Київський національний університет імені Тараса Шевченка
Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка
Міністерство освіти і науки України

Кваліфікаційна наукова
праця на правах рукопису

ЧЕПЕЛЬ ЛЕОНІД ВАЛЕРІЙОВИЧ

УДК 004.72

ДИСЕРТАЦІЯ

МЕТОДИ І ЗАСОБИ ПОБУДОВИ МЕРЕЖ ІНТЕРНЕТУ РЕЧЕЙ НА ОСНОВІ ГІБРИДНОЇ БЛОКЧЕЙН-АРХІТЕКТУРИ З АДАПТИВНИМ КОНСЕНСУСОМ ТА МУЛЬТИФАКТОРНИМ УПРАВЛІННЯМ ВУЗЛАМИ

Спеціальність 123 Комп’ютерна інженерія
Галузь знань 12 Інформаційні технології

Подається на здобуття наукового ступеня доктора філософії

Дисертація містить матеріали власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело
_____ Чепель Леонід Валерійович

Науковий керівник:
кандидат фізико-математичних наук, доцент
Бойко Юрій Володимирович

Київ - 2026

АНОТАЦІЯ

Чепель Л. В. Методи і засоби побудови мереж Інтернету речей на основі гібридної блокчейн-архітектури з адаптивним консенсусом та мультифакторним управлінням вузлами. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 123 «Комп'ютерна інженерія». – Київський національний університет імені Тараса Шевченка, Київ, 2026.

Робота присвячена дослідженню та розробці нових методів і засобів побудови блокчейн-орієнтованих мереж Інтернету речей (IoT) з адаптивним консенсусом та мультифакторним управлінням обчислювальними вузлами. Особливу увагу приділено підвищенню ефективності функціонування таких мереж у значенні категорії Performance Efficiency стандарту ISO/IEC 25010.

У роботі обгрунтовано потребу в гібридному підході до побудови блокчейн-IoT мереж. Проведений порівняльний аналіз блокчейн-платформ засвідчив, що жодна окрема платформа не задовольняє одночасно всі наступні вимоги: низька затримка, підтримка апаратно обмежених пристроїв, масштабованість та безпека. На основі аналізу наукових джерел сформульовано дослідницькі прогалини: обмеженість існуючих методів координації вузлів та відсутність інтегрованих рішень, що поєднують рівень консенсусу з рівнем розподілу обчислювальних задач.

У дисертації проведено дослідження в трьох взаємопов'язаних напрямах.

1. **Гібридна багаторівнева блокчейн-архітектура.** Удосконалено модель тришарової кластерної організації IoT-мережі, що включає рівень сенсорів із симетричним шифруванням для апаратно обмежених пристроїв, рівень локальної блокчейн-мережі з підтримкою смарт-контрактів, рівень зовнішньої блокчейн-мережі з агрегацією транзакцій. Розроблено адаптацію архітектури для спеціалізованих сценаріїв.

2. **Метод адаптивного консенсусу Proof of Indicators (PoI).** Запропоновано та реалізовано консенсус протокол, що поєднує динамічний вибір валідаторів на основі комплексної метрики ефективності, техніку створення оптимізованих блокчейн блоків на основі хешів транзакцій та підхід для зменшення ймовірності розгалужень ланцюга блоків.
3. **Модель мультифакторного вибору периферійних вузлів.** Розроблено мультифакторну скорингову функцію, що поєднує шість факторів для комплексної оцінки обчислювальних вузлів з лінійною обчислювальною складністю.

Запропоновані рішення оцінено експериментально у контрольованому тестовому середовищі з моделюванням IoT-умов. Результати підтвердили правильність теоретичного підходу, що демонструє практичну придатність розроблених методів.

Наукова новизна дисертаційної роботи полягає в обгрунтуванні, розробці та експериментальній перевірці нових методів побудови блокчейн-орієнтованих мереж IoT.

Удосконалено модель гібридної багаторівневої блокчейн-архітектури для мереж IoT, яка, на відміну від відомих рішень, використовує тришарову кластерну організацію з обгрунтованим вибором типу блокчейну для кожного рівня на основі порівняльного аналізу за критеріями пропускної здатності, безпеки та придатності для IoT. Це дозволяє побудувати масштабовану, децентралізовану та безпечну мережу з підтримкою гетерогенних пристроїв та можливістю автономної роботи кластерів.

Уперше запропоновано метод адаптивного консенсусу Proof of Indicators, який, на відміну від консенсусу довірених валідаторів (Proof of Authority, PoA) в реалізації Clique, впроваджує три взаємодоповнюючі удосконалення: динамічний вибір валідаторів на основі комплексної метрики продуктивності, техніку поширення оптимізованих блоків та конфігуровну затримку старту поширення блоку

позачерговими вузлами. Як результат – зменшення мережевого трафіку на 20,5%, скорочення кількості розгалужень ланцюга на 80%, підвищення пропускної здатності (TPS) до 10% та зменшення часу поширення блоків на 8,5% у порівнянні з Clique.

Удосконалено модель мультифакторного вибору периферійних вузлів, яка, на відміну від існуючих підходів, несумісних з виконанням в блокчейні, використовує мультиплікативну скорингову функцію з розділенням обчислень поза блокчейном та координації в блокчейні через смарт-контракти. Це дає змогу запобігти концентрації задач на перевантажених вузлах та досягти автоматичного балансування навантаження з лінійною обчислювальною складністю.

Практична цінність роботи полягає в створенні програмних засобів, придатних для безпосереднього впровадження: реалізації консенсусу PoI мовою Go на базі проекту Go-Ethereum, платформи мультифакторного управління вузлами на базі смарт-контрактів та Python-агентів, результатів порівняльного аналізу блокчейн-платформ та моделей для спеціалізованих сценаріїв. Запропоновані рішення функціонують на пристроях класу Raspberry Pi та орієнтовані на широке практичне використання.

Перспективи подальших досліджень передбачають впровадження моделей прогнозування оцінки стану вузлів, розроблення адаптивних ваг факторів скорингової функції, інтеграцію з механізмом шардингу для масштабування на множину кластерів, використання доказів без розкриття інформації (zero-knowledge proof) для верифікації позаланцюгових обчислень та адаптацію комбінованої системи для сценаріїв федеративного навчання в IoT-мережах.

Ключові слова: Інтернет речей, блокчейн, децентралізовані мережі, комп’ютерні системи, гібридна архітектура, адаптивний консенсус, мультифакторний вибір вузлів, периферійні обчислення, вбудовані системи, смарт-контракти.

SUMMARY

Chepel L. V. Methods and tools for building Internet of Things networks based on hybrid blockchain architecture with adaptive consensus and multifactor node management. – A qualification scientific work on the rights of a manuscript.

Dissertation for the degree of Doctor of Philosophy in specialty 123 "Computer Engineering". – Taras Shevchenko National University of Kyiv, Kyiv, 2026.

The work is devoted to the research and development of new methods and tools for building blockchain-oriented Internet of Things networks with adaptive consensus and multi-factor management of computing nodes. Attention is paid to improving the performance efficiency of IoT networks in the sense of the Performance Efficiency category according to the international standard ISO/IEC 25010.

The dissertation substantiates the need for a hybrid approach to building blockchain-based IoT networks. A comparative analysis of blockchain platforms demonstrated that no single platform simultaneously satisfies all the requirements of IoT environments – low latency, support for hardware-constrained devices, scalability, and security. Based on the analysis of scientific sources, research gaps were formulated: the limitations of existing node coordination methods and the absence of integrated solutions that combine the consensus layer with the layer of computational task distribution.

The dissertation conducts research in three interrelated directions.

1. **Hybrid multi-layer blockchain architecture**. A model of three-layer cluster organization of an IoT network has been improved, comprising the sensor layer with symmetric encryption for hardware-constrained devices, the local blockchain network layer with smart contract support, and the external blockchain network layer with transaction aggregation. The architecture has been adapted for specialized scenarios.

2. **Adaptive consensus method Proof of Indicators (PoI).** A consensus protocol has been proposed and implemented that combines dynamic validator selection based on a

complex performance metric, a technique for creating optimized blockchain blocks based on transaction hashes, and an approach for reducing the probability of chain forks.

3. **Multi-factor model for edge node selection**. A multi-factor scoring function based on multiplicative composition has been developed, which combines six factors for a comprehensive assessment of computing nodes with linear computational complexity.

The proposed solutions were evaluated experimentally in a controlled testbed environment that simulates IoT conditions. The results successfully confirmed the validity of the theoretical approach, demonstrating the practical applicability of the developed methods.

The scientific novelty of the dissertation lies in the substantiation, development, and experimental verification of new methods for building blockchain-oriented IoT networks.

The model of a hybrid multi-layer blockchain architecture for IoT networks has been **improved**. In contrast to known solutions, it employs a three-layer cluster organization with a justified choice of blockchain type for each layer based on a comparative analysis according to the criteria of throughput, security, and IoT suitability. This makes it possible to build a scalable, decentralized, and secure network with support for heterogeneous devices and the capability for autonomous cluster operation.

An adaptive consensus method Proof of Indicators has been **proposed for the first time**. In contrast to the PoA Clique consensus with a static signer order, it introduces three complementary improvements: dynamic validator selection based on a complex performance metric, a light-block propagation technique, and a configurable propagation start delay for no-turn nodes. As a result, the method achieves a 20.5% reduction in network traffic, an 80% reduction in chain forks, up to 10% increase in throughput, and an 8.5% reduction in block propagation time compared with baseline Clique.

The model of multi-factor edge node selection has been **improved**. In contrast to existing approaches that are incompatible with on-chain execution, it employs a multiplicative scoring function with separation of off-chain computations and on-chain

coordination through smart contracts and integration with PoI. This makes it possible to prevent the concentration of tasks on overloaded nodes and to achieve automatic load balancing with linear computational complexity.

The practical value of the work consists in the creation of software tools suitable for direct deployment: a Go-language implementation of the PoI protocol on top of Go-Ethereum, a multi-factor node management platform based on smart contracts and Python agents, the results of comparative analysis of blockchain platforms, and models for specialized scenarios. The proposed solutions operate on Raspberry Pi-class devices and are oriented toward broad practical use.

Further research directions include the introduction of predictive node-state assessment models, the development of adaptive weights for the scoring function factors, integration with a sharding mechanism for scaling to multiple clusters, the use of zero-knowledge proofs for verification of off-chain computations, and the adaptation of the combined system for federated learning scenarios in IoT networks.

Keywords: Internet of Things, blockchain, decentralized networks, computer systems, hybrid architecture, adaptive consensus, multi-factor node selection, edge computing, embedded systems, smart contracts.

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА

Статті у наукових фахових виданнях України:

1. Л. В. Чепель, Ю. В. Бойко, “Підхід до безпеки та організації мереж IoT з використанням блокчейн технології,” Вісник Вінницького Політехнічного Інституту, no. 4, pp. 129–138, Aug. 2024, doi: 10.31649/1997-9266-2024-175-4-129-138.
2. Л. В. Чепель, Ю. В. Бойко, “Побудова мереж IOT військового призначення з використанням блокчейн технології,” Inf. Technol. Comput. Sci. Softw. Eng. Cyber Secur., pp. 186–194, Jun. 2025, doi: 10.32782/IT/2025-2-19.
3. L. Chepel and Y. Boyko, “Proof-of-Indicators: development and validation of an adaptive consensus mechanism for Internet of Things blockchain networks,” Technol. Audit Prod. Reserv., vol. 2, no. 2(88), pp. 15–24, Apr. 2026, doi: 10.15587/2706-5448.2026.356851.
4. L. Chepel and Y. Boyko, “multi-factor hybrid approach for blockchain-enabled IoT edge computing,” Comput. Syst. Inf. Technol., no. 1, pp. 78–87, Mar. 2026, doi: 10.31891/csit-2026-1-8.

Праці, які засвідчують апробацію матеріалів дисертації:

1. Karlash, Bohdan, Leonid Chepel, and Yuriy Boyko. “Comparative Analysis of Smartphone Ad-Hoc Network Based Software.” In Proceedings of the XXIII International Young Scientists Conference on Applied Physics (ICAP 2023), 125. Kyiv, Ukraine: Taras Shevchenko National University of Kyiv, Faculty of RadioPhysics, Electronics and Computer Systems, 2023.
2. Chepel, Leonid, and Yuriy Boyko. “Improving IoT with Blockchain: Consensus and Energy Efficiency.” In Proceedings of the XIX International Scientific Conference Electronics and Applied Physics (APHYS 2023), 76. Kyiv, Ukraine: Taras Shevchenko

National University of Kyiv, Faculty of RadioPhysics, Electronics and Computer Systems, 2023.

3. Sribnyi, Valerii, Leonid Chepel, and Yuriy Boyko. “Decentralization of Information Processing in IoT Infrastructure.” In *XXIV International Young Scientists Conference on Applied Physics (ICAP 2024)*, 127. Kyiv, Ukraine: Taras Shevchenko National University of Kyiv, Faculty of RadioPhysics, Electronics and Computer Systems, 2024.
4. В.О. Срібний, Л.В. Чепель, and Ю.В. Бойко. “Застосування ланцюга блоків для опрацювання даних IMP.” In Матеріали XV Міжнародної конференції з прикладної біофізики, біоніки та біокібернетики, 37–38. Київ, Україна: Інститут магнетизму НАН України та МОН України, 2024.
5. Чепель, Л.В. and Ю.В. Бойко. “Автоматизація процесів в IoT з використанням Blockchain.” In Матеріали II Всеукраїнської науково-практичної конференції з міжнародною участю “PROKYIV”, 479–480. Київ, Україна: КНДУ, Науково-дослідний інститут соціально-економічного розвитку міста, 2025.
6. Chepel, Leonid and Yuriy Boyko. “Blockchain as the Next Evolution Step for IoT.” In Proceedings of the 5th International Scientific and Practical Conference “Scientific Exploration: Bridging Theory and Practice”, 65–67. Berlin, Germany: European Open Science Space, 2025. https://doi.org/10.70286/EOSS-20.10.2025.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ 13

ВСТУП 15

РОЗДІЛ 1. ПОРІВНЯЛЬНИЙ АНАЛІЗ БЛОКЧЕЙН-ПЛАТФОРМ ТА МЕТОДІВ КООРДИНАЦІЇ ВУЗЛІВ ДЛЯ МЕРЕЖ ІНТЕРНЕТУ РЕЧЕЙ 22

1.1. Інтернет речей: виклики безпеки, децентралізації та масштабування 22
1.1.1. Обмеження централізованих архітектур та потреба у децентралізації 23
1.1.2. Виклики масштабованості 24
1.1.3. Класифікація атак в мережах Інтернету речей 24
1.1.4. Протоколи внутрішніх мереж Інтернету речей та їхні безпекові обмеження 27
1.1.5. Виклики для кінцевих пристроїв 28
1.1.6. Взаємозв'язок викликів та необхідність комплексного підходу 29

1.2. Порівняльний аналіз блокчейн-платформ для Інтернету речей 30
1.2.1. Ethereum (публічна мережа, Proof of Stake) 31
1.2.2. Ethereum (приватна мережа, Proof of Authority / Clique) 33
1.2.3. Платформа Hyperledger Fabric 35
1.2.4. DAG-блокчейн та альтернативи 38
1.2.5. Зведений порівняльний аналіз та існуючі блокчейн рішення 40

1.3. Критеріальні методи вибору вузлів у блокчейн-мережах Інтернету речей 42
1.3.1. Існуючі підходи до критеріального вибору вузлів 43
1.3.2. Математичні підходи до критеріального вибору 47
1.3.3. Додаткові критерії спеціалізованих мереж 49

1.4. Підходи до авторизації в блокчейн-системах Інтернету речей 50

1.5. Постановка задачі дослідження 54

РОЗДІЛ 2. МОДЕЛЬ ГІБРИДНОЇ БАГАТОРІВНЕВОЇ БЛОКЧЕЙН-АРХІТЕКТУРИ ДЛЯ МЕРЕЖ ІНТЕРНЕТУ РЕЧЕЙ 57

2.1. Принципи побудови гібридної блокчейн-архітектури для Інтернету речей 57
2.1.1. Формування вимог до блокчейн-архітектури мережі Інтернету речей 57
2.1.2. Використання тришарового підходу для побудови гібридної блокчейн-архітектури 59

2.2. Рівень сенсорів та малопотужних пристроїв 62
2.2.1. Характеристика пристроїв рівня сенсорів. 62

2.2.2. Механізм динамічних таблиць доступу (BDT) 63
2.2.3. Протокол обміну ключами 64
2.2.4. Інтерфейс взаємодії з рівнем блокчейн-мережі 65
2.3. Рівень локальної блокчейн-мережі (IoT-кластер) 66
2.3.1. Організація IoT-кластера 67
2.3.2. Вибір консенсусу та реалізація смарт-логіки для кластерного рівня 67
2.3.3. Розподілене зберігання даних 69
2.3.4. Автономна робота кластера 69
2.4. Рівень зовнішньої блокчейн-мережі 70
2.4.1. Призначення та функції зовнішнього рівня 71
2.4.2. Агрегація транзакцій та механізм повтору 71
2.4.3. Вибір типу блокчейну для зовнішнього рівня 72
2.5. Порівняльний аналіз запропонованої архітектури з існуючими рішеннями .. 75
2.6. Висновки 77
РОЗДІЛ 3. РОЗРОБЛЕННЯ МЕТОДУ АДАПТИВНОГО КОНСЕНСУСУ PROOF OF INDICATORS 78
3.1. Аналіз протоколу Clique як базового консенсусу 78
3.2. Оптимізація розміру блоків: компактні блоки (Light Blocks) 81
3.3. Метрики ефективності та механізм динамічного вибору валідаторів 84
3.4. Модель безпеки адаптивного вибору валідаторів 89
3.5. Конфігуровна затримка старту позачергових вузлів 90
3.6. Програмна реалізація PoI 91
3.6.1. Модифікації Go-Ethereum Clique 92
3.6.2. Конфігурація через файл genesis 93
3.6.3. Модифікації мережевого протоколу ETH68 94
3.7. Експериментальне дослідження PoI 95
3.7.1. Тестове середовище та методика 96
3.7.2. Результати поетапного тестування 99
3.8. Висновки 103
РОЗДІЛ 4. УДОСКОНАЛЕННЯ БЛОКЧЕЙН-МОДЕЛІ МУЛЬТИФАКТОРНОГО УПРАВЛІННЯ ПЕРИФЕРІЙНИМИ ВУЗЛАМИ 105
4.1. Тришарова архітектура блокчейн-координованих периферійних обчислень 105
4.2. Модель мультифакторного вибору вузлів 108
4.2.1. Обмеження класичного підходу 108

4.2.2. Мультифакторна скорингова функція 109
4.3. Механізми забезпечення надійності 113
4.4. Програмна реалізація платформи мультифакторного управління 115
4.4.1. Смарт-контракти 115
4.4.2. Периферійні вузли 117
4.4.3. Взаємодія компонентів 118
4.5. Експериментальне дослідження 120
4.6. Поєднання системи PoI та мультифакторного управління вузлами 122
4.7. Висновки 123
ВИСНОВКИ 125
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 128

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ACL – Список контролю доступу (Access Control List)

AES – Удосконалений стандарт шифрування (Advanced Encryption Standard)

AHP – Метод аналізу ієрархій (Analytic Hierarchy Process)

AMQP – Розширений протокол черги повідомлень (Advanced Message Queuing Protocol)

API – Інтерфейс прикладного програмування (Application Programming Interface)

BDT – Динамічна таблиця блокчейну (Blockchain Dynamic Table)

BLE – Bluetooth з низьким енергоспоживанням (Bluetooth Low Energy)

CoAP – Обмежений прикладний протокол (Constrained Application Protocol)

CPU – Центральний процесор (Central Processing Unit)

CVE – Загальні вразливості та впливи (Common Vulnerabilities and Exposures)

DAG – Спрямований ациклічний граф (Directed Acyclic Graph)

DDoS – Розподілена відмова в обслуговуванні (Distributed Denial of Service)

DID – Децентралізований ідентифікатор (Decentralized Identifier)

DoS – Відмова в обслуговуванні (Denial of Service)

DPoS – Делегований доказ частки (Delegated Proof of Stake)

DRL – Глибоке навчання з підкріпленням (Deep Reinforcement Learning)

ECC – Криптографія на еліптичних кривих (Elliptic Curve Cryptography)

ECDSA – Алгоритм цифрового підпису на еліптичних кривих (Elliptic Curve Digital Signature Algorithm)

ETH – Мережевий протокол Ethereum (Ethereum Network Protocol)

EVM – Віртуальна машина Ethereum (Ethereum Virtual Machine)

HIP – Протокол ідентифікації хостів (Host Identity Protocol)

IPFS – Міжпланетарна файлова система (InterPlanetary File System)

IoT – Інтернет речей (Internet of Things)

MCDM – Багатокритеріальне прийняття рішень (Multi-Criteria Decision Making)

MITM – Атака «людина посередині» (Man-in-the-Middle)

MQTT – Транспорт телеметрії черг повідомлень (Message Queuing Telemetry Transport)

NIST – Національний інститут стандартів і технологій (National Institute of Standards and Technology)

P2P – Однорангова мережа (Peer-to-Peer)

PoA – Доказ повноваженості (Proof of Authority)

PoI – Доказ показників (Proof of Indicators)

PoS – Доказ частки (Proof of Stake)

PoW – Доказ роботи (Proof of Work)

PUF – Фізична неклонована функція (Physical Unclonable Function)

QoS – Якість обслуговування (Quality of Service)

RFID – Радіочастотна ідентифікація (Radio-Frequency Identification)

SPOF – Єдина точка відмови (Single Point of Failure)

TC – Управління трафіком Linux (Linux Traffic Control)

TLS – Безпека транспортного рівня (Transport Layer Security)

TOPSIS – Метод наближення до ідеального рішення (Technique for Order of Preference by Similarity to Ideal Solution)

TPS – Кількість транзакцій за секунду (Transactions Per Second)

VC – Верифіковані облікові дані (Verifiable Credentials)

WPM – Модель зваженого добутку (Weighted Product Model)

WSM – Модель зваженої суми (Weighted Sum Model)

Zk-Rollups – Згортання з доказом нульового знання (Zero-Knowledge Rollups)

ОЗП – Оперативний запам'ятовуючий пристрій

ПЗП – Постійний запам'ятовуючий пристрій

ВСТУП

Обгрунтування вибору теми дослідження. Інтернет речей є однією з найбільш динамічних парадигм сучасної комп'ютерної інженерії, що передбачає об'єднання фізичних пристроїв різного призначення у взаємопов'язану мережу з можливістю збору, передачі та обробки даних. За прогнозами аналітичних компаній, кількість IoT-пристроїв у світі перевищила 21 мільярд у 2025 році, а до 2030 року очікується зростання до 39 мільярдів підключених пристроїв [1]. Це стрімке масштабування зумовлене поширенням мереж 5G, розвитком периферійних обчислень та інтеграцією штучного інтелекту у промислові, логістичні, енергетичні та безпекові системи.

Водночас розширення IoT-екосистем створює фундаментальні виклики, пов'язані з безпекою, децентралізацією та масштабованістю. Традиційні централізовані хмарні архітектури мають критичні обмеження: єдина точка відмови, затримка при передачі даних до хмари, залежність від зовнішніх провайдерів та обмежена масштабованість при зростанні кількості пристроїв. Гетерогенність IoT-мереж, різноманітність пристроїв, протоколів та сценаріїв застосування додатково ускладнює створення єдиної захисної інфраструктури внаслідок обмежених обчислювальних ресурсів кінцевих пристроїв.

Технологія блокчейн, завдяки децентралізації, криптографічному захисту, незмінності записів та підтримці смарт-контрактів, є перспективним інструментом для вирішення викликів IoT-мереж. Проте аналіз існуючих блокчейн-платформ засвідчує, що жодна з них самостійно не задовольняє всі вимоги IoT-середовищ одночасно. Публічна мережа Ethereum з консенсусом на основі застави (Proof of Stake, PoS) має транзакційні комісії та складність розгортання на пристроях з обмеженими ресурсами. Корпоративний блокчейн Hyperledger Fabric забезпечує більшу гнучкість, але є надмірно ресурсоємним для IoT-пристроїв. Рішення на основі спрямованого ациклічного графа дозволяють працювати на малопотужних пристроях, проте мають обмеження детермінованості ланцюгів та продуктивності смарт-контрактів. Приватна

мережа Ethereum з консенсусом довірених валідаторів (PoA) демонструє оптимальний баланс продуктивності, однак має структурні обмеження: не враховує гетерогенність пристроїв і не адаптується до змінних умов функціонування мережі.

Таким чином, розроблення гібридної блокчейн-архітектури з адаптивним консенсусом та ефективними методами координації вузлів є актуальним науково-технічним завданням, спрямованим на підвищення продуктивності, надійності, безпеки та масштабованості IoT-мереж.

Мета і завдання дослідження. Метою дисертаційної роботи є підвищення ефективності функціонування блокчейн-орієнтованих мереж IoT шляхом розроблення гібридної багаторівневої архітектури, адаптивного консенсус-протоколу з динамічним вибором валідаторів та методу мультифакторного управління обчислювальними вузлами.

Варто уточнити, що поняття «ефективність функціонування блокчейн-орієнтованої мережі IoT» у дисертаційній роботі трактується відповідно до категорії Performance Efficiency міжнародного стандарту ISO/IEC 25010 [2], який визначає модель якості програмних і програмно-апаратних систем. Згідно зі стандартом, ця категорія поєднує три підхарактеристики – часові характеристики (time behaviour), використання ресурсів (resource utilization) та пропускна здатність (capacity). Додатково, для оцінки стабільності досягнення консенсусу та завершення делегованих обчислень, використовується показник з категорії Reliability того ж стандарту.

Для досягнення поставленої мети в роботі передбачено вирішення таких дослідницьких завдань:

1. проведення порівняльного аналізу блокчейн-платформ та методів координації вузлів для використання в мережах IoT;
2. розроблення моделі гібридної багаторівневої блокчейн-архітектури для мереж IoT з кластерною організацією та обгрунтування вибору типу блокчейну для кожного рівня;

3. розроблення методу адаптивного консенсусу Proof of Indicators з динамічним вибором валідаторів та оптимізацією трафіку; здійснення програмної реалізації та проведення експериментального дослідження цього методу;
4. удосконалення моделі мультифакторного вибору вузлів для блокчейн-координованих периферійних обчислень IoT; здійснення програмної реалізації.

Об'єктом дослідження є блокчейн-орієнтовані комп'ютерні мережі пристроїв Інтернету речей.

Предметом дослідження є методи і засоби побудови мереж IoT на основі гібридної блокчейн-архітектури з адаптивним консенсусом та мультифакторним управлінням вузлами.

Методи дослідження. У дисертаційній роботі застосовано комплекс наукових методів. Для порівняльного аналізу блокчейн-платформ застосовано метод порівняння за технічними характеристиками з академічних джерел і документації платформ. Для оцінки вузлів на двох рівнях координації використано методи багатокритеріального прийняття рішень (MCDM): адитивну скорингову функцію, концептуально подібну до моделі зваженої суми (WSM), для оцінки ефективності валідаторів у консенсус-протоколі PoI; мультиплікативну скорингову функцію, побудовану на основі моделі зваженого добутку (WPM), для вибору периферійних обчислювальних вузлів. Для розроблення PoI додатково застосовано методи моделювання мережевих процесів та статистичного аналізу мережевих метрик.

Програмна реалізація протоколу PoI здійснена мовою Go на базі проекту Go-Ethereum, смарт-контракти реалізовано мовою Solidity, а периферійні обчислювальні вузли – мовою Python з використанням бібліотеки psutil для моніторингу ресурсів. Для експериментальної валідації використано Docker-контейнеризацію із моделюванням мережевих затримок засобами Linux Traffic Control, а мережевий аналіз проведено за допомогою інструментів Edgeshark та Wireshark. Статистична обробка результатів виконана із застосуванням логнормального розподілу для мережевих метрик.

Наукова новизна отриманих результатів. Наукова новизна дисертаційної роботи полягає в обгрунтуванні, розробці та експериментальній перевірці нових методів і засобів побудови блокчейн-орієнтованих мереж IoT з адаптивним консенсусом та мультифакторним управлінням вузлами.

1. **Удосконалено** модель гібридної багаторівневої блокчейн-архітектури для мереж IoT, яка, на відміну від відомих рішень, використовує тришарову кластерну організацію: рівень сенсорів із симетричним шифруванням для апаратно обмежених пристроїв, рівень локальної блокчейн-мережі з підтримкою смарт-контрактів та адаптивним консенсусом, рівень зовнішньої блокчейн-мережі з агрегацією транзакцій. При цьому вибір типу блокчейну на кожному рівні обгрунтовано на основі порівняльного аналізу за критеріями пропускної здатності, безпеки та придатності для IoT. Такий підхід забезпечує побудову масштабованої, децентралізованої та безпечної мережі IoT з підтримкою гетерогенних пристроїв та можливістю автономної роботи кластерів.
2. **Вперше запропоновано** метод адаптивного консенсусу Proof of Indicators (PoI) для блокчейн-мереж IoT, який, на відміну від інших реалізацій, впроваджує динамічний механізм вибору валідаторів на основі комплексної метрики ефективності, техніку поширення оптимізованих блоків з передачею хешів транзакцій замість повних даних та затримку генерації блоків позачерговими вузлами для зменшення ймовірності форків.
3. **Удосконалено** модель вибору вузлів для блокчейн-координованих периферійних обчислень в мережах IoT, яка, на відміну від відомих підходів, використовує мультифакторну скорингову функцію мультиплікативної композиції, що поєднує репутацію вузла, метрики реального часу, із розділенням обчислень поза блокчейном та координації через смарт-контракти, що надає змогу запобігти концентрації задач на перевантажених вузлах та забезпечити

автоматичне балансування навантаження з лінійною обчислювальною складністю.

Практичне значення отриманих результатів роботи полягає в створенні програмних засобів, придатних для безпосереднього впровадження: реалізації протоколу PoI мовою Go на базі Go-Ethereum, системи мультифакторного управління вузлами на базі смарт-контрактів та Python-агентів, результатів порівняльного аналізу блокчейн-платформ та моделей для спеціалізованих сценаріїв. Запропоновані рішення функціонують на пристроях класу Raspberry Pi та орієнтовані на широке практичне використання.

Структура дисертаційної роботи. Дисертаційна робота складається з чотирьох розділів, які охоплюють теоретичне підгрунтя дослідження, розроблення архітектурних моделей, створення нових методів та їх експериментальну перевірку.

У першому розділі проведено порівняльний аналіз блокчейн-платформ та методів координації вузлів для мереж IoT. Проаналізовано виклики безпеки, децентралізації та масштабованості IoT-мереж. Здійснено порівняльний аналіз блокчейн-платформ Ethereum PoA/PoS, Hyperledger Fabric та блокчейну на базі ациклічного спрямованого графу за критеріями затримки, пропускної здатності, використання ресурсів та безпеки. Проаналізовано критеріальні методи вибору вузлів. Обгрунтовано доцільність гібридного підходу та сформульовано задачі дослідження.

Другий розділ присвячено розробленню моделі гібридної багаторівневої блокчейн-архітектури для мереж IoT. Описано тришарову кластерну організацію, що включає рівень сенсорів із симетричним шифруванням, рівень локальної блокчейн-мережі з адаптивним консенсусом та рівень зовнішньої блокчейн-мережі з агрегацією транзакцій. Обгрунтовано вибір типу блокчейну для кожного рівня та розглянуто адаптацію архітектури для спеціалізованих сценаріїв, включаючи військове та цивільне IoT.

У третьому розділі розроблено метод адаптивного консенсусу Proof of Indicators. Проаналізовано протокол Clique як базовий консенсус, запропоновано техніку компактних блоків для оптимізації мережевого трафіку, розроблено метрики ефективності та механізм динамічного вибору валідаторів. Здійснено програмну реалізацію PoI мовою Go на базі Go-Ethereum та проведено експериментальне дослідження у Docker-середовищі з 15 вузлами, яке продемонструвало зменшення мережевого трафіку на 20,5%, скорочення форків на 80%, підвищення TPS до 10% та зменшення часу поширення блоків на 8,5% у порівнянні з базовим PoA.

Четвертий розділ присвячено удосконаленню моделі мультифакторного управління вузлами. Розроблено мультифакторну скорингову функцію мультиплікативної композиції для вибору оптимального периферійного вузла. Здійснено програмну реалізацію на базі смарт-контрактів Solidity та Python-вузлів. Проведено експериментальне дослідження мультифакторного вибору та проаналізовано отримані результати.

Особистий внесок здобувача. Дисертаційна робота є результатом самостійних розробок автора. Усі роботи виконано у співавторстві з науковим керівником, здобувачу належать наступні результати: [3] – аналіз поточних викликів безпеки мереж IoT та існуючих підходів до їх організації, формулювання вимог до гібридної архітектури, розроблення багатошарової структури IoT мережі з блокчейном; [4] – аналіз актуальних досліджень щодо застосування блокчейну в IoT-системах військового призначення, порівняльний аналіз платформ, розроблення комплексного підходу до побудови захищеної децентралізованої системи управління пристроями IoT військового призначення; [5] – розроблення протоколу консенсусу Proof of Indicators (PoI) на основі механізму Clique платформи Go-Ethereum, реалізація системи динамічного вибору вузлів за показниками ефективності та механізму компактних блоків для зменшення мережевого навантаження, програмна реалізація протоколу мовою Go, проведення порівняльного експериментального дослідження PoI та Clique

у симульованій IoT-мережі з наведеними затримками; [6] – розроблення гібридної платформи периферійних обчислень на основі блокчейну, розроблення мульти-факторної функції вибору вузлів, яка поєднує репутацію вузла з його актуальними метриками, реалізація координаційної логіки на смарт-контрактах Ethereum, проведення практичної валідації у симульованій Docker-мережі та аналіз результатів; [7] – аналіз поточного стану та проблем використання консенсус-механізмів у блокчейн-системах для IoT; [8] – аналіз проблем централізації IoT-інфраструктур та обґрунтування необхідності децентралізації (у співавторстві з В.О. Срібним); [9] – розроблення концепції автоматизації процесів в IoT-мережах на основі блокчейну; [10] – аналіз розвитку застосування блокчейну в IoT-екосистемах за ключовими напрямами (управління ідентифікацією, ланцюги постачання, периферійні обчислення, смарт-контракти для міжмашинна взаємодія); [11] – аналіз проблем безпеки та централізації в інфраструктурі Інтернету медичних речей (IMP), розроблення підходу до застосування блокчейну для зберігання і передачі медичних даних у розподіленій IMP-мережі (у співавторстві з В.О. Срібним).

