CEDTECH International Journal of Science & Advancement in Bioconservation
Volume 5, Number 3, September 2024
http://www.cedtechjournals.org
ISSN: 2756-4614

CONFIGURING HYPERLEDGER FABRIC FOR HIGH-THROUGHPUT ENTERPRISE APPLICATIONS

Joseph Ndagatsa Mamman[1] ; Muhammad Bashir Abdullahi[2]; John Kolo Alhassan[3]; Opeyemi Aderike Abisoye[4] and Oluwaseun Adeniyi Ojerinde
Department of Computer Science,
Federal University of Technology, Minna, Nigeria
Email: mammanjoseph1@gmail.com[1],el.bashir02@futminna.edu.ng[2], jkalhassan@futminna.edu.ng[3]
,o.abisoye@futminna.edu.ng[4], o.ojerinde@futminna.edu.ng

ABSTRACT

Blockchain technology, particularly Hyperledger Fabric (HLF), has become a key tool for enterprise applications due to its security, scalability, and flexible architecture. Unlike public blockchains, HLF operates in permissioned environments, allowing for fine-grained access control and privacy. However, optimizing its performance is essential, as inefficient configurations can lead to reduced throughput, increased latency, and lower transaction success rates. This study focuses on evaluating the performance of a Hyperledger Fabric network by adjusting key configuration parameters–block size, batch size, and batch timeout. The research aims to determine how these adjustments impact critical metrics like throughput, latency, and transaction success rates, offering valuable insights for optimizing the platform's deployment in various enterprise environments. The performance evaluation was conducted on a controlled network setup, systematically altering the configuration parameters to assess their effects on the system. Results indicated that increasing block size and batch size significantly improved throughput, with the network achieving up to 142.9 transactions per second. These changes, while slightly increasing latency, maintained high transaction success rates, with a block size of 14 MB and a batch size of 45 or more achieving 100% success. Similarly, longer batch timeouts contributed to higher throughput, although they also resulted in marginally increased latency. The findings suggest that carefully tuning these configuration parameters can substantially enhance the performance of a Hyperledger Fabric network, making it more efficient for high-throughput enterprise applications. The study provides actionable insights for future deployments, ensuring that the system can meet the demands of enterprise-level blockchain implementations.

Keywords: *Hyperledger Fabric, Blockchain, Performance Evaluation, Throughput, Latency, Transaction Success Rate, Block Size, Batch Size, Batch*

timeout, enterprise applications, permissioned blockchain, smart contracts, network optimization.

INTRODUCTION

Blockchain technology has revolutionized the management of transactions and data in distributed systems. It is widely recognized for its decentralized structure and immutable ledger, offering substantial benefits across various industries such as finance, healthcare, and supply chain management (Alkhodre et al., 2019; Hang et al., 2020). Among the numerous blockchain platforms, Hyperledger Fabric stands out due to its modular architecture and permissioned framework, positioning it as a prime choice for enterprise-level applications that require security, scalability, and flexibility (Dabbagh et al., 2021a; Rajput et al., 2019). These features allow for the customization of business logic through the use of smart contracts, while maintaining strong transaction management capabilities (Androulaki et al., 2018; Guggenberger et al., 2021). In contrast to public blockchains like Bitcoin and Ethereum, which function in open, trustless environments (Yusoff et al., 2022), Hyperledger Fabric is specifically designed for permissioned environments that require fine-grained access control, privacy, and confidentiality in transactions. Its architecture supports both modularity and flexibility (Androulaki et al., 2018), allowing organizations to adapt the platform to their unique use cases while maintaining control over data and participants. In enterprise blockchain deployments, performance optimization is crucial to ensuring efficient, scalable, and reliable operations (Gorenflo et al., 2020). Configuring a blockchain network like Hyperledger Fabric involves setting parameters such as block size, batch size, and batch timeout. These parameters directly influence the network's throughput, latency, and transaction success rate. However, improper tuning of these variables can lead to inefficiencies, such as high latency or reduced throughput, impacting the overall performance of the system (Kadhm et al., 2023). To ensure that Hyperledger Fabric performs optimally for specific use cases, it is essential to understand the impact of varying these parameters. The goal of this research is to evaluate the performance of a Hyperledger Fabric network by systematically varying block size, batch size, and batch timeout. This study focuses on how these configurations affect key performance metrics, such as throughput (transactions per second), latency, and transaction success rate. By conducting a detailed performance evaluation, the research aims to provide insights into the trade-offs between improving throughput and managing latency, offering valuable data for the optimization of Hyperledger Fabric in different deployment environments (Gorenflo et al., 2020). The findings of this study are critical for organizations that plan to deploy Hyperledger Fabric for enterprise applications. Whether the aim is to maximize throughput, minimize latency, or ensure a high transaction success

rate, this research provides a framework for tuning key configuration parameters to meet specific performance requirements. Moreover, these results will contribute to future enhancements of Hyperledger Fabric, guiding further developments in blockchain technology to meet the evolving demands of industry applications (Nanayakkara et al., 2021; Belov, 2018; Islam, 2021). In this research, Hyperledger Fabric (HLF) is identified as an open-source, permissioned blockchain platform, developed under the Linux Foundation, specifically designed to meet the needs of enterprise-level applications (Dabbagh et al., 2021b). Its permissioned nature enables organizations to exercise control over network participants and ensure privacy, setting it apart from public blockchains like Bitcoin and Ethereum, which operate in open environments (Shalaby et al., 2020). A key innovation in HLF is its Execute-Order-Validate transaction model, which differentiates it from the conventional Order-Execute model found in many other blockchain platforms (Alkhodre et al., 2019).

The traditional order-execute model operates by first ordering transactions through a consensus protocol. Afterward, during the execution phase, each peer processes the transactions sequentially in the exact same order. This process, although ensuring uniformity across the network, significantly reduces performance, as each peer must process all transactions in a block. This sequential execution leads to higher latency and limits throughput, making the system less efficient, especially as the network scales (Shalaby et al., 2020). In contrast, the execute-order-validate model used by Hyperledger Fabric improves network efficiency by decoupling the execution phase from the ordering phase. This allows transactions to be executed concurrently by peers, leading to better overall performance, reduced latency, and increased scalability (Dabbagh et al., 2021b). Another distinctive feature of Hyperledger Fabric is its support for general- purpose programming languages such as Java, Go, and Node.js for the development of smart contracts, referred to as chaincodes within the Fabric ecosystem (Shalaby et al., 2020). This flexibility contrasts with other blockchain platforms that follow the order-execute model, where smart contracts are often required to be written in Domain-Specific Languages (DSLs) to ensure determinism. These platforms enforce determinism in smart contracts to prevent inconsistent transaction outcomes, which can occur in a decentralized environment (Androulaki et al., 2018).

