Optimal Resource Utilization in Hyperledger Fabric: A Comprehensive SPN-Based Performance Evaluation Paradigm

Carlos Melo[∓], Glauber Gonçalves[∓], Francisco A. Silva[∓] Leonel Feitosa[∓],
Iure Fé [∓], André Soares[∓], Eunmi Choi[⋇], Tuan Anh Nguyen[⊛], and Dugki Min[†],
[∓]Applied Research on Distributed Systems Lab (PASID), Federal University of Piauí (UFPI), Brazil
[⊛] Konkuk Aerospace Design-Airworthiness Research Institute (KADA), Konkuk University, Seoul 05029, South Korea
[⋇] School of Software, College of Computer Science, Kookmin University, Seoul 02707, South Korea
[†] Department of Computer Science and Engineering, Konkuk University, Seoul 05029, South Korea
Corresponding authors: {anhnt2407, dkmin}@konkuk.ac.kr

***Abstract*—Hyperledger Fabric stands as a leading framework for permissioned block-chain systems, ensuring data security and audit-ability for enterprise applications. As applications on this platform grow, understanding its complex configuration concerning various block-chain parameters becomes vital. These configurations significantly affect the system's performance and cost. In this research, we introduce a Stochastic Petri Net (SPN) model to analyze Hyper-ledger Fabric's performance, considering variations in block-chain parameters, computational resources, and transaction rates. We provide case studies to validate the utility of our model, aiding block-chain administrators in determining optimal configurations for their applications. A key observation from our model highlights the block size's role in system response time. We noted an increased mean response time, between 1 to 25 seconds, due to variations in transaction arrival rates.**

***Index Terms*—blockchain, Petri net, performance evaluation, capacity planning.**

I. Introduction

Blockchain, heralded as a transformative technology, promises significant potential in various sectors through secure, decentralized data storage (1), achieving transaction immutability, audibility, and consistency via asymmetric cryptography, consensus protocols, and peer-to-peer networks (2). To facilitate broader industrial adoption, there's a pivot towards private or permissioned blockchain architectures like Hyperledger Fabric, which enhance performance and security for consortium members across sectors, including healthcare (3) and finance (4). These networks focus on reducing data recording latency and increasing transaction throughput, with Hyperledger Fabric's performance contingent on infrastructure and blockchain configurations, notably block size and *timeout* for block creation.

This study introduces a Stochastic Petri Net (SPN) model to examine Hyperledger Fabric's performance across various configurations, addressing gaps in existing models by considering endorsement, ordering, and the *commit* phase. Our model, validated through case studies, aims to refine Hyperledger Fabric configurations for enhanced industry adoption, analyzing how blockchain configurations and computational capabilities impact transaction times and throughput.

Contributions of this work include:

- Developing an analytical model for Hyperledger Fabric blockchain configuration, providing operational efficiency insights before deployment;
- Conducting a detailed performance evaluation to identify transaction phase bottlenecks;
- Offering case studies as practical application evidence, complemented by a sensitivity analysis to identify key factors affecting Mean Response Time (MRT).

The paper progresses from reviewing relevant literature (Section II), describing the SPN performance model and its importance in Hyperledger Fabric transactions (Section IV), to showcasing case studies for insight (Section V), and concluding with future research avenues (Section VI).

II. Literature Review

Recent advancements in blockchain research have utilized various modeling techniques to analyze the performance and behavior of blockchain deployment architectures. Melo et al. (5; 6) employed Continuous Time Markov Chains (CTMC), Reliability Block Diagrams (RBD), and Stochastic Petri Nets (SPN) to assess the availability and resource provisioning for Hyperledger Fabric and Ethereum platforms, aiding in strategic blockchain application planning. Studies by Jiang et al. (7), Wu et al. (8), and Ke and Park (9) have concentrated on Hyperledger Fabric's performance metrics, with models focusing on throughput, discard rates, queue lengths, and mean

E-mails: {leonelfeitosa, andre.soares, ggoncalves, faps, casm}@ufpi.edu.br, {anhnt2407, dkmin}@konkuk.ac.kr, emchoi@kookmin.ac.kr

This research was partially supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education(No. 2020R1A6A1A03046811).; This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2021R1A2C2094943)

978-1-7281-8086-1/20/$31.00 ©2023 IEEE

response times. Additional research by Xu et al. (10), Yuan et al. (11), and Sukhwani et al. (12) explored throughput and latency effects through analytical models and Generalized Stochastic Petri Nets (GSPN), and Stochastic Reward Networks (SRN), respectively, to analyze transaction steps.

