Titre: Title:	Toward More Performant and Efficient Decentralized Applications on Blockchain Technologies
Auteur: Author:	Mohammadreza Rasolroveicy
Date:	2022
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Rasolroveicy, M. (2022). Toward More Performant and Efficient Decentralized Applications on Blockchain Technologies [Thèse de doctorat, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/10733/

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

TOWARD MORE PERFORMANT AND EFFICIENT DECENTRALIZED APPLICATIONS ON BLOCKCHAIN TECHNOLOGIES

MOHAMMADREZA RASOLROVEICY
Département de génie informatique et génie logiciel

Thèse présentée en vue de l'obtention du diplôme de *Philosophiæ Doctor*
Génie informatique

Décembre 2022

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Cette thèse intitulée :

TOWARD MORE PERFORMANT AND EFFICIENT DECENTRALIZED APPLICATIONS ON BLOCKCHAIN TECHNOLOGIES

présentée par **Mohammadreza RASOLROVEICY**
en vue de l'obtention du diplôme de *Philosophiæ Doctor*
a été dûment acceptée par le jury d'examen constitué de :

Foutse KHOMH, président
Heng LI, membre et directeur de recherche
Marios-Eleftherios FOKAEFS, membre et codirecteur de recherche
Marin LITOIU, membre
Sotirios LIASKOS, membre externe

DEDICATION

To my lovely parents,

To my beautiful sister,

To my wonderful friends,

For their endless love, support, and encouragement

ACKNOWLEDGEMENTS

I would like to express my warm regards and gratitude towards my supervisor Prof. Marios Fokaefs, who has been my advisor for the entire of my doctoral program. I also would like to appreciate his sincere mentorship and his marvellous lucidity and sublime way of thinking towards the development of morals and thoughts during the long path of research to its full fruitfulness to reap the benefits. Sincerely, because of his efficient and timely assistance, as well as his sagacity, acumen, and endeavour, I could complete this a four-year journey. I am eternally grateful to him for all of his encouragement, unending support, and belief in me and my abilities, as well as for sharing his invaluable time to lead my research. "I am indebted to my father for living, but to my teacher for living well."

In addition, I'd like to express my gratitude to my co-supervisor, Prof. Heng Li, for the trust, time, and assistance he provided me with throughout the course of my studies, as well as for the opportunity to serve as a teaching assistant for his class and gain a great deal of knowledge from him.

Also, I would like to express my gratitude to all of the members of my committee, including Prof. Foutse Khomh, Prof. Marin Litoiu, and Prof. Sotirios Liaskos, for taking the time out of their busy schedules to examine and assess my dissertation.

I would want to pay gratitude to all of my wonderful family and friends. My thanks go mostly to my parents, Ziba and Morteza, whose words of love, encouragement, teaching me method facing all the challenges of life, and push for perseverance continue to ring in my ears. I am grateful that they are the people who gave me life.

My sister, Shiva, is a one-of-a-kind individual who has never abandoned me. My thanks and admiration for you cannot be adequately expressed via words. You have been a source of motivation for me, an ally to me, and a teacher to me. You have instilled in me qualities such as self-belief and tenacity via your teachings.

RÉSUMÉ

Les avantages des technologies blockchain en termes d'intégrité, d'immuabilité et de transparence les ont aidées à gagner du terrain. Avec la blockchain, les utilisateurs disposent d'une infrastructure sécurisée, décentralisée et fiable pour enregistrer leurs données indéfiniment. La nature décentralisée et la fiabilité de la blockchain sont rendues possibles grâce à l'utilisation de consensus, éliminant le besoin de faire confiance aux intermédiaires. Pour résoudre les problèmes de performance et d'évolutivité de la technologie de blockchain traditionnelle basée sur le consensus "preuve de travail", plusieurs implémentations de blockchain qui utilisent d'autres consensus ont été développées. Divers outils d'analyse comparative ont été introduits pour évaluer les performances des systèmes de blockchain, permettant la comparaison et la compréhension de ces technologies. Cependant, ces solutions sont soit limitées à des implémentations blockchain spécifiques, soit nécessitent des configurations complexes. Ils donnent également la priorité aux modèles d'évaluation d'unité de travail (charge constante), ce qui peut être n'est pas intuitif pour les instances à exécution plus longue sous des charges de travail continues.

Notre objectif ultime dans cette thèse est d'identifier et de mettre en œuvre un écosystème optimal pour les applications décentralisées avec un grand nombre d'utilisateurs basé sur la blockchain.

La première contribution de cette thèse est BlockCompass, un outil complet de benchmarking pour les technologies blockchain qui est également flexible en termes de configuration et d'extension. BlockCompass peut comparer l'efficacité de plusieurs implémentations de blockchain et consensus sous des charges de travail continues et fluctuantes. De plus, nous présentons les résultats d'une enquête d'utilisabilité sur la facilité d'utilisation de BlockCompass dans l'analyse comparative de la blockchain. Notre outil d'analyse comparative proposé peut évaluer différentes plates-formes de blockchain en termes de débit, d'évolutivité et d'utilisation des ressources.

De plus, nous partageons les résultats de nombreuses enquêtes empiriques que nous avons menées sur plusieurs technologies de blockchain, des comparaisons entre les technologies de blockchain privées et publiques et des évaluations de différentes technologies de blockchain publiques. Nos résultats peuvent guider les praticiens pour choisir la blockchain la plus adaptée à une charge de travail spécifique et à une application décentralisée.

De plus, nous présentons de nouveaux systèmes auto-adaptatifs pour améliorer la qualité et l'efficacité des applications décentralisées. Notre framework peut choisir et déployer

dynamiquement l'algorithme de consensus optimal en temps réel, en tenant compte du nombre actuel de nœuds et de l'utilisation du processeur.

Enfin, nous proposons IntelliChain, une architecture intelligente et évolutive pour les jetons non fongibles (NFT), basée sur les résultats empiriques obtenus à partir de notre outil de benchmarking. Ce cadre facilite une infrastructure performante et rentable pour enregistrer et échanger des actifs numériques via une architecture pilotée par les événements et des réseaux de neurones.

En conclusion, nos contributions dans cette thèse fournissent des outils auto-adaptatifs et d'aide à la décision et éclairent les orientations futures pour améliorer les performances et l'évolutivité des applications décentralisées sur les technologies blockchain.

ABSTRACT

The advantages of blockchain technologies in terms of integrity, immutability, and transparency have helped them gain traction. With blockchain, consumers have a secure, decentralized, and trustworthy infrastructure to record their data indefinitely. Blockchain's decentralized nature and enhanced reliability are made possible by its consensus characteristics, eliminating the need for trust in intermediaries. Multiple blockchain technologies using various consensus protocols have been developed to address traditional blockchain technology's performance and scalability issues based on Proof of Work. Various benchmarking tools have been introduced to evaluate the performance of blockchain systems, allowing for comparison and understanding of these technologies. However, these solutions are either limited to specific blockchain platforms or require complex configurations. They also prioritize single executable work unit evaluation models, which might be counter-intuitive for longer-running instances under continuous workloads.

Our ultimate goal in this dissertation is to identify and implement an optimal ecosystem for decentralized applications with a large number of users on the blockchain.

The first contribution of this thesis is BlockCompass, a comprehensive benchmarking tool for blockchain technologies that are also flexible in terms of configuration and extension. BlockCompass can compare the efficiency of multiple blockchain platforms and consensus protocol setups under continuous and fluctuating workloads. Additionally, we present the results of a usability survey about the convenience and facility offered by BlockCompass in blockchain benchmarking. Our proposed benchmarking tool can evaluate different blockchain platforms in terms of throughput, scalability, and resource utilization.

In addition, we share the results of many empirical investigations we conducted on multiple blockchain technologies, comparisons between private and public blockchain technologies, and evaluations of different public blockchain technologies. Our results can guide practitioners on which blockchain technology can be more suitable for a specific workload and decentralized application.

Furthermore, we present novel self-adaptive systems to improve the quality and efficiency of decentralized applications. Our framework can dynamically choose and deploy the optimal consensus algorithm in real-time, considering the current number of nodes and CPU utilization.

Finally, we propose IntelliChain, an intelligent and scalable architecture for Non-Fungible Tokens (NFTs), based on the empirical findings obtained from our benchmarking tool. This

framework facilitates a performant and cost-efficient infrastructure for recording and trading digital assets through event-driven architecture and neural networks.

In conclusion, our contributions in this dissertation provide self-adaptive and decision support tools and shed light on future directions to improve the performance and scalability of the decentralized applications on blockchain technologies.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ACRONYMS

dApp.	decentralized application
PoW	Proof of Work
PoET	Proof of Elapsed Time
DAG	Directed Acyclic Graph
PBFT	Practical Byzantine Fault Tolerance
PoS	Proof of Stake
aBFT	Asynchronous Byzantine Fault Tolerance
TBFT	Tendermint Practical Byzantine Fault Tolerance
NFT	Non Fungible Token
RPC	Remote Procedure Call
IPFS	InterPlanetary File System
P2P	Peer to Peer
ERC	Ethereum Request for Comments
DDoS	Distributed Denial-of-Service
VM	VM Virtual Machine
AI	Artificial Intelligence
ML	Machine Learning
XGB	eXtreme Gradient Boosting
RF	Ranfom Forest
DT	Decision Tree
GRU	Gated Recurrent Unit
LSTM	Long Short-Term Memory
Bi-LSTM	Bidirectional Long Short-Term Memory
MAE	Mean Absolute Error
RMSE	Root Mean Square Error
NN	Neural Network
MAPE	Monitoring, Analysis, Planning and Execution
SA	Self-adaptive

CHAPTER 1 INTRODUCTION

Blockchain technology is gaining ever-increasing popularity among researchers and practitioners because of its consistency, high availability, transparency, and immutability [1]. Blockchain technology has begun to benefit various fields, including, but not limited to, decentralized finance (DeFi), retail, healthcare, manufacturing, Internet of Things (IoT), and art [2, 3]. The core concept of blockchain is its consensus algorithm which protects the data against undesired alteration. Blockchain, which is also known as a decentralized ledger technology (DLT), is a set of peer-to-peer nodes which do not trust each other. In the blockchain, in order to have a transaction confirmed, the majority of peers in the network should agree to add it to the block, according to the consensus protocol. Each node keeps replicas of the data and has to validate the transaction into the block [4–6].

1.1 Background

In Blockchain systems, each data block that is submitted to the network will be verified by the peers of the network, and then it will be broadcasted to all members. Moreover, the leaders in Blockchain only have permission to append the block to the network, and these leaders are chosen by a consensus mechanism [7]. Unlike traditional distributed databases that provide read/write access to the data, in blockchain, when data has been stored, it can neither be altered nor removed. This technique can effectively prevent attacks, like the Mirai botnet attack, because of non-repudiation: malicious activities are stored and cannot be erased or tampered with, and they can be traced [8]. Consensus protocol is one of the main core layer elements of in blockchain technologies. Consensus in blockchain ensures the consistency and correctness of the data among the peers of the network [9]. The objective of the consensus algorithm is to reach an agreement on the blockchain content among all system nodes. In other words, if one node adds (or commits) a block to the blockchain, the other nodes will also append the same block, and the transaction will be replicated in the blockchain ledger [10]. Proof of Work (PoW) is one of the most notable consensus protocols in blockchain which is employed in Bitcoin and the traditional Ethereum platforms [11].

1.2 Problem

Due to the promising potential of blockchain in different technologies and industries, several blockchain platforms with a variety of consensus algorithms and other properties have been

proposed. Each consensus protocol and blockchain technology comes with strengths and limitations in lights of performance, resource utilization and data integrity.

Due to this variability, the first problem for practitioners is an efficient and easy-to-use benchmark tool than can help them decide the best platform based on their needs. Recently, several research works have been conducted on blockchain technologies and consensus algorithms. These projects [12–15] aim to improve the scalability and resource consumption of traditional blockchain technologies such as Bitcoin and Ethereum with Proof of Work.

Despite the ability of Blockchain to guarantee the immutability of data through consensus, it is also true that this consensus protocol can be a performance and resource bottleneck, which is contraindicated for real-time applications with continuous workloads [16, 17]. High resource consumption can lead to system failures, which by default, will compromise the system's reliability, integrity, and availability. Availability, performance, and integrity, are three critical aspects of real-time applications. Deviations in these three quality attributes may cause many issues for users and applications.

Many different consensus protocols have been proposed for blockchain, including Proof of Work (PoW) [18], Proof of Elapsed Time (PoET) [19], Proof of Stake (PoS) [20] Practical Byzantine Fault Tolerance (PBFT) [21] and Raft [22]. These algorithms have different mechanics and properties, which also include different degrees of scalability, resource utilization, and performance degradation.

To evaluate the aforementioned platforms, benchmarking tools, such as Hyperledger Caliper [23], Blockbench [6], and BCTMark [24] have been proposed. The tools are These benchmarking tools are designed to create *Batch Workloads* to measure the throughput and performance of the blockchains.

Generally, we have two kinds of workloads, *Batch Workloads* and *Online Transnational Workloads*.

A *Batch Workload*, consists of a single, often lengthy executable unit that the entire workload runs until the experiment is completed or if it fails. When we have a batch workload that saturates the system, it affects only this experiment and upon restarting the system, everything is fixed.

Contrary to batch workloads, small work units that are repeated several times while running a benchmark make up a transactional workload. The workload might be constituted of several fluctuating and parallelized transactions to continuously send requests without restarting. Transactional workloads are ideally suited to serve as the foundation for fixed-time benchmarks, this will help us to create a realistic scenario of fluctuating users that are writing the data

Moreover, it allows us to measure the resource utilization in a real-time scenario [25]. In reality, when we normally have a continuously running system, a previously saturated state will carry on to the subsequent normal workload. Meaning that the system may remain saturated for a while even under a reduced normal workload. Existing benchmarking tools are based on the *Batch Workload* which can not entirely simulate a realistic scenario and monitor the resource utilization during the endurance tests.

To evaluate the performance and throughput of different blockchain technologies for real-time applications, we propose *BlockCompass*, an extendable tool that can be used to evaluate the performance, resource utilization, scalability, and efficiency of different blockchain technologies with continuous and fluctuating workloads. BlockCompass has a live monitoring dashboard that can be used to the impact of the current workload on the performance of the target blockchain. The first goal of this thesis aims to present how we can design a more scalable and efficient decentralized application.

Transitioning from a centralized to a decentralized system causes transparency to increase by eliminating any authoritative parties in the transaction. At the same time, integrity is ensured through automatic and peer-to-peer block validation. Consensus is the main characteristic of blockchain, but these validation consensus algorithms in blockchain might increase replication and interconnectivity in the network. Moreover, the problem is that as the number of validating peers (otherwise known as miners [26]) in the network increases, the computational and time overheads are increased, which will have an adverse effect on the system. As a result, the time required to validate and replicate the transactions in the network also increases [27]. There are multiple blockchain technologies with different consensus protocols. We conducted multiple studies to evaluate the difference between private blockchain platforms. We focused on evaluating the costs and performance of public and versus private blockchain technologies, the result of this study guides which kind of blockchain can be suitable for specific kinds of workloads.

Another problem is that, as public blockchain is gaining popularity for decentralized applications, there are tens of novel public blockchain technologies with new consensus protocols in the market. The issue is that the traditional blockchain with Proof of Work (PoW) [28] in Ethereum is expensive and imposes significant time overhead. The new blockchain technologies are designed to overcome these inefficiencies. In this dissertation, we study multiple notable public blockchain technologies to understand and evaluate the differences in costs and performance. Our finding illustrates that no public blockchain platform is the best, and each has weaknesses and strengths. For example, public Avalanche blockchain technology with Snowball consensus protocol [29] is relatively decentralized with thousands of validator nodes

around the workload. However, it imposes high transaction fees on recording transactions on its ledger. On the other hand, the Polygon blockchain with Proof of Stake (PoS) [30] has less than a hundred validator nodes, which means less decentralized and the cost of recording transactions on its ledger is negligible. For this reason, we present *IntelliChain* framework to switch between them dynamically based on transaction fees and throughout.

1.3 Research objectives:

The identified specific objectives of my research are as follows:

OBJ1: Design and implement a benchmarking platform to evaluate resource utilization and performance of various blockchain platforms.

OBJ2: Study and evaluate different public/private blockchain platforms in terms of costs, performance, resource utilization, scalability, integrity, and throughput.

OBJ3: Design and evaluate a self-adaptive framework for private blockchains to dynamically switch the consensus protocol to improve the performance and integrity of the decentralized application.

OBJ4: Design and implement a self-adaptive framework for public blockchains to improve the performance and throughput of decentralized applications.

1.4 Contributions

In this dissertation, I explore five dimensions of blockchain technologies' performance including public vs private, consensus algorithms, DDoS attacks, blockchain vs NoSQL, and comparison of public blockchains. My intention was to investigate how blockchain compares against conventional distributed databases. What is the role of consensus in the performance of blockchain? How is the performance of public and private blockchain different? What is the performance of blockchain under special conditions (e.g., DDoS attacks)?

Moreover, after the knowledge I acquired from these empirical studies, I also realized that studying the performance of blockchain is neither easy nor systematic, so I decided to develop *BlockCompass* benchmarking tool that can help practitioners and researchers to simulate a real-time continuous and fluctuating workloads to evaluate the five aforementioned performance metrics.

Furthermore, after identifying the limitations of blockchain performance, I proposed a series of optimizations and self-adaptive techniques to improve the performance of blockchain for certain applications (NFT [31]) and in specific conditions.

In line with the research objectives stated above, this dissertation presents the following contributions:

1.4.1 BlockCompass: A benchmarking platform for blockchain performance. (Chapter 4)

Existing blockchain benchmarking tools such as Blockbench [5] and Caliper [32] are designed to calculate the response time and throughput of some sample blockchain technologies through the batch workload model [25]. These tools don't measure resource utilization and also require a large amount of time for the users to install and configure their target blockchain technologies. To address these issues, we propose *BlockCompass*, an extendable benchmarking platform to evaluate the performance, and scalability of blockchain technologies through online closed workload model [33]. The closed workload model simulates a fixed number of users in a fixed period of time. The users stay in the system and we can measure from the time that they send the request until the time we receive a response from the receiver node which could be either a success or failure.

BlockCompass simulates virtual concurrent users who want to write data in the blockchain. BlockCompass calculates performance metrics that include latency (the time users send the request and the time they receive the response from the blockchain receiver node/s), and throughput (the number of users that the blockchain can serve). It also measures resource metrics such as CPU utilization, the memory usage of the blockchain validator nodes, and network usage of the validator nodes. BlockCompass provides a live monitoring dashboard that can be used to monitor the performance of the target blockchain in real-time. To understand the usability and quality of BlockCompass, we used the platform to measure the performance of four different popular blockchain technologies and conducted a user survey to get their feedback about the usefulness of the BlockCompass platform.

1.4.2 Empirical studies to evaluate performance metrics of public/private blockchain platforms

Evaluating the performance metrics of the private blockchain platforms (Chapter 5)

In this work, we have simulated four private blockchain platforms, Hyperledger Fabric [34] with PBFT, Hyperledger Sawtooth [35] with PoET, BigchainDB [36], and Hyperledger Burrow [37], to measure the latency, memory utilization, and CPU consumption. As baseline performance, we used MongoDB which does not have any consensus algorithm. Further, we also studied the overheads around the use of blockchain and to study whether there is a single optimal solution with respect to response time and computation overheads. To fairly compare the performance of private blockchains with databases, we measured latency in terms of the response from one blockchain receiver, not the time it takes to actually verify a transaction. Of the four private blockchain platforms, the Hyperledger Sawtooth with PoET consensus protocol was found to have better performance than others. Investigating this, we found that the Hyperledger Sawtooth with PoET performs better because it processes the transactions in parallel using validators, has less transaction processing complexity, and uses the Proof of Elapsed time consensus algorithm.

Comparing the performance metrics of the public blockchain with private blockchain platforms. (Chapter 6)

In this work, we compared the performance of the public Ethereum PoW with private blockchains, Hyperledger Fabric, and Ethereum PoA in terms of costs (transaction fee of public blockchain), CPU utilization, memory usage, network bandwidth, latency, and throughput. To compare their performance metrics and resource utilization, we implemented all Ethereum PoW, Ethereum PoA, and Hyperledger Fabric on private premises and simulated real-time workload. Because we don't have access to the resource utilization of public blockchain, we conducted this study on private infrastructure.

Our study showed that public blockchain can handle large transactions with low arrival rates more efficiently, while private blockchain is performant and cost-efficient for larger workloads of small-sized transactions. Moreover, we have shown that Ethereum PoW is not dependent on network usage whereas Hyperledger with Raft consensus relies heavily on network usage for communication among peers to reach a consensus.

An empirical evaluation of the cost and performance of public blockchain platforms. (Chapter 7)

In this work, we compared the performance of three blockchain technologies, Avalanche [29], Fantom [38], and Polygon [39] networks in terms of the transaction fee, throughput, and error rate (number of rejected transactions due to improper transaction fee). In this case study, we observed that there is no single best public blockchain technology. For example, the Avalanche blockchain has almost 1500 validators which means that it is more decentralized and secure however, the Polygon network with almost 80 validator nodes is the cheapest with the highest throughput. This study also has shown the important features that impact transaction fees and errors. For example, the transaction fee for each public blockchain varies based on the time of the day, pending transactions in the queue, block size, and resource utilization.

1.4.3 Self-adaptive frameworks for private blockchains

Self-adaptive framework to improve integrity in private blockchains (Chapter 8)

In this work, we developed a self-adaptive framework to temporarily mitigate the impact of DDoS (Distributed Denial-of-Service) attacks for private blockchains in the case study of Hyperledger Sawtooth.

To evaluate our framework, we implemented the Hyperledger Sawtooth platform with three built-in consensus protocols including Raft, PoET, and PBFT on the cloud. Then we simulated DDoS attacks to evaluate the integrity and performance of the system and compare the three different consensus algorithms. In this study, we first explore the performance of each consensus protocol under different types of DDoS attacks. We observed that each consensus protocol behaves differently and also depending on the number of validators or resource utilization, the consensus algorithm could have different strengths and weaknesses in terms of security and performance. Therefore, there is no single best consensus protocol. As the Hyperledger Sawtooth consists of three built-in consensus algorithms, the transactions will not be lost if we switch the consensus protocol. The evaluation results demonstrate that our self-adaptive framework reduces the time overheads and the packet loss.

Self-adaptive framework to improve performance in private blockchains (Chapter 9)

In this work, we designed a self-adaptive framework to improve the response time in private blockchain technology. Our self-adaptive framework dynamically switches to the consensus

protocols based on the number of nodes and CPU utilization. As a case study to evaluate our framework, we evaluated different built-in consensus protocols in Hyperledger Sawtooth that include Raft, PoET, and PBFT. We observed that each consensus algorithm behaves differently based on the number of validator nodes. Our results demonstrate that our self-adaptive mechanism has better CPU utilization comparably to PBFT and RAFT, and average Response Time is also better than both RAFT and PBFT. In theory, our adaptive consensus protocol addresses the security limitations of PoET for small networks while maintaining the same performance standards for larger networks.

1.4.4 IntelliChain: A self-adaptive framework for public blockchains to improve performance (Chapter 10)

In this work, we developed *IntelliChain*, a self-adaptive framework for public blockchains. IntelliChain consists of two main features. First, IntelliChain minimizes errors and transaction retires and increases throughput by accurately predicting the required transaction fee for various public blockchains.

Second, it allows to automatically switch between public blockchain platforms depending on the cost of recording transactions and the number of errors based on pending transactions and block size in blockchain gas stations.

IntelliChain currently supports three public platforms, including Avalanche, Polygon, and Fantom. In IntelliChain we used different machine learning and neural network models to predict the blockchain transaction fees and errors. Based on the dataset, we picked the fittest model. Our empirical results have shown that IntelliChain is faster by two hours and has 7 times fewer errors in comparison with the original configurations.

1.5 Outline

The Chapters are organized as follows. Chapter 2 outlines the related literature and provides the background of the concepts and technologies used in my dissertation. Chapter 3 outlines the process of generating research goals, discussing the problems, actionable items, specific milestones, and eventual outcomes in research papers. It illustrates an overall view of the body of this dissertation. This Chapter is followed by seven research papers presented in Chapters 4, 5, 6, 7, 8, 9, and 10.

In Chapter 4, we first present our benchmarking tool: *BlockCompass*. Then we explain how the tool can evaluate the performance, throughput, and scalability of different blockchain networks. We also show how the tool can be made to work with more blockchain platforms.

Finally, we show the results of our survey to figure out how useful our tool is.

Chapters 5, 6, and 7 present three research papers on the performance, costs, and scalability evaluation of public and private blockchain networks. First, we evaluate multiple private blockchain platforms with a NoSQL distributed database in terms of throughput and resource utilization.

In the second research paper, we discuss our research for comparing the costs and throughput in public and private blockchain technologies. Our results can help practitioners decide which blockchain technology could be suitable for the specific type of workloads and dApps.

In the third research paper, we present the evaluation of performance, throughput, and costs of three different public blockchain platforms for large-scale users in the context of the NFT marketplace using our designed benchmarking tool. We exploit machine learning techniques to illustrate the important features that contribute to gas price and throughput.

Chapters 8, and 9 present two research papers with respect to leveraging the self-adaptive systems to improve the performance and integrity of decentralized applications with continuous and fluctuating workloads. In these Chapters, we intend to leverage self-adaptive systems to improve the quality, performance, and integrity of decentralized applications in the blockchain. We first conduct an empirical study on the integrity of blockchain consensus protocols and present our self-adaptive framework to temporarily mitigate the impact of DDoS attacks until a more drastic solution is prepared. In the second research paper, we evaluate our proposed self-adaptive framework to dynamically change the consensus protocol based on the number of nodes and CPU utilization. We have shown that our self-adaptive mechanism has better CPU utilization than PBFT and RAFT, and the average Response Time is also better than both RAFT and PBFT. In theory, our adaptive consensus protocol addresses the security limitations of PoET for small networks while maintaining the same performance standards for more extensive networks.

Chapter 10 presents our framework called *IntelliChain* which aims to provide scalable infrastructure using event-driven architecture and a model for predicting an appropriate gas fee by exploiting neural networks and machine learning models. We found that IntelliChain can improve the throughput of the decentralized application remarkably.

1.6 Publications

The Chapters outlined above are based on the published/submitted research papers listed below.

1. Rasolroveicy, Mohammadreza. "A Self-Adaptive Blockchain Framework to Balance Performance, Security, and Energy Consumption in IoT applications." 2020 IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion (ACSOS-C). IEEE, 2020

2. Rasolroveicy, Mohammadreza, Wejdene Haouari, and Marios Fokaefs. "BlockCompass: A benchmarking platform for blockchain performance" submitted to IEEE Transactions on Knowledge and Data Engineering

3. Rasolroveicy, Mohammadreza, and Marios Fokaefs. "Performance evaluation of distributed ledger technologies for iot data registry: A comparative study." 2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4). IEEE, 2020.

4. Rasolroveicy, Mohammadreza, Wejdene Haouari, and Marios Fokaefs. "Public or private? a techno-economic analysis of blockchain." Proceedings of the 31st Annual International Conference on Computer Science and Software Engineering. 2021.

5. Rasolroveicy, Mohammadreza, and Marios Fokaefs. "Impact of DDoS Attacks on the Performance of Blockchain Consensus as an IoT Data Registry: An Empirical Study" Proceedings of the 32nd Annual International Conference on Computer Science and Software Engineering. 2022.

6. Rasolroveicy, Mohammadreza, and Marios Fokaefs. "Dynamic reconfiguration of consensus protocol for IoT data registry on blockchain." Proceedings of the 30th Annual International Conference on Computer Science and Software Engineering. 2020.

7. Rasolroveicy, Mohammadreza, and Marios Fokaefs. "Performance and Cost Evaluation of Public Blockchain: An NFT Marketplace Case Study." 2022 4th Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS). IEEE, 2022.

8. Rasolroveicy, Mohammadreza, and Marios Fokaefs. IntelliChain: An Intelligent and Scalable Framework for Decentralized Applications on Public Blockchain Technologies: An NFT Marketplace Case Study submitted to IEEE Transactions on Emerging Topics in Computing

CHAPTER 2 LITERATURE REVIEW

Blockchain technology has gained popularity among researchers and stakeholders because of its consistency, high availability, transparency, and immutability [1]. The technology has begun to benefit various fields, including, but not limited to, decentralized finance (DeFi), retail, Internet of Things (IoT), and art [2, 3]. However, blockchain technology imposes significant overhead, delays and costs which are undesirable for real-time applications. Due to the fact that performance problems can have a wide range of underlying causes, identifying and troubleshooting blockchain performance degradation issues can be challenging and time-consuming.

In this chapter, we first review the existing works on performance analysis on the blockchain and the results that have been obtained to improve the performance and efficiency of the blockchain. We review the literature for existing tools for benchmarking the blockchain, and finally, we present some standard definitions and terminology in the world of blockchain technology.

2.1 Background: blockchain architecture and terminology

The goal of this section is to explain some popular definitions and terminology used in the area of blockchain technology. This part is critical to comprehend the remainder of the dissertation.

In essence, blockchain is a peer-to-peer (P2P) and decentralized ledger technology. It is the fundamental technology behind virtual currencies like Bitcoin [40] and Ethereum [41]. Crypto buyers and vendors may record their transactions, and it can ensure that these transactions are authentic, immutable, and transparent. All the data in the blockchain ledger should be verifiable by participating nodes. Generally, blockchain may be classified into two groups based on distinct application scenarios and user needs: public blockchain and private blockchain [4].

Public blockchain: The most decentralized blockchain is the one that is accessible to all and the public. These public blockchain technologies, like those used by Ethereum, are not governed by centralized authorities. Everyone has access to the chain's data records, may engage in transactions, and can compete to create new blocks. Users can interact with the program creators as they choose, and any participating node is free to enter and leave the network and carry out certain tasks [4].

Private blockchain: The private blockchain operates quite differently. An organization or

centralized authority has complete control to grant authorization to the network participants, and it also controls who has access to the data. It may be seen as a system with a weak central authority. Because there are minimal and severe constraints on the participating nodes. In comparison to public blockchain technologies, private blockchain technologies take far less time to achieve consensus, have quicker transactions, are more efficient, and are less expensive [4]. This particular blockchain is better suited for internal usage by certain organizations, such as the integrating Hyperledger Fabric blockchain with Walmart's supply chain [42].

2.1.1 Consensus Algorithm in blockchain

The philosophy of the consensus algorithm is inherited from the notion of the Byzantine Generals (BG) Problem among untrustworthy nodes proposed by Lamport et al. [43]. The problem arose from the early days of blockchain when it was used to mine and trade Bitcoins [44]. In this context, users had two needs; first, avoid the reuse of the same currency in two transactions simultaneously (double spending) and, second, verify the transactions by multiple distributed nodes through a P2P protocol. However, some nodes could be malicious and try to alter communication contents. Moreover, In the blockchain network, by using the consensus algorithms, peers in the system can distinguish between trusted and malicious nodes [45].

In the blockchain network, all the peers in the distributed ledgers have to agree on acceptance by the majority to verify a node. This mechanism guarantees that the output results of the eventual agreement on the state change and all the data of the networks have been shared across multiple replicas. This mechanism has the three following main properties:

- Consistency: The consensus protocol ensures consistency of all the peers to produce the same output and the output produced by the peers is valid and follows the protocol rule. This can also be considered as the consistency of the shared state.
- Availability: The consensus protocol ensures the system's availability even in the presence of faulty nodes.
- Fault Tolerance: The consensus protocol provides fault tolerance by recovering a node if it has failed.

Various algorithms were proposed to cope with several issues in the peer to peer (P2P) networks such as failure of nodes, partitioning of the networks, message delays, corrupted messages, and dealing with selfish and deliberately malicious nodes [46]. Proof of Work

(PoW) was first introduced in the Bitcoin blockchain [47]. PoW is a computational challenge or mathematical puzzle-solving approach to validate the transactions in the blockchain network [48]. Ethereum [49], similar to Bitcoin [44] is based on PoW, and the time to accept a block among the peers is 14s. The PoW algorithm chooses at each round a random peer in the network, which can append a block, where the probability of being selected is based on the peer's computational power. This scenario is to avoid Sybil attacks [50] which commonly happen in decentralized and distributed environments in which the adversary can obtain multiple identities. However, the problem with this strategy is that it consumes a significant amount of energy and computing power, as the network peers spend their CPU cycles solving mathematical puzzles instead of doing otherwise helping work. This technique cannot ensure the complete safety of decentralized networks since it is possible for two peers to be chosen at the same time to be allowed into the blockchain, and both of the blocks that are submitted may be accepted. This will cause forks [51], in the blockchain network, and to avoid this issue, Ethereum adopts a strategy called Greedy Heaviest-Observed Sub-Tree (Ghost) [52] which means that only blocks on the longest chain are considered to be accepted [5].

Because of the complexity of PoW and its strategy which is based on mathematical puzzle solving, it imposes very high latency, and it increases resource consumption, which is not desirable for real-time and time-sensitive applications [53]. To overcome high computational and low response time in PoW, researchers have eliminated this consensus algorithm [16]. Since the consensus protocol in the blockchain is considered its main characteristic, hence, eliminating it, the system would be vulnerable to other attacks, such as double-spending and forgery attacks [54]. To overcome some of these shortcomings of PoW, multiple consensus protocols have been proposed. Next, we describe some of these alternative consensus algorithms.

2.1.2 Proof of Stake

Proof of Stake (PoS) was presented as a solution to the enormous energy consumption caused by PoW. PoS does this by appending to the blockchain system via a pseudo-random selection of stakeholders. In Proof-of-Stake (PoS), a random network member will choose each blockchain branch. A cryptographic signature verification system ensures that only the person who owns the specified transaction can add to the block [55].

2.1.3 Proof of Elapsed time

Proof of Elapsed Time (PoET) is the primary consensus protocol that Intel introduced in Hyperledger Sawtooth. The algorithm randomly chooses the next leader to finalize the block. This consensus protocol employs this method to deal with malicious peers in an open-ended

blockchain network. PoET utilizes the Trusted Execution Environment (TEEm) which is called an Enclave inside of Intel processors to prevent cheating and to provide confidentiality, integrity, and blacklisting malicious behaviors based on asymmetric key cryptography and an additional set of election policies [48, 56].

2.1.4 Tendermint Byzantine Fault Tolerance

Tendermint is a variation of the Byzantine Fault Tolerance (BFT) algorithm, which provides state machine replication using hash-linked branches of transactions in the blockchain network. In this consensus algorithm, each block is committed by a known set of weighted validators. Membership and weighting of this validator set might change over time. This consensus algorithm guarantees the safety and liveness of the blockchain network so long as less than 1/3 of the total weight of the validator set is malicious or faulty [57].

2.1.5 Raft

The Raft [22] algorithm can only be used in private blockchain technologies, where an administrator can add or remove nodes. This algorithm is proposed to solve the inefficiencies of PBFT. It is a lead-based algorithm which means that the peers of the network will choose the leader in an election process. This means that only the leader node is allowed to publish blocks that are validated by other peers. This algorithm cannot tolerate malicious nodes but it can tolerate up to 50% crash faulty nodes. Since in a private blockchain, all the peers are verified by the administrator, this algorithm aims at resolving crash faults other than Byzantine faults [58].

2.1.6 Practical Byzantine Fault Tolerance

The PBFT consensus algorithm is based on the Byzantine Fault Tolerance principles, which means that this algorithm can work correctly as long as 2/3 of the nodes of the network are honest, if the malicious nodes exceed more than 1/3, this consensus stops working work. [59].

2.1.7 Hyperledger

Hyperledger is an ecosystem and an umbrella project consisting of various open-source blockchain platforms hosted by the Linux Foundation [60]. It is a collaborative project to host applications such as financing, banking, supply chains. It ensures transparency, immutability, longevity, and interoperability [61].

There are different Hyperledger platforms, such as Burrow [62], Fabric [13], BESU [63], Indy [64], Iroha [65] and Sawtooth [56] and each of those platforms could be useful for specific applications such as financing or supply chains. Next, we describe some of them that will be used in our study.

2.1.8 Hyperledger Fabric

Hyperledger Fabric is an open-source blockchain platform proposed by IBM. The purpose of this platform is to overcome some limitations of other blockchain platforms, such as Ethereum [49] and Tendermint [66]. In particular, order-executive or hard-coded consensus are two shortcomings that affect the throughput and latency in other blockchain platforms [13]. The consensus algorithm in Hyperledger Fabric relies on Crash Fault Tolerant (CFT) which is based on Raft protocol [67, 68].

2.1.9 Hyperledger Burrow

Hyperledger Burrow is another project which is Hosted by the Linux Foundation and was introduced by Monax [69] and Intel [70]. It provides a modular blockchain client with a Permissioned Smart Contract Interpreter (PSCI). It consists of three main features including Ethereum Virtual Machine, the consensus engine, and a remote procedure called gateway. The main difference between Burrow and other platforms, like Fabric, is that its main focus is on running on Ethereum Virtual Machine(EVM) and on using smart contracts in a permissioned network. It uses Proof of Stake (POS) consensus in which participants in the network can validate transactions based on the held coins or power. PoS is used to overcome the computational overhead in Proof of Work. Burrow provides a tool to implement smart contracts written in Solidity [71] which can formulate all the transactions on the chain network [72].

2.1.10 Hyperledger Sawtooth

Hyperledger Sawtooth [56] is an open-source blockchain platform that is designed especially for supplying chain management. It is part of the Linux Foundation umbrella project developed by Intel. The main difference between Sawtooth and other blockchain platforms is that the transactions and data can be executed in parallel instead of in series, which will result in better performance of the system. Sawtooth supports several consensus algorithms, including Practical Fault Tolerance (PBFT) [67], Raft [22], and PoET [19].

2.1.11 BigchainDB

BigchainDB is a decentralized distributed database with a high-performance throughput [36], it is designed to advocate blockchain characteristics and inherit NoSQL distributed database properties, such as linear scaling throughput and efficient querying. As described in its characteristics, it is developed to solve some limitations of blockchain, such as low latency, high computational power, and low scalability. The consensus algorithm in BigchainDB is Tendermint which is derived from BFT. BigchainDB uses several security measures: (a) Benign Fault [73] means that if there are $2f + 1$ nodes, f malicious or failed nodes could be tolerated; (b) Byzantine Fault Tolerance means that BigchainDB can operate in a trust-less network, which includes measures against malicious or faulty nodes in the network; it uses a voting mechanism to validate transactions. And finally, the last feature of BigchainDB is its implementation to protect against Sybil attacks in a federation with a high barrier of entry based on trust and reputation to avoid participants attacking the clones [74, 75]. BigchainDB is not part of the Hyperledger ecosystem; however, the reason for choosing this platform is that it is currently very popular as a lightweight blockchain network [76].

2.1.12 dApp

In contrast to conventional apps hosted by centralised servers, decentralised applications (dApp) host some of their back-end functions and databases on peer-to-peer networks [77]. A dApp has the following properties:

-Open source: dApp's source codes are available to the public so that outside audits are feasible. However, in practice, more dApps often just disclose their application binary interface to ensure security and demonstrate that the source code has not been altered [77, 78].

- Support for internal cryptocurrencies: The engine that powers a given dApp's ecosystem is its internal currency. A dApp can quantify all credits and transactions among system users, including content creators and consumers, using tokens [77, 78].

– No single point of failure: Since all application components will be stored and run on the blockchain, a completely decentralised system shouldn't have a single point of failure [77, 78].

A non-Fungible Token is a type of decentralized application that incorporates the use of blockchain-decentralized storage for recording digital information.

2.1.13 Smart-contracts

Smart contracts are a piece of code stored in a blockchain ledger that executes when certain criteria are met. They are often used to automate the implementation of an agreement so that all parties may be confident of the outcome right away without the need for an intermediary or additional delay. They may also automate a process such that when circumstances are fulfilled, the following action is executed. These actions can include transferring funds to certain beneficiaries, registering a car, sending out warnings, or issuing a ticket. When the transaction is completed, the blockchain is then updated. Therefore, the transaction cannot be modified, and only those to whom permission has been given may see the output [79]. For example, we can design a smart contract to store digital asset metadata in blockchain and we can transfer the ownership of the digital asset to another person. For example, we can design a smart contract to store digital asset metadata in blockchain and transfer the digital asset's ownership to another person. This approach can be used as proof of ownership of the digital asset, and we can trace the owners of the particular digital asset over time.

2.1.14 NFT

In 2017, non-fungible tokens (NFTs) were proposed as financial security tokens consisting of digital data recorded in a blockchain. An NFT typically contains metadata, including names, descriptions, and image/video/music URLs in decentralized storage such as IPFS [80]. NFT tokens are transparent, immutable, and time-stamped, and their ownership can be transferred by the owner, which allows them to be traded or sold [31]. Due to the emergence of NFTs, Ethereum, a public permission-less blockchain platform, has gained momentum as artists and digital content creators have started to store their digital assets on an immutable Ethereum ledger and demonstrate the proof of the ownership and authenticity of their products to their clients. Opposite to cryptocurrencies, NFTs are non-interchangeable, identical and indivisible [31, 81]. NFTs have identical identification codes and metadata, which can be distinguished from each other [81, 82].

Minting [83] is a special terminology used in the NFT sector. It entails recording the distinctive data of a digital asset, such as the original creator, the moment of creation, and the reference to the digital asset metadata on the blockchain. The NFT may be exchanged between owners once it has been minted. The ownership history is transparent on the blockchain and cannot be altered or removed. [84]. The digital assets or files will be kept in a decentralized ledger and cannot be changed, altered, or removed [31, 84, 85].

2.2 Studies on the performance of blockchain technologies

Ampel et al. [15] used Hyperledger Caliper to study the performance of Hyperledger Sawtooth with the PoET consensus algorithm by using various parameters to test transaction send rate, batch size, throughput, CPU consumption, memory usage, and latency. In their experiments, they observed that throughput reached a maximum of about 2300 tx/sec, which is much higher than any studied permission-less blockchain and also ahead of the throughput found for Hyperledger Fabric. They also observed that if the batch size is set much higher than the send rate, transactions will fail regardless of the total number of transactions. When the batch size and input transaction rate were taken into account, the current Sawtooth platform with the PoET algorithm could not reach a throughput of more than 2300 tx/sec.

Nasir et al. [86] conducted a comparative performance analysis of two versions of Hyperledger Fabric, v0.6 and v1.0 by using Hyperledger Caliper. The performance of the two platforms was studied in terms of execution time, latency, and throughput by changing the workload in each platform up to 10,000 transactions at a time. At the time of these experiments, the Hyperledger Fabric used the PBFT algorithm, and Ethereum used the PoW algorithm. The authors observed that through batch benchmarking, Fabric v1.0 maintained constant performance regardless of the number of peers, while Fabric v0.6 fluctuated continuously. They also observed that the maximum number of nodes that Fabric v1.0 can have is 26, whereas for v0.6, it is 16 nodes. Moreover, the Fabric can handle up to 10,000 concurrent transactions with six nodes.

Hao et al. [87] have conducted an empirical study to analyze the performance of consensus algorithms as a core of blockchain systems. This solves the problem of mutual trust between nodes in distributed systems. At the time of these experiments, the Hyperledger Fabric, which was using PBFT algorithm and Ethereum was using PoW algorithm. Their results have shown that PBFT outperforms PoW in terms of average latency and transactions per second. They have also identified that the consensus algorithm is the main performance bottleneck for both Ethereum and Hyperledger Fabric. In the future, it is better to demonstrate the resource consumption and identify which layer in consensus cause low performance and latency.

Pongnumkul et al. [88] presented a performance analysis of Ethereum and Hyperledger Fabric as private blockchain platforms with a diversified number of transactions. The writers have shown that Hyperledger Fabric attains higher throughput and lower reaction time compared to Ethereum when the workload increases by 10,000 minutes. Also, the differences between Hyperledger Fabric platforms regarding execution time and average latency become more important as the number of transactions increases. The average throughput of Hyperledger

Fabric also changes at a much faster pace than that of Ethereum. Still, they haven't shown how these two programs use resources and energy when used in distributed real-time applications.

Kuzlu et al. [89] have evaluated the impact of network workload on the performance of blockchain platforms by adopting Hyperledger Fabric v.1.4. The evaluation has been instituted in terms of Latency, Throughput, and Scalability. Since several blockchain projects are being conducted, there exist some concerns about the technical challenges of a blockchain platform in terms of its throughput, latency, and its ability to scale. To evaluate the platform's performance, the authors utilized an instance of a Hyperledger Fabric platform. The outcomes of their work have expressed that the instance of the Hyperledger Fabric platform implemented can support up to 100,000 participants on the selected AWS EC2 instance. The authors think that a blockchain network's throughput, latency, and ability to grow to depend on how the hardware is set up, how the blockchain network is designed, and how smart contact operations are done.

Saraf et al. [90] presented a study on different Hyperledger umbrella projects, including Burrow, Fabric, Indy, Iroha, and Sawtooth. They presented a compendium of their usability and functionality. They have shown that Hyperledger could be a good solution when security and privacy must be maintained. However, they have not empirically compared the difference between these platforms in terms of computation and time overheads to see which one performs better in real-time applications.

Dorri et al. [16] explored and outlined thoroughly the various core components and functions of the smart home domain. In their model, each smart home is equipped with an always-online, high-resource device, known as a miner that is responsible for carrying out all communications inside and outside the home. Also, the miner has to maintain a private and secure blockchain, used for controlling and auditing communications. They proposed a novel method to employ blockchain by eliminating the Proof Of Work concept. Their simulation results indicate that the obtained overheads are low and can be managed for low-resource IoT devices. Nevertheless, their model has a significant shortcoming. Eliminating PoW is the blockchain's key characteristic, making the system vulnerable to notable attacks such as double-spending and forgery [54]. To evaluate their model, they simulated a smart home scenario in the Cooja Simulator [91], which is used to simulate Wireless Sensor Networks (WSN).