Структура та обсяг дисертації. Дисертаційна робота складається зі вступу, чотирьох розділів, висновків та списку використаних джерел. Загальний обсяг роботи складає 134 сторінки з яких основний зміст викладено на 112 сторінках. Дисертація містить 19 рисунків, 13 таблиць та список використаних джерел, що включає 85 найменувань.

РОЗДІЛ 1. ПОРІВНЯЛЬНИЙ АНАЛІЗ БЛОКЧЕЙН-ПЛАТФОРМ ТА МЕТОДІВ КООРДИНАЦІЇ ВУЗЛІВ ДЛЯ МЕРЕЖ ІНТЕРНЕТУ РЕЧЕЙ

Розділ присвячено аналізу поточного стану блокчейн-технологій та методів координації вузлів у мережах Інтернету речей. Розглянуто виклики безпеки, децентралізації та масштабованості IoT-мереж, систематизовано основні типи атак і механізми протидії їм. Виконано порівняння блокчейн-платформ – Ethereum PoA/PoS, Hyperledger Fabric та DAG-блокчейнів – за критеріями латентності, пропускної здатності, ресурсоємності та безпеки. Також досліджено існуючі методи координації та вибору вузлів у розподілених системах: репутаційні підходи, DRL-методи, пріоритетні черги. Узагальнення результатів аналізу дало змогу обгрунтувати доцільність гібридного підходу та сформулювати постановку задачі, що розв'язується у подальших розділах роботи.

1.1. Інтернет речей: виклики безпеки, децентралізації та масштабування

Інтернет речей (Internet of Things, IoT) – одна з найбільш динамічних парадигм сучасної комп’ютерної інженерії. Дана концепція передбачає об’єднання фізичних пристроїв різного призначення – від побутових сенсорів і носимих пристроїв до складних промислових систем, медичного обладнання й компонентів розумного міста – у єдину взаємопов’язану мережу, здатну збирати, передавати та обробляти дані [12]. Масштаб поширення IoT-технологій продовжує зростати: за оцінками IoT Analytics, кількість активних IoT-пристроїв у світі досягла приблизно 21,1 мільярда у 2025 році, а прогнози на 2030 рік вказують на подальше зростання до 39 мільярдів підключених пристроїв [1]. Причинами такого зростання є поширення мереж 5G, розвиток периферійних обчислень, інтеграція штучного інтелекту та розширення сфер застосування IoT у промисловості, логістиці, енергетиці та системах безпеки [13]. Разом із тим стрімке масштабування IoT-екосистем породжує виклики, які можна

класифікувати за трьома взаємопов’язаними напрямками: безпека, децентралізація та масштабованість.

1.1.1. Обмеження централізованих архітектур та потреба у децентралізації

Традиційні IoT-системи побудовані за хмароцентричною архітектурою, де всі дані від кінцевих пристроїв передаються до централізованого хмарного сервера для обробки, зберігання та прийняття рішень [14]. Дана архітектура має обмеження, що стають критичними зі зростанням масштабу IoT-мереж. Передусім, централізований сервер є єдиною точкою відмови (Single Point of Failure, SPOF) – компрометація або відмова центрального компонента призводить до повної втрати функціональності всієї системи. За даними досліджень 2024 року, 22% організацій зазнали серйозних інцидентів безпеки, пов’язаних із IoT-системами, протягом року, причому значна частина інцидентів була зумовлена саме централізованою природою архітектурних рішень [15]. Централізоване зберігання даних також створює привабливу мішень для кібератак: компрометація єдиної бази даних надає зловмиснику доступ до всього масиву інформації – даних телеметрії, облікових записів пристроїв та конфігураційних параметрів [16].

Проблема залежності від централізованого постачальника є ще одним обмеженням хмароцентричних архітектур: міграція IoT-інфраструктури між хмарними платформами потребує значних витрат та може призвести до тривалих простоїв [17]. Для спеціалізованих IoT-мереж – зокрема мереж критичної інфраструктури, військових або промислових мереж – залежність від зовнішніх хмарних провайдерів неприйнятна з міркувань безпеки та автономності. Робота [18] на прикладі військового IoT показує, що мережі критичного призначення повинні працювати автономно коли зв'язок із зовнішньою інфраструктурою втрачено. Хмароцентрична модель таку вимогу не витримує. Тобто децентралізація IoT-архітектур стає не бажаним, а необхідним напрямком розвитку, який передбачає розподіл функцій управління, зберігання та обробки даних між вузлами мережі. Однак

децентралізація створює власний набір викликів, що потребують окремих рішень: узгодженість даних, координація розподілених вузлів та управління гетерогенним апаратним забезпеченням [19].

1.1.2. Виклики масштабованості

Масштабованість у IoT має кілька вимірів: кількість пристроїв, обсяг даних, мережевий трафік та обчислювальні задачі. Централізовані архітектури стикаються з проблемами пропускної здатності при передачі даних від мільярдів пристроїв до хмари – канали перевантажуються, затримки зростають, а вартість інфраструктури збільшується пропорційно обсягу даних [13], [14].

Гетерогенність мереж додає інший вимір: пристрої різних виробників використовують різні протоколи, формати даних і програмні інтерфейси [20]. Відсутність домінуючих універсальних стандартів означає, що кожне розширення мережі потенційно вимагає нових адаптерів і мостів між протоколами. Периферійні та туманні (Fog) обчислення зменшують трафік до хмари, знижують затримку передачі даних і дозволяють приймати рішення в реальному часі, проте порушують питання координації між вузлами, узгодженості даних та розподілу задач. Мікросервісна архітектура IoT-платформ реалізує незалежно масштабовані компоненти, але проблема координації між сервісами, управління розподіленим станом та цілісності даних залишається відкритою без механізму довіри та верифікації. Масштабованість, таким чином, тісно пов'язана з децентралізацією: перехід до розподіленої архітектури знімає навантаження з центрального вузла, але створює потребу в механізмах координації.

1.1.3. Класифікація атак в мережах Інтернету речей

Аналіз загроз безпеці IoT-мережі потребує систематичної класифікації існуючих типів атак та відповідних механізмів протидії. Атаки маршрутизації – wormhole, Sybil, Grayhole, blackhole, sinkhole, replay, spoofing та hello flood – являють собою один із

найпоширеніших класів загроз для мережевого рівня IoT [21]. Атака типу wormhole передбачає створення зловмисником тунелю між двома віддаленими точками мережі, через який пакети передаються з мінімальною затримкою, що дозволяє маніпулювати механізмами маршрутизації. Атака Sybil полягає у створенні множинних фальшивих ідентичностей одним вузлом – це порушує механізми голосування та розподілу ресурсів. Атаки selective forwarding та blackhole реалізуються через вибіркове або повне блокування мережевого трафіку скомпрометованим вузлом. Основними механізмами протидії атакам маршрутизації є використання розподіленої мережевої топології, опрацювання ризиків для критичних зон та застосування шифрування на основі еліптичних кривих для обміну ключами [3].

Атаки з використанням радіочастотної ідентифікації (RFID) становлять окремий клас загроз. Зловмисник може клонувати RFID-мітку, перехопити дані під час бездротової передачі або виконати атаку відтворення, повторивши раніше перехоплений сигнал. Ефективним рішенням для запобігання клонуванню вважається використання фізичних методів ідентифікації – зокрема фізичних неклонованих функцій (PUF), що генерують унікальний цифровий відбиток пристрою на основі фізичних процесів, параметри яких є унікальними для конкретного мікроконтролера [22]. Атаки типу «людина посередині» (MITM) та атаки на відмову в обслуговуванні (DoS) – класичні загрози, спрямовані на перехоплення та модифікацію трафіку або перевантаження мережі та її компонентів відповідно. Протидія цим атакам передбачає фільтрування пакетів даних, використання списків контролю доступу (ACL) та шифрування каналів передачі [16].

Ботнет-атаки становлять особливу загрозу для IoT-середовищ: велика кількість слабко захищених пристроїв може бути об’єднана у ботнет для проведення масштабних DDoS-атак. Для протидії застосовується закриття невикористовуваних портів, впровадження шифрування та використання спеціалізованих скриптів перевірки безпеки [23]. Загрози, пов’язані зі шкідливим програмним забезпеченням та

додаванням скомпрометованих вузлів до мережі, потребують впровадження механізмів колективної атестації IoT-оточення, що дає змогу виявляти аномальну поведінку пристроїв та блокувати несанкціоновані зміни в мережевій конфігурації [24]. Зведені результати аналізу основних типів атак та відповідних механізмів протидії наведено у Таблиці 1.1.

Таблиця 1.1 – Класифікація основних атак в IoT-мережах та механізми протидії

Тип атаки	**Механізм атаки**	**Рішення**
Атаки маршрутизації (wormhole, Sybil, Grayhole, blackhole, sinkhole, replay, spoofing, hello flood)	Маніпулювання протоколами маршрутизації, створення фальшивих ідентичностей, блокування трафіку	Використання розподіленої мережі, шифрування на основі еліптичних кривих, опрацювання ризиків для критичних зон
RFID-атаки	Клонування міток, перехоплення даних, відтворення сигналу	Фізичні методи запобігання клонуванню (PUF)
MITM та DoS атаки	Перехоплення/модифікація трафіку, перевантаження мережі	Фільтрування пакетів, ACL, шифрування каналів
Ботнет-атаки	Об’єднання IoT-пристроїв у ботнет для DDoS	Закриття портів, шифрування, скрипти перевірки безпеки
Шкідливе ПЗ / додавання шкідливого вузла	Модифікація ПЗ пристрою, підміна вузлів	Колективна атестація оточення, верифікація цілісності

Джерело: складено автором на основі [16], [21], [22], [23], [24]

1.1.4. Протоколи внутрішніх мереж Інтернету речей та їхні безпекові обмеження

Мережі IoT побудовані на різних протоколах, і кожен із них має специфічні характеристики щодо пропускної здатності, дальності дії, енергоспоживання та рівня безпеки. Протокол ZigBee підтримує шифрування передачі даних, проте не гарантує комплексної безпеки, що вимагає доповнення рішень на основі ZigBee додатковими безпековими компонентами [25]. Протокол 6LoWPAN використовує симетричне шифрування без стандартизованого механізму ротації ключів, та має вразливість до DoS-атак [3]. Z-Wave є закритою системою з обмеженою доступністю інформації про безпекові механізми – це ускладнює незалежний аудит його захищеності. Bluetooth Low Energy (BLE) використовує шифрування AES-128, що може бути скомпрометоване при нечастій ротації за допомогою квантових брут-форс атак [26], що актуалізує питання міграції до більш стійких криптографічних алгоритмів у довгостроковій перспективі.

Попри задокументовані вразливості, ці протоколи продовжують домінувати в реальних розгортаннях IoT. Причина – компроміс між безпекою та практичними обмеженнями: ZigBee забезпечує mesh-маршрутизацію при низькому споживанні енергії, BLE – широку сумісність із мобільними пристроями, а 6LoWPAN – інтеграцію з існуючою IPv6-інфраструктурою. Жоден з альтернативних протоколів не пропонує аналогічного балансу функціональності та енергоефективності, що і пояснює збереження їх ринкової позиції попри відомі ризики. Даний компроміс між безпекою та практичністю є системним обмеженням, а не наслідком недостатньої обізнаності розробників.

На прикладному рівні IoT-мережі використовують протоколи MQTT, CoAP та AMQP для передачі даних між пристроями та серверами. Протокол MQTT, будучи одним із найпоширеніших у IoT-екосистемах, має значну кількість задокументованих вразливостей: за даними бази CVE, зафіксовано понад 129 згадок щодо вразливостей, пов’язаних з MQTT, причому переважна частина стосується програмної реалізації

[27]. CoAP побудований на базі UDP та спроектований для пристроїв із обмеженими ресурсами, однак відсутність обов'язкового шифрування за замовчуванням знижує рівень його захищеності. AMQP надає більш розвинені механізми безпеки, проте його ресурсоємність обмежує застосування на малопотужних пристроях [28]. Різноманітність протоколів, кожен із яких має власну модель безпеки, створює додатковий виклик для наскрізної безпеки: взаємодія пристроїв на різних протоколах потребує додаткового рівня абстракції та координації.

1.1.5. Виклики для кінцевих пристроїв

Кінцеві IoT-пристрої характеризуються обмеженими обчислювальними ресурсами, і це створює специфічні труднощі щодо реалізації криптографічних протоколів та механізмів безпеки. Повноцінний протокол DTLS працює лише на мікроконтролерах із апаратним прискоренням криптографічних функцій [20]. На менш потужних пристроях застосовується оптимізований протокол DTLS, побудований на базі UDP, що є менш ресурсоємним порівняно з TCP-базованими рішеннями. DTLS при цьому має низку проблем: при використанні серверу делегування виникає проблема з масштабуванням та єдиною точкою відмови, а при використанні DTLS на основі сертифікатів – надмірне використання пам'яті, пов'язане зі зберіганням, обміном та перевіркою сертифікатів [3].

Стосовно автентифікації пристроїв існують різні підходи: від вбудованих сертифікатів та токенів до використання протоколу HIP з асиметричним шифруванням. Використання списків контролю доступу та рольової моделі є одними з найпоширеніших сценаріїв авторизації. Проте фізична компрометація пристрою дозволяє зловмиснику дістати дані з незашифрованої пам'яті, включаючи сертифікати для встановлення з'єднання з сервером, і здійснити підміну пристрою. Саме тому, крім захисту на рівні окремого пристрою, система як мережа пристроїв повинна мати здатність виявляти аномалії та вживати проактивних заходів [3]. Обмеженість ресурсів кінцевих пристроїв також безпосередньо впливає на можливості масштабування та

децентралізації – реалізація повноцінних блокчейн-клієнтів, участь у консенсусних протоколах та зберігання великих обсягів розподіленого реєстру є непрактичними або неможливими для значної частини IoT-пристроїв класу мікроконтролерів.

Суттєвим ризиком залишаються механізми оновлення програмного забезпечення IoT-пристроїв. Далеко не всі операційні системи для пристроїв із обмеженими апаратними можливостями мають вбудовані механізми оновлень, що робить їх більш вразливими до відомих експлойтів. За даними дослідження комерційних IoT-пристроїв [20], 70% з них не застосовували шифрування під час передачі даних – це ілюструє системний характер безпекових проблем в IoT-екосистемі. Уповільнений цикл виявлення й усунення вразливостей зумовлений фрагментованістю екосистеми виробників, різнорідністю апаратного забезпечення та відсутністю єдиних вимог до підтримки пристроїв. Можливим рішенням є обмеження взаємодії таких пристроїв із певним шлюзом, який матиме всі необхідні оновлення, проте це не усуває фундаментальну проблему, а лише переносить її на рівень шлюзового пристрою, створюючи додаткову залежність від єдиної точки доступу.

1.1.6. Взаємозв'язок викликів та необхідність комплексного підходу

Узагальнену класифікацію основних викликів IoT-мереж за трьома напрямками наведено на рис. 1.1. Проведений аналіз засвідчує, що три групи викликів – безпека, децентралізація та масштабованість – є глибоко взаємопов’язаними та не можуть бути вирішені ізольовано.

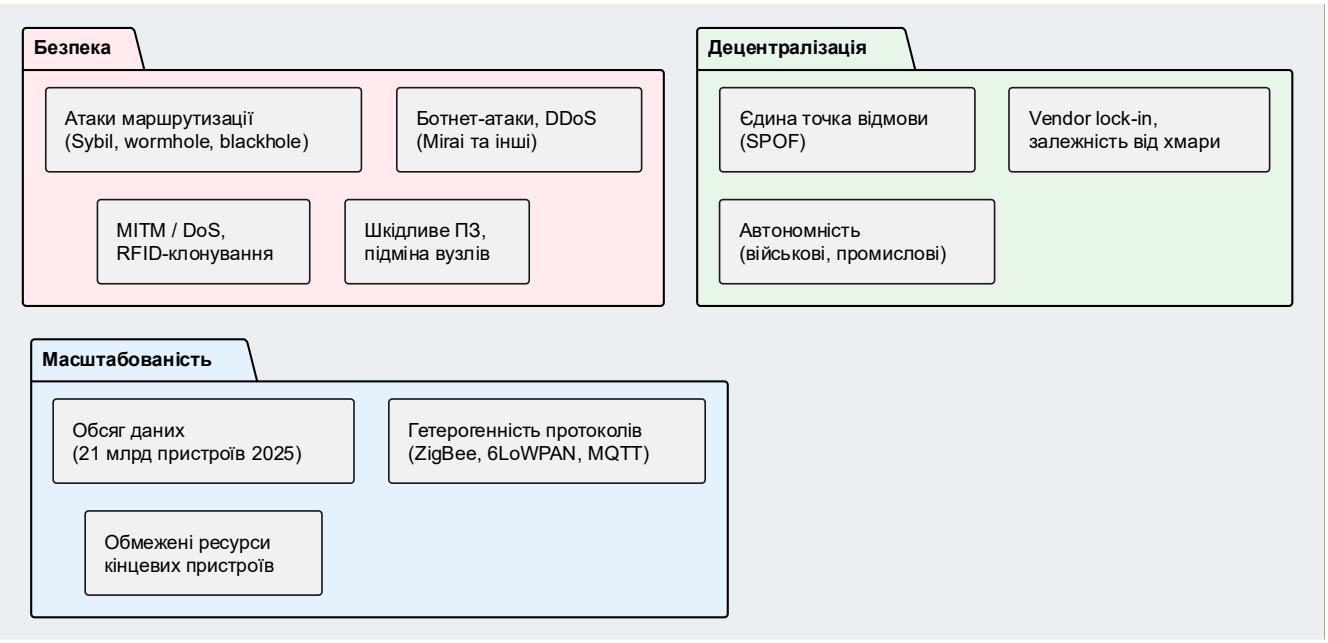

Рисунок 1.1 – Класифікація викликів IoT-мереж

Централізовані архітектури створюють єдину точку відмови та обмежують масштабованість, тоді як децентралізація потребує нових механізмів координації та довіри. Обмеженість ресурсів кінцевих пристроїв ускладнює впровадження як класичних механізмів безпеки, так і повноцінних децентралізованих протоколів. Різноманітність протоколів та стандартів створює бар'єри для масштабування та інтеграції. Це зумовлює доцільність дослідження технології блокчейн як потенційного механізму, здатного одночасно реалізувати децентралізацію управління, додатковий рівень безпеки через криптографічний захист та прозорість, а також координацію гетерогенних мереж через смарт-контракти.

1.2. Порівняльний аналіз блокчейн-платформ для Інтернету речей

Блокчейн як технологія розподіленого реєстру відповідає вимогам IoT-систем до довіри, цілісності та аудиту даних [29], [30], а смарт-контракти дозволяють автоматизувати дії на рівні блокчейн мережі за заздалегідь визначеною логікою [31]. Вибір конкретної блокчейн-платформи для IoT-середовища – нетривіальне завдання:

кожна платформа має специфічні характеристики щодо продуктивності, масштабованості, ресурсоємності та моделі безпеки, і саме ці параметри визначають її придатність для гетерогенних IoT-мереж. Даний підрозділ порівнює основні блокчейн-платформи з точки зору застосовності у таких мережах.

1.2.1. Ethereum (публічна мережа, Proof of Stake)

Ethereum – провідна блокчейн-платформа з підтримкою смарт-контрактів, анонсована у 2014 році [32]. Публічна мережа Ethereum працює на протоколі консенсусу Proof of Stake (PoS), що прийшов на заміну ресурсоємному протоколу консенсусу Proof of Work (PoW) у вересні 2022 року [33]. У моделі PoS валідатори обираються для створення нових блоків пропорційно до обсягу криптовалюти, заблокованої як застава (стейк), що значно знижує енергоспоживання порівняно з PoW. Мова Solidity, яку використовують для смарт-контрактів Ethereum, залишається найпоширенішою мовою розробки децентралізованих застосунків (dApps) і формує найширшу екосистему бібліотек, фреймворків та інструментів.

Протокол Ethereum PoS побудовано на взаємодії двох протоколів: Casper FFG (Friendly Finality Gadget) відповідає за фіналізацію (остаточне підтвердження) блоків, а LMD-GHOST (Latest Message Driven Greedy Heaviest Observed SubTree) визначає правило вибору канонічного ланцюга. Час у мережі Ethereum PoS організовано ієрархічно – кожен слот тривалістю 12 секунд відповідає одному потенційному блоку, а 32 послідовні слоти формують епоху тривалістю 384 секунди. На початку епохи множина активних валідаторів розподіляється на комітети атестації, кожен із яких призначається на конкретний слот для голосування за коректність запропонованого блоку. Для кожного слота окремо визначається валідатор, що пропонує блок, решта – атестують (перевіряють) на валідність виходячи з поточного стану ланцюга. Casper FFG оперує контрольними точками (checkpoints) на межах епох: коли блок-контрольна точка набирає підтримку від щонайменше двох третин загальної заблокованої частки,

він переходить у стан попереднього підтвердження (justified), а попередній justified блок отримує статус остаточного підтвердження (finalized) – після чого його скасування потребує знищення щонайменше третини всієї заблокованої криптовалюти [34]. LMD-GHOST доповнює Casper FFG вибором ланцюга в межах поточної епохи: при виникненні розгалуження (форку) – ситуації, коли два або більше блоків створюються одночасно на одній висоті ланцюга, утворюючи конкуруючі гілки, – валідатори голосують за гілку з найбільшою вагою атестацій, завдяки чому мережа швидко сходиться до єдиного стану.

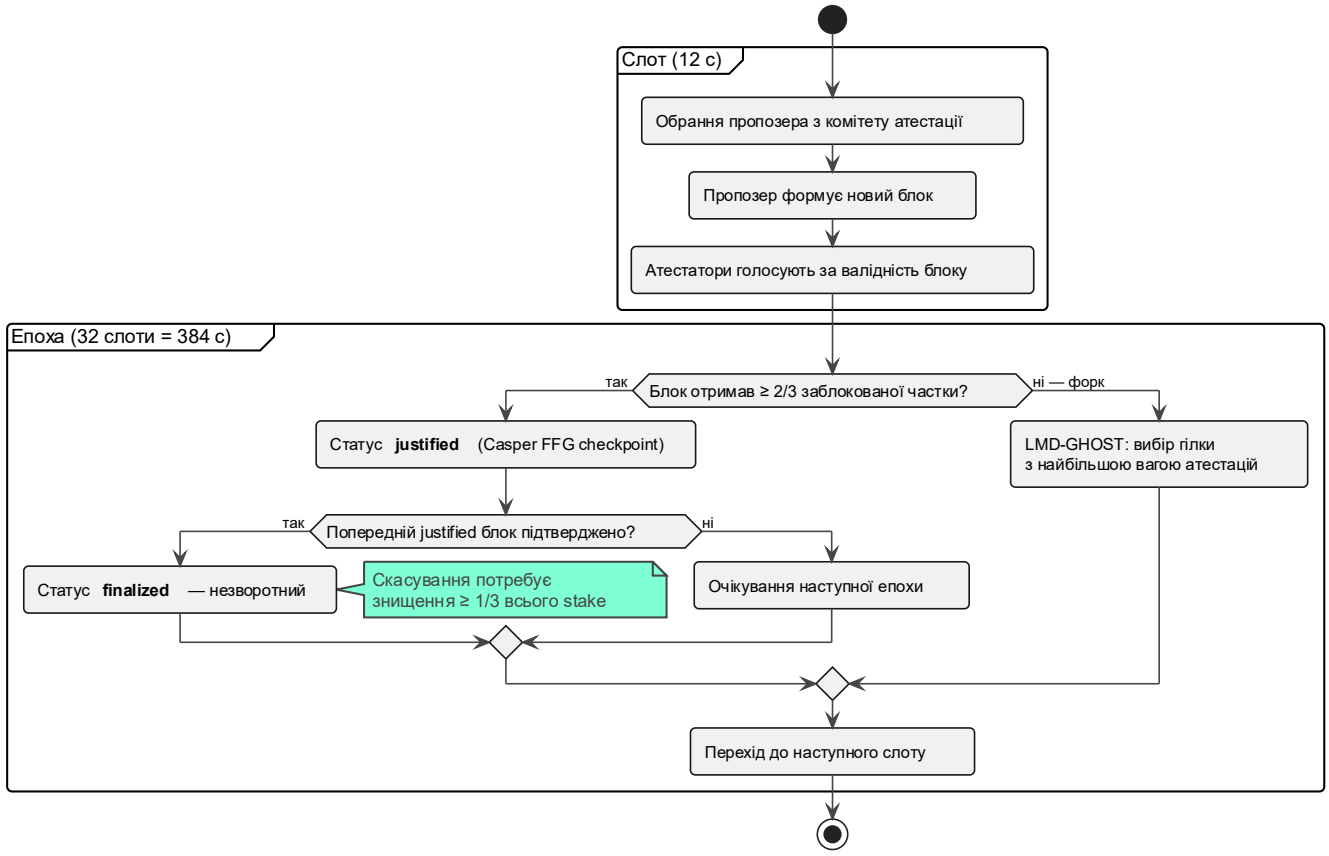

Рисунок 1.2 – Механізм консенсусу Ethereum PoS

Для участі у валідації вузол має заблокувати мінімум 32 ETH як заставу, що на момент переходу мережі на PoS становило суму, еквівалентну десяткам тисяч доларів США [33]. Для IoT-пристроїв це означає значний економічний бар'єр. Протокол також передбачає механізм покарань (slashing): валідатор, який підписує суперечливі

атестації або пропонує конфліктні блоки, втрачає частину заблокованих коштів та примусово виводиться з множини активних валідаторів. У великих публічних мережах даний механізм ефективно запобігає зловмисній поведінці, але додає складності для середовищ, де вузли можуть тимчасово втрачати з'єднання через нестабільні канали зв'язку, що є типовою ситуацією для IoT. Загальну схему функціонування консенсусу Ethereum PoS наведено на рис. 1.2.

Попри переваги, публічна мережа Ethereum має фундаментальні обмеження для IoT-середовищ. Час підтвердження транзакції в PoS Ethereum – щонайменше 12 секунд (один слот), а повна фіналізація потребує від двох до трьох епох, тобто 13–19 хвилин. Для IoT-сценаріїв, де час додавання даних має вимірюватися одиницями секунд, така затримка неприйнятна. Кожна транзакція в публічній мережі Ethereum потребує оплати комісії (gas), що робить систему економічно непрактичною для IoT-пристроїв з великою кількістю дрібних транзакцій телеметрії. Ресурсні вимоги протоколу PoS теж не на користь IoT: валідатор повинен заблокувати значну кількість криптовалюти та підтримувати постійне підключення до мережі з безперервною участю в атестаціях, що непрактично для розподілених IoT-систем із обмеженими ресурсами та нестабільними каналами зв'язку. Як зазначено в [30], в блокчейні Ethereum можна зберігати хеші, тоді як реальні дані варто розміщувати окремо – це зменшує навантаження на мережу. Однак фундаментальні обмеження затримок та комісій залишаються критичними бар'єрами для повноцінного впровадження в IoT.

1.2.2. Ethereum (приватна мережа, Proof of Authority / Clique)

Альтернативою публічній мережі Ethereum для IoT є розгортання приватної мережі з протоколом консенсусу Proof of Authority (PoA), зокрема його реалізацією Clique, специфікованою в EIP-225 (Ethereum Improvement Proposal – стандарт покращення протоколу Ethereum) [35]. Clique базується на концепції уповноважених підписантів (signers) – заздалегідь визначених вузлів, що мають право створювати нові блоки. Протокол розділяє підписантів на три категорії: черговий, позачерговий та

заборонений вузли. Черговий вузол (in-turn) – основний вузол, який створює блок у визначений момент часу. Позачергові вузли (no-turn) – інші вузли, які генерують блоки з випадковою затримкою у разі відмови in-turn вузла. Заборонені вузли (forbidden) - ті, що досягли ліміту підписання і можуть лише отримувати й верифікувати блоки. Схему консенсусу Clique наведено на рис. 1.3.

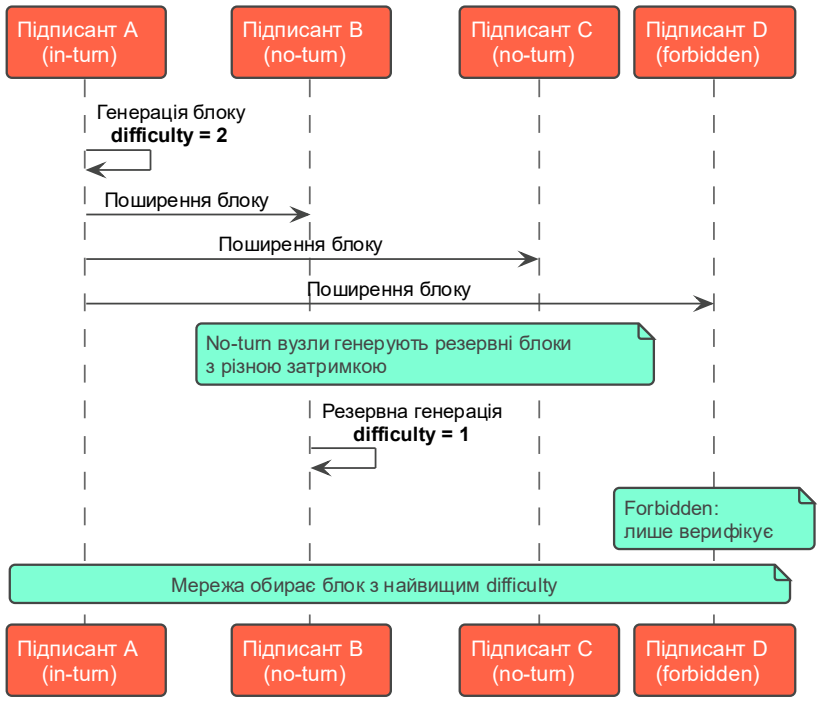

Рисунок 1.3 – Механізм консенсусу Clique (Proof of Authority)

За даними авторів з [36] (тестування приватної блокчейн-мережі до 250 вузлів), PoA / Clique показує найкращий баланс між затримкою підтвердження транзакції (0,6–14,4 мс) та пропускною здатністю (до 8000 транзакцій на секунду), що помітно перевищує результати PoS (12 с). Реалізація Clique в Go-Ethereum є найбільш зрілим та практично перевіреним впровадженням консенсусу довірених валідаторів (PoA), що включає «легкого» клієнта, спеціально розробленого для пристроїв з обмеженими обчислювальними ресурсами [37]. Те, що Clique можна розгорнути на пристроях рівня Raspberry Pi, підтверджує придатність протоколу для ресурсно обмежених IoT-середовищ. Для пристроїв із ще меншими апаратними можливостями передбачено використання спеціалізованої реалізації бібліотеки web3, через яку можна

підключатися до існуючих вузлів-підписантів без повноцінної участі в консенсусі [38]. Альтернативна реалізація PoA – протокол Aura, що конкурує з Clique, однак, як показано в [39], Clique гарантує вищу пропускну здатність транзакцій, що робить його оптимальним вибором для IoT-мереж із різноманітними пристроями.

Попри зазначені переваги, консенсус Clique стикається з трьома структурними обмеженнями, що ускладнюють його роботу в гетерогенних IoT-мережах.

Статичний порядок підписантів. Черговість генерації блоків визначається виключно байтовим представленням Ethereum-адреси вузла. Поточний стан вузла, його обчислювальна продуктивність чи якість мережевого з'єднання – жоден із цих факторів на черговість не впливає. Тож вузол із перевантаженим процесором або нестабільним каналом зв'язку отримує ті самі права підписання, що й потужний сервер.

Надлишкова структура блоків. Кожен стандартний блок Ethereum включає повний текст транзакцій. Для IoT-сценаріїв це надлишково, бо вузли-підписанти зазвичай уже мають відповідні транзакції в локальному буфері транзакцій. Повторна передача цих транзакцій у складі блоку лише збільшує мережевий трафік.

Відсутність адаптації до мережевих умов. Діапазон випадкової затримки позачергових вузлів залишається фіксованим – він не враховує ані поточну топологію мережі, ані реальні затримки між вузлами. Це підвищує ймовірність розгалужень ланцюга блоків і погіршує стабільність мережі загалом.

1.2.3. Платформа Hyperledger Fabric

Hyperledger Fabric – корпоративна блокчейн-платформа консорціумного типу, розроблена під егідою Linux Foundation, що надає розвинені механізми конфіденційності, модульну архітектуру та підтримку каналів для ізоляції даних між організаціями-учасниками [40] [41]. На відміну від традиційної блокчейн-архітектури, де транзакції спочатку впорядковуються і потім послідовно виконуються всіма вузлами, Fabric реалізує архітектуру execute-order-validate, що розділяє етапи

виконання, упорядкування та перевірки транзакцій, забезпечуючи можливість паралельного виконання.