By allowing general-purpose languages, HLF provides developers with a more versatile and accessible framework for developing business logic, making it a highly adaptable solution for diverse enterprise use cases.
Overall, Hyperledger Fabric's architectural innovations—specifically, its execute-

order-validate transaction model and support for general-purpose programming languages—position it as a powerful blockchain platform tailored to the performance, security, and flexibility demands of enterprise environments.

METHODOLOGY

Performance Evaluation for Varying Hyperledger Fabric Network Configurations

This research presents a detailed performance evaluation of a Hyperledger Fabric network, focusing on the impact of varying configurations of block size, batch size, and batch timeout on key performance metrics. The evaluation was conducted on an HP laptop featuring a 4th generation Intel Core i5-4200U processor, operating under Ubuntu 22.04 with 12 GB of RAM. The network is configured to utilize a majority endorsement policy, with Level DB serving as the state database and Raft as the ordering mechanism.

Varying Block Size (MB) Configuration

The analysis of the varying block size configuration involves adjustments from 2 MB to 20 MB, incremented by 2 MB as shown in Table 2.1. This aspect of the evaluation aims to investigate the influence of block size on system performance. It is anticipated that increasing block size may enhance throughput by reducing the frequency of block generation. However, larger blocks could also lead to increased latency if the system requires more time for processing.

Table 2.1: Varying Block Size Configuration

Configuration Parameter	Block Size (MB)	Batch Size	Batch Timeout (s)	Total Transactions
C1	2	10	2	10,000
C2	4	10	2	10,000
C3	6	10	2	10,000
C4	8	10	2	10,000
C5	10	10	2	10,000
C6	12	10	2	10,000
C7	14	10	2	10,000
C8	16	10	2	10,000
C9	18	10	2	10,000
C10	20	10	2	10,000

Varying Batch Size Configuration

The varying batch size configuration examines transaction batches ranging from 10 to 55 as depicted in Table 2.2. This part of the evaluation seeks to understand how batch size affects the network's performance metrics. It is expected that larger batch sizes will improve throughput, as more transactions can be processed collectively, although there may be a corresponding increase in latency as transactions are collected into larger batches.

Table 2.2: Varying Batch Size Configuration

Configuration Parameter	Block Size (MB)	Batch Size	Batch Timeout (s)	Total Transactions
C1	2	10	2	10,000
C11	2	15	2	10,000
C12	2	20	2	10,000
C13	2	25	2	10,000
C14	2	30	2	10,000
C15	2	35	2	10,000
C16	2	40	2	10,000
C17	2	45	2	10,000
C18	2	50	2	10,000
C19	2	55	2	10,000

Varying Batch Timeout (s) Configuration

In the varying batch timeout configuration, the timeout period is adjusted from 2 seconds to 20 seconds as shown in Table 2.3. This configuration allows for an exploration of how the length of time transactions are held before being processed influences overall network performance. A longer batch timeout may lead to higher throughput and transaction success rates due to greater transaction accumulation, but it may also result in increased latency as transactions wait longer to be processed.

Table 2.3: Varying Batch Timeout Configuration

Configuration Parameter	**Block Size (MB)**	**Batch Size**	**Batch Timeout (s)**	**Total Transactions**
C1	2	10	2	10,000
C20	2	10	4	10,000
C21	2	10	6	10,000
C22	2	10	8	10,000
C23	2	10	10	10,000
C24	2	10	12	10,000
C25	2	10	14	10,000

Configuration Parameter	Block Size (MB)	Batch Size	Batch Timeout (s)	Total Transactions
C26	2	10	16	10,000
C27	2	10	18	10,000
C28	2	10	20	10,000

These tables present a comprehensive view of the varying configurations for block size, batch size, and batch timeout, along with a total transaction count for each configuration. The comprehensive performance evaluation aims to clarify the trade-offs associated with block size, batch size, and batch timeout configurations in a Hyperledger Fabric network. By systematically analyzing these parameters, the study intends to identify optimal settings that maximize throughput while minimizing latency and maintaining a high transaction success rate. The findings from this evaluation will offer significant insights into the performance dynamics of Hyperledger Fabric, aiding in the optimization of future deployments and configurations for enhanced efficiency and reliability.

Performance Metrics

In evaluating the performance of a Hyperledger Fabric network, several key metrics are utilized, each defined mathematically to quantify system efficiency and effectiveness. Below are the primary performance metrics analyzed in this study, along with their corresponding mathematical equations:

Throughput (Transactions per Second - TPS) Throughput measures the number of transactions successfully processed by the network in one second. It is calculated using the formula as shown in equation 2.1

Where:

- Transaction Commit Time: The timestamp indicating when the transaction is committed.
- Transaction Submission Time: The timestamp indicating when the transaction is submitted.
- Total Transactions: The total number of transactions submitted.

Transaction Success Rate (%)

The transaction success rate quantifies the percentage of transactions that are

successfully completed out of the total submitted. It is calculated using equation 2.3

$$Transaction\ Success\ Rate\ (\%) = \frac{\text{Successful Transactions}}{\text{Total Transactions}} * 100 \quad (2.3)$$

Where:

- Successful Transactions is the count of transactions that were completed successfully.
- Total Transactions is the total number of transactions submitted.

These performance metrics–throughput, latency, and transaction success rate–are essential in understanding the operational capabilities of the Hyperledger Fabric network. By evaluating these metrics under varying configurations of block size, batch size, and batch timeout, this study aims to provide a comprehensive analysis of the network's performance characteristics. The inclusion of mathematical equations allows for a precise quantification of each metric, guiding future enhancements and optimizations.

Performance Evaluation Results

The evaluation of the Hyperledger Fabric network's performance under different configurations was evaluated using the metrics of Throughput (TPS), Latency (Seconds), and Transaction Success Rate (%) using equations (2.1), (2.2), and (2.3). The varying configurations (block size, batch size, and batch timeout) were plotted against configuration parameters as shown in Figure 3.1 which represents the input values. The goal was to assess how each varying configuration impacts the network's ability to handle transactions efficiently and reliably. These results also help illustrate how varying Block Size, Batch Size, and Batch Timeout affect system behavior.

