UDC 004.72
DOI: 10.15587/2706-5448.2026.356851

Leonid Chepel,
Yuriy Boyko

PROOF-OF-INDICATORS: DEVELOPMENT AND VALIDATION OF AN ADAPTIVE CONSENSUS MECHANISM FOR INTERNET OF THINGS BLOCKCHAIN NETWORKS

The object of this research is the consensus mechanism in blockchain networks for the Internet of Things (IoT). The expansion of IoT requires a decentralized approach, making blockchain a promising solution. The Proof of Authority (PoA) consensus protocol was identified as the most suitable base for heterogeneous IoT, however, it has limitations regarding fork occurrence, data duplication, and the signer selection process.

The obtained results include the creation of the Proof of Indicators (PoI) consensus protocol that optimizes IoT network by reducing block size and prioritizing capable nodes for consensus tasks for devices with different performance and network conditions. PoI is based on Go-Ethereum's PoA Clique implementation; and a comparative performance analysis was conducted between PoI and Clique in a simulated IoT network.

Testing shows that PoI reduces overall network traffic by 20.5% and decreases network forks by 80%. Under testing, PoI improves transaction throughput and decreases block propagation time, compared to Clique. These gains occur with a modest increase in resource consumption: an average rise of 6.5% in CPU usage and 5.4% in memory usage.

A distinctive feature of this work is combining dynamic node selection with light block propagation within the blockchain consensus layer to address previously found limitations.

The PoI system is a suitable solution for secure and purpose-specific blockchain application in IoT, where blockchain node can be hosted on low-powered devices, such as the Raspberry Pi, creating a fully decentralized cloud-independent infrastructure.

Keywords: *proof-of-authority, decentralized networks, embedded systems, node selection, block size optimization.*

Received: 11.01.2026
Received in revised form: 10.03.2026
Accepted: 30.03.2026
Published: 30.04.2026

How to cite
Chepel, L., Boyko, Y. (2026). Proof-of-Indicators: development and validation of an adaptive consensus mechanism for Internet of Things blockchain networks. Technology Audit and Production Reserves, 2 (2 (88)), 15–24. https://doi.org/10.15587/2706-5448.2026.356851

1. Introduction

The Internet of Things (IoT) is rapidly expanding in different fields, including healthcare systems, smart cities, transport, smart homes, security networks, and energy management applications [1]. However, this growth presents significant challenges in data transmission, security, and system management, particularly for small and low-power devices vulnerable to cyberattacks [2].

Traditional cloud-based IoT systems depend on centralized servers and continuous internet connectivity. This creates several problems: system-wide failures when central components malfunction, increased network latency due to data transmission to remote servers, and security risks from sending sensitive data over the Internet [3]. Moreover, many IoT applications require real-time local decision-making that cannot tolerate the delays inherent in cloud-based processing [4].

The quality of IoT data is determined by five factors: accuracy, confidence, trustworthiness, timeliness, and completeness. A primary challenge is handling data streams from heterogeneous IoT devices, which often contain errors due to sensor failures or environmental factors that can lead to poor decisions. Usage of blockchain technology can mitigate this problem, providing solution to data integrity [5].

Effective governance for the IoT requires the management of interoperability, accountability, security, privacy, and trust. These challenges become more complex within large-scale systems. In such contexts, traditional centralized management approaches may not provide the required level of control, whereas usage of governance mechanisms enabled by blockchain technology can address this limitation [6].

The blockchain layer is an additional component in the IoT system architecture. Its purpose is to enhance existing IoT networks. It is not intended to replace current IoT data transfer protocols, including MQTT, AMQP, CoAP, or Zigbee. The focus is on IoT hubs or gateways that operate as blockchain nodes. IoT devices connect to these hubs using standard IoT data transfer protocols. Each hub functions as a blockchain node to form a decentralized IoT blockchain network.

Organizing IoT devices into separate, purpose-specific blockchain networks, local or overlay networks, effectively reduces overall network latency by limiting the number of the participating nodes. Furthermore, this approach improves data security and ensures operational continuity by reducing dependence on internet connectivity. Specifically for this purpose, an optimized blockchain consensus mechanism is required.

ISSN-L 2664-9969; E-ISSN 2706-5448

Blockchain as a peer-to-peer system offers several distinct advantages over traditional technologies, including:

– enhanced security: blockchain provides an immutable and end-to-end encrypted record, which significantly reduces the risk of fraud and unauthorized activities within IoT networks;

– transparency and traceability: by utilizing a distributed ledger, blockchain ensures that transactions and data are recorded identically across multiple locations. Authorized participants in the network gain access to the same information simultaneously, fostering transparency. All transactions are immutably recorded with time and date stamps, allowing the entire history of a transaction to be traceable. This feature minimizes the potential for fraud and ensures accountability;

– automation: blockchain supports the use of "*smart contracts*", enabling automated transactions and enforcing network-level rules for IoT devices and data flows. This automation streamlines processes and enhances operational efficiency [7].

A consensus protocol is a set of rules and procedures that allow multiple computers (nodes) in a distributed network to agree on a single version of data or truth, even when some nodes might fail or act maliciously. The following protocol regulates how new blocks are generated and added to the network.

Traditional blockchain consensus protocols, including Proof of Work (PoW) and Proof of Stake (PoS), often impose high resource demands. Therefore, the adoption of lightweight consensus algorithms that minimize computational and energy requirements is necessary [8].

Consensus protocols for blockchain networks based on directed acyclic graph (DAG) are out of scope for purpose-specific IoT networks as they may occasionally experience unpredictable transaction durations due to the absence of a clearly defined sequence for historical verification. Smart contracts implemented on DAG platforms operate at slower speeds compared to traditional blockchain architectures, as they require conflict resolution across different subgraphs that may also make bigger load to single node compared to traditional blockchain [9].

Performance comparisons indicate that Proof of Authority (PoA), particularly the Clique implementation, is suitable for IoT applications due to its simplicity, low latency, and high throughput according to [10].

Another PoA implementation, Aura, competes with Clique and confirms that PoA protocols are generally compatible with resource-constrained IoT applications. Notably, Clique provides higher availability and transaction throughput than Aura does. Clique becomes an optimal consensus protocol for IoT environments due to its simple single-signer block validation mechanism and ease of implementation compared to other protocols. This simplicity enables deployment on resource-constrained devices, for instance, Raspberry Pi [11].

The Go-Ethereum's project implementation of Clique protocol [12] is production-tested, widely adopted PoA implementation with a lightweight client designed for computationally limited IoT devices. However, not all IoT devices should act as PoA nodes, and they might have more limited computational resources than single-board computers. For these constrained devices, a specialized implementation of the web3 library can be utilized, allowing them to connect to the existing nodes [13].

Existing consensus protocols have a fundamental limitation: they treat all nodes equally, regardless of individual performance capabilities or network conditions. This uniformity creates performance bottlenecks in the IoT networks. Therefore, these problems become more severe when blockchain technology is integrated. While current research focuses on transport protocol enhancements, clustering methods, routing [14], and network optimization techniques via congestion control [15], a different approach may be effective in blockchain network. Optimizing at the consensus level, by prioritizing high-performance nodes with reliable network connections, could improve overall IoT network efficiency.