Aung et al. [53] reviewed different possible approaches that can be applied to Ethereum private blockchain for Smart Home System (SHS). In their paper, they have considered SHS as a case study that integrates home appliances together with sensors to get automatic operations of heating, lighting, air conditioning, home security, health care systems. The authors presented an approach to private blockchain implementation for SHS to cope with its privacy and

security issues. The Ethereum blockchain packages for SHS according to its smart contract features for carrying out access control policy, data storage, and data flow management, has been reviewed. The architecture of their proposed system differs from the work of Dorri et al. [16]. Their architecture consists of a smart home miner (SH miner), private blockchain, and local storage connects to SH sensors and actuator devices. The function of the private blockchain is to store policies for data flow or transaction management. However, they believe that because of the low transaction time of the Ethereum blockchain, it may be difficult for time-sensitive conditions.

The authors analyzed security and privacy based on their testbed devices. A smart home system based on Raspberry Pi acts as an IoT edge device, and privacy information is concentrated on user information. The main constraint of is study is that they are using public Ethereum which is an expensive solution in terms of energy consumption and delays for IoT applications. For future works, we recommend considering an empirical evaluation the performance and scalability of their proposed model.

Alrubei et al. [92] have provided a practical implementation of Proof of Authority (PoA) which is a variation of the Byzantine Fault Tolerance for Ethereum blockchain on an IoT system in a real-world use case. This implementation was performed to examine and highlight possible problems affecting integration of blockchain with IoT, laying the ground for future research and possible solutions to these issues. The results show that by increasing the block period of a PoA Ethereum Network, the stability of the network also increases. In their model, authorities are supposed to be honest nodes (at least N/2 +1) of them. Introducing this new method is to save more energy by replacing PoW.

Novo. [93] has studied the delays and the throughput rate for blockchain systems associated with them and analyzed different configurations of their solution to maximize blockchain scalability. The paper aims to evaluate and compare the centralized architecture (proof of concept) with the comprehensive management solutions for IoT. The evaluation depicts that the author's implementation has significant scalable advantages over the traditional scenarios when they distribute the load among the blockchain network's nodes.In the context of performance comparison, they have shown that when the size of the WSN changes from 1 to 10,000 concurrent clients, the performance remains steady, with over 790 requests per second. The evaluation in this paper proves the effectiveness of the design in some particular IoT scenarios. The performance is steady. However, they have shown that their system can achieve lower latency values for higher concurrency factors (up to 1,000). Accordingly, based on the findings, the proposed solution does not achieve better performance than the optimized centralized IoT. The authors have limited their prototype experiments to WSNs and

Management Hubs, as the rest of the smart contract operations are transaction operations, and the miners need to validate them.

Wang et al. [9] presented a systematic and comprehensive overview of two blockchain-enabled smart contracts, Ethereum and Hyperledger Fabric in the area. They aimed to stimulate further research toward this emerging research area. By introducing the operating mechanism and mainstream platforms of blockchain-enabled smart contracts, the authors proposed a research framework for smart contracts based on novel six-layer architecture. The importance of smart contracts is that it is exploited to automate claims processing, verification, and payment, thus increasing the speed of claim processing and to ban fraud and avoid potential piteous. The authors believe that the traditional centralized Internet system is difficult to fulfill IoT's development needs, such as the security of sensitive information and trusted interaction between multi-devices. Therefore, the synthesis of IoT and blockchain becomes an inevitable tendency, and smart contracts will aid in automating the intricate work flow, elevate resource sharing, save costs, and ensure safety and efficiency. The authors have shown that Hyperledger Fabric attains high degrees of scalability, resilience, and confidentiality as it provides a modular architecture with a delineation of roles between the nodes (e.g., endorsers and orderers) and configurable consensus and membership services. Moreover, in Hyperledger Fabric, there is no built-in cryptocurrency or fuel (such as Ether and gas in Ethereum). The functional limitation is that any activity in the blockchain needs to be generated by a node controlled from outside of the network. The smart contract should implement autonomous mechanisms or complex microservice interactions, which, in totality, realize more sophisticated service logic like autonomous portfolio management service. As the authors suggest, future smart contracts should have a certain autonomy and intelligence.

In summary, these studies show that performance analysis of blockchain can be a complex task. On the one hand, this is because most platforms have a large number of parameters to be configured that can affect performance, including consensus algorithm, transaction rate, batch size, number of nodes, and others. On the other hand, we must evaluate several performance dimensions to provide complete and holistic conclusions, including throughput, packet loss, latency, resource consumption (CPU, memory, network), and others. To achieve meaningful and reliable results and reproducible studies, these concerns necessitate extensive and systematic configurations and instrumented monitoring. In addition, we have found that most of these studies are limited by the tools they use. Most notably, the tools employed by these studies use batch workloads, meaning that they are capable of stress testing the platform and identifying their capacity. However, in longer-running workloads, backlogs may appear, which will make the platform's performance degrade and fail even if there is no high stress at a given moment. For these reasons, in this thesis, we conducted multiple evaluation studies on

the performance of private and public blockchain technologies for real-time applications. We created a real-time workload for our studies which can generate fluctuating and continuous requests. The results of our studies can help practitioners and academia to decide which blockchain technology is suitable for their workload.

2.3 Benchmarking tools for blockchain

Dinh et al. [6] introduced Blockbench, a cross-sectional benchmark architecture, and benchmarking tool. This software can assess the latency, throughput, scalability, and fault tolerance of blockchain technology. The authors have designed drivers for three blockchain platforms including Ethereum, Hyperledger Fabric, and Parity. To understand these systems' performance and security and find bottlenecks, they evaluated them using a variety of batch workloads. The primary shortcomings of this tool are only supporting batch workloads and does not provide resource usage monitoring (CPU, memory, network). Even though the tool is open-source, there is little documentation for extendability, making it difficult to adapt it to new blockchain systems. Some blockchain drivers are out-of-date, and they can't handle large-scale experimentation in some systems (such as 16 nodes in Fabric).

"GoHammer" is created by Birim et al. [94] to assess the latency, throughput, resource consumption, and declined requests on the Quorum blockchain network. The tool and Quorum Profiling is comparable, but the latter only currently supports the Raft consensus method, while the former makes it simpler to deploy the workload and the network. This benchmarking tool is cross-sectional in nature. The software still requires a variety of enhancements, including the ability to monitor a single blockchain node, new workload situations, and other blockchain platforms. Since the tool is open-source, these additions might come from other academics or industry professionals.

Shapiro et al. [95] rearchitected and redesigned Hyperledger Caliper to include 1) multiple client nodes for blockchain platforms, 2) offloading of the cryptographic signature, and 3) removing the unnecessary overheads. Using their extension, they evaluated Hyperledger Burrow, Quorum, and Red Belly blockchain. Results identified the maximum send rate that Burrow and Quorum can process and show that the Red Belly blockchain can offer 8-times higher throughput than the other platforms. The authors observed that a benchmark containing numerous rounds resulted in system crashes more frequently than one-round benchmarks throughout the benchmark execution. They have also observed that once they set the send rate to 2,000 tx/sec, the Hyperledger Burrow begins to fail. These are important findings that support the need for transactional and continuous workloads.

Saingre et al. [24] proposed a benchmark framework for reproducible research on blockchain technology performance (latency, throughput, and energy consumption). The BCTMark (blockchain Technologies Benchmarking) is used for the deployment, comparison, and evolution of different blockchain platforms. The tool does not demonstrate high extensibility and reusability, especially concerning performance metrics.

Baliga et al. [96] have developed an extension for Caliper [23] for the Quorum blockchain [97] to create a batch and various controlled workloads to evaluate the latency and throughput of the platform. Quorum supports smart contracts and transaction confidentiality and privacy, and reliability and Byzantine fault-tolerant consensus algorithms. By default, it provides the Raft and IBFT consensus protocols. Private contracts in Quorum result in lower throughput and a higher load on the system due to extra overhead involved in secure communication and encryption/decryption operations employed between peers for confidentiality. The results indicate that the throughput is comparable for all settings with public contracts until an input transaction rate of 600 tx/sec. Upon further investigation based on their tool, the authors detected a bug in the Quorum code on inappropriate file handling of file descriptors, where the code eventually runs into a scenario where there are not enough file descriptors available.

A review of related literature [9, 24, 94, 96, 98, 99] and our experimentation with existing benchmark tools for blockchain showed that they are often designed and deployed for one or two platforms without a clear or explicit way to add more platforms. More importantly, the existing tools are cross-sectional which means that we need to configure a single batch transaction and we get the results for throughput and latency for the specified configuration.

The main limitation of this kind of workload is that they can not represent a real-time application with continuous and fluctuating users to monitor at which point the systems fail and how much resources each blockchain technology consumes.

To mitigate the limitation of the aforementioned platforms, we propose the *BlockCompass*, an extendable benchmarking platform to create transactional workloads for evaluating the endurance of the blockchain technologies platform under continuous and fluctuating workloads without restarting the system. BlockCompass provides a live monitoring dashboard that allows developers and researchers to configure the workload and evaluate the performance and resource utilization in real-time.

CHAPTER 3 RESEARCH OBJECTIVES, RESEARCH QUESTIONS AND CONTRIBUTIONS

As discussed briefly in Chapter 1, in this dissertation, our ultimate goal is to design a more performant cost efficient decentralized framework for real-time applications with a high number of users. To achieve this goal, we designed a benchmarking platform and conducted multiple analyses of blockchain technologies in terms of integrity, throughput, scalability, and costs. Due to our industrial collaboration through Mitacs Accelerate Program, we used NFTs dAPPs as a case study.

Figure 3.1 Illustrates the connection between different parts of this dissertation.

In Part I, we propose *BlockCompass* a benchmarking platform for the evaluation of public blockchain technologies. In Part II, we conduct multiple studies on the performance and efficiency of private and public technologies. In Part III, we propose self-adaptive frameworks to improve the performance and integrity of decentralized applications in private blockchain technologies. And in Part IV, we propose IntelliChain, an intelligent self-adaptive framework for decentralized applications on public blockchains.

3.1 Benchmarking platform

Part I: Design and implement a benchmarking platform to evaluate resource utilization and throughput of various blockchain platforms. Blockchain technology is introduced to improve the integrity and trust among the parties that don't necessarily trust each other. Consensus is one of the main characteristics of blockchain and as we discussed, it can impose time and computation overheads which are undesirable for large-scale projects with high number of users. There are multiple research on blockchain performance using batch workloads. However, in order to identify the performance limitation of blockchain technologies for real-time applications with fluctuating requests, we first propose *BlockCompass* in Chapter 4, a comprehensive blockchain benchmarking tool that can be easily configured and extended.

In this research project, we demonstrate how BlockCompass can evaluate the performance of a variety of blockchain platforms and configurations, including Ethereum Proof-of-Authority, Ethereum Proof-of-Work, Hyperledger Fabric Raft, Hyperledger Sawtooth with Proof-of-Elapsed-Time, Practical Byzantine Fault Tolerance and Raft consensus algorithms against workloads that continuously fluctuate over time.

Numerous blockchain platforms have been developed, each with a particular set of consensus

algorithms and other characteristics, as a consequence of the blockchain's promising potential in various technologies and businesses. Each consensus mechanism and blockchain technology has advantages and disadvantages in terms of efficiency, effective use of resources, and data integrity. Practitioners and stakeholders require an effective and user-friendly benchmark tool that can assist them in selecting the optimal platform depending on their requirements because of this variety. Numerous studies on blockchain technology and consensus methods have recently been completed. This projects [12–15] aim to improve the scalability, resource consumption, and energy consumption of traditional blockchain technologies such as Bitcoin and Ethereum with Proof of Work.

BlockCompass can help practitioners and stakeholders evaluate and analyze blockchain technologies in real-world scenarios where they are required to run continuously under a fluctuating and increased workload. Lastly, we conducted a **usability study** with 33 participants to evaluate the user's experience with BlockCompass when considering different blockchain platforms compared to other blockchain benchmarking tools and manual effort.

BlockCompass benchmarking platform helped us to design a cost-efficient dApp for our industrial partners and their clients. We believe that this tool can help other practitioners and decision-makers to choose suitable blockchain technology depending on their workload and application.

3.2 Performance evaluation of blockchain

Study and evaluate different public/private blockchain platforms in terms of costs, performance, resource utilization, scalability, integrity, and throughput.

In Chapters 5, 6 and 7, we present three published research papers [100], [101] and [85]. The papers focus on the evaluation of public and private blockchain technologies in terms of scalability, resource consumption, costs, and throughput. To design a decentralized application with a high number of workloads, we need to understand which kind of blockchain platform or consensus protocol could be appropriate for practitioners and industries. There are multiple public and private blockchain platforms in the market and we need to identify which kind of private or public blockchain is suitable for a specific kind of workload.

In the first research paper of this Chapter, we studied the performance of various private blockchain technologies (Hyperledger Fabric, Hyperledger Burrow, Hyperledger Sawtooth, and BigchainDB) against a conventional distributed database. This research project's primary goal is to quantify the overheads brought on by blockchain technology and comprehend how these overheads differ from those of a conventional large-scale database. To do this, we

contrast the four Blockchain installations with a distributed NoSQL installation and put the five systems under the identical continuous workload simulation. This study's secondary goal is to investigate the effects of various variations of the consensus protocol, one of the key Blockchain features, on system performance. This research project helped to identify which kind of private blockchain

Furthermore, in the second research paper, we study and quantify the differences between private and public blockchain solutions in terms of resource utilization, performance and cost and present the impact of concurrent users who tries to continuously write transactions and the number of users fluctuates over time. This research project aims to guide practitioners and industries that which blockchain technologies (public or private) are more suitable for specific kinds of workloads.

Performance varies between private and public blockchain technologies, as do cost management and other security issues. The whole network in the private blockchain is regulated by parties that may have mutual trust. The network peers in a permissioned network control access to the data. The private peer-to-peer network is anticipated to establish agreement very quickly since there are fewer participants, which might enhance performance. However, the network's owners must pay capital expenses to maintain the infrastructure directly. The benefits of private infrastructure include ownership, permanence, and privacy. The objective of this research is to quantify the performance (response time and resource consumption) and network maintenance cost variations between private and public blockchain technologies. Particularly, we have created an experiment to carry out endurance tests on a private Blockchain (Hyperledger Fabric) and a simulated or deployed on private resources, a public Blockchain (Ethereum). We also evaluate other consensus protocols as part of the research, including Proof of Authority (PoA), Proof of Work (PoW), and Raft consensus algorithms. Lastly, we analyze the platforms' costs by attributing resource usage to infrastructure expenses and public Blockchain-specific charges, such as transaction fees and the price of cryptocurrencies.

The result of this study shows that public Blockchain can handle large transactions with low arrival rates more efficiently, while private Blockchain is performant and cost-efficient for larger workloads of small-sized transactions.

Our comparative studies in the past two projects helped us to identify that public blockchain technologies could be suitable for NFTs, our use-case study for our industrial collaborators. Ethereum is a popular and public blockchain technology that is traditionally based on the PoW consensus protocol. PoW imposes high computation and costs and it was not a suitable public blockchain technology to be used. Minting in NFTs at the time of our research in July 2022 was costing us 100 USD and we needed to mint and transfer thousands of

NFTs to collectors. There are multiple public blockchain platforms that are alternative solutions to Ethereum. Some of the popular blockchain technologies as a fork of Ethereum are Avalanche [102], Polygon [39], and Fantom [38]. These systems differ in design, new consensus method, availability, and performance. Most of these platforms are forks of the prominent Ethereum network, with just the consensus mechanism being changed to use more affordable algorithms. In this research article, we simulate an NFT marketplace on these three public blockchain platforms and leverage machine learning models for predicting throughput and cost and highlight the key variables for doing so. The ultimate goal is to help decision-makers design a cost-effective NFT marketplace and identify which parameters are imperative in transaction fees and throughput in the public blockchain.

3.3 Self-adaptive mechanisms in blockchain

Part III: Design and evaluate a self-adaptive framework in blockchain to dynamically change the consensus protocol based on the number of the blockchain validator nodes and CPU utilization.

Chapters 8 and 9 present two research papers concerning self-adaptive systems. Our goal is to use self-adaptive systems to improve the quality and integrity of systems in real-time applications with a high number of changing requests.

In the first research paper [103], our intention is to determine if consensus algorithms can function well when under the pressure of DDoS attacks and continue to uphold the performance and integrity of the blockchain platform. Blockchain technology is proposed as a solution for mitigating DDoS attacks [104–106]. Therefore, we intend to identify what is the impact of DDoS attacks on the performance of different consensus protocols in terms of performance, resource utilization, and integrity. In this research, we study some of the popular consensus protocols in the blockchain. These consensus protocols have different mechanisms and approaches to guarantee the integrity and security of data. Because the loss of data during the attack may seriously impair the integrity and reliability of the blockchain system. Each algorithm has different strengths and weaknesses concerning performance degradation or against attacks thanks to their unique characteristics. In this work, we evaluate the capacity of consensus algorithms to perform under the stress of a DDoS attack and continue to maintain integrity in the blockchain platform.

Under the assumption that different consensus algorithms behave differently under different conditions and under different DDoS attacks, we further hypothesize that a dynamic reconfiguration of the consensus protocol as a response to a DDoS attack can further optimize

blockchain's response against the attack. To this end, we propose a self-adaptive system to allow dynamic configuration of consensus algorithms, based on the MAPE architecture [107]. We believe our self-adaptive approach can be used temporarily to mitigate the impact of the attack until a more drastic solution is prepared

The second research paper [108] of this Chapter 8 presents our proposed self-adaptive mechanism to dynamically change the consensus protocol based on the number of nodes and resource utilization. For our experiments, we chose Hyperledger Sawtooth. It supports four consensus algorithms, including Raft, PoET, and PBFT. We expose the platform to fluctuating and continuous data loads and assessed the performance in terms of CPU and reaction time to assess the performance of each consensus process. It is envisaged that the various algorithms would each have strengths and shortcomings and that these characteristics will appear in multiple ways. As a result, it is also predicted that the stakeholders would wish to choose the algorithm that best suits their needs and professional objectives. Given that these criteria in a real-time application could be dynamic, so should the consensus method choice. We intend to propose a self-adaptive management system in order to implement such dynamic behavior. This system would automatically and dynamically switch between consensus algorithms and deployment configurations based on the needs of the applications and changes in the data load. The MAPE [107] architecture is the foundation of the proposed self-adaptive system. Our results demonstrate that our self-adaptive mechanism has better CPU utilization comparably to PBFT and RAFT, and average Response Time is also better than both RAFT and PBFT.

3.4 IntelliChain

Part IV: Design and implement a self-adaptive framework to improve the throughput and scalability of decentralized applications on public blockchains.

Chapter 10 presents *IntelliChain*, a scalable and intelligent framework to improve the scalability and performance in NFT decentralized applications. In this study, we prototype our proposed NFT marketplace architecture. We discuss our comparative results for evaluating three public blockchain technologies, and then we employ machine learning and neural network models to predict the blockchain transaction fee and error. Then we present a model that can predict the transaction fee with zero error. Lastly, we present our decision-support framework to dynamically switch the public blockchain based on the predicted error and lower transaction fees.

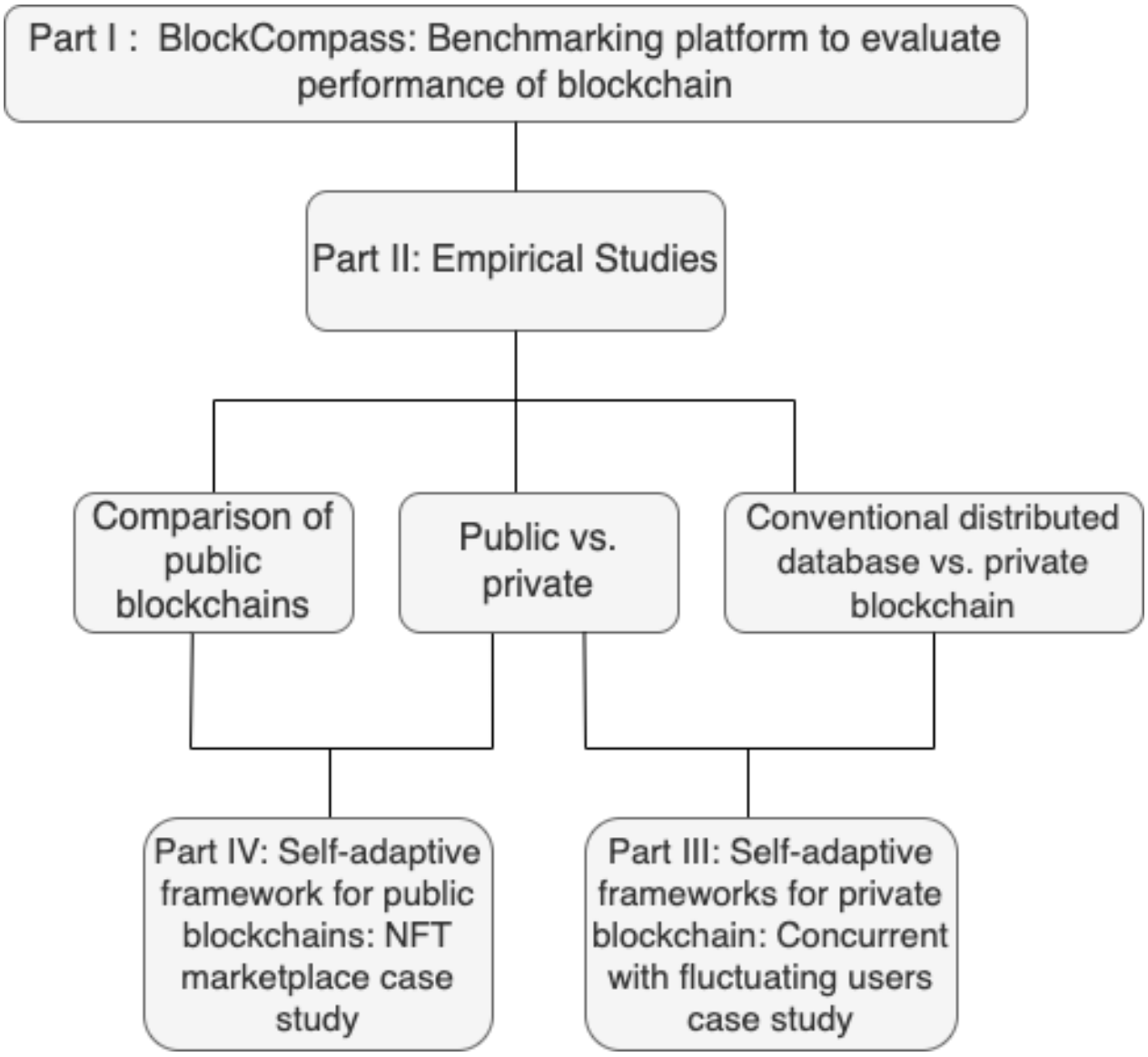


Figure 3.1 A sketch of the connections between different parts of the dissertation

CHAPTER 4 ARTICLE 1: BLOCKCOMPASS: A BENCHMARKING PLATFORM FOR BLOCKCHAIN PERFORMANCE

4.1 Paper Information

Title

BlockCompass: A benchmarking platform for blockchain performance

Authors

Mohammadreza Rasolroveicy, Wejdene Haouari,and Marios Fokaefs

Date of submission: 2022/09/12

Submitted to: IEEE Transactions on Knowledge and Data Engineering

Chapter Overview: This Chapter presents *BlockCompass*, a benchmarking platform for blockchain performance. This extendable benchmarking platform allows researchers and practitioners to automatically set up and configure their intended blockchain technologies and evaluate the resource utilization including CPU, memory, and the total amount of network bandwidth per Byte, and performance metrics including throughput, emit rate, and response time from the intended blockchain receiver node. BlockCompass is configurable, which means that the users of the BlockCompass can change the blockchain, duration of the experiment, size of the workload, type of the users in the workload, size of the data, number of users, and number of blockchain validator nodes.

The main differences between BlockCompass and the existing popular competitor tool, *Blockbench* [5] are the automatic deployment of blockchain technologies through docker containers, live monitoring dashboard, and transactional online closed workload generator. Transactional online workloads help to simulate a real-time application with concurrent and fluctuating users.

The competitor benchmarking tools are mostly designed based on batch workloads. Batch workloads must take their long-running unit of work's setup into account. The starting setup and input data are basically what this involves. Additionally, the explanations may need to address how the batch workload was first placed on the available computer resources [25].

BlockCompass is designed on closed-workload architecture, which means that the system has a fixed number of users for a specific period of time. Non-executable components of a task are described in terms of the activities a virtual user does. Although additional actions are also possible, certain user actions cause a transaction to be executed. The user may

need to wait a particular amount of time before initiating the subsequent transaction, for instance. The user performs the subsequent action after waiting for the previous action to finish. Users never leave the system while executing a potentially limitless series of activities (which are often repeated) until the workload is terminated, hence the workload is termed closed (e.g., after a fixed time). Closed workloads specify a transaction's start time in terms of its characteristics in a way that relies on the end times of certain prior transactions (each user's initial transaction is an exception) [25]. The ability to evaluate blockchain technologies through an online transnational model helps practitioners and researchers to understand which private blockchain technology could be more cost-efficient and performant for real-time applications with continuous and fluctuating workloads.

This Chapter offers a technique for analyzing blockchain performance, which can be applied to the dimensions mentioned in Chapters 5, 6, 7, 8, 9, and 10.

4.2 Abstract

Blockchain technology has gained momentum among researchers and other stakeholders due to its immutability and transparency. Several blockchain platforms with different kinds of consensus protocols have been proposed. However, this makes choosing and configuring such a platform, a non-trivial task. Several benchmarking tools have been presented to test the performance of blockchain solutions. However, these solutions are either limited to specific blockchain platforms or require complex configurations. Moreover, they tend to focus on transactional evaluation models, which may be counter-intuitive for longer-running instances under continuous workloads. In this work, we present *BlockCompass*, an all-inclusive blockchain benchmarking tool that can be easily configured and extended. We demonstrate how BlockCompass can evaluate the performance of a variety of blockchain platforms and configurations, including Ethereum Proof-of-Authority, Ethereum Proof-of-Work, Hyperledger Fabric Raft, Hyperledger Sawtooth with Proof-of-Elapsed-Time, Practical Byzantine Fault Tolerance and Raft consensus algorithms against workloads that continuously fluctuate over time. We also present the results of a usability study about the convenience and facility offered by BlockCompass in blockchain benchmarking.

4.3 Introduction

Blockchain technology is gaining ever-increasing popularity among researchers and practitioners because of its consistency, high availability, transparency, and immutability [1]. The technology has begun to benefit various fields, including, but not limited to, decentralized

finance (DeFi), retail, Internet of Things (IoT), and art [2, 3]. The core concept of blockchain is its consensus algorithm which protects the data against undesired alteration. Blockchain, which is also known as a decentralized ledger technology (DLT), is a set of peer-to-peer nodes which do not trust each other. In the blockchain, in order to have a transaction confirmed, the majority of peers in the network should agree to add it to the block, according to the consensus protocol. Each node keeps replicas of the data and has to validate the transaction into the block [4–6].

Due to the promising potential of blockchain in different technologies and industries, several blockchain platforms with a variety of consensus algorithms and other properties have been proposed. Each consensus protocol and blockchain technology comes with strengths and limitations in terms of performance, resource utilization, and data integrity. Due to this variability, practitioners need an efficient and easy-to-use benchmark tool than can help them decide the best platform based on their needs. Recently, a number of research works have been conducted on blockchain technologies and consensus algorithms. These projects [12–15] aim to improve the scalability, resource consumption, and energy consumption of the traditional blockchain technologies such as Bitcoin and Ethereum with Proof of Work.

To evaluate such platforms, benchmark tools, such as Hyperledger Caliper [1], Blockbench [6] and BCTMark [24] have been proposed. However, one important limitation of these tools is that they are designed to perform "Batch Workloads", which means that the entire workload runs until the experiment is completed or if it fails. This is incompatible with evaluating the system's long-term scalability and endurance under continuous and variable workloads. Such a scenario may stress the platform differently in ways that are not covered by cross-sectional experiments. Our tool is based on "Transactional Open Workloads" [25]: a variable number of virtual users sending small work units repeated over the execution of an experiment. This type of workload achieves a realistic variance in the platform's incoming traffic and long-running experiment that tests the platform's endurance.

The main contributions of this work are as follows.

- **BlockCompass**: a benchmarking tool to evaluate and analyze the blockchain platforms in real-world scenarios where they are required to run continuously under a fluctuating and increased workload.
- A set of extendable and configurable blockchain **drivers** for Ethereum, Hyperledger Fabric, and Hyperledger Sawtooth to serve the user's requests and submit them to the blockchain.

[1] https://hyperledger.github.io/caliper/

- A real-time and online **monitoring dashboard** to observe different metrics: Resource Consumption (CPU consumption, memory usage, network I/O), Performance metrics (Response Time, Throughput, Error Rate, Arrival Rate).
- A **comparative study** on the performance overhead of the available consensus algorithms in Ethereum PoA (Proof of Authority), Ethereum PoW (Proof of Work), Hyperledger Fabric and Hyperledger Sawtooth, as an example of how BlockCompass can be used.
- A **usability study** with 33 participants to evaluate the user's experience with BlockCompass when considering different blockchain platforms compared to other blockchain benchmarking tools and manual effort.

4.4 Related Work

4.4.1 Benchmarking of Blockchain

Dinh et al. [6] presented a benchmarking tool called Blockbench, which is a cross-sectional benchmark framework. This tool can analyze blockchain technologies in terms of latency, throughput, scalability, and fault tolerance. The authors have designed drivers for three blockchain platforms including Ethereum, Hyperledger Fabric, and Parity. They have studied these platforms with various batch workloads to understand their performance and security and detect bottlenecks. This tool has two main drawbacks: it supports only batch workloads and it does not support monitoring of resource consumption (CPU, memory, network). Although the tool is open-source, extending it for new blockchain systems is not easy as there is no proper documentation for extendability. It cannot support large-scale experiments in some systems (e.g., more than 16 nodes in Fabric) and some of its drivers are outdated.

Saingre et al. [24] proposed a benchmark framework for reproducible research on blockchain technology performance (latency, throughput, and energy consumption). The BCTMark (Blockchain Technologies Benchmarking) is used for the deployment, comparison, and evolution of different blockchain platforms. The tool does not demonstrate high extensibility and reusability, especially when it concerns performance metrics.

Wang et al. [109] used Hyperledger Caliper, a tool to evaluate the latency and throughput of Hyperledger Fabric. This tool is developed by the Hyperledger foundation and it can now evaluate several blockchain platforms, including Fabric, Besu, and Ethereum. The authors in their batch experimental study have shown that when the number of users querying the system suddenly increases to 500 users, the response time increases 11 times. Caliper supports

only batch workloads.

Birim et al. [94] designed "GoHammer" to evaluate the latency, throughput, resource utilization and failed requests in Quorum blockchain network. The tool is partially similar to Quorum Profiling but the difference is that it currently supports only the Raft consensus algorithm and it is easier to start the workload and the network. The tool is a cross-sectional benchmarking tool. The tool still requires several improvements, such as the ability to monitor one blockchain node, more workload scenarios, and more blockchain platforms, extensions that can be provided by other researchers or practitioners, as the tool is open-source.

Gupta et al. [98] proposed the BFT-Bench tool to evaluate the performance and efficiency of different Byzantine Fault Tolerance State Machine Replications (BFT- SMRs). However, State Machine Replicas (SMRs) do not support transaction request verification, which is one of the main features of blockchain consensus protocols and has a significant impact on performance. Their experimental results demonstrate that this tool could be helpful for practitioners and researchers to evaluate SMRs based on blockchain.

A review of related literature [9, 24, 94, 96, 98, 99] and our experimentation with existing benchmark tools for blockchain showed that they are often designed and deployed for one or two platforms without a clear or explicit way to add more platforms. More importantly, the existing tools are cross-sectional which means that we need to configure a single batch transaction and we get the results for throughput and latency for the specified configuration. The problem with this kind of benchmarks is that we can not simulate a real-time application and to see when and how the application fails when the workload continuously fluctuates over time. In some cases, the benchmark tools do not provide adequate or even live monitoring of the performance and resource consumption of the application, information critical to maintaining the proper operation of the system in the long term.

4.4.2 Performance Studies on Blockchain

Wang et al. [4] used Hyperledger Caliper to evaluate Ethereum PoW, Hyperledger Fabric, Hyperledger Sawtooth and Fisco-Bcos. They studied the architecture of each platform and what kind of consensus protocols are supported. They only studied the throughput and response time which are the only parameters that Hyperledger Caliper can collect. Their results show that the performance of the Hyperledger Fabric is better than Ethereum and throughput of both Sawtooth and Fisco-Bcos is far better than Hyperledger Fabric.

A limitation of this work is that while Ethereum, Hyperledger Fabric, Hyperledger Sawtooth and Fisco-Bcos support multiple consensus algorithms, it is not explicitly mentioned which

consensus algorithms are used in the study of the blockchain platforms.

Kuzlu et al. [89] examined the effects of various batch workloads on the performance of the Hyperledger Fabric v1.4 regarding throughput, latency, and scalability. The authors used the Hyperledger Caliper tool to evaluate the blockchain platform. The findings of the experiments show that hardware capacity, smart contract complexity, and blockchain network architecture all impact on throughput and latency.

Dabbagh et al. [110] used Hyperledger Caliper to present a comparative performance evaluation of two blockchain platforms, Hyperledger Fabric V.1.1 and Ethereum, against four metrics by executing 100 transactions. With respect to average latency and the throughput, Hyperledger Fabric outperformed Ethereum. On the other hand, Ethereum performs better than Hyperledger Fabric in terms of the successful number of transactions when executing the Transfer function. For future work, the authors intend to conduct a study on the scalability of blockchain platforms and also see what is the highest number of transactions that each blockchain platform can handle.

In our previous work, we compared Ethereum PoW, Ethereum PoA, and Hyperledger Fabric [101]. The experiments were carried out manually, including blockchain installation, running the workload and monitoring the system. We observed that Ethereum PoW is the most expensive solution in terms of CPU utilization, memory usage and response time due to the design of its consensus algorithm. Moreover, we showed that Hyperledger Fabric with Raft consensus algorithm is the most expensive in terms of network usage. Finally, we found that Ethereum PoW consumes much less memory in comparison with Hyperledger Fabric when there are more concurrent users. This study motivated the creation of “BlockCompass," as presented here.

In summary these studies show that performance analysis of blockchain can be a complex task. On one hand, this is because most platforms have a large number of parameters to be configured that can affect performance, including consensus algorithm, transaction rate, batch size, number of nodes and others. On the other hand, we need to evaluate several performance dimensions to provide complete and holistic conclusions, including throughput, latency, resource consumption (CPU, memory, network) and others. To achieve meaningful and reliable results and reproducible studies, these concerns necessitate extensive and systematic configurations and instrumented monitoring. In addition, we have found that most of these studies are limited by the tools they use. Most notably, the tools employed by these studies use batch workloads, meaning that they are capable of stress testing the platform and identify their capacity. However, in longer running workloads, backlogs may appear which will make the platform’s performance degrade and fail even if there is no high stress at a given moment.

For these reasons, we have designed BlockCompass to be configurable to monitor the systems and the experiments to be extensible and to allow for transactional workloads to evaluate the endurance of the platforms under continuous and fluctuating workloads.

4.5 The BlockCompass Benchmarking Tool

In this section, we characterize the architecture and design of the BlockCompass tool, beginning with its requirements as identified in our literature review, and advancing to its components and details about their implementation for specific platforms, in addition to how the tool can be extended to support experiments on different blockchain platforms.

4.5.1 Functional and Non-Functional Requirements

The primary objective of BlockCompass is to enable developers (researchers or practitioners) to conduct experiments on the performance of blockchain systems and their different configurations and allow for their comparison. The basic workflow of using BlockCompass is as follows:

1. The end-user chooses the target blockchain platform and consensus protocol, and configures the workload settings, in terms of number of concurrent users and transactions per user.

2. The *workload generator* creates a continuous flow of data to and from the blockchain network through an adapter.

3. The *blockchain adapter* is responsible for the communication between the benchmark environment and the actual blockchain network. The adapter is specific to the network and exposes an API for communication with the workload generator.

4. The end-user can monitor the experiment's progress through a real-time *dashboard* and get a report with all of the experiment's performance information. The dashboard monitors the workload generator for input related metrics (e.g., transactions sent), the adapter for performance metrics (e.g., latency and error rate) and the network infrastructure for resource consumption.

Metrics are a key component in a benchmark platform. They provide the data to be analyzed during a benchmarking experiment. In its first version, the focus of BlockCompass is on performance and cost and so it currently supports the following metrics:

- **Emit rate**: The workload generator in BlockCompass generates concurrent users. Each user sends the next request only when they receive a response to the previous one. The emit rate indicator represents the average number of requests sent per second by all active users. This metric is mainly used as a baseline to assess the output performance of a blockchain network.
- **Throughput**: The average number of successful transactions that are received by the blockchain receiver node. This metric is used to assess the network's processing capability in terms of the emit rate. The network corresponds the number of total packets in Byte for an individual blockchain node.
- **Error rate**: The average number of failed transactions per second. This metric is used to assess the integrity of data and the robustness of the network. Failed transactions could depend on many reasons: scalability, bottlenecks, attacks, and so on.
- **Latency**: This metric measures the average time between sending a transaction from the workload generator to the blockchain network and receiving a response. When a blockchain node receives a transaction, it first generates a transaction hash and then returns a response. However, generating a transaction hash does not imply that the transaction is added to a block. This is the main performance metric for the network in the context of the benchmark. We currently only measure the time that we send the transaction to a blockchain receiver node till the time that we receive a response code which could be either failure or a success.
- **Resource Consumption**: This measures the consumption of CPU, memory, and network during the experiment. This metric can be used as an indicator of cost or energy consumption and it can be the cause for failures or errors.
- **Scalability**: Measures the variation of the latency, the throughput, the error and emit rates, and the resource consumption when the number of nodes and the number of concurrent users increase or decrease. This is a longitudinal measure that considers changes both in the workload and the network infrastructure. It is assessed as graphs over time presented by BlockCompass dashboard.

BlockCompass is also designed with a few "quality-of-life" properties both for developers and users. The platform guarantees the following quality properties on top of the functional requirements:

- **Extensibility:** BlockCompass is designed in a way that makes it easy to add new workloads, different monitoring tools or support for new blockchain platforms.

- **Usability:** BlockCompass requires minimum understanding of distributed technologies (e.g., cloud) and very little knowledge about blockchain technologies. We have used container technology to automate most parts of the benchmark including the installation of the blockchain network, the monitoring and the generation of workload. Users can easily configure their test scenario and get the results in a user-friendly report.
- **Maintainability:** BlockCompass is designed to be used by active projects and industry practitioners and to allow for continuous evolution. The BlockCompass code is open-source and follows community standard processes with respect to bug reporting or resolution, and adding new features.

4.5.2 Architecture

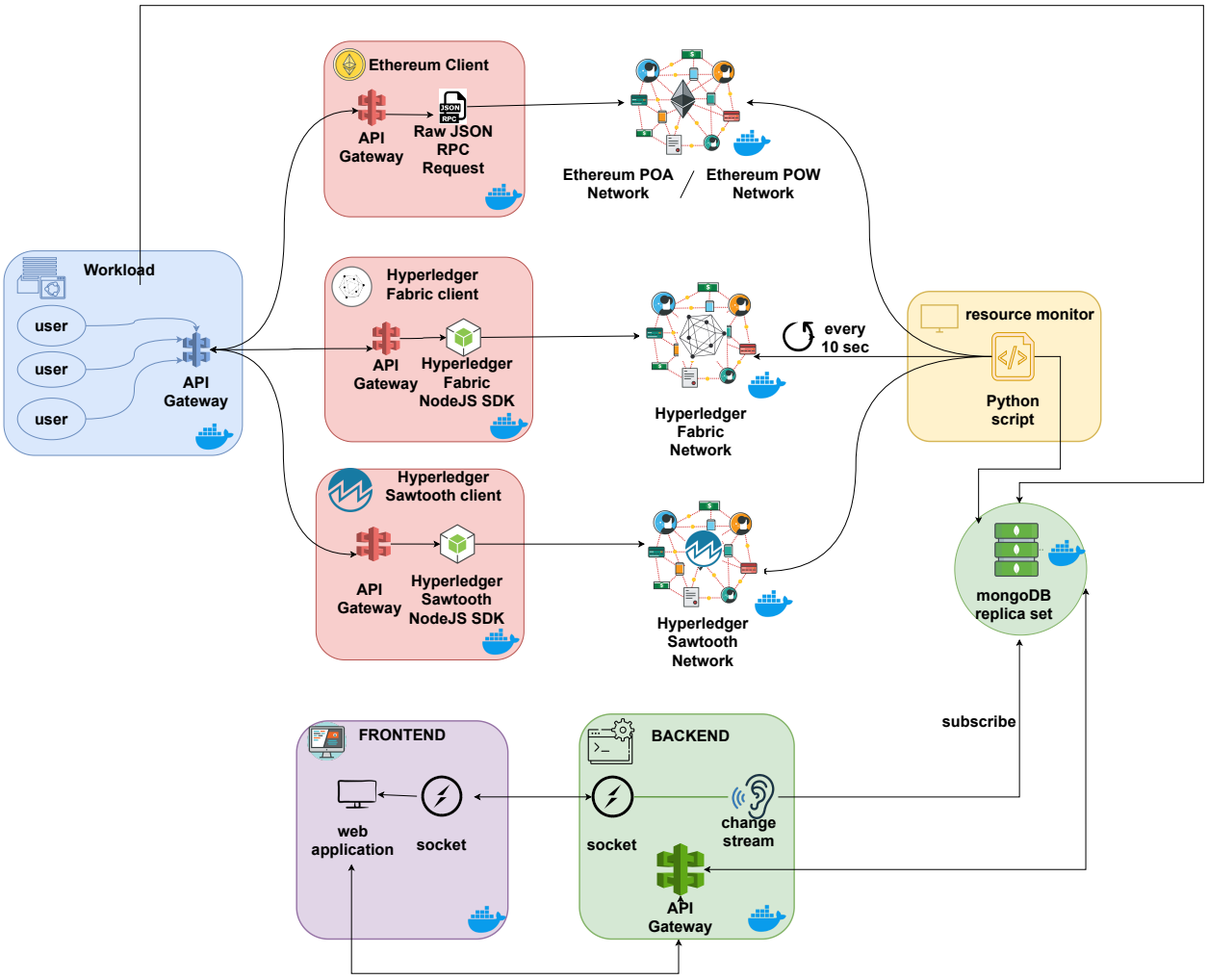


Figure 4.1 The BlockCompass architecture

BlockCompass follows a distributed multi-tier architecture, as shown in Figure 4.1. The first tier is the workload generator, which is responsible for sending requests to the adapter. The two components can be deployed on the same server or even on the client side. The fact that we use containers makes deployment or redeployment of components a fairly trivial task. The adapter then accesses the blockchain network. BlockCompass offers the option to access an existing network, like a publicly available Ethereum network, or a private network, which can also be deployed automatically. Finally, the dashboard, like the adapter and the workload generator, can also be deployed anywhere. Next, we describe each component in detail.

Workload generator: To exercise the subject of blockchain networks, the workload generator simulates users that send requests of different sizes and types of data in different frequencies. The workload is split into time intervals, for which the user of BlockCompass can configure the number of concurrent users, the size of data they can send, and how often they make requests. Finally, the sum of all the intervals determine the duration of an experiment. Given that each interval may have different number of users sending different requests, this creates a continuous and fluctuating workload, giving us the opportunity to test the capacity and endurance of the blockchain network.

Our workload generator is based on a closed transactional model [25, Chapter 8]. In this model, simulated system users send requests and wait for the response, either failed or successful. After some "think time" (as an artificial delay that simulates the processing of a response from the network), the user sends another request. While in an open workload we provide a fixed arrival rate for requests, a closed workload resembles a more realistic scenario with independent users who need to process the system's response before sending another request.

Figure 4.2 shows how the workload generator works in practice. The configuration contains the number of users at each interval and their type, which signifies what size of data they send every 10 seconds which is also customizable. A scheduler reads one line of the configuration file for every interval (approximately every 10-12 seconds) and it adds or removes active users between intervals accordingly. The data is then sent to the adapter. At the same time, we monitor the emit rate, latency and error rate (in case of failed requests) at the workload level. A monitoring thread calculates averages for these values every 10 seconds and stores them in a MongoDB database.

Client Adapter: Each blockchain platform provides its own unique methods to allow applications to interact with the network. The workload generator is agnostic of the underlying network and it uses a single predetermined format to send the data. The client adapter is then responsible for reformatting the data according to its corresponding network and calling its methods to send the data to the blockchain. All client adapters have the same API to

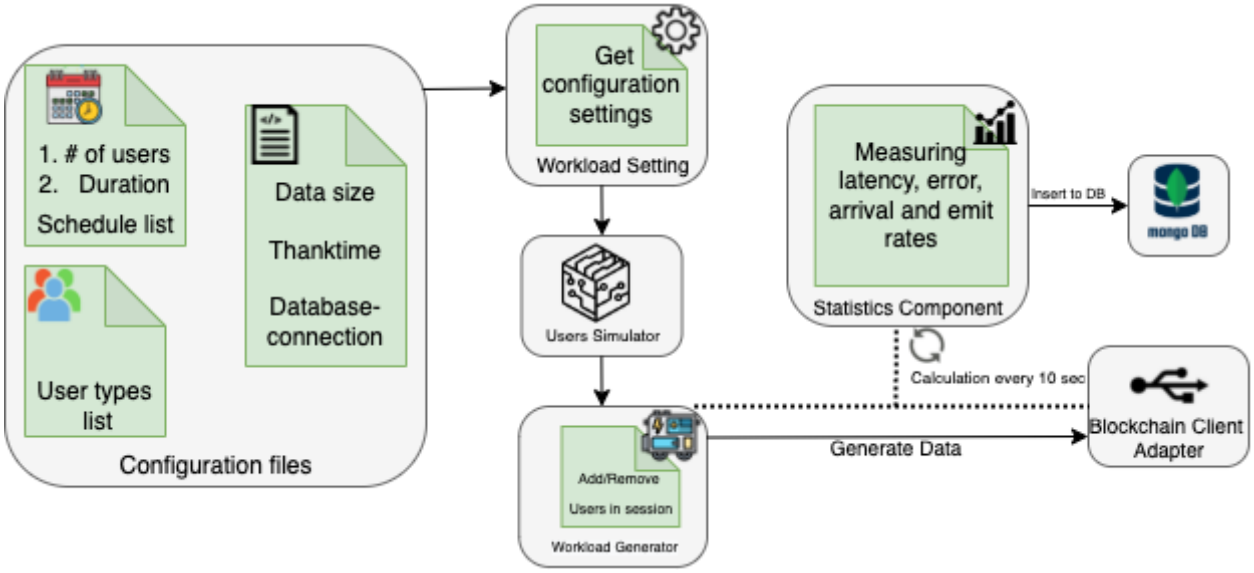


Figure 4.2 The architecture of the workload generator

receive data from the workload. This way the workload generator does not need to be changed for every new blockchain network, but an adapter needs to be created instead.