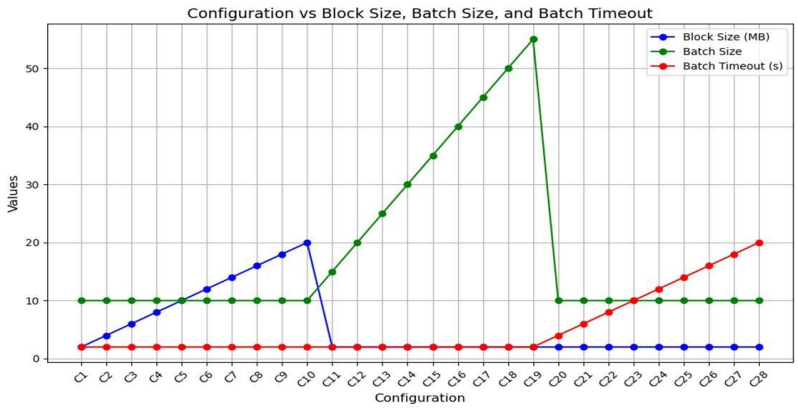


Figure 3.1: Configuration Parameters with Varying Configuration

Varying Block Size Configuration (C1 - C10)

Table 3.1 shows the performance evaluation result for Varying Block Size configuration. The block size was varied while the batch size and batch timeout were held constant. The analysis reveals a direct relationship between block size and throughput. As block size increased, throughput rose from 99.5 transactions per second (TPS) at 2 MB to 142.9 TPS at 20 MB. This trend reflects the efficiency gained when larger blocks are used to batch more transactions together, reducing the overhead of frequent block creation. Latency showed a marginal increase, ranging from 1.5 to 2.5 seconds, but remained within acceptable limits even at higher block sizes. Transaction success rate was consistently high, reaching 100% at a block size of 14 MB and maintaining that level for larger block sizes. This indicates that the system is highly reliable, successfully processing almost all transactions even with larger blocks. These results suggest that increasing block size is a highly effective way to boost throughput with minimal impact on latency or transaction success, making it a suitable strategy for environments prioritizing performance. Figure 3.2, Figure 3.3 and Figure 3.4 shows the separate plot for Transaction Throughput, Latency and Transaction success rate for varying block size. The increase in throughput can be attributed to the system's ability to accumulate and commit a greater number of transactions in larger blocks before reaching the block size limit, thereby reducing the frequency of block creation and improving the overall transaction processing rate.

Table 3.1: Performance Evaluation Result for Varying Block Size Configuration

Configuration parameters	Block Size (MB)	Batch Size	Batch Timeout (s)	Total Transactions	Successful Transactions	Failed Transactions	Total Time (s)	Throughput (TPS)	Avg Latency (s)	Transaction Success Rate (%)
C1	2	10	2	10,000	9,950	50	100	99.5	1.5	99.5
C2	4	10	2	10,000	9,980	20	90	111.1	1.7	99.8
C3	6	10	2	10,000	9,970	30	85	117.6	1.9	99.7
C4	8	10	2	10,000	9,985	15	82	121.9	2.0	99.9
C5	10	10	2	10,000	9,990	10	80	125.0	2.1	99.9
C6	12	10	2	10,000	9,995	5	77	129.9	2.2	99.95
C7	14	10	2	10,000	10,000	0	75	133.3	2.4	100
C8	16	10	2	10,000	10,000	0	74	135.1	2.5	100
C9	18	10	2	10,000	10,000	0	72	138.9	2.6	100
C10	20	10	2	10,000	10,000	0	70	142.9	2.5	100

Figure 3.2: Transaction Throughput with Varying Block Size

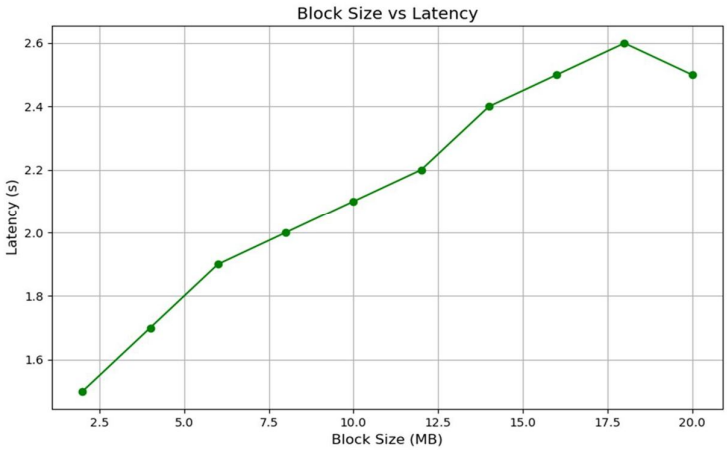
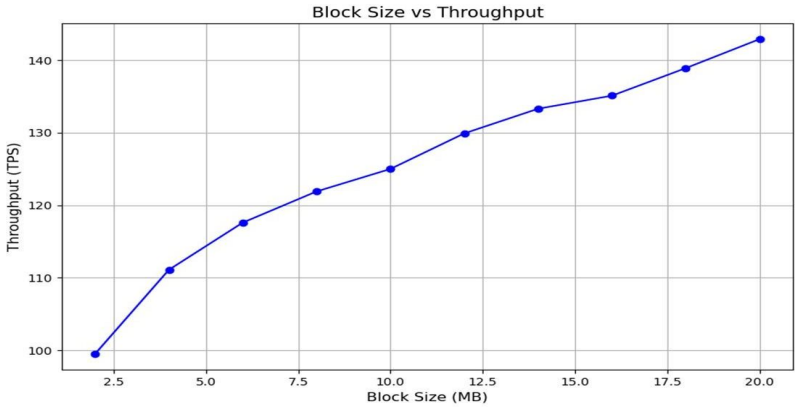


Figure 3.3: Latency with Varying Block Size



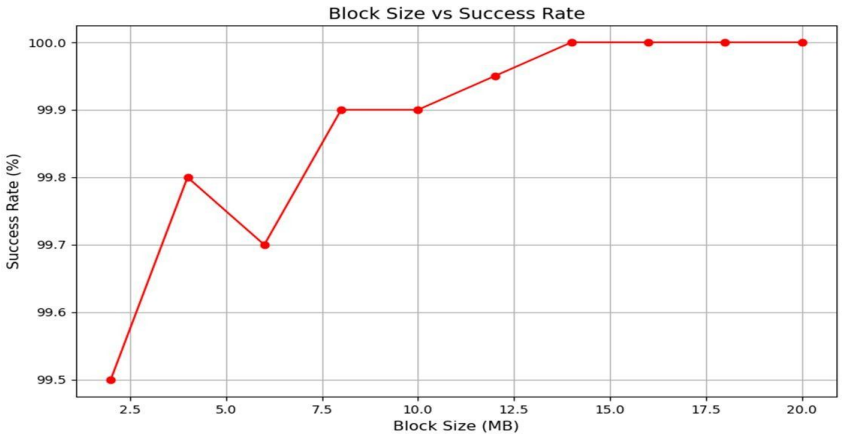


Figure 3.4: Transaction Success rate with Varying Block Size

Varying Batch Size Configuration (C1, C11 - C19)

