ISSN 2219-9365

https://doi.org/10.31891/2219-9365-2025-83-58
UDC 004.51

ANTONENKO Artem
National University of Life and Environmental Sciences of Ukraine
https://orcid.org/0000-0001-9397-1209
e-mail: artem.v.antonenko@gmail.com
TVERDOKHLIB Arsenii
State University of Information and Communication Technologies
https://orcid.org/0000-0002-6591-2866

APPLICATION OF SMART CONTRACTS AND BLOCKCHAIN TECHNOLOGY FOR ENHANCING EFFICIENCY OF DISTRIBUTED BROKER SYSTEMS UNDER HIGH LOAD CONDITIONS

With the growing number of high-load computing systems, the issue of efficient device interaction and reliable data transmission is increasingly relevant. A widely used communication model, the publish-subscribe pattern, despite its efficiency, suffers from significant drawbacks due to reliance on centralized brokers. This approach introduces critical risks, including single points of failure and data security vulnerabilities stemming from potential data manipulation. This article introduces and thoroughly describes the distributed broker system Trinity, which integrates the publish-subscribe model with blockchain technology and smart contracts. Trinity leverages a distributed ledger and consensus algorithm, ensuring the reliability and immutability of transmitted data. A distinctive feature of the platform is its automated data verification and management using smart contracts, executed before data storage in the blockchain ledger. The Trinity platform was implemented using the Mosquitto MQTT broker and Tendermint blockchain framework, achieving high scalability and security. Experimental evaluation conducted on a network of 20 Raspberry Pi 3 nodes demonstrated that Trinity effectively addresses trust-related challenges under high-load conditions, though introducing certain overheads. Specifically, end-to-end message latency increased (up to 3.7 seconds), accompanied by higher network and computational resource usage. However, these overheads are justified by significant improvements in reliability, security, and fault tolerance, essential in distributed environments involving multiple independent participants. Therefore, the Trinity platform is particularly suitable for application areas requiring robust reliability and automated device interaction, such as smart cities, industrial IoT networks, and other critical infrastructures.

Keywords: blockchain, smart contract, distributed broker, high-load systems, publish-subscribe model.

АНТОНЕНКО Артем
Національний університет біоресурсів і природокористування України
ТВЕРДОХЛІБ Арсеній
Державний університет інформаційно-комунікаційних технологій

ВИКОРИСТАННЯ СМАРТ-КОНТРАКТІВ ТА ТЕХНОЛОГІЇ БЛОКЧЕЙН ДЛЯ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ РОЗПОДІЛЕНИХ БРОКЕРНИХ СИСТЕМ В УМОВАХ ВИСОКИХ НАВАНТАЖЕНЬ

Зі зростанням кількості високонавантажених комп'ютерних систем особливо актуальною стає проблема ефективної взаємодії пристроїв та забезпечення надійності передачі інформації. Однією з поширених моделей комунікації є «публікація-підписка», яка, незважаючи на свою ефективність, має суттєві недоліки через використання централізованого брокера. Це створює ризики виникнення центральних точок відмови, а також загрози безпеці даних через можливість їх підробки. У статті запропоновано та детально описано розподілену брокерну систему Trinity, що інтегрує модель «публікація-підписка» з технологією блокчейн та смарт-контрактами. Система Trinity базується на використанні розподіленого реєстру і алгоритму консенсусу, що забезпечує високу надійність і незмінність переданих даних. Ключовою особливістю платформи є автоматизована перевірка та керування даними за допомогою смарт-контрактів, що виконуються перед їх записом до блокчейн-реєстру. Реалізація платформи Trinity здійснювалась з використанням брокера MQTT Mosquitto та платформи блокчейну Tendermint, що дозволило досягти високого рівня масштабованості та безпеки. Проведене експериментальне дослідження на мережі Raspberry Pi 3 з 20 вузлами показало, що платформа Trinity ефективно вирішує задачі забезпечення довіри в умовах високих навантажень, хоча й додає певні накладні витрати. Зокрема, збільшується наскрізна затримка передачі повідомлень (до 3,7 секунд), зростають витрати мережевих та обчислювальних ресурсів. Проте ці витрати компенсуються підвищенням надійності, безпеки та відмовостійкості системи, що має важливе значення у розподілених середовищах, де взаємодіють кілька незалежних сторін. Таким чином, платформа Trinity може бути ефективно використана у сферах, які вимагають високої надійності та автоматизації взаємодії пристроїв, таких як інтелектуальні міста, індустріальні мережі IoT, та інші критичні інфраструктури.