BlockCompass currently supports 4 different blockchain platforms, including: Hyperledger Fabric, Hyperledger Sawtooth, Ethereum with PoA and Ethereum with PoW. Each of the platforms need to have their own client adaptor in order to interact with the main blockchain network. It is usually provided by the developers of the blockchain platform itself.

Ethereum is a well-known public blockchain platform that is primarily used to mine an Ether (ETH) cryptocurrency. However, Ethereum can also be set up as a permissioned/private blockchain platform. Ethereum's public version depends on the network peers' computing capacity to validate blocks.

The **Proof of Work (PoW)** or **Proof of Authority (PoA)** consensus system may be set up on an Ethereum network (see https://geth.ethereum.org/docs/interface/private-network). In PoW, network peers are referred to as miners, and they compete to solve a cryptographic challenge. The first one to solve the cryptographic challenge will be rewarded an Ether as an incentive. Next, the miner broadcasts the block to the other nodes in the network.

Proof of Authority (PoA) is an another consensus protocol in Ethereum, and it utilizes less computational resources because a block is created by authorized and pre-approved signers based on their reputation, which is specified in the first block, also known as the "genesis block" when the network is created. After the creation, signers can be added or removed using a voting mechanism [111]. The processing of adding blocks in Ethereum PoA is called

"sealing" [112, 113]. In BlockCompass configurations, we followed the default parameter and set all the nodes as valid signers.

Ethereum also imposes limitations on gas consumption. The blocks in Ethereum also have a block gas limit which defines the maximum amount of gas a block can include. The initial block gas limit is defined in the genesis block, but this value can continuously fluctuate. As miners use the network, the gas limit can increase or decrease by about 0.1% for each new block.

Hyperledger Fabric is a private blockchain platform. It employs ChainCode [2] (specialized smart contracts) a type of smart contract programming language with specific functions and states. The purpose in which Fabric and Ethereum were designed varies.

In Fabric, peers are divided into three categories: orderer peers, committing peers, and endorser peers. The peer who owns the Chaincode and will be in charge of endorsing (approving) a proposed transaction is known as the endorser peer. Currently, Hyperledger Fabric is based on Raft Consensus algorithm.

Hyperledger Sawtooth [3] is an open-source blockchain platform which is designed especially for supply-chain managements. It is a member of the Intel-developed Linux Foundation umbrella project. Sawtooth's key advantage over other blockchain systems is its ability to process transactions and data in parallel rather than serially, improving system speed. Sawtooth supports both PoET [114], Raft [4] and Practical Fault Tolerance (PBFT) [98].

The **Proof of Elapsed Time (PoET)** consensus algorithm is based on the Trusted Execution Environment (TEE in Intel® [48]. For the PoET to work correctly, we would need to have at least three nodes. The PoET consensus relies on a "fair lottery" mechanism and Intel SGX CPU is responsible to provide a random sleeping time to all the network participants [48, 56].

The **Practical Byzantine Fault Tolerance (PBFT)** protocol is a voting-based algorithm designed on the Byzantine Fault Tolerance principle, which guarantees the system's integrity if up to one-third of the network's total nodes are malicious or corrupted ($n = 3f + 1$, where f is the number of malicious nodes). We need to have $2f + 1$ honest nodes to agree on processing a transaction [21, 59].

The **Raft** consensus protocol [22] is a leader-follower model based on Crash Fault Tolerance (CFT) for small and restricted platforms. The Raft consensus algorithm requires a significant amount of exchanges among the peers of the network as the majorities should agree on selecting the leader and all network progress. Subsequently, the more nodes we have in the

[2] https://hyperledger-fabric.readthedocs.io/en/release-2.3/
[3] https://sawtooth.hyperledger.org/faq/consensus/

network, the more exchanges we will have, making the extensive network impractical [58].

The **resource monitor** component monitors the CPU, memory, network input and output consumption. BlockCompass is currently implemented using Docker containers, and this also includes the deployment of the supported blockchain networks. Therefore, Docker stats is used as a tool to gather the resource consumption metrics. The collected data will be inserted into a MongoDB collection named "performance" as described in Figure 4.3.

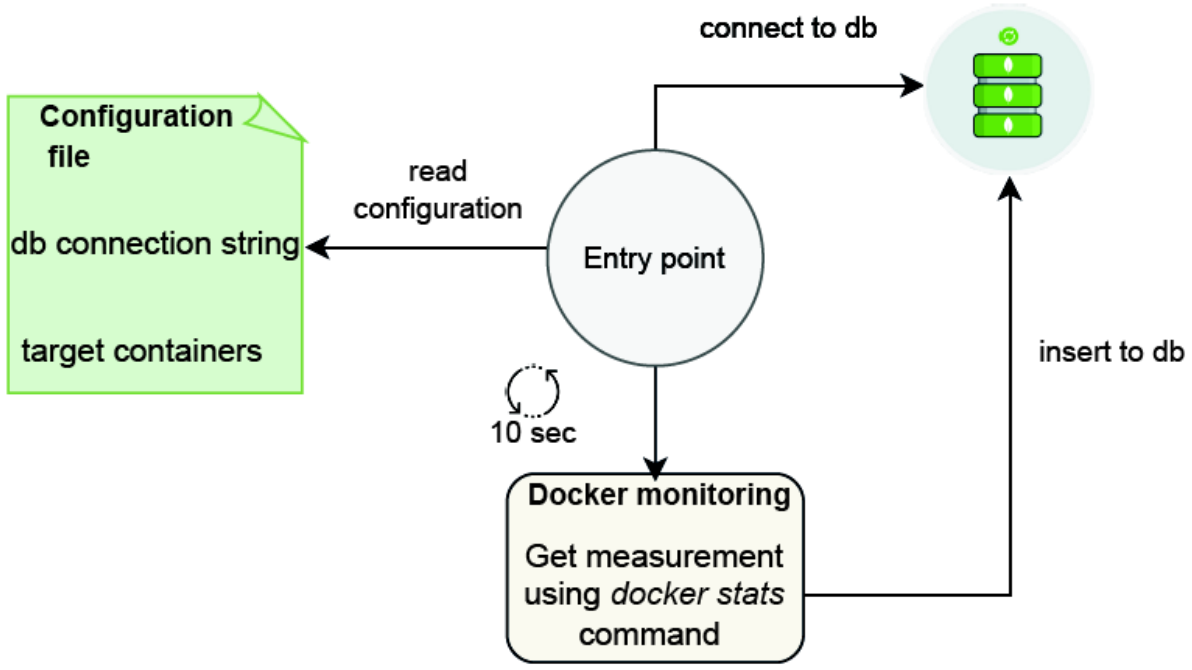


Figure 4.3 The architecture of the resource monitor

The data collected by both the Workload and Resource Monitor components are stored in different collections within a **MongoDB Replica Set**. Change Streams is a feature of the MongoDB Replica Set that allows applications to access real-time data updates. We used this feature to create a data stream used by the dashboard to generate real-time data visualizations as the experiment progresses.

The **Back-end Server** is a simple ExpressJS server that provides API endpoints to:

1. Get configuration details,
2. Get resource consumption data from the database,
3. Get performance metrics from the database.

The back-end server, in addition to the API endpoints, can subscribe to the MongoDB Change Streams and, upon receiving any updates, can push them to the front-end client through socket IO events.

The **Front-end Client** is an Angular 10 Web Application that displays real-time charts of all the collected metrics. The charts show the average of each metric over time as well as statistics per node. Aside from the charts, the front-end client displays statistical metrics such as average, standard deviation, variance, minimum, and maximum values in a summary table. It also provides the option of exporting the results as a PDF report.

The **Launcher script** [4] is a python script that automatically launches all the components needed to run an experiment, namely: the blockchain network, workload, monitor, backend, and frontend components. All containers are run in background.

4.5.3 Extensibility

BlockCompass' primary goal is to facilitate comparisons and analyses of blockchain platforms' performance for the benefit of researchers and other practitioners. As such, it is important to allow BlockCompass users to customize the composition of the tool to fit their needs, by replacing or extending its modules. Therefore, extensibility is an important BlockCompass characteristic. To achieve this, BlockCompass is built as a framework following the OCP (Open-Closed Principle), meaning it can easily allow for extensions but it is closed for modifications of each concrete component. Furthermore, we follow a "design by contract" approach, meaning that the main components of the framework, namely the workload generator, the client adapter, and the resource monitor, expose predefined interfaces, which makes replacing them with different implementations possible as long as the new implementations adhere to these interfaces and do not affect the flow of data between the components.

In this section, we explain how to extend BlockCompass. First, you can add support for a new blockchain platform. Then, you can change the workload. Finally, you can add a new way to track resources.

Extend support for blockchain

The steps to add support for a new blockchain network are as follows.

(i) `Deploy the blockchain network:` To add a new target blockchain, deployment of this network is required. There are options to deploy private networks locally or to simulate

[4] `https://github.com/polytechnique-ease/blockcompass/blob/master/launcher.py`

public networks also locally. It is also possible to connect to a public blockchain network. As BlockCompass is currently built around Docker containers, it is recommended to deploy the target network in containers to facilitate monitoring. If a different deployment is available or desired, the Resource Monitor module must be customized in the manner shown below.

(ii) `Implement the corresponding client adapter:` The client adapter is an intermediate component with two interfaces: one incoming from the workload generator and another outgoing towards the tested blockchain. Each blockchain platform has its own client implementation and supported languages. As a result, it is impossible to create a single common outgoing interface for all client adapters. Instead, each client adapter is treated as part of the tested platform and is the responsibility of the developer. On the contrary, the incoming interface is concretely defined to work with our workload generator as follows. The workload will send data to the client adapter, which will be responsible for submitting the transaction to the blockchain network in the correct format. The payload sent by the workload is in JSON format as follows: `{"function": "Write", "args": args}`. Where `args` is a table containing the key-value pairs to be stored in the blockchain network. By default, the client adapter runs on port 3000 under the `invoke` API endpoint, so that it can be called by the workload generator. After receiving the data, the client adapter needs to call the corresponding smart contract (see next step) to submit the data to the blockchain and return a response, either success or failure, to the user (or in practice to the workload generator).

(iii) `Implement the corresponding Smart Contract:` The smart contract contains the mechanism to submit the data from the workload generator to the blockchain network. In the case of the current version of blockcompass, the smart contract function as a simple key-value storage. Listing 1 shows the `kvstore` smart contract in Solidity for Ethereum. The smart contract implements a key-value map and two accessor functions: a function `get` that retrieves a value from the map based on a key and a function `set` that takes a key and value pair as input and stores them in the map.

```
1  contract KVstore {
2   function get(string key) constant returns(string) {
3      return store[key];
4    }
5    function set(string key, string value) {
6      store[key] = value;
7    }}
```

Listing 1: Ethereum kvstore Smart Contract

The same functionality for Hyperledger Fabric is shown in Listing 2. Hyperledger Fabric uses `chaincode`, which is the equivalent of a smart contract, implemented in the Go language. Because Hyperledger Fabric interprets the ledger as a key-value map by default, it is not required to specify a new structure as in the case of Ethereum.

```
type KVStore struct {
    contractapi.Contract }
func (t *KVStore) Write(ctx contractapi.TransactionContextInterface, key string, value string) error{
    var err error
    err = ctx.GetStub().PutState(key, []byte(value))
    if err != nil {return err    }
    return nil }
```

Listing 2: Hyperledger Fabric kvstore chaincode

(iv) `Adjust monitor script:` The last part of adding a new blockchain platform is adjusting the monitor configuration file by adding the new target blockchain and specifying the containers name pattern to monitor in the environment file[5]. By default, the monitor script will monitor containers whose names match the patterns given in the environment file: (.*name.*). Listing 3 shows an example of the environment file. For each blockchain target, we specify the container pattern names to monitor. For an Ethereum clique network with three miners, the container names are miner1, miner2, miner3, and bootnode. Using the following environment file, we will monitor all the containers that have the word *miner* in their names. It is also possible to provide more than one name pattern by separating the names with a comma.

```
ethereum-clique=miner
ethereum-ethereum-pow=miner
sawtooth-pbft=validator
sawtooth-raft=validator
sawtooth-poet=validator
Fabric=peer
```

Listing 3: Environment file for monitoring

Extend the Workload Generator

The workload generator needs to maintain a connection with the client adapter, as explained earlier, through a given port and a given API endpoint. Besides that, it is possible to change

[5] `https://github.com/polytechnique-ease/BlockCompass/blob/master/monitor/.env`

most of the internal implementation details of the workload generator, through a set of configuration files[6] [7]. The `users.list` file allows to specify the number of users, type of users and emit rate per user type. By adding new user types or changing the existing ones, the workload generator can be extended for different experiments[8].

Extend the Resource Monitor

The current version of BlockCompass is based on Docker containers and as such, it is using a custom monitoring script[9] that scrapes the results of `docker stats` to retrieve the metrics for resource consumption. The script then gives the option to save the data in a database or in a file (for reporting purposes). A developer has the option to change the monitoring tool and/or the database to store the metrics. This can be easily done by extending an abstract `Monitoring`[10] class. This class contains methods to connect and write to a database or write to a file. The code to retrieve and organize data is specific to the underlying monitoring tool and needs to be provided by the developer. Moreover, the developer has the option to change the database by providing a concrete extension to the abstract `Database`[11] class, which provides an interface for connecting to and interacting with a database. In its current implementation, BlockCompass is using an instance of MongoDB to store monitoring data.

4.6 Case Study

We demonstrate the functionality and the steps to configure and use BlockCompass in the context of a large-scale comparative study. The objective of this study is to compare the scalability, performance, and cost, in terms of resource consumption, between three blockchain networks, namely Ethereum, Hyperledger Fabric, and Hyperledger Sawtooth. In this section, we first outline how we implemented the three networks and how we set up BlockCompass for each one of them. The results of the study are then presented, as well as how they can be retrieved efficiently and in a user-friendly manner using BlockCompass.

[6] `https://github.com/polytechnique-ease/BlockCompass/blob/master/workload/IoT/schedule.list`

[7] `https://github.com/polytechnique-ease/BlockCompass/blob/master/workload/IoT/sensors.list`

[8] To extend the workload generator to support different type of users and generate specific data or changing the way it stores the data in database, you can visit: `https://github.com/polytechnique-ease/BlockCompass/blob/master/workload/documentation.md`

[9] `https://github.com/polytechnique-ease/BlockCompass/blob/master/monitor/dockerMonitoring.py`

[10] `https://github.com/polytechnique-ease/BlockCompass/blob/master/monitor/monitoring.py`

[11] `https://github.com/polytechnique-ease/BlockCompass/blob/master/monitor/database.py`

4.6.1 Study Motivation

To evaluate the capability and ease of use of BlockCompass, we deployed four different blockchain platforms, including Ethereum with PoW, and Ethereum with PoA consensus algorithms, Hyperledger Fabric with Raft, and Hyperledger with PoET, PBFT, and Raft consensus protocols. All these platforms are commonly used among practitioners, industries, and researchers. In terms of blockchain design and codebase maturity, they hold various positions. Our collaboration with several industrial partners inspired us to design the BlockCompass tool. We were tasked to design and prototype a decentralized application on both public and private blockchain platforms with a large number of users and requests in real time. In addition, we had to investigate and compare several prominent blockchain platforms, consensus protocols, and configurations to find the most scalable platform. We've done various private blockchain comparative tests before the development BlockCompass. In these tests, we had to install platforms, plan workloads, build monitoring scripts, and incorporate blockchain adapters separately and manually.

In this research, we show how much simpler it is to utilize BlockCompass to analyze the performance and resource consumption of multiple blockchain networks, specifically if the performance and resource consumption of the same consensus protocol in two distinct blockchain networks may be influenced. BlockCompass differs from other cross-sectional tools on the market, such as Blockbench [6] or Caliper, in that it can simulate concurrent users who fluctuate over time. In practice and in collaboration with our industrial partners, BlockCompass assisted us in determining bottlenecks of a blockchain network like Hyperledger Sawtooth. More specifically, our experiments showed that when the network gets overloaded with too many requests and users, the whole platform fails and must be reset and restarted. It would have taken days to install all the necessary dependencies to do a side-by-side performance evaluation of various blockchain systems without BlockCompass. Thanks to BlockCompass, as an all-inclusive tool, it is possible to set up the environment and install all dependencies in less than 15 minutes and then repeat the same experiments for different blockchain platforms under similar, controlled conditions. In this section, we demonstrate how BlockCompass can be set up, installed and configured in the context of a comparative analysis between the four blockchain platforms.

4.6.2 Experimental setup

BlockCompass was installed on a single Amazon `t2.large` virtual machine running Ubuntu 18.04 with 8GB of RAM, 2 vCPUs, and 25GB of hard disc space. This VM served as the host for the Docker containers of: the blockchain network's nodes, workload, monitoring, frontend

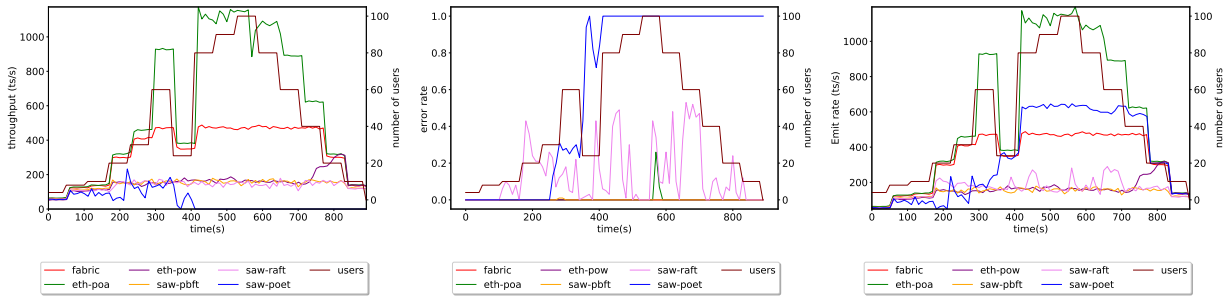
and backend components. In our testing, each Docker container is given only 20% of the VM's total CPU and memory to ensure that we have enough resources for all elements of the infrastructure. We have used one virtual machine for all components and each component such as miners is a docker container.

For the blockchain networks, which were deployed locally, we used geth (v1.10.3) [115] a Go implementation of Ethereum, for Hyperledger Fabric [13], we used v2.3.2, and for Hyperledger Sawtooth, we used v1.2 (Chime) [116]. Each blockchain network is composed of five nodes, deployed in containers. For Ethereum PoW and PoA, we used five miners and one bootnode. For Hyperledger Fabric, we used two organisations, one ordering service and one channel. The first organisation has two peers, and the second one has three peers. We also set the block period to two seconds. Finally, for Hyperledger Sawtooth, we have set up a network for each supported consensus (PBFT, Raft, and PoET), each running five validator nodes.

As far as the workload is concerned, we used different types of users that send queries ranging from 70 to 80 KB. The length of each experiment was fixed at 16 minutes, and the number of users ranged from 0 to 100. The number of users, emit rate and the duration of their existence can be easily customized in `workload/IoT/schedule.list`.

4.6.3 Results

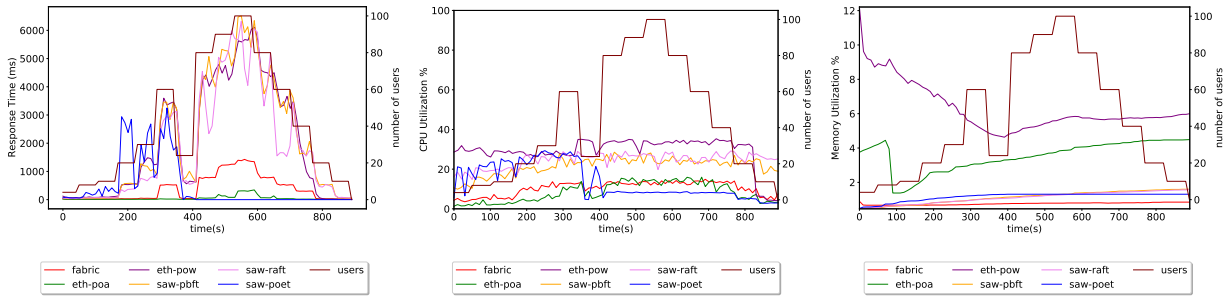
As shown in Figure 6.2a, the number of queries handled within 16 minutes varies due to the variation in response time across platforms. Table 6.1 shows that Ethereum PoA has the greatest overall throughput. This is due to the fact that Ethereum PoA has the shortest response time, while Ethereum PoW is on the other end of the spectrum. As we said before, this divergence is to be expected and is based on how the consensus algorithm works. In contrast, Sawtooth has a lower throughput, despite being implemented in both Hyperledger Fabric and Hyperledger Sawtooth algorithms. This is because, as seen in Figure 6.2b, Sawtooth-Raft began to experience errors and requests were failing. As a result, the throughput is dependent on the consensus and the platform implementation.In contrast, Sawtooth has a lower throughput, despite being implemented in both Hyperledger Fabric and Hyperledger Sawtooth algorithms. And the error rate in Sawtooth Raft varies with the number of users; as the number of users grows, so does the error rate, indicating that the system cannot manage a large number of users at the same time. In our experiments, as shown in Figure 6.2c, Sawtooth PoET can only handle concurrent users until 250 seconds before the network falls down and must be restored. This coincides with the first large increase in the number of users and results in the total collapse of the platform after the second large increase. As we can see, the Sawtooth Raft error rate varies with the number of users. Only the PBFT consensus



(a) Total Throughput Vs Number of users

(b) Error Rate Vs Number of users

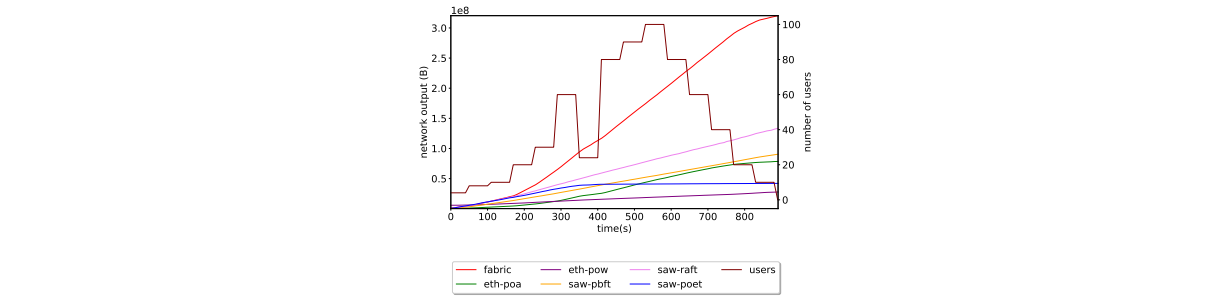
(c) Total Emit Rate Vs Number of users



(d) Avg Response Time Vs Number of users

(e) Avg CPU Utilization Vs Number of users

(f) Avg Memory Utilization Vs Number of users



(g) Cumulative Network Output Vs Number of users

Figure 4.4 Number of send requests, Average Response Time, CPU Utilization, Memory Utilisation and Network throughput using 5 nodes and 1 receiver node

Table 4.1 Summary of results of comparison between Hyperledger Fabric (Raft), Ethereum PoA , Ethereum PoW, Sawtooth PBFT, Sawtooth Raft and Sawtooth PoET

Algorithms	Fabric Raft	E-PoA	E-PoW	S-PBFT	S-Raft	S-PoET
total number of requests	30770	53135	13986	12865	14671	31156
Total throughput	30770	52743	13986	12865	12175	3844
Avg Error (%)	0	0.40	0	0.02	13.00	62.00
St.Dev. Error	0	0.028	0	0.0014	0.16	0.46
Avg CPU (%)	10.36	8.24	29.19	21.62	24.51	13.71
St.Dev. CPU	3.43	4.84	7.5	4.37	3.59	8.65
Avg. Response Time (ms)	385.89	50.60	2034.67	2116.62	1849.53	447.92
St.Dev. Response Time	456.35	79.77	2032.28	2064.89	1937.93	868.03
Avg Memory (%)	0.75	3.53	6.29	1.12	1.1	1.16
St.Dev. Memory	0.07	0.87	1.39	0.35	0.33	0.23
Avg. Network Input (MB)	39.24	58.21	17.45	36.81	61.24	23.73
St.Dev. Network Input	154.15	31.15	7.25	23.39	38.08	10.87
Avg. Network Output (MB)	144.3	35.35	16.27	43.45	67.50	32.65
St.Dev. Network Output	108.14	27.85	6.63	27.49	41.87	12.9

protocol in Sawtooth is error-free throughout the experiment.

Figure 6.2d illustrates the average response time for 6 platforms. We can observe that there is a correlation between workload and response time in terms of the number of users in all the blockchain networks, as is expected. We can also observe based on Figure 6.2d and Table 6.1 that Ethereum PoW and Sawtooth PBFT have the highest response times among all platforms which is 2034.67ms for Ethereum PoW and 2116.62ms for Sawtooth PBFT. This is because the generation of a transaction hash requires more resources for these two algorithms in particular. When comparing response time of Raft in Fabric and Sawtooth, we see a noticeable difference between the two, indicating that response time depends on the consensus as well as the platform itself. Our assumption is that the architecture and implementation of the blockchain platform for both platforms are different, for example, Hyperledger Fabric consists different network peers components including Endorser, Anchor, and Orderer peers, and only Orderer peer is responsible for creating a block and reaching the consensus based on Raft consensus protocol; therefore, for communications, only Orderer peers are involved. Hyperledger Sawtooth only consist of multiple validators, and all of them are responsible for adding a new block; therefore, communication is required among them to reach a consensus. Moreover, in Sawtooth they have extended the Raft algorithm to use a stable storage mechanism as the original implementation of Raft is based on in-memory storage and is faster for communication. The reason for this extension is to enable Sawtooth

to restart in the event of a crash or arbitrary shutdown.

In terms of CPU utilization as it is shown on figure 6.2e and Table 6.1. Ethereum PoW has the highest CPU consumption of about 29.19%, mainly explained by how the consensus algorithm works. We also observed that the CPU consumption for Ethereum PoW remains relatively constant regardless of the fluctuation in the workload. This is the result of the difficulty parameter in the PoW consensus algorithm. When the number of transactions and load increase in the network, the difficulty reduces, and when we have either nothing or a few transactions in the system, the difficulty parameter increases, resulting in a balanced CPU usage throughout the experiment.Ethereum PoA is the most efficient algorithm in terms of CPU consumption, as it is only based on authority of each node of the network. In other words, we trust the miner to reach a consensus, and there is no mathematical puzzle to solve, so fewer resources are required. We can see that high CPU correlates with higher response time, especially for the three slowest networks (Ethereum PoW, Sawtooth PBFT and Sawtooth Raft). This is natural as the first two requiring significant work during the validation of blocks, while in the third Sawtooth in general is the ones that requires the higher workload, which motivated the use of PoET consensus.

As shown in Figure 6.2f memory is not a significant resources for any of the studied networks. Therefore, it is difficult to draw any specific arguments. On the other hand, Figure 6.2g shows the network traffic between the nodes. Here we can see that Raft (either in Fabric or in Sawtooth) and PoA in Ethereum have the highest network activity. For the two of them (Fabric and Ethereum), this compensates for the low CPU usage, as in these cases, validation occurs through communication between the peers rather than any particular computation load on each peer. In Sawtooth Raft the two resources (CPU and network) are used to a similar degree for presumably better balance between performance and security. In fact, Fabric requires fewer nodes to validate a transaction (the endorsed peers), while Sawtooth will use all of its validators.

4.6.4 Discussion

In previous works, we had performed similar studies to compare different blockchain platforms [100] or different consensus algorithms [108]. Without having BlockCompass, we had to firstly, install all the dependencies for each platform individually, setting up environments, checking the latest packages and see whether there are conflicts, installing the blockchain platform individually, designing a custom closed-workload for continuous and fluctuating users with different workload sizes, writing a custom monitoring script to capture the performance and resource usage, writing a script to export the results and finally generate figure and

statistical results. Only the installation, configuration and execution of the experiments (without the final analysis) could take a couple of days by a single researcher. With BlockCompass it is possible to fully automate or at least systematize some of these steps. This results in a set up of less than 15 minutes, again by a single researcher or tester. Thanks to its container technology, BlockCompass allows the execution of an experiment with a single command, and eventually produces a publication ready report with summary results and visualizations. Workload and other configurations can be changed easily and in a rapid manner. Live monitoring helps to decide faster which configuration is suitable compared to waiting for the experiment to finish, and exporting and analyzing results.

4.7 Usability Study

In order to evaluate the usability and efficiency of BlockCompass in an unbiased way, especially in comparison to other similar blockchain benchmarking tools, we conducted a usability study involving a group of senior undergraduate software engineering students enrolled in software architecture and advanced design course at Polytechnique Montreal. The students were asked to conduct a small-scale comparative study with either BlockCompass or with another blockchain benchmarking tool named Blockbench [6]. Blockbench is designed to perform "Batch Workloads", which means that the workload runs until the experiment is completed or if it fails. On the other hand, BlockCompass is designed to create transactional workloads, which means that it simulates concurrent and fluctuating users with different data size to stress the network and simultaneously monitor resource usage and performance in real-time. After the completion of the study, we asked the students to answer a set of questions regarding their experience with the tool in an anonymous survey[12]. The students were given a live demo of each tool in 2 different live sessions. Also, a user manual and offline tutorial were provided for the students to follow the steps required for both Blockbench and BlockCompass.

Figure 4.5 shows a flowchart on the participation in the usability study. 33 students were initially enrolled in the course and all were contacted to participate in the study. Eventually, 26 (or 79%) accepted to participate. Out of these, 15 (or 58%) used BlockCompass and 11 (or 42%) used Blockbench. Also, 14 (or 54%) used Fabric (8 with BlockCompass and 6 with Blockbench) and 12 (or 46%) used Ethereum (7 with BlockCompass and 5 with Blockbench). Overall, we had an acceptable balance between benchmarking and blockchain platforms.

The questions focus on four different measurable quantities that are spread across all four stages of the experimentation process: the installation phase, the configuration phase, the

[12]The usability study received ethics approval from Polytechnique Montreal Research Ethics Committee, Certificate #CER-2122-29-D-14 December 2021

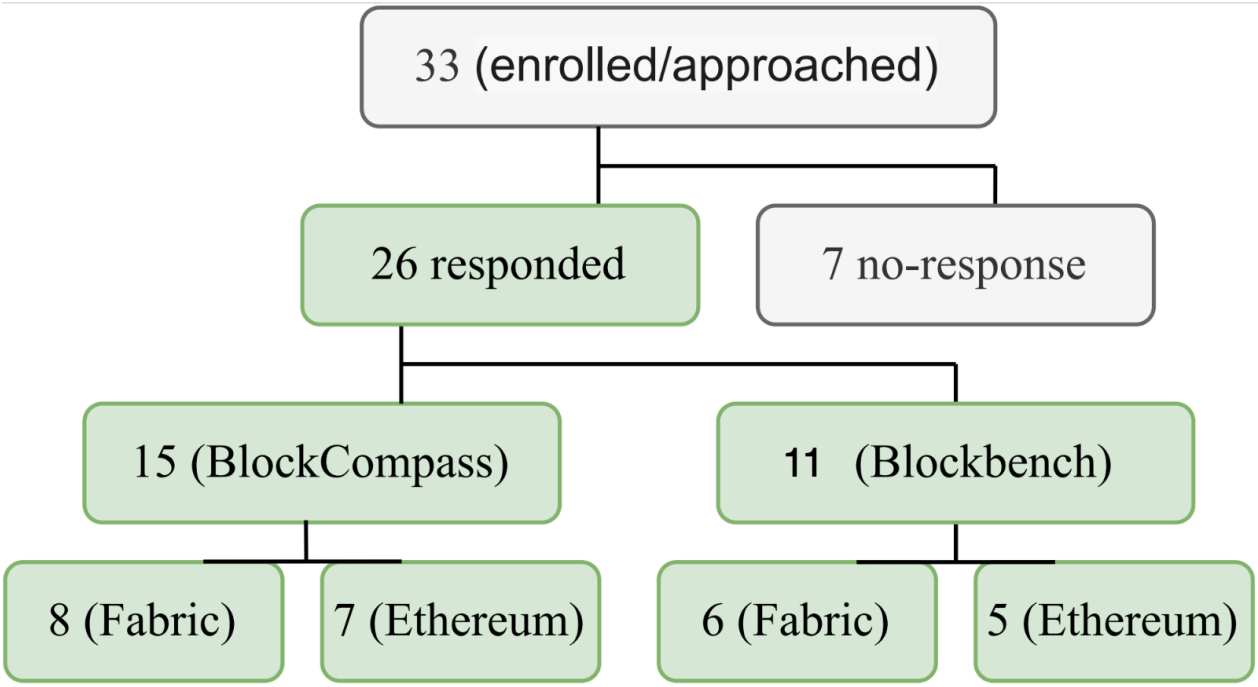


Figure 4.5 Flowchart on the participation in the usability study

experimentation phase, and the reporting phase. Installation refers to installing prerequisites and setting up the platforms, including both the blockchain and the benchmark platforms. Configuration mainly refers to setting up the experiment, including setting up the environment and the virtual machines, configuring the workload and other tasks. Experimentation is simply running the experiment and reporting is getting the output, that may include data files or visualizations. For each of these phases, we enquired about the time it took on average to complete the phase, the number of attempts it required to successfully complete the phase, the number of errors that occurred during these attempts and, finally, about what percentage of these errors the participant believed was due to problems in the documentation or in the instructions provided. We also asked the participants to submit a complete list of their attempts with the associated time and errors, but we only received 4 responses, which we eventually did not analyze further. Finally, we included 5 questions on a Likert scale (1-10) about the participant's general satisfaction of the benchmark platform, its documentation, the reporting of results and whether it is generally easier to use a benchmark tool rather than perform the experiments manually.

To test the statistical significance of a difference between the averages for BlockCompass and Blockbench, we applied the Student's t [117] and the Wilcoxon Mann-Whitney U-tests [118] and for each question, depending on whether the responses were normally distributed or

not, respectively. Normality was checked for each question with the Shapiro-Wilk test [119]. Moreover, as we are doing multiple comparisons (i.e., for multiple questions), we applied a Bonferroni correction [120], so that $p = 0.05/m$, where m is the number of questions, in our case 20. Finally, if $p < 0.0025$ in the original U or t-tests, the null hypothesis (i.e., the two averages do not differ) was rejected.

Table 4.2 shows the average responses for all questions between BlockCompass and Blockbench. As it can be seen with respect to time, the two platforms are comparable, with BlockCompass being slightly faster when it comes to configuration and reporting. Similarly, using BlockCompass resulted in fewer attempts for each phase, with the exception of running experiments, however these differences were not found to be significantly different. Users consistently encountered more errors when working with Blockbench compared to BlockCompass, again with little to no statistically significant difference. Users reported that these errors mainly stemmed from lack of clarity in the documentation and instructions on Blockbench especially when it concerned installation errors (87.37% of errors). This difference was found to be statistically significant. Overall, participants were much happier with their experience on BlockCompass (scoring it between 7-8) than they were with Blockbench (scoring it between 4-5). This made the BlockCompass participants more favorable towards automatic benchmark solutions than manual efforts.

4.8 Conclusion

In this paper, we presented, BlockCompass, a novel benchmarking tool for blockchain platforms and their configurations. BlockCompass comes with a configurable workload generator that simulates concurrent users overtime, three adapters/clients for Ethereum, Hyperledger Fabric, and Hyperledger Sawtooth, a monitoring framework based on Docker containers, and a reporting tool with a live dashboard or exportable results in PDF or CSV format. The benchmarking platform is configurable and extensible and can be used in a variety of studies or stress and capacity tests to evaluate the performance and capabilities of blockchain solutions.

We demonstrate the use of BlockCompass and its capabilities in the context of an empirical study to compare three distinct blockchain platforms Ethereum, Hyperledger Fabric, and Hyperledger Sawtooth that employ distinct consensus protocols. During the study, we discuss how the experiments can be set up, what tools need to be configured and how to prepare the blockchain platforms to be compared. Eventually, we discuss the results of the study as they were obtained with the help of BlockCompass.

To evaluate the quality and usability of BlockCompass, we have conducted a survey study

and compared it with Blockbench, an existing benchmarking tool for blockchain. Overall, the participants of the study were significantly more satisfied with BlockCompass regarding their experience with running benchmark tools, clarity of the documentation, presentation of final results, ease of performing experiments compared to manual effort or to other tools. In general, BlockCompass performed slightly better than Blockbench in terms of the amount of time it took to complete the experiments, as well as configuration, reporting, and documentation.

Table 4.2 Survey results. Each row reports the average of all responses for the two tools and the p-value for the corresponding statistical test (t for Student t-test and normal distribution, U for Mann-Whitney U-test and non-normal distribution). Significant differences ($p < 0.0025$) are noted in grey.

Question	BlockCompass	Blockbench	p-value
Installation-Time	27.5 min	40.27 min	0.01^t
Installation-Attempts	4.13	10.36	0.01^t
Installation-Errors	2.73	3.55	0.19^t
Installation-DocErrors	27.57%	87.37%	0.001^U
Configuration-Time	34.87 min	38.50 min	0.59^t
Configuration-Attempts	4.13	5.18	0.17^U
Configuration-Errors	2.73	4.55	0.028^t
Configuration-DocErrors	34.87%	52.73%	0.006^U
Experiment-Time	53.13 min	53.10 min	0.47^U
Experiment-Attempts	5.07	3.90	0.96^U
Experiment-Errors	3.13	4.60	0.31^U
Experiment-DocErrors	36.67%	41.90%	0.23^t
Report-Time	31.87 min	44.90 min	0.05^U
Report-Attempts	4.20	4.30	0.94^t
Report-Errors	2.21	3.30	0.11^U
Report-DocErrors	19.33%	37.82%	0.09^t
Overall experience	7.07	5.27	0.001^t
Documentation quality	7.20	4.82	0.003^t
Presentation quality	7.40	4.82	0.00004^t
Ease/Speed vs Manual	7.93	5.18	0.0007^U
Ease/Speed vs Other tools	7.05	4.27	0.00001^t

CHAPTER 5 ARTICLE 2: PERFORMANCE EVALUATION OF DISTRIBUTED LEDGER TECHNOLOGIES FOR IOT DATA REGISTRY : A COMPARATIVE STUDY

5.1 Paper Information

Title

Performance Evaluation of Distributed Ledger Technologies for IoT data registry : A Comparative Study

Authors

Mohammadreza Rasolroveicy, and Marios Fokaefs

Date of submission: 2020/04/15

Published in: WorldS4 2020 - LONDON

Chapter Overview: This Chapter presents an empirical study of four blockchain technologies that include Hyperledger Fabric [34] with PBFT, Hyperledger Sawtooth [35] with PoET, BigchainDB [36], and Hyperledger Burrow [37], to measure the latency, memory utilization, and CPU consumption. As baseline performance, we used MongoDB which does not have any consensus algorithm. Further, we also studied the overheads around the use of blockchain and to study whether there is a single optimal solution with respect to response time and computation overheads. To fairly compare the performance of private blockchains with databases, we measured latency in terms of the response from one blockchain receiver, not the time it takes to actually verify a transaction. This Chapter intends to identify the performance bottleneck on different private blockchains in terms of resource utilization and response time and also demonstrate how blockchain performance differs from NoSQL distributed databases. In this study, we observed that Hyperledger Sawtooth is more efficient than other platforms and therefore, we decided to choose this private blockchain technology and compare other built-in consensus protocols and proposed two different self-adaptive frameworks which will be further discussed in Chapters 8 and 9.

This Chapter covers two of the five dimensions of blockchain performance we explore in this thesis, namely blockchain vs NoSQL and consensus.

5.2 Abstract

Internet-of-Things (IoT) technologies have gained prevalence in various areas including Smart Vehicles, Smart Buildings, and Smart Health. Especially in these domains, the use of IoT involves certain challenges around security, privacy, and trust, which can be characterized as sensitive issues. Blockchain technologies have emerged as the potential solution to address these issues. However, Blockchain can be inefficient in terms of both performance and cost due to high bandwidth overhead and delays. In this paper, we have examined four different popular Blockchain platforms (Hyperledger Fabric, Hyperledger Burrow, Hyperledger Sawtooth, and BigchainDB) to identify what are the overheads around the use of Blockchain and to study whether there is a single optimal solution with respect to time and computation overheads, or if there are trade-offs between the four platforms for IoT applications.

5.3 Introduction

The Internet of Things (IoT) refers to the networked interconnection of everyday objects such as devices, vehicles, buildings, which are embedded with electronics, software, sensors, and network connectivity enabling them to collect data and communicate between human being as well as other devices [121]. The main feature of IoT is its notable impact on various aspects of the daily life of the potential users. Some areas where IoT is already being applied, in one way or another, include Smart Buildings, healthcare, and enhanced learning. Unprecedented growth and expansion in the number of IoT devices have raised various challenges for organizations and developers such as security, big-data, heterogeneity, variety and variability, real-time deployment and logging, integration, and many others. Lack of fundamental security safeguards exacerbates the privacy risks of IoT in many of the first generation IoT products like surveillance cameras, which can be considered as “things”.

The vulnerability of IoT systems has been exposed in a number of instances where the most notable one is "The Mirai Botnet Attack" [122] which was on the verge of infecting IoT devices, such as security cameras, and trying to gain access to data. Another example of vulnerable IoT devices is pacemakers. In January 2017, a report stated that the pacemakers from St. Jude Medical’s were vulnerable and hackers tried to drain batteries or even manipulate the patients’ hearts which could threaten their lives[1].

One possible solution to guarantee the security of IoT devices is Blockchain, which acts as an immutable (permanent and unalterable), consensus-driven (based on trust verification), decentralized (through networked copies) and transparent public record of transactions (called

[1]https://bit.ly/2LR0J5q

ledger) guaranteed by a untrusted peers on distributed network along with a number of key security, privacy, and compliance [16, 123].

Security, privacy and trust are sensitive issues and challenging questions for IoT applications. Although these issues are addressed up to a degree by Blockchain, the technology itself can often prove inefficient in terms of both time and cost, due to high bandwidth overhead and delays, which is deemed to be undesirable for IoT devices [54, 124].

In this work, we present a comparative study between four popular distributed ledger technologies (DLTs) , namely Hyperledger Fabric [13], Hyperledger Burrow [72], Hyperledger Sawtooth [56], and BigchainDB [36]. As this study only focuses on DLTs for storing IoT data, we did not study other platforms such as IoT which is based on Directed Acyclic Graph (DAG). Our main objective is to quantify the overheads caused by the use of Blockchain and understand these overheads in comparison to a regular large-scale database. For this purpose, we compare the four Blockchain installations with a MongoDB [125] installation and we stress the five systems under the same workload of a simulated IoT network. A secondary objective of this study is to examine how different variants of specific Blockchain properties, namely the consensus protocol, affect the performance of the system. The rest of the paper is organized as follows. In Sections 5.4 and 5.5 we outline the related literature and provide the background of the concepts and technologies used in our study. In Section 5.6, we present the methodology and preparation for our study, while in Section 5.9, we discuss the results and the threats to validity of our study. Finally, Section 7.13 concludes our work.

5.4 Related Work

5.4.1 Integration of Blockchain and IoT

Papadodimas et al. [126] proposed a decentralized application (DApp) based on Blockchain technology to share IoT sensor data and signify the different challenges tackled during the development process. The objective of this application is to combine Blockchain technology with IoT and operate by means of smart contracts carried out on the Ethereum Blockchain. However, Ethereum generally underperforms resulting in lower transaction processing throughput and higher latency, which are not suitable for IoT networks.

Dorri et al. [16] explored and outlined thoroughly the various core components and functions of the smart home domain. In their model, each smart home is equipped with an always-online, high resource device, known as ”miner” that is responsible for carrying out all communications inside and outside the home. Also, the miner has to maintain a private and secure Blockchain, used for controlling and auditing communications. They proposed a novel method to employ

Blockchain by eliminating the concept of "Proof Of Work". Their simulation results indicate that the obtained overheads are low and can be managed for low resource IoT devices. However, their model has a significant shortcoming which eliminates the key characteristic of Blockchain making the system vulnerable to notable attacks such as double spending and forgery attacks [54]. To evaluate their model, they have presented their result with the Cooja Simulator [91], which is used to simulate Wireless Sensor Networks (WSN). However, this virtualization is only used for device simulation, and not the applications that process the sensor data. The virtual IoT network employed in our study is able to handle data ingestion and test different Blockchain networks from end to end.