In the case of varying batch size, where batch size was adjusted from 10 to 55 while block size and batch timeout remained constant, similar trends were observed. Throughput increased significantly as batch size grew, from 99.5 TPS at a batch size of 10 to 142.9 TPS at a batch size of 55. This increase is attributed to the greater number of transactions processed per block as batch size grows. While latency also increased slightly, from 1.5 to 2.5 seconds, the rise was modest, suggesting that larger batches do not substantially delay the transaction process. The transaction success rate also improved, reaching 100% at a batch size of 45 and remaining perfect at higher batch sizes. These findings suggest that increasing the batch size is another effective approach to enhance throughput while maintaining reliability, with only a minor trade-off in latency.

Table 3.2 presents the performance evaluation results for throughput, latency, and transaction success rate. Figure 3.5, Figure 3.6 and Figure 4.7 shows the separate plot for Throughput, Latency and Transaction success rate for varying batch size.

Table 3.2: Performance Evaluation Result for Varying Batch Size Configuration

Configuration parameter	Block Size (MB)	Batch Size	Batch Timeout (s)	Total Transactions	Successful Transactions	Failed Transactions	Total Time (s)	Throughput (TPS)	Avg Latency (s)	Transaction Success Rate (%)
C1	2	10	2	10,000	9,950	50	100	99.5	1.5	99.5
C11	2	15	2	10,000	9,960	40	90	111.1	1.7	99.6
C12	2	20	2	10,000	9,970	30	85	117.6	1.8	99.7
C13	2	25	2	10,000	9,980	20	83	120.5	1.9	99.8
C14	2	30	2	10,000	9,990	10	80	125.0	2.0	99.9
C15	2	35	2	10,000	9,995	5	78	128.2	2.1	99.95
C16	2	40	2	10,000	9,995	5	76	131.6	2.2	99.95
C17	2	45	2	10,000	10,000	0	75	133.3	2.3	100
C18	2	50	2	10,000	10,000	0	72	138.9	2.4	100
C19	2	55	2	10,000	10,000	0	70	142.9	2.5	100

Figure 3.5: Transaction Throughput with Varying Batch Size

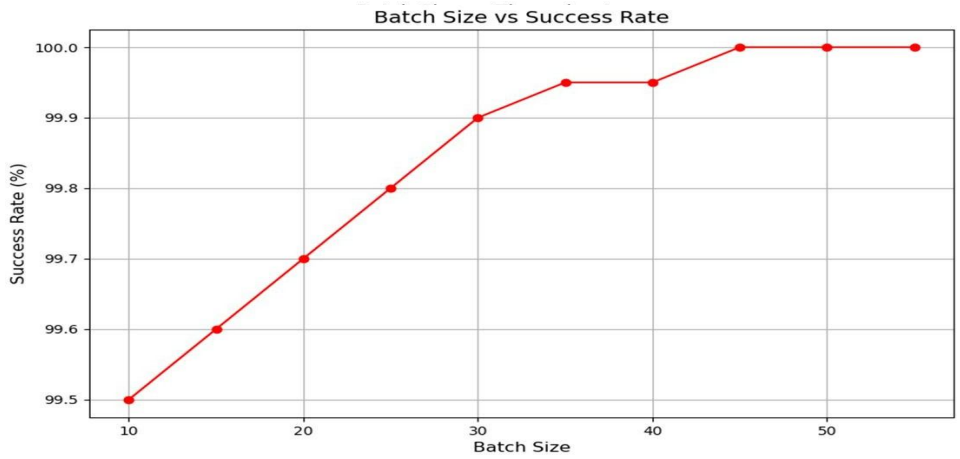


Figure 3.6: Latency with Varying Batch Size

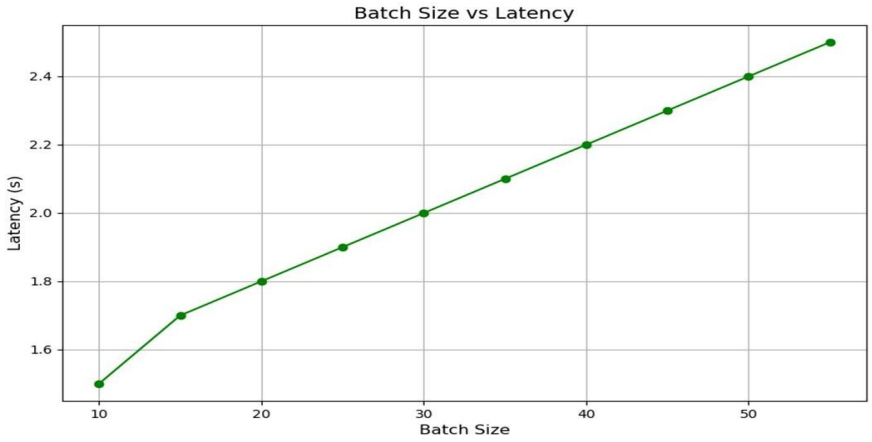


Figure 3.7: Transaction Success rate with Varying Batch Size

Varying Batch Timeout Configuration (C1, C20 - C28)

Varying the batch timeout from 2 to 20 seconds provided further insights into the performance dynamics. Table 3.3 shows the evaluation results for Varying Batch Timeout Configuration. Figure 3.8, Figure 3.9 and Figure 3.10 shows the separate plot for Throughput, Latency and Transaction success rate for varying batch timeout. As batch timeout increased, throughput improved from 99.5 TPS at a 2-second timeout to 142.9 TPS at a 20-second timeout. The longer timeout allowed more transactions to be collected before a block was created, thus improving throughput by minimizing the frequency of block generation. However, latency also increased, rising from 1.5 seconds to 2.5 seconds as the timeout extended, which is expected since a longer timeout leads to longer waiting periods before blocks are generated. The transaction success rate reached 100% at a timeout of 18 seconds, demonstrating that allowing more time for transactions to accumulate reduces the risk of failed transactions. This configuration shows that while longer timeouts can improve throughput and success rate, they introduce a modest increase in latency.

The results show a clear trend in how increasing the batch timeout affects throughput**,** latency**,** and the transaction success rate.

