Accelerating Blockchain Consensus: A Parallel Mining Approach for High-Throughput, Low-Latency Networks

Sohail Jabbar
College of Computer and Information Sciences,
Imam Mohammad Ibn Saud Islamic University (IMSIU),
Riyadh, 11432, Saudi Arabia

Abstract*—*Problem: **Blockchain consensus remains constrained by slow transaction verification, redundant mining effort, high communication overhead, and increased confirmation latency as the number of nodes grows.** ***Objective:*** **This study proposes NCABS, a controlled parallel mining consensus approach intended to improve blockchain throughput, latency, transaction commit rate, and resource utilization.** ***Methodology:*** **NCABS combines verifiable head-miner selection, transaction-validation delegation, parallel block generation, and a 75% confirmation threshold. The framework is evaluated against Practical Byzantine Fault Tolerance (PBFT) through ten repeated experiments over node densities from 40 to 140, using throughput, latency, communication load, transaction commit rate, scalability, and responsiveness as performance metrics.** ***Results:*** **The plotted results show that NCABS achieves higher throughput and transaction commit rates than PBFT, reduces aggregate communication messages by approximately 45%, and lowers latency at 40 nodes from about 310 ms for PBFT to about 195 ms for NCABS. In the transaction-commit-rate experiment, NCABS reaches approximately 3200 transactions per second, compared with about 2200 transactions per second for PBFT.** ***Conclusion:*** **The findings indicate that controlled parallel mining can reduce redundant computation and communication while improving blockchain responsiveness. The current evaluation is performance-oriented; adversarial testing and broader empirical comparison with additional consensus baselines are identified as future work.**

Keywords*—*Blockchain; consensus algorithm; parallel mining; PBFT; high transaction rate; low-latency

I. Introduction

Blockchain-based systems combine cryptography, public key infrastructure, and economic modeling to synchronize distributed databases through peer-to-peer networking and decentralized consensus. This technology has the potential to revolutionize various business areas and is expected to be rapidly adopted into mainstream use in a wide range of domains [1]. Blockchain technology has immense potential, but its market penetration is still relatively low. The ecosystem surrounding blockchain is not yet mature enough to support its widespread use, especially in supply chain applications. This is because most supply chains today involve many geographically dispersed and professionally diverse stakeholders, making them complex and difficult to manage. Therefore, extensive groundwork is required before blockchain can be adopted on a large scale. A recent study by Jabbar *et al.* [2] provides a detailed analysis of blockchain-enabled supply chains, discussing the challenges and future directions. The most significant challenge the blockchain ecosystem faces is scalability, which affects transaction processing speed and efficiency. Compared to traditional databases, blockchain is slower in retrieving and committing records, and it requires more computing resources. The scalability of these resources is a major concern as it affects the system's ability to respond and function effectively as the size of input increases to meet user demand [3]. Scalability is crucial in three key aspects of a supply chain system: 1) Throughput, which refers to the efficiency of producing results; 2) Storage, which involves maintaining transaction records; and 3) Network communication, which pertains to the number of messages exchanged among nodes to finalize a transaction. This study focuses on throughput and network communication scalability.

Considering the founding cryptocurrency, Bitcoin can handle $4--5$ transactions per second. As the average transaction size is 380.04 bytes and the block size is $1,048,576$ bytes, the number of transactions per block is $2,759.12$. The current block generation time is 600 seconds (10 minutes). Conversely, traditional centralized payment systems, e.g., VISA, MasterCard, and PayPal, offer significantly higher transaction rates per second [4]. VISA processes over a million daily transactions, i.e., around 1800 per second. From a storage perspective, the size of the Bitcoin Blockchain has grown steadily since its creation. It reached approximately 242.39 gigabytes in size as in Q3, 2021, while it was only $53,647$ MB in Q4, 2015. In Ethereum, transactions increased from just USD $ 3,000 in Oct 2015 to over one million in Jan 2023. Improving network communication is crucial for reducing energy consumption and propagation delay. In Blockchain, each node acts as a relay, broadcasting all transactions at least twice. When a transaction is generated, it is first broadcast to all nodes. Once a block containing the transaction is mined, it is rebroadcast to all nodes. Unfortunately, this process leads to block propagation delay [5]. Given this process, increasing the data transmission rate to handle a large volume of transactions is not feasible, as it requires more network bandwidth. Hence, it is essential to develop more effective data transmission mechanisms. Additionally, the number of transactions per block, N_{tx}, is determined by the maximum block size, B_{size}, and the average transaction size, T_{size}, as shown in Eq. (1):

$$N_{tx} = \left\lfloor \frac{B_{size}}{T_{size}} \right\rfloor \tag{1}$$

The throughput, θ, is the rate at which transactions are processed, which is defined by the number of transactions per block and the block generation time, T_{block}, as described in Eq. (2):

$$\Theta = \frac{N_{tx}}{T_{block}} = \frac{\left\lfloor \frac{B_{size}}{T_{size}} \right\rfloor}{T_{block}} \tag{2}$$

The delay in block propagation, Δt_{prop}, is a function of the network topology and the number of nodes N, as shown in Eq. (3):

$$\Delta t_{prop} = f(N, networktopology) \tag{3}$$

The total transaction confirmation time, $T_{confirmation}$, is the sum of the transaction propagation delay, the mining time, and the block propagation delay, as defined in Eq. (4):

$$T_{confirmation} = \Delta t_{txProp} + T_{mining} + \Delta t_{blockprop} \tag{4}$$

In a parallel mining system like NCABS, the total consensus time, T_{NCABS}, is the minimum time among all parallel tasks, as shown in Eq. (5):

$$T_{NCABS} = \min(T_1, T_2, \ldots, T_k) \tag{5}$$

The communication complexity of the PBFT algorithm, C_{PBFT} is of the order of the square of the number of nodes, as shown in Eq. (6):

$$C_{PBFT} = O(N^2) \tag{6}$$

The contributions of this research study in the blockchain scalability domain are enumerated as follows. The first two are considered major contributions, while the next two are of second and third-level need and importance.

- A new consensus mechanism, NCABS, has been proposed to speed up consensus in blockchain. This mechanism uses parallel mining instead of solo or repetitive mining, making it a strong solution for achieving the target. To help readers better understand this, a graphical presentation of the various steps involved in consensus-based scalability solutions for blockchain has been included.
- A detailed experimental comparison of PBFT and NCABS is presented, focusing on throughput, network latency, scalability, and responsiveness, with results depicted.
- A comparative analysis of PBFT variants, including Multi-layer PBFT, NBFT, CDBFT, and RBFT, is presented in a summarized tabular format and a discussion.
- A multi-layer categorization of available scalability solutions in the literature is provided, along with their extended versions.

II. Problem Formulation

This study considers a blockchain network consisting of N nodes, where each node participates in transaction validation and block generation. The key challenge is to minimize:

- Transaction confirmation time ($T_{\text{confirmation}}$)
- Resource utilization (CPU, energy)
- Network communication overhead

Subject to the following constraints:

- Maintaining consensus correctness
- Avoiding forks
- Ensuring fairness among miners

Formally, the objective function is defined as Eq. (7):

$$\begin{aligned} \min \quad & T_{\text{confirmation}} + \lambda_1 \cdot \text{ResourceUsage} \\ & + \lambda_2 \cdot \text{CommunicationCost} \end{aligned} \tag{7}$$

Such that [see Eq. (8) and Eq. (9)]:

$$\text{Consensus Validity} \geq 75\% \tag{8}$$

$$\text{Fork Probability} \rightarrow \min \tag{9}$$

The formulation reflects three observations that guide the design of NCABS. First, confirmation delay increases when many miners repeat the same transaction-verification and nonce-generation work. Second, PBFT-style consensus becomes communication intensive as the network size grows because message exchange increases with node density. Third, a controlled validator and head-miner selection process is needed so that performance gains do not come at the expense of consensus correctness or fork resistance.

III. Literature Survey

Tian Junfeng *et al.* [6] proposes a parallel mining-based enhancement to blockchain consensus that improves throughput and reduces latency by enabling concurrent mining operations. It introduces coordination mechanisms to maintain consistency and security while minimizing forks. Experimental results demonstrate significant performance gains, making the approach suitable for scalable, high-performance blockchain applications.

Yin Lingyuanan *et al.* [7] proposes a parallelism-enabled blockchain consensus framework to overcome the limitations of traditional PoW-based systems. By allowing concurrent mining and validation processes, the approach improves throughput and reduces latency, while maintaining consistency and security through coordination mechanisms. Experimental results confirm its effectiveness for scalable and real-time blockchain applications. The work by Lulu Ke *et al.* [8] proposes a blockchain-based framework for vaccine supply management that ensures verifiability, traceability, and data integrity. By eliminating reliance on centralized authorities and introducing an improved digital signature scheme, the

approach enhances both security and efficiency. Experimental results demonstrate reduced computational cost and improved trust, making it suitable for real-world healthcare supply chains.

Soohyeong Kim *et al.* [9] classified the scalability methods as on-chain, off-chain, side-chain, child-chain, and inter-chain solutions. Junfeng Xie *et al.* [10] categorized scalability solutions into four groups: the number of transactions, block interval time, data storage, and data transmission. This study classifies blockchain scalability solutions into three categories: on-chain, off-chain, and consensus mechanism-based. The following paragraphs briefly review each category, and Fig. 1 provides a visual representation of the classification. Although several solutions exist in each category, only representative methods are discussed due to space constraints.

The statistical gap motivating scalability research is substantial. Bitcoin typically processes about 4–5 transactions per second, while Ethereum is commonly reported at around 15 transactions per second and permissioned systems can achieve higher rates under controlled membership. In contrast, conventional payment systems such as VISA are reported to process around 1800 transactions per second in the cited literature [4]. This gap is intensified by communication complexity: PBFT-based designs require all-to-all message exchange with $O(N^2)$ communication cost [11], whereas scalable blockchain deployments require transaction throughput and confirmation latency to remain stable as the number of nodes increases.

A. On-Chain Solutions

Require a structural or fundamental change to the blockchain, and thus a modification to the protocol's underlying rules. This is technically known as a *hard fork* or *controversial hard fork* when there is a split in the community, as groups form who accept or reject the proposed update. Some popular examples of new currencies addressing the scalability issue that have resulted from a hard fork are as follows: *Bitcoin Cash* - Increases the block limit $(1MB - 8MB)$, *Litecoin* - Reduces block generation time $(10min - 2.5min)$, *DASH* - Higher transaction speed. For comparison, in the case of Ethereum, the maximum capacity is 15 transactions per second, and permissioned Blockchains typically offer much higher transaction throughput than an open and permissionless network. Changes to the Bitcoin Blockchain's proof-of-work parameters could be an option to increase scalability; for example, decreasing the block generation time or increasing the block size [12]. Still, these are unlikely to be feasible solutions, since they cause effects such as orphan blocks, forks, chain reorganization, slower network propagation, and, in extreme cases, security vulnerabilities such as selfish mining and double-spend attacks. Another method in this category to improve Blockchain scalability is *sharding*. This is a way to divide blockchain data across nodes, enabling transactions to be processed in parallel. Instead of every node processing every transaction and storing all the data, nodes are grouped, with each group handling and processing a portion of the data. This helps scale the network by enabling high concurrency. *Segwit* (Segregated Witness) is an upgrade to the data structure (transaction format) of the Bitcoin protocol. The transaction is segregated into original data and signature data (witness). The witness data is separated from the Merkle tree record [13] and appended as a separate structure at the end. This upgrade is part of the Litecoin protocol.