Our contribution is an SPN model that evaluates Hyperledger Fabric's performance metrics, including *maximum block size rate* and *timeout block rate*, alongside a calibration approach for these parameters. Our work, summarized in Table I, provides a detailed examination of timeout and block size effects, especially dissecting the ordering phase to highlight block formation issues related to timeouts, offering a more nuanced understanding compared to previous studies.

TABLE I: Related Works

Publication	MRT	Throughput	RU	MSBR	TBR	DTP
Sukhwani et al. (12)	✓	✓	✓	✓	✓	✗
Jiang et al. (7)	✓	✓	✗	✗	✗	✓
Yuan et al. (11)	✓	✓	✗	✓	✓	✗
Xu et al. (10)	✓	✗	✗	✓	✗	✗
Melo et al. (6)	✗	✗	✓	✗	✗	✗
Ke and Park (9)	✓	✗	✗	✗	✗	✗
Wu et al. (8)	✓	✗	✗	✗	✗	✗
Melo et al. (5)	✗	✗	✓	✗	✗	✗

Metrics abbreviations: Mean Response Time (MRT), Resource Utilization (RU), Maximum Size Block Rate (MSBR), Timeout Block Rate (TBR), Discarding transactions probability (DTP)

III. System Architecture

A. Hyperledger Fabric Framework

The Hyperledger Fabric (HLF) has emerged as a leading blockchain platform, supported by over 35 institutions and more than 200 developers.[1] Unlike public blockchains such as Bitcoin and Ethereum, HLF adopts an execute-order-validate model for transactions to improve scalability (13). It allows for diverse configurations, primarily between clients and service providers, with the latter significantly impacting system performance. HLF leverages container technologies for managing "chaincodes," or smart contracts, which encapsulate the business logic necessary for deployment, while clients interact through SDKs in languages like Javascript or Python. The HLF transaction pathway involves endorsement, ordering, and commit phases, functioning across multiple network nodes as depicted in Figure 1, showcasing the integral steps from client initiation to network processing.

The transaction process in Hyperledger Fabric initiates with a client application submitting a proposal to a network node, followed by seeking endorsements from more than half of the nodes. Post-endorsement, the transaction progresses to the *ordering* phase, where it awaits the formation of a new block upon hitting a preset timeout or block size. During the *endorsement* phase, the transaction undergoes simulation, receiving approval or rejection. Subsequently, blocks enter the *commit* phase for validation and integration into the blockchain, updating the global state for swift queries. Illustrated in Figure 2 is a typical configuration featuring two endorsing nodes, one ordering node, and two committing nodes, showcasing the system's scalability and the significance of modeling for optimizing pre-deployment performance and cost efficiency.

[1]For more details, see *https://hyperledger-fabric.readthedocs.io*.

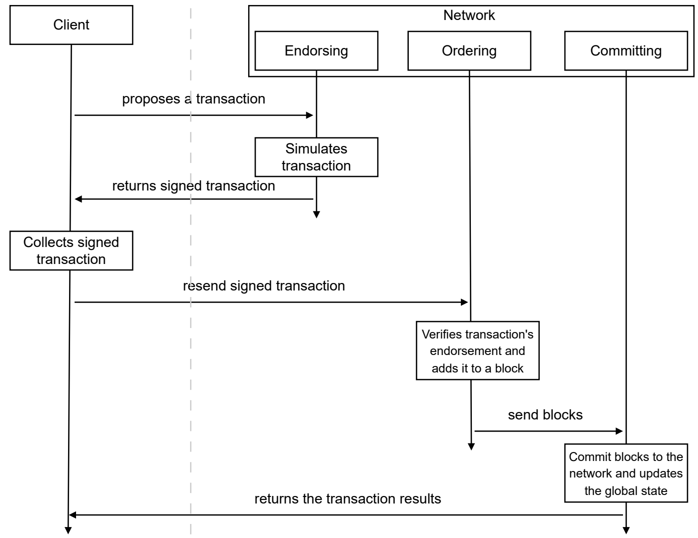


Fig. 1: Transaction's flow in the Hyperledger Fabric platform.