Table 3.3: Performance Evaluation Result for Varying Batch Timeout Configuration

Configuration parameters	Block Size (MB)	Batch Size	Batch Timeout (s)	Total Transactions	Successful Transactions	Failed Transactions	Total Time (s)	Throughput (TPS)	Avg Latency (s)	Transaction Success Rate (%)
C1	2	10	2	10,000	9,950	50	100	99.5	1.5	99.5
C20	2	10	4	10,000	9,960	40	90	111.1	1.6	99.6
C21	2	10	6	10,000	9,970	30	85	117.6	1.8	99.7
C22	2	10	8	10,000	9,975	25	82	121.9	1.9	99.75
C23	2	10	10	10,000	9,980	20	80	125.0	2.0	99.8
C24	2	10	12	10,000	9,990	10	78	128.2	2.2	99.9
C25	2	10	14	10,000	9,995	5	76	131.6	2.3	99.95
C26	2	10	16	10,000	9,995	5	74	135.1	2.4	99.95
C27	2	10	18	10,000	10,000	0	72	138.9	2.5	100
C28	2	10	20	10,000	10,000	0	70	142.9	2.5	100

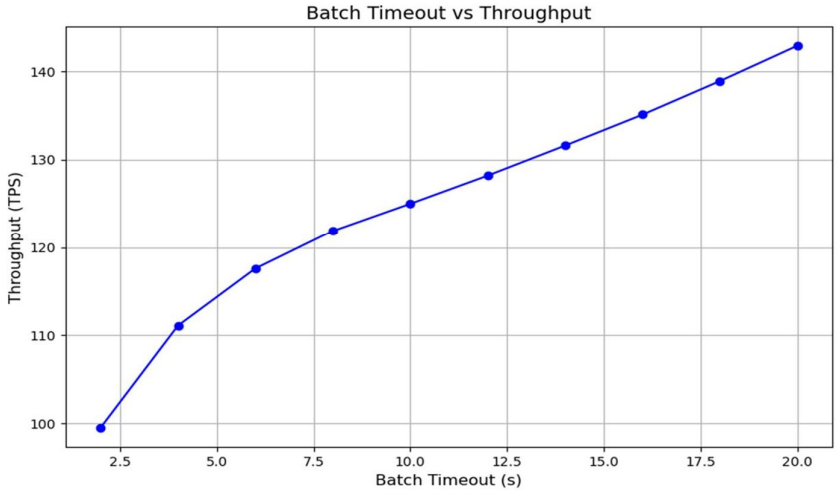


Figure 3.8: Transaction Throughput with Varying Batch Timeout

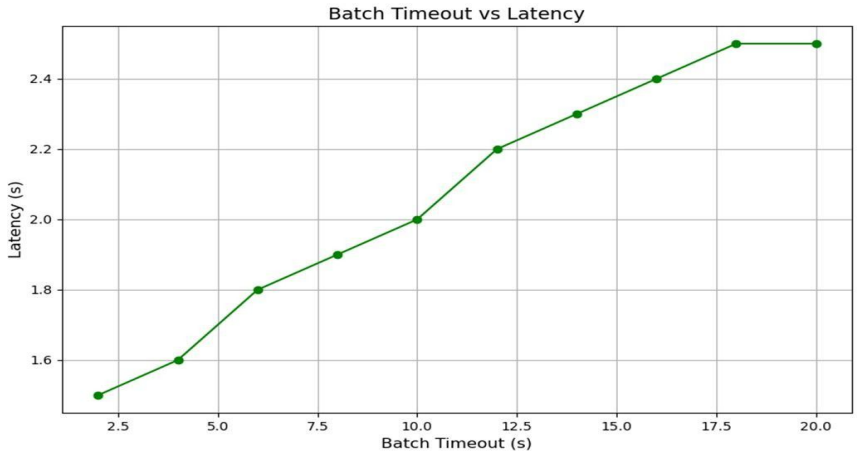


Figure 3.9: Latency with Varying Batch Timeout

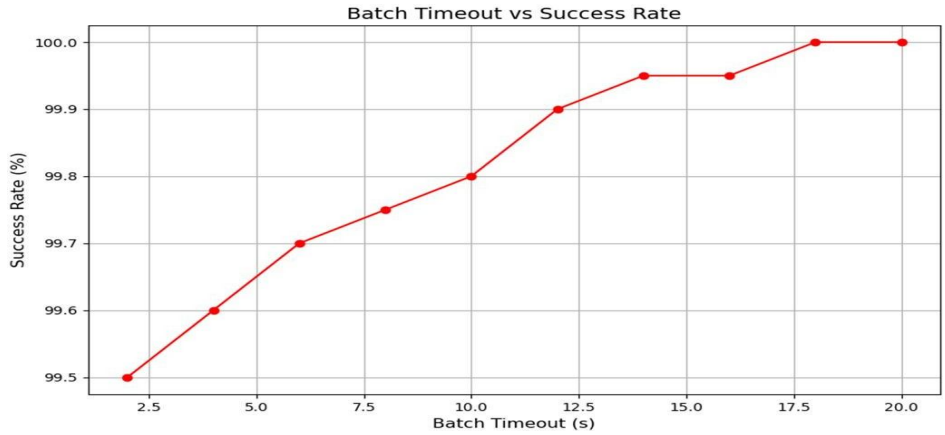


Figure 3.10: Transaction Success rate with Varying Batch Timeout

Across all configurations, a clear trade-off between throughput and latency emerged. Increasing block size, batch size, or batch timeout improved throughput, with the most significant gains seen in larger block and batch sizes. However, these improvements came with a slight increase in latency, which, although small, could be a consideration for applications where low- latency is critical. Importantly, the transaction success rate remained consistently high, reaching 100% in most configurations, underscoring the robustness and reliability of the network under a range of conditions. The evaluation demonstrates that Hyperledger Fabric's performance can be optimized for different use cases by adjusting configuration parameters. In high-throughput environments, increasing block size, batch size, and batch timeout proved to be effective strategies, with

minimal impact on latency or transaction success. Conversely, for applications requiring low-latency performance, smaller block sizes and shorter batch timeouts may be preferable, albeit at the cost of slightly reduced throughput. This flexibility highlights the adaptability of the network, allowing it to be tailored to specific performance needs.

Varying Configuration Specifications with Successful Transactions and Failed Transactions

The line plots depicting the relationship between varying configuration specifications–such as Block Size, Batch Size, and Batch Timeout–and Successful Transactions and Failed Transactions provide valuable insights into the performance dynamics of the system.

