Scaling blockchains without giving up decentralization and security. arXiv preprint arXiv:2005.06665 (2020)
2. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2008)
3. Buterin, V., et al.: Ethereum white paper: a next-generation smart contract and decentralized application platform. Ethereum (2014). http://blockchainlab.com/pdf/Ethereum_white_paper-a_next_generation_smart_contract_and_decentralized_application_platform-vitalik-buterin.pdf
4. Debus, J.: Consensus methods in blockchain systems. Frankfurt School of Finance & Management, Blockchain Center, Technical report, pp. 1–58 (2017)
5. Nguyen, C.T., Hoang, D.T., Nguyen, D.N., Niyato, D., Nguyen, H.T., Dutkiewicz, E.: Proof-of-stake consensus mechanisms for future blockchain networks: fundamentals, applications and opportunities. IEEE Access **7**, 85727–85745 (2019)
6. Castro, M., Liskov, B., et al.: Practical Byzantine fault tolerance. In: OSDI, pp. 173–186 (1999)
7. Lamport, L., Shostak, R., Pease, M.: The Byzantine generals problem. ACM Trans. Programm. Lang. Syst. (TOPLAS) **4**, 382–401 (1982)
8. Kwon, J.: Tendermint: Consensus without mining. Draft v. 0.6, fall. 1 (2014)
9. He, J., Wang, G., Zhang, G., Zhang, J.: Consensus mechanism design based on structured directed acyclic graphs. Blockchain Res. Appl **2**, 100011–100040 (2021)
10. Popov, S.: The tangle. https://assets.ctfassets.net/r1dr6vzfxhev/2t4uxvsIqk0EUau6g2sw0g/45eae33637ca92f85dd9f4a3a218e1ec/iota1_4_3.pdf
11. Baird, L.: The Swirlds hashgraph consensus algorithm: fair, fast, Byzantine fault tolerance. Swirlds Tech Reports SWIRLDS-TR-2016-01, Technical report (2016)
12. Karamachoski, J., Gavrilovska, L.: Extended performance evaluation of the tendermint protocol. In: ETAI 2021 (2021)
13. Buchman, E.: Tendermint: Byzantine fault tolerance in the age of blockchains (2016). https://allquantor.at/blockchainbib/pdf/buchman2016tendermint.pdf
14. Dib, O., Brousmiche, K.-L., Durand, A., Thea, E., Hamida, E.B.: Consortium blockchains: overview, applications and challenges. Int. J. Adv. Telecommun. **11** (2018)
15. Amoussou-Guenou, Y., Del Pozzo, A., Potop-Butucaru, M., Tucci-Piergiovanni, S.: Dissecting tendermint. In: International Conference on Networked Systems, pp. 166–182 (2019)
16. Kwon, J., Buchman, E.: Cosmos - A Network of Distributed Ledgers. Cosmos, dated, pp. 1–41 (2018)
17. Arora, S.K., Kumar, G., Kim, T.: Blockchain based trust model using tendermint in vehicular adhoc networks. Appl. Sci. **11**, 1998 (2021)
18. Karamachoski, J., Gavrilovska, L.: An optimal storage organization for blockchain-based public healthcare system. J. Electr. Eng. Inf. Technol. **5**, 143–152 (2020)
19. Miletic, L.: Formal and simulation analysis of data dissemination algorithms in a blockchain network (2018)
20. MATLAB website. https://www.mathworks.com/products/matlab.html