Aung and Tantidham [53] proposed the idea of applying Ethereum private Blockchain for the smart home system (SHS). In this work, SHS has been considered as a case study which is an integration of home appliances together with sensors to get automatic operations for heating, lighting, air conditioning, home security, health care systems, and others. The authors presented an approach of private Blockchain implementation for SHS to cope with its privacy and security issues. They aimed to improve the works of Dorri et al [16], by introducing smart home miners, private Blockchain and local storage to connect smart home sensors and actuator services. Since Ethereum has very slow transaction time, it is not suitable for real-time applications for IoT devices which are time-sensitive.

Paul et al. [127] proposed a Blockchain architecture, the typical integration of Blockchain in IoT applications such as smart cities causing considerable energy consumption, delay, and computational overhead. Their model is based on the Ethereum platform to ensure confidentiality, authenticity, availability, integrity, and non-repudiation. A comparison with existing architectures depicts that their architecture outperforms others in terms of efficiency and computational overheads. However, they have used Proof of Work for the mining process which is computationally expensive and it is not suitable for IoT devices which are resource-constrained.

5.4.2 Performance Evaluation of Blockchain

Gorenflo et al. [128], have reengineered a permissioned Blockchain system, Hyperledger Fabric, to increase the transaction throughput from 3,000 to 20,000 transactions per second. Their main goal was to implement a series of independent optimizations concentrating on I/O, caching, parallelism and efficient data access. The authors have executed the whole system without deploying Docker containers to avoid extra overhead. The proposed architectural changes decrease computation and I/O overhead during transaction ordering and validation such that throughput is improved significantly.

Pongnumkul et al. [88] have presented a performance analysis of Ethereum and Hyperledger Fabric as private Blockchain platforms with a diversified number of transactions. The authors have shown that Hyperledger Fabric attains higher throughput and lower latency compared to Ethereum when the workloads are increased to 10,000 transactions. Likewise, the differences between Hyperledger Fabric platforms with respect to execution time and average latency become more significant as the number of transactions increases. The average throughput of Hyperledger Fabric also changes at a much faster rate than that of Ethereum. However they have not shown that what is resource and energy consumption when we apply these two platforms in distributed systems.

Hao, Yue, et al. [87], have studied the performance of two permissioned Blockchain platforms including Ethereum and Hyperledger Fabric. The consensus algorithms for Ethereum is PoW and for Hyperledger fabric is PBFT. The results demonstrate that the main bottleneck of Blockchain performance is its consensus algorithm. Their evaluation results depicts that PBFT outweighs PoW in terms of both average response time and throughput. For future studies, they suggested to consider different aspects of Blockchain such as number of nodes, decoupled consensus and the impact of the number of transactions on the performance under case of different number of nodes.

Baliga et al. [96] have provided an experimental approach to show the throughput and latency of Hyperledger Fabric. To do so, they have arranged a Blockchain consortium with two organizations, each contributing two peers to the Blockchain network. The ordering service runs on a separate node managed by a third-party organisation. Organisation 3 is supposed to be a neutral entity. The endorsement policy on transactions was determined to consist of signatures from at least one peer from each organization to successfully commit to the Blockchain. The main goal of the experiment was to study how the transaction latency varies as the size of the read-write sets increases. Their results showed that the throughput of the system is linear for reads. For writes, it is nearly linear below a transaction rate of 1000 transactions per second.

A benchmark tool called as Hyperledger Caliper [32] has been proposed by the Hyperledger foundation to evaluate the performance and throughput of different Blockchain platforms. Our study includes the BigchainDB platform which Hyperledger Caliper doesn't support for performance benchmark. Two studies, by Nasir et al. [86] and Ampel et al. [15], have evaluated the performance of Hyperledger Fabric and Hyperledger Sawtooth using the Hyperledger Caliper. Nevertheless, neither study considers the integration of Blockchain platforms in IoT applications.

5.5 Background

5.5.1 Significance of Blockchain in IoT

The Blockchain methodology is a bridge to settle privacy and reliability concerns in the Internet of Things (IoT). Decentralizing Blockchain eliminates a single point of failure and creates a more resilient ecosystem for the smart devices to run on. More importantly, the cryptographic algorithms in the Blockchain technology would make the consumer data secure and private. In the IoT applications, Blockchain can keep an unalterable record of all smart devices; this feature authorizes the autonomous function of smart devices without the need for a centralized authority [129].

Blockchain can ensure three main security and privacy issues in IoT application namely Availability, Confidentiality, and Integrity [130]. Availability ensures that each service or information is only available upon the user's request. Integrity implies that each message has been received at the destination without any changes. Confidentiality ensures that only the authorized user can access the data, while reducing the risk for Linking Attacks and Distributed Denial of Service Attacks [16].

Transitioning from a centralized to a decentralized system causes transparency to be increased by eliminating any authoritative parties in the transaction, while integrity is ensured through automatic and peer-to-peer block validation. However, this validation is based on the consensus characteristic in the Blockchain, which increases replication and interconnectivity in the network. For these reasons, computational power and time overheads for the peers may as the number of the peers also increases. As a result, the time required to validate and confirm the transactions and replicate them into the network also increases [27].

5.5.2 Consensus Algorithm in Blockchain

The philosophy of the consensus algorithm is inherited from the notion of the Byzantine Generals (BG) Problem among untrustworthy nodes proposed by Lamport et al. [43]. The problem arose from the early days of Blockchain, when it was used to mine and trade Bitcoins. In this context, users had two needs; first, avoid the reuse of the same currency in two transactions at the same time (double spending) and, second, verify the transactions by multiple distributed nodes through a P2P protocol. However some nodes could be malicious and try to alter communication contents. The nodes in Blockchain network need to distinguish between trusted and malicious nodes, for which purpose they use a consensus algorithms [45]. In the Blockchain network, all the peers in the distributed ledgers have to agree on acceptance by majority to verify a node. This mechanism guarantees that the output is the result of the

eventual agreement on the change of state and all the data of the networks have been shared across multiple replicas. This mechanism has the three following main properties:

- Safety: The consensus protocol ensures the safety of all the peers to produce the same output and the output produces by the peers are valid and follows the protocol rule. This can be also considered as consistency of the shared state.
- Availability: The consensus protocol ensures the availability of the system even in the presence of faulty nodes.
- Fault Tolerance: The consensus protocol provides fault tolerance by recovering a node if it has failed.

There are several algorithms that were proposed to cope with several issues in the peer to peer (P2P) networks such as failure of nodes, partitioning of the networks, message delays, corrupted messages and dealing with selfish and deliberately malicious nodes [46]. The first consensus algorithm in Blockchain ,which was introduced in Bitcoin, is called Proof of Work. PoW is a computational challenge or mathematical puzzle-solving approach to validate the transactions in the Blockchain network [48]. PoW has very high latency, which is not desirable for real time and time sensitive IoT applications [53]. To overcome high computational and low response time in PoW, researchers have completely eliminated this consensus algorithm [16] to overcome the issue of overheads and performance, but consensus is the main characteristics of the Blockchain and by removing consensus, the system will be vulnerable to other types of attack such double spending and forgery attacks [54]. To overcome some of these shortcomings of PoW, several other consensus protocols have been proposed. Next, we describe some of these alternative consensus algorithms.

Proof of Stake

Proof of Stake (POS) was proposed to solve the high energy consumption which is caused by PoW. To do so, PoS uses pseudo-random selection of stakeholders to append to the Blockchain system. In PoS, each Blockchain branch will be selected uniformly and randomly from the all participants in the network. A cryptographic signature verification scheme guarantees that only the owner of the selected transaction will have the option of appending to the block [55].

Proof of Elapsed time

Proof of Elapsed time (PoET) is an alternative option for Consensus algorithm which uses a random leader election model where the algorithm randomly chooses the next leader to finalize

the block. The random leader uses this protocol to deal with malicious nodes and open-ended participation of the nodes in the Blockchain network. In order to work correctly, it has to select the leader among all the nodes of the network from a uniform probability distribution and it requires a secure way for the nodes to verify that a given leader was correctly elected without any alterations. This will be achieved by Trusted Execution Environment (TEE) to guarantee the reliability and randomness of electing a leader [48].

5.5.3 Tendermint Byzantine Fault Tolerance

Tendermint is variation of Byzantine Fault Tolerance (BFT) algorithm which provides state machine replication using hash-linked branches of transactions in the Blockchain network. In this consensus algorithm, each block is committed by a known set of weighted validators. Membership and weighting of this validator set might change over time. This consensus algorithm guarantees the safety and liveness of the Blockchain network so long as less than 1/3 of the total weight of the validator set is malicious or fault [57].

5.5.4 Raft

Raft [131] is Crash Fault Tolerant(CFT) [132] ordering services which follows "leader and follower" model. In this model, a leader node will be elected per channel and its decision will be replicated to its followers (CFT) can tolerate up to $N/2$ system failures and it it does not guarantee on adversary nodes. [68].

5.5.5 Hyperledger

Hyperledger is an ecosystem and an umbrella project consisting of different open source Blockchain platforms that are hosted by the Linux Foundation [60]. It is a collaborative project to host different applications such as financing, banking Internet of Things, supply chains, and others, and ensure transparency, immutability, longevity, and interoperability [61].

There are different Hyperledger platforms, such as Burrow [62],Fabric [13], BESU [63], INDY [64], IROHA [65] and Sawtooth [56] and each of those platforms could be useful for specific applications such as financing or supply chains. Next, we provide a description for some of them that we will use in our study.

5.5.6 Hyperledger Fabric

Hyperledger Fabric is an open-source Blockchain platform which was proposed by IBM. The purpose of this platform is to overcome some limitations of other Blockchain platforms such as Ethereum [49] and Tendermint [66]. In particular, order-executive or hard-coded consensus are two shortcomings that affect the throughput and latency in other Blockchain platforms [13]. The consensus algorithm in Hyperledger Fabric relies on Crash Fault Tolerant (CFT) which is based on Raft protocol. [67, 68].

5.5.7 Hyperledger Burrow

Hyperledger Burrow is another project which is Hosted by the Linux Foundation and was introduced by Monax [69] and Intel [70]. It provides a modular Blockchain client with a Permissioned Smart Contract Interpreter (PSCI). It consists of three main features including Ethereum Virtual Machine, the consensus engine and remote procedure called gateway. The main difference of Burrow with other platforms, like Fabric, is that its main focus is on running on Ethereum Virtual Machine(EVM) and on using smart contracts in a permissioned network. It uses Proof of Stake (POS) consensus in which participants in the network can validate transactions based on the held coins or power. PoS is used to overcome the computational power which exists in Proof of Work. Burrow provides a tool to implement smart contracts that are written in Solidity [71] and it can formulate all the transactions on the chain network [72].

5.5.8 Hyperledger Sawtooth

Hyperledger Sawtooth [56] is an open-source Blockchain platform which is designed especially for supplying chain management. It is part of the Linux Foundation umbrella project developed by Intel. The main difference of Sawtooth with other Blockchain platforms is that the transactions and data can be executed in parallel instead of in series, which will result in better performance of the system. Sawtooth supports both Practical Fault Tolerance [67] (PBFT) and PoET [19]. A feature should be chosen efficiently between consensus algorithms, permissioning, and transaction rules according to the user's business requirements.

5.5.9 BigchainDB

BigchainDB is a decentralized distributed database with a high-performance throughput [36], it is designed to advocate Blockchain characteristics and inherit NoSQL distributed databases properties, such as linear scaling throughput and efficient querying. As described in its

characteristics, it is developed to solve some limitations of Blockchain such as low latency, high computational power, and low scalability. The consensus algorithm in BigchainDB is Tendermint which is derived from BFT. BigchainDB is not part of Hyperledger ecosystem however the reason of choosing this platform is that it is currently very popular as a lightweight Blockchain network [76].

It is noteworthy that a new platform called IOTA as an alternative of Blockchain which uses Directed Acyclic Graph (DAG)has been designed for exchanging data and service among IoT devices. The reason that we did not consider this platform is that firstly, IoTA is not using Blockchain and also our study only focuses on evaluating writing data from IoT devices to the Blockchain platform. [133]

5.6 Comparative Study: Methodology and Setup

In the context of our comparison study, our intention is to investigate the performance of Blockchain technologies (in terms of resource consumption and latency) in conjunction with IoT applications. For this reason, we have simulated a Smart Building environment, in which we have evaluated the four Blockchain installations. In this section, we describe the environment of our experiments, the synthesis of the workloads, and the configuration of the Blockchain platforms.

5.7 Application Domain: Overview of the architecture of a Smart Building

A Smart Building refers to an instrumented and interconnected environment, through sensors and other equipment, with the goal to improve several factors for the building occupants, including productivity, comfort, safety and others. In this context, we can talk about either residential buildings (most commonly referred to as Smart Homes) or commercial and office spaces. The use of IoT technologies in smart buildings has greatly improved building automation and management systems with the use of different devices such as lighting, ventilation, smart doors, surveillance cameras and others.

5.8 Experimental Setup

In our study, we are simulating a smart building network with three buildings, each containing several IoT devices, managed by different administrators, including surveillance cameras, thermostats, door sensors, GPS sensors and sound sensors. Data flow from the sensors and devices of each building to a gateway and the gateway pushes the data to a central

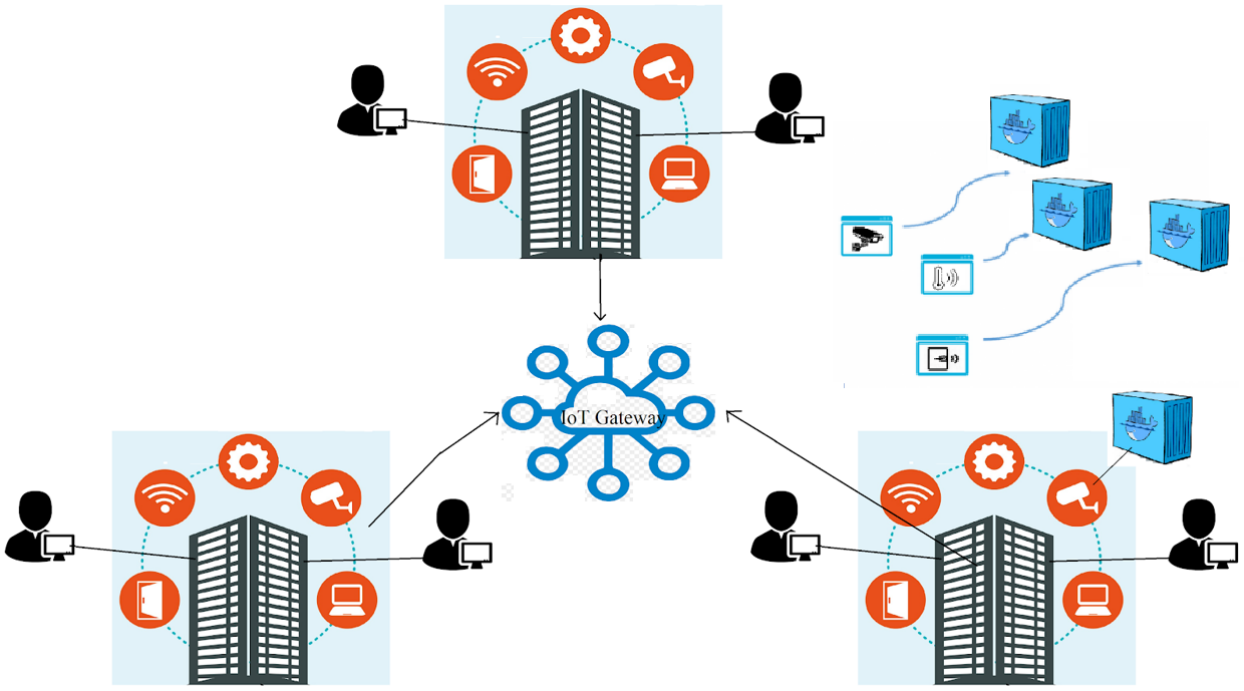


Figure 5.1 Typical IoT architecture for Smart Building applications.

database (Figure 5.1). For our experiments, the final database is represented by one of the four Blockchain installations or the MongoDB NoSQL database. To help simulate the generation of data from multiple sources simultaneously, we use HA-Proxy [134], while Nginx and Flask are used to simulate the IoT gateway. Docker containers are used to simulate the sensors and devices, which will produce the data. Each type of IoT device is implemented as a distinct Docker image to represent different data types and emission rates. More specifically, we have GPS sensors that send a position report every 2 seconds, sound detectors that report an ambient sound also every 2 seconds, thermostats that report the temperature every 1 second and door sensors that reports the state of a door (OPEN or CLOSED) every 5 seconds. Data sizes range from a few bytes to a few kilobytes, depending on the type of data.

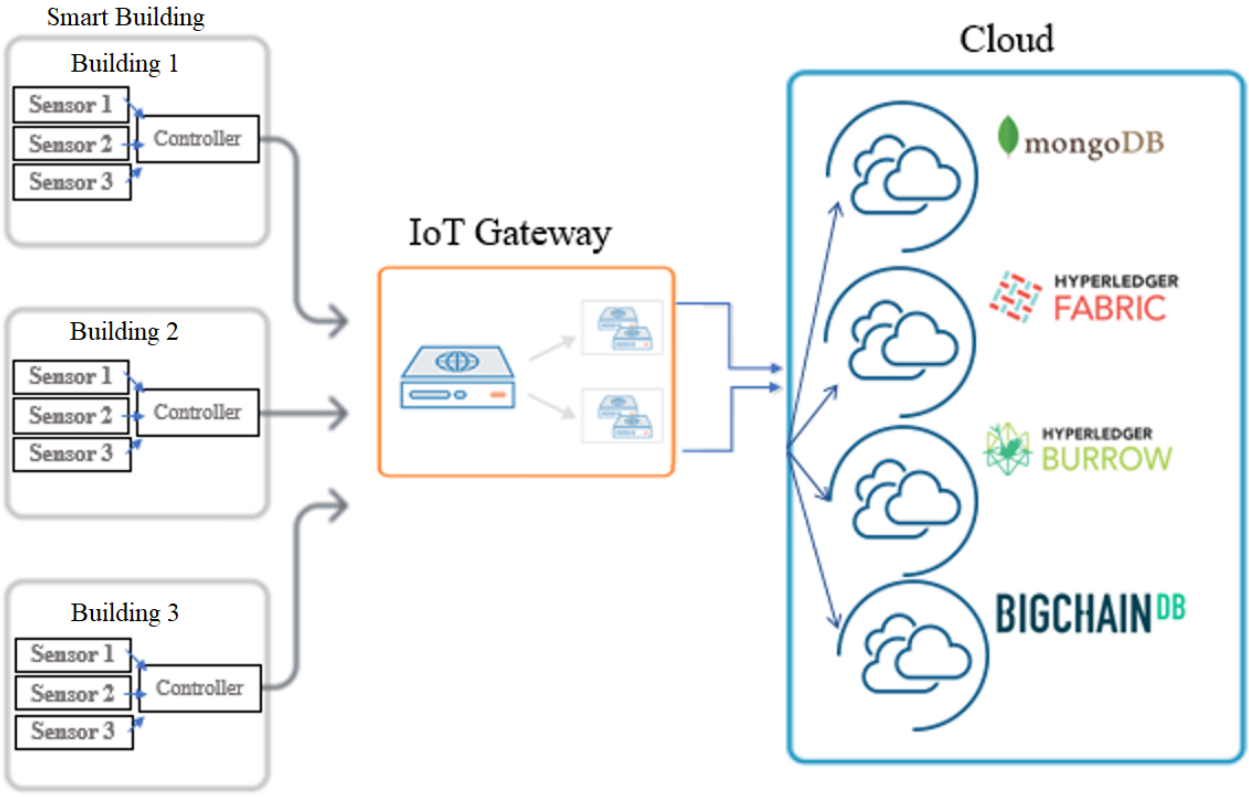


Figure 5.2 Components and data flow for the comparative study

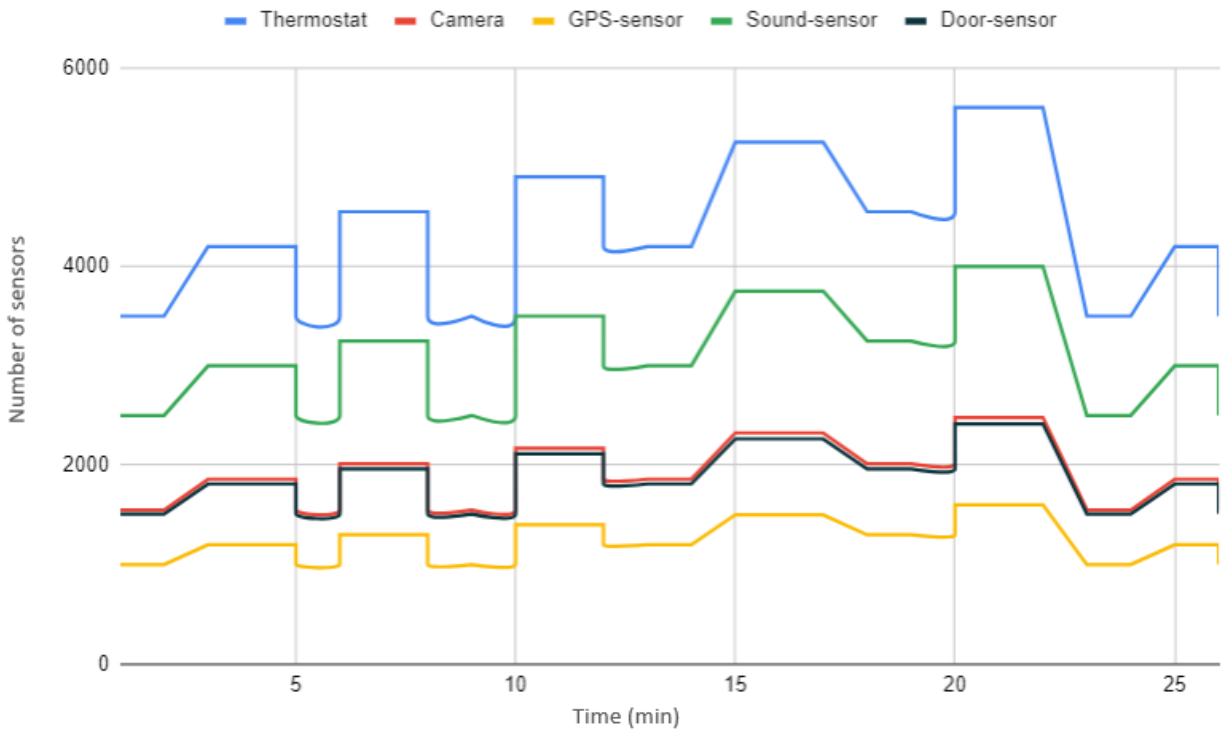


Figure 5.3 The distribution of IoT sensors over the time

Every experiment has a duration of approximately 26 minutes. We have repeated the experiment for each platform 10 times to make sure that the comparison result would be constant. Within this time, we aim for a particular number of sensors at specific time points, starting from 10,000 sensors at the beginning of the experiment and going up to 16,000 sensors. the information that we store in either Blockchain or MongoDB are: Device ID, Device Type, Time Stamp, Sequence Number, Data, Size of the Data, Hash of the Data. The distribution of sensors and devices is kept relatively stable throughout the experiments: 15% cameras, 35% thermostats, 10% GPS, 15% Door-sensor devices and 25% sound devices. Each devices is represents one container in our system. The devices first sends their data to the IoT gateway as presented in Figure 5.2 and then they transfer the data to the cloud which we have one baseline as MongoDB and three different Blockchain platforms. In our code we have considered to include the public key in each container to identify and authenticate which device is going to store the data in the Blockchain network. For the Real case scenario, we might need to off-chain only hash on the Blockchain for immutability but as the purpose of this study is to measure the performance of Blockchain, we have decided to saturate the system as much as possible to see which one outperforms. Figure 5.3 shows the distribution and the total number of sensors for every experiment over time. The reason that the distribution of IoT devices remain in similar values is to follow ceteris paribus to see the one possible configuration for different platforms. IoT data in our simulation will be sent using JASON in a python code to be stored in the Blockchain platform. All these data will first go to to a load balance as an IoT gateway and then will be sent to the Blockchain. As demonstrated in Fig 5.2 , after the data are sent to the gateway, the gateway is responsible for sending data to the Blockchain network. In total we have used 15 containers for camera, 35 containers for thermostat, 10 containers for GPS and 10 containers for door-sensor. The simulated devices were designed according to the guidelines set forth by Ramprasad et al [135], which recommend that simulated IoT devices should have the following characteristics:

1. **Connectivity:** All devices should be consistently online.
2. **Configurability:** Emission rates, size of message and number of sensors in the network should be configurable.
3. **Deployability:** The simulation environment should be able to be instantiated rapidly based on container technology.

Finally, the database part of the architecture (Blockchain or MongoDB) was deployed on three virtual machines on Amazon EC2. Each VM had 4 GB RAM and 2 VCPU cores.

Docker was first installed in these VMs and containers of the databases were deployed on the VMs. Monitoring data for the deployed databases was gathered through the Docker Remote API [136]. We measured CPU consumption and memory utilisation. We have additionally created a customized monitor to measure the latency between sending data to the database and the verification that the data was successfully stored. We have implemented all of our Blockchain platforms, including Hyperledger Sawtooth, Hyperledger Burrow , Hyperledger Fabric and BigchainDB using their default setting and latest update (September 2019) which is available in their GitHub repository. The consensus algorithm which was used for Hyperledger Fabric is Raft, for Hyperledger Sawtooth is PoET, for BigchainDB is Tendermint BFT and for Hyperledger Burrow is PoS. It is noteworthy to mention that in the default configuration of Hyperledger Sawtooth, there is an ability to support parallel transactions where as the other Blockchain platforms do not support this feature.

5.9 Comparative Study: Results

Figure 5.4 demonstrates the average response time of the four Blockchain platforms and MongoDB as the baseline. As can be seen, there is a significant difference between MongoDB and the Blockchain platforms. The writing latency is about 1000 ms higher than the response time indicating that Blockchain in general is far slower.

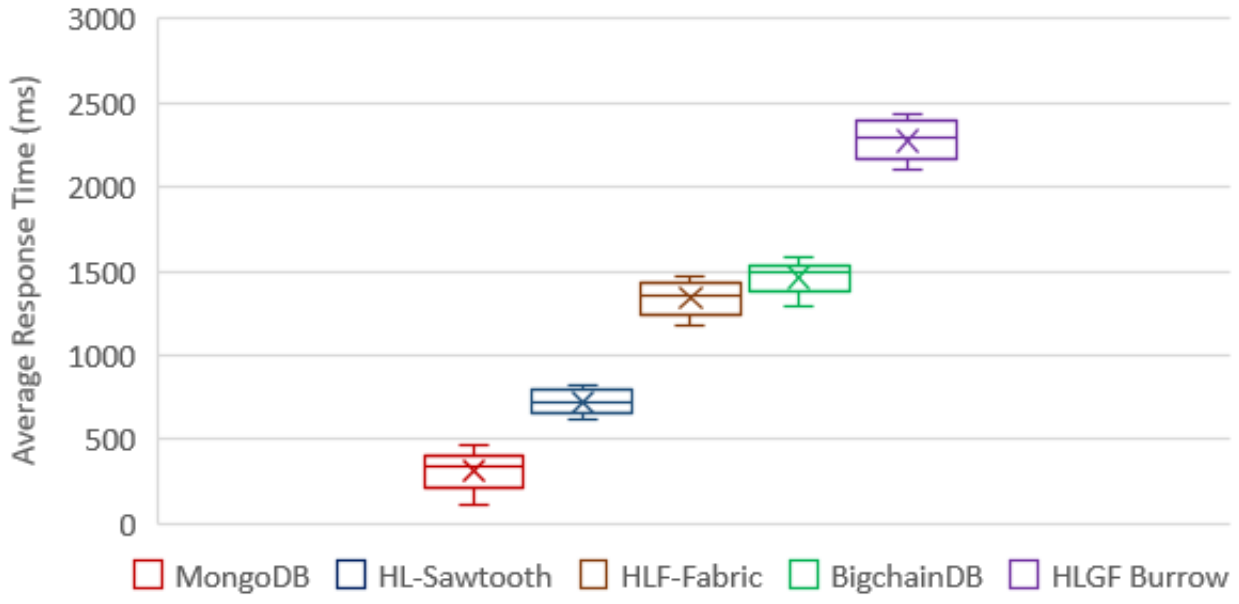


Figure 5.4 The result of experiment for Average Response Time for Writing

On the other hand, the worst scenario among the Blockchain platforms was for Hyperledger Burrow in which the median average response time to write is 2500 ms, and it seems that the average response time for Hyperledger Sawtooth is significantly better than both Hyperledger Fabric and BigchainDB The response times of both Hyperledger Fabric and BigchainDB are almost the same around 1300 ms, with the one of Hyperledger Fabric being slightly better. Our results can be justified on the grounds of two main differences between Hyperledger Sawtooth and the other platforms. It seems that Hyperledger Sawtooth which with PoET consensus protocol is a much more efficient consensus algorithms with respect to transaction validation time. Moreover, the transaction processing in fabric is based on endorsing peers and ordering services whereas in Sawtooth, it is based on validators which seems to be less complicated and less overheads for transaction processing.

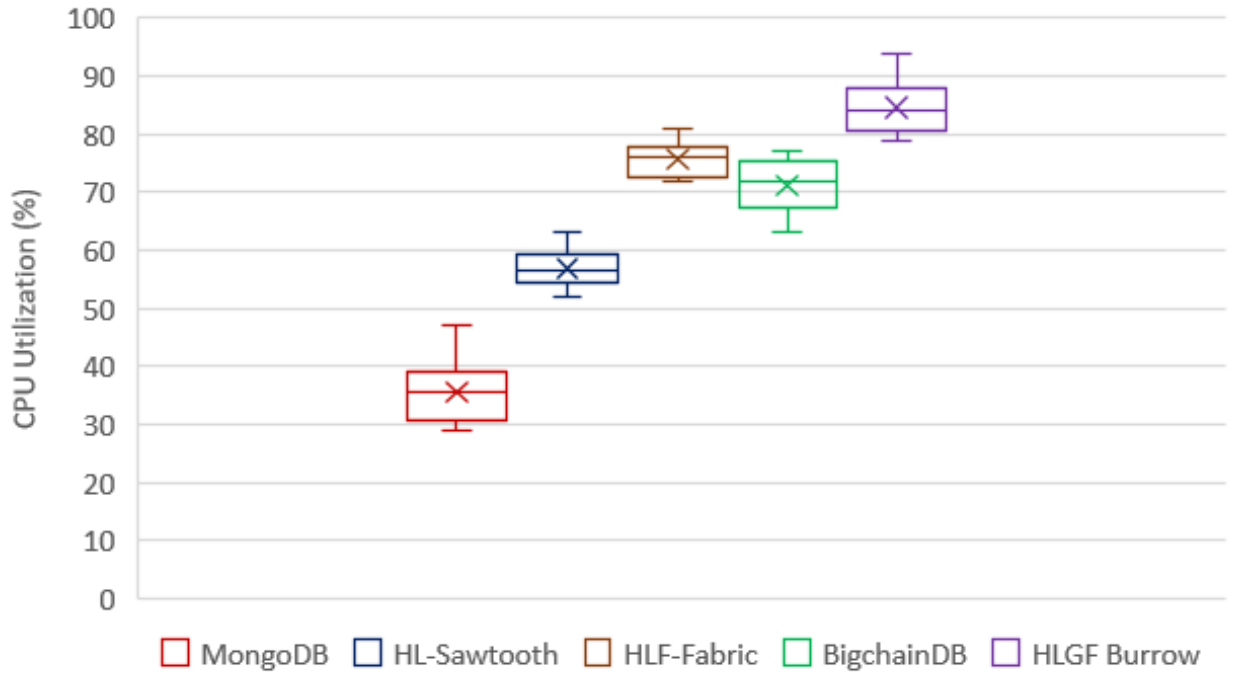


Figure 5.5 The Result of Experiment for CPU Utilization

Figures 5.5 and 5.6 demonstrate the resource utilization for the 5 tested platforms. In the context of this study, we monitor both CPU utilization and memory usage as a proxy for energy consumption. The premise is that the more resources are used by the system, the higher the energy consumption will be. This in turn can result in higher costs and a higher environmental impact, especially when we talk about large data centers. As it can be seen Hyperledger Burrow has the worst resource utilization and MongoDB as a baseline has the best resource utilization, while CPU utilization for both BigchainDB and Hyperledger Sawtooth seems to be

slightly better than Hyperledger Fabric and the worst case here is again Hyperledger Burrow. Hyperledger Burrow uses Proof of Stake and this imposes higher resource consumption and

Figure 5.6 The Result of Experiment for Memory Usage

time overheads comparably to the other platforms. Both Hyperledger Fabric and BigchainDB use Raft and BFT respectively and Hyperledger Sawtooth in our study is configured to use Proof of Elapsed Time. It seems that Raft, BFT and PoET are more efficient in terms of resource consumption and time overheads than PoS. Similar results to CPU can also be drawn for memory utilisation Figure 5.7 demonstrates the impact of the size of the network (for our study, this corresponds to the number of IoT devices) to the average response time for our 5 platforms. Once again as can be seen Hyperledger Burrow has been the worst alternative in terms of average response time, while Sawtooth has been the best among the Blockchain platforms. Even better, Sawtooth avoids some oscillation which is present even in MongoDB, when the size of the network changes abruptly. Another interesting observation is that despite being a more lightweight alternative that tries to avoid the general shortcomings of Blockchain, BigchainDB can be considered an average solution, with a clear saturation point when the network hits its greatest size.

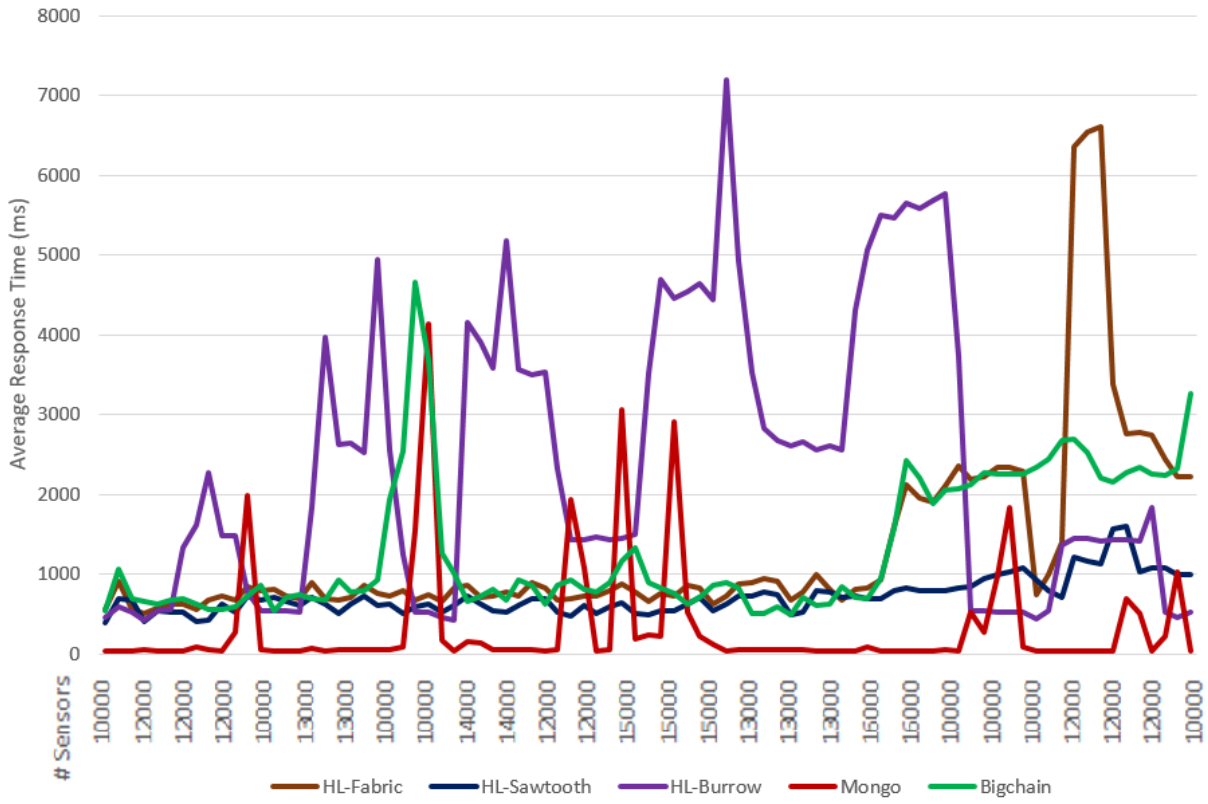


Figure 5.7 The increase and decrease of Average Response time in 26 minutes when the IoT devices are increasing

5.10 Threats To Validity

As our study is based on the simulation of virtual IoT devices in a smart building with private Blockchain, normally, all the stored data generated by IoT devices would be replicated to cluster nodes, we planned to use the real IoT devices for our future studies to mitigate this issue. A second threat could be the number and type of platforms that we have selected. Among several Blockchain platforms proposed with different consensus algorithms, we have picked four of them because there are various studies being conducted on these platforms and researchers have used these for the integration of IoT and Blockchain. We plan to further expand our study and strengthen any general conclusions, especially for what concerns the consensus algorithms. In our study, we chose 4 different distributed ledger technologies with different consensus algorithms, however it is important to note that there are also other important aspects of a Blockchain that are all CPU/memory intensive such as complex cryptography, exchange of messages across peers, and smart contract execution and validation of results. In future studies we plan to study what is the role of other aspects in Blockchain

that affects to performance degradation.

5.11 Conclusion

Generally, it is accepted that Blockchain can be inefficient both in terms of time and cost due to high bandwidth overhead and delays. This study aimed to provide a detailed review of the IoT enabled Blockchain in a simulated smart building environment to monitor and securely store data from IoT devices to the Blockchain, which has been a missing link in the previous studies. The work has been accomplished by evaluating four different Blockchain platforms to see whether there is a single best with respect to time and computation overheads or if there are trade-offs between the four platforms for IoT applications. Our results show that Hyperledger Sawtooth seems to be the best in terms of resource utilization and its performance is comparable with the best options of Hyperledger Fabric and BigchainDB and the Hyperledger Burrow is by far the most expensive Blockchain solution. According to our results, it seems that the Hyperledger Sawtooth has a better performance as compared to other platforms.This is because of its ability to have transactions in parallel using validators , less transactions processing complexity and the use of the Proof of Elapsed time consensus algorithm.

CHAPTER 6 ARTICLE 3: PUBLIC OR PRIVATE? A TECHNO-ECONOMIC ANALYSIS OF BLOCKCHAIN

6.1 Paper Information

Title

Public or Private? A Techno-Economic Analysis of Blockchain

Authors

Mohammadreza Rasolroveicy, Wejdene Haouari,and Marios Fokaefs

Date of submission: 2021/06/01

Published in: CASCONxEVOKE 2021

Chapter Overview: This Chapter intends to demonstrate the performance differences between the Ethereum PoW public blockchain when it is implemented on private infrastructure to other private blockchain technologies that are designed for private usage only such as Hyperledger Fabric with Raft algorithm and Ethereum with PoA algorithm. The implementation of Ethereum PoW in private enables us to obtain and study the resource utilization metrics such as CPU, memory, and network usage of the platform and the consensus algorithm as we don't have access to these metrics on public blockchains. Knowing the resource utilization in public blockchains can help us to understand what is the rationale behind the transaction fees on public blockchains.

Private blockchains could be an alternative solution for public blockchains when we need more control, but it is unclear whether it is more performant or cheaper. This study aims at answering the question: "If both private and public are available, which option is better for what kind of workloads? Can we separate our workloads based on the performance and cost-efficiency of private and public blockchains?

Our study showed that public Blockchain can handle large transactions with low arrival rate more efficiently, while private Blockchain is performant and cost-efficient for larger workloads of small-sized transactions.

The intention of this study is to demonstrate what are the differences between public and private blockchain platforms in terms of costs, resource utilization, throughput, and response time. The result of this study guides prospective blockchain developers and practitioners to decide on appropriate blockchain technology for their decentralized applications.

This comparative study covers two of the five dimensions of blockchain performance we explore

in this thesis, namely public vs private blockchains, and consensus.

6.2 Abstract

Blockchain technology has been rapidly gaining momentum among practitioners to increase security, efficiency, and cost in shared distributed platforms. It introduced a new approach to storing, sharing, and managing data. However, the costs and performance of Blockchain are still challenging and create adoption barriers. Blockchain technology is available either as a public or a private platform. Public Blockchain may reduce capital costs, while private Blockchain offers better control over costs. This work aims to study and quantify the differences between private and public Blockchain solutions in terms of resource utilization, performance, and cost. Our study showed that public Blockchain can handle large transactions with low arrival rates more efficiently, while private Blockchain is performant and cost-efficient for larger workloads of small-sized transactions.

6.3 Introduction

Blockchain is gaining considerable popularity among various stakeholders as one of the most disruptive technologies in the modern era due to its immutability and integrity of networks and data. Various organizations in various sectors are gradually migrating to this novel technology, facilitating trade and other transactions in a secure and trusted manner in an otherwise insecure scenario. The consensus protocol in Blockchain is responsible for ensuring that the data can neither be altered nor removed once validated by the miners. The main popular use case of the Blockchain is mining cryptocurrency on public platforms such as Bitcoin and Ethereum. However, Blockchain can also be used on private infrastructure with restricted members, known also as *private* or *permissioned* Blockchain, which offers even higher restrictions on access control and higher data confidentiality [137].

Taking advantage of Blockchain has proven [8] to be a reasonable solution to mitigate security concerns including Confidentiality, Integrity, and Availability, also known as the *CIA* triad model by introducing a distributed database based on peer-to-peer (P2P) networks [138]. In Blockchain technologies, each data block submitted to the network will be verified by the network peers, and then it will be broadcast to all members. Moreover, Blockchain leaders only have permission to append a block to the network, and these leaders are chosen by a consensus mechanism [7]. Unlike traditional distributed databases, Blockchain only provides read/write access to the data, and when data has been stored, it can neither be altered nor erased. This technique can effectively prevent attacks, like the Mirai botnet attack [139],

because of non-repudiation; once malicious activities are stored and verified, their presence cannot be denied by the actor or any other party. As a consequence, all activities can be easily traced [8].

Although Blockchain is gaining popularity among different industries, adopting Blockchain for real-time applications is not trivial. It imposes several important challenges such as high energy consumption, resource utilization, prolonged delay for transaction confirmations, and low solubility stemming from the consensus protocol [16]. More specifically, due to the consensus protocol, it may take even more time for each transaction to be validated and included in the network, which may cause the entire platform to suffer from reduced performance and quality in general, even if higher security is guaranteed [4, 140].

Performance, but also cost management and other security concerns, differ between private and public Blockchains. In the private Blockchain, the whole network is controlled by parties that may or may not trust each other. The data access in the permissioned network is regulated by the network peers. Since there are fewer participants in the private peer-to-peer network, it is expected to reach consensus relatively faster, so performance may be better. However, the owners of the network are burdened by capital costs to support the infrastructure directly. Privacy, ownership, and persistence are known benefits of private infrastructure [4]. On the other hand, public Blockchain is fully decentralized and there is no specific organization or institution to control the transaction and participants; everyone can join and can become a miner in the network. The most popular public Blockchains are Ethereum and Bitcoin. In public Blockchain, a transaction fee is paid for each transaction submitted to the network [4]. The most popular consensus protocol in public Blockchain is Proof of Work (PoW), which requires significant computation power to validate a transaction, but this power is usually distributed among numerous miners, and it does not burden the submitter or a single peer in the network.

In this study, we aim to establish quantitatively the difference between private and public Blockchain in terms of performance (response time and resource utilization), and associated costs to support the network. More specifically, we have designed a study to conduct endurance tests on a private Blockchain (Hyperledger Fabric) and a simulated, i.e., deployed on private resources, public Blockchain (Ethereum). In the context of the study, we also experiment with different consensus protocols: Ethereum with Proof of Authority (PoA) [141], Ethereum with Proof of Work (PoW) [28] and Hyperledger Fabric with Raft consensus algorithm [68]. Our study evaluates and compares the subject platforms based on performance, in terms of CPU usage, memory consumption, throughput and response time. Finally, we compare the platforms in terms of costs by associating resource consumption with infrastructure costs and

parameters specific to the public Blockchain, like transaction fees and the exchange rate for cryptocurrency. The ultimate objective of this study is to lead to potential decision support to help practitioners choose the most appropriate platform, between public and private, to deploy their data.

The rest of the paper is organized as follows. In Sections 6.4 we outline the related literature. In Section 6.5, we present the methodology and setup for our study. In Section 6.10 we discuss the results of our experimental study and the evaluation of the studied Blockchain platforms. In Section 9.14, we discuss the threats to validity of our study and finally, Section 6.15 concludes our work.

6.4 Related work

Dinh et al. [5] proposed Blockbench, a tool suite which allows for benchmarking Blockchain platforms in terms of latency and throughput. In their study, the authors have considered three platforms including Fabric, Parity and Ethereum. Since the publication of the study results, both Ethereum and Hyperledger Fabric platforms have been changed significantly. More importantly, the current version of Fabric is 2.2 is based on the Raft consensus protocol, while the version used in that study was using the Practical Byzantine Fault Tolerance (PBFT) [59] consensus protocol. One drawback of that tool is that it does not allow for performance evaluation under continuous workloads, i.e., endurance testing. Instead, the tool allows for measuring latency and throughput for fixed transaction rates. Finally, the tool does not offer resource utilization monitoring.