This article suggests an enhancement to the existing PoA consensus protocol to better accommodate the IoT environment where devices have varying computational capabilities and operate under different network conditions. The suggested consensus protocol is based on the Ethereum Clique implementation and introduces additional features to address IoT-specific challenges.

An IoT-efficient blockchain model should prioritize the following system qualities [16]:

– performance efficiency to operate on resource-constrained hardware;

– reliability to provide fault tolerance and enable recovery;

– security to preserve immutable data in a decentralized ledger through cryptographic mechanisms and consensus protocols;

– flexibility to adapt to network changes and scale with demand.

To improve these qualities specifically for IoT, the suggested protocol incorporates mechanisms to continuously monitor and evaluate the performance metrics of blockchain nodes. Based on these metrics, the system dynamically selects the highest-performing nodes to participate in the consensus process. Furthermore, the protocol should reduce block sizes to optimize network performance. These modifications aim to create a more efficient and adaptive consensus mechanism.

The object of this research is the consensus mechanism for blockchain-enabled IoT networks. *The aim of this research* is to develop and validate an adaptive consensus mechanism called Proof of Indicators (PoI), suitable for efficient usage in purpose-specific blockchain IoT networks. To achieve this aim, next objectives are defined:

1) analyze Clique work principles and identify existing gaps;

2) design theoretical optimization approach, main blockchain node indicators and system behavior based on them for PoI. Theoretically compare with Clique;

3) practically implement PoI-based system;

4) configure test environment for experimental testing and define test scenarios;

5) evaluate tests results of the proposed solution with comparison to Clique.

2. Materials and Methods

The following scientific methods were used:

– analytical method for analyzing Clique work principles and identifying its limitations (objective 1) to understand how the existing protocol operates and its potential gaps before designing any improvements;

– mathematical modelling method for describing node selection logic, penalty functions and fork probability (objective 2) to ensure deterministic and reproducible system behavior during further implementation;

– comparative method for theoretical and experimental comparison of the improved PoI-based system with existing Clique-based system (objectives 2,5) to verify if the proposed approach brings any benefits;

– experimental method for simulating various network conditions for both Clique-based and PoI-based systems (objectives 3–5) to obtain and evaluate practical results.

Software and tools. Ubuntu 24.04 LTS on the Windows 11 WSL2 subsystem was used as the host operating system. The Go programming language (version 1.20.7) was used for PoI consensus protocol development, Solidity (version 0.8) for smart contract creation, and Python (version 3.10) and Bash for testing scripts. The Docker application was used to create isolated network environments for testing. The Linux Traffic Control tool was used to simulate IoT network delays. The WireShark and Edgeshark tools were used for network monitoring in addition to Docker statistics. The LabPlot application (version 2.12) was used for figures creation.

Hardware. All experiments were conducted on a laptop equipped with an AMD Ryzen 5 5500U CPU and 24 GB RAM. Both Docker networks of 15 nodes each were deployed on this machine.

Research methodology. The Clique consensus protocol was analyzed using analytical method by accessing its source code and specifications [12]. The PoI consensus protocol was designed using mathematical modelling and implemented on the Go-Ethereum Clique codebase. Experimental validation was performed by deploying Clique and PoI blockchain networks in Docker with same network topology and configuration. To simulate IoT network logic, the IoTDataTracker smart contract was developed alongside testing scripts.

The Linux Traffic Control tool was used to simulate IoT network delays and potential device performance limitations [17]. The configuration used random intervals with up to 200 ms one-way delay and up to 20 ms jitter (packet delay variation). The jitter values for successive packets have 25% correlation with the previous packet's jitter value.

For measurement and monitoring, Docker statistics served as the primary tool for performance evaluation. Additionally, the Edgeshark project [18] was used for network monitoring through the WireShark application [19].

For statistical analysis, a log-normal distribution with normalized probability density was applied, as it is commonly recommended for network-related evaluations [20].

3. Results and Discussion

3.1. Analysis of the Clique work principle

Given n nodes with a signer role, Clique consensus [12] splits them to the following types:

– in-turn node: a main single node that creates a new block;
– no-turn nodes: all nodes excluding in-turn node and forbidden nodes. In case in-turn node fails to produce a block, no-turn nodes produce blocks with a random time interval to prevent race conditions;
– forbidden nodes: nodes that have reached a signing limit. They can receive and verify blocks but not produce them.

The typical operations sequence to generate and add a new block is as follows:

1. In-turn node selection index

$$k = \mathrm{mod}\left(h+1, n\right), \tag{1}$$

where h – current block number; n – total nodes count.

All other nodes, except the forbidden ones, are no-turn nodes.

2. Forbidden nodes signing limit is defined in the source code as

$$l = \left\lfloor \frac{n}{2}+1 \right\rfloor, \tag{2}$$

that means signer must wait l blocks after signing a current block to be able to sign again. The working system contains n_a active signer nodes and n_f forbidden nodes:

$$n_a = \left\lfloor \frac{n+1}{2} \right\rfloor, \tag{3}$$

$$n_f = \left\lfloor \frac{n-1}{2} \right\rfloor.$$

3. A successful scenario occurs when an in-turn node creates a block and broadcasts it within the broadcast period t_b, after which the receiving node validates and adds it during the validation period t_v.

In the Clique consensus mechanism, transactions are initially broadcast to all network nodes. When a block proposer creates a new block, this block, which contains the full transaction data, is propagated throughout the network to all the participating nodes, causing transaction data to be broadcast twice.

Furthermore, the node selection process does not consider nodes performance, which is important for heterogeneous IoT environments.

Fork probability. The random delay t_r for a no-turn node may be shorter than the combined duration of t_b and t_v. When this happens, a no-turn block may be added to the chain before the in-turn block arrives, thus creating a network fork.

After receiving the in-turn block, the node attempts to reorganize the blockchain as in-turn blocks have higher priority. The standard fork resolution mechanism gives preference to in-turn blocks over no-turn ones. In the situations where only multiple no-turn blocks exist, the node selects blocks based on two criteria: valid block number and temporal precedence (the first no-turn block received).

A more complex situation arises when certain nodes attempt to build the following blocks on the top of a forked block. This can trigger a cascade effect, where an incorrect block generates a series of no-turn blocks, thereby extending network instability period. It affects the network overall productivity, especially in IoT environments with various network latencies and nodes performance.

Clique has random delay to start no-turn nodes block production that can be expressed as

$$t_r = random\left(t_s, \left(\frac{n}{2}+1\right)\cdot\omega\right), \tag{4}$$

where ω – a random delay coefficient called wiggle time in Clique and hardcoded to 500 ms; t_s – an initial start value, hardcoded as 0 ms.

Thus, there is probability that for some no-turn node, random time will be 0 ms or near it.

Fork probability caused by a single node is

$$p_f = \frac{t_b + t_v}{t_r}, \tag{5}$$

where t_b is a broadcast period and t_v is a validation period.

Overall probability for all nodes looks like

$$p_a = 1 - \left(1 - p_f\right)^{n_a - 1}. \tag{6}$$

Therefore, fork probability grows rapidly with an increasing nodes count.

3.2. PoI improvements: theoretical optimization approach

Block size improvements. To optimize network performance, a block distribution process can be modified. Blocks would contain only cryptographic transaction hashes instead of complete transaction data. This approach would achieve two primary benefits: faster block propagation across the network and reduced bandwidth consumption during the distribution process. Since nodes already possess the original transaction data from the initial broadcast, they can reconstruct the complete block using the transaction hashes as references. This method eliminates redundant data transmission while maintaining the integrity and completeness of the blockchain.