Ключові слова: блокчейн, смарт-контракт, розподілений брокер, високонавантажені системи, модель «публікація-підписка».

Стаття надійшла до редакції / Received 01.08.2025
Прийнята до друку / Accepted 22.08.2025

STATEMENT OF THE PROBLEM IN A GENERAL FORM AND ITS CONNECTION WITH IMPORTANT SCIENTIFIC OR PRACTICAL TASKS

Modern applications frequently employ messaging models such as "publish-subscribe" and "request-response" to facilitate data exchange between end devices, edge devices, and servers. The "request-response"

messaging model has been extensively tested and standardized within Internet environments; however, it does not ideally suit systems constrained by limited resources due to inherent limitations, including relatively high overhead, latency, and limited scalability in large-scale distributed systems.

Conversely, the "publish-subscribe" model has gained popularity as an alternative, primarily because it offers low communication overhead, effective use of resources, and efficient scalability. This model involves publishers sending data to brokers, which then disseminate the data to subscribers based on predefined topics. Despite its evident benefits, traditional implementations of the publish-subscribe model usually depend on centralized brokers. This dependence creates significant vulnerabilities, including potential single points of failure, risks of service interruptions, and threats to data security and integrity due to centralized control.

With the rise of distributed, high-load systems and applications, there is a growing need for robust, secure, and scalable communication mechanisms capable of handling extensive data exchanges without compromising reliability or resource efficiency. Blockchain technology, characterized by its decentralized architecture, immutability, and inherent security mechanisms, presents promising opportunities to overcome the limitations of traditional centralized messaging systems.

ANALYSIS OF RECENT SOURCES

Currently, numerous sources address the use of blockchain technology. Recent studies and publications provide comprehensive insights into how blockchain technology helps resolve various issues in industry and other sectors [10, 11, 12]. Significant contributions toward resolving fundamental blockchain-related problems have been made by both domestic and international researchers [6, 9, 13].

Blockchain technology has attracted considerable attention across diverse fields due to its decentralized nature, robust security mechanisms, and ability to guarantee data immutability and transparency [1, 2, 10]. Researchers have extensively explored blockchain applications in finance, healthcare, supply chain management, Internet of Things (IoT), and cybersecurity [11, 12, 13]. In particular, blockchain's unique capability to manage distributed transactions transparently and securely is highlighted [1, 2].

Recent literature emphasizes blockchain’s applicability in scenarios requiring high trust among participants who may lack inherent trust relationships. The integration of blockchain with IoT has been a focal point, particularly regarding device interactions, secure data exchanges, and automated trust management through smart contracts [2, 13]. Additionally, extensive research has been conducted on consensus algorithms, analyzing their impact on system performance, scalability, and resource consumption [3, 6, 7].

Moreover, several studies have investigated the integration of blockchain with traditional messaging models, such as publish-subscribe and request-response, aiming to mitigate vulnerabilities associated with centralized architectures, including single points of failure [4, 10, 12]. These integrations have been critically evaluated concerning their effectiveness in enhancing resource efficiency, network overhead reduction, latency optimization, and security enhancements [11, 13]. Future research directions suggested by recent publications include developing hybrid blockchain systems, improving interoperability among diverse blockchain platforms, and optimizing consensus algorithms to accommodate the specific demands of various applications effectively [12, 13].

FORMULATION OF THE PROBLEM

Therefore, exploring innovative integrations of blockchain with conventional communication models, such as the publish-subscribe pattern, can lead to substantial improvements in reliability, scalability, and overall efficiency in data-intensive and resource-constrained environments. This paper addresses such integration by presenting and analyzing Trinity, a distributed broker system enhanced by blockchain technology, aiming to leverage blockchain's advantages to reinforce and expand the capabilities of the publish-subscribe model.

The Constrained Application Protocol (CoAP) is a "request-response" protocol designed specifically for resource-constrained systems, characterized by low resource consumption. Despite its advantages, CoAP lacks adequate scalability and portability across diverse platforms. Consequently, the "publish-subscribe" messaging model emerges as a viable alternative due to its minimal communication overhead and resource efficiency [4].