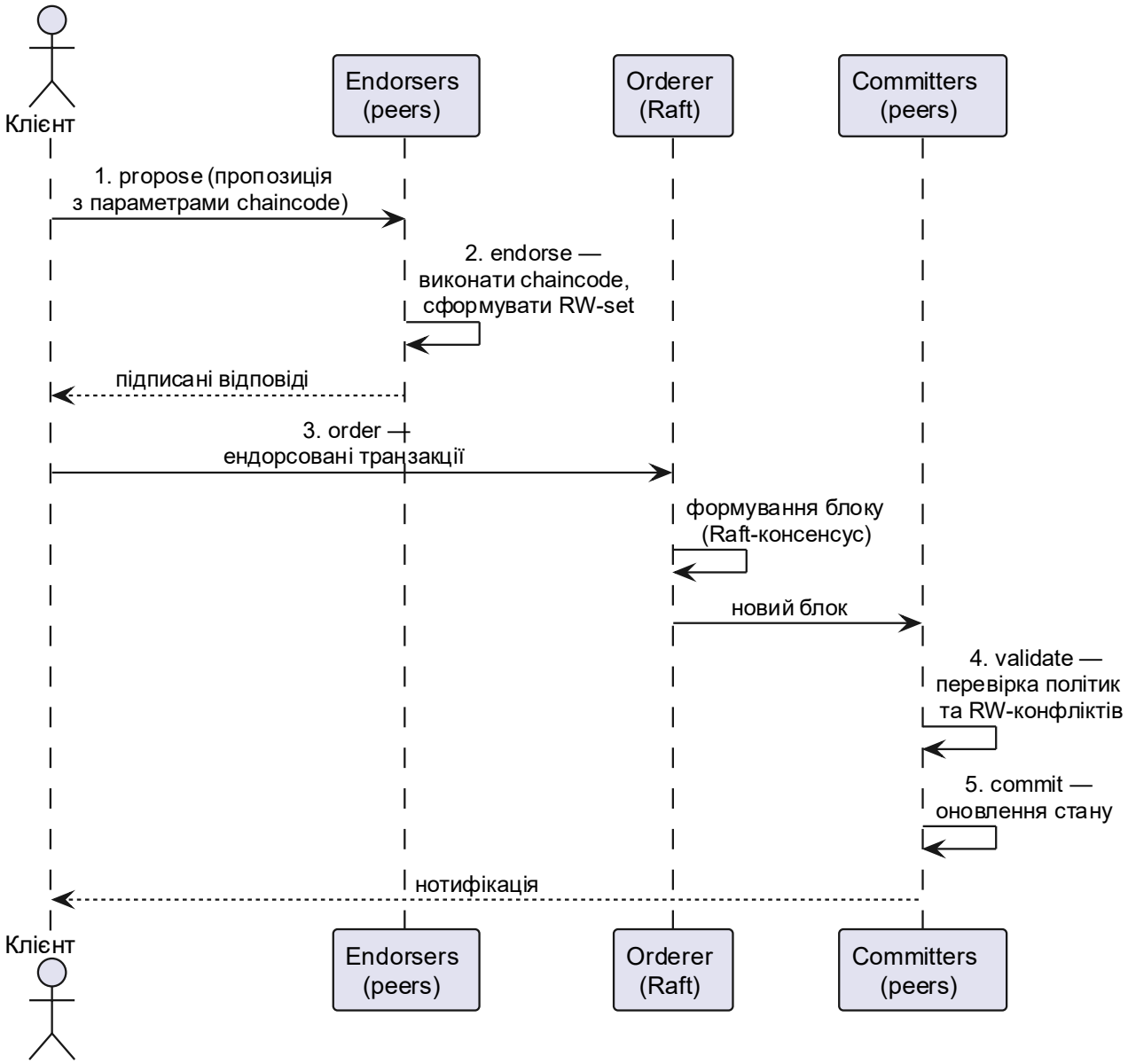

Рисунок 1.4 – Життєвий цикл транзакції в Hyperledger Fabric

Платформа побудована навколо концепції організацій, кожна з яких утримує власні вузли (peers), що підтримують реплікацію розподіленого реєстру. Бізнес-логіку реалізує chaincode – аналог смарт-контрактів Ethereum, що працює у контейнерному середовищі. Центральний компонент управління ідентифікацією – Membership Service Provider (MSP) – визначає правила автентифікації та авторизації учасників на основі

інфраструктури відкритих ключів (PKI), включаючи кореневі сертифікати, проміжні сертифікати та списки відкликаних сертифікатів для кожної організації. За формування порядку транзакцій та створення блоків відповідає служба впорядкування (orderer). Життєвий цикл транзакції в Hyperledger Fabric складається з п'яти етапів, наведених на рис. 1.4.

На першому етапі (propose) клієнт формує пропозицію транзакції та надсилає її обраним вузлам-ендорсерам. Далі, на етапі endorse, вузли-ендорсери незалежно виконують chaincode із запропонованими параметрами, формують набір читання-запису (read-write set) та підписують результат цифровим підписом. Клієнт збирає відповіді та передає їх службі впорядкування (етап order), яка формує блоки з транзакціями у погодженому порядку за допомогою консенсусу Raft. На етапі validate вузли-комітери отримують новий блок та перевіряють кожну транзакцію на відповідність політиці ендорсменту та відсутність конфліктів версій у наборах читання. Останній етап – commit: перевірені (валідні) транзакції додаються до блокчейну, а невалідні позначаються відповідним прапорцем і зберігаються у блоці для аудиту. Політики ендорсменту (endorsement policies) визначають, які організації та в якій комбінації мають підтвердити транзакцію – наприклад, AND('Org1.peer', 'Org2.peer') вимагає підтвердження від обох, тоді як OR приймає підтвердження від будь-якої.

Для IoT-середовищ Hyperledger Fabric має низку переваг. Завдяки каналам можна організувати приватні потоки даних між визначеними учасниками мережі, гарантуючи конфіденційність чутливої інформації. Канали функціонують як ізольовані підмережі з окремим реєстром, власним набором chaincode та незалежною політикою доступу – різні групи організацій обмінюються даними, не розкриваючи інформацію іншим учасникам. Система підтримує управління доступом на основі атрибутів (ABAC). Додатково в ній може бути інтегроване атрибутивне шифрування [42], що реалізує точковий контроль доступу до даних на основі криптографічних

атрибутів. Це особливо актуально для коаліційних сценаріїв із множинними організаціями та різними рівнями довіри, однак являє собою ресурсно важку операцію, особливо для пристроїв з обмеженими апаратними можливостями.

Hyperledger Fabric має помітні обмеження для розгортання на потужностях IoT-пристроїв. Серверні ресурсні вимоги платформи відчутно перевищують аналогічні для Ethereum PoA: кожна організація-учасник повинна підтримувати множину вузлів, які потребують серверного обладнання рівня хмарних віртуальних машин. Складність розгортання та конфігурації каналів, політик підтвердження транзакцій та chaincode-середовищ формує додаткові бар'єри для мобільних або автономних IoT-сценаріїв. Служба впорядкування (orderer), що зазвичай використовує консенсус Raft, може виявитися вузьким місцем при високому навантаженні, а недостатня кількість ресурсів або неоптимальна конфігурація параметрів здатна призвести до зниження пропускної здатності мережі. Тому пряме розгортання Hyperledger Fabric на рівні IoT-пристроїв є непрактичним – платформа більш доцільна для використання на вищих рівнях IoT-архітектури, як координаційний рівень для множини кластерів із підтримкою міжорганізаційної взаємодії.

1.2.4. DAG-блокчейн та альтернативи

Окремий клас блокчейн-платформ побудований на архітектурі спрямованого ациклічного графа (Directed Acyclic Graph, DAG), що принципово відрізняється від традиційного послідовного ланцюга блоків. У DAG-архітектурі нові транзакції додаються через підтвердження попередніх, що теоретично робить можливою паралельну обробку та усунення комісій [43]. Систематизація сучасних DAG-консенсусів, представлена у [46], показує, що паралельна агрегація транзакцій замість лінійного ланцюга є визначальною рисою цього класу протоколів. Найвідомішою реалізацією DAG для IoT є IOTA Tangle, архітектура якої придатна для використання пристроїв рівня ESP32 у ролі клієнта [44].

У Tangle кожна нова транзакція підтверджує дві попередні, обрані алгоритмом tip selection. Оригінальна реалізація – зважене випадкове блукання MCMC від початкової (генезисної) транзакції до кінцевих вершин, де ймовірність переходу визначається кумулятивною вагою транзакцій, де параметр α регулює глибину випадковості (рис. 1.5). Транзакції в основному потоці швидко отримують підтвердження, тоді як транзакції в бічних гілках можуть тривалий час залишатись непідтвердженими.

В наступній версії, Tangle 2.0, впроваджується децентралізоване голосування для консенсусу без централізованого координатора, що значно підвищує масштабованість мережі [45]. Використання нового протоколу імовірнісного консенсусу, механізмів протидій атакам та підтримки смарт-контрактів розширює придатність системи до складніших сценаріїв.

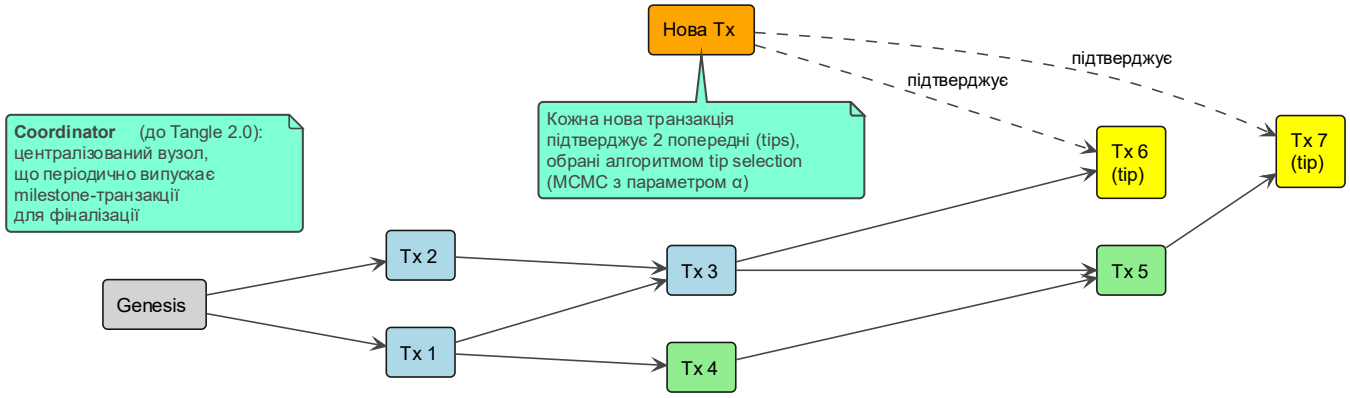

Рисунок 1.5 – Процес підтвердження транзакцій у класичній мережі IOTA

Переваги DAG (відсутність комісій, робота на малопотужних пристроях, теоретична масштабованість) не усувають критичних недоліків для IoT. DAG-структура дає непередбачувану тривалість підтвердження через відсутність чіткої послідовності перевірки історії, що неприйнятно для детерміністичних IoT-сценаріїв. Смарт-контракти на Tangle 2.0 потребують вирішення конфліктів між паралельними підграфами, що підвищує накладні витрати, а розгалужена DAG-структура ускладнює аудит при великих обсягах даних [46].

Інші альтернативи. IoTeX – IoT-орієнтована платформа з консенсусом Roll-DPoS: до 50 делегатів обираються голосуванням токенів із ротаційним випадковим відбором активної множини на епоху, що зменшує ризик змови [47]. Проект має клієнтську бібліотеку для малопотужних пристроїв. Однак розгортання повноцінної системи DPoS (модифікація PoS з механізмом делегування голосів) на малопотужних вузлах без серверних потужностей не є реалістичним. Waltonchain – вузькоспеціалізована IoT-платформа для ланцюгів поставок на базі RFID із власним протоколом консенсусу WPoC, прив'язана до компонентів виробництва Waltonchain, що робить неможливим її широке застосування [48].

Сучасні протоколи консенсусу на основі візантійської відмовостійкості (BFT) (Tendermint, Algorand) дають високу пропускну здатність на серверному обладнанні, але мають обмеження для IoT. Tendermint має квадратичну комунікаційну складність $O(n^2)$, що обмежує масштабованість. Algorand забезпечує лише ~90 TPS при високих системних вимогах [19]. Жоден із BFT-консенсусів не поєднує ресурсну економність, стійкість до мережевих затримок та придатність для embedded-вузлів.

1.2.5. Зведений порівняльний аналіз та існуючі блокчейн рішення

На основі проведеного аналізу окремих платформ у Таблиці 1.2 наведено зведене порівняння ключових характеристик блокчейн-платформ з точки зору їх придатності для IoT-середовищ.

Таблиця 1.2 – Порівняльний аналіз блокчейн-платформ для IoT

Характеристика	**Ethereum (PoS / PoA Clique)**	**Hyperledger Fabric**	**DAG IOTA Tangle 1 / 2**
Затримка	≥12 с, 13–19 хв фін. / 0,6–14,4 мс, ≥1 с фін.	>1 с, фін. залежить від політик	Непередбачувана
Пропускна здатність	<1000 / <8000 TPS	<10000 TPS	Залежить від топології

Характеристика	**Ethereum (PoS / PoA Clique)**	**Hyperledger Fabric**	**DAG IOTA Tangle 1 / 2**
Комісії	Так / Ні	Ні	Ні
Мін. вимоги до пристроїв	Серверні / Raspberry Pi	Серверні	Raspberry Pi
Смарт-контракти	Solidity, повноцінні	Chaincode (Go, Java, Node.js)	Обмежені, повільні
Конфіденційність	Обмежена / Повна (приватна)	Канали, приватні колекції	Обмежена
Адаптивність до IoT	Низька / Середня	Середня	Середня з обмеженнями
Вектор атаки на консенсус	контроль >50% стейку / підписантів	компрометація вузлів ендорсменту	контроль >30% вузлів / >50% впливу

Джерело: складено автором на основі [33], [35], [36], [37], [40], [43], [45]

Серед існуючих блокчейн-IoT рішень можна виокремити кілька характерних підходів, аналіз обмежень яких обгрунтовує потребу в новому гібридному підході. Система IBM Watson IoT інтегрує IoT-пристрої з Hyperledger Fabric через протокол MQTT, реалізуючи блокчейн-обробку даних у хмарі. Слабка ланка тут – централізоване підключення по MQTT, яке створює єдину точку відмови на транспортному рівні, хоча й зменшує безпекові ризики при роботі зі зібраними даними [49]. IOTA підтримує роботу з архітектурою Tangle на пристроях із обмеженими ресурсами, починаючи з ESP32 для клієнта та Raspberry Pi для повноцінного вузла, але має проблеми з детермінованістю підтвердження транзакцій та продуктивністю смарт-контрактів. IoTeX із бібліотекою для малопотужних пристроїв надає можливість підключатись до вузлів мережі, нагадуючи більш централізоване рішення, однак не дозволяє розгорнути повноцінну систему на одноплатних комп’ютерах.

Жодна з досліджених блокчейн-платформ не є універсальним рішенням для IoT. Ethereum PoS має обмеження затримки, комісій та вимог до застави. Ethereum PoA (Clique) демонструє найкращий баланс для IoT-рівня, але, як і інші консенсуси, не враховує гетерогенність. Hyperledger Fabric з розділеними каналами та міжорганізаційною взаємодією доцільний для координаційного рівня, але ресурсоємний для IoT-пристроїв. Потреба в багаторівневому гібридному підході, де на кожному рівні працює відповідна платформа, формує передумову для аналізу методів координації та вибору вузлів у підрозділі 1.3.

1.3. Критеріальні методи вибору вузлів у блокчейн-мережах Інтернету речей

Ефективність функціонування блокчейн-мережі IoT значною мірою визначається тим, за якими критеріями обираються вузли для виконання двох ключових ролей: валідатора, що формує блоки на консенсусному рівні, та, за потреби, обчислювального вузла, що виконує прикладні задачі. У підрозділі 1.2 встановлено, що існуючі системи не враховують параметри продуктивності при виборі валідаторів. Даний підрозділ систематизує існуючі підходи до критеріального вибору вузлів.

В традиційних централізованих системах використовуються пріоритетні черги обчислень [50]. Такі підходи спираються на централізований планувальник задач з глобальним механізмом розподілу наявних ресурсів. Цей архітектурний підхід суперечить децентралізованій моделі блокчейну: пріоритетна черга як єдина точка прийняття рішення створює централізовану залежність, що нівелює переваги децентралізованої координації. У контексті блокчейн-мереж IoT цей підхід може розглядатися переважно як основа для порівняння, а не як практично застосовне рішення.

1.3.1. Існуючі підходи до критеріального вибору вузлів

Проблема статичного порядку валідації не унікальна – більшість консенсусів із фіксованим колом підписантів мають аналогічне обмеження. Систематичний огляд [51] виділяє масштабованість, енергоефективність, стійкість до атак і пропускну здатність як основні критерії порівняння консенсусів; адаптивність вибору валідаторів до стану вузлів окремо не фігурує. Огляд [52] виділяє зосередження більшості досліджень на оптимізації трафіку та архітектурних рішеннях, тоді як динамічний вибір підписантів не розглядається.

Підхід на основі репутаційного консенсусу. Одним із найбільш розвинених напрямків адаптивного вибору вузлів є репутаційні моделі, що уможливлюють динамічне регулювання участі вузлів на основі їхньої поведінки та продуктивності. Автори з [53] запропонували ієрархічну репутаційну модель для транспортних мереж (VANET), яка вирішує проблему досягнення консенсусу в умовах, коли синхронна комунікація між вузлами не може бути гарантована. Модель динамічно розподіляє вузли по консенсусних групах на основі глобальної репутації та географічного розташування – вузли з низькою репутацією виключаються з процесу, що підвищує його ефективність. Глобальна репутація складається з двох компонентів: локальної, що відображає поведінку вузла при публікації подій (коректність та достовірність повідомлень), та консенсусної, що враховує активність вузла в голосуванні.

Достовірність подій оцінюється механізмом на основі баєсівського висновку, що враховує відстань вузла від місця події та його репутаційну вагу; завдяки цьому виявляються фальшиві повідомлення та знижується ймовірність змови між зловмисними вузлами. Реалізація консенсусу спирається на ієрархічний DAG-протокол HDGgraph: кожен вузол незалежно будує локальний DAG-реєстр та досягає локального консенсусу віртуальним голосуванням без потреби у синхронній комунікації, після чого результати передаються на придорожні модулі (RSU), які визначають глобальний статус та оновлюють репутаційні показники. Спрямоване

поширення повідомлень під керівництвом RSU усуває комунікаційну надмірність випадкового gossip-протоколу.

До недоліків можна віднести те, що архітектура жорстко прив'язана до інфраструктури транспортних мереж з наявністю стаціонарних придорожніх модулів та не оптимізована для малопотужних пристроїв.

Підхід на основі специфікацій. Дослідження [54] пропонує підхід до побудови енергоефективного консенсусу для IoT-мереж з використанням периферійних обчислень. Центральна ідея полягає в тому, що обчислювально інтенсивний процес створення нових блоків (майнінгу) переноситься на периферійні вузли мережі, а вибір конкретного вузла-майнера здійснюється на основі оцифрованих специфікацій обладнання кожного вузла – потужності процесора, обсягу доступної пам'яті та енергетичного бюджету. Експериментальна оцінка зафіксувала покращення енергоспоживання приблизно на 21% та утилізації пам'яті на 24% порівняно з класичним підходом. Попри привабливість результатів, модель базується на модифікації Proof of Work із єдиним обраним майнером і не адресує задачу динамічної ротації множини валідаторів у PoA-блокчейнах, де колективне підписання блоків потребує координації між кількома авторизованими вузлами.

Підхід на основі делегування. Автори з [55] досліджували підвищення продуктивності блокчейну для IoT через застосування ресурсоощадного (lightweight) консенсусу із делегуванням валідаторів за стейком (Delegated Proof of Stake, DPoS) у поєднанні з технікою шардингу. При тестовому навантаженні 500 транзакцій на секунду система досягала часу обробки приблизно 11 мс, а паралельне виконання через шарди давало можливість масштабувати мережу для ресурсно-обмежених IoT-пристроїв. DPoS, однак, передбачає обрання делегатів голосуванням власників токенів – для приватних IoT-мереж без токеноміки це складно реалізувати, а механізм шардингу додає викликів при синхронізації між шардами. Систематичний огляд в дослідженні [56], що охопив близько 85 робіт за період 2016–2023 років, виявив

характерну тенденцію: більшість існуючих ресурсоощадних консенсус-протоколів для IoT зосереджені на зниженні обчислювальних витрат та енергоспоживання на окремих вузлах, але не розглядають задачу адаптивного вибору валідаторів на основі реальних метрик продуктивності вузлів. Поняття ресурсоощадного консенсусу у літературі асоціюється переважно зі зменшенням розміру повідомлень та кількості раундів комунікації, а не з інтелектуальним вибором підписантів.

У Таблиці 1.3 систематизовано характеристики проаналізованих підходів за критеріями вибору вузлів.

Таблиця 1.3 – Порівняння критеріїв вибору вузлів у блокчейн-мережах

Категорія підходу	**Критерій вибору**	**Масштаб та середовище**	**Обмеження**
Поведінкова	глобальна репутація + географія розташування	4–50 вузлів (VANET)	Потребує RSU; обмежений DAG-структурою
Ресурсна	за специфікаціями обладнання	Лабораторний	Базується на PoW; один майнер
Делегування	на основі стейку	500 TPS (IoT)	Потребує токеноміки; складна синхр. шардів

Джерело: складено автором на основі [53], [54], [55]

Серед робіт, що адресують вибір вузлів безпосередньо в архітектурах блокчейн-периферійні обчислення, домінує підхід на основі одновимірної репутаційної метрики. Автори з [57] запропонували систему колаборативних периферійних обчислень, у якій репутація кожного вузла обчислюється з фактичного часу виконання попередньо призначених задач, а нові завдання розподіляються пропорційно до репутаційної ваги. Внутрішня суперечність такої схеми проявляється тоді, коли вузол

із високою репутацією, але тимчасово перевантажений, все одно отримує нові задачі та виходить за встановлений час виконання. Репутація такого вузла знижується, попри те, що апаратна спроможність залишається незмінною. Внаслідок відсутності метрик доступності ресурсів, дана модель не дозволяє відрізнити вузол із низькою продуктивністю від високопродуктивного вузла, який тимчасово завантажений. Систематизація літератури в [6] фіксує цю однорідну рису як спільну для більшості репутаційних рішень.

Паралельний клас становлять роботи присвячені питанням безпеки або перенесення задач на інші ресурси. Автори [58] зосереджуються на трасованості та автентифікації вузлів через смарт-контракти. Рішення про призначення задачі ухвалюється на статичних атрибутах вузла без реальної оцінки його стану. Автори моделі BeCome [59] натомість пропонують механізм міграції задач між периферійними серверами та хмарою, координований через блокчейн, із застосуванням генетичного алгоритму. Критеріями вибору виступають локалізація задачі та доступність каналу, проте модель не перевіряє актуальність метрик і черги задач на вузлі. При втраті зв'язку з вузлом задачі продовжують реєструватись за ним, оскільки останні відомі показники зчитуються як актуальні.

Аналіз виявляє чітку прогалину в існуючих дослідженнях. Як зазначено у підрозділі 1.2, консенсус PoA Clique, обраний як найбільш підходящий для мереж IoT, має три структурних обмеження: статичний порядок підписантів, надлишкову структуру блоків із повними транзакціями та відсутність адаптації до мережевих умов. Жоден з розглянутих підходів не застосовує мульти-критеріальний вибір валідаторів за метриками продуктивності в межах PoA-консенсусу. Репутаційні підходи демонструють ефективність динамічного групування вузлів, однак вони прив'язані до VANET-інфраструктури та замінюють блокчейн на DAG-структуру. Енергоефективні методи зосереджені на виборі одного вузла-майнера і не враховують специфіку PoA-консенсусу з множиною підписантів. Ресурсоощадні консенсуси на основі протокола

консенсусу DPoS потребують токеноміки, що може бути неприйнятно. Тобто мульти-метричну оцінку продуктивності вузлів ніхто з дослідників не поєднав із динамічним вибором валідаторів у PoA-консенсусі, що обгрунтовує потребу в новому адаптивному методі, який враховував би гетерогенність IoT-пристроїв безпосередньо на рівні вибору підписантів блоків. Аналіз робіт, присвячених поєднанню блокчейну з периферійними обчисленнями, формує актуальність розробки мультифакторного підходу до вибору периферійних вузлів.

1.3.2. Математичні підходи до критеріального вибору

Систематична побудова методів критеріального вибору вузлів вимагає теоретичної основи, що дозволяє формалізувати вибір з-поміж альтернатив за множиною конкуруючих критеріїв. У гетерогенному середовищі задача оптимального вибору обчислювального вузла формалізується як багатокритеріальне прийняття рішень (Multi-Criteria Decision Making, MCDM) – клас задач, у яких рішення приймається за одночасної оцінки кількох конкуруючих критеріїв [60]. Одновимірні моделі оперують лише репутацією; MCDM натомість працює з множиною факторів – доступністю обчислювальних ресурсів, завантаженістю процесора, обсягом вільної пам'яті, глибиною черги задач.

Серед класичних MCDM-методів найпоширенішими є метод аналізу ієрархій (AHP), метод упорядкування альтернатив за близькістю до ідеального розв’язку (TOPSIS), модель зваженої суми (WSM) та модель зваженого добутку (WPM). Автори з [61] продемонстрували їх ефективність для вибору бездротових технологій у IoT-мережах розумних будівель. Автори з [60] наголошують на ще одній рисі: алгоритмічні MCDM-методи дають повну трасованість критеріїв вибору. Для систем із обов'язковим аудитом рішень це має принципове значення.

Алгоритмічним MCDM-методам як альтернативу зазвичай протиставляють підходи на основі глибокого навчання з підкріпленням (Deep Reinforcement Learning, DRL). Теоретично DRL здатний адаптуватися до нових середовищ без перетасовки вагових коефіцієнтів. У реальних IoT-системах він наштовхується на помітні обмеження [62]. По-перше, навчання DRL-агентів – це тривалі тренувальні цикли та обчислювальні ресурси, яких на периферійних пристроях просто немає. По-друге, у непередбачених ситуаціях поведінка DRL-агентів буває нестабільною, а для критичних інфраструктур це неприйнятно. Алгоритмічний MCDM-підхід натомість поводиться передбачувано: критерії вибору чітко визначені, і адміністратор може відредагувати їх без перенавчання моделі.

Таблиця 1.4 – Порівняння багатокритеріальних методів для задач блокчейн-координації

Метод	**Складність для *n* вузлів з k критеріями**	**Компенсація між критеріями**	**Придатність до виконання смарт-контрактами**
WPM	O(n·k)	блокована (вето-ефект)	висока
WSM	O(n·k)	повна (адитивна)	висока
AHP	$O(k \cdot n^2)$	повна	середня
TOPSIS	O(n·k)	часткова	середня
DRL	залежить від архітектури	визначається моделлю.	низька

Джерело: складено автором на основі [60], [62], [63], [64]

В таблиці 1.4 наведено порівняння багатокритеріальних методів для задач блокчейн-координації. WPM обчислює добуток оцінок критеріїв для кожної альтернативи, тоді як WSM сумує оцінки, що дає принципово різні ефекти – низький результат за будь-яким одним критерієм призводить до різкого зниження оцінки вузла у WPM та компенсації оцінки за іншим критерієм у WSM. AHP має складнішу модель,

що включає визначення ваг пріоритетів за допомогою матриць попарних порівнянь. Низький результат за одним критерієм може бути повністю компенсований залежно від ваг, призначених експертом . TOPSIS ранжує альтернативи за евклідовою відстанню до ідеального та протилежного ідеальному розв'язку шляхом векторної нормалізації. Низький результат за одним критерієм може частково компенсуватися іншими критеріями. Серед усіх представлених в таблиці методів найкраще для використання в блокчейн-мережах підходять саме WPM та WSM через простоту реалізації в обмеженому середовищі виконання смарт-контракту та повну трасованість критеріїв вибору.

1.3.3. Додаткові критерії спеціалізованих мереж

Окрім метрик продуктивності та ресурсів, у спеціалізованих сценаріях розгортання (військових та промислових IoT-мережах) до критеріального вибору вузлів додається організаційний вимір: приналежність пристрою до підрозділу або організації, рівень довіри, ступінь автономності, відповідність політикам доступу. Цей вимір не покривається ні метриками ефективності консенсусу, ні ресурсними метриками, і потребує окремого розгляду.

Автори з [18] на прикладі військового IoT описують три жорсткі вимоги, що мережа повинна задовольнити одночасно: федеративне об'єднання пристроїв різних організацій (зокрема в коаліційних операціях NATO), контекстно-залежний обмін даними між різнорідними сенсорами та робота в умовах переривчастого зв'язку з централізованою інфраструктурою. У всіх цих сценаріях вибір вузла, який отримує конкретні дані або обчислювальну задачу, залежить не лише від його продуктивності або ресурсного стану, а й від того, до якого підрозділу він належить, який рівень допуску має, чи перебуває в активному стані. Технічні критерії у цих умовах накладаються на організаційні і саме сукупність обох визначає прийнятність вузла.

Подібна картина спостерігається у промислових IoT-мережах критичної інфраструктури: енергетика, нафтогазовий сектор, водопостачання, медичні системи. Тут діють регуляторні рамки (NIS2, HIPAA), що накладають жорсткі вимоги до розподілу доступу, аудиту операцій та розмежування відповідальності між операторами різних рівнів. Ресурсний стан вузла й тут є лише частковою відповіддю: без перевірки відповідності нормативним вимогам високопродуктивний вузол не може бути допущений до обробки конфіденційних або регульованих даних.

Таким чином на локальному рівні потрібен ресурсоощадний консенсус, придатний для ресурсно обмеженого обладнання, а для міжорганізаційної координації – корпоративна платформа з розвиненими механізмами автентифікації та конфіденційності.

1.4. Підходи до авторизації в блокчейн-системах Інтернету речей

Після розгляду критеріальних методів вибору вузлів у підрозділі 1.3 наступним аспектом, що визначає практичну придатність блокчейн-архітектури IoT, є модель авторизації пристроїв: без надійної прив'язки криптографічного ключа до апаратної ідентичності вузла динамічний вибір валідаторів та мультифакторне управління ресурсами втрачають довіру до вхідних даних.

Підрозділи 1.1–1.3 розкрили три аналітичні лінії: виклики IoT-середовищ, обмеження блокчейн-платформ і фрагментованість методів вибору вузлів. Поза межами попереднього огляду лишився ще один аспект, без якого розгляд блокчейн-орієнтованої IoT-архітектури неповний – ідентифікація та авторизація суб'єктів. У ресурсно-обмежених і гетерогенних мережах питання того, як пристрій отримує право на запис у розподілений журнал або на отримання обчислювального завдання, виявляється не менш вагомим за вибір протоколу консенсусу [65].

Транзакційна авторизація в блокчейні. Базовий механізм авторизації у публічних блокчейнах спирається на асиметричну криптографію. Пристрій або користувач володіє парою ключів ECDSA (у Ethereum – на основі еліптичної кривої secp256k1). Публічна адреса акаунту формується на основі останніх 20 байтів хешу Keccak256 публічного ключа [66]. Транзакція підписується приватним ключем. Мережеві вузли й смарт-контракти перевіряють підпис і звіряють відправника зі списками доступу, збереженими у стані контракту. У роботі [67] подібну модель застосовано в критичних IoT-мережах військового призначення (Internet of Battlefield Things, IoBT): сенсори підписують повідомлення, а смарт-контракт перевіряє підпис перед передачею даних командному центру. Переваги такої схеми – детермінованість, криптографічна стійкість, відсутність центрального арбітра. Однак у IoT виникають практичні обмеження. Адреса є лише псевдонімом, не прив'язаним до реальної ідентичності пристрою; генерація та перевірка ECDSA-підписів створюють відчутне навантаження на мікроконтролери; управління приватними ключами у незахищеному середовищі та їх ротація залишаються невирішеними задачами.

Централізовані альтернативи: PKI/X.509 та OAuth 2.0. Поза межами блокчейну застосовуються два традиційні підходи. Інфраструктура відкритих ключів (PKI) на базі сертифікатів X.509 будує ієрархію довіри з центральним засвідчувальним центром (CA), що підписує сертифікати кінцевих сутностей. Зрілість стандартів і наявність промислових рішень тут співіснують з низкою обмежень для IoT – розгортання CA на мільйони пристроїв, уразливість від компрометації CA, складність ведення актуальних списків відкликання на пристроях зі нерегулярним зв'язком. Токенова модель OAuth 2.0 і OpenID Connect широко використовується в IoT-cloud сценаріях [20]: пристрій автентифікується через централізованого провайдера (IdP) та отримує токен доступу з обмеженим терміном дії. Для підключених до хмари пристроїв ця схема зручна. Проте в автономних режимах і при тривалій відсутності

зв'язку з IdP вона впирається в одне критичне обмеження – якщо до сервера неможливо дістатися, локальна авторизація просто не відбувається.

Децентралізована ідентичність: DID і VC. Стандарт децентралізованої ідентичності (DID) консорціуму W3C [68] визначає формат ідентифікатора, який не залежить від жодного центрального реєстру. DID-документ містить публічні ключі, методи верифікації та сервісні точки. Отримання цього документу, а отже публічного ключа для перевірки, здійснюється відповідно до специфікації конкретного DID-методу. Наприклад, метод `did:ethr` використовує запис у блокчейні Ethereum, тоді як `did:key` взагалі обходиться без реєстру. Verifiable Credentials (VC) доповнює цю модель: це цифрове свідоцтво, видане довіреним видавцем (issuer) власникові (holder), яке перевіряється третьою стороною (verifier) без звернення до видавця. Формати JSON-LD або JWT дозволяють локальну криптографічну перевірку, а використання доказів без розкриття інформації (Zero-Knowledge Proofs) відкривають можливість селективного розкриття атрибутів [69]. Разом DID і VC складають основу парадигми Self-Sovereign Identity – пристрій контролює власні ідентифікатори й атрибути самостійно.

Ключова перевага DID/VC у контексті IoT – потенціал для динамічного додавання та управління доступом пристроїв у блокчейн-мережах. Новий пристрій миттєво отримує свідоцтво від довіреного видавця, одразу входить у мережу та починає взаємодію з іншими учасниками без попередньої реєстрації в централізованому реєстрі. Повноваження змінюються через оновлення або відкликання VC. Верифікація відбувається децентралізовано – на рівні смарт-контрактів або окремих вузлів. Такий механізм природно поєднується з атрибутною авторизацією (ABAC), де політика доступу перевіряє не конкретну адресу, а набір атрибутів із VC – роль пристрою, його сертифікований клас безпеки, приналежність до організаційного домену. Інтеграція з апаратними коренями довіри (PUF, TPM) дозволяє прив'язати DID до фізично невідтворюваної характеристики пристрою, що

підвищує стійкість до клонування та фізичної компрометації. Цей напрям розвинений для атрибутного шифрування у військовому IoT [67]. Обмеження підходу теж відомі. Розмір DID-документів і VC-сертифікатів накладає вимоги на пам'ять мікроконтролерів, наявність реєстру верифікації (зазвичай блокчейн) лишається необхідною, а екосистема окремих DID-методів усе ще фрагментована.