Efficiency metrics and performance-based node selection. IoT network typically has heterogeneous node performance characteristics, with some nodes demonstrating high computational capabilities, whereas others operate with limited resources. To optimize overall network performance, a performance-based selection mechanism for signer nodes is suggested.

Under this approach, high-performance nodes receive priority selection as in-turn validators. This prioritization strategy offers several advantages:

– enhanced transaction throughput. High-performance nodes can process and validate transactions more efficiently than resource-

ISSN-L 2664-9969; E-ISSN 2706-5448

constrained nodes do, thereby increasing the network overall transaction processing capacity;

– improved network stability. By ensuring that capable nodes handle block production, the system reduces the probability of network forks. Performance-optimized nodes are more likely to generate and broadcast blocks within the required time constraints, maintaining network synchronization.

The efficiency score can be calculated as the combination of base score and penalties. The base score for single node can be calculated as follows

$$S_{base} = \frac{\Delta b}{\Delta h} \cdot K_b, \quad (7)$$

where Δb – blocks produced amount during the measuring period; Δh – total blocks produced during the measuring period; K_b – a base score maximum mark.

The following penalties can be applied:

1. Missed turn means that a current signer is not the same as expected in-turn signer

$$P_{missed} = \frac{M(\Delta b)}{\Delta h} \cdot K_m, \quad (8)$$

where $M(\Delta b)$ – missed turns from all the blocks produced during the measuring period; K_m – a maximum penalty mark.

2. Time-based penalty related to the block delays:

$$P_{timing} = \min\left(D_b \cdot K_t, K_{t\_max}\right),$$
$$D_b = \frac{\sum_b \left(\tau_a(b) - \tau_e(b)\right)}{\Delta b}, \quad (9)$$

where D_b – an average block delay; K_t – an average time penalty multiplier; K_{t_max} – a maximum time penalty mark; $\tau_a(b)$ – an actual block time; $\tau_e(b)$ – scheduled block time.

3. Fork creation penalty

$$P_{fork} = \min\left(\frac{F(\Delta b)}{\Delta b} \cdot K_f, K_{f\_max}\right), \quad (10)$$

where $F(\Delta b)$ – the number of blocks with a different hash detected at the same block number; K_f – a penalty multiplier; K_{t_max} – a maximum fork penalty mark.

Finally, the total efficiency score for a single node

$$S_{total} = S_{base} - (P_{missed} + P_{timing} + P_{fork}). \quad (11)$$

The sum of all penalties should be balanced with base efficiency. The proposal is to put base efficiency mark equal to sum of maximum penalties mark in total.

The final step involves ranking all network nodes in descending order based on their calculated efficiency scores. Each participating node independently generates its own efficiency-based ranking of all network participants and distributes this ranking information to other nodes in the network. To optimize network communication efficiency, the node rankings are transmitted as ordered lists of byte ordered nodes indexes rather than complete performance data.

To find the final order by combining m reports from signer nodes S, a mean position approach is used. For single node $S(k)$, a new position can be defined as:

$$\bar{P}(k) = \frac{1}{m_k} \sum_{i=1}^{m} I_i(k)\, pos_i(k),$$
$$m_k \geq 1,\ I_i(k) \in \{0,1\}, \quad (12)$$

where m_k – the number of reports that include $S(k)$; $I_i(k)$ – an indicator of $S(k)$ inclusion in the current efficiency report; $pos_i(k)$ – the corresponding rank.

Since nodes order has changed, it is important to define how in-turn and no-turn nodes will be selected. The main idea is to take top h efficient nodes from all nodes n, sorted by efficiency. This means signing limit from formula (2) will be smaller for in-turn nodes as $h < n$, they will sign more frequently compared to Clique. However, in this case, the network will be more stable and efficient.

For no-turn nodes, a standard limit from formula (2) is used based on all n nodes. This approach allows all nodes to participate in efficiency ranking and enables them to evaluate other nodes for fair assessment while using high-efficient nodes more often.

Important notes. Formula (4) demonstrates that the wait time for no-turn nodes to begin block production is random and starts from a value that is hardcoded to 0 in Clique. Making this value configurable in PoI would reduce the overall probability of forks by providing additional time for the in-turn node to broadcast its new block.

Efficiency data can be transmitted using the same mechanism used in Clique voting by adding this information to the block header during every L blocks. Additionally, to ensure that majority of nodes apply a new signer order in a deterministic manner, a submission window is introduced. This window represents a specific number of blocks during which efficiency metric data can be sent and nodes can reach agreement on which subsequent block will implement the new efficiency order.

Moreover, not all metrics should be collected during the entire network epoch. Fork metrics and timing metrics require renewal every L block to ensure network adaptability when network or device conditions change.

Consensus comparison and evaluation. By implementing the above changes, the new consensus protocol should provide benefits for IoT systems. A feature comparison is presented in Table 1.

Table 1

Consensus protocol comparison

Feature	Clique	PoI
Signer order	Ascending byte order (static)	Efficiency-based (dynamic)
Fork handling	Basic	Extra penalties mechanism
Adaptability	Low	High
Bandwidth optimization	None	Light blocks
Node benefit	None	Performance-based priority

Therefore, the following outcomes can be evaluated:

– improved IoT network efficiency;
– reduced network bandwidth;
– fair resource allocation.

3.3. Practical implementation blockchain-based IoT solution

The implementation is based on the Go-Ethereum Clique consensus mechanism and is developed in Go. The related source files were copied to a separate directory, where the "clique" namespace was renamed to "poi" to implement new modifications. These changes are configured through additional parameters in the genesis file configuration. When these features are disabled in the genesis file, PoI maintains full compatibility with Clique through a fallback mechanism.

Fig. 1 presents the high-level PoI structure. The primary modifications that affect efficiency metrics are implemented within the consensus layer. However, the lightweight block definition and its integration are primarily associated with the general Ethereum client layer.

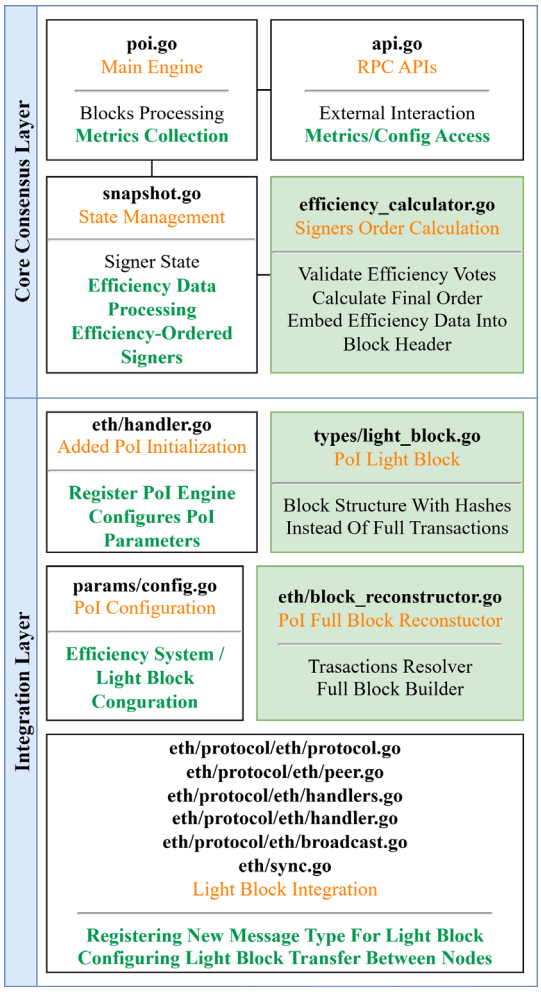


Fig. 1. Proof of Indicators (PoI) Structure

The general configuration includes next PoI-specific parameters to regulate PoI behavior:

– *BlockOptimization* enables or disables Light blocks propagation;
– *EnableMetrics* enables or disables efficiency calculation;
– *EfficiencyCheckPeriodBlocks* and *EfficiencyCheckSubmissionWindow* regulate efficiency calculation and vote submission window;
– *InTurnNodeLimit* limits top *N* efficient nodes to be in-turn signers;
– *RandomStartMultiplier* regulates minimum random delay for no-turn nodes.