The "publish-subscribe" messaging model typically involves three components: publishers, subscribers, and a central broker that coordinates interactions between these entities. While this model efficiently utilizes resources and ensures scalability, it also introduces a critical vulnerability due to its dependency on a centralized broker. Such centralized architecture renders the "publish-subscribe" model susceptible to single points of failure [4]. Additionally, this configuration allows both publishers and subscribers to interact exclusively through a centralized server, controlled by a single organization, posing potential risks to data integrity and security [9, 10].

This study focuses on exploring and presenting the functionality of Trinity, a distributed publish-subscribe broker system enhanced by blockchain technology for data immutability. Trinity integrates a broker system with blockchain structures, specifically utilizing MQTT and the Tendermint blockchain platform. The primary objective of this research is to analyze how the trust advantages provided by the Trinity platform influence resource efficiency, especially regarding the reduction of associated overhead costs.

PRESENTING MAIN MATERIAL

The Trinity platform integrates blockchain technology with the publish-subscribe messaging model, widely adopted in modern application deployments [3, 4]. The core components of blockchain technology include the consensus algorithm, distributed ledger, and public-key cryptography [5, 8]. These components interact closely and coordinate actions within a distributed network of devices managed by multiple organizations [1, 2, 6].

The publish-subscribe communication model typically consists of three primary components: broker, publisher, and subscriber [4]. Publishers transmit data to brokers following a topic-based mechanism, while subscribers, who have subscribed to specific topics, receive messages forwarded by the broker. Topics, potentially hierarchical, usually represent metadata describing the information in a structured string format [4].

Trinity enhances the conventional publish-subscribe broker model by incorporating blockchain technology components, specifically distributed ledgers and consensus algorithms [3, 7]. As a result, the Trinity system comprises three main distributed components: a blockchain network, distributed brokers, and publisher-subscriber entities. The distinguishing feature of Trinity is its support for smart contracts and the maintenance of an immutable ledger within the blockchain structure [2, 13]. In contrast to traditional brokers, which simply relay published data to subscribers without verification, Trinity validates and securely records transactions into the blockchain ledger before disseminating the information to subscribers [2, 9].

Blockchain networks within Trinity play a critical role in replicating system state and verifying the authenticity of data transactions [3, 11]. The Trinity platform offers comprehensive API interfaces for effective interaction with blockchain networks, allowing broker instances to seamlessly connect and perform necessary operations. For example, the DeliverTransaction API is responsible for managing state replication, executing consensus protocols, and block creation [3, 5]. Broker instances in the Trinity system utilize this API to submit data requiring verification, accompanied by essential metadata such as node identifiers and system-level information, to the blockchain infrastructure. Consequently, data submitted by broker instances undergo replication across the Trinity network, with each node performing smart contract executions and consensus algorithm processes before data is permanently recorded into blocks [3, 7].

Blockchain platforms generally comprise several foundational components, such as consensus protocols, block creation logic, distributed ledgers, and public-key cryptography mechanisms [1, 5, 8]. Trinity's platform is designed to be blockchain-agnostic, interacting seamlessly with underlying blockchain structures through dedicated API interfaces [3, 5]. Despite this abstraction, each Trinity instance maintains specific APIs for querying blockchain network information. For instance, the GetCurrentBlockHeight() API allows a Trinity instance to retrieve the current blockchain height. Similarly, the GetBlock(BlockHeight) API fetches the complete block information at a specific block height [3]. While broker instances in Trinity remain isolated from the core blockchain functionalities, the underlying blockchain structure must maintain specific properties, notably immutability and consistency [2, 5].

The consensus algorithm within the Trinity platform ensures data verification and system state replication across a distributed network comprising multiple organizations. All authorized network participants are required to validate messages received by the Trinity brokers. This validation relies on consensus protocols. The state perceived by one broker instance must be consistently replicated across all Trinity instances, requiring a transaction to be approved by a majority of devices within the network [7]. Trinity supports various consensus protocols, including Proof-of-Work (PoW), Proof-of-Stake (PoS), Byzantine Fault Tolerance (BFT) algorithms such as Tendermint, and leader-based protocols such as Raft [3, 6, 7]. The choice of consensus protocol directly influences resource consumption and the system's fault tolerance capability.