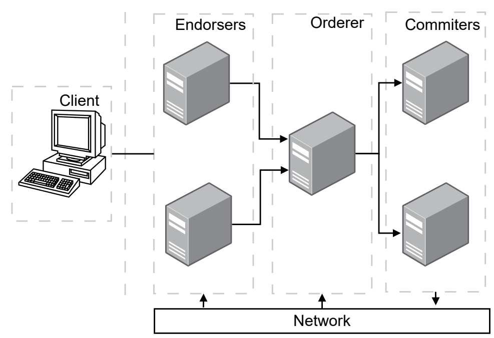


Fig. 2: Architecture Overview.

IV. Base Model

Our research presents a Stochastic Petri Nets (SPN) model designed to evaluate Hyperledger Fabric's performance, a permissioned blockchain. This model enables administrators to adjust parameters optimally before real-world deployment. Illustrated in Figure 3, the SPN model simulates transactions through Hyperledger Fabric's three principal phases: *endorsement, ordering*, and *commit*, spanning across N nodes. For simplification, our model posits that each phase occurs on two nodes, with the exception of the ordering phase, which is managed by a single node. Transaction initiation by an application and its consistent arrival rate are depicted by a deterministic gray transition, labeled as the arrival delay (AD).

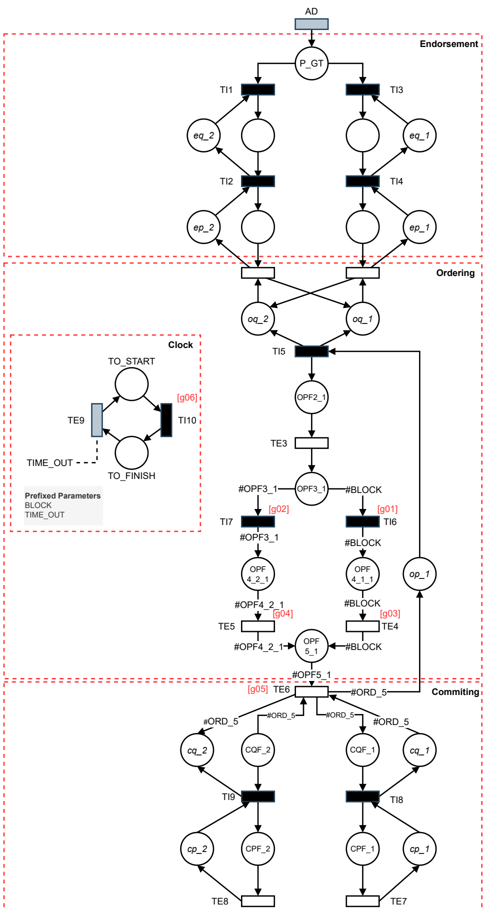


Fig. 3: SPN model for processing transactions on Hyperledger Fabric.

Type	Element	Description
Places	P_GT	Wait for new transactions
Transitions Deterministic	AD TE9	Transactins Arrival Time TIME_OUT assigned time
Timed Transitions	TE1, TE2 TE3 TE4, TE5 TE6 TE7, TE8	Endorsement time Block assembly time Block processing time Commit process Time for committing
Immediate Transitions	TI1, TI3 TI2, TI4 TI5 TI6 TI7 TI8, TI9 TI10	Endorsement queue Endorsement process Ordering queue Ordering due to block size Ordering due to TIME_OUT Commit Process TIME_OUT's restart
Token's Places	eq1_1, eq1_2 ep1_1, ep1_2 oq_1 op_1 cq1_1, cq1_2 cp1_1, cp1_2	Endorsement queue capacity Endorsement process capacity Ordering queue's capacity Ordering process capacity Commit queue's capacity Commit process capacity

TABLE II: Model's elements description, including places and transitions.

Trans.	ID	Guard Expression
TI6	g01	#OPF3_1>0
TI7	g02	#TO_FINISH=1)AND(#OPF3_1>0)
TE4	g03	#OPF4_1_1>0
TE5	g04	#OPF4_1_2>0
TE6	g05	#OPF5_1>0
TI10	g06	#OPF5_1>0

TABLE III: Guard expressions for transitions (Trans.) whose indices (ID) are highlighted in red in the model.

In the Hyperledger Fabric framework, the transaction lifecycle commences with the *endorsement* phase, where a token at `P_GT` initiates processing on one of two computational nodes via transitions `TI1` or `TI3`. Nodes, represented by two triangles for input and processing queues, operate differently: the input queue transitions instantly, while the endorsement process is subject to timed transitions `TE1` and `TE2`, reflecting node capacities like `EP_1` and leading to queue formations when exceeded. The *ordering* phase, handled by a single node, follows with a similar queue setup. It sorts transactions based on `op_1` capacity, with initial processing marked by `TE3` for block formation. Tokens at `OPF3_1` lead to either complete or partial blocks, depending on the `#BLOCK` parameter or a timeout. These blocks then proceed through `TE4` and `TE5`. Upon reaching `OPF5_1`, the system's `Clock` is reset. The *commit* phase is triggered by transition `TE6`, involving all designated nodes and culminating with `TE7` and `TE8`. The model's components and operational conditions are elaborated in Tables II and III, respectively.