Порівняння чотирьох розглянутих класів підходів до авторизації зведено у таблиці 1.5.

Таблиця 1.5 – Порівняння підходів до авторизації в блокчейн-орієнтованих IoT-системах

Підхід	**Переваги**	**Обмеження в контексті IoT**
Транзакційна ECDSA-авторизація	Детермінованість; криптостійкість; відсутність центрального арбітра	Відв’язаність від ідентичності пристрою; навантаження ECDSA на мікроконтролери; складне управління приватними ключами
PKI / X.509	Зрілі стандарти; широка промислова підтримка	Централізований CA як точка відмови; складні списки відкликання; обмежене масштабування
OAuth 2.0 / OpenID Connect	Зручність для хмарних IoT сценаріїв; обмежені в часі токени доступу	Залежність від централізованого IdP; непридатність за відсутності зв'язку
DID + VC (W3C, SSI)	Децентралізована автономна ідентифікація; динамічне додавання пристроїв та	Розмір DID-документів; потреба в реєстрі верифікації; фрагментованість DID-методів

Підхід	Переваги	Обмеження в контексті IoT
	управління доступом; інтеграція з PUF/TPM	

Джерело: складено автором на основі [3], [20], [66], [67], [69], [70]

Таким чином не можна виокремити ідеальне рішення авторизації для Інтернету речей. Натомість необхідно обирати кращий варіант в залежності від потреб конкретної задачі.

1.5. Постановка задачі дослідження

У підрозділах 1.1–1.4 проаналізовано стан безпеки та масштабованості IoT-мереж, порівняно блокчейн-платформи, досліджено методи координації вузлів, систематизовано підходи до авторизації та обгрунтовано необхідність гібридного підходу до побудови архітектури.

Кожне з наявних рішень сильне у конкретній ролі й обмежене поза нею. Приватна мережа Ethereum з Clique економна для ресурсно-обмеженого кластеру, але її статичний порядок підписантів не реагує на гетерогенність вузлів. Hyperledger Fabric дає розвинуті засоби міжорганізаційної взаємодії ціною ресурсоємного розгортання. DAG-протоколи пропонують високу пропускну здатність за рахунок відмови від класичного блочного ланцюга, проте детермінованість підтвердження і стійкість при малих навантаженнях залишаються відкритими. Окремі аспекти задачі адресовані адаптивними методами координації, такими як: репутаційні механізми, DRL-підходи, MCDM. У площині авторизації: PKI та OAuth спираються на зрілі стандарти ціною централізації, DID/VC пропонують децентралізовану автономну ідентифікацію з перспективою динамічного управління доступом. В результаті постає необхідність формування гібридної блокчейн-архітектури та оптимізації роботи її

складових до поточних викликів мереж Інтернету речей. На основі цього аналізу сформульовано об'єкт, предмет, мету та задачі дисертаційного дослідження.

Об'єкт дослідження – блокчейн-орієнтовані комп'ютерні мережі пристроїв Інтернету речей.

Предмет дослідження – методи і засоби побудови мереж IoT на основі гібридної блокчейн-архітектури з адаптивним консенсусом та мультифакторним управлінням вузлами.

Мета дослідження – підвищення ефективності функціонування блокчейн-орієнтованих мереж IoT шляхом розроблення гібридної багаторівневої архітектури, адаптивного консенсус-протоколу з динамічним вибором валідаторів та методу мультифакторного управління обчислювальними вузлами.

Для досягнення мети необхідно вирішити такі **задачі**:

1. Провести порівняльний аналіз блокчейн-платформ, методів координації вузлів та підходів до авторизації для мереж IoT; обгрунтувати доцільність гібридного підходу, мету, задачі та методику досліджень.

2. Розробити модель гібридної багаторівневої блокчейн-архітектури для мереж IoT з кластерною організацією та обгрунтуванням вибору типу блокчейну для кожного рівня.

3. Розробити метод адаптивного консенсусу Proof of Indicators з динамічним вибором валідаторів та оптимізацією трафіку; здійснити програмну реалізацію та провести експериментальне дослідження.

4. Удосконалити модель мультифакторного вибору вузлів для блокчейн-координованих периферійних обчислень IoT; здійснити програмну реалізацію та провести експериментальне дослідження.

Задача 1 має аналітичний характер і реалізується у підрозділах 1.1–1.4 даного розділу. Наукову задачу 2 розв'язано у Розділі 2, де обгрунтовано вибір тришарової кластерної організації з диференційованим типом блокчейну на кожному рівні. У

Розділі 3 розв'язується задача 3 – формалізовано метод адаптивного консенсусу із удосконаленнями, спрямованими на покращення стабільності та ефективності мережі. Четверта задача розв’язується у Розділі 4, де удосконалено модель мультифакторного вибору вузлів для периферійних обчислень що фіналізує комплексний гібридний підхід використання блокчейну в IoT.

Логіка дослідження побудована так, що аналітична задача формує підгрунтя для постановки наукових задач, які, у свою чергу, визначають вимоги до науково-прикладної реалізації та верифікації. Сукупність результатів розв'язання цих задач спрямована на досягнення мети дослідження.

РОЗДІЛ 2. МОДЕЛЬ ГІБРИДНОЇ БАГАТОРІВНЕВОЇ БЛОКЧЕЙН-АРХІТЕКТУРИ ДЛЯ МЕРЕЖ ІНТЕРНЕТУ РЕЧЕЙ

У цьому розділі розроблено модель гібридної багаторівневої блокчейн-архітектури, орієнтованої на мережі Інтернету речей. Результати порівняльного аналізу з Розділу 1 дозволили систематизувати вимоги до архітектури та обгрунтувати тришарову кластерну організацію, де для кожного рівня обирається окремий тип блокчейну. Перший рівень – рівень сенсорів – використовує симетричне шифрування, адаптоване під апаратно обмежені пристрої. На другому рівні функціонує локальна блокчейн-мережа з підтримкою смарт-контрактів на базі PoA Clique. Третій рівень – зовнішня блокчейн-мережа – відповідає за агрегацію транзакцій. Окремо показано, як архітектура адаптується до спеціалізованих сценаріїв, зокрема військового та цивільного IoT. Розроблена модель виступає інфраструктурною основою для методів, описаних у Розділах 3 та 4.

2.1. Принципи побудови гібридної блокчейн-архітектури для Інтернету речей

Порівняльний аналіз Розділу 1 засвідчив, що жодна з існуючих блокчейн-платформ самостійно не покриває вимог IoT-середовищ. Це обгрунтовує потребу в гібридній архітектурі з диференційованим застосуванням платформ і методів координації на різних рівнях. Метою розділу є розроблення такої моделі на основі систематизованих вимог.

2.1.1. Формування вимог до блокчейн-архітектури мережі Інтернету речей

На основі аналізу, проведеного у підрозділах 1.1–1.3, систематизовано шість вимог, яким повинна відповідати блокчейн-архітектура IoT-мережі.

Перша і базова вимога – **масштабованість**, тобто здатність системи підтримувати зростання кількості підключених пристроїв, обсягу даних та кількості кластерів без деградації продуктивності. Прогнозоване зростання кількості IoT-

пристроїв вимагає від архітектури горизонтального масштабування через додавання нових кластерів – без перебудови всієї мережі. Централізовані архітектури, як зазначалося у підрозділі 1.1, стикаються з критичними обмеженнями пропускної здатності при збільшенні кількості пристроїв, тому масштабованість має досягатися саме через розподілену організацію.

Друга вимога – **децентралізація**: відсутність єдиної точки відмови (Single Point of Failure) та стійкість системи до відмов окремих компонентів. Аналіз у підрозділі 1.1 засвідчив, що централізовані IoT-системи вразливі до каскадних збоїв, компрометації центрального сервера та залежності від зовнішніх провайдерів. Архітектура має гарантувати функціонування окремих частин мережі навіть при втраті зв'язку з іншими компонентами.

Третя вимога стосується **підтримки гетерогенних пристроїв** – від малопотужних мікроконтролерів класу ATmega328P (16 МГц, 2 КБ ОЗП) та ESP32 до одноплатних комп'ютерів Raspberry Pi і серверного обладнання. У підрозділі 1.2 було показано, що різні блокчейн-платформи мають суттєво відмінні ресурсні вимоги: Ethereum Clique здатний функціонувати на Raspberry Pi, тоді як Hyperledger Fabric потребує типових корпоративних конфігурацій. Отже, архітектура повинна передбачати диференційоване використання технологій залежно від обчислювальних можливостей пристроїв кожного рівня.

Четверта вимога – **автономність кластерів**: здатність локальної IoT-мережі функціонувати незалежно при тимчасовій втраті зовнішнього зв'язку. Дана вимога особливо актуальна для спеціалізованих мереж – військових IoT-систем, промислових мереж у віддалених локаціях, мереж реагування на надзвичайні ситуації. Кластери повинні підтримувати повний цикл збору, обробки та зберігання даних в автономному режимі, а при відновленні з'єднання – виконувати синхронізацію.

П'ята вимога – **безпека**, що охоплює три аспекти: конфіденційність (захист даних від несанкціонованого доступу), цілісність (гарантія незмінності даних) та

автентифікацію (підтвердження ідентичності пристроїв і вузлів). Аналіз загроз у підрозділі 1.1 показав різноманітність векторів атак на IoT-мережі – від атак маршрутизації до ботнет-атак, – що вимагає багаторівневого підходу до безпеки з використанням криптографічних методів, адаптованих до обчислювальних можливостей кожного рівня.

Шоста вимога – **адаптивність**, тобто здатність системи реагувати на зміни стану мережі: зміну продуктивності вузлів, зміну топології, додавання або видалення пристроїв. У підрозділі 1.3 було показано, що статичні механізми координації (зокрема, порядок підписантів у Clique) не враховують динамічну природу IoT-мереж, що призводить до нераціонального використання ресурсів та зменшення стабільності мережі. Адаптивність має бути закладена як на рівні консенсусного протоколу, так і на рівні управління обчислювальними ресурсами.

2.1.2. Використання тришарового підходу для побудови гібридної блокчейн-архітектури

Систематизовані вимоги засвідчують, що архітектура IoT-мережі повинна одночасно підтримувати пристрої з принципово різними обчислювальними можливостями, гарантувати як локальну автономність, так і глобальну координацію, і при цьому використовувати різні рівні криптографічного захисту. Ці характеристики природно відповідають багаторівневій організації, де кожен рівень відповідає за специфічний клас пристроїв та задач.

На рис 2.1 запропоновано тришарову архітектуру, що складається з рівня сенсорів та малопотужних пристроїв (L_s), рівня локальної блокчейн-мережі (L_b) та рівня зовнішньої блокчейн-мережі (L_e). Вибір саме трьох рівнів обумовлений трьома класами пристроїв та відповідних задач, які не можуть бути ефективно обслуговані однією технологічною платформою.

Рівень сенсорів (L_s) охоплює малопотужні пристрої – мікроконтролери ATmega328P, ESP32, датчики – що характеризуються обмеженими обчислювальними

ресурсами (від 16 МГц та 2–8 КБ ОЗП) і не здатні виконувати повноцінні блокчейн-операції. На даному рівні конфіденційність даних реалізується за допомогою симетричного шифрування, без участі у консенсусі. Рівень локальної блокчейн-мережі (L_b) об'єднує блокчейн-вузли кластера (Raspberry Pi, одноплатні комп'ютери, шлюзи), які формують приватну блокчейн-мережу з підтримкою смарт-контрактів на базі PoA Clique. Цей рівень відповідає за координацію, зберігання даних та виконання бізнес-логіки в межах кластера. Рівень зовнішньої блокчейн-мережі (L_e) уможливлює міжкластерну координацію, агрегацію даних та глобальну узгодженість через потужнішу блокчейн-платформу, вибір якої визначається конкретним сценарієм застосування.

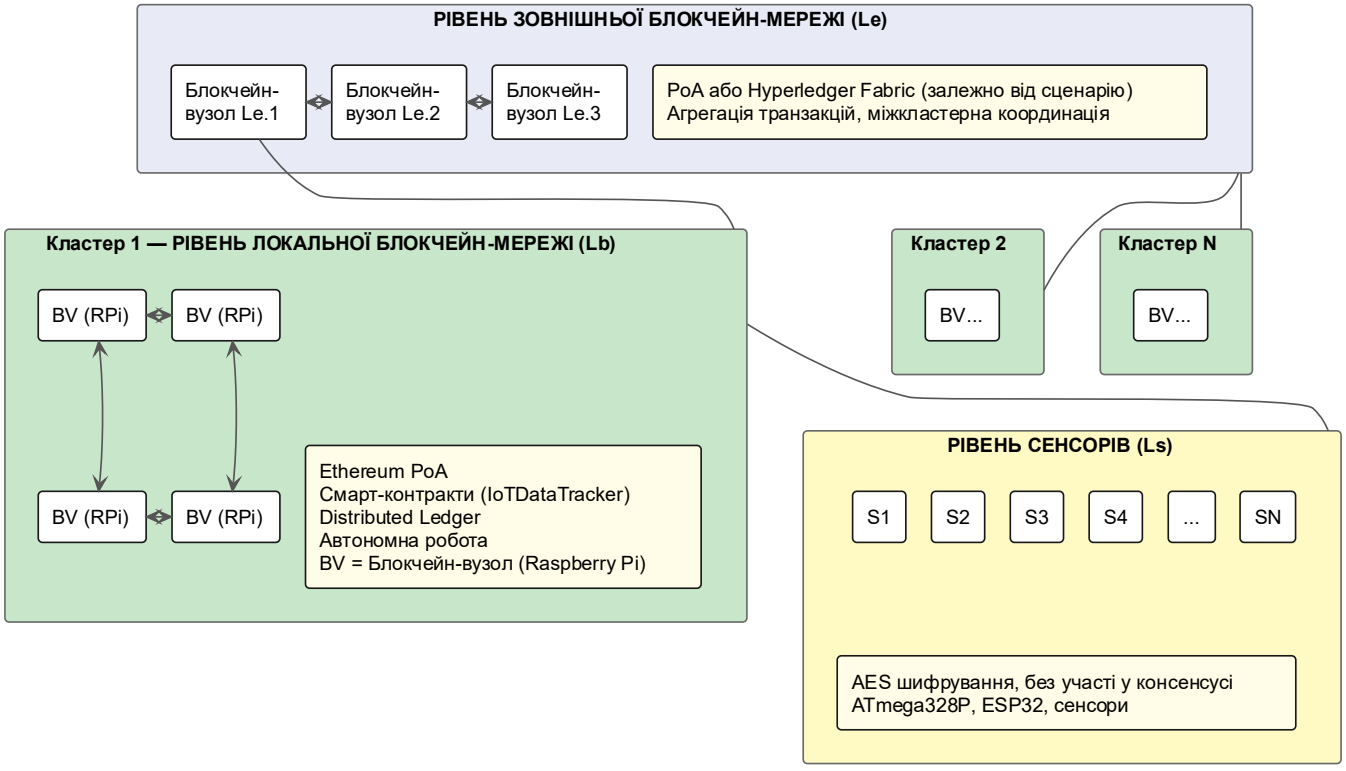

Рисунок 2.1 – Загальна схема тришарової гібридної блокчейн-архітектури для мереж Інтернету речей

Порівняння запропонованої тришарової архітектури з існуючими IoT-блокчейн рішеннями, проаналізованими у підрозділі 1.2, засвідчує її переваги. IBM Watson IoT використовує централізоване MQTT-підключення до хмарного Hyperledger, що

створює єдину точку відмови та не гарантує автономності. IOTA Tangle дозволяє працювати на малопотужних пристроях, проте має непередбачувану тривалість підтвердження транзакцій та підвищені ресурсні вимоги до виконання смарт контрактів. IoTeX має обмеження з розгортання повноцінної мережі на малопотужних пристроях. На відміну від перелічених рішень, запропонована архітектура диференційовано використовує оптимальні технології на кожному рівні та покриває всі шість систематизованих вимог.

Формальне визначення архітектури. Запропоновану тришарову архітектуру можна формалізувати у вигляді впорядкованої сукупності елементів (кортежу):

$$A = \langle L_s, L_b, L_e, C, I \rangle, \tag{2.1}$$

де L_s – множина пристроїв рівня сенсорів, L_b – множина блокчейн-вузлів кластерного рівня, L_e – множина вузлів зовнішньої блокчейн-мережі, $C = \{c_1, c_2, \dots, c_m\}$ – множина IoT-кластерів, та $I = \{I_{s\to b}, I_{b\to e}\}$ – множина інтерфейсів взаємодії між рівнями.

Кожен кластер $c_i \in C$ визначається так:

$$c_i = \langle S_i, B_i, G_i \rangle, \tag{2.2}$$

де $S_i \subseteq L_s$ – множина сенсорів кластера, $B_i \subseteq L_b$ – множина блокчейн-вузлів кластера, та $G_i \in B_i$ – шлюзовий вузол, що підтримує зв'язок із зовнішньою мережею. Інтерфейс $I_{s\to b}$ визначає протоколи передачі даних від сенсорів до блокчейн-вузлів (шифровані канали MQTT/CoAP із AES), а інтерфейс $I_{b\to e}$ – протоколи міжкластерної комунікації (агрегація транзакцій або пряма блокчейн-взаємодія).

Вимога автономності кластерів формалізується так: кожен кластер c_i повинен підтримувати повний операційний цикл незалежно від стану зовнішнього зв'язку $\mathrm{conn}(G_i, L_e)$:

$$O(c_i) = \{collect, process, store, validate\}. \tag{2.3}$$

При $\text{conn}(G_i, L_e) = 0$ (відсутність зв'язку) кластер працює в автономному режимі з буферизацією транзакцій T_i^{buffer}, які відправляються пакетом при відновленні з'єднання:

$$\text{conn}(G_i, L_e) = 1 \Rightarrow \text{sync}(T_i^{buffer}, L_e). \quad (2.4)$$

Таким чином, запропонована формальна модель визначає архітектуру як сукупність трьох рівнів із чіткими інтерфейсами взаємодії та вбудованою підтримкою автономності кластерів, що відповідає всім шести систематизованим вимогам. Наступні підрозділи містять детальний опис кожного рівня архітектури.

2.2. Рівень сенсорів та малопотужних пристроїв

Рівень L_s тришарової архітектури охоплює сенсори та малопотужні пристрої, що збирають дані з фізичного середовища і передають їх блокчейн-мережі кластера. Жорсткі ресурсні обмеження роблять неможливою їхню участь у блокчейн-консенсусі чи виконанні повноцінних криптографічних протоколів. У підрозділі обгрунтовано вибір криптографічних механізмів рівня L_s, проведено порівняння бібліотек та описано інтерфейс взаємодії з кластерним рівнем L_b.

2.2.1. Характеристика пристроїв рівня сенсорів

Пристрої рівня сенсорів можна умовно поділити на дві категорії. Перша – мікроконтролери з мінімальними ресурсами, наприклад ATmega328P (Arduino Uno): 16 МГц, 2 КБ ОЗП, 32 КБ ПЗП [3]. Друга – потужніші пристрої на кшталт ESP32 (160–240 МГц, 520 КБ ОЗП, апаратне прискорення AES [71]), здатні до асиметричних операцій. Жоден пристрій рівня сенсорів не може функціонувати як повноцінний блокчейн-вузол: консенсус, розподілений реєстр і смарт-контракти потребують ОЗП понад можливості навіть другої категорії. Тому архітектура відокремлює рівень сенсорів, а конфіденційність передачі гарантують криптографічні механізми.

2.2.2. Механізм динамічних таблиць доступу (BDT)

Для захисту конфіденційності на рівні сенсорів пропонується використати механізм Blockchain Dynamic Table (BDT) із симетричним шифруванням [72]. BDT – це динамічна таблиця, що зберігається на блокчейн-вузлі кластера і містить зашифровані дані від підключених сенсорів. Кожен запис складається з ідентифікатора сенсора, зашифрованого значення вимірювання, мітки часу та хешу для контролю цілісності.

Чому обрано саме симетричне шифрування? По-перше, симетричні алгоритми вимагають менших обчислювальних ресурсів, ніж асиметричні, по-друге, симетричне шифрування споживає менше оперативної пам'яті – реалізація AES займає приблизно 450 байтів ОЗП, тоді як бібліотека ECDSA потребує понад 1,5 КБ, а це 75% від загального обсягу ОЗП ATmega328P. AES-128 дає достатній рівень криптографічної стійкості для IoT-телеметрії в типових прикладних сценаріях [73]. AES-128 потребує 2^{128} операцій для повного перебору ключа. Проте, як зазначалося у підрозділі 1.1, потенційна загроза квантових обчислень вимагає розглянути збільшення частоти ротації ключів або перехід до постквантових алгоритмів у довгостроковій перспективі. Також за необхідності можна використовувати AES-256 на пристроях рівня ATmega328P.

Вибір конкретної криптографічної бібліотеки для реалізації BDT на малопотужних пристроях – інженерне завдання, яке потребує порівняння доступних рішень за використанням пам'яті, підтримкою потрібних алгоритмів та сумісністю з цільовими мікроконтролерами. Результати такого аналізу для чотирьох основних бібліотек наведено у таблиці 2.1.

Таблиця 2.1 – Порівняння криптографічних бібліотек для малопотужних IoT-пристроїв

Бібліотека	**Пам'ять (ПЗП / ОЗП)**	**Алгоритми (AES / ECC)**	**Сумісність (ATmega328P / ESP32)**
tinyECC + protectedAES	~13,7 КБ / 419 Б	✓ / ✓	✓ / ✓
PSACrypto	~18,8 КБ / 1940 Б	✓ / ✓	✓(недоцільно) / ✓
Crypto (Arduino)	~15,5 КБ / 1449 Б	✓ / ✓	✓ / ✓
MbedTLS (повна)	~170 КБ / 38 КБ	✓ / ✓	✗ / ✓

Джерело: складено автором на основі власних вимірів [74], [75], [76], [77], [78]

Для пристроїв першої категорії (ATmega328P) практично придатні дві бібліотеки: tinyECC + protectedAES (~13,7 КБ ПЗП, 419 байт ОЗП) та Crypto (Arduino) (~15,5 КБ ПЗП, 1449 байт ОЗП). Обидві підтримують AES та ECC, однак tinyECC + protectedAES споживає менше ОЗП – і при загальному обсязі ОЗП ATmega328P у 2 КБ це має вирішальне значення; підтримку ECC позначено як обмежену, оскільки пам'яті не вистачає для складних операцій підпису. Бібліотека PSACrypto недоцільно використовувати з ATmega328P через ресурсні обмеження. Для пристроїв другої категорії (ESP32) усі чотири бібліотеки технічно сумісні, але оптимальними виглядають tinyECC + protectedAES або PSACrypto – обидві підтримують AES і ECC при помірному використанні ресурсів. MbedTLS не працює на мікроконтролерах серії AVR (ATmega328P) і є надмірно ресурсоємною для більшості IoT-сценаріїв.

2.2.3. Протокол обміну ключами

Симетричне шифрування вимагає попереднього узгодження спільного ключа між сенсором і блокчейн-вузлом, тому потрібен безпечний механізм обміну ключами.

Запропоновано двофазний протокол ініціалізації. Перша фаза – початкова конфігурація: блокчейн-вузол генерує пару асиметричних ключів і оприлюднює відкритий ключ; сенсор генерує випадковий симетричний ключ, шифрує його відкритим ключем блокчейн-вузла і відправляє зашифрований пакет; блокчейн-вузол розшифровує AES-ключ за допомогою свого приватного ключа та зберігає його у захищеному сховищі. Друга фаза – робочий режим: уся подальша комунікація між сенсором і блокчейн-вузлом відбувається виключно із використанням симетричного шифрування, що мінімізує навантаження на ресурси при кожній передачі даних.

Для пристроїв першої категорії (ATmega328P), для яких ресурсно важкі операції ECC, обмін ключами здійснюється на етапі фізичної конфігурації пристрою (pre-shared key) – AES-ключ встановлюється програмно під час розгортання мережі. Цей підхід менш гнучкий, але він дає достатній рівень безпеки для сценаріїв зі стабільною конфігурацією мережі. Для підвищення стійкості передбачено механізм періодичної ротації ключів, ініційований блокчейн-вузлом через захищений канал.

2.2.4. Інтерфейс взаємодії з рівнем блокчейн-мережі

Зашифровані дані від сенсорів передаються до блокчейн-вузлів кластера через інтерфейс $I_{s \to b}$, який підтримує два протоколи прикладного рівня: MQTT і CoAP. Вибір протоколу залежить від характеристик мережі та обмежень пристроїв. MQTT працює на базі TCP і підходить для сценаріїв із постійним з'єднанням, де потрібна гарантована доставка повідомлень. CoAP побудований на UDP і краще підходить для мереж із високими втратами пакетів та обмеженою пропускною здатністю – він потребує менше мережевих ресурсів [3]. В обох випадках корисне навантаження повідомлення містить зашифрований AES-блок: ідентифікатор сенсора, значення вимірювання, мітку часу і контрольну суму.

Після отримання зашифрованого повідомлення блокчейн-вузол розшифровує його збереженим AES-ключем, перевіряє контрольну суму для підтвердження цілісності й записує дані до BDT. Потім ці дані можуть бути включені до транзакції

для запису у блокчейн кластерного рівня – це гарантує їх незмінність та прозорість для всіх учасників кластера.

Організація рівня сенсорів, описана вище, дозволяє захистити конфіденційність даних на пристроях з мінімальними обчислювальними ресурсами завдяки криптографічним механізмам, адаптованим до обмежень кожного класу пристроїв. Деталі функціонування блокчейн-мережі кластерного рівня, куди передаються дані від сенсорів, описано у наступному підрозділі.

2.3. Рівень локальної блокчейн-мережі (IoT-кластер)

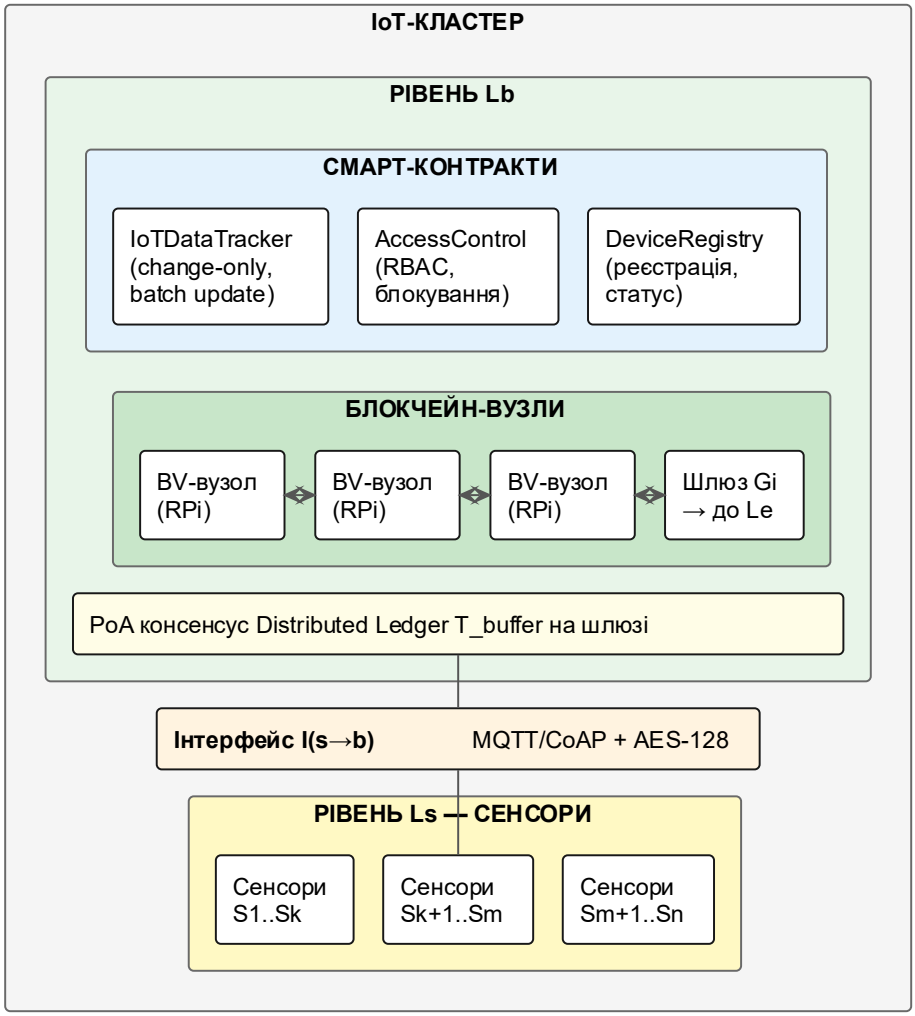

Рисунок 2.2 – Архітектура IoT-кластера із взаємодією між рівнями

Рівень L_b– приватна блокчейн-мережа IoT-кластера, що відповідає за координацію, захищене зберігання даних та виконання смарт-контрактів. Вузли рівня мають достатньо ресурсів для участі у консенсусі та підтримки розподіленого реєстру. Підрозділ описує організацію кластера, вибір консенсусу, смарт-контракти, модель зберігання та автономну роботу.

2.3.1. Організація IoT-кластера

Кластер $c_i = \langle S_i, B_i, G_i \rangle$ складається з множини сенсорів S_i, множини блокчейн-вузлів B_i та шлюзового вузла $G_i \in B_i$. Блокчейн-вузли кластера типово реалізуються на одноплатних комп'ютерах Raspberry Pi або аналогічних пристроях. Це можуть бути також шлюзові пристрої та локальні сервери – головне, щоб ресурсів вистачало для виконання клієнта Go-Ethereum (Geth). Кожен блокчейн-вузол зберігає повну копію розподіленого реєстру кластера і бере участь у консенсусному протоколі. Кількість блокчейн-вузлів у кластері варто підбирати в залежності від мережевих умов та потужності пристроїв – це компроміс між стійкістю до відмов (більше вузлів дає надійність) та навантаженням на мережеву комунікацію.

Шлюзовий вузол G_i додатково відіграє роль інтерфейсу між кластером та зовнішньою блокчейн-мережею (L_e). Окрім стандартних функцій блокчейн-вузла, він агрегує транзакції для передачі на зовнішній рівень, буферизує дані при відсутності зовнішнього зв'язку та маршрутизує міжкластерні запити. Шлюзовий вузол, як правило, реалізується на потужнішому обладнанні порівняно з рядовими блокчейн-вузлами кластера. Додаткова опція – дублювання його функцій на декількох вузлах заради відмовостійкості.

2.3.2. Вибір консенсусу та реалізація смарт-логіки для кластерного рівня

Порівняльний аналіз у підрозділі 1.2 показав, що Proof of Authority (PoA) з реалізацією Clique найкраще підходить як базовий консенсусний протокол для IoT-кластерів. Обгрунтування спирається на кілька переваг. Clique гарантує низьку

затримку підтвердження транзакцій залежно від конфігурації періоду генерування блоків, що важливо для IoT-сценаріїв із потребою в оперативній обробці даних. Пропускна здатність може досягати 8000 TPS при оптимальній конфігурації, а це з великим запасом перекриває потреби типового IoT-кластера [36]. PoA/Clique здатний функціонувати на пристроях класу Raspberry Pi, що відповідає вимозі підтримки обмежених пристроїв на кластерному рівні. Нарешті, відсутність комісій за транзакції усуває економічний бар'єр для масової IoT-телеметрії.

Підтримка смарт-контрактів на кластерному рівні принципово важлива, бо дозволяє реалізувати програмовану бізнес-логіку без централізованого управління. У контексті IoT-кластера смарт-контракти виконують кілька функцій: реєстрацію та управління пристроями (ідентифікація, права доступу, статус), валідацію даних (перевірка допустимих діапазонів значень, виявлення аномалій), агрегацію даних (обчислення статистик, формування звітів), а також управління правилами безпеки – блокування підозрілих пристроїв, ескалація інцидентів.

У рамках запропонованої архітектури розроблено смарт-контракт IoTDataTracker, який реалізує оптимізований підхід до зберігання IoT-телеметрії у блокчейні. Ключова особливість IoTDataTracker – механізм change-only storage: нове значення записується до блокчейну лише тоді, коли воно відрізняється від попереднього запису для даного сенсора. Це суттєво зменшує кількість транзакцій у сценаріях, де показники змінюються повільно (температура, вологість, рівень освітлення). Контракт підтримує багатосенсорну конфігурацію, тобто дає змогу реєструвати та відстежувати множину сенсорів з різними типами даних. Передбачена також підтримка групових оновлень – передача значень від кількох сенсорів в одній транзакції, що знижує мережеве навантаження [5].

Стратегія change-only storage ефективно зменшує обсяг транзакцій, проте має побічний ефект: вона не дозволяє відрізнити ситуацію «значення сенсора стабільне» від «сенсор вийшов з ладу і передає останнє збережене значення». У критичних

застосуваннях (медичний IoT, промислова автоматизація) така невизначеність повинна бути вирішена періодичним примусовим оновленням.

2.3.3. Розподілене зберігання даних

Модель зберігання на кластерному рівні побудована за принципом розділення компонентів, що виконуються на рівні блокчейну та поза ним. Повинні зберігатися у блокчейні кластера метадані (ідентифікатори пристроїв, мітки часу, типи вимірювань), агреговані значення через IoTDataTracker, хеші великих об'єктів даних, транзакції управління (реєстрація пристроїв, зміна прав доступу). Поза блокчейном зберігаються повні набори сирих даних, зображення та відеопотоки, логи діагностики, журнали подій великого обсягу.