Additionally, blockchain platforms implement a distributed ledger to record transactions immutably across all devices participating in the consensus process. To ensure efficiency and manageability, blockchain platforms commonly use Merkle Trees or hash trees [8]. Trinity employs similar mechanisms, such as the Merkle Tree structure, to securely store transaction data.

Public-key cryptography is another critical component utilized by blockchain platforms to secure transaction integrity and participant authenticity during verification processes. Each network participant generates a pair of cryptographic keys, transmitting their public key to the network to participate actively in block creation and consensus procedures [8, 9].

The blockchain network within the Trinity system incorporates specialized devices known as validators, authorized to execute smart contracts, participate actively in consensus processes, and facilitate block creation [3, 5]. In permissioned blockchain environments, only a specific subset of devices may serve as validators, whereas public blockchains like Bitcoin and Ethereum allow any capable device to validate transactions [1, 2]. Given this context, Trinity is particularly suited to permissioned blockchain implementations, where transaction participants are known entities, and subscribers must explicitly register their interests before receiving data [13].

The Trinity platform distinguishes tasks specifically related to blockchain functionalities from broker-specific tasks, interacting between the two through API interfaces [3]. This design allows flexibility and modularity in application development, making it possible to replace the underlying publish-subscribe model with other communication protocols for developing innovative blockchain-based infrastructures [10, 12].

The implementation of Trinity described in this study leverages the MQTT Mosquitto Broker and the Tendermint blockchain platform [3, 4]. Tendermint provides comprehensive tools for achieving consensus within distributed networks, executing smart contracts, and creating blocks [3]. Additionally, Tendermint isolates blockchain-related functionalities from application-specific features, enabling the development of diverse applications ranging from cryptocurrency management to distributed chat servers [3, 5]. The Application Blockchain Interface (ABCI) was employed to integrate the MQTT application with the blockchain framework. Tendermint utilizes a Byzantine Fault Tolerance (BFT) consensus protocol requiring approval from two-thirds of network devices to validate transactions and subsequently commit them to a block [3, 6].

To evaluate Trinity's performance, the experimental setup employed a network of 20 Raspberry Pi 3 devices, each equipped with a quad-core ARM Cortex-A53 processor and 1GB of RAM, running the Hypriot OS. All devices were interconnected through a local network. Performance metrics were obtained by publishing 1,000 messages to an MQTT broker within the network, comparing traditional message delivery against blockchain-validated message delivery across configurations of 5, 10, 15, and 20 nodes [4]. This experimental setup aimed to assess network performance and end-to-end latency scalability, utilizing Tendermint's default configuration parameters [3].

End-to-End Latency Analysis. Unlike traditional brokers, Trinity brokers validate published messages using smart contracts within a blockchain framework and record transactions into a distributed, immutable ledger [3, 5]. Consequently, subscribers receive transactions that are already verified and securely recorded. The validation process introduces additional latency due to the execution of smart contracts, consensus algorithms, and block creation protocols [3, 5, 7]. Subscribers experience this latency primarily due to the overhead associated with blockchain operations, including the consensus mechanism and smart contract execution [3, 13].

The experimental analysis measured end-to-end latency under varying publishing intervals: messages were sent every 0.2 seconds, 0.5 seconds, and 1 second, corresponding to transaction rates of 5 transactions per second (TPS), 2 TPS, and 1 TPS, respectively [4]. The results indicated that end-to-end latency increases proportionally with the number of devices in the network due to the heightened number of validators participating in the consensus process [6, 7].

Trinity disseminates verified data to all brokers within the network. Therefore, all brokers possess validated data immediately after it is recorded in the immutable ledger [3]. Experimental results demonstrated a maximum latency of approximately 3.5 seconds for a 20-node configuration operating at 5 TPS, whereas latency for feedback transactions remained minimal, around 90 milliseconds. When subscribing to verified data from another broker, latency slightly increased from 3.5 seconds to approximately 3.7 seconds. The additional latency introduced by blockchain-based immutability and verification is justified by significant improvements in security and trustworthiness [11, 13].