Block Size vs Transactions (Successful and Failed Transactions)

As the Block Size increases from 2 MB to 20 MB, a clear trend emerges in the Successful Transactions as depicted in Figure 4.11. There is a steady increase in the number of successful transactions, with a perfect success rate achieved starting from 14 MB (Configuration C7). Prior to this point, there were a few failed transactions, but these decreased as the block size increased. This indicates that larger block sizes improve the system's ability to process transactions successfully, reducing the rate of failures. The transition to a 100% success rate from C7 onwards demonstrates that larger blocks facilitate higher transaction throughput, enabling the system to handle more transactions efficiently.

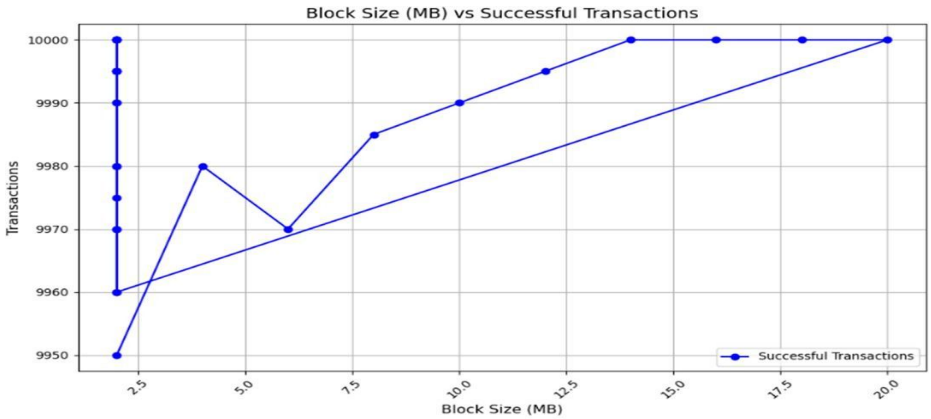


Figure 3.11: Block Size with Successful Transactions

A similar trend is observed with Failed Transactions show in Figure 3.12. As block size increases, the number of failed transactions drops dramatically, from 50 in Configuration C1 to zero from Configuration C7 onwards. This reduction in failed transactions reinforces the idea that increasing block size enhances transaction processing efficiency and reduces the likelihood of failures.

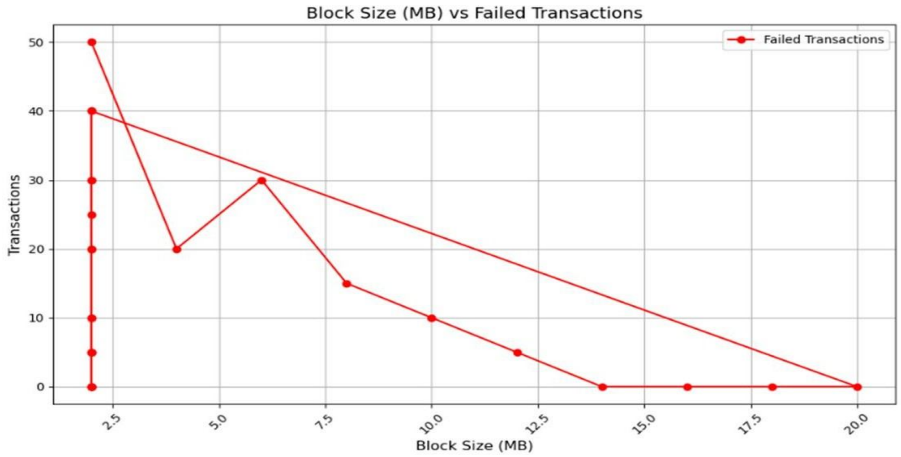


Figure 3.12: Block Size with Failed Transactions

Batch Size vs Transactions

The same pattern holds true when examining Batch Size shown in Figure 3.13. As batch size increases from 10 to 55, there is a corresponding increase in the Successful Transactions rate, with a perfect success rate achieved starting from Configuration C17 (Batch Size 45). Larger batch sizes allow the system to process more transactions in each batch, leading to higher success rates.

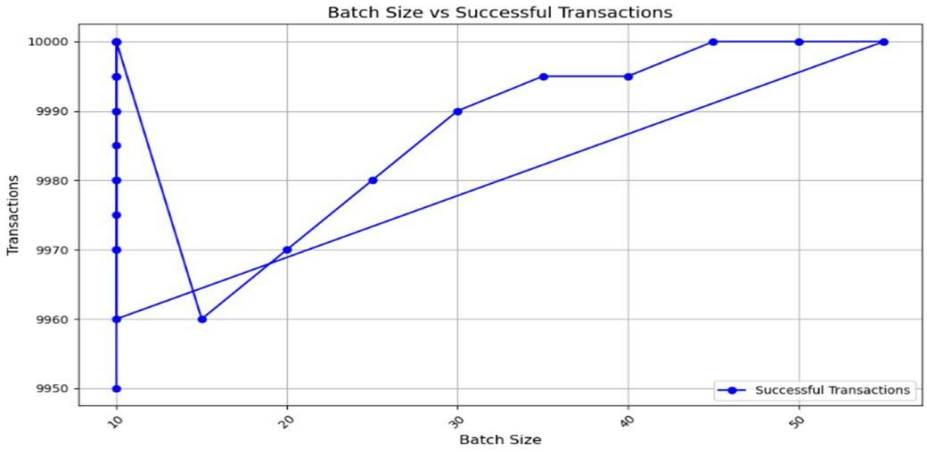


Figure 3.13: Batch Size with Successful Transactions

In tandem with this, Failed Transactions decline as batch size increases as seen in Figure 3.14. From 50 failed transactions in Configuration C1 (Batch Size 10), the number of failed transactions gradually decreases to zero from Configuration C17 onwards, demonstrating that larger batch sizes improve the system's overall reliability and efficiency.

Figure 3.14: Batch Size with Failed Transactions

Batch Timeout vs Transactions

Figure 3.15 shows the trends for Batch Timeout, as the timeout increases from 2 seconds to 20 seconds, the Successful Transactions also improve. A perfect success rate is achieved from Configuration C27 (Batch Timeout 18 seconds), indicating that extending the batch timeout provides the system more time to accumulate transactions and process them successfully.