Malik et al. [142] have proposed a Blockchain-based virtual smart grid architecture that uses smart contracts for market payment and related functions. The authors have implemented the proposed architecture by using Ethereum and Hyperledger Fabric platforms. The work conducts a performance comparison between both platforms and presents a virtual smart grid, first on public Ethereum Blockchain and later on permissioned Hyperledger Fabric v1.2. To experiment, they have only simulated the Hyperledger Fabric in private and used the actual Ethereum platform in public. By consequence, This means that they did not have access to resource utilization or the bandwidth of the network for the public platform. Their experimental results demonstrate that Hyperledger Fabric is faster with higher throughput than Ethereum, but through sacrificing the decentralization. One limitation of this study was that the conditions of experimentation for the two platforms were not the same, and thus we cannot be sure of the reasons why the platforms differ. For example, Ethereum consists of hundred thousand peers and their simulated Hyperledger Fabric contained 5 nodes in total. In our study, we try to eliminate differences like this, in order to concentrate on specific and

controlled differences between the two platforms.

Nasir et .al [86] conducted a performance analysis of two versions of Hyperledger Fabric, v0.6 and v1.0, in terms of execution time, latency, and throughput, using a simple money transfer application (chain code) with a varying workload of up to 10,000 transactions. Moreover, the authors examined the scalability of the two platforms, using up to 20 nodes in each platform. The results demonstrate that Hyperledger Fabric v1.0 consistently outperforms Hyperledger Fabric v0.6. However, Hyperledger Fabric v1.0 performance did not reach the performance levels of current traditional database systems under high workload scenarios.

Dabbagh et al. [110] have conducted an empirical study to evaluate the performance of Hyperledger Fabric and Ethereum. The platforms were compared in terms of four metrics: success rate, average latency, throughput, and resource consumption. By executing 100 transactions, the results showed that Hyperledger Fabric generally surpasses Ethereum's in all four performance metrics. However, this work did not evaluate the scalability of the two platforms in terms of the size of infrastructure or performed spike tests to evaluate the peak performance of each platform under saturation.

Wang et al. [4] have analyzed four mainstream Blockchain systems (Ethereum, Fabric v1.4, Sawtooth v1.0.5, Fisco-Bcos v2.0.0), and conducted a performance comparison using the open-source Blockchain benchmarking tool Hyperledger Caliper [32] designed by the Hyperledger umbrella project [61] to perform cross-sectional studies on Blockchain. The authors use transaction throughput and latency as the primary performance metrics to evaluate the performance of the studied platforms. The authors argue that the performance gap between the public Blockchain and the private Blockchain is significant in favour of the latter and claim that public platforms, like Ethereum, need to focus on improving performance. As for Fabric, one of the most popular private platforms, it was found that its performance is not as good as other emerging private platforms. Compared to our work, Hyperledger Caliper does not allow evaluating Blockchain under a continuous and fluctuating workload to perform endurance tests.

Hao et al. [87] proposed a method to appraise the performance of consensus algorithm in private Blockchain platforms of Ethereum and Hyperledger Fabric V0.4. The authors have attained the performance evaluation results of consensus algorithms with different numbers of transactions through quantitative analysis of latency and throughput. The results indicate that the consensus mechanism induces a performance bottleneck. Furthermore, PBFT consistently outperforms Proof-of-Work (Proof of Work) in latency and throughput under varying workloads.

Aswin et al. [143] qualitatively compared Hyperledger Fabric and Ethereum considering various

factors like architecture, performance, use cases, and ease of use for consensus algorithms. The authors observed that when the number of transactions increases to 10,000, Hyperledger can handle 10 times more transactions per second than Ethereum, and the latency of Ethereum becomes 15 times greater than Hyperledger Fabric. However, Ethereum can manage more simultaneous transactions with similar computational resources.

6.5 Study Methodology and Experimental Setup

The purpose of our study is to compare the performance between Hyperledger Fabric and Ethereum, a private and a public Blockchain, in a controlled environment under a continuous and fluctuating workload. Our study focuses on three different Blockchain configurations, Hyperledger Fabric with the Raft consensus protocol, Ethereum with PoA (Proof of Authority) and Ethereum with PoW (Proof of Work). In this section, we present the architecture and implementation of our experimental environment, including our workload generator, the configuration of the studied platforms and of the deployment infrastructure, and the metrics to evaluate the performance.

6.6 Workload and Experiment Architecture

To evaluate the performance of the three targeted Blockchain configurations, we have designed a custom workload generator which simulates concurrent users, who submit data packets of varying sizes (70-80 kb) to the Blockchain network. The number of concurrent users fluctuates over the time to simulate a real-time system. Our workload generator follows a closed-model approach, meaning that the users are independent of one another, and each user sends the next request only when the Blockchain client has returned a successful/Reject response code. Between a response and the next request, we assume a 10-second think time for each user. In our architecture, we mark the transactions as successful and failed based on the HTTP response code.

Figure 6.1 illustrates the main components of our workload generator. We employed a Nginx [144] web server which acts as a gateway between our workload generator and Blockchain platform, so that users can send data to the Blockchain. HA-Proxy [134] is used as a load balancer deployed in front of the Blockchain. This will allow us to bring down the single Nginx instances while HA-Proxy continues to server the requests from the different concurrent users from multiple containers in our workload generator. In order to connect the user with Blockchain, we have implemented two Blockchain clients, one for Hyperledger Fabric and another for Ethereum which was used for both Ethereum PoA and PoW. Both clients

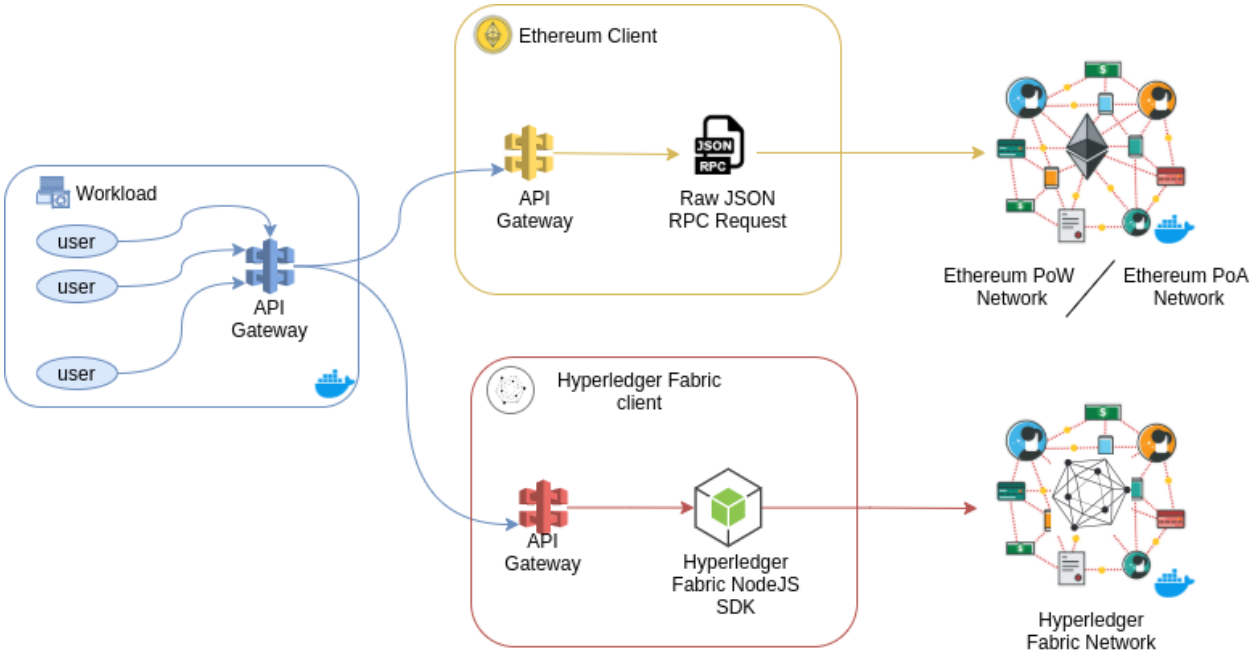


Figure 6.1 Experimental Setup

are implemented using Express framework for JavaScript. For the Hyperledger Blockchain client, we have used the Hyperledger Fabric SDK [1] for Node.js. Finally, for the Ethereum client, we have used raw JSON RPC [145] to send data to the Ethereum Network . In the configuration of the client for the Ethereum, we are able to specify the receiver nodes in our private Ethereum by pointing the address of the miner. In that case one miner will broadcast it to the whole network. We are able to also increase the number of receivers in our client configuration to see how is the effect of sending the transaction to random nodes instead of one.

6.7 Blockchain Network configuration

6.7.1 Ethereum

Ethereum is a popular public Blockchain platform, mainly used to mine the cryptocurrency Ether (ETH), proprietary to the Ether platform. Nevertheless, Ethereum can also be configured as a permissioned/private Blockchain platform. In its public version, Ethereum relies on the computation power of the network peers to perform the validation of the blocks. In contrast, in the permissioned version, the platform is deployed on a dedicated and on-premise infrastructure. In the former case, the platform relies on the extent of the network to build

[1] `https://hyperledger.github.io/fabric-sdk-node/`

trust (the more peers validate the block, the more trustworthy it becomes), but this implies reduced performance. On the contrary, in the permissioned version, the proprietor guarantees security and integrity, so excessive computation power is not required to build trust. Thus, performance is expected to be better.

An Ethereum network can be configured with one of two consensus protocols, Proof of Authority (PoA) or Proof of Work (PoW) [2]. In PoW, miners (as the network peers are called) compete to solve a cryptographic puzzle. The first one to solve the puzzle can mine the block and be rewarded in Ether. Next, the miner broadcasts the block to the other nodes in the network. PoW builds security and integrity because the solution of the puzzle requires significant computation power. A malicious actor that wants to tamper with data, for example, change the data in an already submitted block, would need to change all blocks along the chain of the target block and spend the necessary computation power to do so, which proves to be very costly for the expected benefit.

The other consensus protocol is Proof of Authority which does not require a lot of computational power as a block is created by authorized signers based on their reputation that is specified in the first block, also known as the "genesis block" when the network is created. After the creation, signers can be added or removed using a voting mechanism [111]. The processing of adding block in Ethereum PoA is called "sealing" [112, 113]. In the specific configuration for our experiments, we set all the nodes as valid signers.

For a transaction to be executed, Ethereum uses *Gas*, which is the equivalent of the computational effort required to execute the transaction. When sending a transaction, a user specifies the gas limit (the maximum gas that they are ready to use for that transaction) and the gas fee, which is the gas price per unit of gas denoted in gwei, which is a subunit of Ether[3] [146]. The miners will choose which transaction to mine first, depending on the gas fee. For this reason, it is more likely that a transaction will get into the next block using a higher price. For our experiment, we set the gas limit per transaction to 1,048,576 gas (or 0x100000 in hexadecimal [4]) [146]. For reference, this value is about $2 USD[5].

Besides the user, Ethereum can also impose limitations on gas. In this manner, the blocks also have a block gas limit which defines the maximum amount of gas a block can include. The initial block gas limit is defined in the genesis block, but this value is not constant. As miners use the network, gas limit can increase or decrease by about 0.1% for each new block. For our experiments, we set the initial gas limit to 4,700,000 gas (0×47B760) which is proposed

[2]https://geth.ethereum.org/docs/interface/private-network

[3]1 gwei is equivalent to 0.000000001 ETH

[4]In the configuration of Ethereum gas, corresponding values are usually specified in hexadecimal format.

[5]1 ETH = 1,938.23 USD as of June 21, 2021

by Puppeth CLI wizard [6], a tool created by Ethereum to help setting up the private network . For public Ethereum, the gas limit is around 15,000,000 as of June 2022.

Another critical parameter in a Blockchain network is the block period (or block time) that defines the average time needed to create a new block. For Ethereum Proof of Authority, we can set the block period in the Genesis file using *period* parameter. We set this value to 2 seconds to be similar to the Hyperledger Fabric default configuration. However, For Ethereum Proof of Work, the block period is not configurable, and it's preset to be between 12 seconds and 20 seconds. To maintain the block period to the preset value, Ethereum PoW uses *Difficulty* parameter, which is a scalar value that corresponds to how difficult it is to mine a block in the PoW Blockchain network. If the number of miners increases, the network hash rate [7] increases, and the block period becomes less than the target value. As a consequence, the difficulty will increase until reaching the preset value. Inversely if the number of miners decreases, the difficulty will increase [49, 147]. To sum up, the average block period equals the network difficulty divided by the network hash rate. For our experiments, we set the initial difficulty to 524288 (0×80000) Hashes (H) which is the value proposed by Puppeth CLI wizard.

To be able to fully control the experimental environment and be able to monitor resource consumption on all platforms, we simulated the public Ethereum on private infrastructure. For our simulation, we have used Go Ethereum (Geth) [115] which is an open-source tool that allows us to start Ethereum nodes and interact with the network [148]. In public Ethereum, the gas price is the result of supply and demand for computation power. In our experiments, we monitor the platform's traffic for a few minutes, in which case we can assume that the gas price does not fluctuate much. We set the minimum gas price that a miner can accept to 0 gwei . Geth, by default allows miners to use the total number of processor cores available for mining. Still, we can use the `-miner.threads` parameter to explicitly set the parallel mining thread number, which we have set to 1 for our experiments.

6.7.2 Hyperledger Fabric

Hyperledger Fabric is a private Blockchain platform. It employs chaincodes (specialized smart contracts) [149] that hold defined functions and states. Fabric and Ethereum are built for different purposes. The latter emerged as a platform to mine cryptocurrency, and the former is mainly used as a private ledger, so the terminology for similar entities is different. For example, while Ethereum has miners, Fabric has peer nodes. The nodes are responsible for executing

[6] `https://blog.ethereum.org/2017/04/14/geth-1-6-puppeth-master/`
[7] Hashrate: Number of hashes generated per Second

the chaincode and maintaining the ledger, the entire set of transactions in the network. We distinguish between three types of peers: the *endorser* peer, the *committing* peer and the *orderer* peer. The *endorser* peer is a peer that holds the chaincode, will be responsible of endorsing (approving) a proposed transaction. Then the transaction will be submitted to the Blockchain. The *orderer* peers will then build the block and send it to the *committing* peers that will append the blocks to the ledger. Every *endorsing* peer is a *committing* peer. The *orderer* peer holds the consensus protocol, orders transactions, and controls primary channel access and validation rules. Fabric currently supports the Raft consensus algorithm [149]. Raft [114] is a leader-based algorithm based on Crash Fault Tolerance (CFT). It requires that most participants in the network agree on a transaction process and reach a consensus [150]. In our Fabric configuration, we have used one *orderer* node, one channel and the rest of the nodes are *endorser* peers since *endorser* peer are also *committing* peer.

6.8 Evaluation metric and monitoring

During the experiments, we monitored metrics about latency, resource consumption and some Blockchain parameters. The latency of a transaction, which is the time between sending a transaction from the workload generator to the blockchain client and the generation of the hash of that transaction, which results in either success or failed response code from the client. Latency was measured in the Nginx gateway, since every request and the corresponding response pass through the gateway. In terms of resource consumption, we monitor CPU consumption, memory consumption and network usage by calling `docker stats` [136] each 10sec. Finally, we monitor how the difficulty changes during our experiment using Web3JS [151] that provides an implementation for Ethereum JSON-RPC [152] over HTTP.

6.9 Infrastructure

The experiments were conducted on an Amazon `t2.large` virtual machine with 8GB RAM, 2 vCPUs and 25GB of hard-disk space running Ubuntu 18.04. This VM was used as the host for Docker containers that represented the nodes of our Blockchain networks. Each docker container in our experiment has only 20% of the total CPU and memory of the VM. In our main experiments, 5 nodes were used in each network. An additional experiment was conducted with 8 nodes to validate the scalability of the platforms.

For Ethereum, we used geth (v1.10.3) [115] a Go implementation of Ethereum, while for Hyperledger Fabric we used the release v2.3.2 [34]. The Ethereum nodes run the same solidity smart contract, while the Fabric nodes run a similar Go chaincode.

Finally, each experimental study lasted 20 minutes and the number of concurrent users fluctuates over time to simulate a real-time workload scenario in the sense that it is continuous and fluctuating and not a fixed arrival rate over a fixed period.

6.10 Experimental Results

In the context of our study we conducted 1 main experiment, where we compare the three Blockchain configurations under fixed and static conditions, including size of infrastructure and size and intensity of the workload. We also conducted a number of sensitivity experiments by varying the values of the aforementioned parameters. In this section, we present a comparison of the three Blockchain systems in terms of their performance and we also present a cost analysis to compare the the public and the private configurations.

6.11 Main Experiment

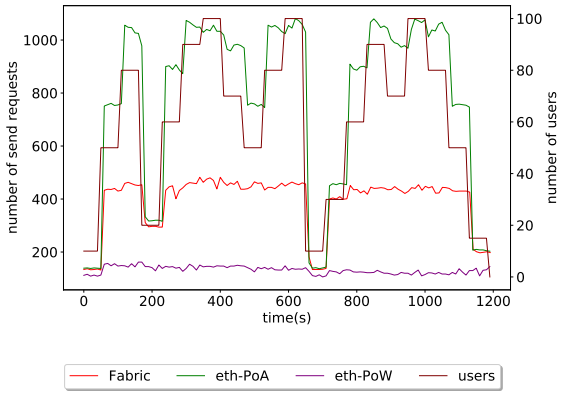
In the main experiment, we deployed all three configurations in an infrastructure with five nodes in the Blockchain network and requests that varied between 70 and 80 KB. For a 20-minute duration for each experiment, the number of users fluctuated between 10 and 100. Due to the difference in response time by each platform, the number of requests processed within the 20 minutes differs, as can also be seen in Figure 6.2 (a). As it can also be seen in Table 6.1, Ethereum-PoA has the highest total throughput because it has the lowest response time, while Ethereum-PoW is at the opposite end of the spectrum. As explained before, this difference is expected and is justified by how the respective consensus algorithms function. The Hyperledger fabric has much worse performance than Ethereum-PoA but much better than Ethereum-PoW in terms of total throughput and response time.

Figure 6.2 (b) shows the average response time for our three platforms. One first observation is a correlation between response time and workload in terms of the number of users in all three cases, which is expected. PoW has so high response time because, for every successful request, the generation of a transaction hash requires higher resource computation than the other algorithms. At the top workload (100 users), PoW reaches a response time of about 9 seconds, while PoA and Fabric have 350ms and 956ms, respectively. Fabric, despite being three times slower than PoA, both Fabric and PoA seem to scale well against variable workload as the response time does not fluctuate as much. Remarkably, the performance of Fabric and PoA is almost equivalent for small workloads.

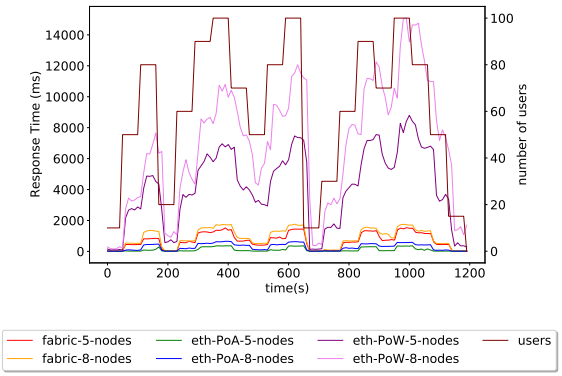
Figure 6.2 (c) shows the average CPU consumption for each platform. One first observation is that PoW has the highest CPU usage of about 35%, mainly explaining how the consensus

Table 6.1 Summary of results of comparison between Hyperledger Fabric, Ethereum PoA and Ethereum PoW

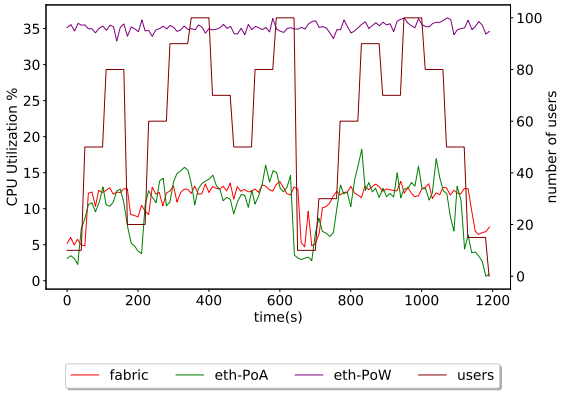
	Algorithms	Fabric	PoA	PoW
	total requests	47151	93939	15866
	St.Dev. requests	105.38	328.86	13.83
5 nodes	Avg CPU	11.33%	17.34%	35.16%
	St.Dev. CPU	2.41	6.97	0.6
1X size	Avg. R.T. (ms)	700.83	113.53	4180.05
	St.Dev. R.T.	498.96	130.18	2676.33
	Avg MEM	1.06%	3.85%	13.18%
	St.Dev. MEM	0.07	0.93	0.61
	Net TX [8] (MB)	232.38	58.21	20.73
	St.Dev. Net TX	140.21	35.90	7.97
	total requests	42119	70553	10696
	St.Dev. requests	85.85	208.19	17.58
8 nodes	Avg CPU	9.56%	11.01%	21.76%
	St.Dev. CPU	1.66	3.79	0.63
1X size	Avg. R.T. (ms)	868.52	283.9	6963.29
	St.Dev. R.T.	604.56	221.11	4041.2
	Avg MEM	1.03%	4.06%	5.6%
	St.Dev. MEM	0.07	0.75	0.12
	Net TX (MB)	369.16	79.79	29.33
	St.Dev. Net TX	222.66	48.78	7.69
	total requests	29119	36795	10568
	St.Dev. requests	52.23	83.19	9.32
5 nodes	Avg CPU	7.43%	4.33%	32.93%
	St.Dev. CPU	1.42	1.49	1.84
4X size	Avg. R.T. (ms)	1675.21	1040.93	6498.02
	St.Dev. R.T.	965.43	582.09	3396.87
	Avg MEM	1.004%	3.09%	12.38%
	St.Dev. MEM	0.05	0.68	0.74
	Net TX (MB)	142.59	28.78	16.22
	St.Dev. Net TX	86.14	16.46	6.86



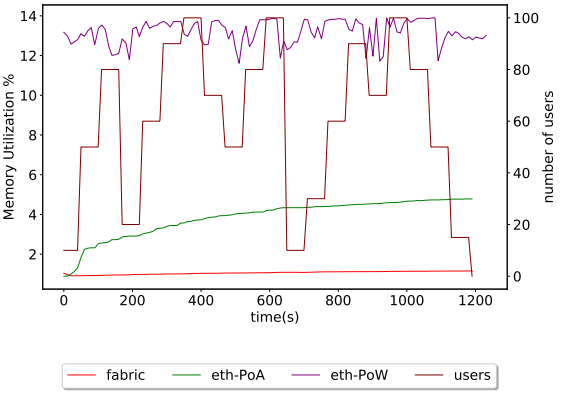
(a) Total Throughput Vs Number of users



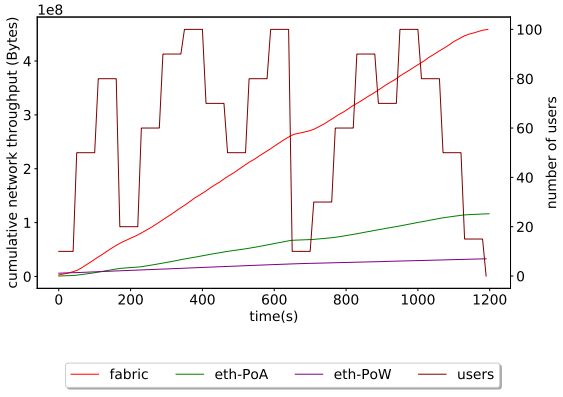
(b) Average Response Time Vs. Number of Users



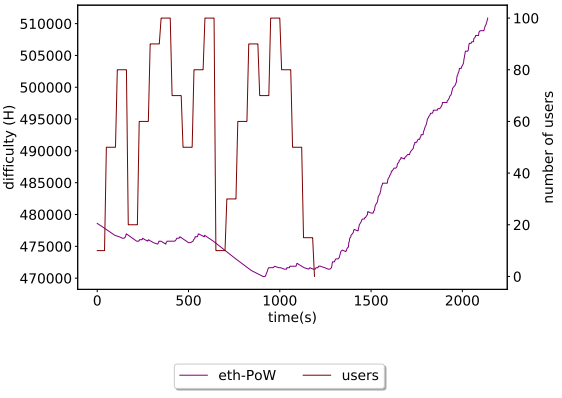
(c) Avg CPU Utilization Vs Number of users



(d) Avg Memory Utilization Vs Number of users



(e) Cumulative Network Throughput Vs Number of users



(f) Ethereum PoW difficulty Vs Number of users

Figure 6.2 Number of send requests, Average Response Time, CPU Utilization, Memory Utilisation and Network throughput using 5 nodes and 1 receiver node

algorithm works. Another observation is that CPU consumption for PoW remains relatively stable regardless of the fluctuations in the workload. This is the result of the changes in difficulty. As shown in Figure 6.2 (f) as long as there is the load to be executed, the difficulty remains low. When the workload ends, the difficulty increases to maintain block period and not allow miners to add extra computation power and compromise the network's security. Concerning PoA and Fabric, we can see that CPU utilization is low and fairly similar for both. The difference is the variance (i.e., the oscillation) of CPU utilization, which is higher for PoA, potentially making Fabric a more stable and manageable solution in resource consumption. As we will discuss next, this difference, even if slight, makes sense when we compare the platforms in terms of cost.

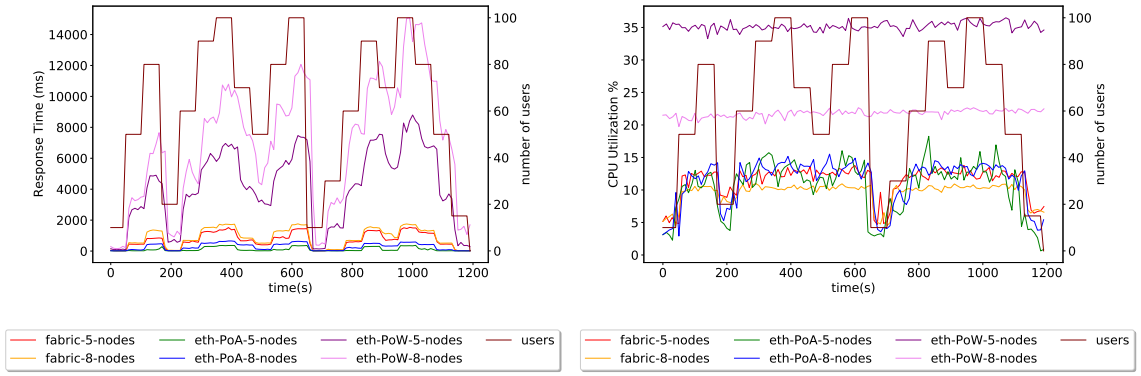
The same finding is corroborated by Figure 6.2 (d) which shows the memory consumption. Besides CPU, Fabric also consumes less memory than PoA and PoW. Another interesting observation is that memory consumption for both PoA and Fabric is independent of the workload, while this is not entirely the case for PoW. In fact, we can see that for PoA, memory consumption seems to have a cumulative behavior continuously increasing for the duration of the experiment.

Figure 6.2 (e) shows the results about the cumulative network throughput in bytes. Conversely to the other resources, Fabric is the platform that consumes the most network throughput out of the three alternatives. The reason once again is how the Raft consensus algorithm is implemented, which requires extensive communication between the peers to validate a transaction. Over time this communication adds up, as can be seen in the figure. The same can be observed for PoA, while PoW shows the lowest network consumption of all. The reason is that PoW depends more on computation power rather than communication between peers to validate transactions.

6.12 Sensitivity Analysis

To investigate the impact of certain configuration parameters, we conducted 3 additional experiments for sensitivity analysis. We first evaluated the scalability of the three platforms in terms of the size of infrastructure (5 vs 8 nodes). Next, we evaluated the scalability in terms of the size of each transaction (70-80 KB (1x) vs 280-320 KB (4x)). The third sensitivity experiment evaluates the impact of receiver nodes in the Ethereum configurations.

Table 6.1, which contains a summary of our experimental results, also contains the evaluation metrics for the sensitivity experiments on the size of infrastructure and the size of data. Figures 6.3a and 6.3b show the difference in response time and CPU consumption respectively



(a) Average Response Time Vs. Number of Users (b) CPU utilization Vs. Number of Users

Figure 6.3 Average Response Time and CPU utilization using 5 nodes and 8 nodes

between the configurations with 5 and 8 nodes. As it can be observed, across all three platforms, both CPU usage and response time increase when the infrastructure increases in terms of nodes. While this may seem counter-intuitive, it makes sense for Blockchain platforms. According to the consensus protocol, the majority of nodes need to validate a transaction so that security and integrity are guaranteed. Consequently, the larger the infrastructure, the more time and power it takes to validate a transaction. In addition, network throughput also increases as more nodes need to communicate with each other. Finally, as response time increases, the total throughput decreases. In summary, while a larger infrastructure may improve security and integrity, its impact on performance is negative.

As far as the size of the transaction is concerned, Figure 6.4 shows that when increasing the size of the transaction, response time also increases for all platforms. While transaction size is multiplied four times, the impact on response time for each platform is different. Average response time increased 2.4 times for Fabric, 9.2 times for PoA, and 1.55 times for PoW. As response time increases, total throughput decreases. Consequently, the network throughput also decreases as fewer communications need to occur between the nodes, and so does CPU consumption as fewer requests need to be processed. In summary, transaction size reduces the capacity of the Blockchain platforms, which also results in reduced resource consumption. Another interesting conclusion is that while Ethereum-PoA is an efficient and fast platform, Hyperledger Fabric may be preferable for larger but fewer transactions.

The last sensitivity experiment involved the number of receiver nodes in Ethereum. As explained before, Ethereum allows our client to define multiple receiver nodes to send transactions. This way, transactions can be shared between the receiver nodes, effectively functioning as load balancing. As it can be seen in Figure 6.5a, when using a single receiver,

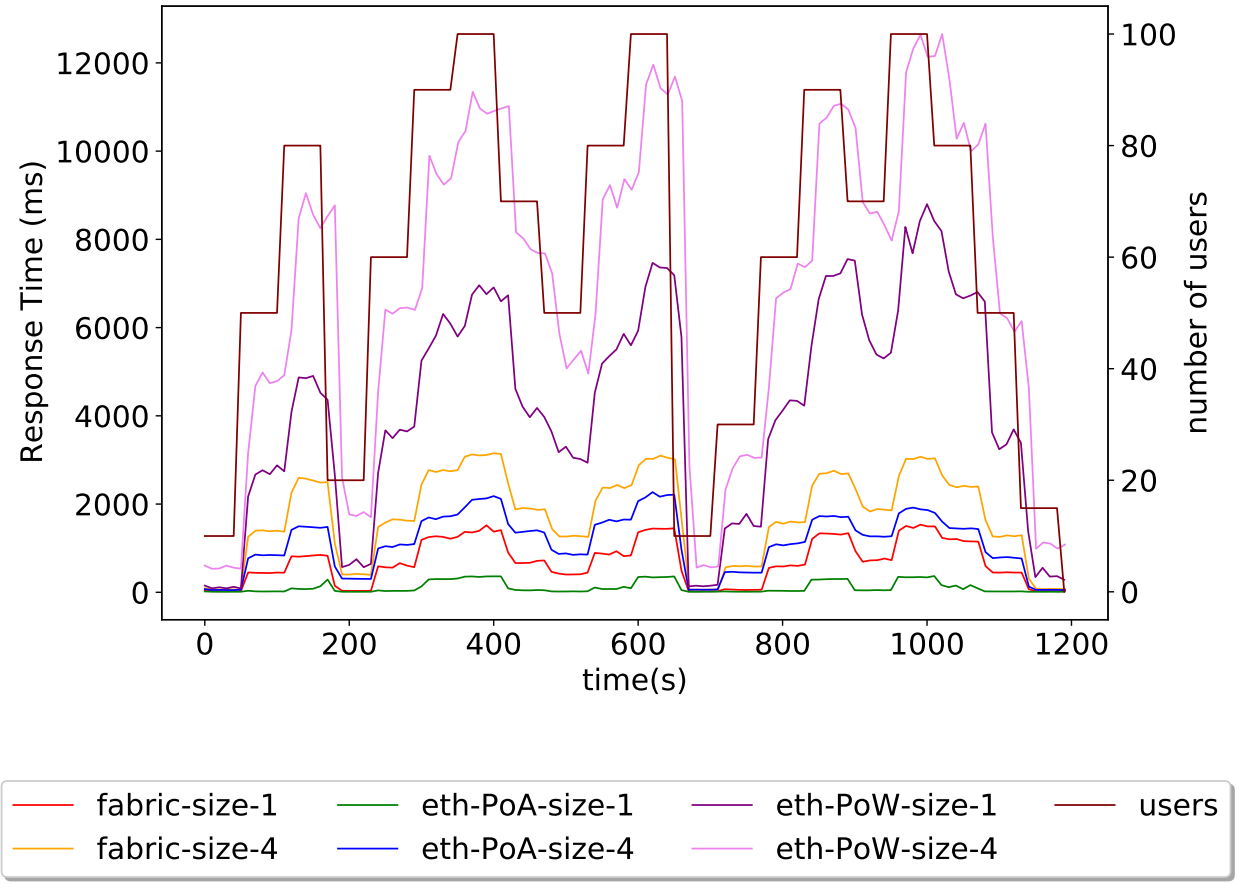
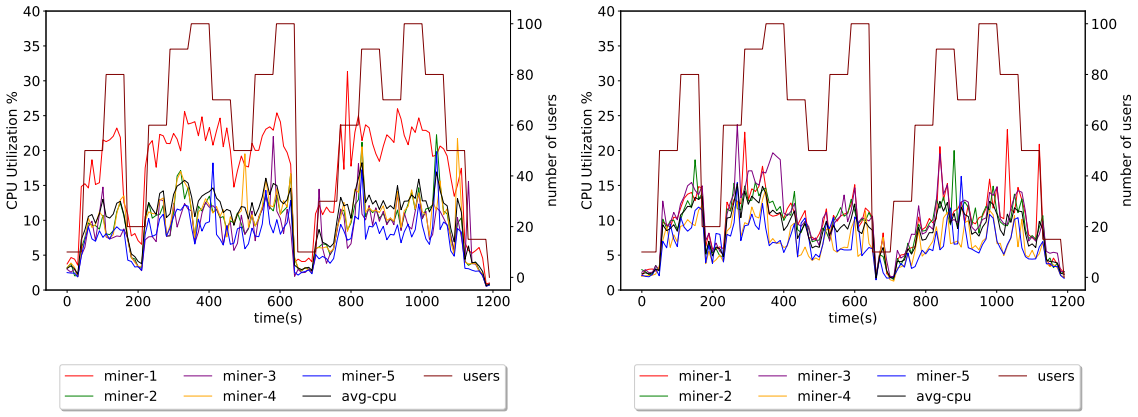


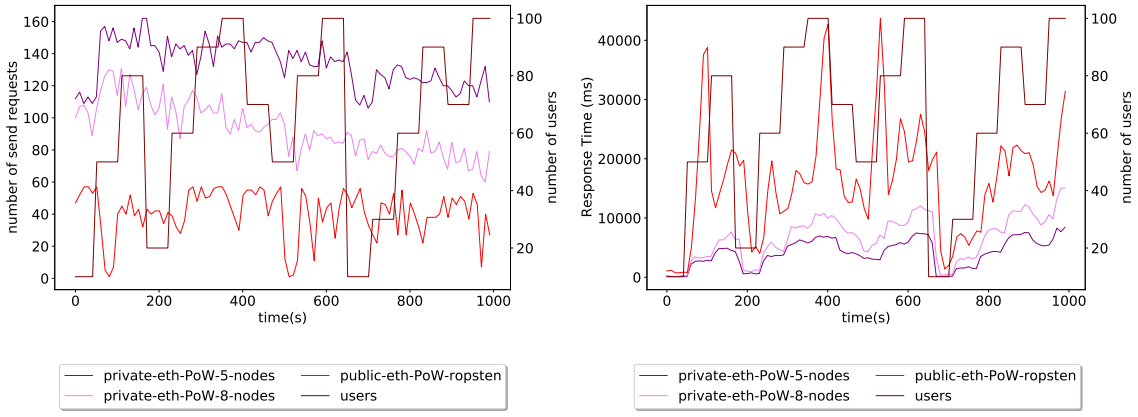
Figure 6.4 Average Response Time Vs. Number of Users before and after multiplying data size by 4

this node (miner-1 in our experiment) takes all the load and may become saturated, making it a potential bottleneck, However, when using 3 receiver nodes (miner-1, miner-2 and miner-3 in our experiment), as shown in Figure 6.5b, the load is balanced between the 3 nodes, reducing the average CPU consumption from 17.34% with 1 receiver to 10.05% with 3 receivers. As a consequence, the platform is also capable on processing more requests at a faster rate. More specifically, the total throughput increased from 93,939 requests to 96,340 requests, while average response time decreased from 108.92ms to 113.53ms. We also performed the same experiment for PoW, but there was no observed difference, mainly due to the consensus algorithm, since all nodes continuously mine, even when there are no requests, causing all nodes to maintain the same levels of CPU consumption. As a result, we do not present the results for PoW here.



(a) CPU per miner for PoA using 1 receiver node (b) CPU utilization Vs. Number of Users

Figure 6.5 CPU per miner for PoA using 3 receiver node



(a) Number of send request for Private Ethereum Proof of Work using 5 and 8 nodes Vs. for Public Ethereum Proof of Work

(b) CPU utilization Vs. Number of Users

Figure 6.6 Number of request, Average Response Time for Private Ethereum Network using 5 and 8 nodes Vs. Public Ethereum Network (Ropsten Network)

6.13 Cost Analysis

In the previous sections, we have shown how the three studied platforms perform under continuous and fluctuating workloads in terms for response time and resource consumption. However, another important concern when considering between a private or a public Blockchain is cost. Therefore, the next question of our study is whether an optimization in resource consumption also translates in a higher cost efficiency. When employing a private solution, the owner would incur mainly capital costs to maintain the deployment infrastructure, which implies that the costs remain even if the resources are not in use. In a public Blockchain, the costs are of a more operational nature and occur on a per-transaction basis. The owner has to pay the transaction gas fee to the miners. The gas fee mainly depends on supply and demand of computation power. The actual cost also depends on the exchange rate between the cryptocurrency and the actual currency, which is known to be rather volatile.

In order to evaluate the economic aspects of our study, we used the data obtained from the previous experiments, including throughput and resource consumption. Given that the public Ethereum we presented in the previous experiments was in fact a simulated version based on private resources, we also conducted an experiment on the Ropsten Ethereum network[9], which is a PoW testnet. The configuration of this experiment was exactly the same as the previous ones.

As it can be seen in Figure 6.6a, the latency of public Ethereum is much higher than running Ethereum on private resources with either 5 or 8 nodes. This is because in public Ethereum it takes much longer to generate the hash and approve a transaction as we have thousands of nodes that will need to accept the request. This is what we observed when we scaled the infrastructure on our private Ethereum from 5 to 8 nodes. Moreover, as shown in Figure 6.6b, due to the higher latency of public Ethereum the total throughput is again much lower. In fact, the total number of requests processed by the public Ethereum was 4715.

For the experiment on public Ethereum, the total amount that was charged for 16 minutes was about 4.09 Ether which is approximately $7,927.36 USD. The gas price per transaction was set to 26 gwei which resulted to a transaction fee of 0.0008 ether or $1.55 USD. It is important to note that transaction fee also depends on the demand in the network, or in other words the workload; the more transactions are sent to the Ethereum network, the higher the transaction fee becomes. This is because the miners need to use more resources to process the transactions. In a single day, the cost of gas may vary between 5 and 200 gwei. In our experiment, this could result to a total cost from around $8,000 USD to about $64,000 USD.

[9] `https://ethereum.org/en/developers/docs/networks/`

For comparison purposes, if we consider the same gas price and transaction fee (although this varies as just mentioned), the equivalent total cost for our private Ethereum-PoW would be 12.69 Ether for 15866 requests processed (5 nodes, 1x transaction size) or about $24,601 USD.

Another thing which affects the cost of transaction in public Ethereum is the size of the data that we are sending. When multiplying the transaction size by 4, the cost per transaction increases from $1.55 USD to $3.87 or 0.002 Ether. This shows that, in public Ethereum, one large transaction is much more cost-efficient than several small transactions. In summary, the recommendation, from a cost perspective, would be to process workloads consisting of large, but few requests on the public Ethereum and more numerous, but smaller requests on the private Ethereum.

6.14 Threats to Validity

In this study, our main objective is to compare public network versus private network in terms of performance, resource consumption and cost. One threat to validity of our study could be generalizability since we have chosen only three platforms among various available solutions. We picked Ethereum Proof of Work as our public Blockchain because of its maturity and popularity. We chose Ethereum Proof of Authority as a private network because it is the recommended consensus by Ethereum for Private networks [111]. In addition, we used Hyperledger Fabric as another Private Network for diversity and because of its popularity. We believe that the options selected are representative enough for the objectives of our study, so our findings should be equally as representative if not generalizable.

A second threat is using a private Ethereum to simulate a public Blockchain system. However we intentionally chose to study the performance of Ethereum miners on a private network to have a better control on the number of nodes and the transaction flow. We partially mitigated this threat by conducting an experiment on an actual public Ethereum testbed for our cost analysis.

To conduct the study, we have used the default configuration for Hyperledger Fabric and a similar configuration for Ethereum Proof of Work and Ethereum Proof of Authority to ensure a fair comparison. However, using different configurations may affect the performance and cost as well. Studying the effect of configuration on performance and cost could be a future extension of our study.

Moreover, in our study, we have only used one public Ethereum which is based on PoW and the most expensive Blockchain platform which supports smart contract. Ethereum2 platform has released its first phase to migrate from PoW to Proof of Stake [153], which is

yet to be finalized, and it seems it will take 2 more years to reach to maturity. There are also several others platform that are based on Proof of Stake including Binance Smart Chain [154], Tron [155] and Matic [155]. These smart chains allow us to deploy our smart contract and take advantage of relatively lower transaction fees. However, although these platforms are cheap, they are not as reliable since there is a limited number of validators on each of these platforms, which makes these validators vulnerable to DDoS attacks and manipulations [156].

Finally, we have run our workload in the same virtual machine as our network, this could affect the latency. However, we did all our experiment using the same method to ensure a fair comparison. Moreover, we deployed our nodes in the same VM using docker compose, which implies a negligible network I/O. We are aiming to expand our study by deploying each node on a different VM to study the effects of the network on performance and cost.

6.15 Conclusion

In this study, we have conducted an empirical experiment to explore the performance difference between private and public Blockchain platforms including Hyperledger Fabric and Ethereum with PoA which are used only in private conditions and also Ethereum PoW which is mainly used in public Blockchain platforms and can be simulated in private infrastructures. We have compared the costs, latency, CPU utilization, memory usage and network usage among all three platforms and showed the impact of concurrent users which tries to continuously write transactions and the number of users fluctuates over the time. We have seen that in terms of CPU usage, memory usage and response Ethereum with PoW is really expensive and has very high overhead due to its consensus algorithm which means that it is much more costly to implement Ethereum with Proof of Work. Moreover, we have shown that Hyperledger Fabric with Raft consensus algorithm is the most expensive in terms of network usage. Finally, we have seen that Ethereum with proof of work when there is more concurrent users, it consumes much less memory in comparison with Hyperledger Fabric. For future studies we are planning to include more Blockchain platforms which are based on lighter consensus algorithms for public usage such as Proof of Stake.

CHAPTER 7 ARTICLE 4: PERFORMANCE AND COST EVALUATION OF PUBLIC BLOCKCHAIN: NFT MARKETPLACE CASE STUDY

7.1 Paper Information

Title

Performance and cost evaluation of public blockchain: NFT Marketplace Case Study

Authors

Mohammadreza Rasolroveicy, and Marios Fokaefs

Date of submission: 2022/07/02

Published in: BRAINS'22

Chapter Overview: This Chapter presents an empirical study of public blockchain technologies including Avalanche Polygon and Fantom networks in terms of throughput per minute, transaction fees, and error rate due to insufficient proposed transaction fees per minute. Machine learning methodologies were also used to illustrate the important features for the prediction of transaction fees and throughput in public blockchain networks. In this study, we discussed that there is no single best public blockchain. Some blockchains focus on decentralization and some on transaction fees, and others on throughout. The results of this study helped us to design a self-adaptive mechanism to dynamically change the public blockchain will be discussed in Chapter 10.

This study covers two of the five dimensions of blockchain performance we explore in this thesis, namely consensus, and comparison of public blockchains.

7.2 Abstract

Non Fungible tokens (NFTs) are receiving unprecedented attention among digital creators and traders. This technology allows creators to certify their digital assets on blockchain as a decentralized, immutable, and transparent database. They can transfer the ownership of NFTs, which can easily be traced without the risk of manipulation. The trading volume for NFTs has surged to $10 billion in the third quarter of 2021. Although the high complexity of the consensus algorithm in blockchain ensures better security, it imposes higher transaction costs and limited scalability. To alleviate high transaction fees, energy inefficiency, and delays, tens of public blockchain platforms with different consensus protocols are being proposed as alternatives for NFT marketplaces. A crucial design choice for such an NFT marketplace is,

in fact, to select the best public blockchain platform. In this work, we evaluate the cost and the performance of three public blockchain platforms, Fantom, Avalanche, and Polygon, in a use case for minting and transferring NFTs. We present machine learning models to predict the transaction cost and the throughput for the three platforms as decision parameters to choose the most appropriate platform. Our experimental results in terms of transaction fees and throughput demonstrate that the Polygon network is more efficient than the other two platforms.

7.3 Introduction

With the unprecedented demand for trading digital assets, verifying their authenticity has become a serious consideration. According to reports, US customers spent more than $80 million on buying forged artifacts [1]. Collectors and traders are willing to buy something genuine and authentic rather than counterfeit [157]. An individual piece of art has value when it is original, and the authenticity is verifiable. Blockchain, known for its immutability and ability to verify ownership and authenticity of transactions, can put an end to the forgery of digital assets [158].