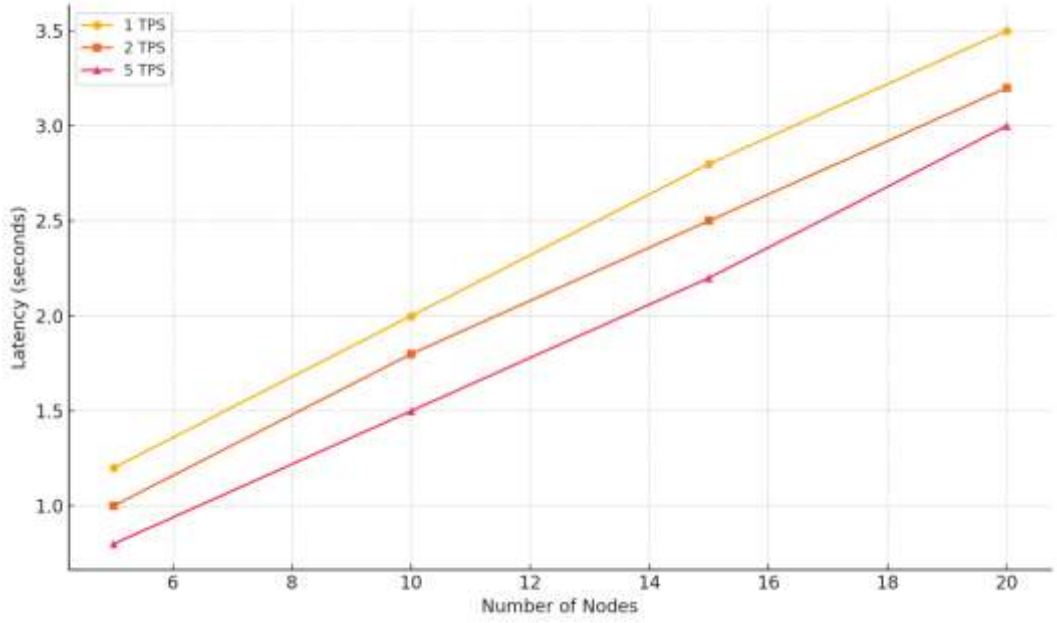

Fig 1. End-to-end latency of Trinity platform

Network Overhead Analysis. The Trinity platform utilizes blockchain infrastructure for state replication, consensus protocol execution, and block creation, resulting in network traffic generation through coordination among all network devices [3, 11]. Network overhead was measured within the experimental setup and represented the overhead incurred by the Tendermint platform.

Experimental data indicated that network overhead increases proportionally with the number of devices in the Trinity network. Notably, lower transaction rates (TPS) resulted in higher network overhead due to the creation of a greater number of blocks, each containing fewer transactions [3, 7]. Conversely, higher TPS values generally aggregated multiple transactions within a single block, reducing per-transaction overhead [3, 11].

Overall, maximum network overhead was approximately 300 MB per 1,000 seconds for a 20-device setup at 1 TPS, considered acceptable compared to the benefits provided by the Trinity infrastructure [11, 13]. Aggregating multiple transactions in a single block was identified as a potential method for reducing network overhead; however, this approach could result in increased end-to-end latency [3, 12, 13].