Для зберігання поза блокчейном можлива інтеграція з розподіленою файловою системою IPFS (InterPlanetary File System), що підтримує адресацію контенту за хешем [79]. При використанні IPFS хеш файлу записується як транзакція у блокчейн кластера – це гарантує верифіковану прив'язку даних поза блокчейном до записів в блокчейні. Дана модель дає можливість зберігати великі обсяги даних без перевантаження блокчейну і при цьому зберігати гарантії цілісності через криптографічну прив'язку хешів.

2.3.4. Автономна робота кластера

Автономність IoT-кластера є однією з переваг запропонованої архітектури перед централізованими рішеннями. Автономна робота реалізована на кількох рівнях. На рівні блокчейну: повна копія розподіленого реєстру зберігається на кожному вузлі кластера, консенсусний протокол працює незалежно від зовнішнього зв'язку, смарт-контракти виконуються на основі локального стану блокчейну. На рівні даних: сенсори продовжують передавати дані до блокчейн-вузлів кластера через інтерфейс $I_{s\to b}$, IoTDataTracker продовжує записувати дані в блокчейн. На рівні міжкластерної

комунікації: шлюзовий вузол буферизує транзакції, призначені для зовнішньої мережі, у локальному сховищі T_i^{buffer}.

Інтеграція з IPFS для зберігання великих об'єктів (зображення, відео, діагностичні логи) передбачає, що блокчейн-вузли або мають локальну службу IPFS, або доступ до зовнішнього IPFS-провайдера. В обох випадках виникає питання доступності: IPFS не гарантує збереження даних без явного запиту закріплення, і якщо жоден вузол кластера не підтримує цього для певного хешу, дані можуть бути втрачені під час процесу збірки сміття. При автономній роботі кластера ця проблема загострюється – дані поза блокчейном, на які посилаються буферизовані транзакції, мають зберігатися локально до відновлення зовнішнього з'єднання.

Потреба в автономності особливо гостра для військових IoT-мереж, де зв'язок із зовнішньою інфраструктурою може бути переривчастим або цілеспрямовано порушеним. У промислових сценаріях автономність дає стійкість до відмов зовнішніх каналів зв'язку, що типово для віддалених промислових об'єктів – нафтових платформ, шахт, сільськогосподарських угідь. Механізм синхронізації при відновленні зовнішнього зв'язку описано у підрозділі 2.4.

Рівень локальної блокчейн-мережі дає IoT-кластеру повноцінне функціонування з підтримкою смарт-контрактів, оптимізованим зберіганням даних та здатністю працювати автономно. Координація між кластерами через зовнішню блокчейн-мережу описана у наступному підрозділі.

2.4. Рівень зовнішньої блокчейн-мережі

Верхній рівень (L_e) тришарової архітектури відповідає за координацію між IoT-кластерами, агрегацію даних для глобальної аналітики і підтримку узгодженого стану розподіленої системи. Якщо кластерний рівень обслуговує обмежену кількість вузлів у межах одного кластера, то зовнішня мережа поєднує географічно віддалені кластери

– причому деякі з них можуть мати різний рівень довіри. Даний підрозділ розкриває механізми міжкластерної комунікації, агрегацію транзакцій, стійкість при переривчастому зв'язку та обгрунтовує вибір типу блокчейну для зовнішнього рівня.

2.4.1. Призначення та функції зовнішнього рівня

Зовнішня мережа виконує чотири функції: координацію між географічно розподіленими кластерами (обмін та синхронізацію стану, наприклад між районами розумного міста); глобальну узгодженість через спільний реєстр з агрегованими записами як єдине джерело істини для аналітики; аудит і звітність через незмінний журнал подій з верифікацією ланцюжка від сенсора до звіту; управління ідентичністю – у сценаріях із Hyperledger Fabric зовнішній рівень надає DID та Verifiable Credentials для пристроїв і організацій [4].

Взаємодія між кластерами. Міжкластерна комунікація реалізована через шлюзи G_i, що утримують подвійне з'єднання: з локальною PoA-мережею кластера (повноцінний вузол) та із зовнішньою мережею (клієнт або вузол). Потік даних: смарт-контракт кластерного рівня формує агреговану транзакцію, шлюз підписує її та передає зовнішньому рівню, де вузли валідують і включають її до глобального блокчейну; інші кластери отримують оновлений стан через синхронізацію. Оскільки кластер генерує тисячі транзакцій на годину, індивідуальна передача створила б неприйнятне навантаження, тому запропоновано механізм агрегації.

2.4.2. Агрегація транзакцій та механізм повтору

Для стиснення міжкластерних даних застосовано підхід Zk-Rollups (Zero-Knowledge Rollups) [80] – метод, за якого множина транзакцій об'єднується в одну з криптографічним доказом (zero-knowledge proof), що дозволяє верифікувати коректність агрегації без перевірки індивідуальних транзакцій.

Механізм працює так: шлюз G_i накопичує транзакції $\{t_1, \dots, t_k\}$ протягом інтервалу або до порогу кількості, обчислює агрегований стан (середнє, мінімум,

максимум, кількість відхилень замість індивідуальних значень), генерує криптографічний доказ коректності та передає на зовнішній рівень одну агреговану транзакцію t_{agg} з доказом. Якщо кластер генерує 1000 транзакцій за інтервал, передається лише одна – обсяг даних скорочується на порядки, а верифікація коректності лишається доступною через доказ.

Автономність кластерів вимагає обробки ситуацій тимчасової недоступності зовнішнього зв'язку. Запропоновано механізм повторних спроб на основі буферизації. При $\text{conn}(G_i, L_e) = 1$ агреговані транзакції передаються одразу; при $\text{conn}(G_i, L_e) = 0$ зберігаються у буфері T_i^{buffer} з дотриманням хронологічного порядку і криптографічних доказів. Після відновлення зв'язку буферизовані транзакції передаються пакетом зі збереженням порядку. Для обмеженої пропускної здатності передбачено пріоритизацію: критичні транзакції (інциденти безпеки, відмови пристроїв) – першочергово, рутинна телеметрія – по мірі звільнення каналу. Механізм повторних спроб особливо важливий для військового IoT, де зовнішній зв'язок може бути відсутнім годинами або днями.

2.4.3. Вибір типу блокчейну для зовнішнього рівня

На відміну від кластерного рівня, де вибір PoA є однозначним через ресурсні обмеження й потребу в низькій затримці, для зовнішнього рівня тип блокчейну залежить від конкретного сценарію застосування. Архітектура передбачає два основних варіанти: PoA для цільових IoT-мереж та Hyperledger Fabric для корпоративних/військових сценаріїв. Зведений порівняльний аналіз наведено у таблиці 2.2.

Таблиця 2.2 – Порівняння варіантів блокчейну для зовнішнього рівня архітектури

Критерій	**PoA (Ethereum)**	**Hyperledger Fabric**
Затримка	від 1 с	від 1 с
TPS	До 8000	До 10000
Приватність каналів	Не підтримується	Підтримується
Організаційна структура	Однорідна мережа	Організації, канали, ACL
DID / Verifiable Credentials	Не вбудовано	Підтримується
Складність розгортання	Низька	Висока
Ресурсні вимоги	Помірні (Raspberry Pi)	Високі (серверне обладнання)
Цільовий сценарій	Розумне місто, промисловість	Enterprise, військове IoT

Джерело: складено автором на основі результатів порівняльного аналізу (підрозділ 1.2) та [4]

Військові IoT-мережі – клас систем, для яких визначальними є безпека, автономність та стійкість до компрометації. Типові елементи – рої БПЛА, розподілені сенсорні масиви, мобільні командні пункти, системи автоматизованого управління. Специфічні вимоги: організаційна ізоляція, стійкість до тривалої відсутності зв'язку (години чи дні), децентралізована ідентифікація, захист від фізичного захоплення вузлів. Тому тут обрано Hyperledger Fabric на L_e та PoA на L_b. Fabric підтримує організаційну структуру, канали приватності та контроль доступу: кожен підрозділ чи командний пункт – окрема організація з власними сертифікатами, політиками та каналами, що реалізує принцип мінімального доступу. Надійний обмін

повідомленнями забезпечує Raft як служба впорядкування з гарантованою доставкою, стійкістю до відмов та високою пропускною здатністю. На L_b рої БПЛА та мобільні масиви утворюють автономні PoA-кластери; retry-механізм буферизує дані та передає їх при відновленні зв'язку з командним пунктом через Fabric. Використання децентралізованої ідентифікації W3C DID та технологій фізичних неклонованих функцій додатково доповнюють безпекову модель.

Цивільні IoT-мережі масового застосування характеризуються великою кількістю пристроїв, однорідним рівнем довіри та потребою у простоті розгортання. Типові сценарії – розумні лічильники (електрика, вода, газ), промислові сенсори (температура, тиск, вібрація), екологічний моніторинг, управління рухом та розумне освітлення. Єдиний PoA-стек на обох рівнях (L_b та L_e) зменшує складність розгортання, спрощує міжкластерну комунікацію та має помірні ресурсні вимоги. Зовнішній рівень може інтегруватися з хмарними сервісами для аналітики та візуалізації: агреговані через Zk-Rollups дані надходять як до глобального блокчейну, так і до платформ машинного навчання та статистичного аналізу.

Аналіз показує, що вибір типу блокчейну для зовнішнього рівня визначається двома факторами: потребою в приватності каналів і організаційною складністю системи (таблиця 2.3).

Таблиця 2.3 – Рекомендації щодо вибору блокчейн-платформи за сценарієм IoT

Сценарій IoT	**Рекомендована платформа**	**Ключовий критерій**
Військове IoT	Hyperledger Fabric + PoA-кластер + DID/VC	ізоляція даних, автономність кластерів
Розумне місто	Ethereum PoA, опц. Rollups	масштабованість, регуляторна звітність

Сценарій IoT	**Рекомендована платформа**	**Ключовий критерій**
Промислове IoT / Логістика	Hyperledger Fabric або Hyperledger Besu	трасованість, крос-організаційність
Медичне IoT	Hyperledger Fabric + DID/VC	приватність пацієнтів, відповідність GDPR
Масовий цивільний IoT (smart home)	Ethereum PoA	простота розгортання, низька вартість підтримки

Отже, зовнішній рівень архітектури реалізує координацію між IoT-кластерами, забезпечує стійкість при переривчастому зв'язку з кластерним рівнем та дає можливість гнучкого вибору типу блокчейну під вимоги конкретного сценарію розгортання.

2.5. Порівняльний аналіз запропонованої архітектури з існуючими рішеннями

Для оцінки переваг запропонованого рішення проведено зіставлення з IoT-блокчейн архітектурами: IBM Watson IoT (на базі Hyperledger Fabric), та IOTA Tangle за вимогами, сформованими в підрозділі 2.1.1 (таблиця 2.4).

Таблиця 2.4 – Порівняння запропонованої архітектури з існуючими IoT-блокчейн рішеннями

Критерій	**Запропонована архітектура**	**IBM Watson IoT (Hyperledger Fabric)**	**IOTA Tangle**
Рівень масштабованості	високий	високий	високий

Критерій	**Запропонована архітектура**	**IBM Watson IoT (Hyperledger Fabric)**	**IOTA Tangle**
Рівень децентралізації	високий	середній (в рамках хмарного сервісу)	середній (використання координатора)
Мінімальні вимоги до вузла	міні-ПК	сервер	міні-ПК
Рівень автономності	високий (виконання сценарію в локальному кластері + Zk-Rollups)	низький	низький
Рівень безпеки (по вектору атаки на консенсус)	високий	високий	середній
Рівень адаптивності	середній (обмеження Clique)	високий	високий

Недоліком запропонованого рішення є середній рівень адаптивності, що вирішується в розділі 3.

Таким чином, тришарова кластерна організація з диференційованим вибором типу блокчейну для кожного рівня дає змогу побудувати масштабовану, децентралізовану та безпечну мережу IoT з підтримкою гетерогенних пристроїв і можливістю автономної роботи кластерів.

2.6. Висновки

Потребу в гібридному підході обгрунтовано сформованими вимогами: масштабованість, децентралізація, підтримка гетерогенних пристроїв, автономність кластерів, безпека, адаптивність.

Розроблена модель гібридної багаторівневої блокчейн-архітектури для мереж IoT дозволяє поєднати гетерогенні пристрої в повнофункціональну децентралізовану систему та адаптується під різні сценарії використання.

Порівняльний аналіз з існуючими рішеннями підтверджує, що запропонований варіант є єдиним серед досліджених рішень, що одночасно підтримує пристрої класу одноплатних комп’ютерів, забезпечує високу пропускну здатність кластерного рівня та повну автономність кластерів.

РОЗДІЛ 3. РОЗРОБЛЕННЯ МЕТОДУ АДАПТИВНОГО КОНСЕНСУСУ PROOF OF INDICATORS

Розділ присвячено розробленню методу адаптивного консенсусу Proof of Indicators (PoI) для блокчейн-мереж IoT. Спершу проаналізовано внутрішні механізми протоколу Clique як базового консенсусу та виявлено його структурні обмеження у гетерогенних IoT-середовищах. На основі цього аналізу запропоновано три взаємодоповнюючих удосконалення: техніку пропагації легких блоків для зменшення мережевого трафіку, метрики ефективності з механізмом динамічного вибору валідаторів та конфігуровну затримку старту no-turn вузлів для зменшення ймовірності форків. Також проведено порівняльний аналіз PoI з іншими консенсус-протоколами, описано програмну реалізацію мовою Go на базі Go-Ethereum та представлено результати експериментального дослідження у Docker-середовищі з 15 вузлами.

3.1. Аналіз протоколу Clique як базового консенсусу

Порівняльний аналіз, проведений у Розділі 1, показав, що консенсусний протокол Proof of Authority з реалізацією Clique (EIP-225) найкраще підходить як базовий механізм консенсусу для IoT-кластерів у рамках запропонованої тришарової архітектури (Розділ 2). Вибір обгрунтовується низькою латентністю, високою пропускною здатністю, здатністю працювати на пристроях класу Raspberry Pi та відсутністю комісій за транзакції. Проте детальніший розгляд внутрішніх механізмів Clique виявляє структурні обмеження, які знижують його ефективність у гетерогенних IoT-мережах. Даний підрозділ містить формальний аналіз протоколу Clique – це необхідна передумова для обгрунтування трьох удосконалень, описаних у підрозділах 3.2, 3.3 та 3.5.

Принцип роботи протоколу Clique. Clique, специфікований в EIP-225, реалізує механізм Proof of Authority, в якому право генерації блоків належить обмеженій множині авторизованих підписантів (signers). На відміну від Proof of Stake, де валідатори обираються залежно від розміру стейку, або Proof of Work, де блоки генеруються через обчислювальне змагання, підписанти у Clique визначаються адміністративно і зберігаються у заголовках блоків у полі extraData [35]. Протокол розрізняє три стани вузла відносно конкретного блоку:

- черговий (in-turn) вузол– єдиний основний вузол, який створює новий блок;
- позачергові (no-turn) вузли – усі інші вузли, окрім in-turn та заборонених. Якщо in-turn вузол не справляється з формуванням блоку, no-turn вузли продукують блоки з випадковим часовим інтервалом, аби уникнути колізій (race conditions);
- заборонені (forbidden) вузли – вузли, які досягли порогу підписання. Вони здатні приймати та перевіряти блоки, але не продукувати їх.

Вибір номеру in-turn вузла для генерації наступного блоку відбувається за детерміністичною формулою:

$$k = (h + 1) mod n, \tag{3.1}$$

де n – загальна кількість авторизованих підписантів, а h – порядковий номер поточного блоку (або висота ланцюга).

Поріг підписання для заборонених вузлів у вихідному коді визначається як:

$$l = \left\lfloor \frac{n}{2} + 1 \right\rfloor. \tag{3.2}$$

В результаті підписант має зачекати l блоків після підписання поточного блоку, щоб мати змогу підписати знову. З цього випливає, що робоча система містить a активних та f заборонених підписантів:

$$a = \left\lfloor \frac{n+1}{2} \right\rfloor, f = \left\lfloor \frac{n-1}{2} \right\rfloor. \tag{3.3}$$

Порядок вузлів задається лексикографічним сортуванням їхніх Ethereum-адрес за байтовим представленням. Черговий вузол генерує блок без додаткової затримки. Блок від чергового вузла отримує більшу вагу (difficulty = 2) порівняно з блоком від

позачергового вузла (difficulty = 1), що дає пріоритет черговим блокам при розв'язанні конфліктів.

Позачерговий вузол має право генерувати блок, але із додатковою випадковою затримкою (wiggle time). У стандартній реалізації Clique вона визначається як:

$$t_w = rand\left(t_i, \left(\frac{n}{2} + 1\right) \cdot \omega\right), \tag{3.4}$$

де t_i = 0мс, ω = 500мс, n - загальна кількість авторизованих підписантів.

Ймовірність виникнення розгалуження ланцюга блоків у Clique. Форк (тимчасове розгалуження ланцюга блоків) виникає, коли два чи більше вузлів одночасно генерують конкурентні блоки для однієї позиції в ланцюжку. У випадку Clique це трапляється, коли позачерговий вузол генерує та поширює блок раніше, ніж мережа отримає блок від чергового вузла. Ймовірність форку для одного вузла оцінюється за формулою:

$$p_f = \frac{t_b + t_v}{t_w}, \tag{3.5}$$

де t_b –час поширення блоку мережею, t_v – валідаційний період.

Залежність ймовірності форків від кількості вузлів має нелінійний характер – зі зростанням n показник p_f зростає непропорційно швидко. Причини дві. По-перше, більша кількість no-turn вузлів збільшує шанси того, що один із них отримає малу випадкову затримку і встигне згенерувати конкурентний блок до поширення чергового блоку мережею. По-друге, збільшення кількості хопів між найвіддаленішими вузлами збільшує t_b, що додатково збільшує p_f. В IoT-мережах із частково-зв’язною mesh-топологією та мережевими затримками ці ефекти проявляються особливо гостро.

Обгрунтування напрямків удосконалення. Три виявлених структурних обмеження Clique в підрозділі 1.2.2 та більш глибоке розуміння в підрозділі 3.1 визначають три напрямки удосконалення, які разом формують метод адаптивного консенсусу Proof of Indicators (PoI). Надлишкова структура блоків потребує оптимізації розміру – заміни повних транзакцій на їхні хеші (легкі блоки, підрозділ

3.2). Статичний порядок підписантів разом з однаковим ставленням до гетерогенних вузлів вимагає динамічного механізму вибору валідаторів на основі метрик ефективності (підрозділ 3.3). Недостатній діапазон wiggle time потребує конфігурованої затримки старту позачергових вузлів (підрозділ 3.5).

3.2. Оптимізація розміру блоків: компактні блоки (Light Blocks)

Перше удосконалення протоколу Clique в рамках методу PoI – техніка пропагації компактних блоків (Light Blocks), яка спрямована на скорочення обсягу мережевого трафіку під час поширення нових блоків між вузлами IoT-кластера. Так як стандартний блок Ethereum містить повне тіло кожної транзакції, і в приватних IoT-мережах це надлишково, оскільки дані транзакції вже присутні у мемпулах вузлів-отримувачів. Запропонована техніка спирається на підхід, аналогічний Compact Block Relay у мережі Bitcoin (BIP 152) [81], але адаптований під специфіку Ethereum-сумісних IoT-мереж.

Концепція та обгрунтування компактних блоків. Суть компактних блоків – у заміні повних тіл транзакцій на їхні криптографічні хеші (transaction hashes). Це стає можливим через специфіку приватних IoT-мереж: транзакції перед включенням до блоку вже поширюються мережею через стандартний одноранговий (peer-to-peer, P2P) механізм мережі Ethereum.

Принцип BIP 152 у мережі Bitcoin підтвердив ефективність такого підходу: замість передачі повних транзакцій у блоці передаються короткі 6-байтні ідентифікатори, за якими отримувач реконструює повний блок з локального мемпулу. У запропонованій реалізації для Ethereum використовуються повні 32-байтні хеші Keccak-256 транзакцій – це гарантує криптографічно надійну ідентифікацію без ризику колізій, які притаманні коротким ідентифікаторам.

Порівняння структури стандартного та легкого блоку. Стандартний блок Ethereum складається з заголовку (header) фіксованого розміру та масиву повних тіл транзакцій (transactions). Компактний блок зберігає ідентичний заголовок, проте замість масиву тіл транзакцій містить масив їхніх хешів (TxHashes). Розмір хешу транзакції – 32 байти (Keccak-256), тоді як типова IoT-транзакція пакетного оновлення через IoTDataTracker має розмір від 500 до 2000 байтів залежно від кількості сенсорів у пакеті.

Для кількісної оцінки зменшення розміру розглянемо типовий IoT-блок із k транзакціями пакету оновлень із середнім розміром s_{tx} байтів. Розмір даних транзакцій у стандартному блоці дорівнює $V_{full} = k \cdot s_{tx}$, а у легкому – $V_{light} = k \cdot 32$. Відношення зменшення:

$$\rho = 1 - \frac{V_{light}}{V_{full}} = 1 - \frac{32}{s_{tx}}. \quad (3.6)$$

При $s_{tx} = 500$ байтів відношення зменшення для однієї транзакції становить $\rho = 0{,}936$ (93,6%), а при $s_{tx} = 2000$ байтів – $\rho = 0{,}984$ (98,4%). Для типового IoT-блоку з 10–50 транзакціями пакету оновлень зменшення розміру варіюється від 93,6% до 98,4% залежно від розміру транзакцій. Заголовок блоку передається без змін, тому фактичне зменшення загального розміру блоку є дещо меншим і визначається співвідношенням розміру заголовку та даних транзакцій.

Розширення мережевого протоколу ETH68. Для підтримки компактних блоків запропоновано розширити протокол ETH68 трьома новими типами повідомлень, що реалізують повний цикл відправки, запиту та отримання відсутніх транзакцій [5].

Перший тип повідомлення – NewLightBlockMsg – призначений для розсилки легкого блоку мережею. Замість стандартного NewBlockMsg, що містить повний блок, підписант відправляє NewLightBlockMsg із заголовком блоку та масивом хешів транзакцій. Це повідомлення надсилається всім підключеним вузлам (peers) аналогічно стандартному механізму поширення блоків.

Другий тип – GetMissingTxsMsg – запит відсутніх транзакцій. Якщо вузол-отримувач не знаходить деякі транзакції у своєму мемпулі за їхніми хешами, він формує GetMissingTxsMsg із масивом хешів відсутніх транзакцій і відправляє його піру, від якого отримав легкий блок.

Третій – MissingTxsMsg – відповідь на запит. Вузол, що отримав GetMissingTxsMsg, знаходить запитані транзакції у своєму мемпулі або блокчейні і повертає їх повні тіла у відповідь.

Алгоритм реконструкції повного блоку. Процес обробки легкого блоку вузлом-отримувачем формалізовано у вигляді Алгоритму 3.1.

Алгоритм 3.1. Реконструкція повного блоку з легкого блоку

```
Input: LightBlock LB = (Header, TxHashes[])
Output: FullBlock FB або Fallback

missing ← ∅
reconstructed ← []
for each hash h in LB.TxHashes do
    tx ← Mempool.Lookup(h)
    if tx ≠ null then
        reconstructed.Append(tx)
    else
        missing.Add(h)
    end if
end for

if |missing| = 0 then
    FB ← AssembleBlock(LB.Header, reconstructed)
    return FB              // Повна реконструкція
end if

if |missing| / |LB.TxHashes| ≤ 0.5 then
    response ← SendRequest(GetMissingTxsMsg, missing)
    for each tx in response do
        reconstructed.Insert(tx, correctPosition)
    end for
    FB ← AssembleBlock(LB.Header, reconstructed)
    return FB              // Реконструкція з запитом
end if

FB ← RequestFullBlock(peer)     // Fallback
```

27: return FB

Алгоритм має три фази. Перша (рядки 1–10) – пошук кожної транзакції за хешем у локальному мемпулі. Якщо всі транзакції знайдено, реконструкція повного блоку відбувається без додаткових мережевих запитів. На другій фазі (рядки 12–24), коли частина транзакцій відсутня, але їх не більше 50% від загальної кількості, виконується запит відсутніх транзакцій через GetMissingTxsMsg. Поріг 50% обраний як компроміс: при меншій кількості відсутніх транзакцій вигода від компактного блоку зберігається, бо обсяг запитаних даних менший за повний блок. Третя фаза (рядки 26–27) – резервний механізм (fallback): якщо більше половини транзакцій відсутні, виконується запит повного стандартного блоку за стандартним механізмом Ethereum, що забезпечує зворотну сумісність та коректну роботу навіть у ситуаціях десинхронізації мемпулів.

Зменшення розміру блоків дає кумулятивний ефект на мережеву ефективність. Кожен блок поширюється всім вузлам кластера через механізм gossip, і зменшення розміру одного блоку скорочує загальний обсяг мережевого трафіку пропорційно кількості вузлів.

3.3. Метрики ефективності та механізм динамічного вибору валідаторів

Центральним удосконаленням протоколу Clique в межах методу PoI є механізм динамічного вибору валідаторів, побудований на комплексній метриці ефективності. Запропонований механізм відходить від принципу статичного порядку підписантів: пріоритет доступу до генерації блоків встановлюється через багатовимірну оцінку фактичної продуктивності кожного вузла.

Базова оцінка ефективності вузла. Перший компонент комплексної метрики – базова оцінка, що показує частку блоків, згенерованих конкретним вузлом протягом визначеного вікна спостереження. Для вузла j вона обчислюється так:

$$S_{base}(j) = K_b \cdot \frac{\Delta_b(j)}{\Delta_h}, \quad (3.7)$$

де K_b – коефіцієнт базової оцінки (нормалізуючий множник), $\Delta_b(j)$ – кількість блоків, згенерованих вузлом j протягом вікна спостереження, та Δ_h – загальна кількість блоків у вікні. Відношення $\frac{\Delta_b(j)}{\Delta_h}$ відображає відносну продуктивність вузла: той, хто генерує блоки частіше, отримує вищу базову оцінку. K_b масштабує результат до визначеного діапазону і дає можливість налаштовувати вагу базової оцінки відносно штрафних компонентів.

Штрафні функції. Базова оцінка фіксує позитивний внесок вузла. Проте для повної картини потрібно також враховувати якість роботи – і тут вступають у дію три типи штрафних функцій, кожна з яких формалізує окремий прояв неефективності. Штрафний підхід дозволяє диференціювати вузли не лише за продуктивністю, а й за надійністю їхньої участі в консенсусному протоколі.

Першим є штраф за пропуск черги. Він фіксує ситуацію, коли вузол, будучи черговим підписантом (in-turn), не генерує блок протягом відведеного часу (block period). Пропуск змушує позачерговий вузол компенсувати відсутність чергового блоку, що підвищує затримку і збільшує ймовірність розгалуження ланцюга:

$$P_{missed}(j) = K_m \cdot \frac{M(j)}{\Delta_h}, \quad (3.8)$$

де K_m – коефіцієнт штрафу за пропуск, $M(j)$ – кількість пропусків in-turn черги вузлом j протягом вікна спостереження, а Δ_h – загальна кількість блоків у вікні. Нормалізація на Δ_h робить штраф порівнянним незалежно від розміру вікна спостереження та кількості вузлів.

Другий тип – штраф за затримку генерації блоків. Навіть вузол, який не пропускає черги, може систематично генерувати блоки повільніше за очікуваний час через обмежені обчислювальні ресурси. Така повільність збільшує загальну затримку мережі. Штраф має вигляд:

$$P_{timing}(j) = \min(D_b(j) \cdot K_t, K_{t\_max}), \quad (3.9)$$

де $D_b(j)$ – середня затримка генерації блоків вузлом j відносно очікуваного часу, K_t – коефіцієнт штрафу за затримку, а K_{t_max} – максимальне значення штрафу. Функція min з верхньою межею K_{t_max} запобігає надмірному штрафуванню вузлів із тимчасовими проблемами продуктивності – це підтримує стабільність системи оцінювання.

Третій тип – штраф за створення розгалужень ланцюга (форків). Форки особливо небажані в IoT-мережах: тимчасова неузгодженість стану між вузлами може спричинити некоректну обробку даних. Штраф визначається як:

$$P_{fork}(j) = \min\left(K_f \cdot \frac{F(j)}{\Delta_b(j)}, K_{f\_max}\right), \quad (3.10)$$

де $F(j)$ – кількість форків, спричинених блоками вузла j протягом вікна спостереження, $\Delta_b(j)$ – кількість блоків, згенерованих вузлом j, K_f – коефіцієнт штрафу за форки, а K_{f_max} – максимальне значення штрафу. Тут нормалізація виконана на $\Delta_b(j)$ (а не на Δ_h), що дозволяє оцінити частку форків серед блоків саме даного вузла.

Загальна метрика ефективності. Інтегральна оцінка ефективності вузла j визначається різницею базової оцінки та суми штрафів:

$$S_{total}(j) = S_{base}(j) - (P_{missed}(j) + P_{timing}(j) + P_{fork}(j)). \quad (3.11)$$

Формула (3.11) реалізує адитивну скорингову модель із субтрактивним урахуванням штрафів: базова оцінка, що відображає позитивний внесок вузла (кількість згенерованих блоків), зменшується на суму штрафів за негативні аспекти роботи. Вузол із високою продуктивністю та відсутністю пропусків, затримок і форків отримує максимальне S_{total}. Натомість вузол, що часто пропускає чергу або спричиняє форки, отримує знижену оцінку – навіть якщо його базова продуктивність висока.

Обгрунтування використання адитивної скорингової функції. Запропонована у формулі 3.11 оцінка є адитивною моделлю, наближеною до канонічної WSM [60]. Вибір саме WSM-подібної композиції, а не WPM, зумовлений тим, щоб уникнути надто жорсткого вето-ефекту, за якого навіть короткочасна

аномалія одного фактору (наприклад, тимчасовий сплеск затримки через мережеву деградацію чи одиничний форк після збою живлення) призводила б до майже нульової підсумкової оцінки, тоді як на рівні консенсусу валідаторів часткова взаємна компенсація між тимчасовими провалами окремих метрик є бажаною – вона захищає системно надійний вузол від випадкового «випадіння» з ротації. Контрастом слугує сценарій в підрозділі 4.2, де для розподілу обчислювальних задач обрано саме мультиплікативну композицію: там вето-ефект корисний, бо запобігає призначенню важких задач перевантаженим або ресурсно-дефіцитним вузлам. Таке розмежування фіксує свідомий методологічний вибір моделі MCDM залежно від контексту оптимізації: адитивна композиція для агрегації метрик якості роботи валідатора, мультиплікативна композиція для відбору ресурсно-здатного обчислювача.

Механізм агрегації рангів. У розподіленій мережі кожен вузол бачить лише частину загального стану: ті блоки й транзакції, що досягають його через мережеві з'єднання. Тому метрики ефективності інших вузлів обчислюються локально – на основі власного стану блокчейну та мемпулу. Щоб отримати консенсусну оцінку ефективності, запропоновано механізм агрегації рангів: кожен вузол подає звіт із впорядкованим списком вузлів за їхньою ефективністю.

Агрегований ранг вузла k обчислюється як зважене середнє позицій, наданих іншими вузлами:

$$\bar{P}(k) = \frac{1}{m_k} \cdot \sum_{i=1}^{m} I_i\,(k) \cdot pos_i(k),$$
$$m_k = \sum_{i=0}^{m} I_i(k),\ I_i(k) \in \{0,1\}, \qquad (3.12)$$

де m– кількість поданих звітів, $I_i(k)$ – індикаторна функція (дорівнює 1, якщо вузол i включив вузол k до свого звіту, 0 – інакше), а $pos_i(k)$ – позиція вузла k у ранзі вузла i. Нормалізація на m_k гарантує коректне порівняння рангів навіть коли різні вузли мають різну кількість звітів. Найефективнішим вважається вузол з найнижчим агрегованим рангом $\bar{P}(k)$.

Вікно подачі звітів. Порядок підписантів оновлюється на основі метрик ефективності періодично, а не при кожному новому блоці – це зменшує обчислювальне навантаження і підтримує стабільність порядку. Працює механізм так. Кожні E блоків відкривається вікно подачі звітів тривалістю W блоків. Протягом вікна кожен авторизований підписант може включити свій звіт про ефективність інших вузлів у заголовок блоку (поле extraData). Після закриття вікна (через W блоків) виконується агрегація всіх отриманих звітів за формулою 3.12 та оновлення порядку підписантів відповідно до нових рангів (рис. 3.1).

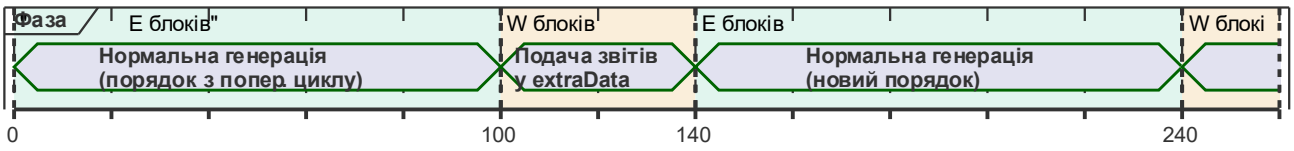

Рисунок 3.1 – Часова діаграма циклу оновлення ефективності

Механізм обмеження непродуктивних вузлів. Третій ключовий елемент динамічного вибору валідаторів – концепція обмеження кількості вузлів, що можуть ставати черговими вузлами. У стандартному Clique всі n авторизованих підписантів по черзі отримують статус чергового вузла для кожного блоку. PoI пропонує інший підхід: кількість чергових вузлів обмежується до n_e найефективніших (параметр *InTurnNodeLimit*). В результаті лише перші n_e вузлів за агрегованим рангом $\bar{P}(k)$ отримують привілей чергового підписання. Решта вузлів завжди залишаються у позачерговому стані – без пріоритету та з обов'язковою затримкою перед генерацією блоку. Наслідки такого рішення: зменшення кількості потенційних чергових вузлів знижує ймовірність конфліктів між швидким позачерговим вузлом та повільним черговим вузлом. Найефективніші вузли (з найменшою кількістю пропусків, затримок та форків) отримують постійний пріоритет, що стабілізує мережу. Менш ефективні вузли при цьому продовжують функціонувати як позачергові підписанти – вони зберігають можливість генерації блоків у разі відмови чергових вузлів. Важливо відмітити, що при обмеженні непродуктивних вузлів йде окремий розрахунок

заборонених вузлів згідно формули 3.3 для чергових та позачергових, де для чергових $n = n_e$.