A. Metrics

The mean response time (*MRT*) is derived using Little's Law(14) as $\frac{\texttt{TransactionsInProgress}}{\texttt{Arrival\_Rate}}$, correlating the system's average transactions in progress with the arrival rate, where $Arrival\_Rate = \frac{1}{Arrival\_Delay}$. This calculation presupposes a stable system, where the transaction rate does not exceed the server's processing capability. To determine the system's transactions in progress, we sum the tokens in places representing ongoing transactions, using $E(Place)$ for the expected number of tokens per place, calculated as $E(Place) = (\sum_{i=1}^{n} P(m(Place) = i) \times i)$. The discard rate probability is determined by the likelihood of n tokens in a place being zero, expressed as `P((EQ_1=0) ∧ (EQ_2=0))`, with $\wedge$ signifying a logical AND. Resource utilization is assessed by comparing the expected tokens in a queue or process to its capacity, exemplified by the endorsement step's processing utilization as $\frac{\texttt{E(EP\_1)}}{\texttt{ep\_1}}$. System throughput, particularly at the *commit* phase performed across N computers, is the average exit rate, calculated by dividing the expected tokens by the service time, $\frac{\texttt{E(CPF\_1)}}{\texttt{t(TE7)}}$. Forking paths from `OPF3_1` determine transition rates for timeouts or complete blocks, adhering to the aforementioned calculation principles. Additional metrics, BLOCK_CALL_RATE and TIME_OUT_CALL_RATE, are detailed in equations 1 and 2, respectively.

$$\textbf{BLOCK\_CALL\_RATE} = \frac{\texttt{E(OPF\_1\_1)}}{\texttt{t(TE4)}} \tag{1}$$

TABLE IV: Initial model configuration parameters used in case studies.

Type	Parameters	Values
Capacity	ep_1, ep_2, op_1, cp_1, cp_2	6
	eq_1, eq_2, oq_1, cq_1, cq_2	100
Time	TE1, TE2, TE3	5 ms
	TE6	10 ms
	TE7, TE8	80 ms
	TE4, TE5	2 ms

$$\textbf{TIME\_OUT\_CALL\_RATE} = \frac{\texttt{E(OPF\_2\_1)}}{\texttt{t(TE5)}} \tag{2}$$

V. Results

This section presents four case studies, each focusing on distinct parameters to demonstrate our model's effectiveness and its applicability to real-world systems. Using the Mercury tool (15) for model representation and numerical analysis, we maintain the architecture and computational node setup as described in Section IV. Initial model parameters, detailed in Table IV, include queue capacity (cq), processing capacities for the endorsement (ep), ordering (op), and commit (cp) phases, along with service times for transitions (TE). These parameters, informed by Hyperledger platform testing, reflect computational nodes' processing capabilities and total memory allocations for queue capacities in each phase, highlighting that parameter values may differ based on the computational environment examined.

A. Case Study 01 - Committer Capacity

In our first case study, we examined the impact of varying processing capacities (`cp_1` and `cp_2`) during the *commit* phase, testing configurations with each node's capacity set to 2, 4, and 6 containers. To comprehensively understand the effects, we also adjusted the transaction arrival rate from a low of 0.0025 requests per millisecond (req/ms) to a high of 0.3 req/ms, in increments of 0.01565 req/ms. For analytical clarity, we opted for a configuration where each block contained a single transaction, setting a long *timeout* (`TIME_OUT` = 10000ms) and a minimal block size (`BLOCK` = 1). This approach primarily focused the model's dynamics on the generation of complete blocks.

Figure 4 presents the outcomes of our first case study through eight distinct plots, highlighting the effects of parameter changes on system performance.

Utilization trends across Figures 4a, 4b, and 4c show uniform increases with rising arrival rates, with *commit* phase utilization notably accelerating as computational capacities are adjusted. An important observation is that beyond a 0.05 req/ms arrival rate in the *commit* phase, changing capacities no longer impacts utilization, indicating a threshold for optimizing hardware investments.