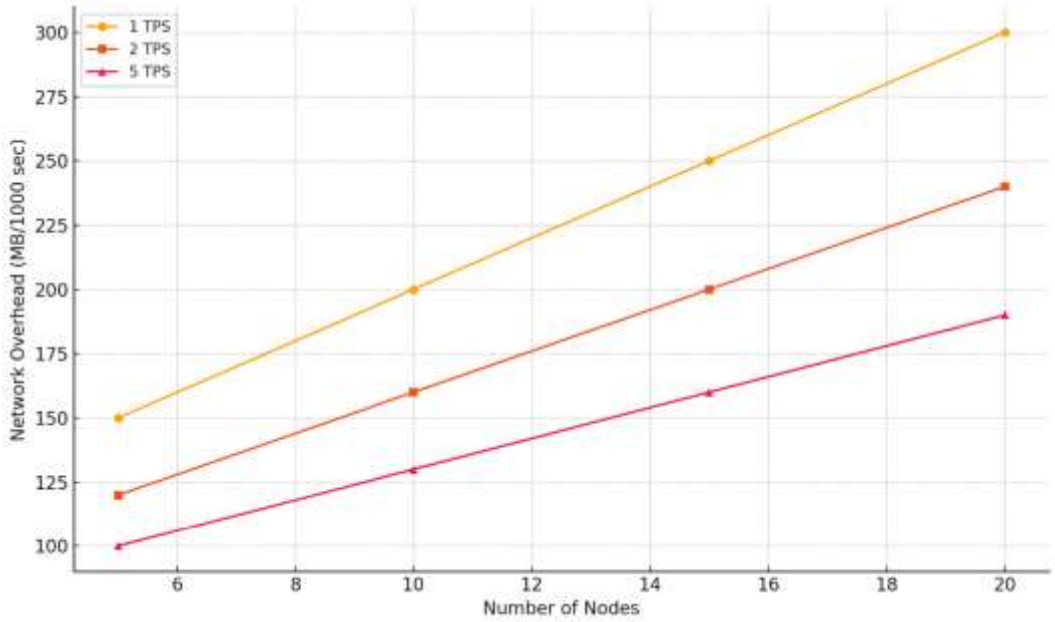

Fig 2. Network overhead of Trinity platform

CPU and RAM Utilization Analysis. Resource efficiency, specifically CPU and memory consumption, is an essential consideration in evaluating the feasibility and performance of the Trinity platform. In the conducted experiments, resource usage was measured using a network composed of 20 Raspberry Pi 3 nodes. Each node was equipped with a quad-core ARM Cortex-A53 processor and 1 GB of RAM [4].

The experiments indicated that the Trinity framework utilized approximately 85% of CPU resources on the Raspberry Pi 3 platform during the publication of 1,000 transactions at a rate of 1 TPS in a 20-node network. This high CPU usage is attributed to the continuous execution of smart contracts, consensus algorithms, and block creation protocols required by the blockchain infrastructure [3, 7, 11].

Similarly, the peak memory utilization was measured during the publication of 1,000 transactions at 1 TPS over a 15-minute interval within the same 20-node network setup. The resource consumption by the Trinity platform increased proportionally with the number of blocks created. Consequently, resource utilization could be optimized by aggregating multiple transactions into fewer blocks, effectively reducing CPU and RAM load [11, 12].

This experimental analysis underscores the importance of optimizing block transaction aggregation strategies to minimize resource consumption. Balancing the number of transactions per block can substantially reduce computational overhead, enhancing the overall resource efficiency of the Trinity platform [11, 13]. Despite the increased overhead compared to traditional publish-subscribe systems, the reliability, security, and trust benefits offered by Trinity justify these additional resource expenditures [10, 13].

Implementation of the Trinity Platform. The implementation of the Trinity platform was conducted using the MQTT Mosquitto Broker combined with the Tendermint blockchain framework [3, 4]. Tendermint provides essential tools for consensus achievement within distributed networks, smart contract execution, and block creation processes. A notable advantage of the Tendermint platform is its capacity to isolate blockchain-specific functionalities from application-specific operations, enabling a broad range of applications development, from cryptocurrencies to distributed chat servers [3, 5]. To integrate MQTT functionality effectively with blockchain features, the Application Blockchain Interface (ABCI) was employed. The Tendermint platform utilizes the Byzantine Fault Tolerance (BFT) consensus protocol, requiring approval from two-thirds of devices within the network for transaction validation and block inclusion [3, 6].

For the experimental evaluation, Trinity was deployed in a network consisting of 20 Raspberry Pi 3 devices. Each device featured a quad-core ARM Cortex-A53 processor and 1GB RAM and operated on the Hypriot OS [4].

All devices were interconnected within a local network infrastructure. The evaluation involved publishing a total of 1,000 messages to one of the MQTT brokers in the network [4]. Upon message receipt, the broker could either forward the message directly to subscribers, as typical in conventional publish-subscribe systems, or route it through the blockchain structure for verification [3, 4].

The evaluation was systematically carried out using varying configurations of 5, 10, 15, and 20 nodes to assess Trinity's network performance, end-to-end latency, and resource utilization scalability [3, 11]. During the evaluation, the default configuration provided by the Tendermint framework was maintained to ensure consistency and comparability of experimental outcomes [3].