B. Off-Chain Solutions

It uses secondary protocols built on top of the main blockchain and are, therefore, referred to as a *second-layer* scalability solutions. In this approach, transactions are offloaded from the main blockchain and performed privately between the interacting partners. It confers the advantages of reduced congestion on the MainNet, increased throughput, reduced transaction fees, and space savings. Many off-chain solutions for various platforms are available in the literature, e.g., [14], [15]. *Lightning Network* [14] covers small and everyday transactions that do not need to be stored on the Bitcoin MainNet. Off-chain approaches to scalability issues in Ethereum include the Raiden network and Plasma cash [16], [17]. These off-chain mechanisms reduce the number of transactions of the entire network processes, though they still require at least two transactions to initialize and terminate the off-chain channels.

Optimizing the working of the ***Consensus algorithm*** can also aid in addressing scalability [15] issues. The **Proof of Work** (PoW) was first introduced with Bitcoin. It engages all miners simultaneously in a competition to create a block, and only the first to produce a valid block is accepted and rewarded. Thus, the energy and computational resources of the other nodes are wasted [15]. In contrast to PoW, **Proof of Stake** (PoS) chooses a validator among all the participating nodes using a stake-based selection algorithm. An exclusive right to create a block is assigned to this selected node, while the remaining nodes do not need to waste energy or computational resources. In contrast to PoS, **Delegated Proof of Stake** (DPoS) operates democratically, with delegates elected by token-holding nodes. EOS.IO is a decentralized Operating System that uses DPoS as its consensus algorithm, focused on scalability and capable of processing millions of transactions per second. *Steemit* featuring STEEM, SteemPower, and SteemDollar as native cryptocurrencies is a Blockchain-based social media platform. It uses DPoS and Graphene to process millions of transactions per second efficiently. In the domain of financial transactions, *Bitshare* is a decentralized platform designed to enhance transaction processing speed. Another DPoS-based solution to the scalability issue in blockchain at the second layer is *Lisk*.

It is a modular blockchain platform with smart contract functionality, which offers a partial solution to the blockchain scalability problem through the use of sidechains. Until now, all the extensions mentioned here have not reached maturity and are still under development. The **Byzantine Fault Tolerance** (BFT) consensus algorithm gives a solution to a classic problem in distributed computing, the *Byzantine General Problem*. BFT takes different forms: practical BFT, Federated Byzantine Agreement, Delegated BFT, and others. Practical BFT (PBFT) can scale to large computational loads with only a minimal increase in latency [11]. Among many other projects that use the BFT are LibraBFT, Hyperledger Fabric, and Zilliqa. The *Stellar* Consensus Protocol (SCP) is an improved implementation of the Federated Byzantine Agreement. It requires multiple rounds of voting to reach

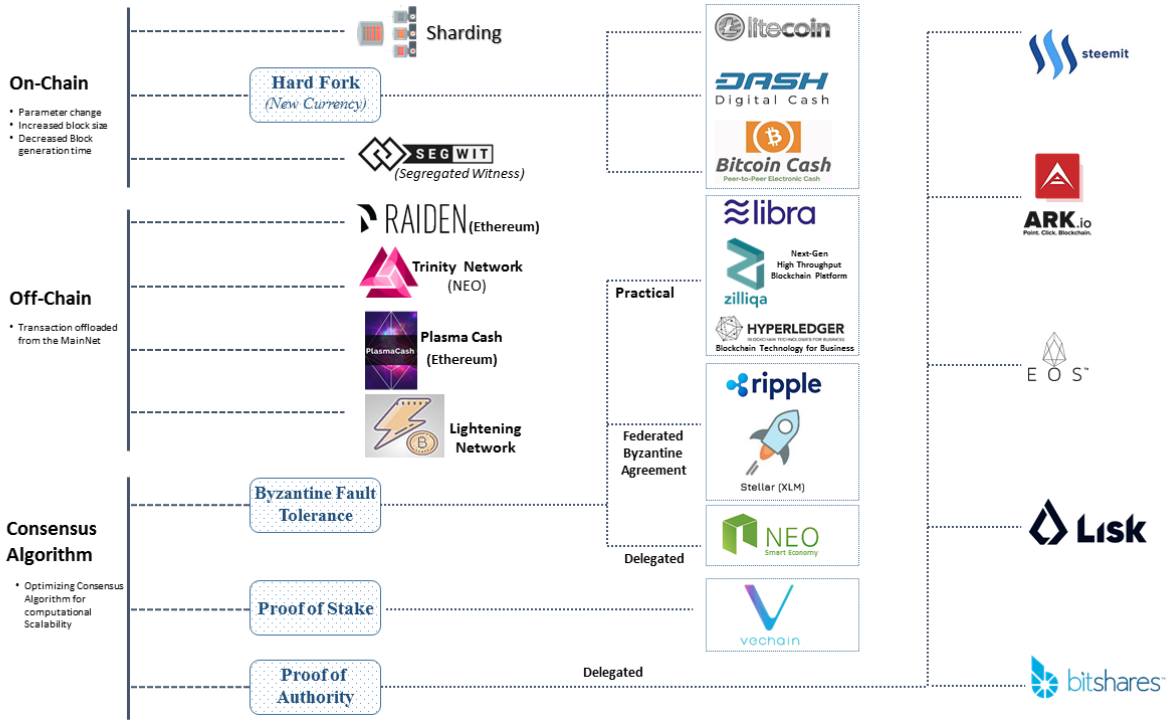

Fig. 1. Categorization of scalability solutions for the blockchain.

consensus, which generates significant network traffic. However, the messages are lightweight and within the capabilities of the currently available Internet. A further extension to SCP is the *Pi* Consensus algorithm, specifically designed for low-energy/computationally power devices such as home PCs and mobile phones. The *Bitcoin-NG* consensus algorithm [18] reduces latency and increases throughput by using leader election to append microblocks.

Some other extensions of PBFT that work for the scalability of the Blockchain network are Kauri [19], Hotstuff [20], and DBFT [21]. A comprehensive performance modeling and analysis of Hotstuff for Blockchain consensus is given in [22]. A detailed article on the challenges of PBFT-inspired consensus for blockchain and enhancements over neo dBFT is available at [23]. This proposed scalability solution, NCABS, is categorized as optimizing the Consensus algorithm to improve the scalability of the Blockchain network. Existing methods either improve throughput at the cost of communication overhead or reduce energy consumption while sacrificing decentralization. NCABS addresses both issues, simultaneously. Table I presents the comparison of blockchain consensus methods.

TABLE I. COMPARISON OF BLOCKCHAIN CONSENSUS METHODS

Method	Parallelization	Limitation	Gap
PBFT	No	$O(N^2)$ communication	Not scalable
PoW	No	Energy waste	Inefficient
Parallel Mining (existing)	Yes	Fork risk	No control
NCABS (Proposed)	Yes	—	Controlled & efficient

IV. PRELIMINARIES

The following is a brief description of key terminology and concepts to better understand the proposed solution and the rest of the study.

A. Cryptocurrency Addressing

An alphanumeric identifier of 26-35 characters representing different cryptocurrency presentation formats, beginning with the numbers 1, 3, or bc1, is shown in Fig. 2. The figure provides a self-explanatory depiction of the formats, characteristics, validation, and generation of cryptocurrencies.

The public key (K_pub) is first hashed using SHA-256 and then RIPEMD-160 to produce a public key hash $(Hpub)$.

$$H_{pub} = \text{RIPEMD160}(\text{SHA256}(K_{pub})) \tag{10}$$

The resulting public key hash is shown in Eq. (10). Next, a version byte (V) is prepended to the hash to create the versioned payload (P_V):

$$P_V = V \mid H_{pub} \tag{11}$$

Eq. (11) illustrates the concatenation of the version byte with the hash. A 4-byte checksum (C) is generated by performing a double-SHA-256 hash on the payload and extracting the first four bytes.

$$C = \text{Substr}_{0,4}(\text{SHA256}(\text{SHA256}(P_V))) \tag{12}$$

The checksum calculation is formally defined in Eq. (12). The checksum is appended to the versioned payload to form the complete binary address (A_bin).

$$A_{bin} = P_V \mid C \tag{13}$$

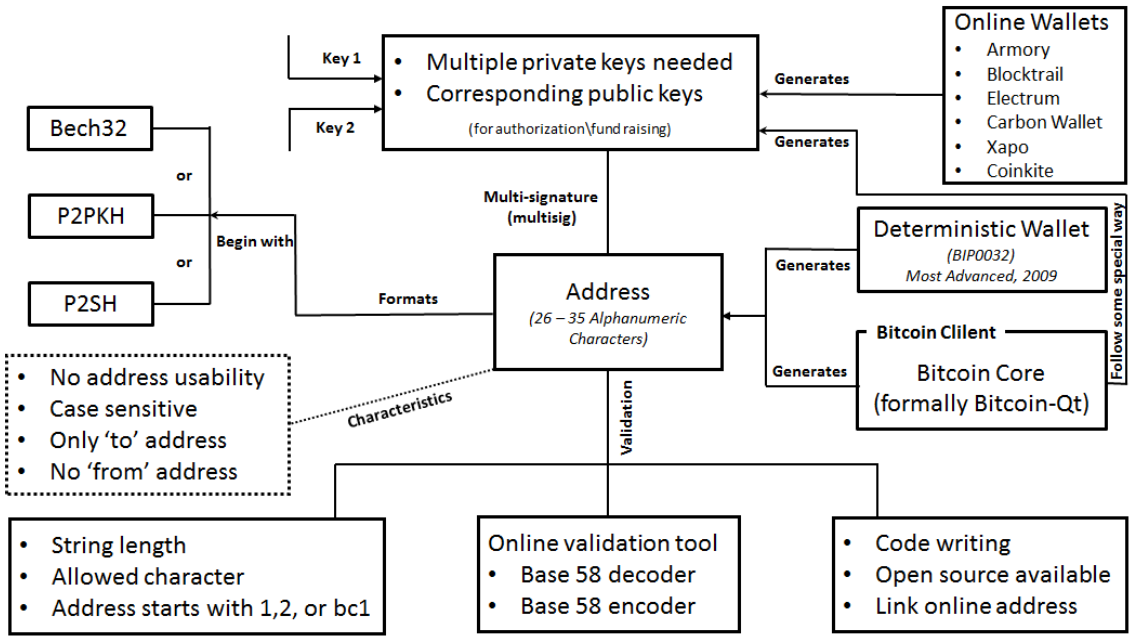

Fig. 2. Cryptocurrency address format, characteristics, generation, and validity.

The final binary address is constructed by appending the checksum, as shown in Eq. (13).

Finally, the binary address is encoded using the Base58 alphabet (A_58) to produce the human-readable address (A_addr).

$$A_{addr} = \text{Encode}\mathcal{A}58(A_{bin}) \tag{14}$$

The human-readable address is generated through Base58 encoding as per Eq. (14).