Зберігання метрик у заголовку блоку. Для верифіковності та незмінності метрик ефективності, їх запропоновано зберігати безпосередньо у заголовку блоку – в полі extraData. Стандартна специфікація Clique вже використовує extraData для списку підписантів. У PoI це поле розширено: до нього включено звіти про ранжування.

Запропонований механізм динамічного вибору валідаторів формує адаптивну систему, яка автоматично пріоритизує найефективніші вузли для генерації блоків. На відміну від статичного порядку Clique, де позиція вузла визначається виключно його Ethereum-адресою, у PoI позиція залежить від фактичної продуктивності. Для гетерогенних IoT-мереж із різними обчислювальними можливостями вузлів це має особливе значення.

3.4. Модель безпеки адаптивного вибору валідаторів

Агрегація рангів валідаторів на основі комплексної метрики ефективності (вираз 3.12) формується у середовищі, де базова авторизація вузлів-підписантів забезпечується штатними механізмами PoA Clique: список авторизованих адрес фіксується у конфігурації genesis і змінюється лише через голосування більшості чинних підписантів. Це означає, що PoI успадковує від Clique базову протидію Sybil-атакам через вимогу попередньої авторизації адреси, а також протидію зовнішнім атакам шляхом криптографічного підпису блоків.

Водночас формалізація динамічного рангування валідаторів виявляє три специфічні класи загроз. Перший – шахрайство верхньої частини авторизованих підписантів: якщо більшість з них координовано занижує ранги конкурентів, рангова агрегація поступово віддає чергу генерації блоків змовницькій підмножині. Другий – маніпуляція штрафними метриками: валідатор може поводитися чесно, накопичуючи

високий базовий ранг, а потім різко знижувати якість роботи, використовуючи інертність штрафних функцій. Третій – селективне поширення легких блоків: зловмисний черговий вузол може навмисно затримувати доставку повних тіл транзакцій частині вузлів, провокуючи виконання резервного механізму і знижуючи ранг конкурентів.

Часткові контрзаходи реалізовані в самій формалізації: обмеження верхніх меж штрафних коефіцієнтів запобігає необмеженому заниженню рангу; часове вікно з агрегацією звітів кількох вузлів знижує вплив ізольованої маніпуляції. Резервний механізм із порогом відсутніх транзакцій 50% обмежує вигоду від селективного поширення. Водночас повноцінна модель безпеки з формальним доведенням стійкості до шахрайства та порівнянням з репутаційними протоколами виходить за межі обсягу цієї роботи і розглядається як напрямок подальших досліджень.

3.5. Конфігуровна затримка старту позачергових вузлів

Третє удосконалення протоколу Clique в межах PoI – конфігурована затримка старту позачергових вузлів. Мета цього механізму полягає у зменшенні ймовірності розгалуження ланцюга: черговий вузол має отримати достатньо часу, щоб його блок поширився мережею.

Формулювання проблеми. За стандартного Clique позачерговий вузол здатний розпочати генерацію блоку вже через нуль мілісекунд, тобто, разом з черговим вузлом, оскільки нижня межа wiggle time дорівнює нулю, що збільшує ймовірність розгалуження ланцюга при одночасному поширенні двох або більше блоків в різні частини мережі.

Запропоноване рішення. Для усунення даної проблеми введено конфігураційний параметр *RandomStartMultiplier*. Таким чином можна визначити

початкову затримку з формули 3.4, що створює додаткове часове вікно, протягом якого черговий блок може поширитися мережею:

$$t_i = \omega \cdot \text{RandomStartMultiplier}. \quad (3.13)$$

Збільшення мінімальної затримки позачергових вузлів прямо впливає на ймовірність розгалуження ланцюга. Водночас існує компроміс: більше значення RandomStartMultiplier підвищує середню затримку генерації блоків у ситуаціях, коли черговий вузол відмовляє. Позачерговим вузлам за таких обставин потрібно більше часу для запуску компенсаторної генерації. Для IoT-мереж цей компроміс є прийнятним, оскільки стабільність і відсутність розгалужень тут важливіші за мінімальну затримку в аварійних сценаріях.

Поєднання з іншими удосконаленнями. Конфігуровна затримка працює у синергії з двома іншими удосконаленнями PoI. Компактні блоки (підрозділ 3.2) скорочують час поширення блоку, а отже – зменшують мінімально необхідну затримку для позачергових вузлів. Динамічний вибір валідаторів (підрозділ 3.3) гарантує, що черговими вузлами стають найпродуктивніші учасники мережі, які генерують блоки швидше. Вікно вразливості до розгалужень ланцюга при цьому додатково звужується.

3.6. Програмна реалізація PoI

Запропонований метод адаптивного консенсусу PoI реалізовано як модифікацію клієнта Go-Ethereum (Geth) – офіційної реалізації протоколу Ethereum мовою Go [82]. Go-Ethereum було обрано через відкритий вихідний код, модульну архітектуру та широку підтримку спільноти. Даний клієнт здатний функціонувати на пристроях класу Raspberry Pi, що є принциповим для IoT-застосувань. Підрозділ описує архітектуру внесених модифікацій, конфігурацію через genesis file та зміни у мережевому протоколі ETH68. Рисунок 3.2 показує які саме файли було змінено (прозорі блоки) та які нові створено (світло-зелені блоки) для реалізації PoI.

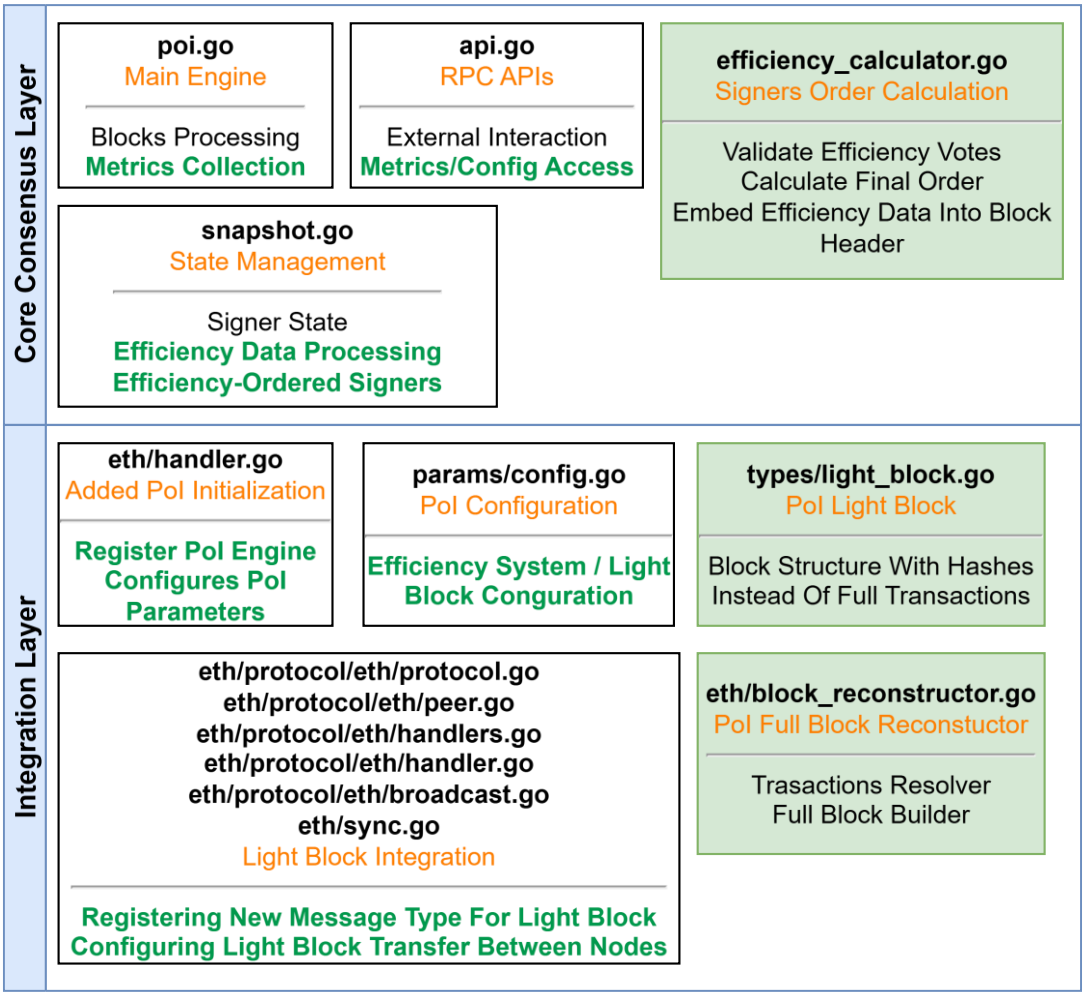

Рисунок 3.2 – Структура реалізації PoI в проекті Go-Ethereum

3.6.1. Модифікації Go-Ethereum Clique

Модифікації Go-Ethereum побудовані за принципом мінімальної інвазивності – PoI реалізується не як окремий консенсусний механізм, а як розширення вже існуючого модуля Clique. Директорію consensus/clique скопійовано в директорію consensus/poi та доповнено додатковими структурами даних та методами, які активуються тільки при наявності відповідних параметрів у файлі genesis. Якщо PoI-параметри відсутні, модуль працює ідентично стандартному Clique, тим самим зберігаючи повну зворотну сумісність.

Основні модифікації стосуються трьох компонентів ядра клієнта. Перший – модуль обчислення метрик ефективності. Він відповідає за збір статистики про роботу

кожного підписанта: кількість згенерованих блоків, пропуски черги, затримки, розгалуження ланцюга. На основі цих даних обчислюються значення S_{total} за формулою 3.11. Модуль інтегровано у процес верифікації нових блоків, що дає можливість оновлювати метрики при кожному новому блоці без додаткових мережевих запитів. Другий компонент – модуль управління порядком підписантів, що реалізує механізм збору звітів у межах вікна подачі, агрегацію рангів за формулою 3.12, після чого оновлюється порядок підписантів. Третій компонент відповідає за компактні блоки: формує їх при генерації та виконує реконструкцію при отриманні (Алгоритм 3.1).

При відсутності у файлі genesis PoI-специфічних параметрів (BlockOptimization, EnableMetrics) модулі PoI залишаються неактивними. Це дозволяє розгортати PoI-сумісні вузли у змішаних мережах, де частина вузлів використовує стандартний Geth.

3.6.2. Конфігурація через файл genesis

Конфігурація PoI відбувається через секцію poi у файлі genesis.json – стандартному файлі ініціалізації блокчейн-мережі Ethereum. Було запропоновано шість додаткових до стандартної конфігурації з clique конфігураційних параметрів, і кожен з них контролює конкретний аспект поведінки PoI.

Таблиця 3.1 – Конфігураційні параметри PoI та їх вплив на поведінку мережі

Параметр	**Тип**	**Призначення**
BlockOptimization	bool	Увімкнення компактних блоків (Light Blocks)
EnableMetrics	bool	Увімкнення метрик ефективності та динамічного вибору
EfficiencyCheckPeriod Blocks	int	Період перевірки ефективності (кількість блоків між циклами)

Параметр	**Тип**	**Призначення**
EfficiencyCheckSubmissionWindow	int	Тривалість вікна подачі звітів (кількість блоків)
InTurnNodeLimit	int	Кількість пріоритетних чергових вузлів
RandomStartMultiplier	int	Множник затримки старту позачергових вузлів

За замовчуванням усі параметри налаштовані так, щоб поведінка відповідала стандартному Clique. Якщо BlockOptimization має значення true – активується генерація та обробка легких блоків. EnableMetrics при значенні true вмикає збір метрик ефективності разом із динамічним вибором валідаторів. Параметри EfficiencyCheckPeriodBlocks та EfficiencyCheckSubmissionWindow задають частоту і тривалість циклів оновлення ефективності відповідно. InTurnNodeLimit визначає кількість ефективних вузлів, яким надається привілей бути черговими вузлами; при значенні 0 усі авторизовані підписанти отримують цю можливість, що еквівалентно стандартному Clique. RandomStartMultiplier впливає на множник затримки для позачергових вузлів.

Модульна конфігурація створює умови для того, щоб адміністратори IoT-мережі активували окремі удосконалення PoI незалежно одне від одного. Можна, наприклад, увімкнути лише компактні блоки (BlockOptimization = true) без динамічного вибору валідаторів, або навпаки – це робить можливим тестування окремих компонентів, гнучке розгортання та поетапне впровадження.

3.6.3. Модифікації мережевого протоколу ETH68

Для підтримки легких блоків було модифіковано мережевий протокол ETH68, реалізований у пакеті eth/protocols/eth клієнта Go-Ethereum. Зміни торкнулися п'яти файлів.

У файлі **protocol.go** додано три нові константи типів повідомлень: NewLightBlockMsg, GetMissingTxsMsg та MissingTxsMsg. Коди повідомлень не конфліктують з існуючими типами стандартного протоколу ETH68. Файл **peer.go** розширено методами відправки та отримання нових типів повідомлень через наявний механізм однорангових (P2P) з'єднань. У **broadcast.go** реалізована умовна трансляція: коли BlockOptimization увімкнено, замість стандартного NewBlockMsg надсилається NewLightBlockMsg із хешами транзакцій замість повних тіл. Файл **handlers.go** доповнено обробниками для нових типів повідомлень – тут міститься логіка реконструкції блоку (алгоритм 3.1). Файл **handler.go** модифіковано для маршрутизації нових повідомлень до відповідних обробників.

На рівні смарт-контрактів реалізовано контракт IoTDataTracker мовою Solidity для оптимізованого зберігання IoT-телеметрії. Контракт підтримує три можливості: change-only storage (запис нового значення лише при відмінності від попереднього), мультисенсорну конфігурацію (реєстрація та відстеження множини сенсорів з різними типами даних), а також пакетні оновлення – групову передачу значень від кількох сенсорів в одній транзакції. Як зазначалося у підрозділі 2.3, IoTDataTracker є компонентом кластерного рівня тришарової архітектури, що дозволяє зберігати дані від сенсорів без перевантаження блокчейну надлишковими транзакціями.

3.7. Експериментальне дослідження PoI

Для кількісної оцінки ефективності запропонованого методу адаптивного консенсусу PoI було проведено серію експериментів у контрольованому тестовому середовищі. Мета експерименту – підтвердити теоретичні переваги PoI, а саме: зменшення мережевого трафіку, зменшення ймовірності розгалужень ланцюга, підвищення пропускної здатності. Окремо оцінювалося додаткове навантаження на обчислювальні ресурси.

3.7.1. Тестове середовище та методика

Тестове середовище побудоване на базі Docker-контейнерів, кожен з яких містить модифікований клієнт Go-Ethereum із підтримкою PoI. Docker дає ізольоване та відтворюване середовище, що дозволяє контролювати параметри мережі й виключити вплив зовнішніх факторів.

Мережа складається з 15 вузлів Go-Ethereum, з'єднаних за топологією partial mesh – її обрано як найбільш реалістичну для IoT-мереж, де кожен вузол підключений до підмножини інших вузлів, але не до всіх. Для моделювання реалістичних мережевих умов застосовано утиліту Linux Traffic Control, що дозволяє програмно додавати затримки до кожного з'єднання між контейнерами. Діапазон затримок – до 200 мс із параметром jitter до 20 мс, що відповідає типовим характеристикам IoT-мереж з географічною розподіленістю вузлів [5]. Параметри даного тестового середовища систематизовано у Таблиці 3.2.

Таблиця 3.2 – Параметри тестового середовища

Параметр	**Значення**
Кількість вузлів	15
Платформа	Docker (контейнеризація)
Клієнт	Go-Ethereum (модифікований Geth)
Топологія	Partial mesh
Мережева затримка	До 200 мс (Linux TC)
Jitter	До 20 мс
Block period	3 секунди
Інструмент мережевого аналізу	Edgeshark + Wireshark

Параметр	**Значення**
Інструмент ресурсного моніторингу	Docker statistics
EfficiencyCheckPeriodBlocks	100 блоків
EfficiencyCheckSubmissionWindow	10 блоків
InTurnNodeLimit	5 (з 15 вузлів)
RandomStartMultiplier	1

Методика тестування побудована за принципом поетапного вмикання удосконалень – це дозволяє оцінити внесок кожного компоненту PoI окремо та їхній кумулятивний ефект. Перший етап передбачає конфігурацію з увімкненими лише метриками ефективності (EnableMetrics = true, BlockOptimization = false, RandomStartMultiplier = 0), тобто ізолюється ефект динамічного вибору валідаторів від решти оптимізацій. Другий етап – повна конфігурація PoI (EnableMetrics = true, BlockOptimization = true, RandomStartMultiplier = 1, InTurnNodeLimit = 5), що відображає кумулятивний ефект усіх трьох удосконалень. Базовою конфігурацією є стандартний Clique без PoI-параметрів.

Для кожної конфігурації мережа функціонувала протягом визначеного періоду з генерацією IoT-навантаження (пакетні транзакції через IoTDataTracker). Метрики збиралися через програмний інтерфейс клієнта Geth, статистику Docker та Edgeshark у поєднанні з Wireshark. Детальніше про деталі збору метрик – у таблиці 3.3.

Таблиця 3.3 – Механізми обчислення ключових метрик з обох мереж PoI та Clique

Метрика	**Джерело**	**Механізм обчислення**
Час блоку	Geth API, eth.get_block(n)	Різниця між часом поточного і попереднього блоку

Пропускна здатність	Geth API, eth.get_block(n, full_transactions=True) для всіх блоків у вікні 10 с	Сумарна кількість транзакцій за період, поділена на різницю в часі між крайніми блоками
Час поширення блоку	Geth API, eth.block_number на кожному вузлі кожні 0.3с	Запитується висота ланцюга і фіксується час збільшення висоти.
Розгалуження ланцюга (форки)	Geth API, eth.get_block(n) на кожному вузлі кожні 0.3с	Подія форку – неспівпадіння хешів блоку на однаковій висоті ланцюга.
Споживання процесору	Docker Stats API	Середнє значення за 10с
Споживання пам’яті	Docker Stats API	Середнє значення за 10с
Мережевий трафік	Wireshark / Edgeshark	Загальна статистика трафіку з кроком в 10с

Для кожного запуску генерувалися 10 акаунтів з початковим балансом 10 ETH на акаунт. Спочатку в мережі реєструвалися IoT-пристрої, а потім оновлювались їх дані. Розмір транзакцій варіювався від 500 байт у нормальному режимі до 51 200 байт у режимі --heavy для імітації передачі великих IoT-даних.

З метою рівномірного розподілу транзакцій між вузлами, кожна наступна транзакція відправлялась на інший блокчейн-вузол. Загальна тривалість тестування складає близько 30 хвилин, де на першій фазі протягом 15 хвилин виконуються всі заміри окрім кількості форків (два сценарії: SmartHome – 2 TPS та Load – 10 TPS). На другій фазі (15 хв) спеціально подавались транзакції без перерви між ними на різні вузли та конфліктні транзакції (з однаковим nonce), що дозволило створити екстремальні умови для мережі та заміряти кількість розгалужень ланцюга.

3.7.2. Результати поетапного тестування

Етап 1: тільки метрики ефективності. При увімкненні лише динамічного вибору валідаторів (без компактних блоків та без збільшеної затримки для позачергових вузлів) зафіксовано помірне покращення основних метрик. Виміряна середня швидкість передавання даних у мережі для PoI становила 1,27 Мбіт/с, тоді як Clique досягав 1,4 Мбіт/с, що відповідає різниці приблизно 9%. Для статистичного аналізу було застосовано логнормальний розподіл з нормованою щільністю ймовірності, оскільки саме він рекомендується для оцінювання мережевих характеристик [83]. Отриманий ефект пояснюється зменшенням кількості конкурентних блоків: пріоритизація ефективних чергових вузлів знижує ймовірність одночасної генерації кількох блоків, а отже – скорочує обсяг даних, що поширюються мережею. Кількість розгалужень ланцюга при цьому впала на 25%, що підтверджує вплив динамічного вибору на стабільність ланцюга.

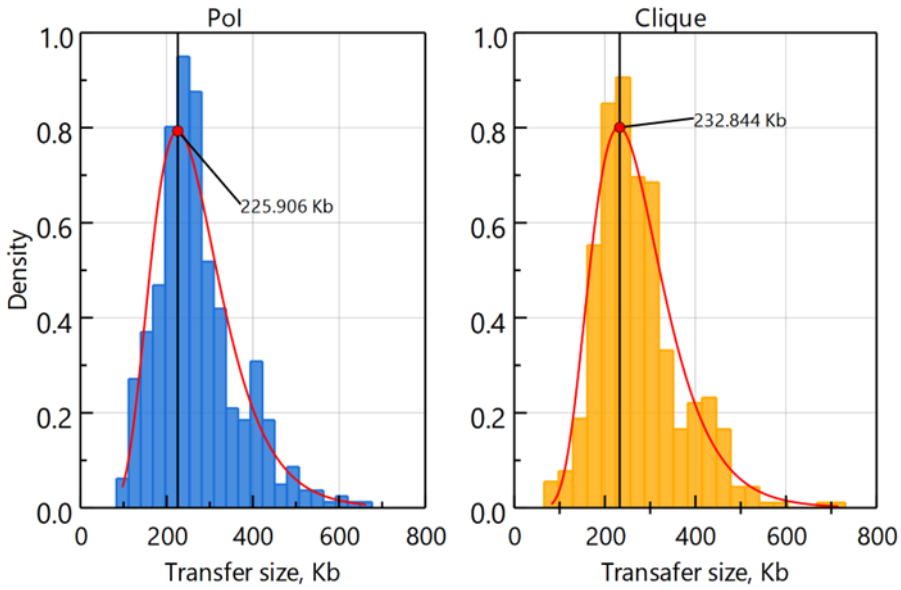

Рис. 3.3 – Аналіз мережевого трафіку з увімкненим механізмом динамічного вибору валідаторів PoI

Реалізація PoI демонструє покращену продуктивність. Зокрема, середній час поширення блоку зменшується приблизно на 8,5%, а час блоків скорочується приблизно на 3,1%. Як проілюстровано на рис. 3.4 та рис. 3.5, аналіз логнормального

розподілу часу поширення блоків та часу блоків показує перевагу у продуктивності, досягнуту завдяки реалізації метрик ефективності PoI.

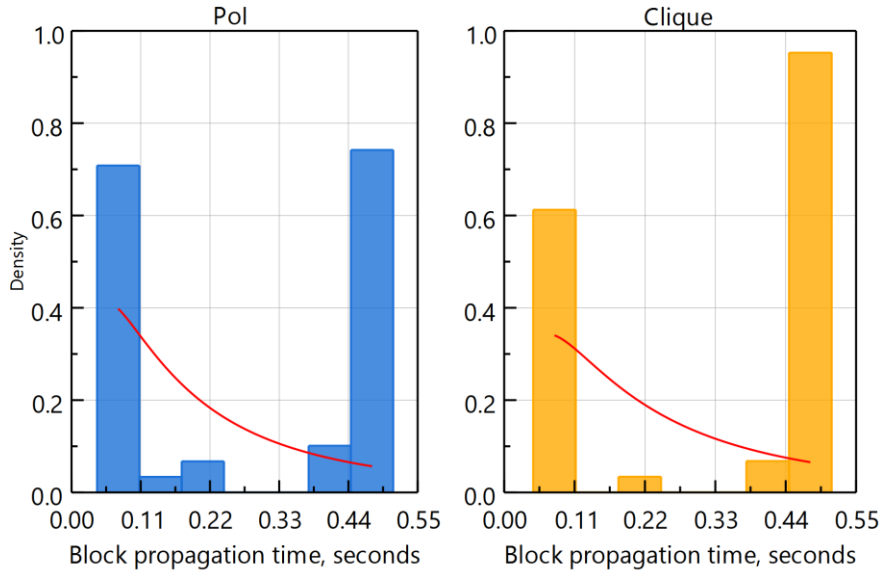

Рис. 3.4 – Аналіз часу поширення блоків з увімкненим механізмом динамічного вибору валідаторів PoI

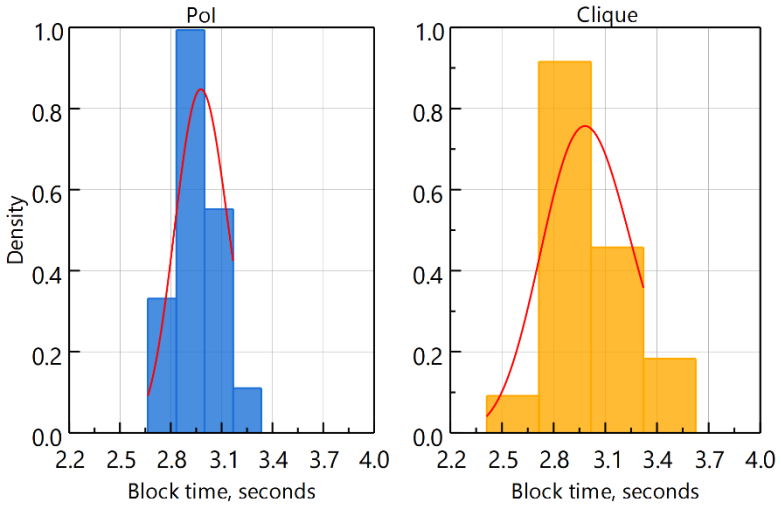

Рис. 3.5 – Аналіз часу блоків з увімкненим механізмом динамічного вибору валідаторів PoI

Етап 2: повна конфігурація PoI. При увімкненні всіх трьох удосконалень зафіксовано суттєве покращення за всіма ключовими метриками. Нижче наведено результати порівняння повної конфігурації PoI зі стандартним Clique.

Найбільш виражений ефект – зменшення мережевого трафіку: повна конфігурація PoI дала зниження на 20,5% порівняно зі стандартним Clique. Це результат поєднання двох механізмів – компактні блоки зменшують обсяг кожного переданого блоку (тіла транзакцій замінюються на хеші), а динамічний вибір валідаторів скорочує кількість конкурентних блоків, що потребують поширення мережею. Разом з цим зафіксовано зменшення кількості розгалужень ланцюга на 80%, оскільки комбінація пріоритизації ефективних вузлів та збільшення затримки поширення позачергових блоків різко зменшує ймовірність одночасної генерації конкурентних блоків.

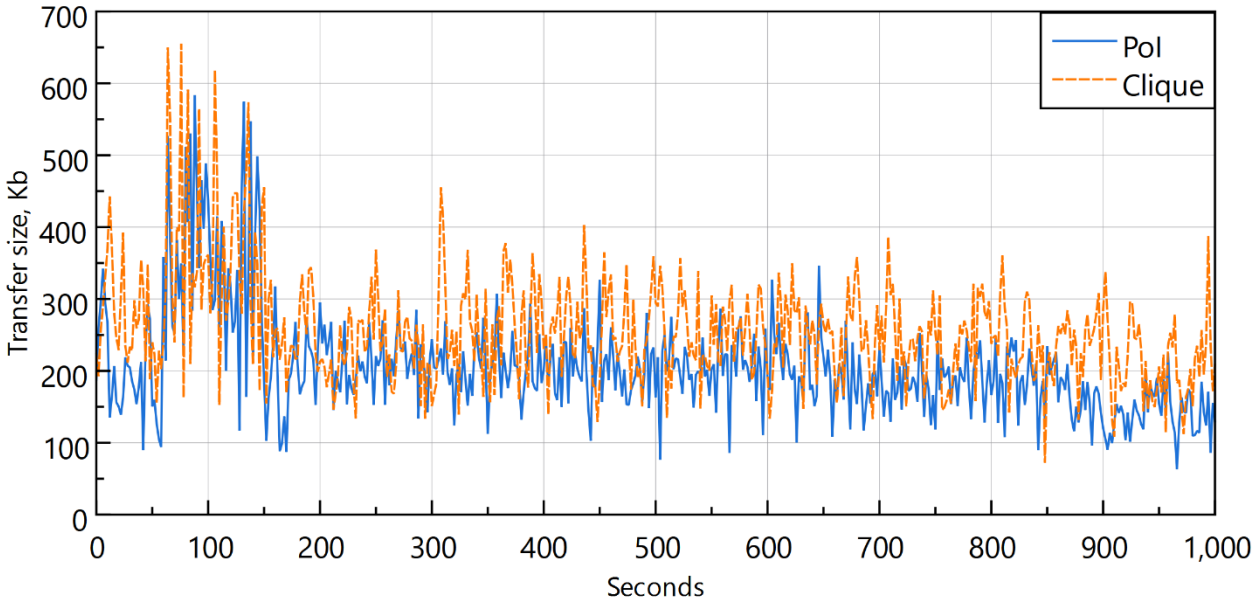

Рис. 3.6 – Захоплення мережевого трафіку у WireShark з усіма увімкненими оптимізаціями PoI

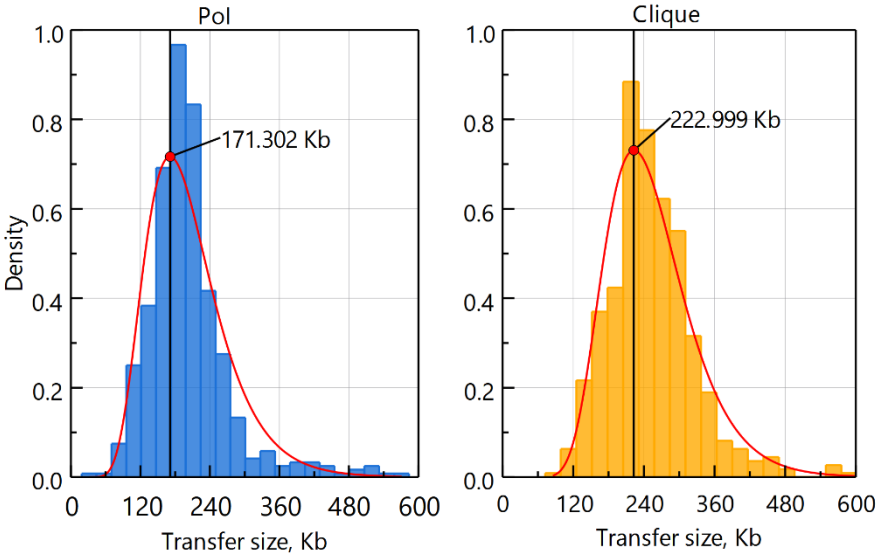

Рис. 3.7 – Аналіз мережевого трафіку з усіма увімкненими оптимізаціями PoI

Пропускну здатність (TPS) у розрахунку на пристрій вдалося підвищити в середньому приблизно на 5–10% залежно від сценарію тестування (рис. 3.8). При цьому отриманий результат не відображає повного потенціалу підвищення TPS, оскільки оцінювання зосереджувалося на навантаженні, характерному для спеціалізованої IoT-мережі.

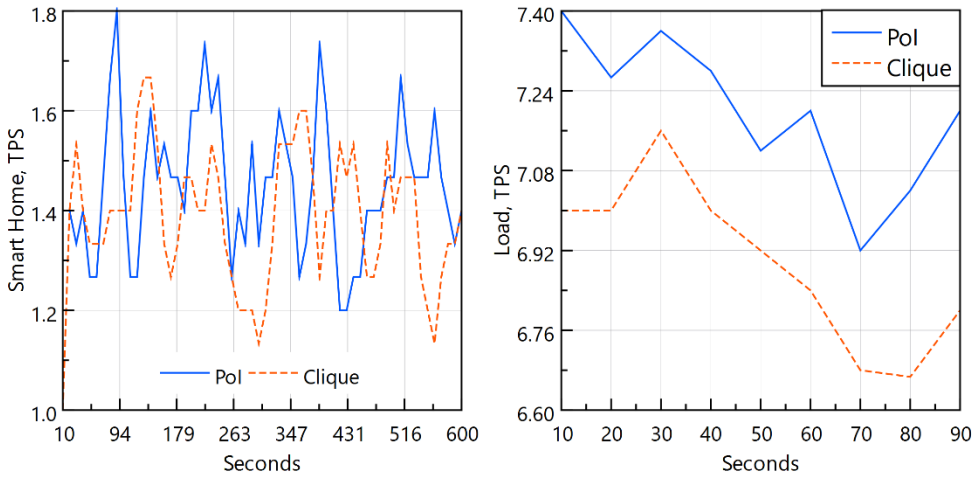

Рис. 3.8 – Захоплення TPS на пристрій з 10-секундними інтервалами за умови увімкнення всіх оптимізацій PoI

Водночас додаткове навантаження на обчислювальні ресурси залишається помірним. Зростання використання процесору – 6,5%, що пов'язано з додатковими

обчисленнями метрик ефективності при валідації кожного блоку та обробці звітів. Використання оперативної пам'яті зросло на 5,4% через зберігання додаткових структур даних: масиву метрик для кожного підписанта, буфера звітів та кешу хешів транзакцій для реконструкції компактних блоків.

3.8. Висновки

В результаті аналізу протоколу Clique встановлено ряд недоліків, які вирішено у представленому методі адаптивного консенсусу Proof of Indicators (PoI).

Використання техніки пропагації компактних блоків, динамічного вибору валідаторів та затримки старту позачергових вузлів дозволило покращити ефективність роботи системи (в порівнянні з Clique) згідно стандарту ISO/IEC 25010 в категоріях Performance Efficiency та Reliability. За часовою характеристикою (Time Behaviour) відбулось скорочення часу поширення блоків на 8,5% за рахунок динамічного вибору валідаторів та компактних блоків. За підкатегорією використання ресурсів (Resource Utilization) мережевий трафік зменшився на 20,5%, одночасно помічено відбулось зростання використання процесору (6,5%) та пам’яті (5,4%), що пояснюється використанням метрик ефективності та перебудовою блока з локального кешу. Збільшення пропускної здатності (Capacity) до 10% також пояснюється введеними оптимізаціями. В категорії надійності (Reliability) покращена безвідмовність системи (Faultlessness) внаслідок скорочення кількості розгалужень ланцюга на 80% завдяки динамічному вибору валідаторів та конфігуровній затримці позачергових вузлів PoI.