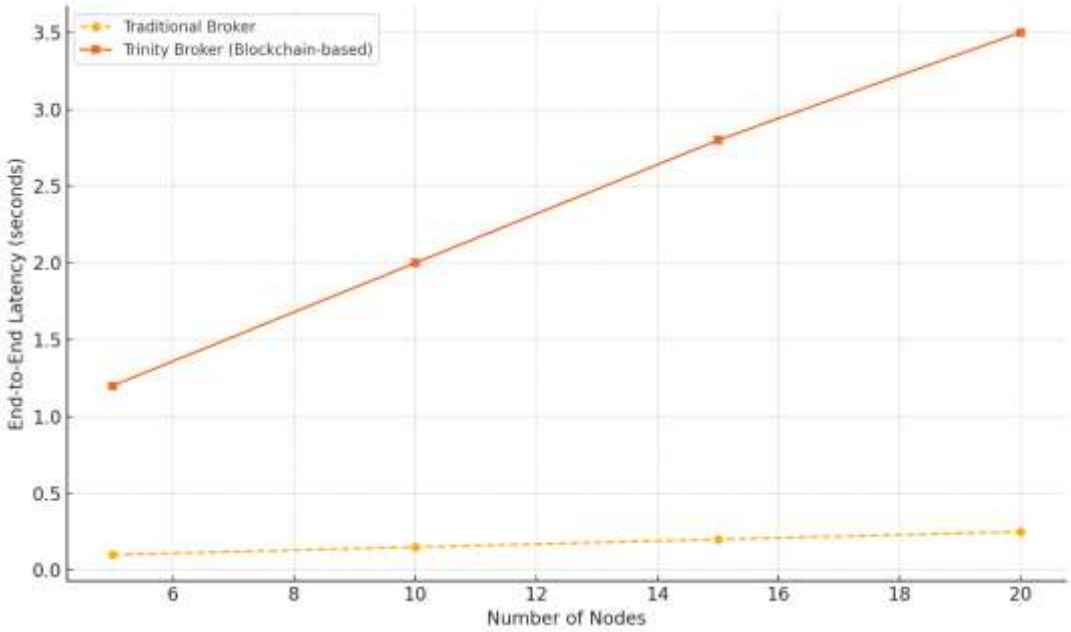

Fig 3. Comparative analysis of end-to-end latency: Traditional vs. Trinity blockchain-based brokers

The experimental results clearly indicated that the Trinity platform introduces specific overheads in terms of end-to-end latency, network traffic, and computational resource usage compared to traditional publish-subscribe systems [11, 13]. Nonetheless, these overheads are significantly outweighed by the enhanced security, reliability, and trustworthiness facilitated by the blockchain and smart contracts, making Trinity highly suitable for critical distributed applications requiring secure and reliable data exchange [12, 13].

CONCLUSIONS FROM THIS STUDY AND PROSPECTS FOR FURTHER RESEARCH IN THIS DIRECTION

Blockchain technology has significantly impacted various domains beyond cryptocurrencies, leveraging its fundamental components such as consensus protocols and distributed ledgers. These characteristics make blockchain highly promising for applications requiring reliability, transparency, and security beyond financial transactions.

This study specifically examined the Trinity platform, a distributed publish-subscribe broker system integrated with blockchain technology to ensure data immutability and enhance trust among participants. Trinity effectively addresses scenarios involving multiple independent stakeholders, such as food supply chain monitoring, smart cities, and industrial IoT environments. By clearly separating blockchain-specific tasks from broker-specific functionalities, Trinity provides application developers flexibility in integrating their brokers with different blockchain platforms.

The smart contract functionality in Trinity automates data verification processes, particularly critical for sensitive transactions in multi-organizational infrastructures. The practical implementation and experimental evaluation of Trinity, using MQTT and Tendermint frameworks on a network of 20 Raspberry Pi 3 devices, demonstrated the feasibility and effectiveness of this approach. Experimental results highlighted increased end-to-end latency (approximately 3 seconds), network bandwidth consumption, and computational resource usage. Despite these overheads, the trust, security, and reliability advantages provided by Trinity significantly outweigh the additional resource demands.

Consequently, the Trinity platform represents a highly viable solution for distributed environments requiring secure, reliable, and automated communication, such as smart cities, industrial IoT systems, and other critical infrastructures. Furthermore, Trinity's modular design, which isolates blockchain functionalities from application-specific broker tasks, provides flexibility for future developments and expansions, supporting various blockchain platforms and communication protocols.