B. Genesis Block

The Genesis Block is the first block in a blockchain. In the case of the Bitcoin Blockchain, it was initially numbered '1', but modern versions start with '0'. This block is hard-coded and serves as a trusted starting point for the participants. It is an unspendable subsidy that does not refer to a previous block. The genesis block is indexed as 0 in modern implementations.

C. Unspent Transaction Output: UTXO

UTXO is an indivisible chunk that currently represents the "Right to Spend" of someone at a given moment and is recorded on the blockchain. Every received currency chunk is recorded as a UTXO in the blockchain and is scattered throughout the blockchain, even from a particular owner. The Wallet scans the blockchain and calculates the user's balance. The UTXO data on the blockchain is stored in a database table in local memory or in a permanent storage called the UTXO set or UTXO pool.

D. Transaction Format

The UTXO consumed by a transaction (i.e., payment or a transaction fee) is termed transaction input, while transaction output is the UTXO created by a transaction (i.e., change of "Right to Spend" authority or small money returned as change by the spending money with a large valued UTXO).

E. Transaction Fee

A transaction fee is a charge levied on users for each cryptocurrency transaction. This fee is collected to process the transaction on the network. The transaction fee is calculated on the basis of the transaction size in kilobytes. Some wallets automatically calculate and include the transaction fee. Therefore, the user needs to choose a wallet that suits their needs. Additionally, it is crucial to note that the wallet must source a set of UTXOs that add up to sufficient currency (the transaction amount plus the transaction fee). Manual accounting for the transaction fee is recommended when wallet-level fee estimation is unavailable.

F. Merkle Tree

Merkle Root comes up with a Merkle tree when a repeated encryption process forms a tree structure and ends up at a single hashed value. In Ethereum, the state's information is stored in the *Merkle Particia*. It is for preventing the Block from any counterfeit. The *Merkle Path* in the Merkle Tree helps a node find a particular transaction quickly and easily. The conditional state update of the blockchain is represented in Eq. (15):

$$\mathcal{C}_{t+1} = \begin{cases} \text{Append}(\mathcal{C}_t, B_{\text{new}}), & \text{if } \left(\sum_{m_j \in M} V(B_{\text{new}}, m_j) \right) \geq \frac{3}{4}|M|, \\ \mathcal{C}_t, & \text{otherwise.} \end{cases} \tag{15}$$

V. Proposed Solution

In the proposed idea, only some miners participate in the nonce generation and hence do not pick up the transaction from the unconfirmed pool. Each miner generates a random number, and their timer is set to it. The one with the lowest random number value can pick up the transaction. To reduce miners' computational load and save verification time, the relevant data is communicated to the specific node. The selection of this particular node is on the part of the selected miner. The selected miner generates a random number from the range of the total number of nodes and sends the required data to that selected node. The selected node verifies the shared transaction and performs the necessary hashing, culminating in the generation of the Merkle tree that ends at the Merkle root. This selected node shares the data with the head miner. The head miner solves the puzzle for generating the nonce value (explained in the subsection Nonce Generation).

This contrasts with the traditional nonce-value generation competition among all miners, which consumes and wastes computational power and delays block generation. Next to it is the new block generation by the head miner. This newly generated block is appended to the existing blockchain and shared among miners for validation. On having validity confirmation from $\frac{3}{4}th$ of the total miners, this blockchain is replicated to all the miners. A transaction confirmation message is sent to the transaction's direct stakeholders. The detailed workings of the proposed scheme, along with the architectural and communication models, are presented in subsequent subsections. Table II gives some commonly used terms and their descriptions. A schematic diagram of the consensus algorithm-based scalability solution for blockchain is given in Fig. 3. Also, a detailed presentation of the Merkle Tree-Generation algorithm, which is self-descriptive, is given in Algorithm 1. The components of Fig. 3 are presented separately in Fig. 4 and Fig. 5, and their detailed workings are well explained in the proposed solutions. The consensus condition for a new block to be replicated is based on a threshold of valid confirmations, as shown in Eq. (16):

$$N_{confirm} \geq \frac{3}{4}N \tag{16}$$

where, $N_confirm$ is the number of miners confirming validity, and N is the total number of miners. The selection of a miner for nonce generation is based on the lowest random number, as represented in Eq. (17):

$$S = \{i \in M \mid R_i = \min_{j \in M}(R_j)\} \tag{17}$$

where, S is the set of selected miners, M is the set of all miners, and R_i is the random number generated by miner i. To prevent a miner from choosing or modifying R_i after observing competitors' values, the election can use a lightweight commit-and-reveal process. Each eligible miner first broadcasts a commitment $c_i = H(u_i \| pk_i \| H_{prev} \| e)$, where u_i is a locally generated cryptographic random seed, pk_i is the miner identity, H_{prev} is the previous block hash, and e is the current election epoch. After the commitment window closes, miners reveal u_i, and the score is recomputed as $R_i = H(u_i \| pk_i \| H_{prev} \| e)$. Missing or inconsistent reveals are discarded and may trigger slashing. The total consensus time for the proposed system, T_{NCABS}, is a function of the parallel mining time for the head miner, which is a significant improvement over traditional mining methods, as described in Eq. (18):

$$T_{NCABS} < T_{solo} \tag{18}$$

where, T_solo is the time for a traditional solo mining process.

TABLE II. Some Frequently Used Terminologies and their Description.

Term	Description
$V_{(Block)}$	Block validators/miners
$V_{(Tx)}$	Transaction Validators
$Act_{(nodes)}$	Actors
$Power_{(Comp)}$	Computational power
$Wealth_{(Tkn)}$	Token wealth
$Pool_{(U-Tx)}$	Unconfirmed Transactions Pool
$Pool_{(C-Tx)}$	Confirmed Transactions Pool
$Gen_{(Block)}$	Block Generator

A. Architectural Model

The stakeholder nodes are grouped into three major categories namely; Block Validators $[V_{(Block)}]$, Transaction Validators $[V_{(Tx)}]$, and Actors $[Act_{(nodes)}]$. Computational power $[Power_{(Comp)}]$ and Token wealth $[Wealth_{(Tkn)}]$ are two parameters to set the threshold for categorizing the applicant nodes into Block validators and Transaction validators. Actors are transaction nodes that have installed the Blockchain client and wallet applications to interact with the Blockchain network and perform transactions (sending and receiving currency). Transaction validators are nodes with applicants that meet the lower thresholds for $Power_{(Comp)}$ and $Wealth_{(Tkn)}$. The applicant nodes that meet the upper limit criteria for the threshold values, i.e., $Power_{(Comp)}$ and $Wealth_{(Tkn)}$, are categorized as Block validators. These are also the full nodes. Since their work is heavier and more resource-intensive than that of the transaction nodes (Actors), e.g., due to nonce generation. First, the Transaction is entered into the system and placed in the Unconfirmed Transaction Pool $[Pool_{(U-Tx)}]$. Later, once it reaches back from the Transaction Validators $[V_{(Tx)}]$, it is placed by the miner in its own confirmed transactions pool $[Pool_{(C-Tx)}]$ that is explained in Section V-C1a and

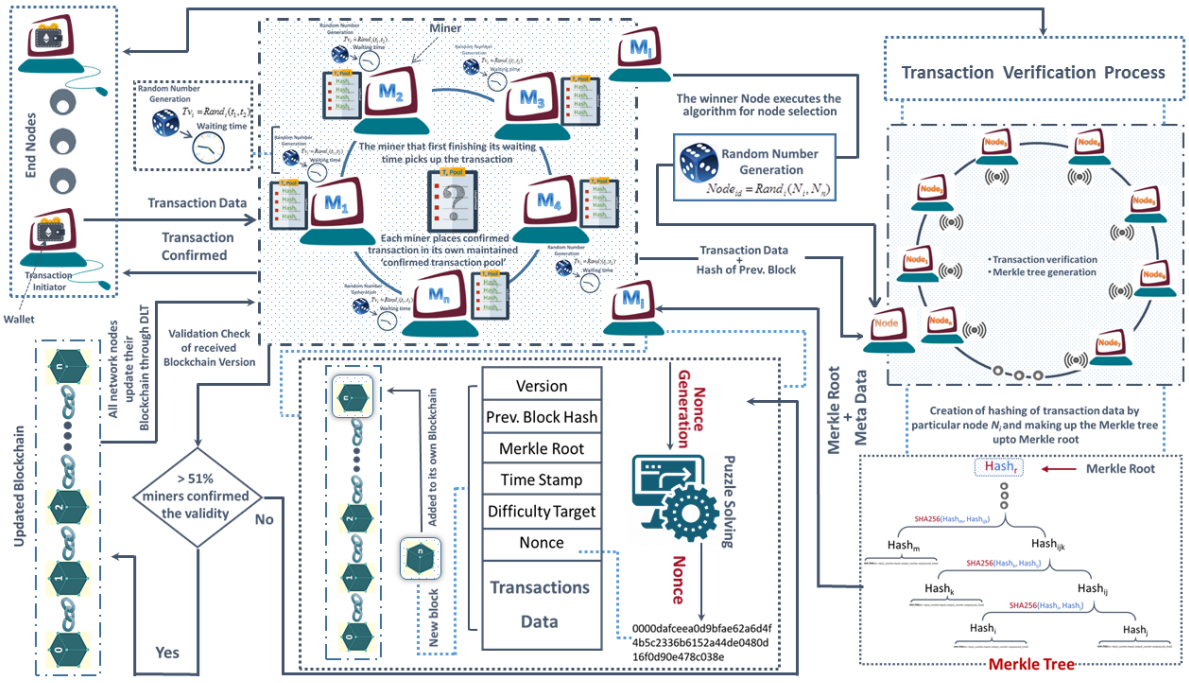

Fig. 3. Consensus algorithm-based scalability solution for blockchain.

are graphically represented in Fig. 4. The description of the terminologies used in this study is tabulated in Table II.

B. Communication Model

Moreover, the higher the number of incoming transactions, the lower the risk of such inconveniences caused by outdated node status. As with each node's response to the transaction verification data, the updated list of alive nodes is shared. So there is very little chance of incorrect information. Every node that sends the transaction verification data back to the head miner will attach the list of live nodes. This list of received messages is then communicated to co-miners. The node collects this information while getting its transactions verified.

C. Detailed Working of the Proposed Solution

The proposed solution operates through a structured consensus algorithm where validation precedes consensus to ensure the running blockchain is the final, agreed-upon version. The process begins when a transaction is placed into an unconfirmed transaction pool, $Pool_(U-Tx)$. A block generator, $Gen_(Block)$, is then selected via a protocol in which all miners generate a random number and set a corresponding timer; the miner with the shortest timer value wins the right to pick up the transaction bunch. This winning miner, or head miner, then delegates the computationally intensive verification task by randomly selecting a dedicated Transaction Validator node, $V_(Tx)$. This selected $V_(Tx)$ node verifies the transaction's legitimacy, syntax, and data structure, and performs the necessary hashing to generate a Merkle tree and its corresponding Merkle root. The validator sends this data back to the head miner, who then solves the puzzle to generate a nonce, creates the new block, and appends it to its local blockchain. This newly updated blockchain is shared with other miners for validation. Upon receiving confirmation from at least 3/4ths of the total miners, it is replicated across the entire network, and a confirmation message is sent to the transaction's stakeholders.