In 2017, non-fungible tokens (NFTs) were proposed as financial security tokens consisting of digital data recorded in a blockchain. An NFT typically contains metadata, including names, descriptions, and image/video/music URLs in decentralized storage such as IPFS [80]. NFT tokens are transparent, immutable, and time-stamped, and their ownership can be transferred by the owner, which allows them to be traded or sold [31]. Due to the emergence of NFTs, Ethereum, a public permission-less blockchain platform, has gained momentum as artists and digital content creators have started to store their digital assets on an immutable Ethereum ledger and demonstrate the proof of the ownership and authenticity of their products to their clients. However, due to high transaction fees on the Ethereum network, storing and transferring an NFT could cost up to 100 USD, an enormous burden for small and medium-scaled digital creators. Multiple alternative public platforms with distinct characteristics in terms of architecture, novel consensus algorithm, availability, and performance, such as Polygon [39], Fantom [38], and Avalanche [159] have been proposed to overcome the limitations mentioned above in Ethereum network. The majority of these platforms are primarily forks of the leading Ethereum network with only the consensus protocol replaced with more cost-efficient algorithms.

This paper simulates an active NFT marketplace to evaluate the performance and transaction

[1] https://www.cbc.ca/documentarychannel/features/brilliant-forgers-works-sell-for-millions-and-expose-flaws-in-the-venerable

fees and compare the three public blockchain platforms, including Fantom, Avalanche, and Polygon networks. We developed a machine learning model to predict throughput and cost and show the important factors to predict these metrics. The ultimate objective is to guide decision-makers in the proper designing of an efficient NFT marketplace.

7.4 Related Work

7.4.1 Performance Evaluation of public and private blockchain Networks:

Tang et al. [160] have evaluated the public blockchain platforms in terms of technology (basic technology, applicability and transaction per second), recognition (market capitalization, number of forks in GitHub, number of commits in GitHub, GitHub stars and number of Twitter followers) and activity (Google search heat, number of commits, and turnovers in GitHub in the previous month). They have shown that the weights of technology, recognition and activity are 16.10%, 59.00% and 24.20%, respectively. However, throughput, transaction overheads and transaction fees are also important factors that need to be taken into account when practitioners want to choose a public blockchain network.

Rasolroveicy et al. [101] have conducted an empirical study on comparison of public versus private blockchain platforms in terms of costs and performance. Their study has shown that public blockchain can handle larger transactions with low arrival rate more efficiently, while private blockchain is more performant and cost efficient for the larger workload of small-sized transactions. Other metrics are required to be discovered in future works, such as considering the metrics that can affect the transaction fees on public blockchain and number of nodes and consensus protocol on security of the private blockchain.

7.4.2 NFT Marketplaces on blockchain:

OpenSea [161] is one of the most popular blockchain NFT marketplaces that allows digital creators and buyers to trade their digital assets through their crypto wallet [162]. This marketplace has two significant limitations: 1) the need for both creators and buyers to have crypto wallets and pay the fees in cryptocurrency, and 2) the need for creators to mint and transfer thousands of NFTs in a short period, resulting in low scalability 3) There is only limited number of supported blockchain platforms.

NBAtopshots is another popular platform that is based on FLOW blockchain [163]. This is a closed blockchain marketplace that is designed only for specific creators, such as NBA teams, to sell their NFTs. This marketplace has two significant limitations. First, this is a closed

blockchain network which restricts the user base and it is not open to the general public. Second, this is a private and permissioned blockchain network with only 37 peers [164] at the time of writing this paper. This makes it effectively distinct from public, fully decentralized blockchain and all the benefits that come with one in terms of integrity and trustworthiness. Chiliz [165] and ArtChain [166] are two more examples of permissioned NFT marketplaces. In general, permissioned marketplaces imply the control of the data and the verification process by one or more parties playing the role of a central authority, which is against the objectives of blockchain.

Rodrigo et al. [167] proposed a decentralized framework to deal with limited scalability and high transaction fees in current NFT marketplaces. The framework is based on a TrustChain network [168], which employs the Directed Acyclic Graph (DAG) technology. DAG is a block-less network with no blocks or miners to mine the blocks. However, this property deprives data of being consistent which is one of the main properties of blockchain to ensure the integrity.

7.5 Methodology

In this paper, we intend to study the performance and costs of the various blockchain platforms to guide and demonstrate what metrics affect throughput and transaction fees when we are minting [83] an NFT. Minting is a special term used in the NFT industry, and it involves registering the unique information of a digital asset on the blockchain including the original creator, the time of creation, and the link to the digital asset. After minting it is possible to transfer the NFT between owners. Ownership history are transparent on blockchain and can neither be altered nor verified, which makes the token non-fungible. In our study we consider and compare 3 different public blockchain platforms in the context of NFT minting and transfer, including Polygon Matic, Avalanche, and Fantom. The three platforms were compared on their throughput, error rate, and costs. As there are dozens of blockchain platforms with the capability of writing smart contracts, we had the following criteria to select the public blockchain platforms for our study: 1) There should be a gas station tracker for the blockchain platform to track the transaction fees, the number of transactions in the queue, and the average block size. There are also other metrics that might play a role on both transaction fees and throughput such as pool size of the pending transaction but these information are not presented public blockchain gas stations. 2) There should be a faucet that can easily supply test crypto coins. To submit transactions on the public blockchain, we should pay in the cryptocurrency of the platform. 3) Different blockchain platforms support different smart-contract programming languages. For this study, it was essential to use the

same programming language for all of our chosen platforms to allow similar conditions. All Fantom, Avalanche, and Polygon networks support the solidity programming language because they are a fork of the Ethereum blockchain.

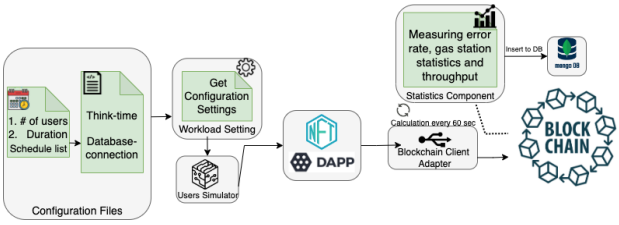


Figure 7.1 Schematic components of workload design

To evaluate the performance and costs of the blockchain above platforms, we have designed a system, as demonstrated in figure 10.1, with the following components:

7.6 Workload generator:

We have implemented a closed workload generator that simulates concurrent and fluctuating users in real-time to submit transactions in the corresponding blockchain platform. At frequent intervals, the generator creates several simulated users who send their requests simultaneously. They will have to wait for a response before submitting a new request. After some time, the number of users increases or decreases to create fluctuation. The experiment ends after a given number of transactions have been submitted or after a certain time.

7.7 Subject application:

We have designed a decentralized application (dApp) using the web3 framework [151] to allow users to make transactions on blockchain. For each user that signs up to our system, we need to create a web3 crypto wallet [169] automatically, and then the user would need to interact with our API to make a transaction. Our dApp simulates an NFT marketplace, which allows for tokenizing digital artifacts and allows users to buy them. To perform our analysis and design our experiments, we have simulated an NFT drop for an NFT marketplace. During an NFT drop, the marketplace is open for users for a limited number of transactions or a limited period. In our case, we will only mint an NFT on the blockchain when it is purchased by a customer and transferred to their crypto wallet.

The system is designed to do “lazy minting” [170], which means that we open the NFT marketplace for a specific time. We announce the quantity and the type of unique NFTs that

we are minting; after the store is closed and the payments are received, we start minting the NFTs on the blockchain and transferring them to the client's unique crypto wallet, based on a First Come, First Serve model. In our decentralized application, the client will link their crypto wallet to their account, and then they can purchase with either a credit card or crypto wallet. After the process of minting and transferring the NFT has been completed; they would receive a notification. Overall, our simulated NFT marketplace architecture consists of following steps: 1) Simulate 1000 users, 2) Create 1000 unique crypto wallet 3) An API to mint 1000 unique NFTs in blockchain and place them in the FCFS queue to proceed. 4) An API to transfer minted NFT to the client's wallet and place them in the FCFS queue to proceed.

The rationale for simulating 1000 individuals is that we only have a limited number of crypto coins to conduct our research. Second, we have developed a real-world use case based on actual NFT marketplaces. In this use case, we usually drop approximately 1000 NFTs in a short time. People typically have one hour to complete their transactions. Totally, we have 2000 transactions, 1000 for minting and 1000 for transfers. In the future, we want to produce more transactions by overloading the network to collect more data for better analysis.

7.8 Blockchain platform:

After the workload generator, the transactions need to be submitted and propagated among the nodes in blockchain platform. Given the differences between the three tested platforms, a "client adapter" is required. The client adapter receives transactions from the workload generator, transforms the data to the expected format and submits the transaction to blockchain. We could not use the public nodes as they have limitations for being used in DApps, therefore, we have used a dedicated RPC [171] node (miner) for our "client adapter". In each blockchain platform, we have created an admin wallet responsible for paying the blockchain transaction fees, and we have deployed a smart contract based on the ERC1155 [172] standard. An ERC1155 smart contract could also support batch minting and batch transfer, but in our use case, we will only mint after a successful purchase by the user, and we are not always sure when the sales will happen and how many of the NFTs would be sold out at a time. It is also worth mentioning that we have used the same payload for all our transactions according to ERC1155 standard.

Table 10.1 shows the details of the platforms which used in our study. As it is shown, the most decentralized network among all three platforms is Avalanche network which is based on DAG (Directed Acyclic Graphs) [29] technology for the consensus with 1567 validators. Avalanche has optimized the DAG technology to be more robust and scalable, it does not

require special hardware for processing, and it can be resilient to "51% attacks" [29], a type of attack where 51% of the network's nodes become compromised. It has also integrated the concept of Proof of Stake (PoS), which means you should contribute a certain amount of money to be a validator in the Avalanche platform. The more money you have in Avalanche, the more voting power you have in the consensus process. In return, the validators will be rewarded based on the amount of money they have contributed. It is noteworthy to mention that as Avalanche is based on the DAG algorithm, it is an entirely new concept as it records only the data instead a linear chain of blocks [29].

The Polygon network is solely based on Proof of Stake [30], which is currently open to the public. It now has 80 validators in the system, and to becoming a validator, you will need to use hardware with special requirements to produce blocks and contribute to the consensus. A minimum amount will be required to contribute to becoming a validator. In PoS, as long as 2/3 of validators are honest, it will operate accurately [173].

The Fantom network has integrated DAG and Asynchronous Byzantine Fault Tolerance (aBFT) in a unique consensus protocol called Lachesis [174]. It aims at improving the efficiency, security and latency issues. The aBFT is a modified version of Practical Byzantine Fault Tolerance, which reduces the number of exchanges required among the nodes to reach consensus. In the aBFT algorithm, unlike PBFT, the messages can be either delayed or lost together, resulting in a faster network. In the DAG-aBFT consensus algorithm, the nodes record event blocks, and each block contains several transactions. The past 2-3 events are periodically exchanged among peers and marked as confirmed. Although the DAG-aBFT reduces the number of exchanges required, its consistency is still based on exchange among peers, and the more nodes we have, the more messages will be needed to be exchanged, which affects the system's latency. The Fantom network currently has 89 validators. The Fantom network can perform accurately as long as two thirds plus one nodes are honest [174].

Table 7.1 The blockchain networks we have used in our experiments

	Consensus	Validators
Avalanche	DAG	1567
Polygon	PoS	80
Fantom	DAG-aBFT	89

7.9 Monitoring:

This component plays a significant role in our project, as we need to collect different types of data in real-time from different services simultaneously.

Our monitoring component includes the following sections:

1. A monitoring script will collect the response time, error from the APIs, and arrival rate in our system in frequent intervals.

2. Blockchain gas station: We have written a web scraper to collect the data for the evaluation of the public blockchain networks, including the number of transactions in the pending queue, average block size, average utilization, and timestamp from all three tested blockchain platforms.

3. Blockchain gas station API: We use the official script of each gas station to receive the specific platform's estimated gas limit and gas price. We have used this data for the prediction of blockchain transaction fees.

4. Blockchain transactions recorder: We have written a script to capture all the live events for minting and transferring on specified smart contracts and store them in the database. Separately, we have created a function to calculate the throughput as the number of transactions per minute. This data helps us to compare which blockchain is more performant in terms of throughput.

5. Retry and errors calculator: To transmit transactions to the blockchain, we created a retry system that attempts to send a given transaction into the blockchain until it succeeds. Then we figure out how many retries we must make every minute to submit a transaction on a specific blockchain platform, which corresponds to the error rate. This data helped us to understand which blockchain platform is error-prone.

7.10 Analysis:

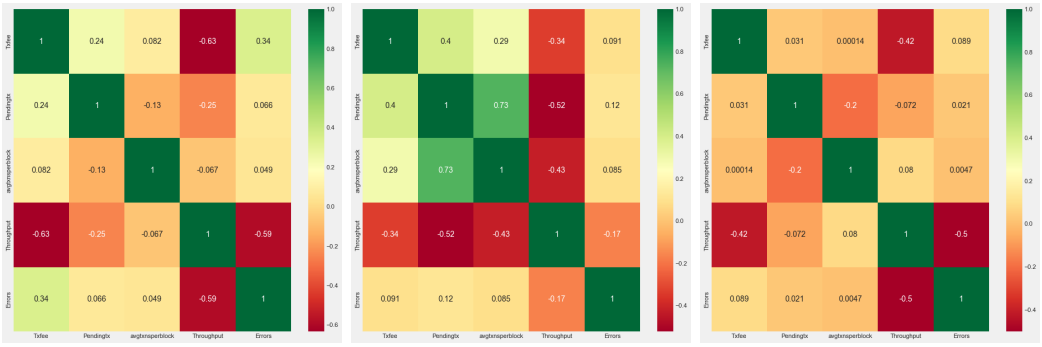
We used Python 3.6 [175] and Scikitlearn [176] libraries to train machine learning models to analyze the data collected in the previous phase. To find the best model for prediction and feature importance, we have implemented regression models and tried different machine learning algorithms such as eXtreme Gradient Boosting (XGB) [177], Random Forest (RF) [178], and Decision Tree [179]. A regression model [180] aims at optimizing the R^2 score, the Root Mean Square Error (RMSE), and the Mean Absolute Error (MAE). Therefore,

these measures are used to compare the different prediction models. Both MAE and RMSE metrics are commonly used to measure the accuracy of continuous variables [181]. The MAE metric measures the average size of prediction mistakes without considering their direction. The RMSE metric is a quadratic scoring method that computes the average volume of the error [182]. It is equal to the square root of the average squared difference between expected and observed values [181]. A Random Forest is a supervised machine learning algorithm popular for classification and regression problems. We have also used hyperparameter tuning to improve the accuracy of the model; however, in this work, our primary focus is not the classifier. Instead, it is on the explanatory factors. We have also considered 75% of data for training and 25% of that for testing. After applying hyperparameter tuning for all our machine learning algorithms candidates, we have received between 0.0 to 0.03 in our R^2 score. Lastly, to account for potential variation in model performance, we used 10-Fold cross-validation and built ten different models using our dataset. As mentioned previously, we have submitted 1000 transactions for minting and 1000 transactions for transferring NFTs to 1000 unique wallets, and for each platform, we had recorded different amount of data as it was dependant on how long it took to successfully record the transactions in the blockchain. Therefore, for Avalanche, we recorded 1179 data in 1179 minutes, 953 data in 953 minutes and 1115 data in 1115 minutes.

7.11 Results

Here we present the analysis of the data collected in three different blockchain networks. As a preliminary step, we analyzed the correlation of various parameters for each blockchain platform to identify potential redundancy and important features. Fig. 7.2 illustrates the correlation between transaction fees, pending transactions (the size of the queue), average number of transactions per block (the size of the payload), throughput and error rate in our three blockchain networks. As we can see, in the Avalanche network, transaction fees and errors have the highest correlation with throughput, -0.63 and -0.59, respectively. As both of them are negative, the higher the transaction fee or the number of errors, the lower the throughput. Next, we can observe that pending transactions have a -0.25 correlation with throughput, and a 0.24 correlation with transaction fee, implying that a large pending queue reduces the throughput but drives the cost higher due to higher demand. In Fantom blockchain, throughput is strongly correlated with most other parameters. The highest correlated parameter is pending transactions with a value of -0.52, then average transaction per block with a -0.43 correlation, and transaction fee with a -0.34 correlation. In practice, this implies that higher throughput is achieved when size of the queue and of the payload, as

well as the transaction fee are kept low. In addition, we can observe that pending and average transactions per block correlate at 0.4 and 0.29 with transaction fee, while there is a strong correlation of 0.73 between pending transactions and average transactions per block. The Polygon network exhibits similar behavior to the Avalanche network, in which transaction fee (-0.42) and error rate (-0.5) are strongly but negatively correlated with throughput. There is also an important correlation (-0.2) between pending transactions and average transactions per block, as in the case of the Fantom network.



(a) Avalanche network (b) Fantom network (c) Polygon network

Figure 7.2 Correlation matrix heatmaps for Avalanche, Fantom and Polygon blockchain networks

Additionally, we have measured the correlation between time of the day and different parameters in 3 various blockchain networks. In the parameter time of the day, we consider whether a transaction was submitted in the morning, in the afternoon or in evening. The time is considered from the time of view of the submitter. Given that in our study, all users come from a single workload generator, time of submission is the same for all. However, other users from different time zones may be using the network, which may affect different parameters. As it s shown in Table 7.2, time of the day does not remarkably affect the other parameters, whereas, in Avalanche, we can see a negative correlation of 0.27 in throughput and a positive correlation of 0.49 with transaction fees. In practice, the later we submit a transaction, the more expensive it will be and the lower the throughput we will achieve, potentially indicating a higher traffic at night for the Avalanche network.

Table 7.3 demonstrates the results of various machine learning regression models, including Random Forest (RF), eXtreme Gradient Boosting (XGB), Decision Tree (DT), and Linear Regression (LR). As we can observe, Random Forest has the fittest algorithm among all four algorithms in R^2 score, MAE, and RMSE. It can solve one of the main bottlenecks in the machine learning process, which is a fitting problem. As part of the training process, Random Forest assigns each feature a numeric value that shows how important that feature is. The

Table 7.2 Correlation of time of the day with different blockchain parameters

	Avalanche- Time	Polygon-Time	Fantom-Time
Tx Fees	0.49	0.02	0.004
Errors	0.07	0.05	0.16
Pending TX	0.12	-0.001	-0.003
Avg Tx/Block	0.13	0.02	0.06
Throughput	-0.27	0.12	-0.08

feature importance value can be retrieved from the model after training the classifier. Factors with higher feature importance values have more predictive power [183]. We have the best prediction performance for transaction fees in the Polygon network, with R^2 score of 0.93, MAE of 5.44, and 8.21. We got 0.86 for R^2 score, 7.77 for MAE, and 11.89 for RMSE for Avalanche. The Fantom network had the lowest performance for transaction fee predictions, with an R^2 score of 0.57, MAE of 50.55, and RMSE of 243.13, suggesting that the forecast is less trustworthy. With an R^2 score of 0.73, an MAE of 0.33, and an RMSE of 0.52, the Avalanche network has the best throughput prediction performance. Fantom and Polygon have almost identical R^2 scores, with MAEs of 0.31 and RMSEs of 0.43 and 0.44, respectively.

Table 7.3 Summary of the results for RMSE, MAE and R^2 score for both transactions fee and throughput predictions among different machine learning algorithms

		TX FEE				Throughput			
		RF	XGB	DT	LR	RF	XGB	DT	LR
Avalanche	RMSE	11.89	14.97	16.35	20.61	0.52	0.58	0.65	0.93
	MAE	7.77	8.44	8.78	13.36	0.33	0.38	0.36	0.65
	R^2 score	0.86	0.77	0.73	0.57	0.73	0.68	0.59	0.16
Polygon	RMSE	8.21	11.8	8.64	13.91	0.44	0.47	0.4	0.72
	MAE	5.44	7.07	5.12	8.18	0.31	0.34	0.35	0.46
	R^2 score	0.93	0.85	0.92	0.79	0.67	0.63	0.4	0.13
Fantom	RMSE	243.13	424	272.95	378.98	0.43	0.47	0.6	0.69
	MAE	50.55	68.5	58.38	75.33	0.31	0.34	0.34	0.43
	R^2 score	0.57	-0.32	0.45	-0.06	0.67	0.63	0.4	0.19

Table 7.4 summarizes the results for evaluating Avalanche, Polygon, and Fantom networks. For all three blockchain platforms, we have simulated 1000 unique users with 1000 unique crypto wallets that are acquiring 1000 unique NFTs for each of them. We have deployed

Table 7.4 Summary of the results in Polygon, Fantom and Avalanche networks

	Avalanche	Polygon	Fantom
Total time (min)	1179	953	1115
Avg Tx fee (gwei)	86.41	51.95	337.17
Avg mint fee (US)	$0.59	$0.02	$0.05
Transfer Fee (US)	$0.15	$0.00	$0.01
Avg Throughput/min	1.69	2.09	1.77
Total Errors	230	138	83

three unique smart contracts based on ERC1155 standards and minted 1000 unique NFTs simultaneously, and transferred each of them to 1000 unique crypto wallets. As we can see in table 7.4, the Polygon network is the fastest in terms of the total time it took to mint and transfer 1000 NFTs, which resulted a processing 2000 transactions in 953 minutes. After that, Fantom with 1115 minutes and Avalanche with 1179 minutes. Among the three, Polygon has the lowest gas fee of 51.95 gwei, followed by Avalanche with 86.41 gwei and Fantom with 337.17 gwei. If we take a look at the conversion of minting fees in US dollars, we can observe that Avalanche is the most expensive among the three blockchain platforms. This is because the actual price that we pay for a transaction also depends on the currency conversion rate. Based on the reports on marketcap [2], one Avalanche token is equivalent to $24.34 USD ranked #14, one Polygon Matic coin is equivalent to $0.61 USD ranked #17, and one Fantom token is equivalent to $0.32 ranked #57. These indexes are confirmed in our experiments.

In comparison with Polygon and Avalanche, the gas fee of Fantom is six times more than the Polygon network, and the price of the token is half of that of the Polygon network. However, we still pay 2.5 more for minting in Fantom, which is $0.05 on average for Fantom and $0.02 in Polygon network. Surprisingly, the price for transferring an NFT in the Polygon network is around 0.00001, which is almost free to transfer, and the cost of transferring in Fantom is $0.01. According to our research, the reason for this is to increase the popularity of the Polygon network for NFT marketplaces, and the Polygon network's validators unanimously agreed that it would be nearly free for people to transfer and trade existing NFTs on their network, and they would not charge validation rewards. However, to prevent spamming the network, the transaction charge is not absolutely $0.

We have also evaluated the throughput of minting each unique NFT in one transaction per minute. As a result of our experiments, the maximum throughput for Avalanche and Polygon networks was four transactions per minute and for the Fantom network the maximum

[2]https://coinmarketcap.com/ last accessed on 2 June 2022

throughput was 3. For Avalanche, we have reached three transactions per minute for the duration of our experiment. We cannot reach much higher throughput rates than this as all tested platforms implement the Ethereum Virtual Machine, which does not allow single users or group of users to issue too many transactions, since these would be perceived as DDoS attacks aiming at saturating the network, especially as the price of creating a smart-contract is not high, and their platform could be saturated quickly. We have also monitored several errors that we might receive in our back-end console when submitting our transactions. We have captured the numbers of errors per minute when we try to submit our transaction to the blockchain and we receive an error. The errors are typically "Returned error: replacement transaction underpriced," which occurs when we provide a transaction fee based on gas station predictions, but it is still refused because it is immediately changed, or "Returned error: err: max fee per gas less than block base fee: address, maxFeePerGas: 361200000000 base fee: 417585263770 (supplied gas 15025147)" this means the nonce that we have used in our transaction is incorrect, and the blockchain node in the network that is being communicated demonstrates that there is more recent transaction is pending. This error usually happens when we have submitted several transactions simultaneously and they have no been processed in the FIFO order and subsequently there nonce has became invalid and we need to calculate the nonce again and resubmit the transaction.

We can observe that the average throughput for Avalanche is 1.69 per minute, which is the slowest platform with 230 errors which resulted in retrying the transaction submission in the queue. After that, Fantom has average 1.77 transactions per minute with the lowest number of errors for the duration of the experiment, which is 83 in total. Polygon has the highest throughput among all three blockchain networks, and it was 2.09 transactions per minute with a total of 138 errors.

It is important for decision makers to understand the reasons why a certain prediction for throughput or transaction fees was made and based on what factors. To better understand the prediction process, we also studied the importance of the prediction parameters used in the Random Forest classifier. One of the parameters for demonstrating feature importance in Random Forest is the Gini index. The metrics show how much, on average, the impurity of the regressor will be decreased when we add that specific feature. The higher value is the Gini index for a feature, the more important it is [184].

Table 7.5 illustrates the Gini importance for predicting transaction fees as the output and taking pending transactions, average transaction per block, and time of the day as the input. We can observe those pending transactions among all three blockchain networks have the highest impact on the transaction fee: 0.4 in Avalanche and 0.41 in both Polygon and Fantom

networks. Moreover, we can see that in all three blockchain networks, time of the day plays a role in the prediction of the transaction fees, which has the highest importance in the Polygon network with 0.45 and in the Polygon network and 0.44 and in the Avalanche network is 0.35. Average transaction per bock has the lowest impact among all parameters, which is the lowest in Avalanche network with 0.09, 0.12 in Polygon, and 0.21 in Fantom network.

Table 7.5 Features importance on transaction fee

Avalanche	Pending TX	0.4
	Avg Tx/Block	0.09
	Time of the Day	0.35
Polygon	Pending TX	0.41
	Avg Tx/Block	0.12
	Time of the Day	0.45
Fantom	Pending TX	0.41
	Avg Tx/Block	0.21
	Time of the Day	0.44

Table 7.6 illustrates the Gini importance for predicting throughput; we have taken transaction fees, error rate per minute, pending transactions, and time of the day as features. As shown, transaction fees have the highest impact on our random forest model, meaning that while the transaction fee fluctuates, we can predict our throughput, which is the number of transactions minted per minute. In the Avalanche network, the Gini's importance of transaction fee is 0.5, in Polygon 0.49, and in Fantom, it is 0.54. We can also observe that while the Gini importance for error rate in Avalanche is 0.17, it has a negligible impact for both Polygon and Fantom, which is only 0.3. The Gini importance for both Polygon and Fantom is also the same, which is 0.2, and for Avalanche, it is lower, which is 0.13. The average transaction per block for all is quite similar, which for Avalanche is 0.12, Polygon 0.15, and Fantom 0.14. Concerning time of the day, it has a negligible impact on all three blockchain networks in our case study, and the Gini index for both Avalanche and Polygon is 0.05 and for Fantom is 0.06.

7.12 Discussions:

We provided and analyzed the performance assessment and cost analysis findings in three public blockchain platforms in the case study of the NFT marketplace with a slow-minting approach in this article. As previously stated, we exclusively mint NFTs on blockchain until they are sold in a lazy minting approach. All three platforms are forks of Ethereum, and

Table 7.6 Features importance on throughput

Avalanche	Tx Fees	0.5
	Errors	0.17
	Pending TX	0.13
	Avg Tx/Block	0.12
	Time of the Day	0.05
Polygon	Tx Fees	0.49
	Errors	0.03
	Pending TX	0.2
	Avg Tx/Block	0.15
	Time of the Day	0.05
Fantom	Tx Fees	0.54
	Errors	0.03
	Pending TX	0.2
	Avg Tx/Block	0.14
	Time of the Day	0.06

they have adjusted the consensus algorithm and validation technique to solve the Ethereum network's inefficiencies and high transaction costs.

In terms of transaction fee prediction, we obtained the lowest value of R^2 score in Fantom using the Lachesis consensus protocol when considering outstanding transactions in the queue, average transaction per block, and time of day. This depended on the fact that the Lachesis consensus algorithm is less reliant on pending transactions and average transactions per block since the consensus often processes many transactions concurrently, and the processing time might be prolonged. We obtained the highest R^2 score (0.93) in Polygon, implying that since the system is built on Proof of Stake and incentive mechanisms, the more transactions in the queue, the more validators charge for transaction processing. Given the significance of features in determining transaction prices, We found that the time of day affects transaction fees in all three blockchain networks. Depending on the time of day, the load on the system may differ, causing variable transaction fees. We discovered that the influence of average transaction per block is far more critical in Fantom transaction fee estimates, which seems to be related to its consensus as the node comprises event blocks, each of which includes numerous transactions. We only recorded 0.09 for the Gini importance of average transaction per block in Avalanche since it has no notion of block and records it rather than a linear series of blocks. In terms of throughput, we obtained almost identical amounts of throughput for the whole network ranging from 0.67 to 0.73. We investigated the number of mistakes per minute, transaction costs, the average transaction per block, and the time of day. We found

that the time of day has little effect on throughput in all networks. Transaction fees primarily determine it; as the cost of transactions changes, we acquire varying amounts of throughput.

We have shown that the number of failures per minute has a more significant influence on throughput prediction since the more rejection we get, the less throughput we get. We have also captured the type of errors in all three platforms during our experiments. In terms of transaction fees, Avalanche is the most costly for both minting and transferring NFT. This is because the value of the Avalanche coin is more expensive than others due to its popularity, level of decentralization and demands. It also seems that gas consumption in Avalanche with 1560 validators costs only 30 gwei more than in Polygon with only 80 validators, indicating that the network's design is more efficient. As a result, the gas price in the Fantom network is much higher than in the other three networks despite lower transaction costs. We should anticipate hefty transaction fees on this network if Fantom's price rises. Avalanche also has the most considerable amount of errors among the other networks, related to the constant fluctuation in transaction fees in the Avalanche network, our transaction was refused because the price had risen since we submitted the transaction. The number of validators needed to re-submit transactions on the Avalanche network constantly changes. Therefore, this might be affected by the number of validators. Moreover, 111 errors out of 230 errors in Avalanche network were due to incorrect amount of nonce, which means that when we submit several transactions to Avalanche node simultaneously, it does not process the transactions in FIFO order and as a result the nonce calculation becomes invalid. For both Fantom and Polygon networks, the type of error has negligible impact on our throughput, thanks to our retry mechanism in our queue that we have retried to resubmit the transaction immediately.

7.13 Conclusion

For academics and decision-makers, studying the best prospects for using public blockchain networks in their decentralized NFTs markets have never been more critical, especially as the number of public blockchain platforms grows. When selecting a public blockchain, decision-making, public acceptability, robustness, transaction fees, and throughput are all considerations. We've outlined our findings in this paper to help decision-makers pick the optimal public blockchain for their real-time NFT applications. Although Matic has surpassed Ethereum as the most popular blockchain platform at the time of writing, we should provide academics with guidelines on determining the most cost-effective and efficient blockchain network. More blockchain networks will be studied in the future since our statistics are based on a single day for all three platforms. We want to expand our research period better to understand the long-term resilience of a blockchain network. The results may alter in the

future. We propose to analyze further criteria such as the possible time for a platform to go down, transaction confirmation time, the amount we meant to pay for a transaction, the amount we've paid for a transaction, and blockchain platform response time for transaction hash.

CHAPTER 8 ARTICLE 5: IMPACT OF DDOS ATTACKS ON THE PERFORMANCE OF BLOCKCHAIN CONSENSUS AS AN IOT DATA REGISTRY: AN EMPIRICAL STUDY

8.1 Paper Information

Title

Impact of DDoS Attacks on the Performance of Blockchain Consensus as an IoT Data Registry: An Empirical Study

Authors

Mohammadreza Rasolroveicy, and Marios Fokaefs

Date of submission: 2022/06/19

Published in: CASCONxEVOKE 2022

Chapter Overview: This Chapter presents a self-adaptive framework for Hyperledger Sawtooth private blockchain to dynamically switch the consensus protocol at run-time to improve the integrity. Hyperledger Sawtooth has been designed with three consensus algorithms including Raft, PBFT, and PoET. It allows changing the consensus protocol if you implement and run all the validators at the same time. In multiple empirical studies, it was shown that there is a single best consensus platform that can fit addressing all requirements such as cost-efficiency, performance, and security, therefore, a self-adaptive framework is required to meet the dynamic needs. By changing the consensus protocol, we will only change the validation methodology of the consensus protocols. This means that, when we switch consensus protocol, the validators for the newly selected consensus protocol will be used for verifying the transactions. In this study, the Hyperledger Sawtooth blockchain platform has been saturated through different kinds of DDoS attacks and the self-adaptive framework was used to change the consensus algorithm based on the CPU utilization to increase the integrity by reducing the packet loss. This is because based on the empirical studies, it was observed that higher CPU utilization increases the number of packet loss in Hyperledger Sawtooth build-in consensus protocols.

This study covers two of the five dimensions of blockchain performance we explore in this thesis, namely consensus, and DDoS attacks.

8.2 Abstract

The current proliferation of blockchain technologies is paving the way for enhancing security in novel technologies such as the Internet of Things (IoT). As the number of IoT devices continues to rise, the likelihood of Distributed Denial-of-Service (DDoS) attacks also increases. Blockchain technology has been proposed as a mitigation strategy for DDoS attacks. The consensus protocol is the main power of blockchain, as it ensures that participants of the network validate each transaction. Different consensus algorithms have been proposed that behave differently with respect to performance, resource utilization, and security when the system is under a DDoS attack. In this work, we benchmark a Hyperledger Sawtooth blockchain infrastructure under various DDoS attacks to study the performance of three different consensus protocols, including PBFT, Raft, and PoET and demonstrate the impact of the attack on performance and resource utilization. Moreover, we propose a self-adaptive management system to automatically reconfigure the consensus protocol's properties to reduce the time overheads and packet loss. Our experiments demonstrate that each consensus protocol behaves differently under DDoS attacks as the workload changes.

8.3 Introduction

With state-of-the-art development in device manufacturing and communication technologies, the Internet of Things (IoT) is expected to build efficient resource management in different industries such as smart buildings, health care, smart cities [185]. It is projected that by 2030 the number of IoT devices that will be connected to the Internet will surpass 125 billion [186]. Due to the nature of devices that produce sensitive data, IoT application security is a critical concern. IoT devices are usually resource-constrained in terms of computational resources and memory, making them vulnerable and challenging in implementing proper security configurations.

The recent DDoS attacks on IoT devices [187] demonstrate the critical need for a proper solution to address the problem. The integration of IoT with blockchain has been proposed [188] to be used as a defense mechanism against DDoS attacks on physical IoT devices. In addition, blockchain technologies can be used to mediate transactions between IoT devices without relying on third parties [189]. Integration of blockchain technology as an immutable data store and a secure network for IoT applications is gaining popularity. One of the main strengths of blockchain against data alteration and DDoS attacks [190] is the consensus protocol. Consensus ensures the immutability and tractability of the IoT data, so we can easily trace the anomalies in the system [108].

A variety of consensus protocols have been proposed for blockchain technologies, including Proof of Work (PoW) [191], Proof of Elapsed Time (PoET) [19], Practical Byzantine Fault Tolerance (PBFT) [21], and Raft [45]. These consensus protocols have different mechanisms and approaches to guarantee the integrity and security of data. Each algorithm has different strengths and weaknesses concerning performance degradation or against attacks thanks to their unique characteristics.

The objective of this work is to evaluate the capacity of consensus algorithms to perform under the stress of a DDoS attack and continue to maintain security and integrity in the blockchain platform. Towards this end, we make three contributions. First, we benchmark the three consensus protocols available on Hyperledger Sawtooth, namely PBFT, PoET and Raft, in a typical IoT simulation environment under fluctuating workload to be used as a baseline. Second, we simulate different configurations of DDoS attacks against our IoT-blockchain network to stress the validators of blockchain. For each experiment, we have captured three metrics, including CPU utilization, packet loss, and response time. The comparison between the experiments helps us to quantify the impact of DDoS on consensus. Finally, we evaluate the effectiveness of self-adaptive systems to dynamically configure the consensus protocol as a potentially better and more efficient and cost-effective alternative.

The rest of this paper is organized as follows. Section 10.4 discusses selected works from relevant literature, while Section 8.5 provides brief descriptions of the compared consensus protocols. Section 9.9 outlines the details of our study design and the experimental setup, while Section 8.14 introduces the design for a self-adaptive consensus protocol. Section 9.11 presents the results of our comparative study for the three consensus protocols, including a comparative evaluation of the proposed self-adaptive consensus protocol. Finally, Section 9.14 discusses the threats to validity of our study and Section 8.20 concludes our work.

8.4 Related Work

Schorradt et al. [192] designed a prototype using blockchain technology to mitigate DDoS attacks. The experimental study used public Ethereum blockchain to show excellent cryptographic integrity protection and a high distribution degree. However, the main bottleneck of the public platform is its low performance. The authors used a tool called Hping3 to flood the network traffic. Their experimental results showed that public blockchain imposes significant latency and low throughput, making it inappropriate for industrial scenarios.

Andola et al. [137] conducted an empirical study to analyze Hyperledger Fabric vulnerabilities and security limitations. They used Hping3 to perform DDoS attacks and continuously

flood syn packets to a target peer. Then, they used Hyperledger Caliper to measure the performance and transaction throughput. They saw that a DDoS attack to the Hyperledger Fabric reduces the throughput significantly and significantly increases the latency. They proposed an approach to mitigate DDoS attacks to the Hyperledger and improve the security and efficiency, but not necessarily to mitigate the impact on performance.

Sun et al. [193] proposed a lightweight framework for integrating IoT and private blockchain technology to mitigate DDoS attacks. Their framework consists of different channels to identify cross-domain access control. They compared their framework with other blockchain-based IoT access control systems and showed that their framework is more secure and resistant to DDoS attacks. They showed that the illegal access requests would not impact the throughput (transactions per second) of the blockchain.

Rasolroveicy et al. [101] conducted an empirical study on comparison of public versus private blockchain platforms in terms of costs and performance. They have simulated blockchain platforms including Ethereum with PoW and PoA consensus algorithms and Hyperledger fabric with Raft consensus protocol. Their experimental results showed that in terms of CPU usage, memory usage and response Ethereum with PoW is really expensive and has very high overhead due to its consensus algorithm which means that it is much more costly to implement Ethereum with Proof of Work. Moreover, Hyperledger Fabric with Raft consensus algorithm is the most expensive in terms of network usage as the Raft consensus protocols relies on communication and exchanges among the peers.

8.5 Background

The Proof-of-Work (PoW) consensus has been popular since the beginning of blockchain technology, including public blockchain networks, like Bitcoin and Ethereum. However, this particular consensus protocol is exceptionally demanding in terms of latency and computational costs [16, 194]. To address these issues, several alternative consensus protocols have been introduced. Hyperledger Sawtooth can support 3 such consensus protocols, namely PoET, PBFT, and Raft.

8.6 Proof of Elapsed Time:

The Proof of Elapsed Time (PoET) consensus algorithm is based on the Trusted Execution Environment (TEE) [195] in Intel® [48]. For the PoET to work correctly, we would need to have at least three nodes. The PoET consensus relies on a “fair lottery” mechanism and Intel SGX CPU is responsible to provide a random sleeping time to the all the network participants

and the peer with the shortest sleeping time will wake up and be responsible to commit the block to the network. The PoET consensus algorithm is very efficient because the nodes of the network don't require to solve any mathematical puzzle and compute resources to add new block in the network [48, 56].

8.7 Practical Byzantine Fault Tolerance:

The practical Byzantine Fault Tolerance (PBFT) protocol is a voting-based algorithm designed on the Byzantine Fault Tolerance principle, which guarantees the system's integrity if up to one-third of the network's total nodes are malicious or corrupted ($n = 3f + 1$, where f is the number of malicious nodes). We need to have $2f + 1$ honest nodes to agree on processing a transaction. In PBFT, only one node can commit a block in the chain, and it needs to exchange several messages among the nodes to add new block to the network [21, 59].

8.8 Raft:

The Raft consensus protocol [22] is a leader-follower model based on Crash Fault Tolerance (CFT) for small and restricted platforms. The Raft consensus algorithm requires a significant amount of exchanges among the peers of the network as the majorities should agree on selecting the leader and all network progress. Subsequently, the more nodes we have in the network, the more exchanges we will have, making the extensive network impractical. The Raft algorithm can tolerate up to 50% faulty nodes but it does not guarantee the security of the network based on Byzantine Fault Tolerance [58, 196].

8.9 Study Methodology and Experimental Setup

The first objective of our work is to assess and quantify the impact of a DDoS attack on the performance of a blockchain system. To this end, we set up an instance of Hyperledger Sawtooth and tested how the three consensus algorithms in Sawtooth (PBFT, PoET, and Raft) behave under a DDoS attack. The purpose of our experiments is to study which consensus protocol provides better efficiency even under a DDoS attack.

8.10 Experiment environment architecture

For our experiments, we designed a custom workload generator that simulates data generated by smart building sensors environment. We assume that the smart building is equipped with

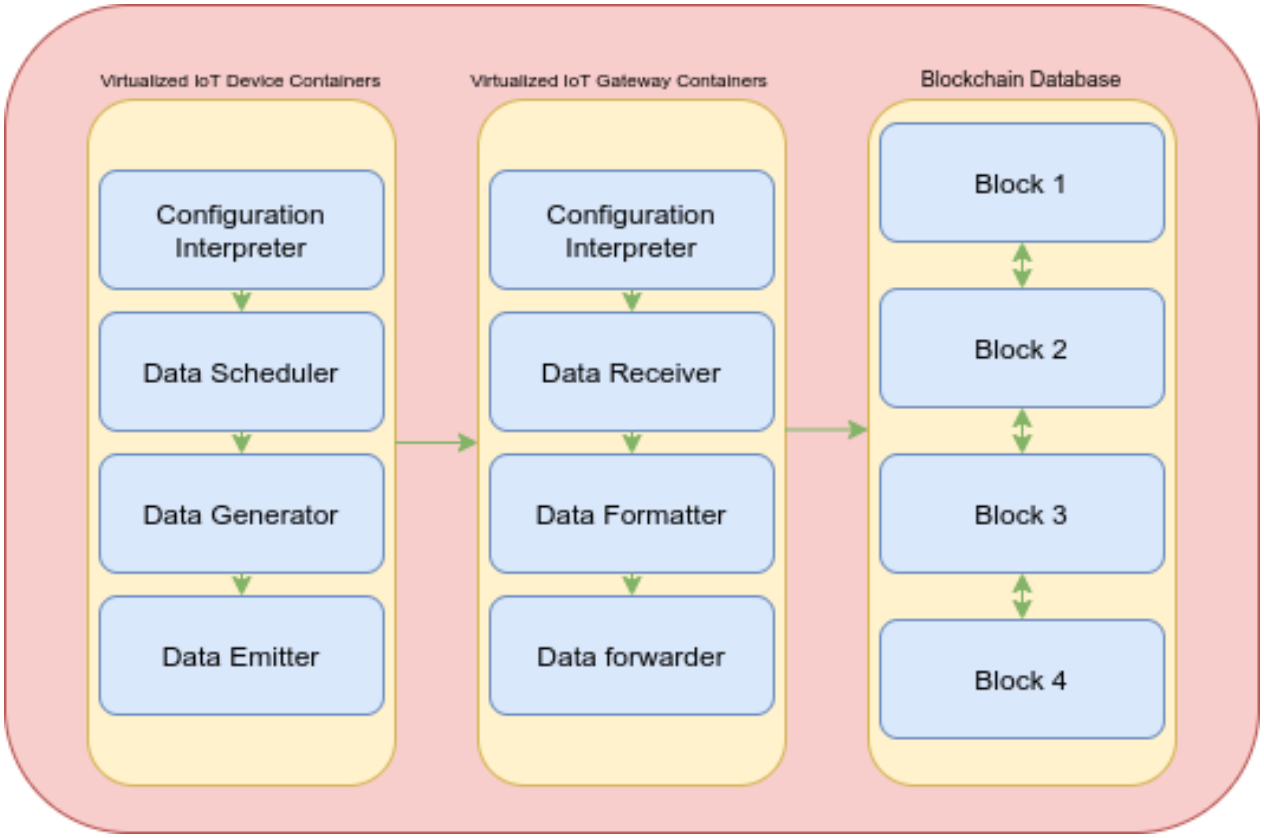


Figure 8.1 Simulated IoT Lab

several IoT devices, including surveillance cameras, GPS sensors, sound sensors (capturing random noise), door sensors, and thermostat. The devices differ with respect to the transmission rate and the amount of data transmitted. For instance, GPS sensors report the geographical altitudes every 2 seconds, thermostats report the temperature every 1 second, door sensors sends the port for the state of a door (Close/Open) every 5 seconds, sound sensors reports the sounds data and samples per second and camera sensors reports video data and FPS(frames per second) is set 15 frames per second. The size of data ranges from a few bytes to few kilobytes based on the type of sensors. These virtual sensors transmit their data to a gateway and the gateway transmits the data to the Hyperledger Sawtooth database through Sawtooth REST API [197]. The simulated workload is designed in order to stress and saturate the system, but under normal legitimate traffic.

Figure 8.1 depicts the architecture of our virtual IoT lab, which is based on Docker containers and python scripts to simulate each device, designed according to the instruction set forth by Ramprasad et al. [135].

The sensors are implemented as Docker containers, on which a Flask Web server [198] is

deployed that hosts a python script that sends a message with a particular payload and transmission rate. NginX [199] is used as the gateway to accept data from the sensors and transmit them to the blockchain. HA-Proxy [200] is used as a load balancer deployed in front of the blockchain.

In our experimental study, the number of IoT devices varies over time, but the distribution of the devices remains relatively stable: 15% Thermostats, 25% GPS devices, 10% door sensors, 30% sound sensors and 20% Cameras. During an experiment, the number of IoT devices continuously fluctuates to periodically saturate the system resources. Besides this, the behavior of each device remains constant throughout the experiment. One experiment has a duration of approximately 30 minutes.