PoI оптимізує лише мережевий рівень консенсусу і не враховує поточний стан обчислювальних ресурсів вузлів при розподілі прикладних задач. Для повноцінної координації периферійних вузлів IoT-мережі потрібен механізм прикладного рівня з

ресурсно-зваженим розподілом обчислень – модель мультифакторного управління вузлами, що формує разом із PoI дворівневу систему координації, описано у Розділі 4.

РОЗДІЛ 4. УДОСКОНАЛЕННЯ БЛОКЧЕЙН-МОДЕЛІ МУЛЬТИФАКТОРНОГО УПРАВЛІННЯ ПЕРИФЕРІЙНИМИ ВУЗЛАМИ

Даний розділ присвячено удосконаленню моделі мультифакторного вибору вузлів для блокчейн-координованих периферійних обчислень. Тут представлено тришарову архітектуру периферійних обчислень, розроблено мультифакторну скорингову функцію мультиплікативної композиції із шістьма факторами оцінки, описано механізми забезпечення надійності та програмну реалізацію на базі смарт-контрактів Solidity та Python-агентів. Проведено експериментальне дослідження запропонованого підходу у Docker-середовищі.

4.1. Тришарова архітектура блокчейн-координованих периферійних обчислень

Тришарова архітектура IoT-кластера, запропонована у Розділі 2, визначає три рівні організації: сенсорний рівень, рівень локального блокчейну та рівень зовнішнього блокчейну. Метод адаптивного консенсусу PoI (Розділ 3) реалізує ефективну координацію на мережевому рівні – визначає порядок генерації блоків та оптимізує мережевий трафік. Проте PoI не вирішує задачі розподілу прикладних обчислювальних задач між вузлами кластера. Консенсусний рівень оперує блоками та транзакціями, а не обчислювальними ресурсами. У даному підрозділі представлено адаптацію тришарової архітектури для сценарію блокчейн-координованих периферійних обчислень, що є базовою інфраструктурою для мультифакторного управління вузлами [6].

Зв'язок із тришаровою архітектурою Розділу 2. Мультифакторне управління вузлами є надбудовою прикладного рівня над блокчейн-координаційним рівнем кластера. У Розділі 2 прикладний рівень визначено як сукупність смарт-контрактів та бізнес-логіки, що виконуються на платформі EVM локального кластера. Контракт IoTDataTracker (підрозділ 2.3) відповідає за зберігання телеметрії від сенсорів, підхід

BDT (підрозділ 2.2) гарантує безпечну передачу даних. Мультифакторне управління додає до цієї інфраструктури два нових контракти – IoTRegistry та ComputeManager – що разом реалізують повний цикл розподілу обчислювальних задач: від реєстрації вузлів та збору метрик до вибору оптимального вузла та верифікації результатів.

Тришарова модель периферійних обчислень. Для сценарію edge computing архітектура IoT-кластера конкретизується у три функціональних рівні. Кожен з них відповідає конкретній ролі у процесі обробки обчислювальних задач (рис. 4.1).

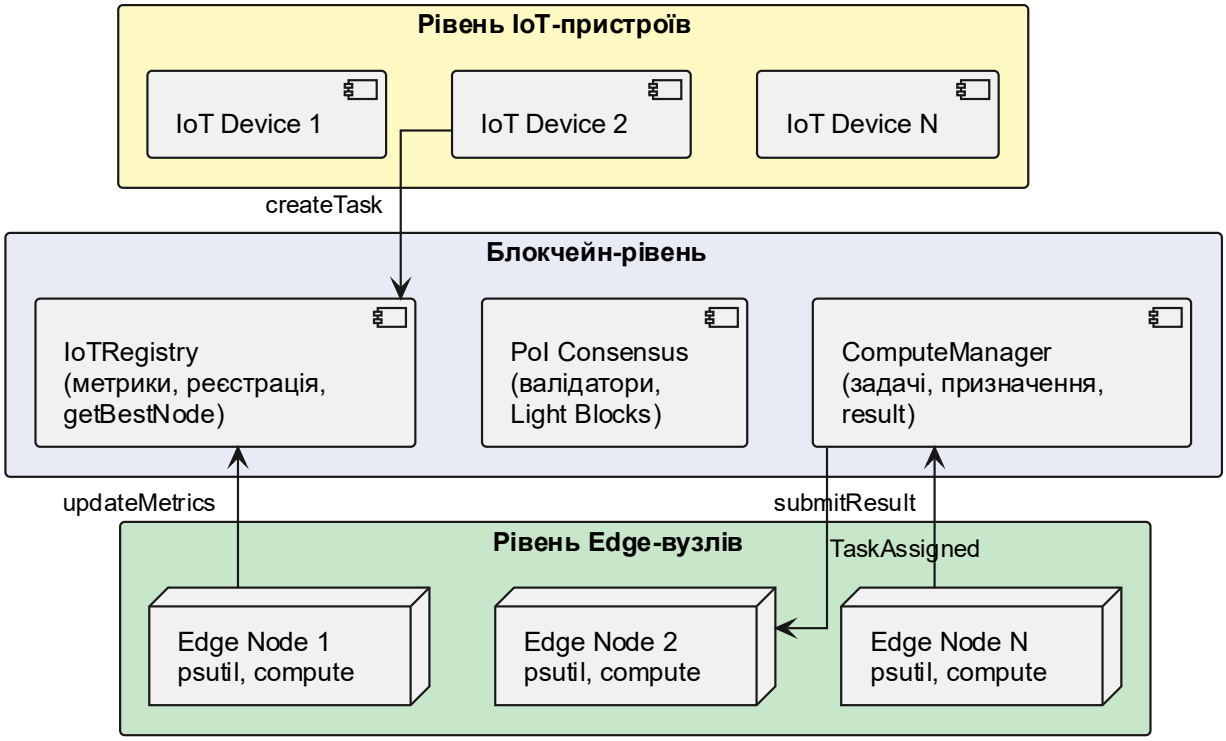

Рисунок 4.1 – Тришарова архітектура блокчейн-координованих периферійних обчислень

IoT Device Layer – нижній рівень, що об'єднує IoT-пристрої, які генерують обчислювальні задачі. Пристрої цього рівня мають обмежені обчислювальні ресурси та не здатні самостійно виконувати ресурсоємні операції, такі як аналіз часових рядів, машинне навчання на потоках даних або агрегація великих масивів сенсорних даних. Замість локальної обробки вони формують запити на виконання задач та надсилають їх до рівня блокчейну у вигляді транзакцій до смарт-контракту ComputeManager.

Blockchain Layer – середній рівень, на якому реалізовано координацію через смарт-контракти. Два контракти – IoTRegistry та ComputeManager – містять логіку управління: IoTRegistry зберігає реєстрацію вузлів, їхні метрики (процесор, пам'ять, черга задач, свіжість метрик) та обчислює оцінку для кожного вузла; ComputeManager приймає запити на виконання задач, викликає IoTRegistry для вибору оптимального вузла, контролює виконання та приймає результати. Взаємодія між контрактами є атомарною – вибір вузла для задачі виконується в рамках однієї транзакції, що гарантує консистентність стану навіть при одночасному надходженні кількох задач.

Edge Node Layer – верхній рівень, що об'єднує периферійні вузли, які фактично виконують задачі. Дані вузли можуть бути повноцінними серверами або одноплатними комп'ютерами (Raspberry Pi, NVIDIA Jetson) з достатніми ресурсами для виконання обчислювальних скриптів. На кожному периферійному вузлі працює Python-агент, що виконує три функції: моніторинг локальних ресурсів (процесор, пам'ять, диск) та передачу метрик до контракту IoTRegistry; підписку на події контракту ComputeManager (TaskAssigned) для отримання призначених задач; виконання обчислювальних скриптів та повернення результатів до блокчейну.

Розмежування операцій. Принциповою характеристикою даної архітектури є чітке розділення операцій, що виконуються у межах блокчейну (on-chain) та поза ним (off-chain). Це дозволяє мінімізувати навантаження на блокчейн при збереженні повної координаційної функціональності. До on-chain операцій належать реєстрація вузлів (IoTRegistry.registerNode), оновлення метрик (IoTRegistry.updateMetrics), обчислення score (IoTRegistry.getBestNode), створення задач (ComputeManager.createTask), призначення вузлів (ComputeManager.assignNode), контроль часу виконання (ComputeManager.checkTimeout) та верифікація результатів (ComputeManager.submitResult). Off-chain операції включають моніторинг ресурсів (psutil), виконання обчислювальних скриптів, зберігання вхідних та вихідних даних задач, та мережеву комунікацію з IoT-пристроями. Таке розділення означає, що

блокчейн використовується виключно для координації та верифікації, а не для зберігання великих масивів даних або виконання ресурсоємних обчислень.

Автоматизація, керована подіями. Повний цикл обробки обчислювальної задачі реалізований як ланцюг подій (events) – від створення задачі (TaskCreated) через призначення вузла (TaskAssigned) до отримання результату (ResultSubmitted) та оновлення репутації (MetricsUpdated). Це робить можливою автоматичну координацію без централізованого оркестратора. Детальний опис архітектури взаємодії компонентів та повну діаграму послідовності подій наведено у підрозділі 4.4. Якщо вузол не завершує задачу протягом визначеного часу, активується механізм перепризначення, що описано у підрозділі 4.3.

Запропонована тришарова архітектура периферійних обчислень формує інфраструктурну основу для мультифакторного управління вузлами, де блокчейн виступає координаційним механізмом, а не обчислювальною платформою. Ключова відмінність від централізованих рішень (хмарні оркестратори, Kubernetes-кластери) полягає у відсутності єдиної точки відмови: навіть при відмові частини вузлів координаційна логіка зберігається у блокчейні та залишається доступною всім учасникам мережі

4.2. Модель мультифакторного вибору вузлів

Даний підрозділ є ядром наукового внеску Розділу 4 та формулює модель мультифакторного вибору вузлів для блокчейн-координованих периферійних обчислень. Модель удосконалює класичні підходи до вибору вузлів, що базуються на одновимірних метриках і відповідає наступному науковому результату дисертації: удосконалено мультифакторний вибір периферійних вузлів.

4.2.1. Обмеження класичного підходу

Класичний підхід до вибору обчислювального вузла у блокчейн-мережі зводиться до одновимірного критерію – репутації, що визначається як відношення

кількості успішно виконаних задач до загальної кількості призначених задач. Формально класичний вибір визначається як:

$$n_s = \arg\max_{n \in N} R(n), \tag{4.1}$$

де N – множина зареєстрованих вузлів, $R(n)$ – репутація вузла n, а n_s – обраний вузол.

Даний підхід не враховує поточне навантаження вузла за ресурсними характеристиками, а одновимірна метрика не дає змоги відрізнити вузол, що тривалий час не оновлював свої метрики, від вузла, що активно працює та регулярно звітує про стан ресурсів. Альтернативні підходи, розглянуті у Розділі 1, також мають обмеження. Методи глибокого навчання з підкріпленням (DRL) дають змогу здійснювати адаптивний вибір, проте мають недетерміністичну поведінку та обчислювальну складність, що складно реалізувати в блокчейн-середовищі. Пріоритетні черги із централізованим управлінням потребують складності $O(n\log n)$ та централізованого оркестратора, що суперечить принципу децентралізації блокчейн-архітектури.

4.2.2. Мультифакторна скорингова функція

Для подолання зазначених обмежень запропоновано мультифакторну скорингову функцію мультиплікативної композиції, що поєднує шість факторів для оцінки кожного зареєстрованого вузла:

$$n_s = \arg\max_{n \in N} \left[R(n) \cdot F_{cpu}(n) \cdot F_{mem}(n) \cdot F_{queue}(n) \cdot F_{fresh}(n) \cdot F_{time}(n)\right], \tag{4.2}$$

де $R(n)$ – репутація вузла, $F_{cpu}(n)$ – фактор доступності CPU, $F_{mem}(n)$ – фактор доступності пам'яті, $F_{queue}(n)$ – фактор ємності черги задач, $F_{fresh}(n)$ – фактор свіжості метрик, а $F_{time}(n)$ – фактор часових гарантій. Кожен фактор нормалізовано до діапазону $[F_{min}, 1]$, де F_{min} – мінімальне допустиме значення, що запобігає нульовому score при периферійних значеннях окремих факторів.

Фактор 1: Репутація вузла $R(n)$. Репутація відображає історичну надійність вузла та визначається як:

$$R(n) = \frac{T_s(n)}{T_a(n)} \cdot R_{max}, \quad (4.3)$$

де $T_s(n)$ – кількість успішно виконаних задач вузлом n, $T_a(n)$ – загальна кількість призначених задач, а R_{max} – максимальне значення репутації (нормалізуючий коефіцієнт).

Доцільність даного підходу до репутації підтверджується ієрархічною моделлю репутації для VANET [53], де динамічне оцінювання вузлів також базується на відношенні успішних взаємодій до загальної кількості. Для новозареєстрованих вузлів ($T_a(n) = 0$) встановлюється початкове значення репутації R_0, що дає їм можливість отримувати задачі та накопичувати історію.

Фактор 2: Доступність CPU $F_{cpu}(n)$. Цей фактор відображає поточне завантаження центрального процесора вузла:

$$F_{cpu}(n) = \max\big(F_{min},\, 1 - L_{cpu}(n, \Delta t)\big), \quad (4.4)$$

де $L_{cpu}(n, \Delta t)$ – середнє завантаження CPU вузла n за вікно спостереження Δt, а F_{min} – мінімальне допустиме значення фактора, що зберігає ненульовий score та теоретичну можливість вибору вузла при відсутності альтернатив.

Фактор 3: Доступність пам'яті $F_{mem}(n)$. Аналогічно до фактору CPU, доступність оперативної пам'яті визначається як:

$$F_{mem}(n) = \max(F_{min},\, 1 - L_{mem}(n, \Delta t)), \quad (4.5)$$

де $L_{mem}(n, \Delta t)$ – відношення використаної пам'яті до загального обсягу за вікно Δt.

Фактор 4: Ємність черги задач $F_{queue}(n)$. Даний фактор відображає ступінь завантаженості черги задач вузла:

$$F_{queue}(n) = \max\big(F_{q\_min},\, 1 - L_{queue}(n, \Delta t)\big), \quad (4.6)$$

де $L_{queue}(n, \Delta t)$ – відношення поточної кількості задач у черзі до максимальної ємності черги, а F_{q_min} – мінімальне значення фактора черги.

На відміну від факторів, що відображають апаратні ресурси, фактор черги характеризує прикладний стан – кількість задач, які вузол ще не розпочав або не завершив.

Фактор 5: Актуальність метрик. Цей фактор вирішує проблему вузла, що припинив функціонування або втратив мережеве з'єднання, але формально залишається зареєстрованим у IoTRegistry.

Фактор актуальності визначається як функція часу, що минув з моменту останнього оновлення метрик:

$$\mathrm{F}_{\mathrm{fresh}}(n) = \begin{cases} 1 & \Delta t_m < T_{\mathrm{m}} \\ \mathrm{F}_{\mathrm{min}} & \Delta t_m > T_{\mathrm{m}} \end{cases}, \tag{4.7}$$

де Δt_m– різниця між поточним часом та часом останнього оновлення метрик вузлом n, а T_m – порогове значення, після якого метрики вважаються повністю застарілими.

Фактор 6: Часові гарантії $F_{time}(n)$. Цей фактор дає змогу враховувати здатність вузла виконувати задачі в межах встановленого часу:

$$F_{time}(n) = \max\left(F_{min}, \min\left(1, \frac{T_{timeout}}{T_{expected}(n)}\right)\right), \tag{4.8}$$

де $T_{timeout}$ – максимально допустимий час виконання задачі, а $T_{expected}(n)$ – очікуваний час виконання задачі вузлом n, розрахований на основі усередненого часу попередніх виконань. Схожий принцип використано в [84] для периферійних обчислень, де дотримання часових обмежень є ключовим критерієм якості розподілу задач. Якщо вузол систематично наближається до $T_{timeout}$ або перевищує його, фактор знижується, що зменшує ймовірність призначення задач повільному вузлу.

Обгрунтування мультиплікативного підходу. Запропонована скорингова функція (формула 4.2) використовує мультиплікативну композицію факторів, що відповідає моделі WPM з теорії багатокритеріального прийняття. Для поточної задачі мультиплікативна композиція має принципову перевагу перед адитивною (WSM) – вона обмежує компенсаційний ефект між факторами. При адитивній композиції

високе значення одного фактора може компенсувати низьке значення іншого, що призводить, наприклад, до вибору вузла з високою репутацією, але перевантаженим процесором. При мультиплікативній композиції $S = R \cdot F_{cpu} \cdot ...$ навіть один фактор, близький до F_{min} (наприклад, $F_{cpu} = 0{,}1$ при CPU = 90%), різко знижує загальну оцінку, що ефективно виключає перевантажений вузол з розгляду. Лінійна складність гарантує масштабованість для кластерів з десятками та сотнями вузлів при мінімальних обчислювальних витратах на блокчейн-вузлах.

Алгоритм getBestNode(). Процес вибору оптимального вузла формалізовано у вигляді Алгоритму 4.1, що реалізується як функція смарт-контракту IoTRegistry.

Алгоритм 4.1. Вибір оптимального вузла (getBestNode)

Input: N – множина зареєстрованих вузлів; Fmin, Fq_min – мінімальні допустимі значення факторів; Tm – поріг свіжості метрик; Ttimeout – максимально допустимий час виконання задачі
Output: bestNode – оптимальний вузол

```
bestScore ← 0
bestNode ← null

for each node n in N do
    // Обчислення 6 факторів (формули 4.3–4.8)
    r      ← R(n)                          // Репутація (4.3)
    f_cpu  ← max(Fmin,  1 − L_cpu(n))      // CPU (4.4)
    f_mem  ← max(Fmin,  1 − L_mem(n))      // Пам'ять (4.5)
    f_queue ← max(Fq_min, 1 − L_queue(n))  // Черга (4.6)
    f_fresh ← 1 if (t_now − t_last(n)) < Tm else Fmin // Свіжість (4.7)
    f_time ← max(Fmin, min(1, Ttimeout / T_exp(n))) // Час (4.8)

    // Мультиплікативна композиція (4.2)
    score ← r · f_cpu · f_mem · f_queue · f_fresh · f_time

    if score > bestScore then
        bestScore ← score
        bestNode ← n
    end if
end for

return bestNode
```

Запропонована мультифакторна скорингова функція дозволяє запобігти концентрації задач на перевантажених вузлах та автоматично враховує як історичну надійність, так і поточний стан ресурсів.

4.3. Механізми забезпечення надійності

Мультифакторна скорингова функція (формула 4.2) гарантує оптимальний вибір вузла на момент призначення задачі, проте не вирішує задач, що виникають після призначення: відмову вузла під час виконання, паралельне надходження транзакцій та підтримання цілісності результатів. У даному підрозділі описано механізми, що доповнюють скорингову функцію та створюють умови для наскрізної надійності системи.

Контроль часу. Контроль часу виконання задачі є критичним механізмом захисту від зависання або відмови периферійного вузла після призначення задачі. Кожна задача, створена через ComputeManager.createTask, отримує параметр timeout – максимально допустимий час виконання. Якщо периферійний вузол не надсилає результат протягом цього часу, задача переходить у стан «прострочена» та стає доступною для перепризначення.

Механізм перепризначення функціонує наступним чином. Будь-який вузол мережі може викликати функцію ComputeManager.checkTimeout для конкретної задачі. Якщо задачу прострочено, контракт генерує подію TaskStatusChanged та оновлює репутацію вузла-порушника: $T_a(n)$ збільшується на 1 без збільшення $T_s(n)$, що знижує репутацію $R(n)$ у формулі 4.3. Подія TaskStatusChanged слугує тригером для Python-агентів периферійних вузлів, які через механізм періодичного опитування (polling) виявляють зміну статусу та ініціюють повторний вибір виконавця. ComputeManager повторно викликає IoTRegistry.getBestNode для вибору нового вузла, враховуючи оновлену репутацію. Цей механізм інтегрується з фактором ємності

черги: перепризначена задача призначається вузлу з найменшим навантаженням, а не тому ж перевантаженому вузлу. З точки зору безпеки, механізм протидіє атакам, коли зловмисний вузол приймає задачу, проте навмисно затримує виконання для блокування ресурсів мережі.

Моніторинг ресурсів у реальному часі. Збір метрик використання процесору, пам'яті та стану черги задач здійснюється бібліотекою psutil (Python System and Process Utilities) – крос-платформенною бібліотекою для моніторингу системних ресурсів. Psutil обрано з кількох причин: підтримка Linux, macOS та Windows, мінімальне навантаження на CPU (< 0,1% при періодичному опитуванні), та уніфікований API для різних типів метрик [85].

Моніторинг виконується у окремому потоці з конфігурованою періодичністю оновлення метрик у 30 секунд, що є компромісом між актуальністю метрик та витратами на блокчейн-транзакції: частіше оновлення (наприклад, кожні 5 секунд) генерує надмірний потік транзакцій, тоді як рідше оновлення (кожні 2 хвилини) створює ризик прийняття рішень на основі застарілих даних. Зібрані метрики пакуються у транзакцію до IoTRegistry.updateMetrics та надсилаються до блокчейну. Час оновлення автоматично фіксується контрактом як block.timestamp, що підтримує актуальність фактору свіжості $F_{fresh}(n)$ у формулі (4.7).

Верифікація цілісності даних. Результати обчислень, виконаних поза блокчейном на периферійних вузлах, потребують механізму верифікації цілісності для запобігання підміні або модифікації даних. Для цього використовується хеш-функція Keccak-256 (стандартна хеш-функція Ethereum), що дає криптографічно стійке хешування результатів.

Механізм працює так. Після завершення обчислення периферійний вузол обчислює Keccak-256 хеш результату (наприклад, агрегованих даних сенсорів або результату аналітичного скрипту). Хеш надсилається до ComputeManager.submitResult разом із результатом та записується у блокчейн. Будь-який учасник мережі може

верифікувати коректність результату, повторно обчисливши хеш та порівнявши його із записаним значенням. Це не замінює повноцінну верифікацію правильності обчислень, проте гарантує цілісність: результат, що був модифікований після обчислення, буде виявлений за невідповідністю хешу. Для сценаріїв, що потребують підвищеної довіри, можлива інтеграція із доказом з нульовим розголошенням (ZKP), проте це виходить за межі поточного дослідження.

Окремого розгляду потребує модель довіри до метрик, що звітуються периферійним вузлом. У поточній реалізації значення доступності процесору, пам'яті та поточної довжини черги задач формуються на стороні вузла через бібліотеку psutil і передаються в смарт-контракт без криптографічної верифікації обчислювального стану. Формально це означає, що зловмисний периферійний вузол здатен завищувати декларовані вільні ресурси для отримання більшої кількості задач із подальшим їх провалом. Часткову протидію забезпечує реактивний механізм: невиконання задачі у межах визначеного часу зменшує репутацію R(n) вузла через штрафну компоненту скорингової функції, а повторні порушення переводять вузол у пасивний стан.

4.4. Програмна реалізація платформи мультифакторного управління

Запропоновану модель мультифакторного вибору вузлів реалізовано як програмну платформу, що включає два смарт-контракти на мові Solidity та Python-агент для периферійних вузлів. Даний підрозділ описує архітектуру компонентів, їхню взаємодію та систему подій, яка робить можливою автоматизацію на основі подій.

4.4.1. Смарт-контракти

Координаційну логіку мультифакторного управління реалізовано у двох смарт-контрактах, розгорнутих у приватній Ethereum-мережі IoT-кластера: IoTRegistry та ComputeManager. Перелік всіх подій у смарт контрактах приведено в таблиці 4.1.

Таблиця 4.1 – Перелік смарт-контрактних подій та їх призначення

Подія	**Контракт**	**Параметри**	**Призначення**
NodeRegistered	IoTRegistry	nodeAddress, timestamp	Реєстрація нового периферійного вузла
MetricsUpdated	IoTRegistry	nodeAddress, cpu, mem, queue	Оновлення метрик після виклику updateMetrics
ScriptRegistered	ComputeManager	scriptHash, description	Реєстрація нового обчислювального скрипту
TaskCreated	ComputeManager	taskId, scriptHash, creator	Створення нової задачі IoT-пристроєм
TaskAssigned	ComputeManager	taskId, nodeAddress, timeout	Призначення задачі обраному вузлу
TaskStatusChanged	ComputeManager	taskId, oldStatus, newStatus	Зміна статусу (pending → assigned → completed/expired)
ResultSubmitted	ComputeManager	taskId, nodeAddress, resultHash	Успішне завершення та верифікація результату

Контракт IoTRegistry відповідає за реєстрацію вузлів, зберігання метрик та обчислення оцінки вузла. Реєстрація відбувається через функцію registerNode – вона додає вузол до масиву зареєстрованих вузлів та ініціалізує його метрики початковими

значеннями. Для кожного вузла зберігається структура даних, що містить основні показники метрик.

Функція updateMetrics викликається периферійним вузлом при кожному циклі моніторингу. Вона оновлює поточні метрики, одночасно встановлюючи час останнього оновлення, що автоматично оновлює фактор актуальності $F_{fresh}(n)$. Ключова функція getBestNode реалізує Алгоритм 4.1. Обчислення здійснюються у цілочисельній арифметиці з масштабуванням, що є стандартною практикою для Solidity, де операції з плаваючою точкою не підтримуються.

Функція updateReputation викликається ComputeManager після завершення задачі. Вона оновлює кількість призначених та виконаних задач для вузла, що змінює фактор $R(n)$ у формулі (4.3).

Контракт ComputeManager реалізує створення задач, призначення вузлів та контроль виконання. Через функцію createTask створюється нова задача, зберігаються вхідні дані, після чого автоматично викликається IoTRegistry.getBestNode для вибору оптимального вузла.

Функція submitResult – її викликає периферійний вузол після завершення обчислення. Контракт верифікує, що результат надійшов від призначеного вузла, обчислює хеш результату та порівнює з наданим хешом для перевірки цілісності. Після успішної верифікації статус задачі змінюється на «completed», а контракт викликає IoTRegistry.updateReputation.

Функція checkTimeout перевіряє, чи перевищено час виконання для конкретної задачі. Якщо задача прострочена, присвоюється статус «expired», репутація вузла оновлюється, та ініціюється повторний вибір вузла.

4.4.2. Периферійні вузли

Периферійний вузол реалізовано як Python-застосунок з трьома основними функціями: моніторинг ресурсів, обробка блокчейн-подій та виконання

обчислювальних скриптів. В основі реалізації лежить асинхронна модель із використанням бібліотеки web3.py для взаємодії з блокчейном.

Модуль моніторингу ресурсів збирає метрики CPU, пам'яті та стану черги задач засобами бібліотеки psutil, як описано у підрозділі 4.3. Зібрані метрики додаються у транзакцію updateMetrics та надсилаються до контракту IoTRegistry через web3.py.

Модуль обробки подій здійснює підписку на контрактні події TaskAssigned через web3.py event filter. Коли надходить подія TaskAssigned, де поле assignedNode відповідає адресі поточного периферійного вузла, модуль завантажує вхідні дані задачі, ідентифікує відповідний обчислювальний скрипт за його хешом та передає його на виконання модулю обчислень.

Модуль обчислювальних скриптів реалізує інтерфейс *compute(input_data)* → *dict*, де *input_data* – вхідні дані задачі (наприклад, масив сенсорних значень), а результат – словник із результатами обчислення. Скрипти реєструються за хеш-значенням, зберігаються вони локально на периферійному вузлі. Після завершення обчислення модуль формує хеш з даних та надсилає транзакцію до ComputeManager.

4.4.3. Взаємодія компонентів

Взаємодія між IoT-пристроями, смарт-контрактами та периферійними вузлами побудована на ланцюзі подій, що дозволяє автоматичну координацію без централізованого оркестратора. Повний цикл обробки задачі проілюстровано на рис. 4.2.

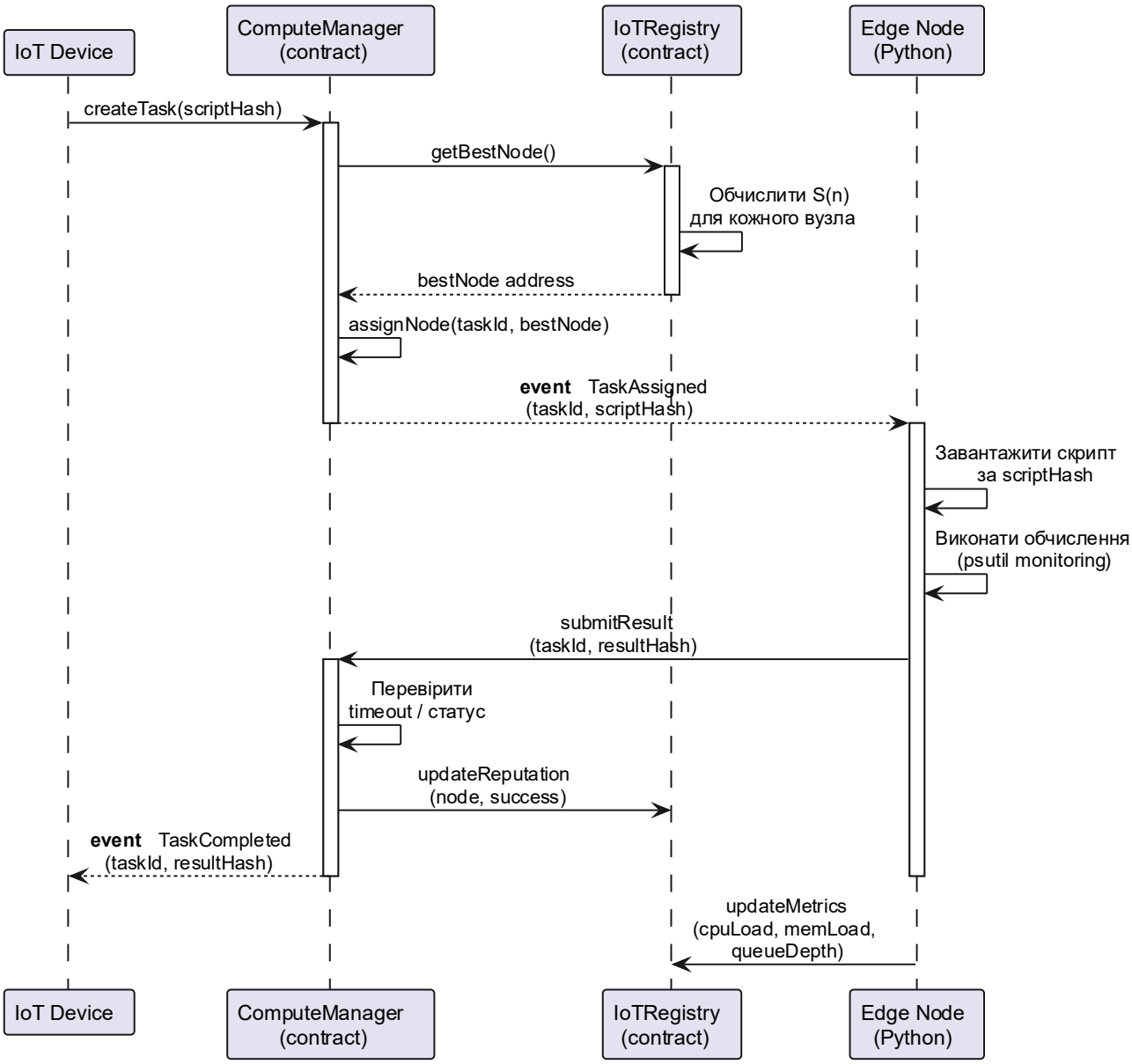

Рисунок 4.2 – Діаграма взаємодії компонентів при обробці обчислювальної задачі

Дана система подій формує слабку зв'язність між компонентами: периферійні вузли не потребують прямого з'єднання з IoT-пристроями, а лише підписки на контрактні події. Це спрощує масштабування – додавання нових периферійних вузлів не потребує перенастроювання існуючих. Водночас підвищується стійкість, оскільки відмова одного вузла не впливає на інших, а всі події записуються у блокчейн та залишаються незмінними, що забезпечує повний аудит.

4.5. Експериментальне дослідження

Тестова система складається з Docker-мережі з десятьма вузлами Go-Ethereum Clique та десятьма периферійними вузлами. Час формування блоку – три секунди, інтервал опитування вузлів – п'ять секунд. Кожен вузловий скрипт було налаштовано на симуляцію реальних сценаріїв з різним завантаженням процесору та пам'яті, тож вузли надсилали різні метрики. Кожна задача була налаштована на симуляцію споживання 20% процесору та 10% пам'яті у симульованих метриках, тоді як фактичною задачею був скрипт із п'ятихвилинною затримкою. Як наслідок, метрики відображали це додаткове навантаження, знижуючи підсумкову оцінку вузла.