By disabling *BlockOptimization* and *EnableMetrics*, PoI behavior will be equal to Clique.

Light block. Standard Ethereum blocks contain transactions that already exist on nodes. This results in data duplication. To optimize network usage, a new block type called the Light Block is introduced. The Light Block maintains the same structure as a standard block but replaces transactions with transaction hashes, as shown in Fig. 2.

While standard blocks include Uncles and Withdrawals fields, these fields are not utilized in Clique. The PoI-specific Light Block omits Withdrawals while maintaining Uncles as an empty field for compatibility purposes, consistent with the Clique implementation.

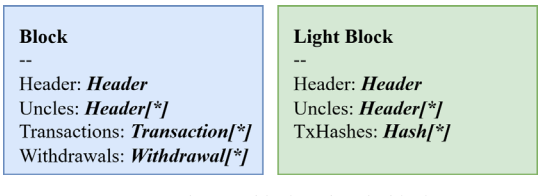


Fig. 2. Ethereum block and Light block

By extending the ETH68 protocol, the following PoI light block message types were introduced:

– *NewLightBlockMsg* – a message that contains the light block;
– *GetMissingTxsMsg* – a request message used to obtain a set of missing transactions;
– *MissingTxsMsg* – a response message that provides the missing transactions.

It should be noted that the Light Block is not constructed internally as a replacement for the Full Block. The Light Block is used for transmission to other nodes, while the full block is calculated on the node and subsequently transmitted as a light block.

Light block constructor:

```
func NewLightBlock(block *Block) *LightBlock {
 txHashes := make([]common.Hash, len(block.transactions))
 for i, tx := range block.transactions {
 txHashes[i] = tx.Hash()
 }
 return &LightBlock{
 Header: CopyHeader(block.header),
 TxHashes: txHashes
 }
}
```

Block propagation in eth/handler.go:

```
func (h *handler) BroadcastLightBlock(block *types.Block,
propagate bool) {
 poiEngine := getPOIEngine(h.chain.Engine())
 if poiEngine == nil || !poiEngine.IsBlockOptimizationEnabled() {
 h.BroadcastBlock(block, propagate) // Fallback to full blocks
 return
 }
 // Create light block from full block
 lightBlock := types.NewLightBlock(block)
 }
```

As illustrated in Fig. 1, Light Block integration affects multiple files responsible for block transmission under eth/protocols/eth:

– *protocol.go* defines ETH68 constants for Light Block functionality;
– *peer.go* and *broadcast.go* contain peer Light Block broadcast logic;
– *handlers.go* manages incoming Light Block propagations;
– *handler.go* extends the ETH68 message protocol to support Light Block messages.

The main objective is to prevent situations where slow nodes fail to receive all transactions and subsequently freeze. Therefore, nodes must have the capability to request missing transactions for block reconstruction. Other scenario occurs when a node receives no transactions or only a minimal number of transactions in its local cache. This situation results in data duplication when all transactions are re-transmitted. In such cases, it is more efficient for the node to request the complete block instead (Fig. 3).

As result, full block will be requested only in case when more than 50% of transactions are missing.

Efficiency metrics. Every signer node acts as an observer and records the following metrics for each signer node to provide input parameters for formulas (7)–(11):

– blocks produced value Δb is incremented when the observed node signs a block;
– missed turns $M(\Delta b)$ for the expected in-turn signer is incremented when the current signer node differs from the expected signer;
– block delays are recorded as the sum of differences between actual and expected time according to formula (9);
– fork detection verifies blocks with the same height (block number). If block hashes differ, this indicates that a fork was created.

ISSN-L 2664-9969; E-ISSN 2706-5448

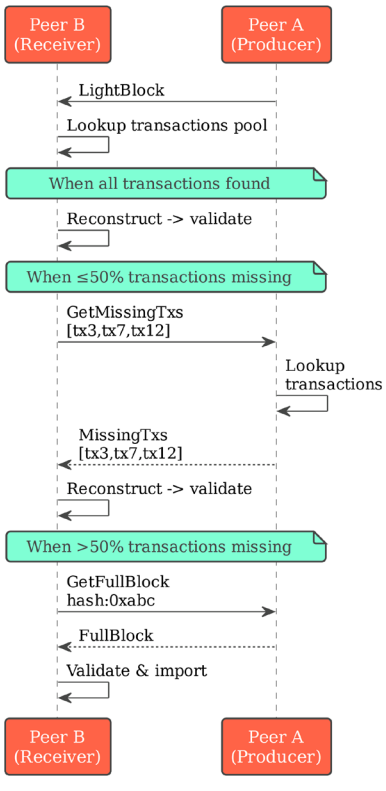


Fig. 3. Light block reconstruction timeline

Penalty maximum scores K_{t_max}, K_{f_max} were set to 30, while the missed turn penalty maximum score K_m was set to 40, as this covers the main indicator of an in-turn signer responsibility to propagate blocks on time. The base maximum score K_b was set to 100 to maintain balance in the total score. The delay multiplier K_t was set to 6, which ensures that the maximum delay penalty occurs when the average delay over the expected block time reaches 5 seconds or more. The fork multiplier K_f was set to 100, meaning that 30% or more produced blocks from a signer node will result in the maximum fork penalty of 30.

As forks and block delays can be specific to changing network conditions, these metrics are reset every efficiency period to build a more accurate efficiency report. In contrast, blocks produced and missed turns are reset every epoch. This approach ensures balance between the network temporary situation and overall node reputation in efficiency calculation.

Efficiency reports are integrated into block headers to avoid additional load from separate transactions. This approach ensures that only one efficiency report is submitted per block. To cover all possible nodes, a submission window is introduced. The submission window regulates the number of blocks during which an efficiency report can be submitted.

The efficiency reporting logic can be described as follows:

$$
\begin{aligned}
h_{start} &= \left\lfloor \frac{h}{E_p} \right\rfloor \cdot E_p, \\
h_{end} &= \left(\left\lfloor \frac{h}{E_p} \right\rfloor + 1 \right) \cdot E_p - 1, \\
h_{sub} &= h_{end} - E_s, \\
E_g &= E_p - E_s, \; E_p > E_s, \\
h_{eff} &= h_{end} + E_g,
\end{aligned} \tag{13}
$$

where h_{start} – a block number, indicating the efficiency period start; h_{end} – a block number, indicating the efficiency period end; h_{sub} – a block number, indicating the submission period start; h_{eff} – a block number, indicating a new efficiency order was set; E_p – an efficiency period (number of blocks) from genesis configuration; E_s – a submission window (number of blocks) from genesis configuration; h – a current block number; E_g – a grace period, i. e., the number of blocks to wait before applying a new efficiency order.