The system's discard rate at the endorsement phase (Figure 4d) parallels the endorsement utilization graph, suggesting that high utilization at this stage leads to increased discard probabilities due to congestion.

Mean Response Time (MRT) analysis (Figure 4e) reveals performance degradation with increasing arrival rates, especially for a capacity (`cp_n`) of 2. For capacities of 4 and 6, a significant MRT difference emerges at a 0.05 req/ms arrival rate, indicating the effect of capacity on response times.

The system's flow behavior (Figure 4f) initially increases then stabilizes, limited by *commit* phase utilization saturation, indicating a cap on performance gains beyond certain utilization levels.

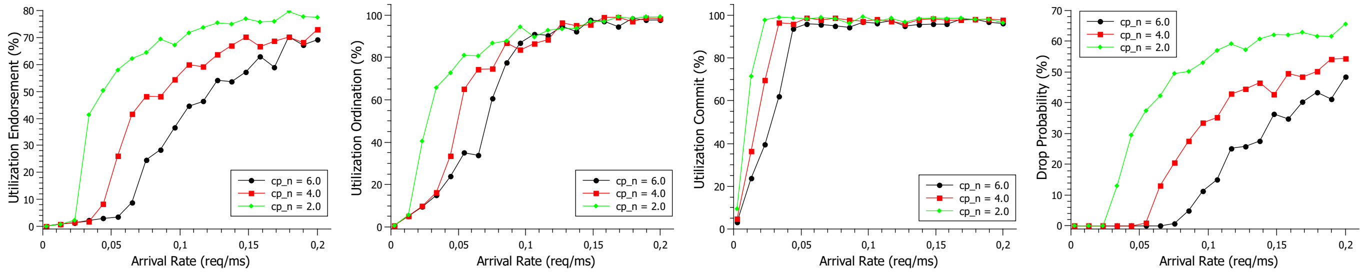
Timeout activation rates (Figure 4g) mostly remain negligible due to the extended 10000ms timeout, favoring complete block formation. Conversely, block activation rates (Figure 4h) are expected to correlate with the arrival rate, but are constrained at lower `cp_n` values, reflecting limited throughput due to restrictive capacities.

This case study underscores the *commit* phase's critical role in system performance, demonstrating that optimal capacity tuning is essential to prevent performance drops. Despite increased capacities, the similarity in MRT for `cp_n` values of 4 and 6 suggests that a `cp_n` of 4 could be adequate for the studied arrival rates, offering an opportunity for resource optimization.

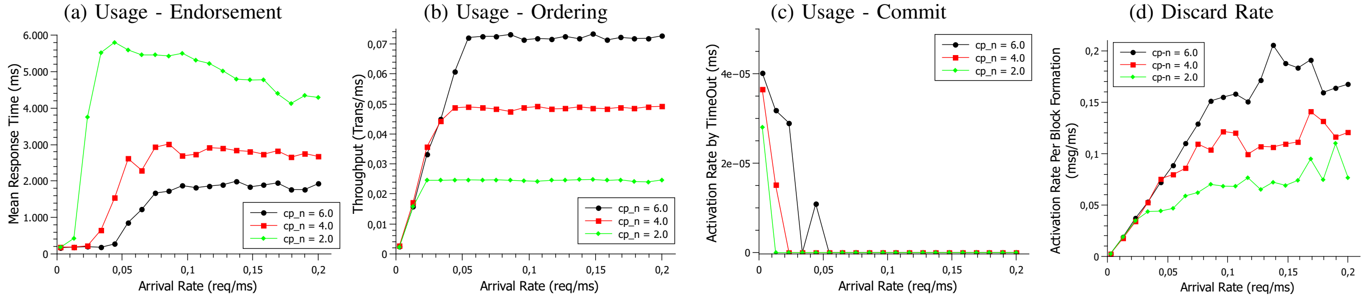
B. Case Study 02 - Evaluating the Effects of Block Size and Timeout

In our second case study, our objective is twofold. Initially, we adjust the block size while maintaining a substantial *timeout* (`TIME_OUT` = 10000ms). Subsequently, we modulate the *timeout*, keeping the *block size* constant at (`BLOCK` = 10). The intention behind this study is to methodically discern the individual ramifications of each parameter on the system's behavior. For this analysis, we standardized the arrival rate at 0.1 req/ms and adhered to the parameters delineated in Table IV. The outcomes of this investigative endeavor are illustrated in Figure 5.