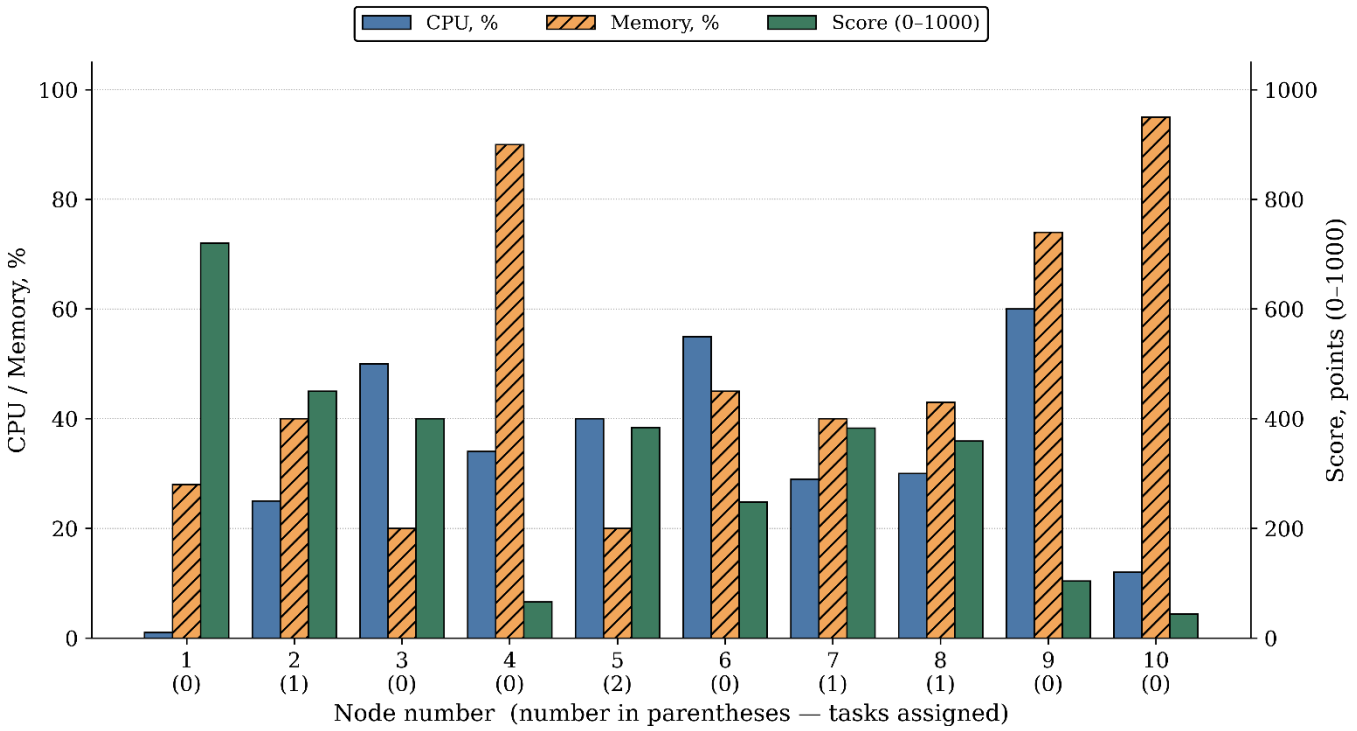

Рисунок 4.3 – Виміри, зафіксовані перед черговим призначенням задачі

Для тесту був реалізований окремий скрипт-генератор задач, що створював 20 задач у мінібатчах по п'ять задач щохвилини. Скрипт зберігав ідентифікатори задач і відстежував події початку та завершення. Після того як всі задачі завершувалися, скрипт отримував з блокчейну історію метрик, оцінки та інформацію про вузли для

подальшого аналізу. Рис. 4.3 показує знімок системи в конкретний момент перед призначенням задачі.

Тестування показало, що при одночасному поданні декількох задач у межах одного міжблокового інтервалу оцінка вузла не оновлюється як очікується, що може викликати перевантаження. Попри те, що оновлення оцінок ініціюються поданням задачі і також відбуваються автоматично, фактична оцінка може змінитися щонайбільше один раз за блок. Дану проблему можна вирішити через відкладене призначення: система має обирати найкращий вузол на основі наступного блоку, а не поточного. Це додає одноблокову затримку перед стартом задачі, проте підвищує надійність вибору.

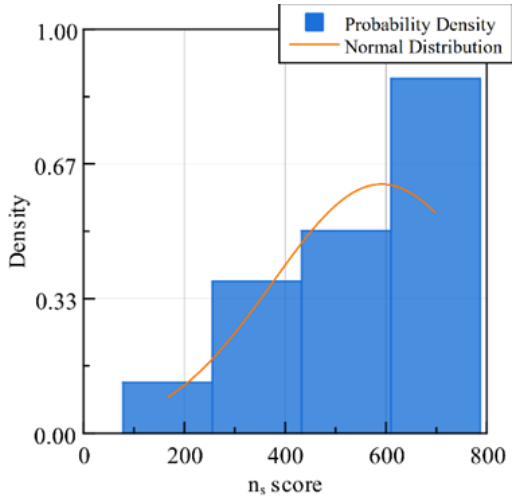

Рисунок 4.4 – Розподіл оцінок обраних вузлів

Оцінка фіксувалась перед кожним поданням задачі. Результати вказують, що у більшості випадків обираються вузли з вищими оцінками, що підтверджує теоретичний підхід з рівномірного розподілу навантаження між гетерогенними вузлами (рис. 4.4).

Можливі покращення. Мультиплікативна композиція (формула 4.2) породжує так званий «вето-ефект» – якщо хоча б один фактор наближається до мінімуму, загальна оцінка різко падає. Треба чітко враховувати в залежності від потреб середовища, які фактори мають бути підсилені, а які послаблені шляхом калібрування

констант з формул 4.3-4.8, що обмежує можливості автоматичного розгортання. Інтеграція методів прогнозування дала б змогу передбачати навантаження вузла на наступний період блоку і враховувати прогнозований стан при виборі, зменшуючи кількість «конфліктних» призначень при пакетній подачі задач. Також використання профілів задач у смарт-контракті дозволить адаптивно підлаштовувати фактори залежно від типу задачі.

4.6. Поєднання системи PoI та мультифакторного управління вузлами

Метод адаптивного консенсусу PoI (Розділ 3) та модель мультифакторного вибору вузлів (підрозділ 4.2) вирішують різні, але взаємодоповнюючі задачі координації вузлів в IoT-мережі. Комбінована система утворює дворівневу архітектуру управління вузлами. Кожен рівень оперує власним набором метрик та реалізує оптимізацію конкретного аспекту функціонування мережі.

Рівень консенсусу (PoI) працює з метриками ефективності генерації блоків. На основі цих метрик PoI визначає динамічний порядок підписантів та обмежує кількість in-turn вузлів, реалізуючи стабільну мережеву інфраструктуру з мінімальною кількістю форків та оптимізованим трафіком.

Прикладний рівень (мультифакторний вибір) оперує метриками ресурсів периферійних вузлів. Скорингова функція (формула 4.2) на основі цих метрик вибирає оптимальний обчислювальний вузол для кожної задачі, що реалізує розподіл навантаження, запобігання перевантаженню та мінімізацію часу виконання задач.

Ці два рівні є ортогональними за метриками та незалежними за механізмами, проте їхня комбінація створює синергетичний ефект, що перевищує суму окремих покращень. Жоден із рівнів окремо не дає повного покриття задач координації: PoI оптимізує мережу, але не враховує ресурси; мультифакторний вибір оптимізує ресурси, однак залежить від стабільності мережі. Таким чином, лише комбінація обох

рівнів дозволяє досягти комплексного вирішення задачі ефективної координації вузлів в IoT-мережі, що відповідає науковим цілям дисертації. Експериментальна валідація комбінованої системи виходить за межі цієї дисертації та становить напрям подальших досліджень.

4.7. Висновки

Теоретичний підхід з використанням методу зваженого добутку в мультифакторному виборі периферійних вузлів показує ефективність в експериментальному дослідженні внаслідок так званого «вето-ефекту». Лінійна складність моделі дозволяє обчислювати фінальну оцінку для кожного вузла в блокчейні.

Виявлене технічне обмеження мультиплікативної моделі при синхронній подачі задач (затримка оновлення оцінки у межах одного блоку) вирішується механізмом відкладеного призначення.

Удосконалена модель забезпечує такі атрибути якості згідно ISO/IEC 25010: ефективність використання ресурсів (Performance Efficiency / Resource Utilization) – вето-ефект перешкоджає концентрації задач на перевантажених вузлах та забезпечує рівномірний розподіл навантаження між гетерогенними периферійними вузлами; відмовостійкість (Reliability / Fault Tolerance) – фактор актуальності метрик автоматично виключає вузли, що припинили звітування, а механізм перепризначення повертає задачі у потік навіть при відмові призначеного вузла; безвідмовність (Reliability / Faultlessness) – репутаційна компонента штрафує вузли з історією невиконання задач та поступово знижує ймовірність їх повторного вибору.

Експериментальна валідація комбінованої системи «PoI та мультифакторне управління» становить напрям подальших досліджень.

Отримані результати разом з Розділами 2 і 3 підтверджують **досягнення загальної мети дисертації** – підвищення ефективності блокчейн-орієнтованих мереж IoT через гібридну багаторівневу архітектуру, адаптивний консенсус PoI та метод мультифакторного управління обчислювальними вузлами. Ці наукові результати утворюють цілісну систему з взаємодоповнюючих компонентів.

ВИСНОВКИ

У дисертації вирішено актуальне науково-технічне завдання підвищення ефективності функціонування блокчейн-орієнтованих мереж IoT за категоріями Performance Efficiency та Reliability згідно стандарту ISO/IEC 25010:2023.

1. Запропонована гібридна багаторівнева архітектура задовольняє сформовані внаслідок аналізу вимоги до блокчейн IoT-мереж та дозволяє поєднати в одній мережі автономні кластери гетерогенних пристроїв та зовнішню масштабовану інфраструктуру.

2. Реалізація моделі динамічного вибору вузлів та оптимізації блоків у методі адаптивного консенсусу PoI підвищує ефективність роботи блокчейн мережі в порівнянні з Clique у відповідності зі стандартом ISO/IEC 25010:2023.

3. Рішення мультифакторного вибору вузлів дозволяє забезпечити зниження оцінки вузлів за критично низьким значенням одного з факторів: репутації вузла, доступності ресурсів процесору та пам’яті, ємність черги задач, актуальність метрик та часових гарантій. Внаслідок цього вдається уникнути призначення задач на перевантажений вузол та знаходити найбільш придатний вузол для виконання задачі.

4. Запропонований підхід поєднання PoI та мультифакторного вибору доволяє вирішити задачі координації на мережевому та ресурсному рівнях: PoI підвищує показник безвідмовності мережі (зменшення числа форків на 80%), тоді як мультифакторний вибір периферійних вузлів оптимізує розподіл задач.

Розроблені програмні засоби реалізовано як готові до інтеграції компоненти, що відкриває можливість їх безпосереднього впровадження у спеціалізовані блокчейн-мережі IoT. Рекомендації щодо вибору блокчейн-платформ можуть слугувати методичною основою для проектування подібних систем, а отримані моделі та програмні засоби – навчальними матеріалами для курсів з комп'ютерної інженерії.

Таким чином, мета дисертаційного дослідження досягнута. Експериментально підтверджено підвищення ефективності функціонування блокчейн-орієнтованих мереж IoT за категоріями продуктивності та надійності стандарту ISO/IEC 25010:2023:

- **часові характеристики** (Performance Efficiency / Time Behaviour) скорочення часу поширення блоків на 8,5% (Розділ 3); фактор часових гарантій у скоринговій функції враховує очікуваний час виконання задач та обмежує призначення задач вузлам, що систематично перевищують допустимий час (Розділ 4);
- **використання ресурсів** (Performance Efficiency / Resource Utilization) зменшення мережевого трафіку на 20,5% при невеликому зростанні навантаження на процесор (6,5%) та пам'ять (5,4%) (Розділ 3); рівномірний розподіл обчислювальних задач між гетерогенними периферійними вузлами на основі вето-ефекту мультиплікативної скорингової функції (Розділ 4);
- **пропускна здатність** (Performance Efficiency / Capacity) – збільшення пропускної здатності до 10% (Розділ 3)
- **безвідмовність** (Reliability / Faultlessness) – скорочення кількості розгалужень ланцюга на 80% завдяки динамічному вибору валідаторів та конфігуровній затримці позачергових вузлів PoI (Розділ 3); репутаційна компонента скорингової функції штрафує вузли з історією невиконання задач (Розділ 4);

Експериментальне дослідження дворівневої комбінованої системи «PoI та мультифакторне управління» становить предмет подальших досліджень.

Перспективи подальших досліджень охоплюють кілька напрямів. Перший – впровадження предиктивних моделей оцінки стану вузлів на основі методів прогнозування для зменшення кількості конфліктних призначень при одночасній batch-подачі задач. Другий – розроблення адаптивних ваг факторів скорингової функції, які б налаштовувалися залежно від типу задачі. Третій – інтеграція PoI з механізмом шардингу для масштабування на множину IoT-кластерів зі збереженням лінійної складності у межах кожного кластера. Четвертий напрям – дослідження

інтеграції доказів без розкриття інформації (zero-knowledge proof) або верифікованих облікових даних (VC) у процес верифікації результатів обчислень периферійних вузлів. Окремо – адаптація комбінованої системи для сценаріїв федеративного навчання в IoT-мережах, де блокчейн координує розподілене навчання моделей між периферійними вузлами з мультифакторним вибором учасників на основі обчислювальних ресурсів та якості локальних даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] IoT Analytics, “State of IoT — Spring 2025.” [Online]. Available: https://iot-analytics.com/number-connected-iot-devices/

[2] International Organization for Standardization, “Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Product quality model,” International Organization for Standardization, Geneva, Switzerland, Standard ISO/IEC 25010:2023, 2023. [Online]. Available: https://www.iso.org/standard/78176.html

[3] Л. В. Чепель and Ю. В. Бойко, “Підхід до безпеки та організації мереж IoT з використанням блокчейн технології,” *Вісник Вінницького Політехнічного Інституту*, no. 4, pp. 129–138, Aug. 2024, doi: 10.31649/1997-9266-2024-175-4-129-138.

[4] Л. В. Чепель and Ю. В. Бойко, “Побудова мереж IoT військового призначення з використанням блокчейн технології,” *Inf. Technol. Comput. Sci. Softw. Eng. Cyber Secur.*, pp. 186–194, Jun. 2025, doi: 10.32782/IT/2025-2-19.

[5] L. Chepel and Y. Boyko, “Proof-of-Indicators: development and validation of an adaptive consensus mechanism for Internet of Things blockchain networks,” *Technol. Audit Prod. Reserv.*, vol. 2, no. 2(88), pp. 15–24, Apr. 2026, doi: 10.15587/2706-5448.2026.356851.

[6] L. Chepel and Y. Boyko, “Multi-factor hybrid approach for Blockchain-enabled IoT edge computing,” *Comput. Syst. Inf. Technol.*, no. 1, pp. 78–87, Mar. 2026, doi: 10.31891/csit-2026-1-8.

[7] L. Chepel and Y. Boyko, “Improving IoT with Blockchain: Consensus and Energy Efficiency,” in *Proceedings of the XIX International Scientific Conference Electronics and Applied Physics (APHYS 2023)*, Kyiv, Ukraine: Taras Shevchenko National University of Kyiv, Faculty of RadioPhysics, Electronics and Computer Systems, Oct. 2023, p. 76.

[8] V. Sribnyi, L. Chepel, and Y. Boyko, “Decentralization of Information Processing in IoT Infrastructure,” in *XXIV International Young Scientists Conference on Applied Physics (ICAP 2024)*, Kyiv, Ukraine: Taras Shevchenko National University of Kyiv, Faculty of RadioPhysics, Electronics and Computer Systems, May 2024, p. 127.

[9] Л. В. Чепель and Ю. В. Бойко, “Автоматизація процесів в IoT з використанням Blockchain,” in *Матеріали II Всеукраїнської науково-практичної конференції з міжнародною участю «PROKYIV»*, Київ, Україна: КНДУ, Науково-дослідний інститут соціально-економічного розвитку міста, 2025, pp. 479–480.

[10] L. Chepel and Y. Boyko, “Blockchain as the Next Evolution Step for IoT,” in *Proceedings of the 5th International Scientific and Practical Conference “Scientific Exploration: Bridging Theory and Practice,”* Berlin, Germany: European Open Science Space, Oct. 2025, pp. 65–67. doi: 10.70286/EOSS-20.10.2025.

[11] В. О. Срібний, Л. В. Чепель, and Ю. В. Бойко, “Застосування ланцюга блоків для опрацювання даних IMP,” in *Матеріали XV Міжнародної конференції з прикладної*

біофізики, біоніки та біокібернетики, Київ, Україна: Інститут магнетизму НАН України та МОН України, Apr. 2024, pp. 37–38.

[12] R. Minerva, A. Biru, and D. Rotondi, "Towards a definition of the Internet of Things (IoT)," Telecom Italia S.p.A. and Politecnico di Torino, Technical Report, 2015. [Online]. Available: https://iot.ieee.org/definition.html

[13] H. Kuchuk and E. Malokhvii, "INTEGRATION OF IOT WITH CLOUD, FOG, AND EDGE COMPUTING: A REVIEW," *Adv. Inf. Syst.*, vol. 8, no. 2, pp. 65–78, Jun. 2024, doi: 10.20998/2522-9052.2024.2.08.

[14] K. Cao, Y. Liu, G. Meng, and Q. Sun, "An Overview on Edge Computing Research," *IEEE Access*, vol. 8, pp. 85714–85728, 2020, doi: 10.1109/ACCESS.2020.2991734.

[15] "2024 IoT Security Crisis: By The Numbers," Viakoo, Inc., Mountain View, CA, USA, Technical Report, 2024. [Online]. Available: https://www.viakoo.com/2024-iot-security-crisis/

[16] M. M. Ogonji, G. Okeyo, and J. M. Wafula, "A survey on privacy and security of Internet of Things," *Comput. Sci. Rev.*, vol. 38, p. 100312, Nov. 2020, doi: 10.1016/j.cosrev.2020.100312.

[17] J. Opara-Martins, R. Sahandi, and F. Tian, "Critical analysis of vendor lock-in and its impact on cloud computing migration: a business perspective," *J. Cloud Comput.*, vol. 5, no. 1, p. 4, Apr. 2016, doi: 10.1186/s13677-016-0054-z.

[18] Ł. Pióro, J. Sychowiec, K. Kanciak, and Z. Zieliński, "Application of Attribute-Based Encryption in Military Internet of Things Environment," *Sensors*, vol. 24, no. 18, Sep. 2024, doi: 10.3390/s24185863.

[19] "A Survey of IoT Applications in Blockchain Systems: Architecture, Consensus, and Traffic Modeling: ACM Computing Surveys: Vol 53, No 1." Accessed: Apr. 29, 2026. [Online]. Available: https://dl.acm.org/doi/10.1145/3372136

[20] M. Hossain, G. Kayas, R. Hasan, A. Skjellum, S. Noor, and S. M. R. Islam, "A Holistic Analysis of Internet of Things (IoT) Security: Principles, Practices, and New Perspectives," *Future Internet*, vol. 16, no. 2, Jan. 2024, doi: 10.3390/fi16020040.

[21] Z. Alansari, N. B. Anuar, A. Kamsin, and M. R. Belgaum, "A systematic review of routing attacks detection in wireless sensor networks," *PeerJ Comput. Sci.*, vol. 8, p. e1135, Oct. 2022, doi: 10.7717/peerj-cs.1135.

[22] O. I. Abiodun, E. O. Abiodun, M. Alawida, R. S. Alkhawaldeh, and H. Arshad, "A Review on the Security of the Internet of Things: Challenges and Solutions," *Wirel. Pers. Commun.*, vol. 119, no. 3, pp. 2603–2637, Aug. 2021, doi: 10.1007/s11277-021-08348-9.

[23] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: Mirai and Other Botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017, doi: 10.1109/MC.2017.201.

[24] N. Asokan *et al.*, "SEDA: Scalable Embedded Device Attestation," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, in CCS '15. New York, NY, USA: Association for Computing Machinery, Oct. 2015, pp. 964–975. doi: 10.1145/2810103.2813670.

[25] A. Ghobakhlou, D. Z. Al-Hamid, S. Zandi, and J. Cato, “A Comprehensive Analysis of Security Challenges in ZigBee 3.0 Networks,” *Sensors*, vol. 25, no. 15, Jul. 2025, doi: 10.3390/s25154606.

[26] A. Alomari and S. A. P. Kumar, “Securing IoT systems in a post-quantum environment: Vulnerabilities, attacks, and possible solutions,” *Internet Things*, vol. 25, p. 101132, Apr. 2024, doi: 10.1016/j.iot.2024.101132.

[27] “CVE: Common Vulnerabilities and Exposures for MQTT.” Accessed: Mar. 08, 2026. [Online]. Available: https://www.cve.org/CVERecord/SearchResults?query=MQTT

[28] I. Petrescu, E. Niculae, V. Vulturescu, A. Dimitrescu, and L. M. Ungureanu, “Transport and Application Layer Protocols for IoT: Comprehensive Review,” *Technologies*, vol. 13, no. 12, Dec. 2025, doi: 10.3390/technologies13120583.

[29] I. Ullah and P. J. M. Havinga, “Governance of a Blockchain-Enabled IoT Ecosystem: A Variable Geometry Approach,” *Sensors*, vol. 23, no. 22, Nov. 2023, doi: 10.3390/s23229031.

[30] X. Xu, I. Weber, and M. Staples, *Architecture for Blockchain Applications*. Cham: Springer International Publishing, 2019. doi: 10.1007/978-3-030-03035-3.

[31] S. Rouhani and R. Deters, “Security, Performance, and Applications of Smart Contracts: A Systematic Survey,” *IEEE Access*, vol. 7, pp. 50759–50779, 2019, doi: 10.1109/ACCESS.2019.2911031.

[32] V. Buterin, “Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform.” Accessed: Nov. 01, 2025. [Online]. Available: https://ethereum.org/en/whitepaper

[33] “Proof-of-stake (PoS),” ethereum.org. Accessed: Jan. 29, 2026. [Online]. Available: https://ethereum.org/developers/docs/consensus-mechanisms/pos/

[34] V. Buterin and V. Griffith, “Casper the Friendly Finality Gadget,” arXiv.org. Accessed: Apr. 29, 2026. [Online]. Available: https://arxiv.org/abs/1710.09437v4

[35] P. Szilágyi, “EIP-225: Clique proof-of-authority consensus protocol,” no. 225. Mar. 2017. [Online]. Available: https://eips.ethereum.org/EIPS/eip-225

[36] A. Ahmad, A. Alabduljabbar, M. Saad, D. Nyang, J. Kim, and D. Mohaisen, “Empirically comparing the performance of blockchain’s consensus algorithms,” *IET Blockchain*, vol. 1, no. 1, pp. 56–64, 2021, doi: 10.1049/blc2.12007.

[37] “Proof-of-authority (PoA),” ethereum.org. Accessed: Apr. 29, 2026. [Online]. Available: https://ethereum.org/developers/docs/consensus-mechanisms/poa/

[38] Kopanitsa, “Web3 Arduino: An Arduino (or ESP32) library to use web3 on Ethereum platform.” GitHub repository. [Online]. Available: https://github.com/kopanitsa/web3-arduino

[39] Md. M. Islam, M. M. Merlec, and H. P. In, “A Comparative Analysis of Proof-of-Authority Consensus Algorithms: Aura vs Clique,” in *2022 IEEE International Conference on Services Computing (SCC)*, Jul. 2022, pp. 327–332. doi: 10.1109/SCC55611.2022.00054.

[40] "Hyperledger Fabric Documentation." Accessed: Apr. 29, 2026. [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/release-2.5/

[41] A. Vera-Rivera and E. Hossain, "Decentralizing Trust: Consortium Blockchains and Hyperledger Fabric Explained," Feb. 10, 2025, *arXiv*: arXiv:2502.06540. doi: 10.48550/arXiv.2502.06540.

[42] M. Rasori, M. L. Manna, P. Perazzo, and G. Dini, "A Survey on Attribute-Based Encryption Schemes Suitable for the Internet of Things," *IEEE Internet Things J.*, vol. 9, no. 11, pp. 8269–8290, Jun. 2022, doi: 10.1109/JIOT.2022.3154039.

[43] S. Popov, "The Tangle," 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:4958428

[44] "IOTA Documentation." Accessed: Apr. 29, 2026. [Online]. Available: https://docs.iota.org/

[45] S. Müller, A. Penzkofer, N. Polyanskii, J. Theis, W. Sanders, and H. Moog, "Tangle 2.0 Leaderless Nakamoto Consensus on the Heaviest DAG," *IEEE Access*, vol. 10, pp. 105807–105842, 2022, doi: 10.1109/ACCESS.2022.3211422.

[46] M. Raikwar, N. Polyanskii, and S. Müller, "SoK: DAG-based Consensus Protocols," in *2024 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, May 2024, pp. 1–18. doi: 10.1109/ICBC59979.2024.10634358.

[47] "IoTeX Whitepaper," GitHub. Accessed: Apr. 29, 2026. [Online]. Available: https://github.com/iotexproject/files/blob/main/publications/IoTeX_Whitepaper_1.5_EN.pdf

[48] WaltonChain, "WaltonChain WhitePaper." Accessed: Apr. 29, 2026. [Online]. Available: https://github.com/WaltonChain/WhitePaper

[49] "IBM Watson IoT Platform." Accessed: Apr. 30, 2026. [Online]. Available: https://www.ibm.com/docs/en/www.ibm.com/docs/en/watson-iot-platform

[50] G. Zhang, G. Posluns, and M. C. Jeffrey, "Multi Bucket Queues: Efficient Concurrent Priority Scheduling," in *Proceedings of the 36th ACM Symposium on Parallelism in Algorithms and Architectures*, in SPAA '24. New York, NY, USA: Association for Computing Machinery, Jun. 2024, pp. 113–124. doi: 10.1145/3626183.3659962.

[51] S. M. H. Bamakan, A. Motavali, and A. Babaei Bondarti, "A survey of blockchain consensus algorithms performance evaluation criteria," *Expert Syst. Appl.*, vol. 154, p. 113385, Sep. 2020, doi: 10.1016/j.eswa.2020.113385.

[52] L. Lao, Z. Li, S. Hou, B. Xiao, S. Guo, and Y. Yang, "A Survey of IoT Applications in Blockchain Systems: Architecture, Consensus, and Traffic Modeling," *ACM Comput Surv*, vol. 53, no. 1, p. 18:1-18:32, Feb. 2020, doi: 10.1145/3372136.

[53] R. Chen and Y. Guan, "Hierarchical Reputation Consensus Model for VANET," *IEEE Access*, vol. 13, pp. 3521–3531, 2025, doi: 10.1109/ACCESS.2024.3525188.

[54] S. Wadhwa, S. Rani, Kavita, S. Verma, J. Shafi, and M. Wozniak, "Energy Efficient Consensus Approach of Blockchain for IoT Networks with Edge Computing," *Sensors*, vol. 22, no. 10, May 2022, doi: 10.3390/s22103733.

[55] E. U. Haque *et al.*, “Performance enhancement in blockchain based IoT data sharing using lightweight consensus algorithm,” *Sci. Rep.*, vol. 14, no. 1, p. 26561, Nov. 2024, doi: 10.1038/s41598-024-77706-x.

[56] V. Upadhyay, Dr. A. Vaish, and Dr. J. Kokila, “The need for Lightweight Consensus algorithms in IoT environment: A review,” in *Proceedings of the 2024 Sixteenth International Conference on Contemporary Computing*, in IC3-2024. New York, NY, USA: Association for Computing Machinery, Oct. 2024, pp. 366–376. doi: 10.1145/3675888.3676072.

[57] Q. Gao, J. Xiao, Y. Cao, S. Deng, C. Ouyang, and Z. Feng, “Blockchain-based collaborative edge computing: efficiency, incentive and trust,” *J. Cloud Comput.*, vol. 12, no. 1, p. 72, May 2023, doi: 10.1186/s13677-023-00452-4.

[58] S. Asaithambi, S. Nallusamy, J. Yang, S. Prajapat, G. Kumar, and P. S. Rathore, “A secure and trustworthy blockchain-assisted edge computing architecture for industrial internet of things,” *Sci. Rep.*, vol. 15, no. 1, p. 15410, May 2025, doi: 10.1038/s41598-025-00337-3.

[59] X. Xu, X. Zhang, H. Gao, Y. Xue, L. Qi, and W. Dou, “BeCome: Blockchain-Enabled Computation Offloading for IoT in Mobile Edge Computing,” *IEEE Trans. Ind. Inform.*, vol. 16, no. 6, pp. 4187–4195, Jun. 2020, doi: 10.1109/TII.2019.2936869.

[60] H. Taherdoost and M. Madanchian, “Multi-Criteria Decision Making (MCDM) Methods and Concepts,” *Encyclopedia*, vol. 3, no. 1, pp. 77–87, Jan. 2023, doi: 10.3390/encyclopedia3010006.

[61] A. Bašić *et al.*, “Multi-Criteria Decision Analysis of Wireless Technologies in WPANs for IoT-Enabled Smart Buildings in Tourism,” *Buildings*, vol. 14, no. 10, Oct. 2024, doi: 10.3390/buildings14103275.

[62] G. Nieto, I. de la Iglesia, U. Lopez-Novoa, and C. Perfecto, “Deep Reinforcement Learning techniques for dynamic task offloading in the 5G edge-cloud continuum,” *J. Cloud Comput. Adv. Syst. Appl.*, vol. 13, no. 1, pp. 1–24, May 2024, doi: 10.1186/s13677-024-00658-0.

[63] R. K. Dewi, B. T. Hanggara, and A. Pinandito, “A Comparison Between AHP and Hybrid AHP for Mobile Based Culinary Recommendation System,” *Int. J. Interact. Mob. Technol. IJIM*, vol. 12, no. 1, pp. 133–140, Jan. 2018, doi: 10.3991/ijim.v12i1.7561.

[64] L. Boubekri, H. Aberkane, M. C. Abounaima, and L. Lamrini, “GPU-TOPSIS: A Complete Vectorized and Parallel Reformulation of the TOPSIS Method for Large-Scale Multi-Criteria Decision Making,” *Big Data Cogn. Comput.*, Apr. 2026, doi: 10.3390/bdcc10050138.

[65] S. Pal, A. Dorri, and R. Jurdak, “Blockchain for IoT Access Control: Recent Trends and Future Research Directions,” Jun. 09, 2021, *arXiv*: arXiv:2106.04808. doi: 10.48550/arXiv.2106.04808.

[66] G. Rafaiani, P. Santini, M. Baldi, and F. Chiaraluce, “Implementation of Ethereum Accounts and Transactions on Embedded IoT Devices,” Jun. 29, 2022, *arXiv*: arXiv:2206.14782. doi: 10.48550/arXiv.2206.14782.

[67] A. Rashid and A. U. R. Khan, “Blockchain-Based Autonomous Authentication and Integrity for Internet of Battlefield Things in C3I System,” *IEEE Access*, vol. 10, pp. 91572–91587, 2022, doi: 10.1109/ACCESS.2022.3201815.

[68] “Decentralized Identifiers (DIDs).” Accessed: Apr. 30, 2026. [Online]. Available: https://www.w3.org/TR/did-1.0/

[69] S. L. Nita and M. I. Mihailescu, “A Novel Authentication Scheme Based on Verifiable Credentials Using Digital Identity in the Context of Web 3.0,” *Electronics*, vol. 13, no. 6, Mar. 2024, doi: 10.3390/electronics13061137.

[70] C. D. N. Kyriakidou, A. M. Papathanasiou, I. Pittaras, N. Fotiou, Y. Thomas, and G. C. Polyzos, “Attribute-Based Access Control Utilizing Verifiable Credentials for Multi-Tenant IoT Systems,” in *2024 IEEE 4th International Conference on Electronic Communications, Internet of Things and Big Data (ICEIB)*, Apr. 2024, pp. 57–62. doi: 10.1109/ICEIB61477.2024.10602646.

[71] Espressif Systems, *ESP32 Technical Reference Manual*. Espressif Systems, 2024. [Online]. Available: https://documentation.espressif.com/esp32_technical_reference_manual_en.pdf

[72] S. S. Hameedi and O. Bayat, “Improving IoT Data Security and Integrity Using Lightweight Blockchain Dynamic Table,” *Appl. Sci.*, vol. 12, no. 18, Sep. 2022, doi: 10.3390/app12189377.

[73] E. Barker, “Recommendation for Key Management: Part 1 – General,” National Institute of Standards and Technology, NIST Special Publication (SP) 800-57 Part 1 Rev. 5, May 2020. doi: 10.6028/NIST.SP.800-57pt1r5.

[74] Arduino Crypto Project, “Arduino Crypto.” [Online]. Available: https://rweather.github.io/arduinolibs/crypto.html

[75] Mbed TLS Project, “MbedTLS.” [Online]. Available: https://github.com/Mbed-TLS/mbedtls

[76] ProtectedAES Project, “ProtectedAES.” [Online]. Available: https://github.com/RaffaeleMorganti/protectedAES

[77] PSA Crypto Project, “PSA Crypto.” [Online]. Available: https://github.com/machinefi/psa-crypto-arduino

[78] TinyECC Project, “TinyECC.” [Online]. Available: https://github.com/ShubhamAnnigeri/tinyECC-ArduinoIDE/tree/main

[79] J. Benet, “IPFS - Content Addressed, Versioned, P2P File System,” Jul. 14, 2014, *arXiv*: arXiv:1407.3561. doi: 10.48550/arXiv.1407.3561.

[80] L. T. Thibault, T. Sarry, and A. S. Hafid, “Blockchain Scaling Using Rollups: A Comprehensive Survey,” *IEEE Access*, vol. 10, pp. 93039–93054, 2022, doi: 10.1109/ACCESS.2022.3200051.

[81] M. Corallo, “BIP 152: Compact Block Relay.” Apr. 2016. [Online]. Available: https://github.com/bitcoin/bips/blob/master/bip-0152.mediawiki

[82] Go-Ethereum, “Official Go implementation of the Ethereum protocol.” 2024. [Online]. Available: https://github.com/ethereum/go-ethereum

[83] M. Alasmar, R. Clegg, N. Zakhleniuk, and G. Parisis, “Internet Traffic Volumes are Not Gaussian—They are Log-Normal: An 18-Year Longitudinal Study With Implications for Modelling and Prediction,” *IEEEACM Trans. Netw.*, vol. 29, no. 3, pp. 1266–1279, Jun. 2021, doi: 10.1109/TNET.2021.3059542.

[84] Y. Zhang, B. Tang, J. Luo, and J. Zhang, “Deadline-Aware Dynamic Task Scheduling in Edge–Cloud Collaborative Computing,” *Electronics*, vol. 11, no. 15, Aug. 2022, doi: 10.3390/electronics11152464.

[85] G. Rodolà, “psutil documentation. Cross-platform library for process and system monitoring in Python.” 2024. [Online]. Available: https://psutil.readthedocs.io/