Lightweight Consensus Algorithm

Pankaj Kumar Magar (222IT018)
Information Technology
National Institute of Technology,
Karnataka, Surathkal, India 575025
pankajkumarmagar19@gmail.com

Vedant Parwal (222IT034)
Information Technology
National Institute of Technology,
Karnataka, Surathkal, India 575025
vedant2000parwal@gmail.com

Dr. Kiran Manjappa
Information Technology
National Institute of Technology,
Karnataka, Surathkal, India 575025
kiranmanjappa@nitk.edu.in

***Abstract* - Blockchain is an emerging distributed ledger technology that has numerous applications in many areas. To comprehend the significance of blockchain and its applicability, it is crucial to understand its fundamental elements, functional properties, and architectural design. Consensus algorithm is considered as the foundation of blockchain technology that has a large impact on security, efficiency and scalability of the blockchain network. Different algorithms provide the platform different benefits and drawbacks, which might result in significant variances. There have been many different consensus algorithms proposed up to this point, each with their own performance and security. In this article, we attempt to introduce a novel consensus algorithm called a Lightweight consensus algorithm which uses hashing for verification by selecting a subset of nodes. The proposed algorithm allows the validation of transaction as well as blocks while taking less computation time and the probability of attacking is also low. The examination and evaluation of the system's overall performance in terms of memory, computation time, and security aspects reveals a significant improvement.**

Keywords – Blockchain, Consensus, Scalability, Transaction rate, Distributed ledger.

I. Introduction

To overcome the inefficiencies in transactions, blockchain technology was developed. It has completely altered how businesses and industries are organised. Blockchain technology has received a lot of interest recently because of its salient features, including distributed network, security, immutability, trust, and anonymity.

A distributed ledger that is shared across the nodes of a business network is what blockchain is, according to definition. Blocks in blockchain are linked to each other and consists of transactions [6]. As Blockchain is based on distributed systems the ability to scale horizontally is a distributed system's main benefit. Vertical scaling is a technique used in regular systems to enhance computation power by increasing the hardware resources. Building a distributed system makes it possible to use horizontal scaling, which involves adding another system or node when one system is experiencing too much traffic or load. Low latency and fault tolerance are made possible by this kind of scaling. Consensus is a concept used to address the problem of consistency and concurrency, which is a key flaw in such systems.

Data collection, transport, storing, and protection is a challenge that can be successfully solved by blockchain technology. Because of its intrinsic immutability and decentralisation, this technology has drawn increased interest in recent years. Blockchain technology is now being studied and used in a variety of industries, including finance, health, supply chain management, the Internet of Vehicles (IoV), industrial IoT, Digital media etc.

Decentralization, transparency, traceability, and tamper-resistance are the main benefits of blockchain technology. Blocks are verified, shared, synced, and produced among nodes in a distributed network using blockchain through a peer-to-peer, distributed, and decentralized consensus mechanism [4].

In this article, we attempt to introduce a novel consensus algorithm called a Lightweight consensus algorithm to overcome the limitations and also to increase the adaptability of blockchain technology based on the use. The proposed solution does not compromise on the security and verification of transactions. The algorithm is scalable for large systems.

II. Background knowledge

Satoshi Nakamoto invented blockchain technology in 2008 [1]. Without the involvement of a trusted centre, blockchain is a decentralised technology that enables us to guarantee the integrity of a distributed system that holds data on all system participants transactions. Transactions are specific actions. Information about transactions is grouped into blocks and chained together using hashing.

Everything in blockchain is transaction and to validate these transactions a distributed ledger is created among the group of unknown peers which is peer-to-peer network. Blockchain stands out with qualities of public accessibility (i.e. transparency) and immutability.

The consensus algorithm is a particular technique that is used to broadcast blocks to all nodes for validation [5]. It seeks to make it challenging for a possible attacker to compromise blockchains. The consensus algorithm refers to the process by which network nodes agree on the arrangement and make-up of the data that is stored concerning the transactions they carry out. A key component of blockchain technology is the consensus algorithm. It specifies the system's overall architecture as well as the order in which network nodes communicate.

The chosen consensus mechanism has a direct impact on how well blockchain technology is applied in a certain field. Therefore, selecting the best consensus algorithm and assessing its performance indicators are truly required while examining the application of blockchain technology.