Analyzing Figures 5a and 5b reveals that system utilization increases with block size and *timeout* adjustments, with *timeout* changes exerting a stronger impact. This leads to a decrease in committer utilization due to bottlenecks, more so with *timeout* alterations that drain resources faster than block size modifications. Figures 5c and 5d demonstrate varying trigger rates, identifying critical inflection points at a block size of 7 and a *timeout* of 5000ms, beyond which discard rates and mean response times (MRT) significantly increase, with MRT oscillating between 1000ms and 25000ms and discard rates reaching up to 80%. Throughput trends, as seen in Figures 5i and 5j, mirror the decline in committer utilization, highlighting system inefficiencies with an initial throughput of 0.07req/ms against an arrival rate of 0.1 req/ms. This analysis underscores the critical impact of block size and *timeout* on system performance, indicating pronounced bottlenecks, especially in the ordering phase. A block size increment to 7 or a *timeout* beyond 5000ms marks a significant turning point, where even minor changes can dramatically influence MRT, showcasing the delicate balance required in configuring these parameters.

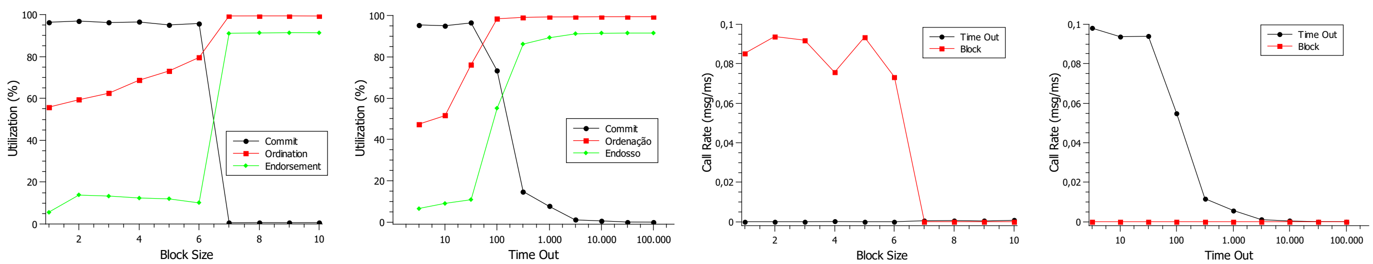


(a) Usage - Endorsement (b) Usage - Ordering (c) Usage - Commit (d) Discard Rate

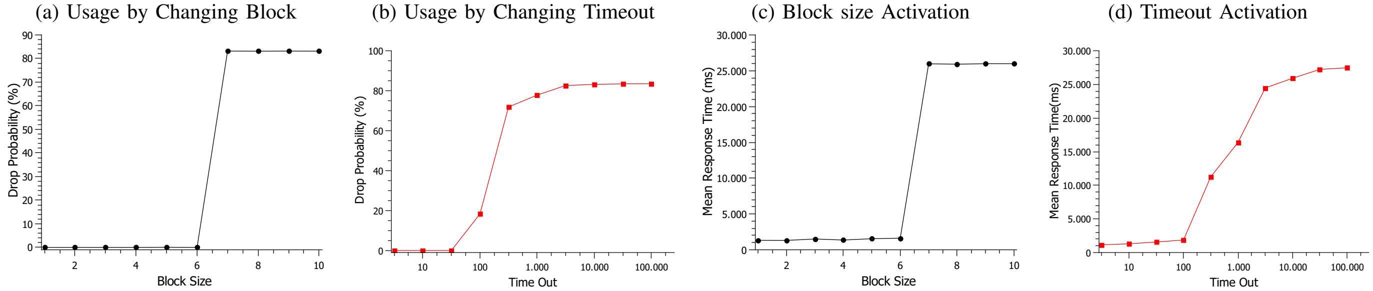


(e) MRT (f) Flow (g) Timeout Activation (h) Size Activation Block

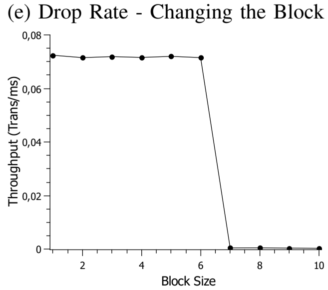
Fig. 4: Case Study 01 - Committer Capacity



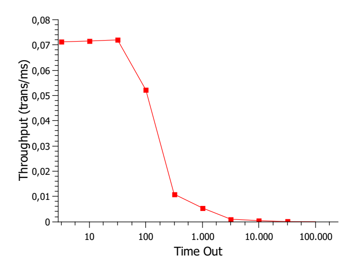
(a) Usage by Changing Block (b) Usage by Changing Timeout (c) Block size Activation (d) Timeout Activation