The efficiency metrics reporting process is presented in Fig. 4. A silence period occurs before the submission window begins. This time is utilized for period metrics collection. The submission window size and the efficiency period size determine the grace period, which represents a special delay after the submission window to ensure all nodes apply the new efficiency order at the same block. The submission period should not be less than the number of signer nodes in the network; otherwise, not all nodes will be able to submit their efficiency reports.

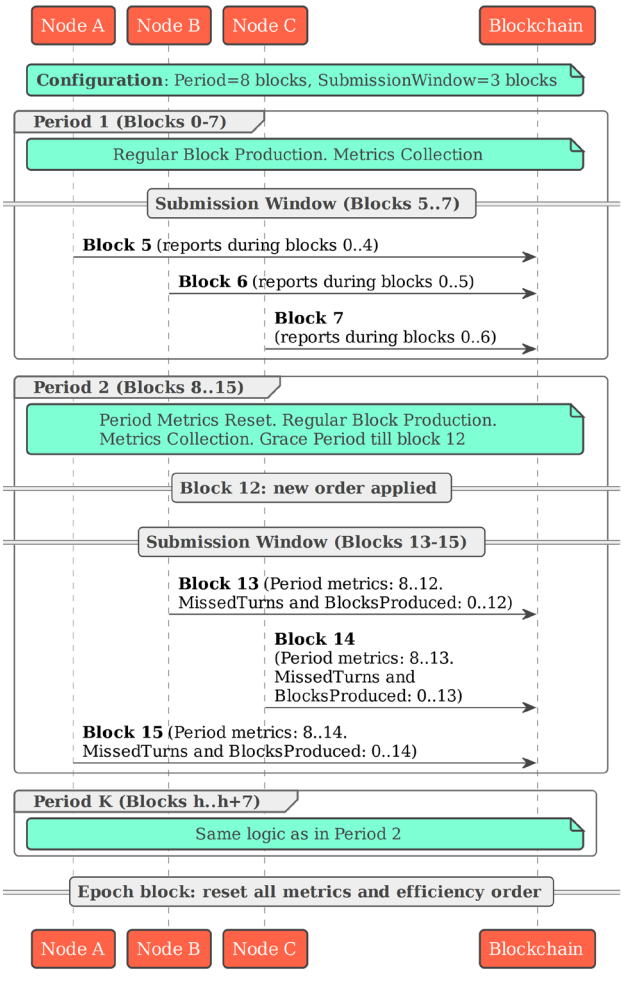


Fig. 4. The efficiency metrics reporting process

By specifying efficiency order votes in block headers, efficiency orders from all nodes are stored in the blockchain. There is no final block containing the definitive order. The final order corresponds to the submission window blocks and is calculated by each node during the grace period. This approach eliminates speculation and provides deterministic node order generation. The final efficiency order is created by combining all reports and determining the mean position according to formula (12).

To enable efficiency order transmission in the block headers, the ExtraData field was expanded. In Clique, this field is used for specifying epoch block signers and remains unused during normal block production. To preserve that mechanism, additional space for the efficiency order was defined, as shown in Fig. 5.

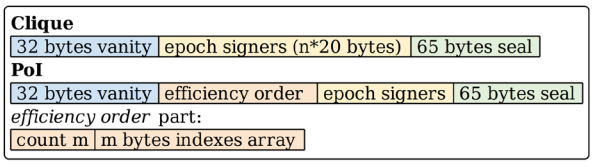


Fig. 5. ExtraData block header format

The distinction between the epoch signers format and the efficiency order format enables clear identification of each data type within the block header. The epoch signers format contains 20 bytes for each signer, while the efficiency order format begins with a 1-byte length specifier that indicates array length *m*, followed by *m* bytes of indexes. This byte must always be present to ensure a proper distinction mechanism. When *m* equals 0, it indicates the absence of efficiency order data. When *m* is greater than 0, it signifies that the next *m* bytes contain efficiency data indexes.

It is important to note that the current efficiency order encoding mechanism covers a maximum of 255 nodes, which should be sufficient for separate, purpose-specific IoT networks where each node serves as a hub for various smaller IoT devices.

For every epoch transition, the epochs block acts as a checkpoint from which capable clients should be able to synchronize without requiring any previous state. Therefore, the system writes the current efficiency order in every epoch block. In cases where an epoch block falls within the submission window, it is not interpreted as a submission report.

The efficiency order will not impact network performance if the Clique in-turn node selection mechanism from formula (1) is still used. The proposal to select top performers from the efficiency ordered list is implemented through the *InTurnNodeLimit* property in the genesis file. When this property is greater than 0, the limit applies. This approach creates two signing limits: for no-turn nodes, the general limit from formula (2) applies, while for in-turn nodes, *InTurnNodeLimit* replaces *n* in the calculation.

To preserve network stability, the efficiency order calculation considers only the votes from the in-turn blocks. This approach is necessary because no-turn blocks originating from different nodes can be applied concurrently. Such concurrency may lead to inconsistent final orderings, thus causing a network split or fork.

To provide additional time for in-turn nodes to broadcast blocks in networks with high latencies, *RandomStartMultiplier* genesis property is introduced. According to formula (4), t_s can be configured as follows

$$t_s = \omega \cdot RandomStartMultiplier. \quad (14)$$

Therefore, the extra start period for no-turn blocks can additionally reduce fork probability while maintaining it at 0 for networks with standard latencies.

3.4. Experimental unit and test scenarios

For experimental validation, two Docker networks were established to evaluate Clique consensus and PoI respectively. Each network comprises 15 nodes, with the first node serving as a bootnode.

The connection topology implements a partial mesh configuration designed to simulate real-world conditions where complete node interconnection is absent. In this topology, each node connects to one previous node and two subsequent nodes. Additionally, cross-connections are established at the 1/3 position (node 6) and 2/3 position (node 11). As a result, each node maintains connections to 5 peers.

Both networks were configured with a 3-second period for block production. For PoI, several parameters were modified during testing to enable comprehensive analysis. The evaluation proceeded in stages: the efficiency metrics system was tested separately, followed by the final testing with all the features enabled.

The efficiency period was set to 100 blocks, while the submission window was configured to equal the total node count of 15. The in-turn nodes limit was established at 1/3 of all nodes, resulting in 5 in-turn nodes.

A specialized smart contract, IoTDataTracker, was designed for and deployed to both networks to support IoT device functionality. The contract implements several key features tailored for IoT applications:

– the change-only storage mechanism stores sensor data exclusively when values undergo actual changes, optimizing storage efficiency;

– multi-sensor support enables individual devices to manage multiple sensor types, such as temperature and humidity sensors;

– batch update functionality allows multiple sensors to be updated within a single transaction, reducing transaction overhead;

– the contract maintains history tracking capabilities, preserving change records with associated timestamps for temporal analysis. Additionally, device management features enable the registration, activation, and deactivation of IoT devices within the network.