In this section, we try to highlight the existing implementation of different consensus algorithms in blockchain for large scale networks:

The consensus algorithm employed by bitcoin is called as Proof of Work (PoW) [6] [7]. The fundamental idea behind it is to distribute incentives and the ability to add new blocks to the blockchain through competition among nodes for hashing power. The data from the previous block is used by the individual nodes to determine the precise solution to a mathematical problem. It takes skill to solve the math puzzle. The validators, also known as miners in POW, search for a nonce value (n), a unique variable used solely in PoW. Put the nonce in the candidate block's header for each possible nonce, then conduct hashing. This hash's output must be smaller than the target value that all system nodes have specified. The target is configured so that,a new block is created every 10 minutes. Miners develop pools when the PoW difficulty rises, where a group of miners collaborate on a single block and distribute the profits among themselves. Creating a mining pool may result in centralization since powerful mining pools with many participants may benefit.

. In PoW the miner which finds the nonce value first will add the next block in the blockchain and is rewarded. This process takes so much of energy consumption and resource utilization of other miners that simultaneously worked on the same block, therefore the malicious miner will not try for this process as it requires a large amount of computation.

Proof-of-stake (PoS) [7] requires participants to verify their ownership of the number of moneys instead than requiring nodes to solve a puzzle in an infinite area it is the mindset that those miners having more coins will not try to become malicious or attack the network. In PoS, the likelihood of selecting a miner is inversely correlated with the number of coins possessed. Stake can occasionally refer to other things. The core mechanism of PoS is the requirement that nodes who want to take part in the block building process should possess a particular amount of coins early on and they should also deposit a set quantity of their coins as stake. The stake serves as a promise that it will follow the protocol's regulations. Although PoS can solve the problems with energy and resource consumption, the phenomena of the affluent getting richer can show up throughout the PoS consensus process.

When the Practical Byzantine Fault Tolerance (PBFT) consensus method was first created, it was intended to be a system to protect a dispersed network's integrity [7]. This algorithm stipulates that every node must vote in order to add the subsequent block. Consensus must be reached by a two-thirds vote. Blockchain-based audit systems use the PBFT consensus method. The suggested system's security is impacted by the consensus algorithm chosen. Because of its high throughput and low latency, they use PBFT in their investigation. To maintain network integrity, the PBFT algorithm exchanges messages between nodes relatively frequently. PBFT has down point that it is suspectible to sybil attacks (one entities control many identities).and network Scaling Problem is also there.

III. Observation

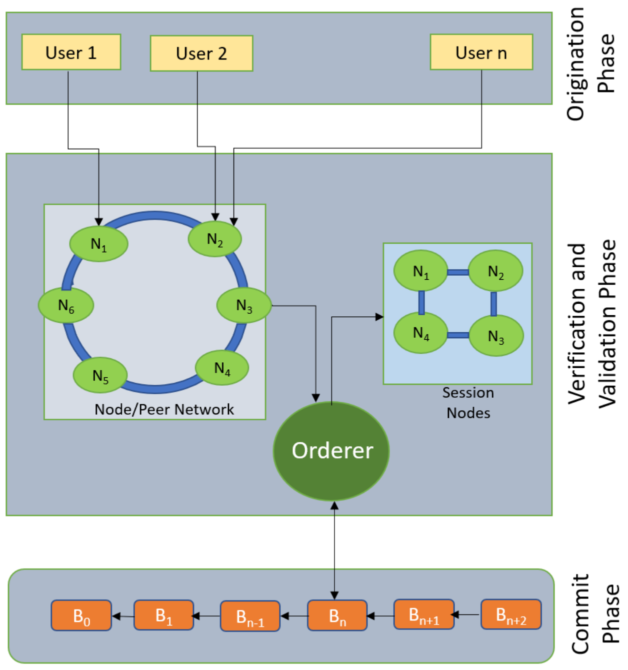
Be aware that the major drawback of proof-of-work is the processing power required to solve mathematical problems for blockchain transaction authentication and struggle with costly fees and sluggish transaction times. It took a lot of energy, wasn't eco-friendly, and frequently needed expensive equipment for mining and also observe that rich become more richer which has fast and efficient equipment.

The need that interested users acquire the native coin before becoming validators is the primary difficulty faced by all block chains that operate the proof-of-stake consensus protocol and also observe that rich become richer.