(e) Drop Rate - Changing the Block (f) Drop Rate - Changing Timeout (g) MRT - Changing Block (h) MRT - Changing Timeout



(i) Flow - Changing Block



(j) Flow - Changing Timeout

Fig. 5: Case Study 02 Results Varying Block Size and Timeout

C. Case Study 03 - Dissecting the Interplay between Block Size and Timeout

In our previous analysis, we separately evaluated the impact of block size and *timeout* on system performance metrics, maintaining a *timeout* of 10000ms to nullify its trigger rate while altering block size. Now, we shift focus to varying the *timeout* across the same range, with the block size fixed at a lower value (BLOCK = 6), to uncover the intricate dynamics between these two parameters. Our objective is to understand how changes in *timeout* influence both the *timeout* and block activation rates, aiming to reveal their complex interplay as depicted in Figure 6b.

The graph shows a clear intersection of two distinct trajectories as the *timeout* parameter increases, leading to a noticeable decrease in the activation rate per *timeout*. This trend suggests that a longer *timeout* period makes triggering by *timeout* less frequent, increasing the chances of forming a complete block. The challenge of precisely adjusting these interrelated parameters, due to their complex interactions, has been underlined by researchers like Sukhwani et al. (12). Consequently, the model in this study provides crucial insights, enabling more accurate predictions of blockchain network behavior. From our third case study, key findings include:

1) The model effectively identifies the point where the formation of complete blocks overtakes partial blocks.
2) A consistent trigger rate for blocks indicates the system has reached a high queuing state.
3) Setting `TIME_OUT` to 10ms leads to primarily partial blocks.
4) A `TIME_OUT` of 10000ms ensures the formation of complete blocks with 6 transactions each.

This underscores the challenge of managing complete and partial block formations, affecting transaction time predictability. Blockchain network administrators must carefully balance block size and *timeout* settings against transaction rates.

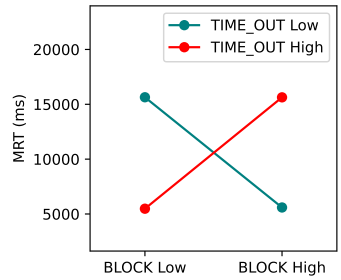
D. Case Study 04 - Sensitivity Analysis

This case study uses a 2^k Design of Experiments (DoE) for Sensitivity Analysis to identify the key factors and their interactions affecting Mean Response Time (MRT), guided by previous insights. An effect plot in Figure 6c shows the impact of different variables on MRT. Initial findings reveal `TIME_OUT` as the most influential factor, with `block size` and its interaction with `TIME_OUT` also significantly impacting system performance.

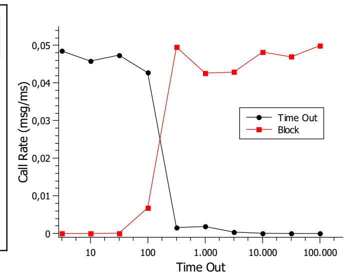
The values in this sensitivity analysis varied from **low** to **high**, indicating a minimum and a maximum value for each factor (2^k). Figure 6a presents the strength plot for the relationship the standout, which is the relationship between block and TIME_OUT.

The final case study reveals that the relationship between *timeout* and block size significantly impacts system performance, with Mean Response Time (MRT) potentially escalating from 5000ms to over 15000ms, a 200% increase. Such delays, far exceeding contemporary expectations for transaction speeds, could adversely affect user experience, with stakeholders possibly preferring public blockchains for their efficiency and lower operational costs.

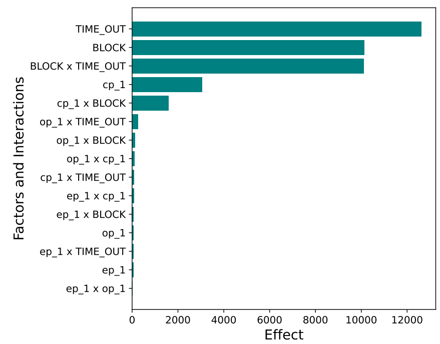
However, the preference for private or permissioned blockchains among many governmental and industrial entities,



(a) Strength plot for Block x Timeout relationship.



(b) Interaction between Block and Timeout



(c) Factors and Interactions and their respective effect.