1) Consensus algorithm: The validation precedes the consensus. The consensus protocol ensures that the currently running Blockchain is the final, true version on which at least the system's key stakeholders have agreed. In the proposed consensus algorithm, the Blockchain is replicated to all the miners, once having the validity confirmation from $\frac{3}{4}th$ of total miners. This $\frac{3}{4}th$ ratio is chosen following the renowned previous practices in the literature. Section V-C1c explains it in detail. Before it, nonce generation is the core activity miners perform to make the process fairer, as described in Section V-C1b. After the Block generation and the replication of the Blockchain throughout the network through DLT. The transaction is confirmed and committed. This process is presented in Section V-C1d.

a) Transaction pool: The transaction pushed from the Wallet into the blockchain system is placed in the transaction pool. In Bitcoin, it is named the Mempool. The selected miner picks up the transaction(s) out of the available transactions in the transaction pool [Unconfirmed Transactions Pool $(Pool_{(U-Tx)})$]. Once the transactions are returned from the $V_{(T_x)}$ after confirmation, each miner places them in their in-house maintained pool, the Confirmed Transactions Pool $[Pool_{(C-Tx)}]$. There are some issues of delaying in picking up the transaction from $Pool_{(U-Tx)}$ due to low transaction fees. One solution to it is increasing the transaction fee to push it faster to get it done. SegWit-enabled wallets like Ledger Nano S, Exodus, Segwitaddress, Trezor, or Electrum are worth considering to get faster confirmation from the transaction

Algorithm 1 Merkle Tree Generation

Input: A set of transactions in a block, $T = \{tx_1, tx_2, \ldots, tx_n\}$.
Output: The Merkle Root, M_R.
Definitions:
    Let $H(\cdot)$ be a cryptographic hash function (e.g., SHA-256).
    Let $||$ denote the byte-wise concatenation operator.
    Let L_j be the list of hash nodes at level j of the tree.
function COMPUTEHASH($data$)
    return $H(data)$
end function
function BUILDMERKLETREE(T) ▷ Generate leaf nodes (Level 0)
    Initialize $L_0 \leftarrow \emptyset$
    for $i = 1$ to n **do**
        $l_i \leftarrow$ ComputeHash(tx_i)
        Append l_i to L_0
    end for
    $j \leftarrow 0$ ▷ Iteratively build the tree until one root remains
    while $|L_j| > 1$ **do**
        Initialize $L_{j+1} \leftarrow \emptyset$
        Let $N_j = |L_j|$ ▷ Iterate over pairs of nodes in the current level
        for $k = 1$ to $\lceil N_j/2 \rceil$ **do**
            Let the left child hash be $h_{left} \leftarrow L_j[2k-1]$.
            Let the right child hash be $h_{right} \leftarrow \begin{cases} L_j[2k] & \text{if } 2k \leq N_j \\ L_j[2k-1] & \text{if } 2k > N_j \end{cases}$ (Handles odd number of nodes) ▷ Compute the parent hash by hashing the concatenated children's hashes
            The parent hash h_{parent} is computed as:

$$h_{parent} = H(h_{left}||h_{right})$$

            Append h_{parent} to L_{j+1}
        end for
        $j \leftarrow j + 1$
    end while ▷ The final list contains the single Merkle Root
    $M_R \leftarrow L_j[0]$
    return M_R
end function

pool. In the experimental setup, submitted transactions are processed according to their priority, and no external acceleration application is used to push selected transactions ahead of the normal queue. A concise depiction of transaction pools is shown in Fig. 4.

b) Nonce generation: A nonce is a number that results from repeatedly hashing the input until it meets a certain difficulty threshold. In a Block, *nonce* generation is a major task performed by miners and takes most of their time. The process of generating nonces is called mining, and miners mainly do it. It is also called *puzzle solving* because it is a completely random, trial-and-error process that continues until the eligible signature is reached and the required difficulty level is met. It could take many forms, ranging from spaces to question marks to numbers, periods, capital and lowercase

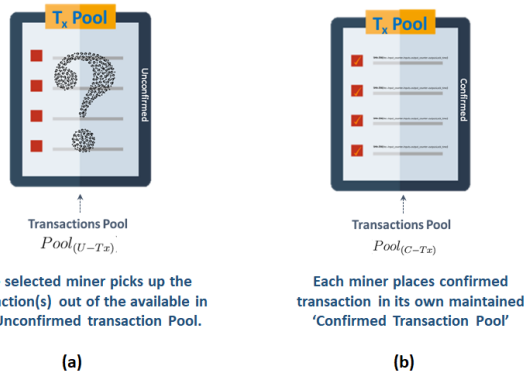

Fig. 4. (a) Unconfirmed transaction pool, (b) Confirmed transaction pool

letters, and digits. Once *nonce* is generated, it is disseminated among other miners along with its key for verification. This also indicates to other miners that they should pause and stop proceeding to actual block generation. Since each block generation process requires the *previous block hash* as a prerequisite, if all nodes start generating blocks, it can lead to confusion about the *previous block hash*. This sharing of the newly generated *nonce* also confirms the miner's message and prevents it from being misleading. This successful node completes the block generation process by setting its *previous block hash* parameter to the latest record in the Blockchain. The locally upgraded Blockchain at the particular miner is sent to the other miners for validation. This received message is a *Go Ahead* call for two actions at the message recipient *i.e.*, Miners. 1) Validation check of the received Block version and respond to the outcome. 2) Proceed to the block generation process.

c) Block generation: The Block Generator $[Gen_{(Block)}]$ is selected by following a special protocol. All the miners generate a random number and set their timers accordingly. The miner with the least timer value takes the right to pick the transaction bunch from the $Pool_{(U-Tx)}$. From a practical perspective, the miner completes its waiting time, which is equal to the randomly generated value, wins the competition, and takes the turn to start the Block generation process, first by picking the transaction(s). A message disseminated to all other miners from this winning miner stops them from waiting further. The message sender miner, *i.e.* the previously winning miner, does not take part in the next competition and will wait for a message from the winner node of the next turn to take part therein. The message also contains its random number value. All the recipient miners wait for this time, then generate a random value as previously presented and discussed on completion of this phase of transaction bunch picking. The transaction verification process is initiated by the $V_{(T_x)}$ node. Once all the required data for a Block is collected and produced, the selected $Gen_{(Block)}$ node proceeds to its generation. Here, another important point needs attention, *i.e.* a serious clash surely will be raised on getting the previous *Block Hash* and including it in the required data for generating the *nonce* and hence the Block generation. Suppose two miners proceed with Block generation using the same *previous hash value*. In

that case, it will disrupt the Blockchain, leading to multiple forks that would be impossible to resolve later. Hence, it is critically important to focus on it and to resolve it.

The fairness of this selection depends on making the random values unpredictable before the election and verifiable after the election. NCABS therefore assumes that random numbers are generated using a cryptographically secure pseudo-random number generator and are bound to the previous block hash and the current epoch through the commit-and-reveal process described above. This binding prevents a miner from reusing a favorable value across epochs, and the public reveal enables other miners to recompute the score before accepting the head-miner result. If two miners obtain the same score, the tie is resolved deterministically by selecting the miner with the smaller hash of $(pk_i \| H_{prev} \| e)$.

The miner that starts the process of generating the *nonce* value up to the Block generation disseminates a message to stop the other miners and to start the same phase in parallel due to the issue of repeated usage of *previous block hash*. During miner selection, it is assumed that candidates must meet a minimum computational power threshold; hence, all miners are capable of generating the Block with proper efficiency. Once the Block is generated and appended to the local blockchain, it is communicated to miners for sufficient validation. On having validity confirmation of $\frac{3}{4}th$ of miners, it is replicated to the blockchain network through DLT. After a successful Block generation, the other miners begin the next Block generation using the *previous block hash*. This is a repetitive process. A concise depiction of Key processes in Block generation by the Block generator and its interaction with the transaction validator module is shown in Figure 5.

d) Transaction verification: Transaction verification is at the end of Transaction Validators' Cluster (TVC), comprising of $(V_{(Tx)})$ nodes. The miner that picks the transactions bunch generates a random number having value from v_i to v_{n-1} (n is the number of nodes in ***TVC***). The $V_{(T_x)}$ corresponds to the outcome number of this random number generation assigned to the transaction bunch. The $V_{(T_x)}$ picks a transaction from the transaction bunch at a time and verifies it by following the number of encapsulated tasks. Firstly, transaction legitimacy is checked, i.e., whether the transactions are legal, not malicious, and not double-spent. In addition, the transaction syntax and data structure, the input and output values, the transaction size (i.e., must be less than the block size), the transaction value (i.e., must be between the minimum and maximum limit values), and the signature limit are verified. When a transaction is marked as correct, the next step is checking that its matching transaction (i.e., child transaction) exists, and if so, then both the parents and child transactions are processed further by placing them in the transaction pool; otherwise, it is put in the *Orphan Transaction Pool*. Once it passes the initial screening, it is hashed. This process is repeated for all transactions in the transaction bunch, and the *Merkle tree* is completed, yielding the *Merkle Root*. To maintain privacy while obtaining the desired data from the neighboring node to verify the payment, *Bloom Filter* is part of the transaction verification process. It comprises various patterns to aggregate data as defined in *BIP37*. Fig. 6 depicts the transaction processing flow as a block diagram, starting with the availability of currency in the wallet and ending with committing the transaction to lock the wallet. This figure presents another schematic diagram (at the abstract level) of Fig. 3.

2) Slashing: The cheater participant (miner and transaction verified) is penalized by having its stake confiscated. This removal of the cheater's stake reduces the token supply on the blockchain, thereby elevating the token price at the individual level. This is another way for bad actors to cover up the impact of their dishonesty. Also, this removed stake is non-transferable and is directly dumped for no use. The full process of leader election, transaction validation, mining, and consensus is detailed in Algorithm 2.

The transaction verification algorithm details the procedure performed by a dedicated validator node. It checks the validity of each transaction in a batch and computes the final Merkle Root, as shown in Algorithm 3.

D. Security Considerations and Adversarial Model

NCABS is designed primarily for a permissioned or consortium blockchain setting in which validator identities are known and eligibility is controlled through the computational-power and token-wealth thresholds described in the architectural model. This membership assumption is important for security. A Sybil adversary cannot freely create unlimited validator identities unless it also satisfies the admission criteria for each identity. The slashing rule further discourages strategic behavior by penalizing miners or transaction validators that submit inconsistent election reveals, invalid validation results, or malformed block data.

The 75% confirmation threshold provides a Byzantine-fault-oriented safety condition. In a validator set of size N, an adversary controlling fewer than one-fourth of the confirming miners cannot independently finalize an invalid block because the block must receive at least $\frac{3}{4}N$ valid confirmations. This threshold also reduces the likelihood of forks because blocks generated from stale or conflicting previous hashes do not receive sufficient confirmations. However, a large colluding majority remains a serious risk in any consensus protocol; 51% attacks have been shown to affect cryptocurrency value and trust [24]. Therefore, NCABS should be interpreted as resistant under the stated permissioned-membership and honest-majority assumptions, rather than as a complete defense against majority capture in an open permissionless network.

Double-spending attacks are addressed at the transaction-verification layer. The selected transaction validator checks transaction legitimacy, syntax, input and output values, signatures, transaction size, and double-spending conditions before generating the Merkle root. A transaction batch that fails these checks is rejected before block assembly. The current work evaluates these mechanisms from a protocol-design and performance perspective; systematic adversarial simulation, including coordinated Sybil attempts, double-spending campaigns, selfish mining, and majority-capture scenarios, remains part of the future validation plan.