The *IoTDataTracker* smart contract is used for evaluating network performance in both Clique and PoI consensus protocols. The test suite includes two distinct test categories: performance tests and fork tests.

Performance tests include two scenarios: load tests that process 10 IoT transactions per device per second and smart home simulation tests that operate at 2 transactions per device per second.

Blockchain-specific metrics include block times, block propagation times, and transactions per second (TPS):

$$TPS = \frac{S}{t_{\max} - t_{\min}},$$
$$T(b_n) = \tau(b_n) - \tau(b_{n-1}),$$
$$T_{prop} = \max\left(T_i(b_n) - T_0(b_n)\right), \quad (15)$$

where S – successful transactions sent in period between $t_{\max}$ and $t_{\min}$; $T(b_n)$ – block time; T_{prop} – propagation time; $\tau(b_n)$ – timestamp of block b_n; $T_i(b_n)$ – time node i received block b_n.

System resource metrics are gathered from Docker containers and comprise CPU utilization, memory consumption, and network usage.

Fork tests belong to the second evaluation category, where the system performs assessment at maximum achievable TPS for 300 seconds to measure network resilience under high transaction loads. Fork tests generate multiple competing transactions with identical nonces but different content. This approach creates nonce conflicts where multiple valid transactions compete for the same nonce slot, forcing consensus mechanisms to select winners and potentially causing temporary forks during resolution.

Additionally, fork tests divide the blockchain network into two groups of nodes and send conflicting transactions to each group. This method creates realistic network partition scenarios where the isolated node groups apply different blockchain states, simulating network split conditions.

3.5. Tests results evaluation

Network packets were captured using Wireshark, while additional data were collected from Docker and the Geth API through scripting. The measured average bandwidth for PoI was 1.27 Mbps, whereas Clique reached 1.4 Mbps, which corresponds to approximately a 9% difference. For statistical analysis, a log-normal distribution with normalized probability density was applied. As illustrated in Fig. 6, PoI results have smaller transfer sizes than those in Clique. Since the raw dataset was collected at 2-second intervals, the reduced occurrence of network forks may explain this result. Fork evaluation further indicates that enabling PoI efficiency order reduces forks by about 25%.

The PoI implementation demonstrates improved performance. The average block propagation time is reduced by 8.5%, and block times are shortened by 3.1%. As illustrated in Fig. 7 and 8, a log-normal

ISSN-L 2664-9969; E-ISSN 2706-5448

distribution analysis of block propagation and block times shows the performance advantage achieved through the PoI efficiency order implementation.

The implementation of light blocks resulted in a 20.5% decrease in overall network traffic (Fig. 9, 10). Furthermore, enabling delayed no-turn blocks by setting the RandomStartMultiplier property to 1 in the genesis file achieved 80% reduction in total forks when compared to the Clique implementation.

With all optimizations enabled, including efficiency order, light blocks, and delayed no-turn block generation, the transaction throughput (TPS) per device was improved, ranging from 5% to 10% on average, depending on the test scenario (Fig. 11).

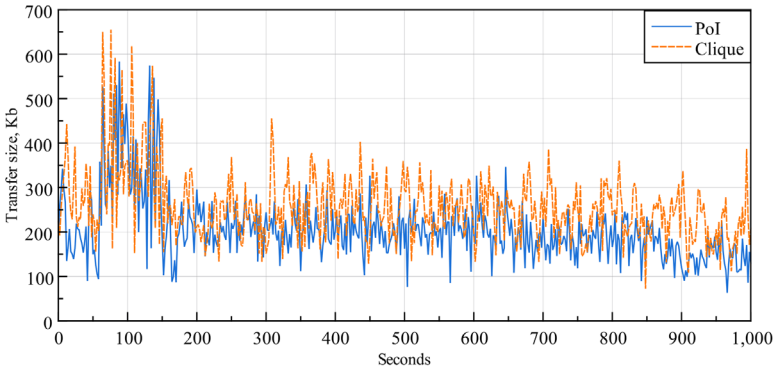


Fig. 9. WireShark network traffic capture with all the optimizations enabled

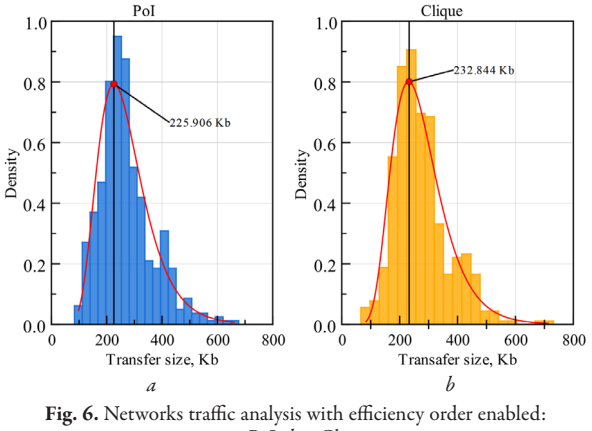


Fig. 6. Networks traffic analysis with efficiency order enabled: *a* – PoI; *b* – Clique

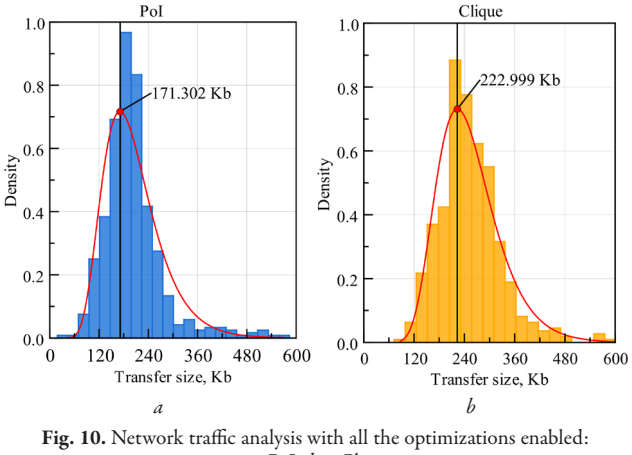


Fig. 10. Network traffic analysis with all the optimizations enabled: *a* – PoI; *b* – Clique

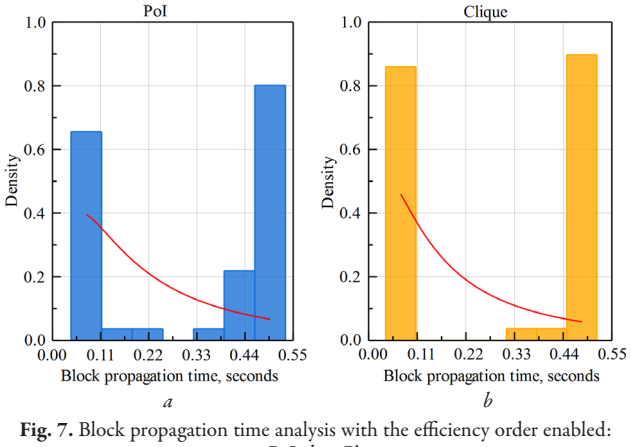
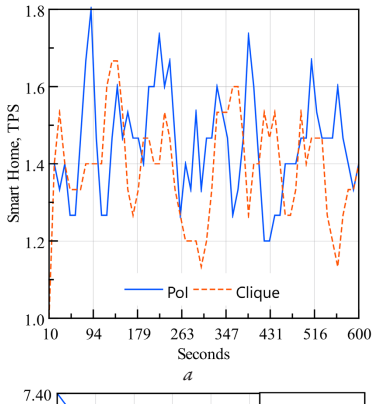