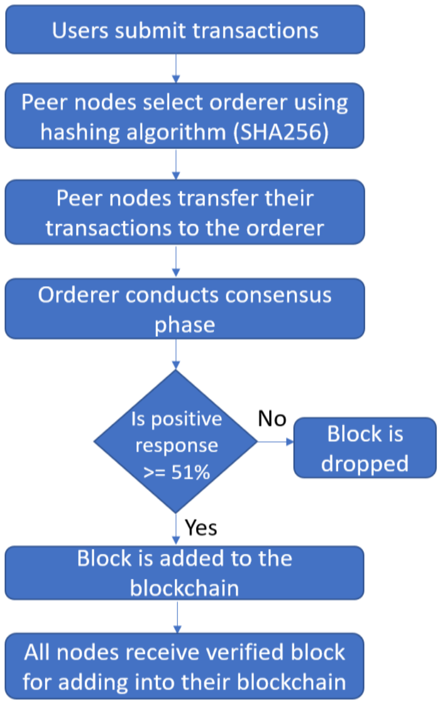
In private block chain security is more important rather than transaction fees and rewards [2]. As a component of a security lightweight consensus algorithm adds new feature on it by selecting a orderer through hashing and has a two way check protocol that validates a block through a group of nodes by assigning a random number to each one of them.

IV. Methodology

We propose a lightweight consensus algorithm that is less computationally expensive without compromising security aspect of blockchain. The entire framework consists of Peer nodes, Session nodes and orderer. Blockchain is acting as a key distribution center (KDC) for providing keys, signature to all the entities in the network. A Peer node is part of the blockchain network. They are specialized nodes that have enough resources to execute consensus algorithm and maintain the distributed ledger. A user is connected to a peer node, which processes the transactions submitted by that user. Peer nodes are shown in Fig. 1 as ($N_1, N_2, N_3, \ldots$). Orderer is one of the peer node [4] which is responsible for grouping all approved transactions into a newly generated block.



Framework and processes of the proposed system


Users submit transactions
Peer nodes select orderer using hashing algorithm (SHA256)
Peer nodes transfer their transactions to the orderer
Orderer conducts consensus phase
Is positive response >= 51%
No
Block is dropped
Yes
Block is added to the blockchain
All nodes receive verified block for adding into their blockchain

Flowchart

User: A user is a human who is utilizing the blockchain to store its data. Each user submits its data in the form of transactions to the peer node to add it to the blockchain.

A. Transaction Originating Phase

Each user is connected to a unique peer node N_i. A node can have multiple users connected to it but each user must be connected to only one peer node. A user submits a transaction to the peer node to add it to the blockchain.

B. Verification and Validation Phase

The correct validation of each user's signature using their public key is necessary for the verification of all incoming transactions. To establish a personal signature, a user's transaction and its private key are entered into the sign function. The sender then sends a transaction and an algorithm-generated version of his signature to the peer node. The verifier utilises the sender's public key, which is widely known, to confirm that this request originates from the intended sender. With the use of the public key and signature, the verifier can determine whether the request originates from the correct individual using a mathematical function. Because a signature is made up of 256 bits, it is impossible to forge one.

C. Orderer Selection Phase

Announce: Peer nodes that are interested in functioning as orderers generate a orderer interest transaction (t_0) that is structured as <timestamp, pk, sign>, where pk is the public key, sign is the signature corresponding to a public key and timestamp is the time at which the transaction was created. Transaction t_0 is broadcasted to the network and any t_0 that is generated after some ծ1 is discarded by the network.

Algorithm 1: Orderer selection

```
1: Interested peer nodes generate orderer transaction $t_0$.
2: Apply SHA256 on $t_0$.
3: Calculate KWV of the output hash value.
4: Broadcast $t_0$ and KWV to participating nodes.
5: Each participating node verifies the correctness of the
   KWV value of all the participating nodes.
6: if KWV of $j$ is maximum among participating nodes then
7:     $j$ is selected as orderer.
```

Calculate: Peer nodes verify public key and matches sign with pk. The potential validators then calculate Key weight Value (KWV) which is the sum of the numerical weight corresponding to each character of the hash function (SHA256) output applied on the t_0 transaction. The weight corresponding to each character in the hash function output is extracted from a key weight Value (KWV) dictionary. All participating nodes apply the same KWV dictionary to ensure that all have the same view of KWV. It is the responsibility of blockchain to populate KWV dictionary to all the peer nodes and main its consistency. The one with highest KWV value is selected as orderer to create the candidate block. At the end of each session, peer nodes need to use a new public key. This make sure that the process of selecting orderer remains randomized.

Agree: Each participating peer node agree on the selection of the orderer.