E. Complexity Analysis

The computational complexity of NCABS is reduced compared to PoW due to selective miner participation. In

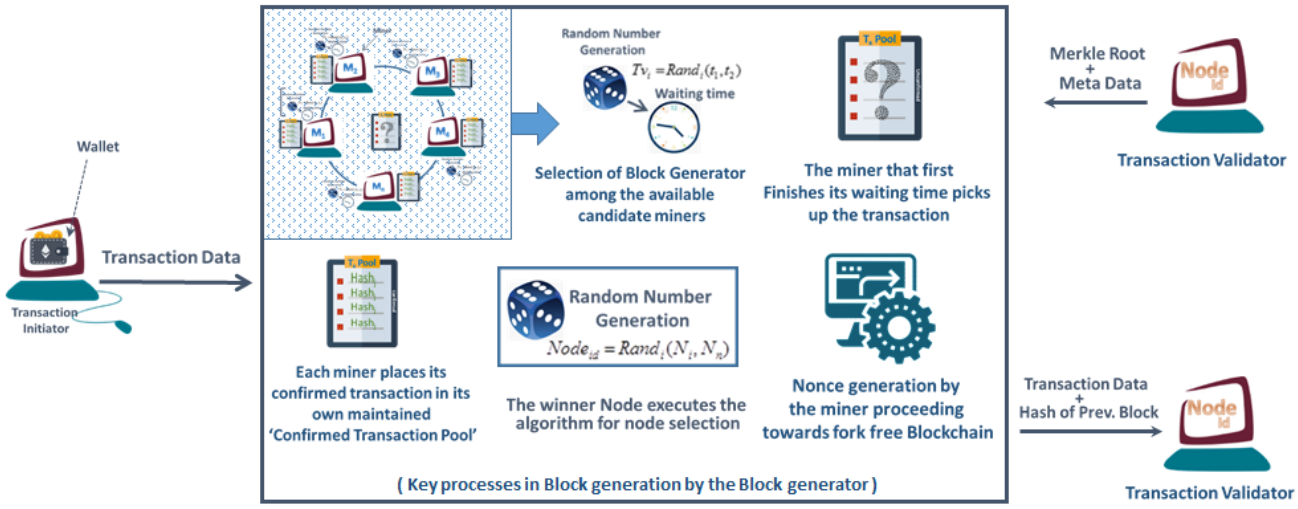

Fig. 5. Key processes in block generation by the Block generator and its interaction with transaction validator module.

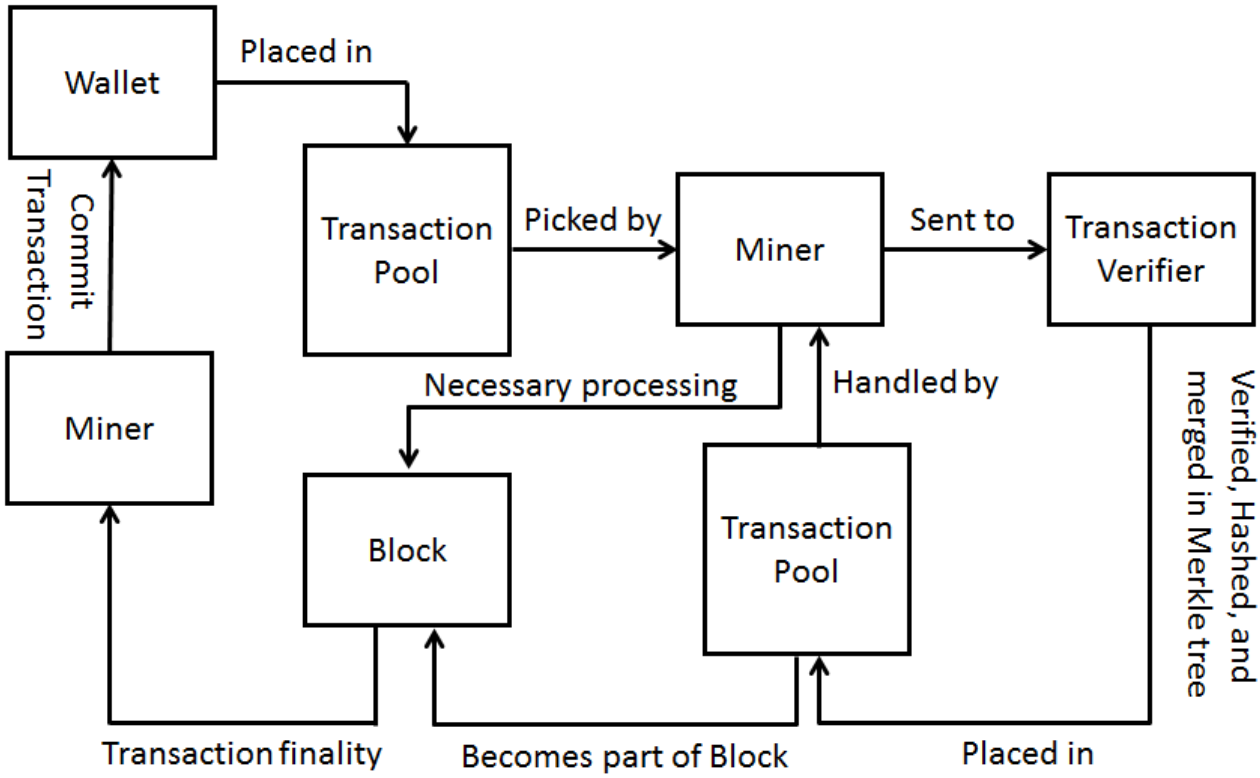

Fig. 6. Transaction processing flow in the proposed blockchain.

traditional PoW, $O(N)$ miners compete to solve the same block-generation problem. In NCABS, only $O(k)$ selected miners participate in the active mining phase, where $k \ll N$. The corresponding computational reduction ratio is therefore approximately k/N. From the communication perspective, PBFT has $O(N^2)$ message complexity because of its all-to-all

Algorithm 2 Proposed Consensus and Block Creation Process

procedure BLOCKCREATIONPROCESS($H_{prev}, Pool_{U-Tx}, \mathcal{M}, \mathcal{V}$)
▷ **Phase 1: Block Generator Election**
  Initialize timers $T_i \leftarrow \infty$ for all $M_i \in \mathcal{M}$
  for each miner $M_i \in \mathcal{M}$ in parallel **do**
    Generate random seed u_i using a CSPRNG
    Broadcast commitment $c_i \leftarrow H(u_i \| pk_i \| H_{prev} \| e)$
    Reveal u_i after the commitment window closes
    Compute score $r_i \leftarrow H(u_i \| pk_i \| H_{prev} \| e)$
    Set timer $T_i \leftarrow r_i$
  end for
  Wait for the first timer to expire
  $M_{gen} \leftarrow \arg\min_{M_i \in \mathcal{M}}(T_i)$
  M_{gen} broadcasts a halt signal to all other miners in $\mathcal{M}$
▷ **Phase 2: Transaction Validation Delegation**
  $\mathcal{T}_{bunch} \leftarrow M_{gen}$ selects a batch of transactions from $Pool_{U-Tx}$
  $V_{Tx} \leftarrow$ randomly select a validator from $\mathcal{V}$
  Send $\mathcal{T}_{bunch}$ to the selected validator V_{Tx}
  $(\mathcal{T}_{valid}, H_{Merkle}) \leftarrow$ VERIFYTRANSACTIONS($\mathcal{T}_{bunch}$)
  if H_{Merkle} is null **then**
    return Block Generation Failed (Validation Error)
  end if
▷ **Phase 3: Mining (Proof-of-Work)**
  Initialize $Nonce \leftarrow 0$
  $H_{header} \leftarrow$ CreateHeader(H_{prev}, H_{Merkle}, timestamp, ...)
  while $H(H_{header}, Nonce) > D_T$ **do**
    $Nonce \leftarrow Nonce + 1$
  end while
▷ **Phase 4: Block Assembly and Consensus**
  $B_{new} \leftarrow$ AssembleBlock($H_{header}, Nonce, \mathcal{T}_{valid}$)
  M_{gen} broadcasts B_{new} to all miners in $\mathcal{M}$
  Initialize $confirm\_count \leftarrow 1$ ▷ M_{gen} votes for its own block
  for each miner $M_j \in \mathcal{M} \setminus \{M_{gen}\}$ **do**
    if M_j validates B_{new} as correct **then**
      $confirm\_count \leftarrow confirm\_count + 1$
    end if
  end for
  if $confirm\_count \geq \frac{3}{4}|\mathcal{M}|$ **then**
    Append B_{new} to the global blockchain
    return B_{new}
  else
    Discard B_{new}
    return Block Generation Failed (Consensus Not Reached)
  end if
end procedure

Algorithm 3 Transaction Verification and Merkle Root Generation

procedure VERIFYTRANSACTIONS($\mathcal{T}_{bunch}$)
  $\mathcal{T}_{valid} \leftarrow \emptyset$
  for each transaction $Tx_i \in \mathcal{T}_{bunch}$ **do**
    if $\Phi(Tx_i)$ is true **then**
      $\mathcal{T}_{valid} \leftarrow \mathcal{T}_{valid} \cup \{Tx_i\}$
    end if
  end for
  if $\mathcal{T}_{valid}$ is not empty **then**
    $H_{Merkle} \leftarrow$ MerkleRoot($\mathcal{T}_{valid}$)
    return $(\mathcal{T}_{valid}, H_{Merkle})$
  else
    return $(\emptyset, \text{null})$
  end if
end procedure

voting structure, whereas NCABS reduces the active consensus communication to approximately $O(kN)$. Hence, NCABS reduces both computational and communication overhead when the selected active set remains much smaller than the complete node set.

VI. Experimentation and Performance Evaluation

A. Performance Metrics

The following metrics are used to evaluate the performance of the proposed model [see Eq. (19) to Eq. (21)]:

$$\text{Throughput} = \frac{\text{Number of Transactions}}{\text{Time}} \tag{19}$$

$$\text{Latency} = \text{Time per Block Confirmation} \tag{20}$$

$$\text{Efficiency} = \frac{\text{Throughput}}{\text{Resource Usage}} \tag{21}$$

Each experiment is repeated 10 times, and average values with standard deviation are reported to ensure robustness. To validate the performance of the proposed NCABS consensus mechanism, a series of experiments was conducted and compared against the Practical Byzantine Fault Tolerance (PBFT) algorithm. The evaluation was performed using a setup that included specialized GPU chips, the CGminer software package, and membership in the Poolin online mining pool. The core performance metrics for this comparative analysis were throughput, network latency, network scalability, and responsiveness. Experiments were conducted across a range of network node densities (40–140) to assess the system's behavior under increasing load. The results consistently demonstrated that NCABS offers superior performance, achieving higher throughput, lower resource utilization, a higher transaction commit rate, and lower latency than PBFT. For example, NCABS reduces aggregate communicated messages by approximately 45% in the scalability experiment, and the responsiveness experiment shows a latency reduction from about 310 ms for PBFT to about 195 ms for NCABS at 40 nodes.