Fig. 7. Block propagation time analysis with the efficiency order enabled: *a* – PoI; *b* – Clique



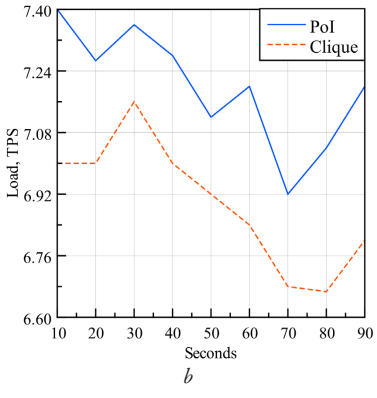


Fig. 11. TPS capture per device with 10 second intervals with all the optimizations enabled: *a* – PoI; *b* – Clique

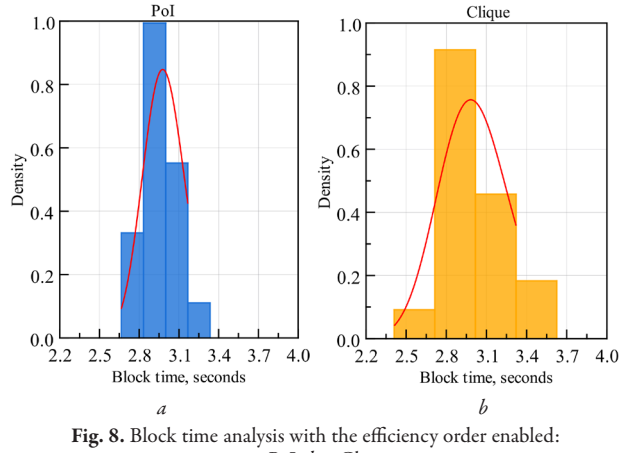


Fig. 8. Block time analysis with the efficiency order enabled: *a* – PoI; *b* – Clique

It is worth noting that Fig. 11 does not represent the full potential for TPS improvement as the evaluation focused on a load suitable for a purpose-specific IoT network.

3.6. Discussion

Interpretation of the results. The observed improvements can be explained by new approaches introduced in PoI. The traffic reduction is explained by the difference in block payload size. Since nodes already hold the original transactions from the initial broadcast, redundant retransmission inside blocks was eliminated by using light blocks containing transaction headers only.

The fork reduction is achieved by combining two mechanisms. First, dynamic node selection prioritizes nodes with a reliable network connection and good computational capabilities as in-turn validators, reducing block production failures and late broadcast probability. Including fork creation penalties in the efficiency scoring additionally lowers the probability of an underperforming node being selected as an in-turn node. Second, the delayed no-turn block generation provides additional time for the in-turn block to propagate.

Other performance characteristics, including TPS, block propagation times and block times, were improved as a result of the PoI mechanisms described above.

The slight increase in resource consumption is caused by new operations that do not exist in Clique: metrics collection, composite score calculation, block reconstruction for light blocks, and in some cases, network requests for missing transactions.

Distinctive features and practical relevance. Existing IoT network efficiency approaches mostly focus on transport-level optimizations, while blockchain-oriented research offers alternative consensus architectures. PoI takes a different approach by introducing optimizations within the consensus layer of an existing PoA protocol that includes two independent optimization strategies: performance-based node selection and block size reduction.

The proposed solution is applicable to industrial decentralized IoT networks where sensor hubs with different hardware capabilities form a local blockchain network for security and data integrity. It can also be used for smart city infrastructure where geographically distributed gateways operate under different network conditions. Besides its traditional usage as part of an IoT decentralized platform, it is possible to create an advantage in military operations, including drone swarms that need to build a local network where network conditions are not stable and data traffic is limited.

Research limitations. All experiments were conducted in a Docker-based simulation on a single machine with a partial mesh topology. Artificial network latencies were configured to simulate physical IoT network conditions. However, actual hardware constraints such as limited memory, CPU throttling, and real wireless channel characteristics were not replicated. Different network topologies or significantly different latencies may not produce the same results as those obtained. PoI is designed for purpose-specific IoT networks and is not applicable for public blockchain deployments.

The implementation is built on the Go-Ethereum Clique codebase, inheriting its architectural constraints. The PoI efficiency order system uses a single-byte index, limiting the maximum network size to 255 nodes. Penalty coefficients were set based on logical assumptions and preliminary testing without formal optimization.

Prospects for further research. To address the environment limitation, PoI should be deployed on physical IoT hardware such as Raspberry Pi devices, and comprehensive testing should be conducted under real conditions. Additionally, networks of 50–255 nodes should be tested to evaluate how the efficiency reporting system and light block propagation behave during submission window growth.

Formal optimization could be used to calibrate penalty coefficients and improve the node selection process.

A possible limitation with the delayed no-turn block generation mechanism should be stress-tested for scenarios where multiple in-turn nodes fail to deliver a block, causing additional delays.

An evaluation of PoI interaction with an edge computing architecture would broaden the practical applicability of the proposed solution.

Martial law impact. There is no influence of martial law in Ukraine on current research.

4. Conclusions

1. The Proof of Authority consensus protocol can be used for organizing IoT devices into separate, purpose-specific blockchain networks, as PoA nodes can run on resource-constrained devices like the Raspberry Pi. The Go-Ethereum project's PoA implementation, called Clique, was used as a base for improvements. Clique's working principles were analyzed, revealing gaps regarding fork occurrence, data duplication, and a signer selection process that does not consider node performance.

2. A new optimization approach was designed to improve Clique's performance for IoT networks. A dynamic node selection process was developed based on node efficiency. This process uses metrics such as the number of successful and missed blocks, the number of forks created by the current node, and time delays to rank signers. This enhancement should improve transaction throughput and overall network stability under inconsistent network conditions and across various node computing capabilities. Another improvement substitutes complete transaction data in blocks with cryptographic transaction hashes to reduce overall network traffic. These optimizations are expected to achieve fair resource allocation, reduce network bandwidth consumption and improve IoT network efficiency.

3. To experimentally verify the proposed theoretical approach, the Proof of Indicators consensus protocol was implemented in the Go programming language, based on Go-Ethereum Clique codebase. The efficiency metrics system collects node indicators and combines them into a composite score according to the theoretical approach. The resulting efficiency order is represented as an array of node indexes and embedded into block headers to avoid additional load from separate transactions. A submission window was introduced to provide sufficient time for all nodes to submit reports. Each node calculates the final order in deterministic manner by finding the mean position of every node across all received reports. Additionally, light blocks replace full transaction data with transaction hashes during propagation to reduce network traffic. If a node has not received all required transactions and cannot build a full block, a fallback mechanism is implemented to retrieve missing transactions or request the full block when it cannot reconstruct the block from its local transaction cache. The scientific originality lies in combining dynamic node selection with light block propagation within the blockchain consensus layer to address Clique's previously found limitations. The PoI system is backward compatible with Clique through genesis configuration flags. It is designed to run purpose-specific IoT blockchain networks with up to 255 nodes. The expected results should confirm the theoretical expectations.