D. Consensus Phase

The orderer executes the consensus on a candidate block that includes a number of validated transactions that it accumulates over the course of a particular session. Peer nodes that need to report transactions to the orderer make up the session network and take part in consensus building. The range [1, |Ns|] is used to generate a distinct random number for each session node. The cardinality of the session nodes is represented by |Ns|. Each Tr'j and associated Rj are transmitted to a randomly chosen but distinct Nis for verification in order to ensure that Nis!= Njs. In other words, each node in Ns only receives one Tr'j, and it wasn't the one that forwarded the transaction to the orderer in the first place.

The candidate block and a random number selected from the aforementioned range are then transmitted to each session node. The consensus phase makes sure that either 51% (ceil([|Ns|/2]+1)) of the nodes answer positively to the verification request or that a timeout occurs, preventing block construction from continuing indefinitely. To make sure that the chosen node is responding, the verification response messages must include the random number. |Ns| * (1 + ((|Ns| 1)/2) calculates the total sum of all random numbers for this session.

The method totals the received random numbers as it processes the confirmed responses. The entire sum should equal the computed amount, and at least (ceil([|Ns| / 2] + 1) nodes correctly answered. When the algorithm returns true, the orderer only approves the new block and distributes it to all linked nodes in the network along with his or her signature. Each peer node authenticates the orderer's signature before adding the block to its ledger.

Algorithm 2: Consensus Phase Algorithm

```
Input: (Txn, P[])
Output: True or False
1. Struct Node contains
2.   int addr; byte txn, r;
3. end
4: set q ← 0; p ← 0; i ← count(P);
5: while $q < i$ do
6:   set $R^q \leftarrow$ Random_number($R_n$)
7:   Group all transactions $Txn^q \leftarrow P[q].addr$;
8:   set $P[q].r \leftarrow R^q$ ; $P[q].txn \leftarrow R^q$;
9:   $q{+}{+}$;
10: end
11: $Verify(Txn'^{L[q].addr}, R^q) \rightarrow N^q_s // (\exists N^q_s \in N_s) \wedge N^q_s \; != N^{L[q].addr}$
12: Initialize total $\leftarrow (1+((i-1)/2)) \times i$; $k \leftarrow 0$; $n \leftarrow (i/2)+1$
13: while ($k < n \wedge$ !session_timeout) do
11:   set $z \leftarrow 0$;
12:   Receive $R^q, N_s^{sign}, N_s^{addr} \leftarrow verify(R^q, N_s^{sign}, N_s^{addr})$
13:   for $z < q$ do
14:         if $(P[z].addr \equiv N_s^{addr}) \wedge (R^q \equiv P[z].r)$ then
15:         $sum \leftarrow sum + P[z].r$;
16:         $k{+}{+}$; break;
17:         end
18:   end
19: end
20: Set $sum \leftarrow sum + Faulty\_nodes(R^x) + Non\_responsive(R^y)$
21: if $(sum \equiv total) \wedge (k \geq n)$ then
22:   return true;
23: else
24:   return false;
25: end
```

Security analysis

1) Selection of orderer: The components of a transaction are timestamp, pk, and sign. Only the timestamp and pk may be changed by the adversary when using brute force. The transactions' generation must take place within a certain window of the block's generation window, which restricts the timestamp's possible range of values. The adversary experiences tremendous processing overhead while creating new keys and, consequently, new signs. Because of this, it is extremely difficult to guess who will place an order for a session.

2) Validation of Block (at Block Formation): All proposed transactions are aggregated and checked by session nodes to make sure no illicit trade is carried out (Ns). In this case, Ns varies based on the transaction for each block. In order for a rogue node to be included in Ns, a validated transaction must also be submitted to the orderer. The candidate block must, however, receive validation from at least 51% Ns, because it is not possible to predict which nodes will participate in any particular session.

3) Illegal Formation of Block by N_s Nodes: This is challenging since only Ns have access to the candidate block and random numbers. As a result, they are unaware of the transactions or participants in session formation. Even if this knowledge were to somehow come to light, session nodes (Nns) (Ns/2) + 1 would still need to hold. Here, Nns must make up 51% of Ns and must each provide a valid random number to validate the candidate block. This is countered by adding and cross-checking a random number that the orderer has given to the relevant nodes in Ns. Ns cannot thus create a block.