At this stage, the goal of this IoT simulation is only to saturate the system resources with random data as much as possible to understand the performance bottleneck of each consensus protocol with genuine and authenticated requests. In a real-case scenario of IoT applications, to decrease network size, we should store data off-chain and record only the hash in the blockchain database to guarantee immutability and aggregate the data and store only relevant portions in a secure blockchain database. Since, in Hyperledger Sawtooth, validators are responsible for verifying the data, it will not be efficient for a large volume of data to be stored on permanent blockchain storage, especially those relevant only for a short time. Our experimental study aims to cause as much saturation as possible, so we record and store all data under a 'worst-case scenario'.

Figure 8.2 depicts the Hyperledger Sawtooth architecture and its components, which are used in our experimental study. In our Sawtooth network, we have deployed multiple containers. The default framework consists of multiple processes, including the Rest API, Validator Nodes, the Consensus Engine, and others. We have applied some scripts to our Hyperledger Sawtooth network to enable the system for the consensus protocol's dynamic configuration. This allows us to study different consensus protocols under the IoT simulation and various DDoS attacks. It also allows us to implement and apply our self-adaptive mechanism to the Hyperledger Sawtooth network, which will be described in the next section. To enable the dynamic configuration of consensus algorithms, we have followed the Hyperledger Sawtooth guidelines [56, 201].

8.11 DDoS setup

The second objective of our study is to quantify the impact of DDoS attacks on blockchain consensus. Distributed Denial of Service attacks (DDoS) can be carried out from multiple

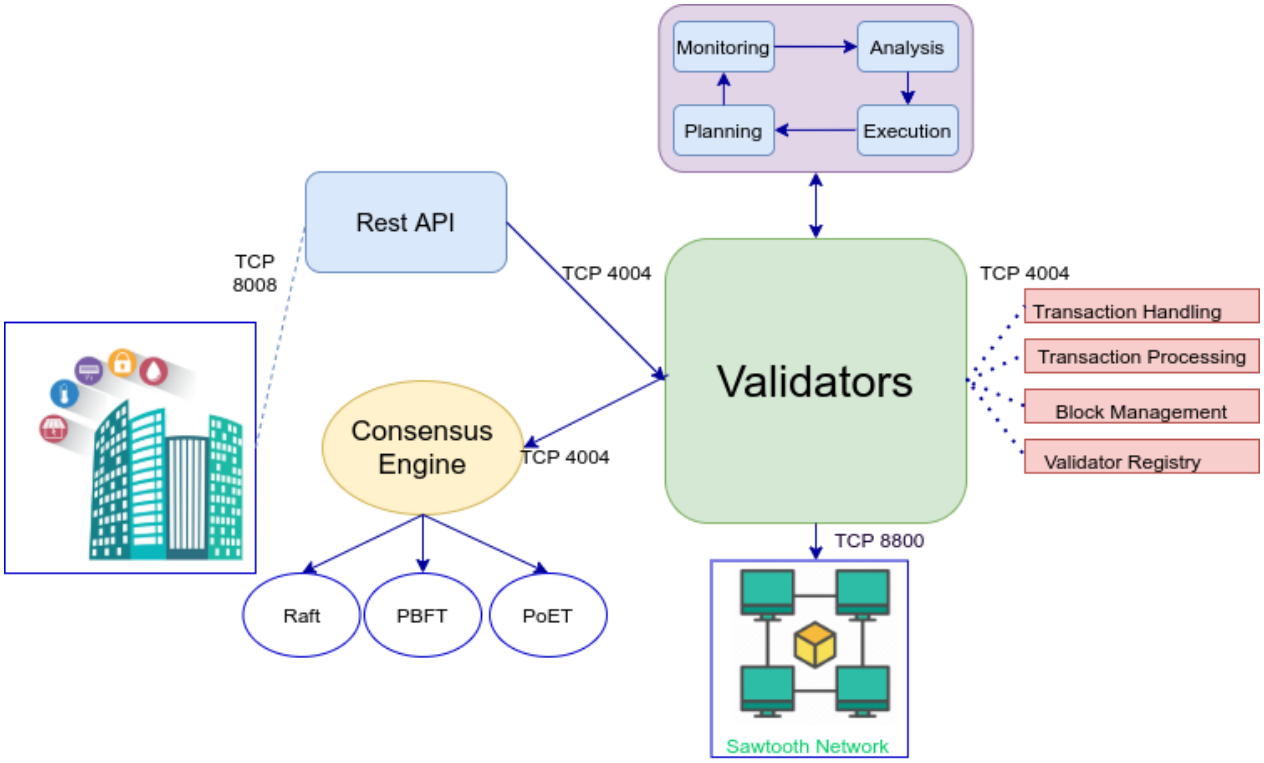


Figure 8.2 Hyperledger Sawtooth architecture and experimental topology

nodes simultaneously. DDoS attacks are commonly launched by botnets, which are automated scripts from a malicious node to carry out an automated task. An attacker launches a botnet and infects many nodes, and the nodes in turn can launch more DDoS attacks. Such an attack undermines the blockchain network and can be launched from either inside or outside the network. To simulate a DDoS attack and generate malicious traffic, we have used Hping3 [202] for our experiments. Hping3 allows to create and transmit custom TCP/IP packets and monitor the replies from the receiving end in a similar manner as ICMP for ping requests. Hping3 handles fragmentation; arbitrary packet body and size can be used to transfer files encapsulated under supported protocols. SYN flood is the most common protocol-based DDoS attack method [203]. This attack exploits the TCP handshake mechanism to saturate the victim with a large number of TCP “initial connect request” SYN packets with a spoofed source IP address. When the target destination responds to each connection request, it needs to wait for the handshake’s final step, the “Acknowledgment”, which is never sent. As this waiting time consumes server resources, it renders the server unresponsive to genuine requests.

8.12 Experiment infrastructure

The actual infrastructure for all experiments was deployed on Amazon EC2 cloud. The infrastructure of Hyperledger Sawtooth consists of a single virtual machine (VM) with 2 vCPUs, 8 GB RAM, and 25 GB SSD (T2.Large). The validator nodes which run the consensus algorithm were deployed as Docker containers inside the VM. To maintain consistency and fairness, our infrastructure included 4 validators as imposed by PBFT for the minimum number of validators. We have installed and deployed the Hping3 tool inside of this virtual machine due to security constraints imposed by EC2. We have used another virtual machine with 2 vCPUs and 4 GB memory (T2.Medium) to run our IoT virtual lab and send the blockchain data. Both machines ran Ubuntu 20.04. As mentioned earlier, we have evaluated the system performance in terms of packet loss, CPU consumption, and response time. For CPU, we have used the Docker Remote API [204] to store the CPU consumption for the validator nodes in the Sawtooth network. For both packet loss and response time, we have used a python script for data emission by every sensor and configured it to receive a response from the Sawtooth Network and record all the requests, including successful requests, lost packets, and response time.

8.13 Statistical analysis

Finally, to analyze the results of our experiments, we have used the Mann–Whitney–Wilcoxon (MWW) [205] test to confirm if the differences observed among algorithms or setups are significant. The reason we used the MWW test is because it is a non-parametric statistical test that does not require the data to be normally distributed, and it also works with a small dataset [206]. In addition, we also computed Cliff's δ [207] effect size to argue about which setup may be better than another. We chose Cliff's δ because it has been shown that it is a more robust statistical method for empirical software engineering datasets when the distribution of the data differs among groups [208]. We have used the "effsize"[1] and "MASS"[2] packages in R for the calculation of MWW and Cliff'sδ. To ensure consistency and reduce the effect of perturbation (such as cloud variability) we repeated each experiment 5 times and we report the averages over these.

[1] `https://cran.r-project.org/web/packages/effsize/index.html`
[2] https://cran.r-project.org/web/packages/MASS/MASS.pdf

8.14 Self-Adaptive System Design

Under the assumption that different consensus algorithms behave differently under different conditions and under different DDoS attacks, we further hypothesize that a dynamic reconfiguration of the consensus protocol as a response to a DDoS attack can further optimize blockchain's response against the attack. To this end, we propose a self-adaptive system to allow dynamic configuration of consensus algorithms, based on the MAPE architecture [107]. Following this architecture, our system consists of four components: 1) a monitoring component which collects the validator CPU consumption from Docker, b) an analysis component which checks whether the CPU consumption has crossed a predefined threshold, c) a planning component that will automatically change the consensus protocol as a response to the threshold violation, and d) an execution component that sends a command to Hyperledger to change its consensus protocol. We have integrated our MAPE mechanism to the Hyperledger Sawtooth network, as demonstrated in Figure 8.2.

The analysis components in our framework check the average CPU consumption of the Sawtooth validator containers. If it surpasses 70%, our MAPE system automatically changes the consensus protocol to a different (predefined) alternative. And in our MAPE architecture it exceeds 90% for a short period, it selects a less secure but more efficient consensus algorithm to reduce packet loss and reduce the CPU consumption. Preliminary results have shown that the increase in CPU consumption can be an indicator for impending packet loss, and as a result, leads to losing data. The goal of a DDoS attack is to reduce the performance of a system and cause harm to legitimate users, rather than compromised privacy or integrity [209]. Under this assumption, it is more important to protect the capacity and the performance of the system. Therefore, when CPU consumption exceeds 90%, the system can afford lower security to better manage the traffic caused by a potential DDoS attacks. Our intention is to reduce the impact of the DDoS on the performance and integrity of system due to data loss.

The planning component in our MAPE architecture has the duty to reduce the number of lost packets and manage resource consumption. This objective can be achieved by changing the consensus protocol to mitigate the burden of additional traffic. The planning phase for the choice of consensus algorithm is described in Algorithm 2. The thresholds to change the consensus algorithm were determined empirically based on the results of study, as presented in Section 9.11. After an adaptive action is taken, the MAPE loop rests for 2 minutes to allow the system to settle and avoid oscillations from immediate counteractions. While performance in IoT applications is a desired and important quality, the main objective of integrating blockchain in IoT applications is to guarantee the confidentiality, integrity and availability of the system.

In order to design our MAPE algorithm, we carefully studied the documentation and relevant research around consensus to evaluate the security capabilities of each consensus algorithm. PoET seems to be the most vulnerable protocol if an adversary hijack the system by simulating the fastest honest node in the system if they successfully compromise $\theta(\frac{loglogn}{logn})$ of the total nodes (where n is total number of nodes in the network) [19]. This implies that the algorithm becomes more secure as the system grows and the number of validator nodes increases. This relies on the fact PoET is based "fair lottery" mechanism and the nodes with the shortest waiting time will have the privilege to add new block. As confirmed in the first experiments of this study (Section 9.11), PoET is the most performant among the three algorithms in terms of CPU utilization, packet loss and response time due to its simple implementation for choosing a leader based on random waiting time. The Raft algorithm can tolerate 51% crashed faulty nodes and PBFT algorithm can tolerate 1/3 of malicious nodes [210]. In summary, we can employ PoET when the network and traffic become too large, but keep PBFT or Raft when there are fewer peers in the network.

Algorithm 1 Self-Adaptive framework for Hyperledger Sawtooth to guarantee the efficiency and performance of the system by choosing the right consensus algorithm

```
1: Input: Consensus-Algorithm - Retrieves the current Consensus Algorithm of the Sawtooth Protocol
2: Input: utilization — the average CPU utilization of containers in a VM
3: Input: lower threshold and upper threshold — the limits of the desired range for the CPU consumption of Validator containers in a VM
4: if utilization ≤ 70 then
5:     Consensus-Algorithm = Raft;
6:     time.sleep(120);
7: else if 70 < utilization ≤90 then
8:     Consensus-Algorithm = PBFT;
9:     time.sleep(120);
10: else if utilization > 90
11:     Consensus-Algorithm = PoET;
12:     time.sleep(120);
```

8.15 Experimental Results

8.16 Baseline performance vs DDoS Attack

The first objective of our study is to benchmark Hyperledger Sawtooth and its consensus algorithms under normal, albeit fluctuating, traffic and establish a baseline for the rest of the study. Table 8.1 summarizes the results for this experiments for the three consensus

algorithms. The results are also visualized in the charts on the left column of Figure 8.3. PBFT has the highest average CPU consumption and highest response time, with high fluctuation for both measures according the standard deviation, among all three algorithms. According to the baseline results, PoET seems the optimal solution on average under normal traffic.

In the second stage of the experiments, we initiated a DDoS attack using Hping3. We first ran the Hping3 command with 1200 packets: `hping3 -c 10000 -d 1200 -S -w 64 -p 4004 -flood -rand-source targetIP`, where `targetIP` is the IP address of the victim, `-c 1000` is the lost packets sent concurrently, `-d 1200` is the size of each packet in bytes, `-s` means we are only sending SYN packets, `-w 64` is the TCP window size, `-p 4004` is the destination port which in our case is the port of validators which is configured in docker-compose, `-flood` instructs Hping3 to send packets as fast as possible without taking care of incoming responses, and finally `-rand-source` means that we are initiating random source IP address to hide the track of attack. For our experiments, we tested two configurations for the DDoS with respect to the packet size, having attacks with 1200 bytes packets and 1800 bytes packets.

Accordingly, Figure 8.4 as well as Table 8.2 show the performance of each consensus algorithm under a DDoS attack. Using Hping3, we simulated two DDoS attacks of different intensities, one with a packet size of 1200 bytes and another with 1800 bytes. The right column of Figure 8.3 visualizes the results for the 1200 DDoS attack for reference and comparison with the baseline. Overall, PoET remains performant even after the DDoS attacks, but Raft significantly deteriorate and had worse performance than PBFT.

As it is shown in Figure 8.3 CPU utilization in PBFT is the highest compared to Raft and PoET before the attack. We also see an increase in CPU utilization among all algorithms as the number of sensors increases. If we compare the before and after the attack (Figures 8.3.a and 8.3.b), when the CPU utilization increases over 70% in both Raft and PBFT, both response time and packet loss increase significantly. PoET algorithm has zero packet loss in the entire baseline experiment. Conversely, after the DDoS attack with 1200 packet size, from Figure 8.3.c, we can see that Raft uses more CPU compared to both PBFT and PoET. When we compare Figures 8.3.c and 8.3.d, the packet loss and response time of Raft and PBFT are very high when the CPU utilization goes over 70% while PoET has the lowest packet loss and response time. In summary, as shown in Table 8.1, when there is no DDoS attack, PBFT has the highest packet loss (2%), highest average CPU usage (52.09%) and highest average response time (133.29 ms). PoET is the best algorithm with 0% packet loss, 43.25% average CPU usage and 123.82ms average response time. Conversely, as shown in Table 8.2, after we have a DDoS attack with 1200 packet size, Raft is the worst consensus

algorithm with the highest packet loss (10%), the highest average CPU usage (73.13%) and the highest average response time (621.92ms). However, the PoET algorithm remains the most performant protocol among Raft and PBFT with 4% packet loss, 66.4% CPU usage and 249.56ms response time.

At a first glance, we can see that DDoS has an immediate impact on resource utilization for all three algorithms, with a consistently higher CPU consumption. On the other hand, DDoS becomes an actual problem when the system is already stressed with high regular traffic. At this point, the DDoS pushes the system off the edge and we can see a highly elevated rate of packet loss and an increased response time at the points of high traffic. While all algorithms experience the same symptoms, PoET seems to handle the situation better, while Raft has a more significant problem compared to regular traffic as opposed to PBFT. The PoET is efficient because of the way it adds new blocks to the network as no resource computation is required, and the leader of the network is chosen based on a lottery-based waiting time and the one with the shortest waiting time wins the lottery and can submit the block. With respect to Raft consensus algorithm, when the system is under DDoS attack, both the leader and follower nodes are overwhelmed with requests to be processed, and since this algorithm relies on communication between peers, the network is under more strain, leading to more lost packets.

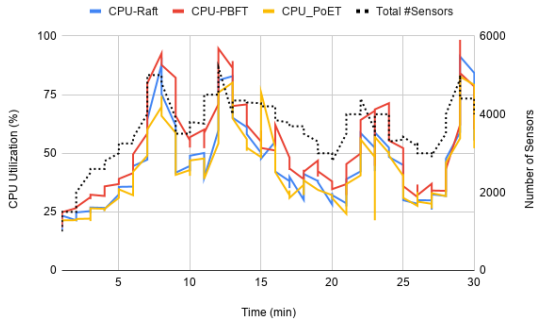
Table 8.1 Summary of results of comparison between Raft, PBFT and PoET before DDoS attack

Algorithms	Raft	PBFT	PoET
Avg packet loss	1%	2%	0%
St.Dev.	0.03	0.06	0
Avg CPU	45.39%	52.09%	43.25%
St.Dev. CPU	17.222	18.66	16.08
Avg Response time (ms)	128.65	133.29	123.82
St.Dev. Response time	14.34	15.15	14.47

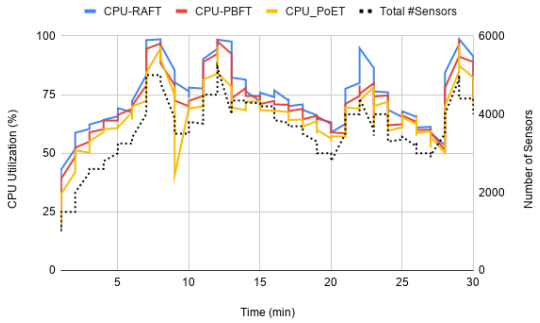
8.17 Comparison between different DDoS attacks

8.17.1 DDoS attack with increased packet size

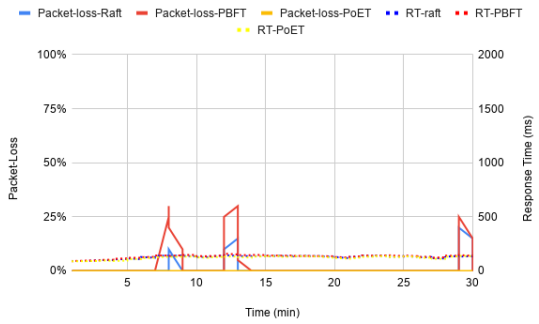
We have repeated the DDoS experiment by performing another simulated DDoS attack with a larger size of packets of 1800 bytes (`-d 1800`) to analyse the impact of packet size on performance. Figure 8.4 shows the difference of the system performance when we increase the DDoS packet size from 1200 to 1800. As we can see in both Figures 8.4.a and 8.4.b,



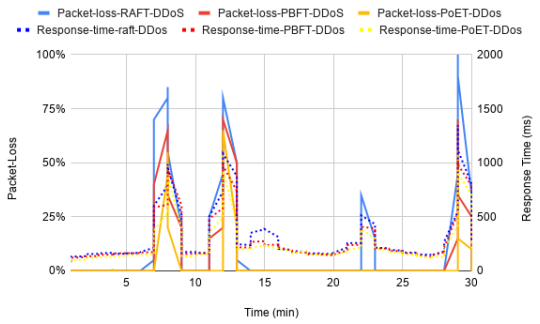
(a) CPU Utilization Vs. Sensors Before DDoS Attack



(b) CPU Utilization Vs. Sensors After DDoS Attack With 1200 Packet Size



(c) Packet-Loss Vs. Response Time Before DDoS Attack



(d) Packet-Loss Vs. Response Time After DDoS Attack with 1200 Packet Size

Figure 8.3 All Response Time, Packet loss and CPU Utilization before and after DDoS attack with 1200 packet size

Raft is still impacted the most by DDoS with respect to CPU and consequently, as shown in Figures 8.4.c and 8.4.d, both response time and packet loss for all three algorithms increase significantly when we have larger packets. In this scenario, the impact is highly aggravated for all algorithms. More specifically, CPU remains above 70% for the entire experiment for Raft, and above 60% for all three algorithms. At peak utilization, all three algorithms reach a point of 100% packet loss, which Raft maintains for longer periods.

Table 8.2 summarizes the result of our experiment and provides a numeric comparison between the two types of DDoS attacks (1200 vs 1800). To better understand this difference, we have calculated the p-value, to perform the MWW test, and Cliff's δ[3] of the differences of averages for each pair of algorithms, the values of which are presented in Table 8.3. When we have 1200 packet size, the difference is not statistically significant for the CPU differs between Raft and PBFT but when we have larger packets, the difference becomes more prominent, which means that at 1800B, PBFT has indeed lower average CPU consumption than Raft. The difference in packet loss between Raft and PBFT remains negligible for both 1200B and 1800B. Response time difference between Raft and PBFT for 1200B is negligible, but it is certainly lower in PBFT at 1800B. PoET has lower CPU consumption than PBFT at both 1200B and 1800B. The packet loss difference between PBFT and PoET for both 1200 and 1800 is negligible. Finally, with respect to response time PoET is better than PBFT, but there is no significant difference at 1800B.

In summary, the situation seems to worsen at 1800B, which makes any difference between the algorithms to become more prominent and consistent. With the exception of packet loss, PoET seems to be the most performant solution of the three. However, given that packet loss may imply integrity and availability problems in blockchain, the impact of DDoS can be considered important in any case.

8.17.2 Simple SYN Flood DDoS attack by Spoofed IP address and unknown port

In the previous two DDoS attacks, the attacker sent packets of data directly to the validators (remember that we specified the port that corresponded to the validator containers inside the Sawtooth VM). These packets needed to be validated and stored in blockchain. As we saw, the larger the packet the higher the impact on the validators and on the consensus algorithm. We designed one more experiment with a different variant of DDoS attack. In this case, the attacker targets the entire blockchain network, not just the validators, by flooding it with

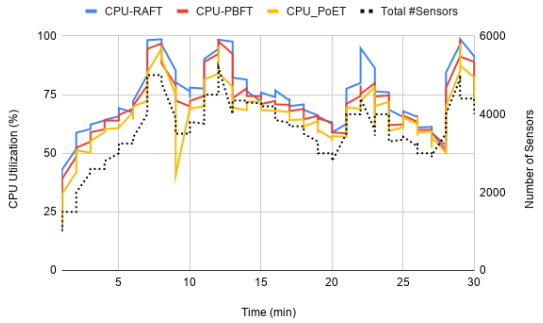
[3]The magnitude is assessed using the thresholds provided by Romano et al. [211], i.e. $|d| < 0.147$ "negligible", $|d| < 0.33$ "small", $|d| < 0.474$ "medium", otherwise "large"

Table 8.2 Summary of the results of comparison between Raft, PBFT and PoET after DDoS SYN attacks with 1200 & 1800 packet size

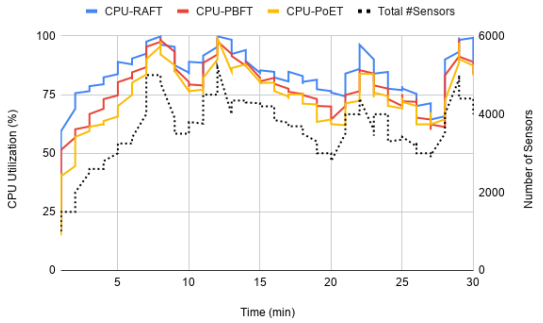
Algorithms	Raft	PBFT	PoET
Avg packet loss (1200)	10%	7%	4%
St. Dev.	0.24	0.17	0.13
Avg packet loss (1800)	34%	23%	16%
St. Dev	0.36	0.32	0.27
Avg CPU (1200)	73.13%	70.11%	66.44%
St.Dev. CPU	14.19	13.48	13.39
Avg CPU (1800)	83.16%	77.46%	74.23%
St.Dev. CPU	11.13	12.60	13.30
Avg RT. (ms) (1200)	312.14	281.65	249.56
St.Dev. Response time	259.11	230.27	203.49
Avg RT. (ms) (1800)	621.92	539.91	498.45
St.Dev. Response time	344.02	279.78	260.53

Table 8.3 Summary of the P-Value and Cliff's delta results of comparison between Raft, PBFT and PoET after DDoS SYN attacks with 1200 & 1800 packet size

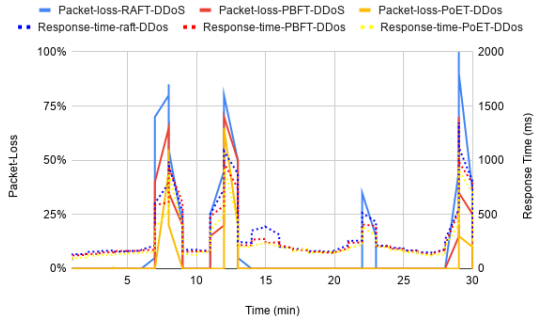
Algorithms	Raft-PBFT	Raft-PoET	PBFT-PoET
CPU-P-Value (1200)	0.08	0.0001	0.026
CPU-Cliff's δ	0.13 (neg)	0.29 (s)	0.17 (s)
CPU-P-Value (1800)	0.0001	6.304e-08	0.055
CPU-Cliff's δ	0.30 (s)	0.43 (m)	0.15 (s)
Packet-loss-P-Value (1200)	0.33	0.021	0.21
Packet-loss-Cliff's δ	0.052 (neg)	0.11 (neg)	0.27 (neg)
Packet-loss-P-Value (1800)	0.055	0.0004	0.105
Packet-loss-Cliff's δ	0.14 (neg)	0.25 (s)	0.114 (neg)
RT-P-Value (1200)	0.127	0.0015	0.0015
RT–Cliff's δ	0.121 (neg)	0.25 (s)	0.25 (s)
RT-P-Value (1800)	0.001	0.009	0.26
RT–Cliff's δ	0.25 (s)	0.206 (s)	0.088 (neg)



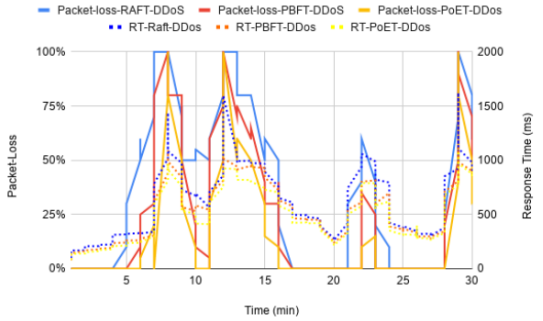
(a) CPU Utilization Vs. Sensors After DDoS Attack with 1200 Packet Size



(b) CPU Utilization Vs Sensors After DDoS with 1800 Packet Size



(c) Packet-Loss vs Response Time After DDoS Attack with 1200 Packet Size



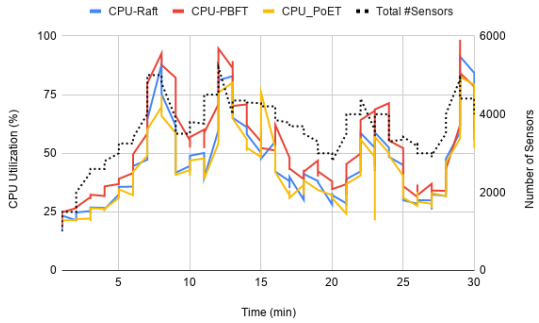
(d) Packet-Loss Vs. Response Time After DDoS Attack with 1800 Packet Size

Figure 8.4 The Comparison of All Response Time, Packet loss and CPU Utilization before and after DDoS attack with 1200 and 1800 packet size

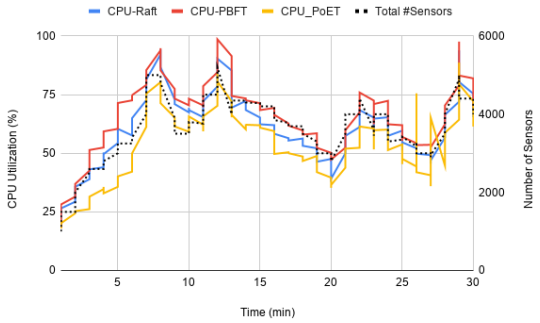
simple ping (SYN) messages. blockchain receives a great number of these messages, that need neither to be validated nor stored, and needs to acknowledge them. The SYN messages flood the input queue which leaves no room for legitimate messages.

To simulate this attack in Hping3, we simply need to avoid specifying the port: `hping3 -S -P -U -a 'FAKE IP' 'destination IP address'`, where `-p` means the packet is marked with PUSH, `-U` means the packet is marked with URG flag and `-a` is the spoofed IP address. To initiate this attack, we have closed the ports and set the IP-table to accept only the trusted IP addresses to connect to the virtual machine. In Hping3, we have used the trusted IP address as a spoofed address to evaluate whether we can initiate attacks with this spoofed address.

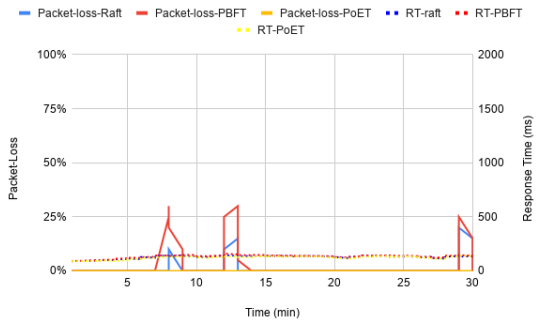
Figure 8.5 compares the results of this third DDoS attack against the baseline. Taking into consideration the results of the other DDoS attacks as well, we can see the impact on performance in this case is not as consistent (CPU is not saturated for a long time, neither is response time as affected), but possibly more prominent especially with respect to packet loss.



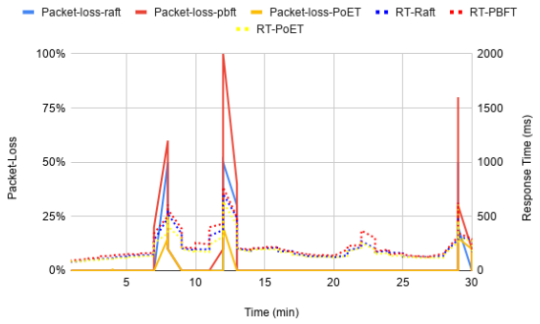
(a) CPU Utilization Vs. Sensors Before DDoS Attack



(b) CPU Utilization Vs. Sensors After DDoS with unknown Port



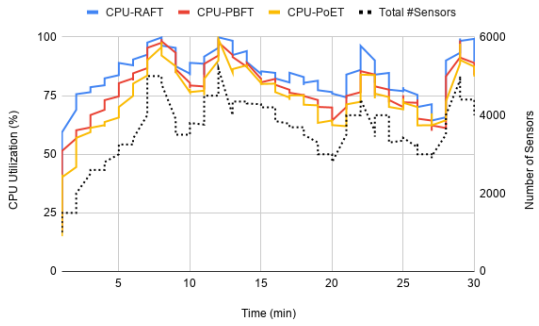
(c) Packet-Loss Vs. Response Time Before DDoS Attack



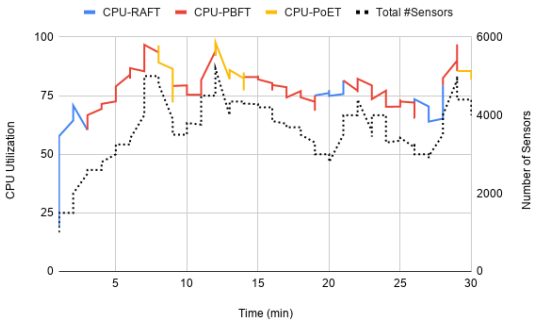
(d) Packet-Loss Vs. Response Time After DDoS Attack with unknown Port

Figure 8.5 All Response Time, Packet loss and CPU Utilization before and after DDoS attack with unknown port and spoofed IP

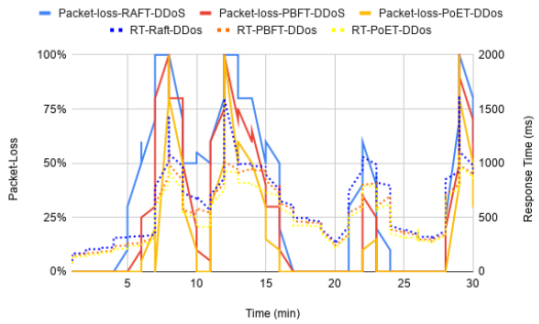
From this perspective, PBFT seems to be more affected than the other two algorithms in this particular attack.



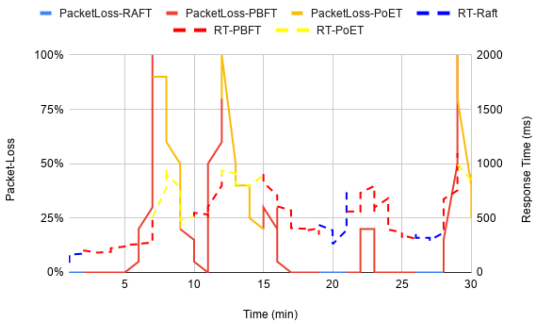
(a) CPU Utilization Vs. Sensors After DDoS Attack with 1800 Packet Size



(b) CPU Utilization Vs Sensors For Self-Adaptive Mechanism After DDoS with 1800 Packet Size



(c) Packet-Loss vs Response Time After DDoS Attack with 1800 Packet Size



(d) Packet-Loss Vs. Response Time for Self-Adaptive Mechanism After DDoS Attack with 1800 Packet Size

Figure 8.6 The Comparison of All Response Time, Packet loss and CPU Utilization After DDoS attack with 1800 and our self-adaptive Mechanism with DDoS Attack with 1800 packet size

8.18 Self-Adaptive Results

In the last experiment of this study, we evaluated the performance of our self-adaptive mechanism as specified in Algorithm 2 to enable the dynamic reconfiguration of the consensus protocol. By default our self-adaptive MAPE mechanism starts with Raft as the consensus algorithm, as the cheapest alternative good enough for low traffic, and changes the algorithm when CPU utilization surpasses 70%. Figure 8.6 shows the performance of Hyperledger Sawtooth under a 1800B DDoS attack with (right) and without (left) dynamic consensus reconfiguration. In the case of the self-adaptive version, each metric is now a single line with

the time periods under a different algorithms colored differently. Table 8.5 summarizes the results of this experiment.

As shown in Table 8.4, our self-adaptive mechanism performs exceptionally well compared to Raft, and at the worst case it is as good as the other two algorithms, at least to what concerns the average values. However, in practice, if we look at Figure 8.6, we will see that the dynamic reconfiguration mitigates well the worst impact of the attack. CPU has fewer violations, packet losses remain high for shorter periods and response time only marginally increases compared to the static consensus version. Another interesting finding is with respect to the number of validators. To illustrate this point, in the dynamic version, PBFT and its 4 validators are used for only a fraction of the total duration of the experiment, and only when it can be of optimal use, while in the rest of the experiment we use the more economical PoET and even Raft that requires only 1 validator. It is important to note that while all our experiments were executed with 4 validators for all algorithms (for consistency and fairness), in practice it is possible to fluctuate the number of validators as the algorithm changes [108]. The ability to deploy validators in containers, which are rather flexible to start and stop, makes the self-adaptive version of consensus a more efficient solution with respect to resource utilization.

Table 8.4 Summary of experimental results for p-value and cliff's delta for the comparison of self-adaptive mechanism with PBFT, PoET and Rat with 1800 packet size DDoS attack

Algorithms	Raft-SA	PBFT-SA	PoET-SA
CPU-P-Value (1800)	3.182e-06	-0.028	0.07
CPU-Cliff's δ	0.373 (m)	0.43 (neg)	0.144 (neg)
Packet-loss-P-Value (1800)	0.005	0.47	0.31
Packet-loss-Cliff's δ	-0.208 (s)	0.052 (neg)	0.0702 (neg)
RT-P-Value (1800)	0.033	0.60	0.54
RT–Cliff's δ	-0.16 (s)	-0.041 (neg)	0.049 (neg)

Table 8.5 Summary of experimental results for self-adaptive mechanism

Algorithm	SA
Avg CPU (st.dev.)	77.11% (10.46)
Avg Packet-loss (st.dev.)	18% (0.27)
Avg Response Time (ms) (st.dev.)	517.63 (254.8)

8.19 Threats to Validity

In this study, synthetic data was used mainly for the sake of availability, but also for better control of the experimental environment. When addressing the impact of IoT devices on the network, we have to consider several parameters such as messaging format (JSON vs. Protobug), communication protocols (such as GRPC), security, and unique workload characteristics of IoT devices, which was the purpose of this research study. However, the same type of data was used across all experiments and variants, which means that any conclusions should not have been affected by the nature of the data.

Another threat to validity of this study could be the number of consensus protocols taken into account. We have chosen PBFT, Raft, and PoET because Hyperledger Sawtooth supports all these protocols. It is possible that other protocols exhibit different behaviors, but we consider this sample to be representative enough, both because the algorithms are sufficiently different from each other and because they are popular enough.

The results of the presented experiments could have been affected by different configurations of the consensus algorithms. To ensure a fair comparison, our experiment uses the default configuration for each algorithm as suggested by the Hyperledger Sawtooth network. It is expected that manual configuration may render an algorithm much more efficient than we reported in our study. By keeping the same default configurations for all consensus algorithms, we assured that the comparison was equally fair for all consensus algorithms. How configuration affects the performance of consensus could be an interesting variant for a future extension of our study.

In our study we have used a popular tool, Hping3, for simulating DDoS attack. This tool has been used in several studies and it seems to be reliable. It allows us to configure and control the quantity, size and the fragmentation of packets in order to stress and overload the target resources and bypass or attack firewalls. This tool can facilitate a variety of experiments. Our results are as good as the quality of the simulated attacks.

Another threat to this study's validity is the number of validators used for this study, as the consensus algorithms will perform differently with more validator nodes. We chose four because the minimum number of validators for PoET is 3 and for PBFT is 4. It is shown that PoET is more secure when we have a large number of validators; however, the performance of the Raft consensus algorithm drastically degrades when we have more than 9-10 validators [108]. In the future, it is our intention to repeat this study with a variable, and possibly dynamic, number of validator nodes.

Finally, some threats may be present due to the deployment infrastructure. First, we have

used a single virtual machine for our empirical study, which implies the network I/O is negligible. Once again, all experiments were conducted on the same infrastructure, so any impact can be factored out. Second, as we used a public cloud infrastructure, there is always the risk of cloud variability. For this reason, we repeated each of our experiments 5 times and we reported the averages.

8.20 Conclusion

We have conducted an empirical study to quantify the impact of DDoS attacks on the performance of systems for IoT applications. We compared the behavior of three consensus algorithms, PBFT, PoET and Raft, in Hyperledger Sawtooth under various DDoS attacks that varied in intensity and form. Our results showed that the attack had a remarkable impact on the performance of all algorithms. Importantly, the loss of data during the attack may seriously impair the integrity and reliability of the blockchain system. We believe this self-adaptive approach can be used temporarily to mitigate the impact of the attack until a more drastic solution is prepared. For future works, we are working on more sophisticated self-adaptive systems to capture multiple concerns and utility functions such as manual configurations of algorithms and packet loss. What happens, for example, if there are situations in which we would prefer to minimize packet loss at the expense of using more CPU power.

CHAPTER 9 ARTICLE 6: DYNAMIC RECONFIGURATION OF CONSENSUS PROTOCOL FOR IOT DATA REGISTRY ON BLOCKCHAIN

9.1 Paper Information

Title

Dynamic Reconfiguration of Consensus Protocol for IoT Data Registry on Blockchain

Authors

Mohammadreza Rasolroveicy, and Marios Fokaefs

Date of submission: 2020/06/15

Published in: EVOKE CASCON 2020

Chapter Overview: This Chapter presents a self-adaptive framework to dynamically change the consensus protocol in a Hyperledger Sawtooth private blockchain to improve the performance of a decentralized application. Hyperledger Sawtooth is built in with three consensus protocols including Raft, PBFT, and PoET algorithms and it allows to change of the consensus protocol at run-time by following set-up commands. To avoid the loss of the transactions, at the beginning of running the Hyperledger Sawtooth, we simultaneously run all validator nodes for all three consensus protocols. By changing the consensus protocol, the validation methodology for adding a new block to the blockchain will be changed automatically for the next pending transactions. The intention of this study is to present the possibility of exploiting self-adaptive systems in private technologies to improve the performance and efficiency of the systems.

This study covers one of the five dimensions of blockchain performance we explore in this thesis, namely consensus.

9.2 Abstract

Blockchain technology has recently gained momentum among practitioners to increase security in shared distributed platforms. One of the main characteristics of Blockchain is a consensus that prevents double-spending attacks on the network and lowers the chance of Distributed Denial-of-service (DDoS) attacks. However, each consensus algorithm has its strengths and its limitations. The purpose of this paper is to design a self-adaptive mechanism that can dynamically choose the appropriate consensus algorithm for the network. This framework provides an ability to easily manage and change the consensus algorithm for IoT data registries

based on Blockchain in an effort to manage the cost, performance, and security of the network. Moreover, it can dynamically reconfigure properties of the consensus protocol including the number of validation nodes, validation time, and others. The goal is to provide an effective and cost-efficient consensus protocol while ensuring the quality of service. Our experiments show that different consensus algorithms behave differently as the workload changes and we demonstrate how our MAPE-k architecture can allow additional flexibility under dynamic conditions.

Moreover, we propose a self-adaptive management system to automatically reconfigure properties of the consensus protocol to reduce the time overheads and packet loss. Our experiments demonstrate that each consensus protocol behaves differently when they are under DDoS attacks as the workload changes and we show how our MAPE-k architecture can improve the efficiency and quality of the service.

9.3 Introduction

Internet of Things (IoT) applications are rapidly gaining popularity and by 2030 the number of IoT devices that will be connected to the internet is projected to surpass 125 billion [186]. IoT has been applied on several domains such as smart homes, smart agriculture, and smart hospitals [185]. Nevertheless, IoT is not without challenges. Due to improper configuration and limited computation resources of IoT devices, they are vulnerable and a main target of hackers. In October 2016 [212], an attack, known as the Mirai botnet attack, exploited the poor security configuration of thousands of IoT devices, connected to the internet using IP addresses, such as security cameras and wireless printers. The attackers created several DDoS (Distributed Denial of Service) attacks and targeted the DNS (Domain Name System) provider. Subsequently tens of thousands of IoT devices were used to produce several DNS lookup requests which resulted in downtime of the internet for several users in the USA, Canada and Europe [8].

IoT applications usually include a variety of sensors, as well as controllers and network gateways. These distributed IoT peers constantly store their generated data to the database and they need to access the database for reading and making appropriate decisions. The massive amount of generated data by IoT applications will make it difficult to control and monitor the system properly [213].

Distributed Ledger Technology (or more commonly known as Blockchain) has been proposed as a secure and immutable network for IoT applications which can addresses three main concerns: availability, confidentiality, and integrity [27]. Blockchain principle is built around

a consensus protocol [45] that secures the network against double spending attacks [5] and DDoS attacks [190] in IoT applications. This means that when we store IoT data in the network, we can ensure that our data is almost immutable and tractable, so that we can easily trace the anomalies inside the system.

Taking advantage of Blockchain has proven [8] to be a valid solution to mitigate DDoS attacks by introducing a distributed database that is based on peer-to-peer (P2P) networks. In Blockchain systems, each data block that is submitted to the network will be verified by the peers of the network, and then it will be broadcast to all members. Moreover, the leaders in Blockchain only have permission to append the block to the network and these leaders are chosen by a consensus mechanism [7]. Blockchain, unlike the traditional distributed databases, only provides read/write access of the data and when data has been stored, it can neither be altered nor erased. This technique can be effective in preventing attacks, like the Mirai botnet attack, in IoT applications because of non-repudiation; malicious activities are stored, they cannot be erased or tampered with and they can be traced [8].

Despite the ability of Blockchain to guarantee immutability of data through consensus, it is also true that this consensus protocol can be a performance and resource bottleneck, which is contradicted for real-time IoT applications. A number of different consensus protocols have been proposed for Blockchain including Proof of Work (PoW) [214], Proof of Elapsed Time (PoET) [19], Practical Byzantine Fault Tolerance (PBFT) [21], Raft [45] and Devmode [215]. These algorithms have different mechanics and properties, which also include different degrees of performance degradation. One of the objectives of this work is to study and quantify this impact on a variety of IoT application scenarios. To achieve this, we selected Hyperledger Sawtooth [56], a Blockchain platform that supports at least four of these consensus algorithms [216], namely Raft, PoET, PBFT and Devmode. To evaluate the performance of each consensus algorithms, we subjected the platform to varying IoT data loads and measured the performance in terms of CPU and response time.

It is expected that the different algorithms will have both advantages and limitations and that these would manifest under different scenarios. Consequently, it is also expected that the stakeholders would want to choose the algorithm that fits best their requirements and business goals. Given that in a real-time IoT application, these requirements may be dynamic, so should the decision about the consensus algorithm. In order to implement such a dynamic behavior, we invest in the design of a self-adaptive management system that would automatically and dynamically switch between consensus algorithms and deployment configurations according to the requirements of the applications and based on changes in the data load. The proposed self-adaptive system is based on the MAPE-k architecture [107].

Each consensus protocol in Blockchain has strengths and drawbacks which would manifest under different scenarios. As a results, we expect that the stakeholders would like to use an algorithm which can perform best based on the business goals. In this work we have simulated a real-time IoT environment to create a scenario which the system is live and the data are being stored in Blockchain ledger. And we have simulated a DDoS attack scenario to see which consensus protocol will outperform when there is an attack to the system and can perform better when the system is under the attack. Afterwards, based on the results of the first experiment we invest in the design of a self-adaptive management system system which automatically and dynamically switch between consensus algorithms and deployment configuration according to the requirements of the application and based on the changes in the data load. Our self-adaptive architecture is based on MAPE-K management system [107].

This work makes two contributions:

- A comparative study on the performance overhead of the available consensus algorithms in Hyperledger Sawtooth including PBFT, PoET and RAFT.
- A self-adaptive system to dynamically and automatically choose and deploy the right consensus algorithm at run-time.

The rest of the paper is organized as follows. In Sections 10.4 and 9.7 we outline the related literature and provide the background of the concepts and technologies used in our study. In Section 9.9, we present the methodology and preparation for our study, while in Section 9.10, we present our novel adaptive and dynamic consensus protocol. In Section 9.11 we discuss the results of our experimental study and the evaluation of our self-adaptive mechanism. In Section 9.14, we discuss the threats to validity of our study and finally, Section 9.15 concludes our work.