B. Theoretical Evaluation

Usually, miners experience latency when hearing about transaction validation at different points in their life cycle. Furthermore, different miners select transactions based on their convenience (e.g., transaction fees) and begin validating them to perform the activities required for block generation. But only one block is accepted based on consensus, to be appended to the blockchain and then replicated across the network of miners using *Distributed Ledger Technology*. Both processes are time-and computational resource-intensive. In the proposed solution, both issues are addressed by assigning the transaction verification role to a dedicated node, *i.e. Transaction verifier*. Therefore, miners do not need to remain engaged in this process, and these resources are better used to keep the system up and running.

The input transactions are readily picked by the miners and put in the processing queue for further assignment to the transaction verifier. This utilizes available resources efficiently and effectively, increasing the system throughput *i.e.*, more transactions are processed and reach the finality state per unit time. Miners have more time to dedicate to block-generation processes, such as nonce generation, without competing with one another, resulting in more efficient resource utilization and higher throughput. A single transaction is picked by only one miner rather than multiple miners (as is practised in other blockchains). It reaches finality with a high probability, as there is no risk or uncertainty of dropping or deselecting the generated block at the end. What comes in the flow will gain the finality status. The verifying nodes are well aligned with their assigned tasks, without redundant effort on the same transaction (which may become part of multi-transaction bunches picked by various miners on their turn). *Transaction verifier* works with one miner at a time, while one miner can assign transaction bunches to more than one transaction verifier. This further contributes to effective resource utilization with better block throughput.

During the block generation, the winning miner prevents the other miners from completing their waiting time to pick up the transaction batch. This algorithm has twofold benefits. It saves time wasted unnecessarily and contributes to better system throughput. Secondly, it helps them autonomously synchronize and enter the transaction-picking phase again almost simultaneously, *i.e.*, a fair deal. By using the *temporal analysis*, the introduced nonce-generation process results in the block generation taking a bit longer. If the set procedure in Bitcoin and other blockchains is followed, carrying out the process simultaneously is more time-consuming. Although the temporal impact is not well appreciated in the proposed mechanism, it is far less than the usual process, as mentioned before. Selecting transactions redundantly across multiple miner picks also delays the block generation process.

C. Implementation Detail

Specialized GPU chips, sufficient cooling for the hardware, an always-on internet connection, a mining software package, and membership in both an online cryptocurrency exchange and an online mining pool are the major implementation parameters. The market is well-supplied with crypto-mining software packages, including CGminer, Bitminter, BFGminer, Multiminer, and EasyMiner. CGminer is used as the representative mining software in this implementation because it supports multi-threaded and multi-pool mining configurations [25]. A group of crypto-miners who combine their computational resources over a network can increase their collective computational power. Poolin, Autopool, Slush Pool, Bitfury, and FZpool are among the widely used mining pools. Poolin is used as the representative mining-pool setting in this study and was reported to mine approximately 20% of blocks at the time considered.

1) Performance metrics: To validate the proposed consensus mechanism's performance, it is compared with PBFT across various evaluation parameters. Those are summarized into throughput, network latency, network scalability, and responsiveness. The results are presented from Fig. 7 to Fig. 17. The detailed discussion is given in subsequent sections.

a) Throughput and network latency: To measure the concurrency in the Blockchain system, an important metric is transaction throughput per Second $T_(TPS)$, i.e., $P = \frac{T_(sum)}{\delta t}$ where $T_(sum)$ is block time and δt is the total transaction count during that block time. It reflects the number of transactions packaged per unit of time. It states that the higher the throughput value, the greater the consensus efficiency and the greater the potential for transaction processing. Hence, $Throughput \propto ConcurrencyEfficiency$. Transaction latency is the time between submitting a transaction and its confirmation. It measures the blockchain's consensus algorithm's computational efficiency, communication efficiency, and runtime. Reduced latency will increase transaction confirmations and decrease the risk of blockchain bifurcation.

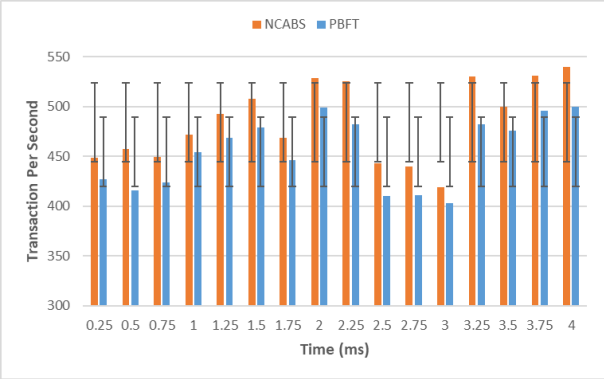

Fig. 7. Average results of throughput - time(ms) vs. transaction per second.

$$Trans_{(Latency)} = Block_{(Conf)}t - Trans_{(Init)}t$$

where, $Trans_{(Latency)}$ is the transaction latency that is the difference between the block confirmation time $Block_{(Conf)}t$ and the transaction initiation time $Trans_{(Init)}t$. Fig. 7 shows the average result of throughput, time vs. transactions per second, and reflects the outperforming of NCABS to PBFT. The variability in results over time indicates changes in the nodes' status. In NCABS, $Power_{(Comp)}$ and $Wealth_{(Tkn)}$ are two parameters used to categorize nodes into Block validators and transaction validators. The status change also affects throughput, as shown in Fig. 7 and Fig. 8. This is also shown in Fig. 9, which compares the change in throughput to the transaction commit rate (Fig. 10). The same also impacts the

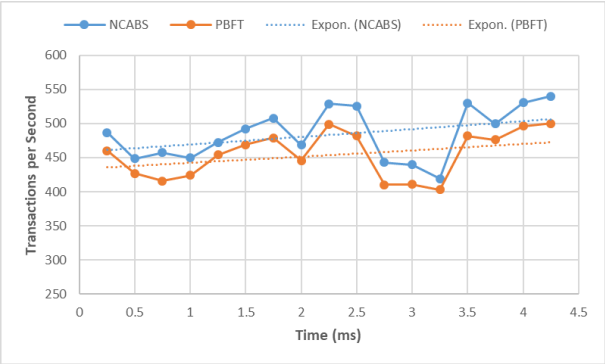

Fig. 8. Exponential trend of competing algorithms on throughput, time(ms) vs. transaction per second.

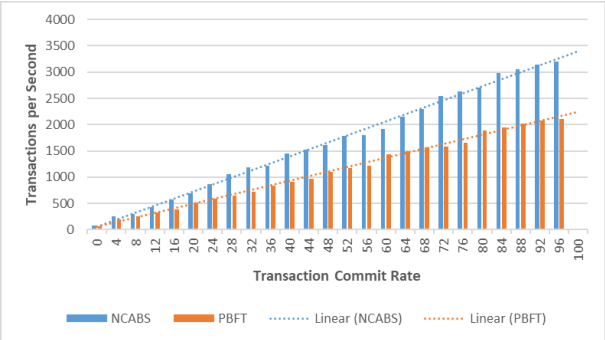

Fig. 9. Transaction commit rate vs. transaction per second for the competing algorithms.

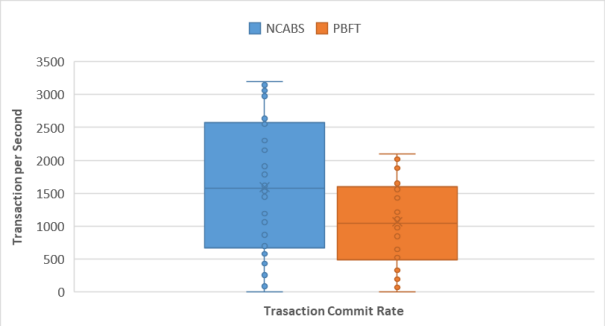

Fig. 10. A summarized view of the throughput comparison of NCABS and PBFT.

graph representing the blockchain network resource utilization, as in Fig. 11. Consequently, this status change also affects message communication since it increases inter-node message passing for node status reallocation. That is reflected in Fig. 11 and Fig. 12.

Across all the above-cited figures, the comparison of NCABS with PBFT for throughput and resource utilization as performance evaluation parameters also shows the network's stability. The higher the frequency of switching the node status, the greater its negative impact on the network. The variation in throughput and resource utilization over time is due to varying node involvement. At t_0, a fraction of nodes serve as validators or supervisor nodes, while at t_1, upon the status

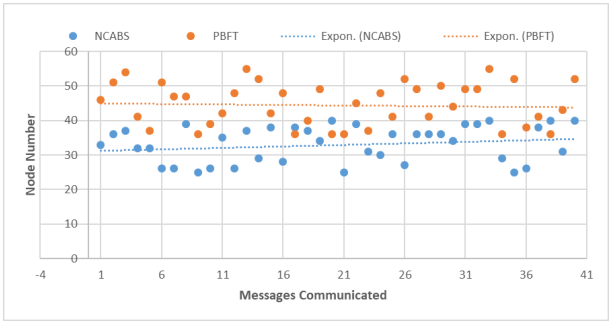

Fig. 11. Blockchain network resource utilization. A comparative view of the per node messages communicated by NCABS and PBFT along with their exponential trend.

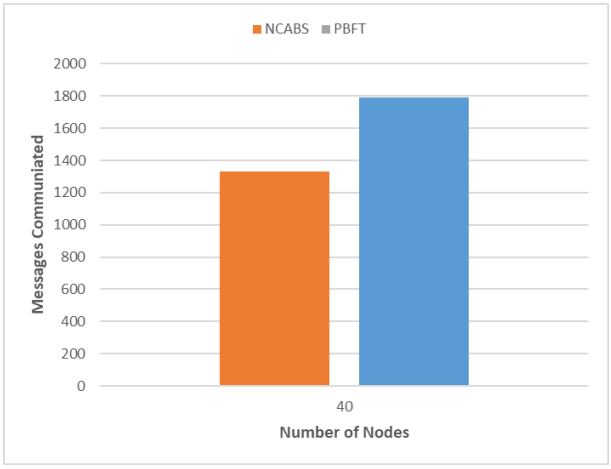

Fig. 12. Blockchain network resource utilization. An aggregated comparative view of the messages communicated in the blockchain network of 40 nodes.

switch, another fraction of nodes assume the same role. This plays a role in the said graph style. In short, as explained above, NCABS achieved higher throughput, lower resource utilization, a higher transaction commit rate, and lower latency than PBFT. Considering Fig. 9, it is clearly evident that the higher the transaction commit rate, the more the throughput, i.e., transactions per second. Over time, throughput increases along with the transaction commit rate for NCABS, and it is higher than that of PBFT. The performance of NCABS is improving compared to PBFT. The reason for this better result is the minimal communication during the validation process. Extending this, the time required to confirm a transaction is also shorter for NCABS than for PBFT.

b) Network scalability: Among many other scalability solutions in Blockchain that require close attention for improvement, throughput and latency are the most important concerns. Adding more nodes significantly increases the time needed to confirm transactions, reducing the number of transactions per second. To address the weakness of the slow transaction verification process, as in the founding systems and their descendant solutions, this job is at the end of the Transaction Validators' Cluster, which comprises selected transaction validator nodes. The same approach is used in PBFT and its variants. One approach to improve the scalability or speed of transaction processing is to design a faster proof-

of-work solution. NCABS uses a method to accelerate the proof-of-work process by parallel rather than solo mining. This decreases the overall burden on the Blockchain network nodes and helps ensure that no more than two miners exert the same effort on a given block.