References

1. Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. URL: https://bitcoin.org/bitcoin.pdf
2. Buterin, V. (2014). Ethereum Whitepaper. URL: https://ethereum.org/en/whitepaper/
3. Kwon, J. (2014). Tendermint: Consensus without Mining. URL: https://tendermint.com/static/docs/tendermint.pdf
4. Banks, A., & Gupta, R. (2014). MQTT Version 3.1.1 Protocol Specification. Oasis Standard. URL: https://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html
5. Wood, G. (2014). Ethereum: A Secure Decentralised Generalised Transaction Ledger. Ethereum Yellow Paper. URL: https://ethereum.github.io/yellowpaper/paper.pdf
6. Lamport, L., Shostak, R., & Pease, M. (1982). The Byzantine Generals Problem. ACM Transactions on Programming Languages and Systems, 4(3), 382–401.
7. Ongaro, D., & Ousterhout, J. (2014). In Search of an Understandable Consensus Algorithm. Proceedings of the 2014 USENIX Annual Technical Conference, 305–319.
8. Merkle, R.C. (1980). Protocols for Public Key Cryptosystems. IEEE Symposium on Security and Privacy, 122–134.
9. Antonopoulos, A.M. (2014). Mastering Bitcoin: Unlocking Digital Cryptocurrencies. O'Reilly Media, Inc.
10. Swan, M. (2015). Blockchain: Blueprint for a New Economy. O'Reilly Media, Inc.
11. Zheng, Z., Xie, S., Dai, H., Chen, X., & Wang, H. (2017). An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends. IEEE International Congress on Big Data (BigData Congress), 557–564.
12. Crosby, M., Nachiappan, Pattanayak, P., Verma, S., & Kalyanaraman, V. (2016). Blockchain Technology: Beyond Bitcoin. Applied Innovation Review, Issue 2, 6–19.
13. Mohanta, B.K., Panda, S.S., & Jena, D. (2018). An Overview of Smart Contract and Use Cases in Blockchain Technology. 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 1–4.

Література

1. Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. [Електронний ресурс]. URL: https://bitcoin.org/bitcoin.pdf
2. Buterin, V. (2014). Ethereum Whitepaper. [Електронний ресурс]. URL: https://ethereum.org/en/whitepaper/
3. Kwon, J. (2014). Tendermint: Consensus without Mining. [Електронний ресурс]. URL: https://tendermint.com/static/docs/tendermint.pdf
4. Banks, A., & Gupta, R. (2014). MQTT Version 3.1.1 Protocol Specification. Oasis Standard. [Електронний ресурс]. URL: https://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html
5. Wood, G. (2014). Ethereum: A Secure Decentralised Generalised Transaction Ledger. Ethereum Yellow Paper. [Електронний ресурс]. URL: https://ethereum.github.io/yellowpaper/paper.pdf
6. Lamport, L., Shostak, R., & Pease, M. (1982). The Byzantine Generals Problem. ACM Transactions on Programming Languages and Systems, 4(3), 382–401.
7. Ongaro, D., & Ousterhout, J. (2014). In Search of an Understandable Consensus Algorithm. In Proceedings of the 2014 USENIX Annual Technical Conference (pp. 305–319).
8. Merkle, R.C. (1980). Protocols for Public Key Cryptosystems. IEEE Symposium on Security and Privacy, 122–134.
9. Antonopoulos, A.M. (2014). Mastering Bitcoin: Unlocking Digital Cryptocurrencies. O'Reilly Media, Inc.
10. Swan, M. (2015). Blockchain: Blueprint for a New Economy. O'Reilly Media, Inc.
11. Zheng, Z., Xie, S., Dai, H., Chen, X., & Wang, H. (2017). An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends. IEEE International Congress on Big Data (BigData Congress), 557–564.
12. Crosby, M., Nachiappan, Pattanayak, P., Verma, S., & Kalyanaraman, V. (2016). Blockchain Technology: Beyond Bitcoin. Applied Innovation Review, Issue 2, 6–19.
13. Mohanta, B.K., Panda, S.S., & Jena, D. (2018). An Overview of Smart Contract and Use Cases in Blockchain Technology. 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 1–4.