Fig. 6: Comparative analysis of Block x Timeout relationship, factors, and interactions.

driven by the need for strict data governance and access control, remains strong. This necessitates improvements in private blockchain technology to meet performance expectations. Our model serves as a crucial tool for addressing these challenges, helping to optimize blockchain configurations for enhanced performance and user satisfaction.

VI. Remarks

This paper introduces an SPN model designed to analyze the performance of permissioned blockchains, specifically within the Hyperledger Fabric framework. The model evaluates key performance indicators such as Mean Response Time (MRT), Throughput (TP), and Utilization, and assesses the impact of blockchain parameter changes—block size, *timeout*, and transaction arrival rates—on the likelihood of transaction discards.

Through various parameter adjustments, including resource capacities and queuing dynamics, the model identifies critical areas for performance improvement. Four case studies demonstrate how blockchain configurations and the computational capacities of system components affect network latency and throughput, offering insights into optimizing Hyperledger Fabric's performance.

Looking ahead, plans include enhancing the model to incorporate weighted decision-making for protocol steps distribution among network machines based on probabilistic heuristics. This future direction aims to further dissect performance implications, potentially guiding subsequent case studies.

References

[1] X. Xu, I. Weber, and M. Staples, *Architecture for blockchain applications*. Springer, 2019.

[2] F. Greve, L. Sampaio, J. Abijaude, A. A. Coutinho, I. Brito, and S. Queiroz, "Blockchain e a Revolução do Consenso sob Demanda," in *Proc. of SBRC Minicursos*, 2018.

[3] E. J. Aguiar, A. J. dos Santos, R. I. Meneguette, E. Robson, and J. Ueyama, "A blockchain-based protocol for tracking user access to shared medical imaging," *Future Generation Computer Systems*, vol. 134, pp. 348–360, 2022.

[4] R3, "Spunta case study. fast and transparent interbank reconciliation powered by distributed ledger technology," https://www.r3.com/wp-content/uploads/2020/11/Corda_Spunta_Case_Study_R3_Nov2020.pdf, 2020, accessed: 2022-07-06.

[5] C. Melo, J. Araujo, J. Dantas, P. Pereira, and P. Maciel, "A model-based approach for planning blockchain service provisioning," *Computing*, vol. 104, no. 2, pp. 315–337, 2022.

[6] C. Melo, J. Dantas, P. Pereira, and P. Maciel, "Distributed application provisioning over ethereum-based private and permissioned blockchain: availability modeling, capacity, and costs planning," *The Journal of Supercomputing*, vol. 77, no. 9, pp. 9615–9641, 2021.

[7] L. Jiang, X. Chang, Y. Liu, J. Mišić, and V. B. Mišić, "Performance analysis of hyperledger fabric platform: A hierarchical model approach," *Peer-to-Peer Networking and Applications*, vol. 13, pp. 1014–1025, 2020.

[8] O. Wu, S. Li, L. Liu, H. Zhang, X. Zhou, and Q. Lu, "Performance modeling of hyperledger fabric 2.0," in *Proceedings of the International Conference on Evaluation and Assessment in Software Engineering 2022*, 2022, pp. 357–365.

[9] Z. Ke and N. Park, "Performance modeling and analysis of hyperledger fabric," *Cluster Computing*, pp. 1–19, 2022.

[10] X. Xu, G. Sun, L. Luo, H. Cao, H. Yu, and A. V. Vasilakos, "Latency performance modeling and analysis for hyperledger fabric blockchain network," *Information Processing & Management*, vol. 58, no. 1, p. 102436, 2021.

[11] P. Yuan, K. Zheng, X. Xiong, K. Zhang, and L. Lei, "Performance modeling and analysis of a hyperledger-based system using gspn," *Computer Communications*, vol. 153, pp. 117–124, 2020.

[12] H. Sukhwani, N. Wang, K. S. Trivedi, and A. Rindos, "Performance modeling of hyperledger fabric (permissioned blockchain network)," in *Proc. of NCA*. IEEE, 2018, pp. 1–8.

[13] E. Androulaki and et al., "Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains," in *Proc. of the EuroSys Conference*, 2018.

[14] R. Jain, *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling*. John Wiley & Sons, 1990.

[15] P. Maciel, R. Matos, B. Silva, J. Figueiredo, D. Oliveira, I. Fé, R. Maciel, and J. Dantas, "Mercury: Performance and dependability evaluation of systems with exponential, expolynomial, and general distributions," in *Proc. of PRDC*. IEEE, 2017, pp. 50–57.