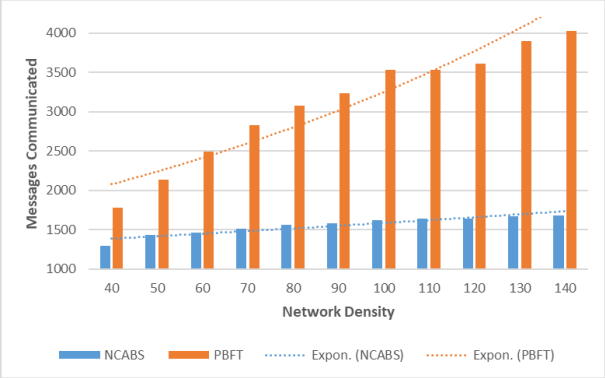

Fig. 13. Comparative results of network scalability over varied network densities in NCABS and PBFT, along with their exponential trend.

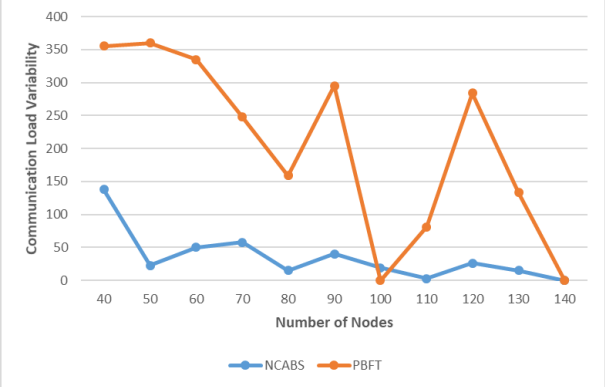

Fig. 14. Communication load variability over varied Number of Nodes in NCABS and PBFT.

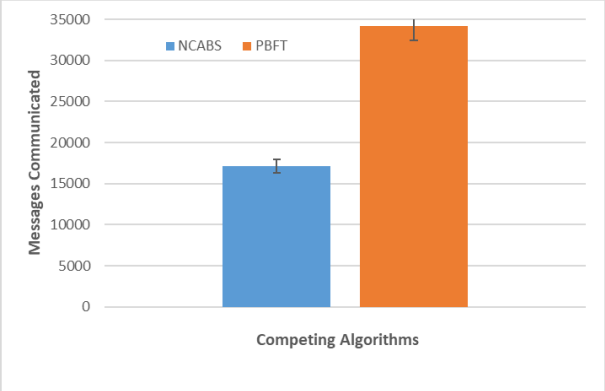

Fig. 15. Network scalability.

Fig. 13, Fig. 14, and Fig. 15 show the network load leading to Network scalability at different network densities. The results shown in the figures are averages across multiple simulations. The results are obtained at varying numbers of nodes, ranging from 40 to 140, with a difference of 10 nodes per experiment. Fig. 15 shows the summary of the inferred and calculated results (averaged) from all the experiments with different blockchain network node densities. It reflects the response of message communication to set up the transaction's confirmation. The total number of messages communicated by the proposed mechanism, NCABS, is approximately 45% fewer than that of PBFT. Referring to Fig. 13, the difference in the number of communicated messages for a network with a node density of 40 and a node density of 140 is huge, especially in the case of PBFT. This difference is due to network instability, specifically to the redundancy introduced by multiple miners mining the same transaction. In contrast, NCABS follows the one-time, one-node policy, as discussed previously, to reduce this communication load. Fig, 13 and Fig. 14 also endorse the network stability scenario. NCABS is far more stable compared to PBFT. Since the variability in the communication load is minimal and gradually improving in various node densities from 40 to 140 in NCABS. In the case of PBFT, this difference increases exponentially. It performs better for small-scale networks but worsens as network scale increases. Also, the exponential trend line for the resulting PBFT graph is much higher and steeper than that for NCABS.

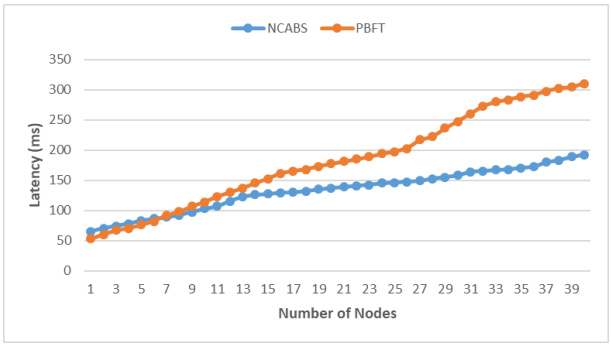

Fig. 16. Network latency under increasing node density for NCABS and PBFT.

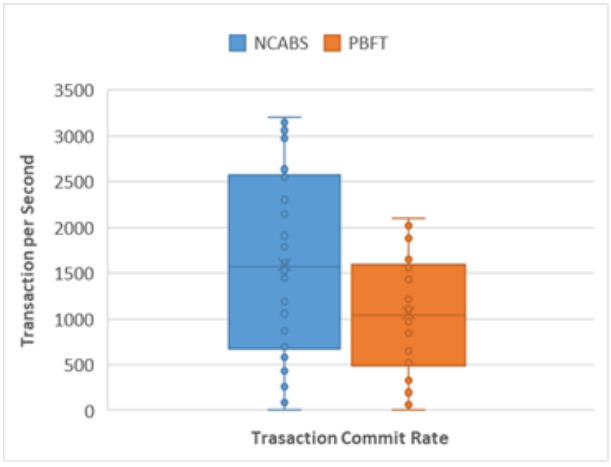

Fig. 17. Aggregated responsiveness comparison using transaction commit rate.

The data in Fig. 13 are also represented as a line graph in Fig. 14, showing far less variability in communication

TABLE III. COMPARISON OF CONSENSUS ALGORITHMS

Algorithm	Communication Complexity	TPS	Groupings of Nodes	Nodes Participation in Consensus Mechanism	Supervision Mechanism	Scalability
PBFT [26]	Average	Low	No	No	No	low
Multi-Layer PBFT [27]	Average	Average	Yes	No	No	Average
NBFT [28]	High	Average	Yes	No	No	Average
GH [29]	High	Low	Yes	No	No	Average
CDBFT [30]	Average	Low	No	Yes	No	Poor
RBFT [31]	High	Average	No	No	Yes	Poor
Algorithm [32]	High	Low	Yes	No	No	Average
PoW [33]	High	Low	No	No	No	Poor
PoS [34]	High	Low	No	No	No	Poor
DPoS [35]	High	Average	No	No	Yes	Average
NCABS (Proposed)	Low	High	Yes	Yes	Yes	Good

load for NCABS compared to that of PBFT. Hence, it results in better network scalability. Since network stability, communication, and network throughput are directly related to network scalability, NCABS delivers far better scalability than its competitor. Fig. 16 and Fig. 17 depict the results of network responsiveness from the perspective of the number of nodes versus latency in milliseconds. In Fig. 16, keeping the node density up to forty, the time interval from transaction submission to transaction confirmation is almost the same for both algorithms for thirteen nodes somewhere in the network. As network latency measures the computational efficiency and runtime of the blockchain's consensus algorithm, throughput is also closely related to it. But later, the latency difference in PBFT increases from node thirteen onward. This decreases its throughput, impacting network scalability. The aggregated responsiveness results are also shown in Fig. 17, indicating that higher latency is associated with lower responsiveness and lower scalability, and vice versa.

D. Comparative Analysis of Typical Consensus with NCABS

Table III compares consensus algorithms, focusing on the extended PBFT ideas. A similar comparison is made by Wu X. *et al.* [36]. The comparison indicators are communication complexity (which also includes communication overhead), Throughput (TPS), nodes' grouping (aka sharding), nodes' participation in the consensus mechanism, supervision of the consensus process, and scalability. The empirical evaluation uses PBFT as the primary baseline because it is the closest directly implemented Byzantine-fault-tolerant comparator in the experimental setup. The remaining consensus protocols are included as qualitative baselines to clarify how NCABS differs from common PBFT variants, PoW, PoS, DPoS, and related group-based protocols. As far as comparing Proof of Work (PoW) and Proof of Stake (PoS), considering the open environment for public blockchain, PoW and PoS are more suitable. Comparing these two, the PoS algorithm prioritizes the highest equity when electing nodes, consumes the least energy, and exhibits a higher degree of centralization than the PoW algorithm, but it lacks security. Its optimized version, DPoS, fine-tunes the number of participating nodes. PoW has stronger decentralisation, a relatively simple implementation, and high fault tolerance, but with the drawbacks of slow transaction speeds, higher energy consumption, and lower throughput [37].

The GH algorithm divides the network nodes into groups, each with a primary node. Local consensus leads to the global consensus achieved by the collective work of only the primary nodes of the designed groups. This increases communication complexity by increasing overhead among interacting nodes. The same is true for NBFT, RBFT, PoW, and its variants, i.e., PoS and DPoS. Hence, it also impacts transaction throughput. Data processing speed has been significantly increased, and security has been improved, but the decentralization aspect needs further optimization. The Paxos, Raft, and PBFT algorithms are better suited to permissioned chains, such as consortium and private chains, that require identity verification and have limited node counts. These consensus algorithms can provide excellent information-processing capabilities, low resource consumption, and tolerance to non-Byzantine errors. However, implementing the Paxos consensus algorithm is more difficult and only suitable for consortium chains and private chains with relatively high security. The Raft algorithm is more practical and performs similarly to Paxos. Still, its shortcomings are similar to those of the Paxos consensus algorithm, and it is only suitable for consortium and private chains with relatively high security. The PBFT consensus algorithm offers strong data processing capabilities and enhanced security and is only applicable to consortium and private chains. In addition, the PoH and CW-PoW consensus algorithms proposed in 2021 are improvements and optimizations of the typical consensus algorithm, and they feature high computational efficiency, low energy consumption, and strong safety and reliability. NCABS is well-suited for the permissioned blockchain for the same reason as PBFT. A detailed experimental-based comparison of these two algorithms is made in the previous section. The comparative analysis results for the algorithms above are shown in Table III.

VII. DISCUSSION

The experimental results show that the main advantage of NCABS comes from reducing redundant work during transaction verification and block generation. In PBFT, consensus communication grows rapidly with the number of participating nodes because voting and message exchange are broadly distributed across the validator set. NCABS narrows the active consensus path by selecting a head miner, delegating transaction verification to a specific validator, and requiring a 75% confirmation threshold before replication. This design

explains why the scalability experiment reports approximately 45% fewer communicated messages than PBFT and why the latency curve remains lower for NCABS as node density increases.

The proposed method is also consistent with the related-work discussion on consensus-based scalability. PoW improves openness and decentralization but wastes computation because many miners compete for a single accepted block. PBFT improves deterministic finality in permissioned environments but can suffer from high communication overhead. PBFT variants such as Multi-Layer PBFT, NBFT, CDBFT, RBFT, and HotStuff-inspired approaches attempt to reduce this overhead through grouping, leadership, or pipelined voting. NCABS follows the same broad goal of reducing redundant coordination, but it does so through controlled parallel mining and explicit separation between transaction validation and block generation. This separation is the main methodological difference from the compared baselines in Table III.