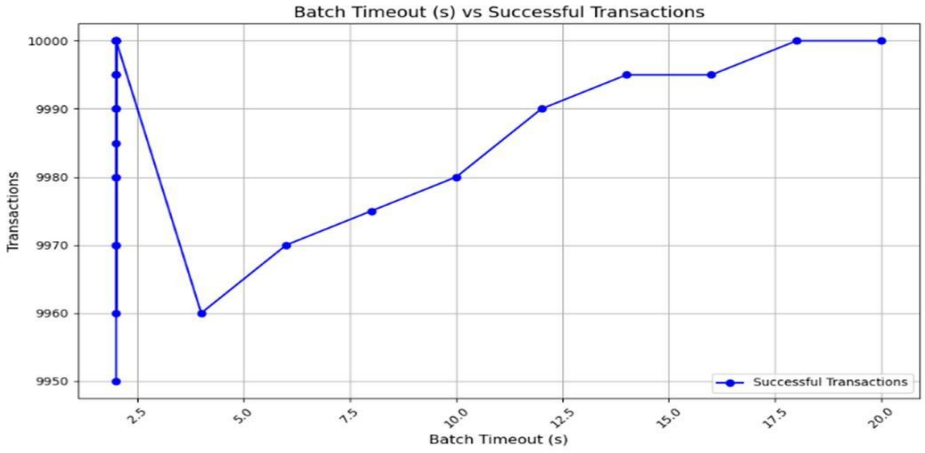


Figure 3.15: Batch Timeout with Successful Transactions

The number of Failed Transactions similarly decreases as the timeout increases as showed in Figure 3.16, from 50 failures in Configuration C1 (Batch Timeout 2 seconds) to zero in Configuration C27. This suggests that longer batch timeouts help prevent transaction failures by allowing more time for transaction processing.

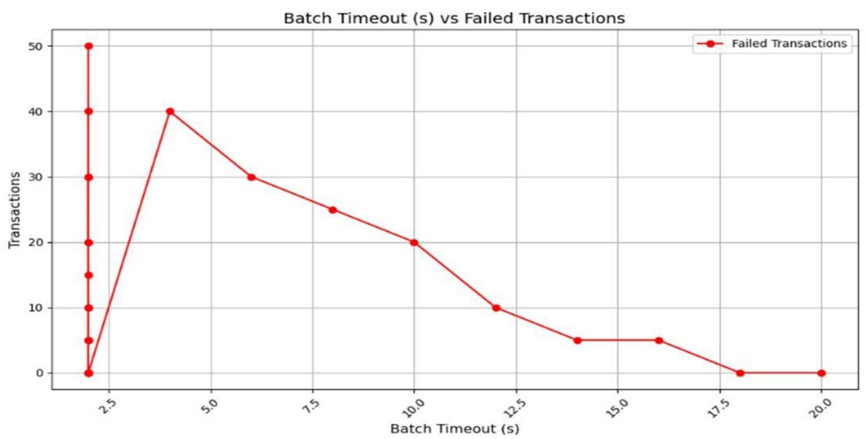


Figure 3.16: Batch Timeout with Failed Transactions

In general, increasing Block Size, Batch Size, and Batch Timeout all contribute to improved transaction success rates and a reduction in failed transactions. These findings indicate that larger configurations facilitate better processing and grouping of transactions, thereby enhancing the system's overall performance. However, the impact of each parameter is not linear, as the benefits become more pronounced after certain thresholds–14 MB block size, 45 batch size, and 18 seconds batch timeout–where the system consistently reaches optimal performance with near-perfect transaction success rates. The analysis highlights that optimal configuration settings, characterized by larger block sizes, batch sizes, and extended batch timeouts, result in improved transaction success and fewer failures. This suggests that careful tuning of these parameters can significantly enhance the performance of the system. However, while increasing these parameters can yield better performance, it is essential to balance them with latency requirements, particularly in environments where low-latency is critical.

Configuration with Successful Transactions and Failed Transactions

Figure 3.17 and Figure 3.18 shows the plots for Successful and Failed Transactions respectively across configurations (C1 to C28) highlighting the system's performance under different settings. Configurations like C7 to C10, C17 to C19,

and C27 to C28 achieve a 100% success rate, indicating optimized settings. Early configurations (C1 to C3) show slightly fewer successful transactions, suggesting inefficiencies that improve as parameters such as block size and batch size increase.

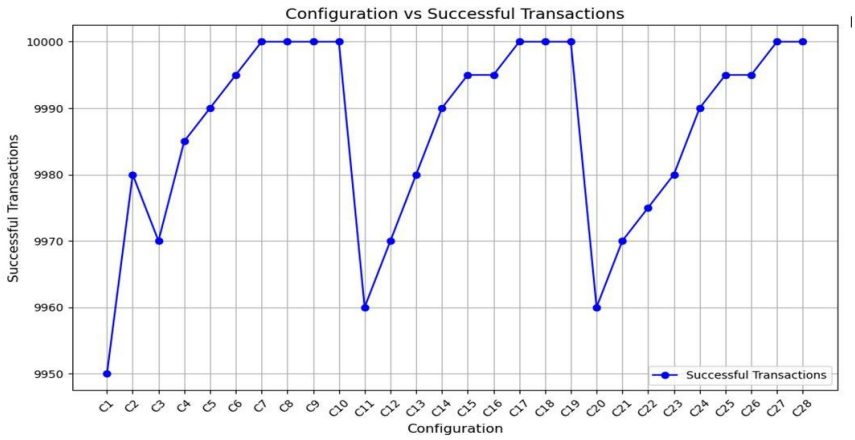


Figure 3.17: Configuration with Successful Transactions

Failed transactions are highest in configurations C1 to C6, but drop to zero starting from C7, showing the system's optimal performance when tuned. This shift underscores the importance of adjusting key parameters like block size and batch timeout to reduce failures and maximize throughput.

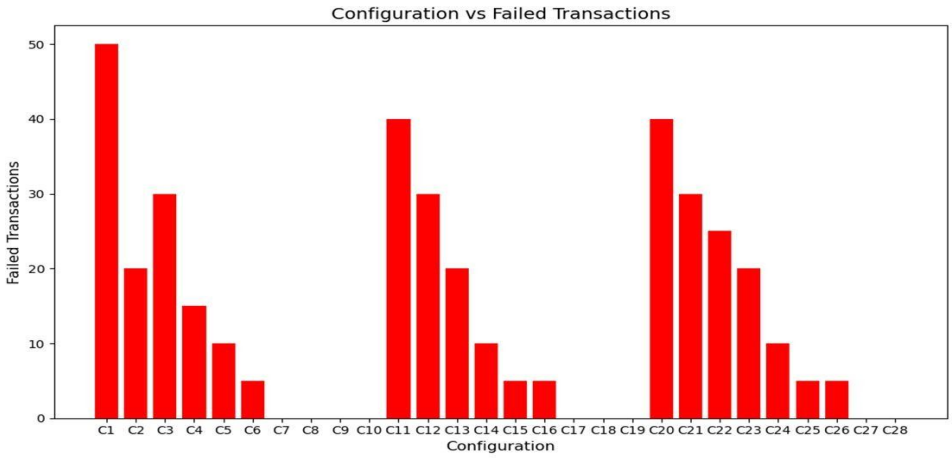


Figure 3.18: Configuration with Failed Transactions

CONCLUSION

The performance evaluation of the Hyperledger Fabric network presented in this study offers valuable insights into how varying block size, batch size, and batch timeout configurations impact key performance metrics such as throughput, latency, and transaction success rate. The experiments conducted demonstrated clear patterns in the network's behavior, allowing for an informed approach to optimizing Hyperledger Fabric deployments based on specific use case requirements.