4. For experimental validation, two Docker networks of 15 nodes each were configured with a partial mesh topology. To evaluate consensus stability under variable network conditions, the test environment included random latencies created by using the Linux Traffic Control tool. The Edgeshark tool was set up to capture Docker traffic for analysis. A specialized IoTDataTracker smart contract was deployed to both networks for IoT workload simulation. The expectation of this environment is to perform reproducible comparison between the two blockchain networks under identical conditions.

5. Comparative testing confirmed the theoretical expectations. With all optimizations enabled, overall network traffic was reduced by 20.5% and the number of forks decreased by 80% compared to Clique. The dynamic node selection mechanism alone reduced block propagation time by 8.5%, block times by 3.1% and forks by 25% while the

ISSN-L 2664-9969; E-ISSN 2706-5448

remaining fork reduction was achieved in combination with delayed no-turn blocks and light blocks. While it improves network stability and efficiency, a modest and acceptable increase in CPU (6.5%) and memory (5.4%) consumption occurs due to an extra logic layer with metrics and block reconstruction. The demonstrated gains present PoI as a more stable and efficient solution for real-world IoT blockchain deployments with further possibilities for scoring algorithms improvements and physical IoT hardware validation.

Conflict of interest

The authors declare that they have no conflict of interest in relation to this research, whether financial, personal, authorship or otherwise, that could affect the research and its results presented in this paper.

Financing

This research was conducted without financial support.

Data availability

Data will be made available upon reasonable request.

Use of artificial intelligence

In addition to the standard Scopus search, Perplexity AI's research mode was used in the "Introduction" section to search for sources published in the last 5 years. The authors verified the results provided by reviewing the original sources and their content and confirm that it did not influence research conclusions.

Grammarly was used for all sections except the bibliography. It was used for grammar, punctuation and spell checks. During the use of this tool the authors reviewed recommendations provided by the tool and the decision to apply them was made by the authors. The use of Grammarly improved grammar but had no impact on results.

Authors' contributions

Leonid Chepel: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data Curation, Writing – original draft, Writing – review and editing, Visualization; ***Yuriy Boyko***: Conceptualization, Methodology, Supervision, Writing – review and editing.

References

1. Minerva, R., Biru, A., Rotondi, D. (2015). *Towards a definition of the Internet of Things (IoT).* IEEE Internet Initiative.
2. Kuchuk, H., Malokhvii, E. (2024). Integration of IoT with cloud, fog, and edge computing: a review. *Advanced Information Systems, 8 (2),* 65–78. https://doi.org/10.20998/2522-9052.2024.2.08
3. Hossain, M., Kayas, G., Hasan, R., Skjellum, A., Noor, S., Islam, S. M. R. (2024). A Holistic Analysis of Internet of Things (IoT) Security: Principles, Practices, and New Perspectives. *Future Internet, 16 (2),* 40. https://doi.org/10.3390/fi16020040
4. Cao, K., Liu, Y., Meng, G., Sun, Q. (2020). An Overview on Edge Computing Research. *IEEE Access, 8,* 85714–85728. https://doi.org/10.1109/access.2020.2991734
5. Miller, R., Whelan, H., Chrubasik, M., Whittaker, D., Duncan, P., Gregório, J. (2024). A Framework for Current and New Data Quality Dimensions: An Overview. *Data, 9 (12),* 151. https://doi.org/10.3390/data9120151
6. Ullah, I., Havinga, P. J. M. (2023). Governance of a Blockchain-Enabled IoT Ecosystem: A Variable Geometry Approach. *Sensors, 23 (22),* 9031. https://doi.org/10.3390/s23229031
7. Susnjara, S., Smalley, I. What is blockchain? *IBM.* Available at: https://www.ibm.com/topics/blockchain
8. Upadhyay, V., Vaish, A., Kokila, J. (2024). The need for Lightweight Consensus algorithms in IoT environment: A review. *Proceedings of the 2024 Sixteenth International Conference on Contemporary Computing.* ACM, 366–376. https://doi.org/10.1145/3675888.3676072
9. Lin, S.-Y., Zhang, L., Li, J., Ji, L., Sun, Y. (2022). A survey of application research based on blockchain smart contract. *Wireless Networks, 28 (2),* 635–690. https://doi.org/10.1007/s11276-021-02874-x
10. Ahmad, A., Alabduljabbar, A., Saad, M., Nyang, D., Kim, J., Mohaisen, D. (2021). Empirically comparing the performance of blockchain's consensus algorithms. *IET Blockchain, 1 (1),* 56–64. https://doi.org/10.1049/blc2.12007
11. Islam, Md. M., Merlec, M. M., In, H. P. (2022). A Comparative Analysis of Proof-of-Authority Consensus Algorithms: Aura vs Clique. *2022 IEEE International Conference on Services Computing (SCC).* IEEE, 327–332. https://doi.org/10.1109/scc55611.2022.00054
12. Szilágyi, P. (2017). EIP-225: Clique proof-of-authority consensus protocol. *Ethereum Improvement Proposals.* Available at: https://eips.ethereum.org/EIPS/eip-225
13. Kopanitsa. Web3 Arduino: An Arduino (or ESP32) library to use web3 on Ethereum platform. *GitHub.* Available at: https://github.com/kopanitsa/web3-arduino
14. Almudayni, Z., Soh, B., Samra, H., Li, A. (2025). Energy Inefficiency in IoT Networks: Causes, Impact, and a Strategic Framework for Sustainable Optimisation. *Electronics, 14 (1),* 159. https://doi.org/10.3390/electronics14010159
15. Verma, L. P., Kumar, G., Khalaf, O. I., Wong, W.-K., Hamad, A. A., Rawat, S. S. (2024). Adaptive congestion control in IoT networks: Leveraging one-way delay for enhanced performance. *Heliyon, 10 (22),* e40266. https://doi.org/10.1016/j.heliyon.2024.e40266
16. Gordieiev, O., Rainer, A., Kharchenko, V., Pishchukhina, O., Gordieieva, D. (2024). A Unified Approach to the Development of Technology-Based Software Quality Models on the Example of Blockchain Systems. *IEEE Access, 12,* 118875–118889. https://doi.org/10.1109/access.2024.3448271
17. Hemminger, S. (2011). Tc-netem – Linux manual page. *Linux Foundation.* Available at: https://man7.org/linux/man-pages/man8/tc-netem.8.html
18. Albrecht, H. (2023). Introduction. *Edgeshark. Siemens.* Available at: https://edgeshark.siemens.io
19. Combs, G. *Wireshark: Network protocol analyzer.* Wireshark Foundation.
20. Alasmar, M., Clegg, R., Zakhleniuk, N., Parisis, G. (2021). Internet Traffic Volumes are Not Gaussian – They are Log-Normal: An 18-Year Longitudinal Study With Implications for Modelling and Prediction. *IEEE/ACM Transactions on Networking, 29 (3),* 1266–1279. https://doi.org/10.1109/tnet.2021.3059542

✉ ***Leonid Chepel,*** *PhD Student, Department of Computer Engineering, Taras Shevchenko National University of Kyiv, Kyiv, Ukraine, e-mail: leonid.chepel@knu.ua, ORCID: https://orcid.org/0009-0008-4209-8102*

Yuriy Boyko, *PhD, Associate Professor, Department of Computer Engineering, Taras Shevchenko National University of Kyiv, Kyiv, Ukraine, ORCID: https://orcid.org/0000-0003-1417-7424*

✉ *Corresponding author*