From a deployment perspective, NCABS is best interpreted as a permissioned or consortium-chain mechanism. Its security assumptions depend on controlled validator admission, verifiable random selection, slashing, and an honest-majority condition under the 75% confirmation threshold. These assumptions make NCABS suitable for applications such as supply-chain blockchain systems, where participant identities and computational eligibility can be managed. They also limit direct transfer to open permissionless networks without additional defenses against Sybil identities, selfish mining, coordinated double spending, and majority capture.

The comparison with PBFT provides a direct empirical baseline because PBFT is the closest implemented Byzantine-fault-tolerant comparator in the present evaluation. The qualitative comparison with additional consensus protocols strengthens the positioning of NCABS but does not replace full empirical benchmarking against all modern variants. Therefore, the results should be read as evidence that NCABS improves performance under the stated experimental conditions, while broader adversarial and multi-baseline testing is required before generalizing the claim to all blockchain consensus settings.

VIII. Conclusion

This study presented NCABS, a controlled parallel-mining consensus mechanism for improving blockchain scalability. The conclusions of the study are as follows:

- NCABS reduces redundant mining and transaction-verification effort by separating head-miner selection, validator delegation, block generation, and confirmation into coordinated stages.
- Across ten repeated experiments with node densities from 40 to 140, NCABS achieves better throughput, lower communication overhead, and lower latency than PBFT; in the reported scalability result, aggregate communicated messages are reduced by approximately 45%.
- The comparative analysis indicates that NCABS is particularly suitable for permissioned or consortium blockchain applications where validator identity, computational eligibility, and confirmation thresholds can be managed.

Two limitations remain. First, the current evaluation is simulation/prototype-based and uses PBFT as the direct empirical baseline, while other modern consensus variants are compared qualitatively. Second, the framework assumes minimum computational capability among miners, controlled validator admission, and honest-majority behavior; adversarial experiments involving Sybil behavior, double spending, selfish mining, and majority capture have not yet been conducted.

Future work will extend NCABS through broader empirical benchmarking against additional PBFT variants and modern consensus protocols, real deployment testing, and adversarial evaluation under coordinated attack scenarios.

Acknowledgment

This work was supported and funded by the Deanship of Scientific Research at Imam Mohammad Ibn Saud Islamic University (IMSIU) (grant number: IMSIU-DDRSP2604).

References

[1] R. M. A. Latif, M. Farhan, O. Rizwan, M. Hussain, S. Jabbar, and S. Khalid, "Retail level blockchain transformation for product supply chain using truffle development platform," *Cluster Computing*, vol. 24, pp. 1–16, 2021.

[2] S. Jabbar, H. Lloyd, M. Hammoudeh, B. Adebisi, and U. Raza, "Blockchain-enabled supply chain: analysis, challenges, and future directions," *Multimedia systems*, vol. 27, pp. 787–806, 2021.

[3] M. A. Habib, M. B. Sardar, C. N. F. Faisal, and M. Nasir, "Blockchain-based supply chain for the automation of transaction process: Case study based validation," in *2020 International Conference on Engineering and Emerging Technologies (ICEET)*. IEEE, 2020, pp. 1–7.

[4] R. Sujatha, V. Mareeswari, J. M. Chatterjee, A. A. A. Mousa, and A. E. Hassanien, "A bayesian regularized neural network for analyzing bitcoin trends," *IEEE access*, vol. 9, pp. 37 989–38 000, 2021.

[5] M. Abdallah, O. A. Dobre, P.-H. Ho, S. Jabbar, M. J. Khabbaz, and J. J. Rodrigues, "Blockchain-enabled industrial internet of things: Advances, applications, and challenges," *IEEE Internet of Things Magazine*, vol. 3, no. 2, pp. 16–18, 2020.

[6] J. Tian, Y. Li, and C. Jing, "Rvre: a secure and efficient committee-based consensus protocol leveraging reputation power," *The Computer Journal*, p. bxag019, 2026.

[7] L. Yin, Y. Liu, J. Shi, and Y. Zhang, "A general and efficient redactable blockchain scheme based on voting committee," *The Computer Journal*, p. bxag024, 2026.

[8] L. Ke, C. Hsu, M. H. Au, and Z. Xia, "A practical blockchain-based vaccine supply management framework with verifiability and traceability," *The Computer Journal*, p. bxaf115, 2025.

[9] G. A. F. Rebello, G. F. Camilo, L. A. C. de Souza, M. Potop-Butucaru, M. D. de Amorim, M. E. M. Campista, and L. H. M. Costa, "A survey on blockchain scalability: From hardware to layer-two protocols," *IEEE Communications Surveys & Tutorials*, vol. 26, no. 4, pp. 2411–2458, 2024.

[10] A. F. Mahdi and F. Rabee, "A blockchain mining proof of work approach based on fog computing virtualization for mobile crowdsensing," in *2024 Third International Conference on Distributed Computing and High Performance Computing (DCHPC)*. IEEE, 2024, pp. 1–9.

[11] G. A. F. Rebello, G. F. Camilo, L. A. C. de Souza, M. Potop-Butucaru, M. D. de Amorim, M. E. M. Campista, and L. H. M. Costa, "A survey on blockchain scalability: From hardware to layer-two protocols," *IEEE Communications Surveys & Tutorials*, vol. 26, no. 4, pp. 2411–2458, 2024.

[12] S. Reno and K. Roy, "Navigating the blockchain trilemma: A review of recent advances and emerging solutions in decentralization, security, and scalability optimization." *Computers, Materials & Continua*, vol. 84, no. 2, 2025.

[13] C. K. Lao, S. Zhou, L. Zhang, F. Zhang, and K. Y. Wang, "Unpacking long-latency transactions in ethereum," in *Proceedings of the Workshop on Decentralized Finance and Security*, 2024, pp. 11–20.

[14] S. Lee and H. Kim, "On the robustness of lightning network in bitcoin," *Pervasive and Mobile Computing*, vol. 61, p. 101108, 2020.

[15] X. Wang, H. Zhu, Z. Ning, L. Guo, and Y. Zhang, "Blockchain intelligence for internet of vehicles: Challenges and solutions," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 4, pp. 2325–2355, 2023.

[16] L. T. Thibault, T. Sarry, and A. S. Hafid, "Blockchain scaling using rollups: A comprehensive survey," *IEEE Access*, vol. 10, pp. 93 039–93 054, 2022.

[17] J. W. Heo, G. S. Ramachandran, A. Dorri, and R. Jurdak, "Blockchain data storage optimisations: a comprehensive survey," *ACM Computing Surveys*, vol. 56, no. 7, pp. 1–27, 2024.

[18] A. Imteaj, U. Thakker, S. Wang, J. Li, and M. H. Amini, "A survey on federated learning for resource-constrained iot devices," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 1–24, 2021.

[19] R. Neiheiser, M. Matos, and L. Rodrigues, "Kauri: Scalable bft consensus with pipelined tree-based dissemination and aggregation," in *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles*, 2021, pp. 35–48.

[20] G. A. F. Rebello, G. F. Camilo, L. A. C. de Souza, M. Potop-Butucaru, M. D. de Amorim, M. E. M. Campista, and L. H. M. Costa, "A survey on blockchain scalability: From hardware to layer-two protocols," *IEEE Communications Surveys & Tutorials*, vol. 26, no. 4, pp. 2411–2458, 2024.

[21] B. Lashkari and P. Musilek, "A comprehensive review of blockchain consensus mechanisms," *IEEE access*, vol. 9, pp. 43 620–43 652, 2021.

[22] Y. Shahsavari, K. Zhang, and C. Talhi, "Performance modeling and analysis of hotstuff for blockchain consensus," in *2022 Fourth International Conference on Blockchain Computing and Applications (BCCA)*. IEEE, 2022, pp. 135–142.

[23] I. M. Coelho, V. N. Coelho, R. P. Araujo, W. Yong Qiang, and B. D. Rhodes, "Challenges of pbft-inspired consensus for blockchain and enhancements over neo dbft," *Future Internet*, vol. 12, no. 8, p. 129, 2020.

[24] S. Shanaev, A. Shuraeva, M. Vasenin, and M. Kuznetsov, "Cryptocurrency value and 51% attacks: evidence from event studies," *The Journal of Alternative Investments*, vol. 22, no. 3, pp. 65–77, 2019.

[25] C. Kolivas, "multi-threaded multi-pool fpga and asic miner for bitcoin," Last accessed 12 September 2025. [Online]. Available: https://github.com/ckolivas/cgminer

[26] H. Xu, J. Wu, Q. Pan, X. Guan, and M. Guizani, "A survey on digital twin for industrial internet of things: Applications, technologies and tools," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 4, pp. 2569–2598, 2023.

[27] W. Li, C. Feng, L. Zhang, H. Xu, B. Cao, and M. A. Imran, "A scalable multi-layer pbft consensus for blockchain," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 5, pp. 1146–1160, 2020.

[28] J. Yang, Z. Jia, R. Su, X. Wu, and J. Qin, "Improved fault-tolerant consensus based on the pbft algorithm," *IEEE Access*, vol. 10, pp. 30 274–30 283, 2022.

[29] R. Jabbar, E. Dhib, A. B. Said, M. Krichen, N. Fetais, E. Zaidan, and K. Barkaoui, "Blockchain technology for intelligent transportation systems: A systematic literature review," *IEEE Access*, vol. 10, pp. 20 995–21 031, 2022.

[30] W. Issa, N. Moustafa, B. Turnbull, N. Sohrabi, and Z. Tari, "Blockchain-based federated learning for securing internet of things: A comprehensive survey," *ACM Computing Surveys*, vol. 55, no. 9, pp. 1–43, 2023.

[31] J. Xiao, T. Luo, C. Li, J. Zhou, and Z. Li, "Ce-pbft: A high availability consensus algorithm for large-scale consortium blockchain," *Journal of King Saud University-Computer and Information Sciences*, vol. 36, no. 2, p. 101957, 2024.

[32] R. Chen, L. Wang, C. Peng, and R. Zhu, "An effective sharding consensus algorithm for blockchain systems," *Electronics*, vol. 11, no. 16, p. 2597, 2022.

[33] W. Issa, N. Moustafa, B. Turnbull, N. Sohrabi, and Z. Tari, "Blockchain-based federated learning for securing internet of things: A comprehensive survey," *ACM Computing Surveys*, vol. 55, no. 9, pp. 1–43, 2023.

[34] J. Xu, C. Wang, and X. Jia, "A survey of blockchain consensus protocols," *ACM Computing Surveys*, vol. 55, no. 13s, pp. 1–35, 2023.

[35] A. Gangwal, H. R. Gangavalli, and A. Thirupathi, "A survey of layer-two blockchain protocols," *Journal of Network and Computer Applications*, vol. 209, p. 103539, 2023.

[36] X. Wu, W. Jiang, M. Song, Z. Jia, and J. Qin, "An efficient sharding consensus algorithm for consortium chains," *Scientific Reports*, vol. 13, no. 1, p. 20, 2023.

[37] M. Wendl, M. H. Doan, and R. Sassen, "The environmental impact of cryptocurrencies using proof of work and proof of stake consensus algorithms: A systematic review," *Journal of Environmental Management*, vol. 326, p. 116530, 2023.