V. Expected results

For Computation time we have consider time taken for orderer selection and transactions validation by N_s during consensus formation.

$$T_b = T_o + \sum_{i=1}^{\left(\frac{|N_s|}{2}\right)+1} N_s^{i}(t)$$

where, T_o is the amount of time used to choose the orderer and $N_s^i(t)$ is the amount of time used by N_s to validate the consensus. In comparison to current state-of-the-art methods, the proposed technique will take less time to validate blocks because we only considered a subset of nodes (session nodes).

Theoretically, As we have consider only those nodes who have transactions sent to orderer to be session we can say that we have reduce the computation time. Also, since each session node receives transactions other than its own transactions the communication cost is also reduced. This is done without hampering the security aspect of the blockchain.

Hashing algorithm for orderer selection is the one we implemented on top of the base paper and it is not that much time-consuming operation. Hence, we present some statistics that are given in the base paper excluding orderer selection time. Here we are comparing consensus phase time with the existing Hyperledger Fabric consensus algorithm. When number of concurrent transactions are 100, Fabric requires ≈ 200ms while ours take ≈ 80ms. Similarly, for 300 transactions, Fabric time is ≈ 600ms while ours take 210ms which is 1/3 compared to the Hyperledger Fabric. As concurrent transactions increases, both Fabric and our algorithm times increases. However, ours algorithm computation time is significantly less than that of Hyperledger Fabric.

The number of transactions per block (varying from 5 to 250) and the accompanying number of consensus members are the next two variables we present transaction finalisation against (ranging from 2 to 50).This is import factor as number of session node participants directly affect system performance. Firstly, In case of 5 transactions with 2 session nodes per session, the time required is ≈ 4ms and ≈ 3.8ms, which is almost comparable. However, when the number of session nodes per session are 20 and transactions are 100, then validation time increases sharply (≈ 200ms) for Fabric, while our algorithm takes only ≈ 80ms. When session nodes are 50 for 250 trades, then Fabric takes 1.25sec while our algorithm takes 200ms. From this analysis, it is clear that our algorithm has superior performance in terms of time required to complete the transaction validation. The time needed would be much longer if 100% of the nodes participated in the consensus process. Because of this, the suggested approach works better as the number of transaction and participating nodes grows.

VI. future work

There can be few issues in current design, A timeout error happens if the orderer was unable to reach consensus in sufficient time for whatever reason (such as bandwidth, a cyberattack, a mistake, etc.). Every transaction is turned down and returned to the sources. Such an abnormality should be recognised by the application, which should thereafter resubmit the transactions without the user's knowledge. Also, memory requirement in blockchain can be improved using local and global blockchain strategies.

VII. Conclusion

Security and Scalability are major factor for the blockchain system. Blockchain's success is largely dependent on the creation of consensus among more than half of the peer nodes for each block. In a large-scale system, however, this results in a drop-in transaction rate as the amount of time needed for consensus development increases exponentially. We propose a lightweight algorithm that reduces communication and computation overhead without compromising verification/consensus process. Orderer selection provides a high level of security as it is very difficult to predict the output of SHA256 algorithm

References

[1] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” Decentralized Business Review, p. 21260, 2008

[2] D. Puthal, S. P. Mohanty, V. P. Yanambaka, and E. Kougianos, “Poah: A novel consensus algorithm for fast scalable private blockchain for large-scale iot frameworks,” arXiv preprint arXiv:2001.07297, 2020.

[3] B. Lucas and R. V. P ́aez, “Consensus algorithm for a private blockchain,”in 2019 IEEE 9th International Conference on Electronics Information and Emergency Communication (ICEIEC). IEEE, 2019, pp. 264–271

[4] S. Biswas, K. Sharif, F. Li, S. Maharjan, S. P. Mohanty, and Y. Wang, “Pobt: A lightweight consensus algorithm for scalable iot business blockchain,” IEEE Internet of Things Journal, vol. 7, no. 3, pp. 2343–2355, 2019.

[5] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, “An overview of blockchain technology: Architecture, consensus, and future trends,” in 2017 IEEE international congress on big data (BigData congress). Ieee, 2017, pp. 557–564.

[6] J.-T. Kim, J. Jin, and K. Kim, “A study on an energy-effective and secure consensus algorithm for private blockchain systems (pom: Proof of majority),” in 2018 International Conference on Information and Communication Technology Convergence (ICTC). IEEE, 2018, pp. 932–935.

[7] D. Mingxiao, M. Xiaofeng, Z. Zhe, W. Xiangwei, and C. Qijun, “A review on consensus algorithm of blockchain,” in 2017 IEEE international conference on systems, man, and cybernetics (SMC). IEEE, 2017, pp. 2567–2572.