Increasing the block size from 2 MB to 20 MB resulted in significant improvements in throughput, reaching a peak of 142.9 transactions per second (TPS) with minimal increases in latency. Additionally, a perfect transaction success rate of 100% was observed from a block size of 14 MB onwards, indicating that larger block sizes enhance performance without compromising reliability. Similarly, increasing the batch size from 10 to 55 transactions led to higher throughput, with a 100% success rate achieved from a batch size of 45 and beyond. These findings underscore the efficiency gains that can be realized through larger block and batch sizes in environments where throughput is prioritized.

The impact of batch timeout was also notable. Longer batch timeouts, ranging from 2 to 20 seconds, allowed the network to accumulate more transactions before processing, thereby improving throughput and success rates. While this configuration resulted in a slight increase in latency, the overall trade-off was favorable for high-throughput applications.

In summary, the study demonstrates that Hyperledger Fabric's performance can be significantly optimized through careful tuning of key parameters. Larger block sizes, batch sizes, and extended batch timeouts all contribute to enhanced throughput and transaction success, with only marginal increases in latency. For environments where high throughput is essential, such as financial transactions or large-scale data processing, these configurations provide a practical path to maximizing network efficiency. Conversely, for applications requiring low-latency performance, adjustments to smaller block sizes and shorter batch timeouts may be more appropriate, though at the cost of reduced throughput. The robustness and reliability of the Hyperledger Fabric network were evident across all configurations, highlighting its potential for diverse enterprise applications. These findings not only provide a framework for optimizing Hyperledger Fabric networks but also contribute to broader

efforts in improving blockchain performance and scalability.

REFERENCES

Alkhodre, A., Ali, T., Jan, S., Alsaawy, Y., Khusro, S., & Yasar, M. (2019). A Blockchain-based value added tax (VAT) system: Saudi Arabia as a use-case. *International Journal of Advanced Computer Science and Applications*, *10*(5),708–716. https://doi.org/10.14569/ijacsa.2019.0100588

Androulaki, E., Barger, A., Bortnikov, V., Muralidharan, S., Cachin, C., Christidis, K., De Caro, A., Enyeart, D., Murthy, C., Ferris, C., Laventman, G., Manevich, Y., Nguyen, B., Sethi, M., Singh, G., Smith, K., Sorniotti, A., Stathakopoulou, C., Vukolić, M., ... Yellick,

J. (2018). Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains. *Proceedings of the 13th EuroSys Conference, EuroSys 2018*,*2018-Janua*. https://doi.org/10.1145/3190508.3190538

Belov, A. V. (2018). Tax revenues, public investments and economic growth rates: evidence from Russia. *Journal of Tax Reform*, *4*(1), 45–56. https://doi.org/10.15826/jtr.2018.4.1.044

Dabbagh, M., Choo, K. K. R., Beheshti, A., Tahir, M., & Safa, N. S. (2021a). A survey of empirical performance evaluation of permissioned blockchain platforms: Challenges and opportunities. *Computers and Security*, *100*, 102078.

https://doi.org/10.1016/j.cose.2020.102078

Dabbagh, M., Choo, K. K. R., Beheshti, A., Tahir, M., & Safa, N. S. (2021b). A survey of empirical performance evaluation of permissioned blockchain platforms: Challenges and opportunities. *Computers and Security*, *100*(October 2020), 102078.

https://doi.org/10.1016/j.cose.2020.102078

Gorenflo, C., Lee, S., Golab, L., & Keshav, S. (2020). FastFabric: Scaling hyperledger fabric to 20 000 transactions per second. *International Journal of Network Management*, *30*(5), 1–18. https://doi.org/10.1002/nem.2099

Guggenberger, T., Sedlmeir, J., Fridgen, G., & Luckow, A. (2021). *An In- Depth Investigation of Performance Characteristics of Hyperledger Fabric*. http://arxiv.org/abs/2102.07731

Hang, L., Jamil, F., Jin, W., Kahng, H., & Kim, S. (2020). *Transaction Data Control for Blockchain Networks Transaction Data Control for Blockchain Networks So far , Bitcoin and Ethereum have met the Transactions per Second (TPS) bottom*. *December*.

Islam, N. (2021). *Relationship between tax revenues and economic growth in Bangladesh*. *March*.

Kadhm, O. I., Hamad, A. H., & Saeed, F. (2023). High Transaction Rates Performance Evaluation for Secure E-government Based on Private Blockchain Scheme. *Al-Khwarizmi Engineering Journal*, *19*(3), 44–55. https://doi.org/10.22153/kej.2023.06.002

Nanayakkara, S., Rodrigo, M. N. N., Perera, S., Weerasuriya, G. T., & Hijazi, A. A. (2021). A methodology for selection of a Blockchain platform to develop an enterprise system. *Journal of Industrial Information Integration*, *23*. https://doi.org/10.1016/j.jii.2021.100215

Rajput, A. R., Li, Q., Taleby Ahvanooey, M., & Masood, I. (2019). EACMS: Emergency Access Control Management System for Personal Health Record Based on Blockchain. *IEEE Access*, *7*, 84304–84317. https://doi.org/10.1109/ACCESS.2019.2917976

Shalaby, S., Abdellatif, A. A., Al-Ali, A., Mohamed, A., Erbad, A., & Guizani, M. (2020). Performance Evaluation of Hyperledger Fabric. *2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies, ICIoT 2020*, *April*, 608–613. https://doi.org/10.1109/ICIoT48696.2020.9089614

Yusoff, J., Mohamad, Z., & Anuar, M. (2022). A Review: Consensus Algorithms on Blockchain. *Journal of Computer and Communications*, *10*(09), 37–50. https://doi.org/10.4236/jcc.2022.109003