9.4 Related work

Our work studies the performance of Blockchain systems, focusing on the consensus protocols, when they act as data registries for IoT applications. In addition, it proposes a self-adaptive system to automatically manage the deployment infrastructure of Blockchain and dynamically switch between consensus algorithms. As such, the present work is related to the integration of Blockchain with IoT and self-adaptive systems.

9.5 Integration of IoT and Blockchain

Dorri et al. [16] proposed a Blockchain-based architecture for resource constrained IoT network. They have removed the Proof of Work (PoW) consensus algorithm to increase efficiency and overheads for real-time IoT applications. They argue that using Blockchain for IoT security and privacy is vital as we can store our produced data in a private immutable ledger that performs similarly to Blockchain and it is also manageable. They have evaluated their proposed method for smart home applications using the Cooja simulator [91]. The results of their study demonstrated that the proposed solution reduces the processing and packet overhead comparably to the Blockchain implementation which was used in traditional Blockchain PoW-based algorithms in Bitcoin.

Aung and Tantidham [53] reviewed different possible approaches that can be applied to an Ethereum private Blockchain for a Smart Home System (SHS). In their work, they have considered SHS as a case study which is an integration of home appliances together with sensors to get automatic operations of heating, lighting, air conditioning, home security, health care systems, and others. The authors presented an approach for a private Blockchain implementation for SHS to deal with privacy and security issues. The Ethereum Blockchain packages for SHS, according to its smart contract features for carrying out access control policy, data storage, and data flow management, have been reviewed. As compared with the work of Dorri et al. [16] the architecture of the two systems is different. The architecture proposed by Aung and Tantidham consists of a smart home Validator nodes (SH-Validators), a private Blockchain and a local storage, which connects to SH sensor and actuator devices. SH-Validators carries out a private Blockchain. The purpose of the private Blockchain is to store policies for data flow or transaction management. However, they argue that because of the low transaction time of Ethereum Blockchain, it may be inappropriate for time-sensitive conditions.

Liang et al. [217] proposed a secure Hyperledger Fabric Blockchain-based dynamic secret sharing mechanism to secure data transmission for Industrial Internet of Things (IIoT). The security authentication in Hyperledger Fabric power, a Blockchain-based network, can guarantee secure communication between the power node and the system, as well as between systems. Their experiments showed that the optimized Fabric power data storage and transmission show high security and reliability. The proposed technique can improve the transmission rate and packet receiving rate by 12% and 13%, respectively. According to the authors, it is important to evaluate how efficient their model is in terms of resource and energy consumption.

Alaslani et al. [218] have studied the effect of emerging IoT applications on the end-to-end delay encountered by Byzantine-based Blockchain systems. The effects of a broad range of IoT traffic characteristics including packet arrival rate, payload size, device density, and surface area were studied. The results illustrated the importance of incorporating the unique IoT characteristics in designing Byzantine-based Blockchain systems for IoT networks. They suggested enhancing the semi-distributed BFT (Byzantine Fault Tolerance) algorithms by partitioning the large-scale IoT network and reducing the communication complexity to facilitate and enable the wide adoption of Blockchain in the IoT context. However, sending large-scale data to the validator nodes (consensus protocol) will introduce another network and bandwidth bottleneck.

Ahmed et al. [8] have proposed a Blockchain-based solution to the problem of mitigating Mirai botnet attacks on IoT devices. The solution depends on dividing the network into AS (Autonomous Systems) which communicate through the Blockchain network to share malicious node information. Blockchain platforms are used to store and share a list of IP addresses of different hosts connected to an AS, to show which of these have been identified as malicious. The proposed approach is simulated on a custom simulator, which is adapted to use an appropriate value for the malicious threshold. The authors are interested in finding the delay incurred in propagating the information of malicious hosts between the AS over an Ethereum Blockchain. To this end, they have simulated a scenario in which the AS is very large and thousands of malicious hosts are connected to the AS. The simulation indicates that the proposed approach, when appropriately tuned, can yield a true detection rate of 95%.

Lingering Zhao and Jiangshan Yu [219], have studied an alternative of Blockchain technology for integration of lightweight IoT devices. They have summarized the central features of IoTA by analyzing the functionality of this Directed Acyclic Graph (DAG)-based model. DAG-based examples are expected to provide greater scalability and fast economical benefits that most of the Blockchain-based platforms fail to accomplish. They have shown that, IOTA has a transaction rate under 20 tps according to live transaction monitor provided by the Tangle [220] A peak transaction rate more than 400 tps has been seen in IOTA before and it has been handled well with a confirmation ratio over 99%. Although, IoTA [221] is much more faster than Blockchain. However because there is neither validator nodes nor blocks to participate in consensus and integrity of the data, it is shown that IoTA is still vulnerable to double-spending attacks [222].

9.6 Integration of Blockchain and Self-Adaptive Systems

Alzahrani and Bulusu [223], proposed "Block-Supply", a decentralized anti-counterfeiting supply chain that exploits NFC (Near-field communication) and Blockchain technologies. This Block-Supply chain can detect modifications, cloning, and tag reapplication attacks, in addition to tracking products without a centralized managing server. The authors developed a new truly decentralized consensus protocol that does not need PoW and dynamically uses a set of validators of varying size each time a new block is proposed based on random algorithms. The proposed protocol employs a game-theoretical model to analyze the risk likelihood of the block's proposing nodes. Likewise, the proposed protocol uses a novel, decentralized, dynamic mapping between the nodes that participate in the consensus process. The protocol protects against several real attacks mounted by powerful adversaries. The simulation results depict that the new protocol is scalable for large networks using a relatively small number of validators. Moreover, it maintains a satisfactory level of security. However, concerning the simulation, they have used OMNET++ [224], which is not considered to be an ideal real-time simulation for IoT applications.

Liaskos et al. [225] proposed a simple model for quantitative reasoning about the parameters that impact and are influenced by the consensus process of public open-access Blockchain networks and attempted one of the first formulations of such a model as an adaptive system. To motivate the approach, the authors experimented with an idealized controller, only to subsequently analyze it and reveal the challenges in designing real-world controllers. The authors identified variables that influence its sustainability, including those imposed by the environment and those that can be directly configured by the network's governance. Next, they proposed a model to show how these variables relate and affect each other. With that model underway, they worked out the problem as a standard control engineering problem in which a controller (Blockchain governance) optimizes the parameter choices so that the output of the controlled system (Blockchain network) meets certain objectives.

Casado-Vara et al. [226] proposed a new adaptive strategy for Blockchain-based system by exploiting game theory and prediction of accuracy of future states to reduce the tracking error and enhance the effectiveness of the algorithm aiming at ensuring the efficiency of an adaptive temperature control algorithm. The authors have designed a Blockchain-based platform to enhance the operation of the monitoring and control of the IoT networks to improve energy efficiency. This control system optimizes the temperature of a smart building uses state prediction module, which uses Markov chains to predict the accuracy states of IoT nodes in future time. The experiment results indicate that the predicted temperature signal is surrounded by a small interval close to the collected temperature data.

Hovland and Kucera [227] designed and implemented a self-adaptive simulation model for the Proof-of-Work consensus algorithm in Blockchain. The model has been validated by a statistical analysis of miner solutions to PoW challenges with varying difficulty levels from a sample of more than 500,000 solutions. The controller in their architecture consists of a nonlinear inverse model, which estimates the next difficulty level from the desired block time, the current average block time, and the previous difficulty level. By using the simulation model, controller designs can be tested and analyzed in a few seconds compared to typically several days for implementation on a test network. The purpose of the study is to develop a Blockchain difficulty adjustment controller and to study the stability of the closed-loop system from a control engineering perspective. Their results show that their model has fast response and high scalability in all cases.

9.7 Background

9.8 Consensus in Blockchain

Consensus in Blockchain is introduced to address two problems, namely the Byzantine Generals Problem [43] and Double Spending. Double Spending means that we can not reuse the digital currency in two separate transactions at the same time. Blockchain can address this issue by verifying all the transactions by several distributed nodes in the decentralized network, and not by a single authority. This is more robust especially when the single authority can also be corrupted. The Byzantine Generals problem is a common issue in distributed environments. The data in the network will be delivered between several nodes through peer-to-peer communications. However, some of the nodes in the distributed network could act maliciously which can corrupt data. Healthy nodes need to able to distinguish information delivered to them that has been altered legitimately and obtain consistent results with other healthy nodes. Byzantine Fault Tolerance implies that the system has enough nodes, so that it is resilient in the presence of some malicious nodes. The consensus protocol in Blockchain has been designed to provide data verification (by consensus) and fault tolerance [45].

Proof of Work (PoW) is the first consensus algorithm that has been used in Blockchain. Its core idea is to give rewards based on hashing power competitions among peers in the Blockchain network. Each node calculates and solves a specific mathematical problem. The first node which is successful to solve the mathematical puzzle will be allowed to create the next block and will be rewarded a certain amount of currency, which could be either Bitcoin or the Ether currency in the Ethereum network [47]. PoW is computationally expensive and has a time overhead which is undesirable for real-time IoT applications [16].

Next, we will discuss three alternative consensus protocols that are introduced in Hyperledger Sawtooth.

9.8.1 Proof of Elapsed Time

Proof of Elapsed Time (PoET) is the main consensus protocol that was introduced by Intel in Hyperledger Sawtooth. The algorithm randomly chooses the next leader to finalize the block. This consensus protocol employs this method to deal with malicious peers in an open-ended Blockchain network. PoET utilizes the Trusted Execution Environment (TEE) which is called Enclave inside of Intel processors to prevent cheating and to provide confidentiality, integrity, and blacklisting malicious behaviors based on asymmetric key cryptography and an additional set of election policies [48, 228].

9.8.2 Practical Byzantine Fault Tolerance

The PBFT consensus algorithm is based on the Byzantine Fault Tolerance principles, which can tolerate that less than one third faulty nodes of total nodes in the $n = 3f + 1$ total nodes in the network. PBFT extends this principle to the concept of consensus: at least $2f + 1$ nodes need to agree to reach the consensus and confirm transactions [59].

9.8.3 Raft Algorithm

The Raft [22] algorithm can be used in private Blockchains, where an administrator of the network can add or remove nodes. This algorithm is proposed to solve the inefficiencies of PBFT. It is a lead-based algorithm which means that the peers of the network will choose the leader in an election process. This means that only the leader node is allowed to publish blocks which are validated by other peers. This algorithm cannot tolerate any malicious nodes but it can tolerate up to 50% crash faulty nodes. Since in a private Blockchain all the peers are verified by the administrator, this algorithms aims at resolving crash faults other than Byzantine faults [58].

9.9 Study Methodology and Experimental Setup

With respect to our first objective, our intention is to evaluate the performance and scalability of the various consensus algorithms in Hyperledger Sawtooth to guide the right decisions and to motivate the design for our self-adaptive system. As we mentioned before, Hyperledger Sawtooth has four main consensus protocols, namely Raft, PoET, Devmode, and PBFT.

As Devmode is only considered for testing purposes, but not for the production network, we decided to experiment only with the other three protocols. The three algorithms were compared based on their resource consumption and latency in the context of a simulated IoT application.

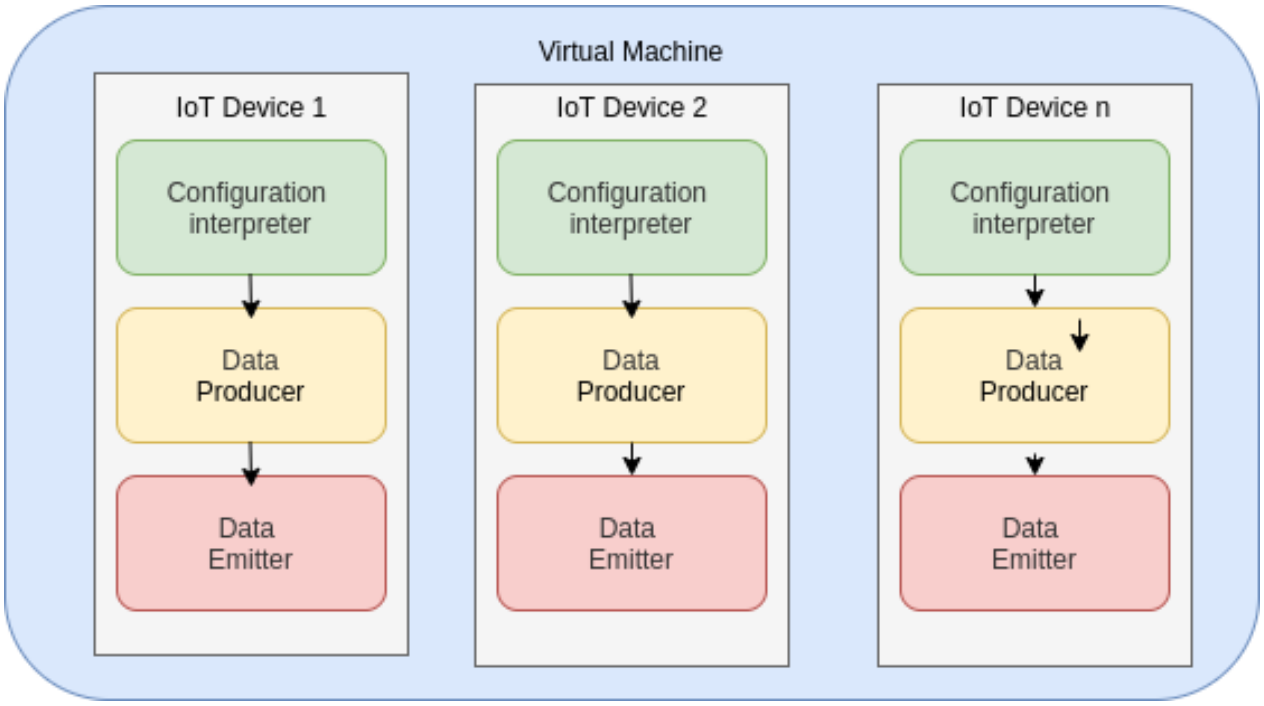


Figure 9.1 Simulated IoT environment

To evaluate the performance of the system, we created a custom workload generator that simulates data generated by sensors in a smart home environment. In our simulated environment, we assume that the smart home is equipped with several IoT devices, including surveillance cameras, thermostats, door sensors, GPS sensors and sound sensors (capturing random noise). The devices differ with respect to the transmission rate and the amount of data transmitted. For example, GPS sensors send a position report every 2 seconds, thermostats report the temperature every 1 second and door sensors report the state of a door (Open / Closed) every 5 seconds. The size of data is ranging from few bytes to few kilobytes based on the type of sensor. These devices would send their data to a gateway and the gateway pushes the data to a Blockchain database, which in our case is a Hyperledger Sawtooth network. Figure 9.1 shows the architecture of our simulated IoT lab, which uses python scripts and Docker containers to simulate each device, designed according to instructions set forth by Ramprasad et al. [135]. We specify the number of IoT devices, type of devices (camera, thermostat, GPS, door sensor device and sound device), time-scheduling in the Configuration

Interpreter, the Data Produces component produces the data and Data Emitter transfers the data to the Blockchain for storing the produces data. To implement the virtual IoT environment, every sensor is implemented as a python script, with a message that corresponds to the appropriate size of data and transmission rate, deployed on a Flask web server hosted by a Docker container. In addition, we use Nginx as the web server that implements the gateway and accepts data from the sensors and HAProxy is a load balancer deployed in front of the Blockchain. The devices first generate and transmit their data to the gateway, as presented in Figure 9.2, and then the gateway sends it to the Sawtooth network to be stored securely.

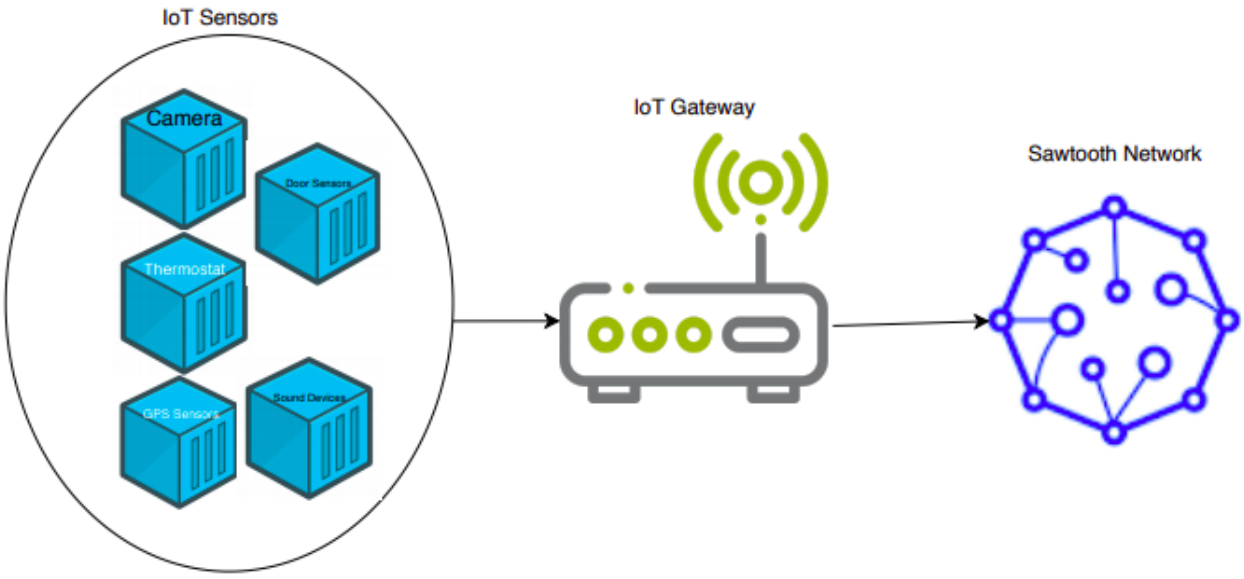


Figure 9.2 Architecture of experimental setup

Figure depicts that our study took about 30 minutes. During our experiment, the number of IoT devices continuously fluctuates to periodically saturate the system resources. The change in the load is that which will trigger the adaptation and will stress the Blockchain network. Besides that, We have set the configuration of the IoT sensors constant throughout the study, since our aim is to evaluate the impact of load on the consensus protocol in Hyperledger Sawtooth and identify the performance of each consensus protocol. For this reason, we have tried to eliminate the effect of different configurations.

In our experiments, the number of sensors varies over time but the distribution of sensors and devices is kept relatively stable: 20% cameras, 30% thermostats, 10% GPS, 15% door sensor devices and 25% sound devices. Figure 9.3 shows that our experimental study takes about 43 minutes per experiment. During this time, the number of IoT devices continuously increases

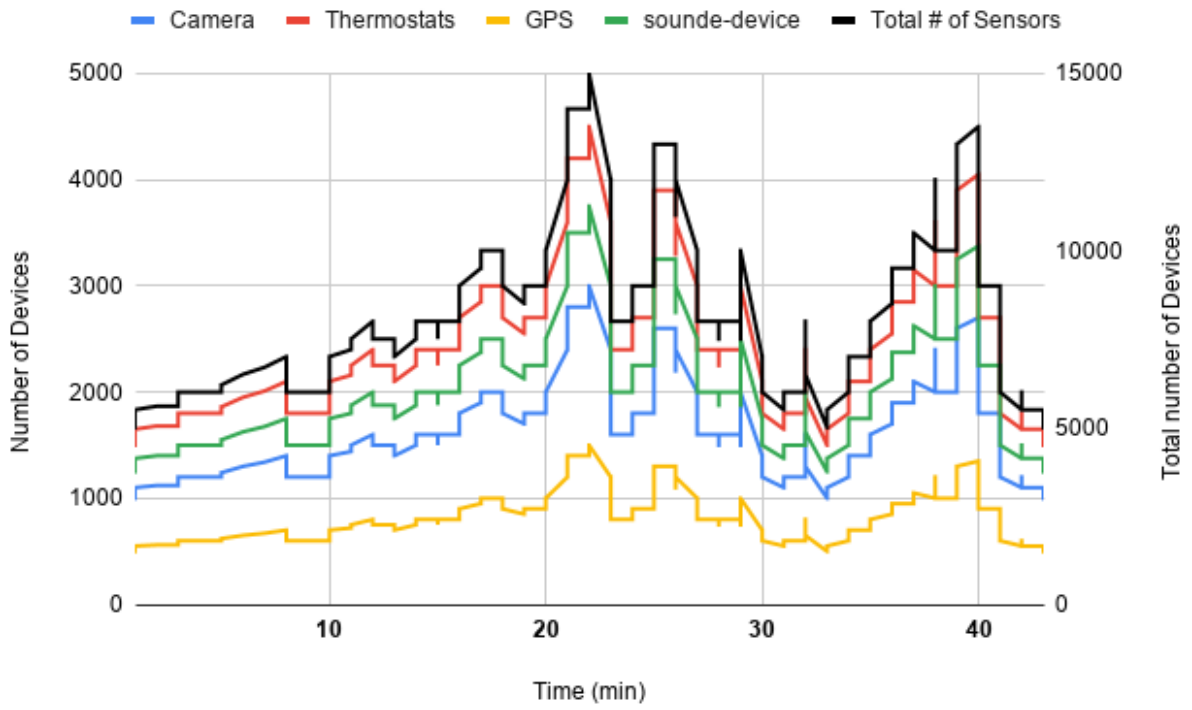


Figure 9.3 The distribution of devices over time

and decreases to periodically stress and saturate the network. The change in load is that which will trigger the adaptation and will stress the Blockchain system. Besides that, the configuration of the IoT network is kept constant throughout the experiment, since our goal is to examine the impact of load on the consensus algorithm and by extent on the performance of the Blockchain, so we try to eliminate the effect of different configurations.

The data we are storing in our scenario includes: Device ID, Time Stamp, Message, Message Size, Message Hash and Device Type. It is important to mention that the purpose of this IoT simulation is only to saturate the network with random data as much as possible to evaluate the performance of the consensus protocols in the same scenario. However, in a real case, in order to reduce the size of the network, we could store data off-chain and record only the hash on Blockchain for immutability, aggregate the data and store only relevant portions on Blockchain. Since we need to verify all the data by all validators, it will not be suitable for large volumes of data on a permanent Blockchain store, especially those that may by relevant only for a short time. In our experiments, our goal is to cause as much saturation as possible, so we send and store all data, under a “worst case scenario”.

Figure 9.4 demonstrates the Hyperledger Sawtooth infrastructure and its components as

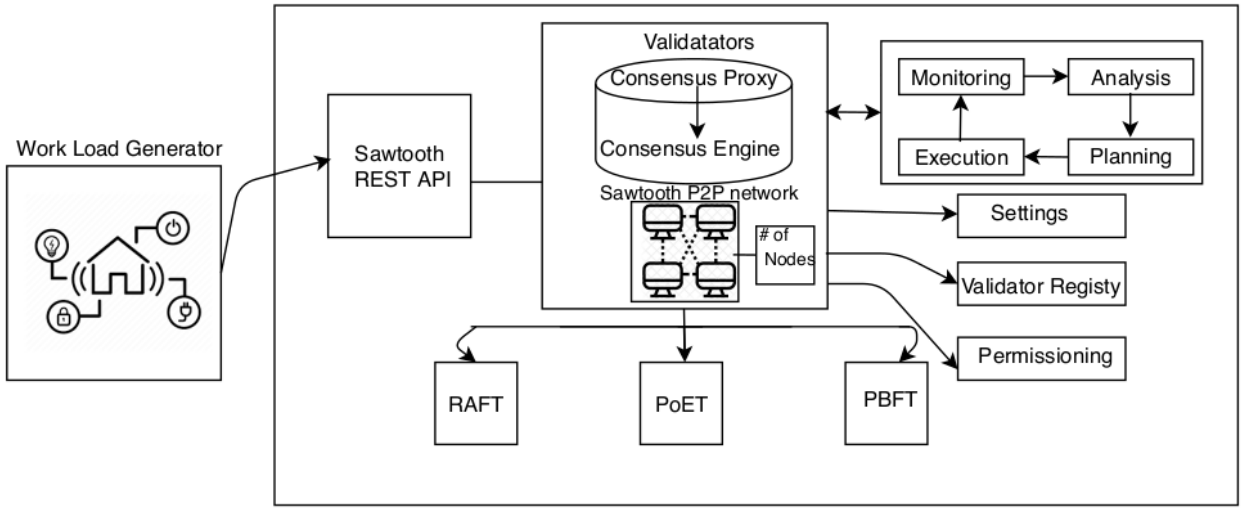


Figure 9.4 Our Sawtooth Network

it was used in our experiments. We used multiple containers to deploy Sawtooth as the default framework consists of multiple processes, including among others the supply Rest API, validator nodes, the consensus engine, a Postgres database, and others. By default, the consensus protocol is designed to be "Devmode", which is a simplified random-leader consensus algorithm mostly for testing purposes. For our experiments, we have made some small modifications to Sawtooth to enable dynamic configuration of the consensus protocol [35]. This allowed us to evaluate the different consensus algorithm under our simulated algorithm, but it also enabled the implementation of our self-adaptive system. We have applied the default configuration for each consensus algorithm based on Sawtooth documentation [35] guidelines.

The infrastructure of our experiment was deployed on the Amazon EC2 cloud. One virtual machine with 8 VCPUs and 15 GB RAM (rc4.2XLARGE) was used to deploy the Hyperledger Sawtooth. Docker and Docker-compose were installed in this machine to enable the use of containers. The containers that corresponded to validators, which run the consensus protocol, and were hosted by this VM were limited to use only 25% of the total CPU of the host VM. It has been shown that when left unrestricted containers can take up all the resources of the host VM, rendering scaling ineffective [229]. Another VM with 2 VCPUs and 8 GB RAM (T2.Large) was used for the simulation of the IoT workload. Both machines ran Ubuntu 18.03. As mentioned earlier, we evaluated performance in terms of CPU consumption and response time. For CPU, we used the Docker Remote API [136] to collect CPU consumption for the Sawtooth network. For response time, the python script for data emission by every sensor was configured to receive back a response from the Sawtooth network and record the response

time.

9.10 Self-Adaptive System Design

In order to implement our self-adaptive mechanism to enable the dynamic configuration of the consensus protocol, we followed the MAPE-k architecture [107] with four components: a) a monitoring component which gathers CPU data from Docker and response time using our custom script, b) an analysis component that checks if the CPU has crossed the predefined thresholds, c) a planning component that will decide to add or remove a validator, based on CPU consumption, and to dynamically change the consensus algorithm, and d) an execution component that connects to Docker to scale the validators or to Sawtooth to change the consensus algorithm. We have integrated our MAPE-k framework to the Sawtooth network as illustrated in Fig. 9.4.

The aim of our MAPE-k is not designed to neither detect nor mitigate the DDoS attack. Whilst the system is under DDoS attack, by adding more containers and validators, we are benefiting the attackers, and not adding the container will cause packet loss and high response time. As a results, we are designing a self-adaptive mechanism to maintain the high performance, better response time and lower packet loss.

The analysis component checks the average CPU consumption of the Sawtooth validator containers and if that surpasses 70%, our MAPE-k system automatically adds a new validator node in the network. Preliminary tests have shown that when there is an increasing number of peers in the network (sensors in our case), Sawtooth can handle the traffic to validators and it crashes. Increased CPU consumption can be an indicator for impending crashes, so scaling of the validators is an appropriate action. Conversely, when average CPU drops below 30%, the analysis component signals the removal of a validator in order to reduce costs, but also to speed up the validation process as less nodes are necessary for a reduced traffic.

The planning component has to make two decisions. The first concerns the scaling of the validator cluster as described above. The second decision concerns the choice of a consensus algorithm. Preliminary results have shown that the size of the network and the number of validators impacts the performance of consensus differently for each algorithm. According to Sawtooth documentation, each consensus protocol has different communication complexity. As was mentioned earlier, the number of nodes in the network has an impact on the performance of the system and we are interested to experimentally measure the threshold when the performance of the system decreases and at the same time to see which consensus protocol outperforms when adding more nodes. The planning phase for scaling and for the choice of

consensus algorithm is described in Algorithm 2. The thresholds of validator nodes to trigger a change in the consensus algorithm were determined empirically, based on the results of our study, as presented in Section 9.11.

Performance of the distributed systems, and IoT applications in particularly is significantly vital, however the main objective of integrating with Blockchain is to guarantee the confidentiality, integrity and availability of the system. We have conducted relevant studding and corresponding documentation to evaluate the security capabilities of each consensus algorithm and build our planning component accordingly. PoET algorithm seems to be the most vulnerable protocol since an adversary can jeopardize the integrity of the system by compromising $\theta(\frac{loglogn}{logn})$ [19]. This implies that this algorithm will be more secure when our system is particularly large. However, based on the results demonstrated in 9.11, PoET is the most performance algorithm among the others in terms of CPU utilization, packet loss and response time. The Raft algorithm can tolerate 51% crashed faulty nodes and PBFT algorithm can tolerate 1/3 of malicious nodes [210].

It is worth noting that while performance is a crucial aspect of distributed systems, and IoT systems in particular, the main motivation behind using Blockchain is security. We have studied relevant studies and corresponding documentation to assess the security capabilities of each algorithm and design our planning module accordingly. PoET is deemed the most vulnerable algorithm since an adversary can jeopardized the integrity of the network by compromising $\theta(\frac{loglogn}{logn})$ of the total participating nodes in the network [19]. This implies that this algorithm can be more secure when the number of nodes is relatively large. However, based on our experimental study (see Section 9.11), PoET is the most performant of the three in terms of resource consumption and average response time even when the network scales up to a large number of nodes. The Raft algorithm can tolerate 50% crashed faulty nodes and PBFT can tolerate 1/3 of faulty nodes [210].

Based on our performance results, Raft algorithm performance (in terms of both resp one time and resource consumption) reduces when it scales to 9 nodes thereby, PBFT can perform better than RAFT even with more nodes. For this reason in our self-adaptive mechanism, we start with Raft algorithm, when it reaches to 9 nodes we switch to PBFT and when PBFT reaches to 15 nodes as both resource consumption and response time start worsening we will switch to the PoET as it can assure the security with a large number of nodes.

Algorithm 2 Self-Adaptive framework for Hyperledger Sawtooth for choosing the right consensus algorithm

Input: *Nodes — the number of Validator nodes in the Sawtooth network*
Input: *Consensus-Algorithm - Retrieves the current Consensus Algorithm of the Sawtooth Protocol*
Input: *utilization — the average CPU utilization of containers in a VM*
Input: *lower threshold and upper threshold — the limits of the desired range for the CPU consumption of Validator containers in a VM*
if if utilization ≥ 70 **then**
    Validator = Validator + 1;
else if utilization ≤ 30 **then**
    Validator = Validator - 1;
Consensus-Algorithm = Raft
if Nodes < 9 **then**
    Consensus-Algorithm = PBFT;
else if Nodes ≥ 9 **then**
    Consensus-Algorithm = Raft;
else if Nodes > 14 **then**
    Consensus-Algorithm = POET;
else if Nodes ≤ 14 **then**
    Consensus-Algorithm = PBFT;
else
    Consensus-Algorithm = Raft;

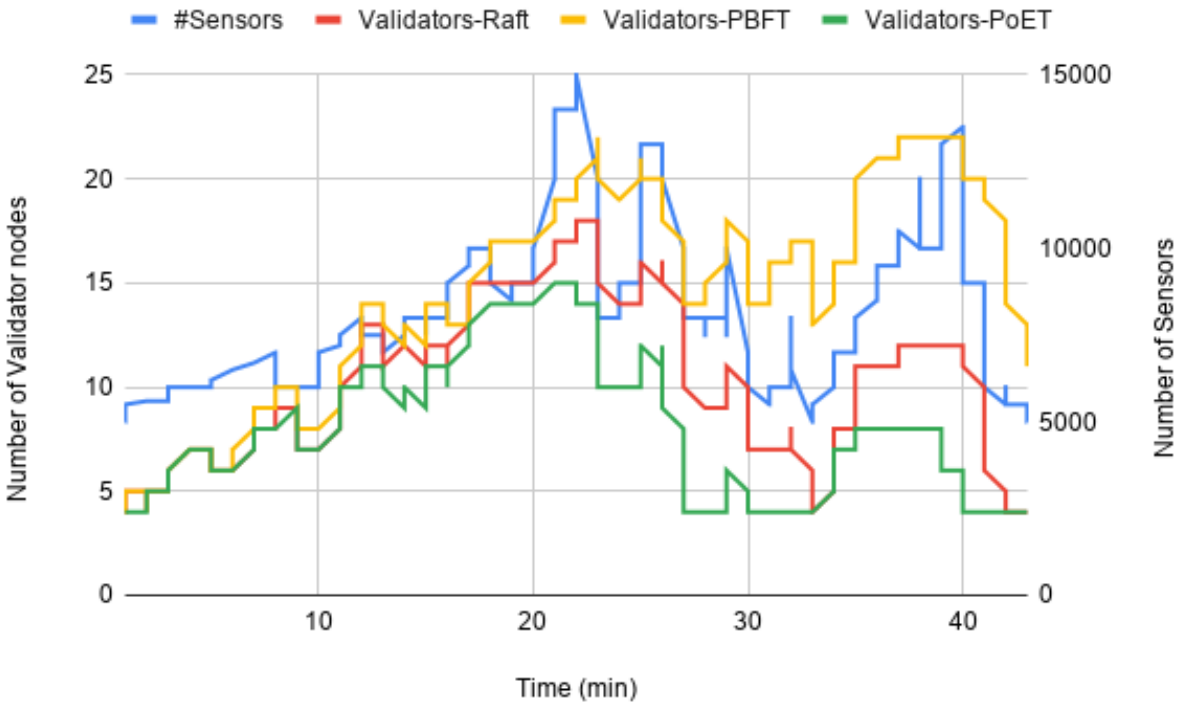


Figure 9.5 The distribution of number of sensors to all three validators for the studied consensus algorithms

9.11 Experimental Results

9.12 Performance Evaluation of consensus protocols in Hyperledger Sawtooth

The first set of experiments concerns the study of how each consensus algorithm performs in terms of CPU consumption and response time under a varying number of peers in the Blockchain network. Since we know that none of the configurations automatically scales according to the size of the network and we also know that if the validators are not scaled, the network crashes, we enabled a simple scaler (lines 5-8 of Algorithm 2) to protect the system and be able to continue our experiments. Figure 9.5 shows the number of validators for Raft (red line), PBFT (yellow line) and PoET (green line) against the number of sensors (blue line) representing the size of the Blockchain network. As it is obvious from the graph, the number of validators increases and decreases according to the corresponding fluctuations to the size of the network. However, it is also obvious that PoET consistently need less validators compared to the other two algorithms for the same number of sensors, while we find PBFT at the opposite end of this comparison. In fact, as it is summarized in Table 10.2, PoET needs

an average of 7.5 validators throughout the experiment, compared to 15.3 for PBFT and 10 for Raft. At the peak of the traffic (just after the 20th minute of the experiment), PoET requires a maximum of 15, while PBFT and Raft require 22 and 18 respectively. Already, this shows that PoET is a close optimal choice concerning cost-efficiency and resource utilization.

Table 9.1 Summary of results of comparison between Raft, PBFT and PoET

Algorithms	Raft	PBFT	PoET
Avg Validators	10	15.3	7.5
Max Validators	18	22	15
Avg CPU	51.26%	55.70%	45.72%
St.Dev. CPU	24.92	23.14	24.44
Avg Response time (ms)	874.12	853.04	759.14
St.Dev. Response time	100.96	122.64	106.62

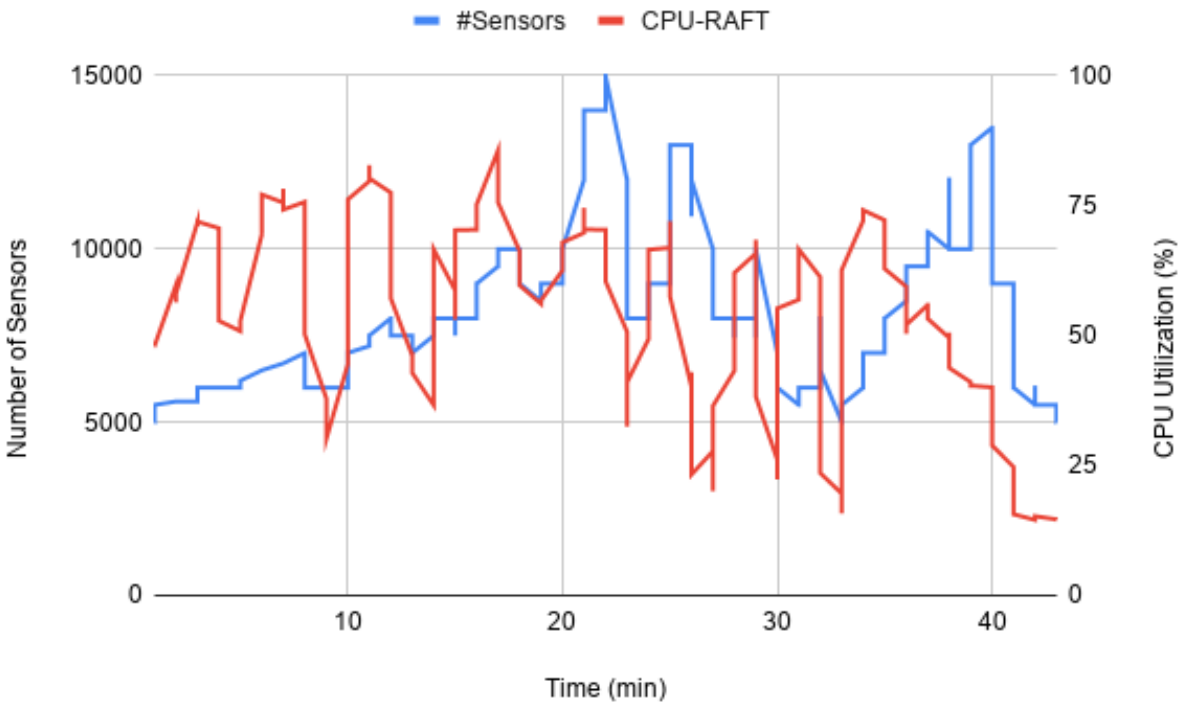


Figure 9.6 Raft CPU Utilization VS Number of Sensors

When looking at CPU consumption, results do not differ much. PoET is by far the most efficient algorithm recording an average CPU consumption of 45.72%. At the other end, PBFT is still the worst of the three with 55.7%, while Raft is the middle solution with an

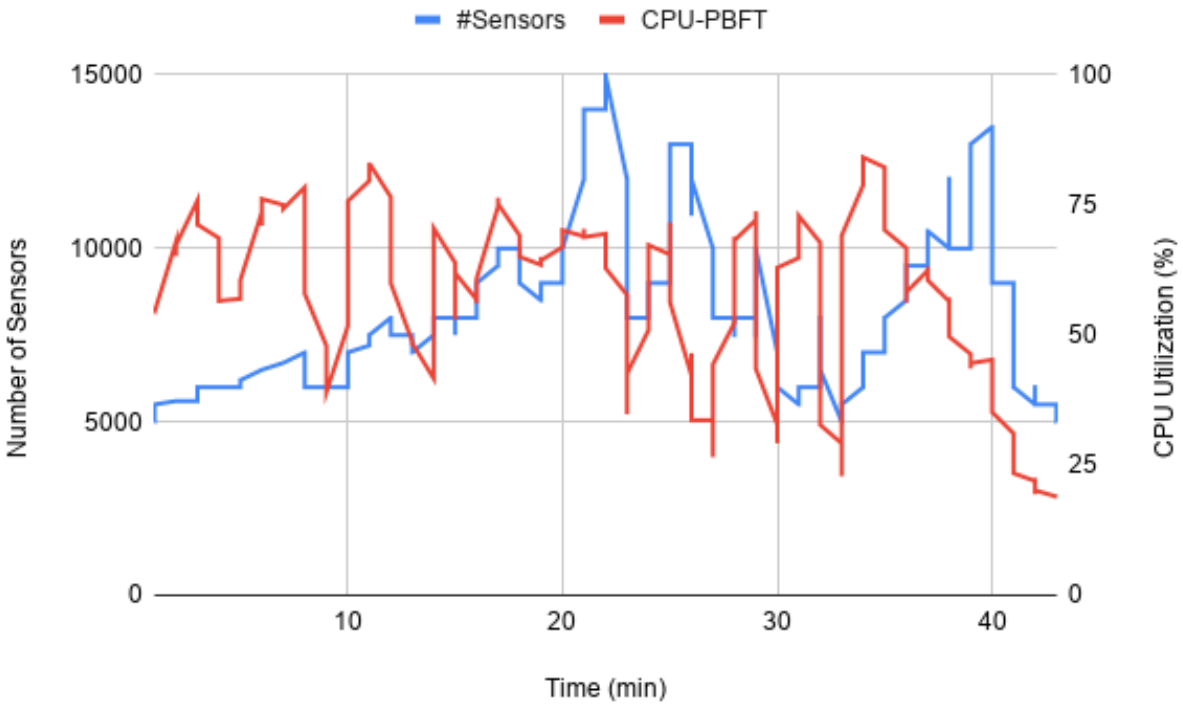


Figure 9.7 PBFT CPU Utilization VS Number of Sensors

average CPU consumption of 51.26%. Figures 9.7, 9.6, and 9.8 show graphically the progress of CPU consumption for PBFT, Raft and PoET respectively. One first observation is that CPU *oscillates* (drastically fluctuates), especially close to scaling action. The reason for this is that the impact of a scaling action is prominent to the output (in our case, CPU) especially right after the action and especially if the system is small (as is the case for the number of validators in our experiments). This condition is systemic throughout our experiments, in the sense that it is present in all three algorithms, so it does not affect our conclusions. To further minimize this impact, we should not that after taking any scaling or reconfiguration action, our analysis freezes for 10 seconds to allow the system to settle and avoid opposite scaling actions immediately after. Nevertheless, our monitoring module is not stopped, which is the reason why the oscillations are visibile in the graphs.

Finally, PoET is also the winner when considering response time with an average of about 760 ms. However, in this case we have a change for the second most performant algorithm, where PBFT had 853 ms average response time, while Raft had 874 ms. Nevertheless, PBFT had a more variance (122 standard deviation) compared to Raft, which implies that their difference may be neither significant nor consistent. The difference between PBFT and Raft is visible in

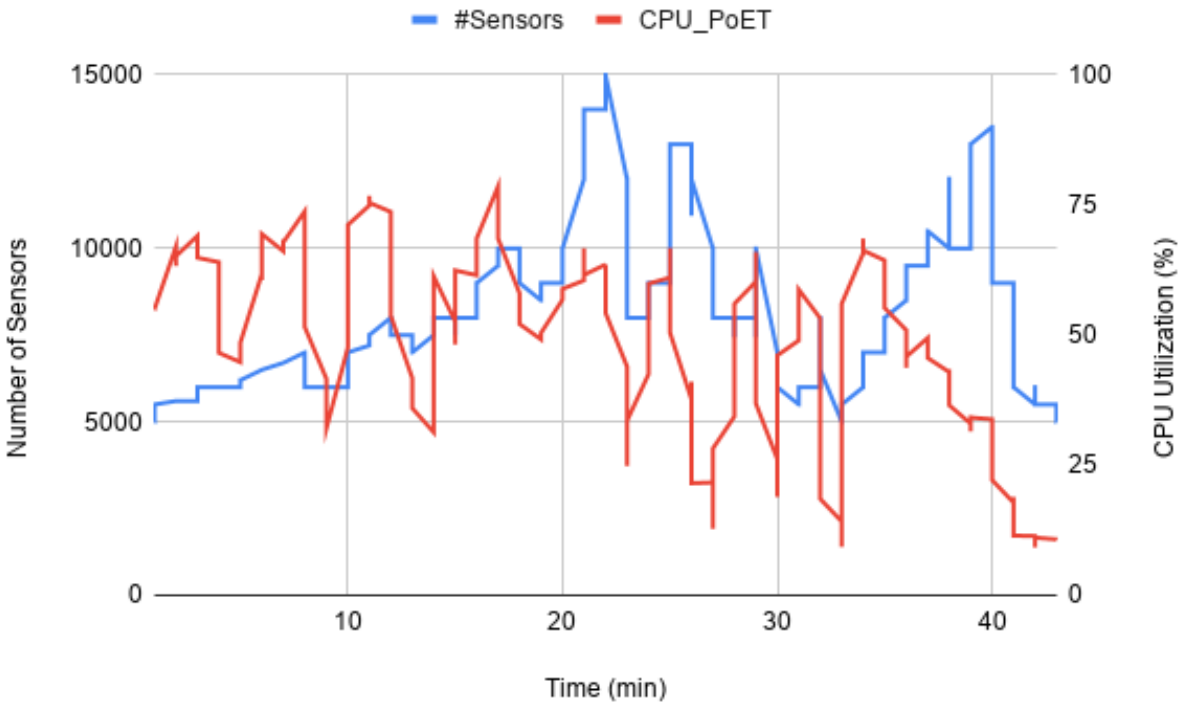


Figure 9.8 PoET CPU Utilization VS Number of Sensors

Figures 9.10 and 9.11 respectively, where we see that in Raft response time stays higher for a longer period. Figure 9.9 shows clearly that PoET maintains low response time at all times.

Looking at Figures 9.9, 9.10, and 9.11 with the progress of response time for PoET, PBFT and Raft respectively, we can observe that scaling validators has a positive impact on response time as the size of the network grows, until we reach a certain point where there is a plateau and the response time remains at least constant even when we keep adding validators. The interesting observations is this point is different for each algorithm. We found that the plateau is reached for about 9 validators for PoET and Raft and for almost 14 validators for PBFT. This also roughly corresponds to the average number of validators necessary for each algorithm. We can see that from this perspective PoET is once again the most efficient of the three algorithms. However, as mentioned before, PoET is not as robust and secure with a small number of validators. Given its increased security thanks to higher number of validators and its ability to maintain good performance under this condition, we argue that PoET is the recommended consensus algorithm, when the size of the network increases considerably.

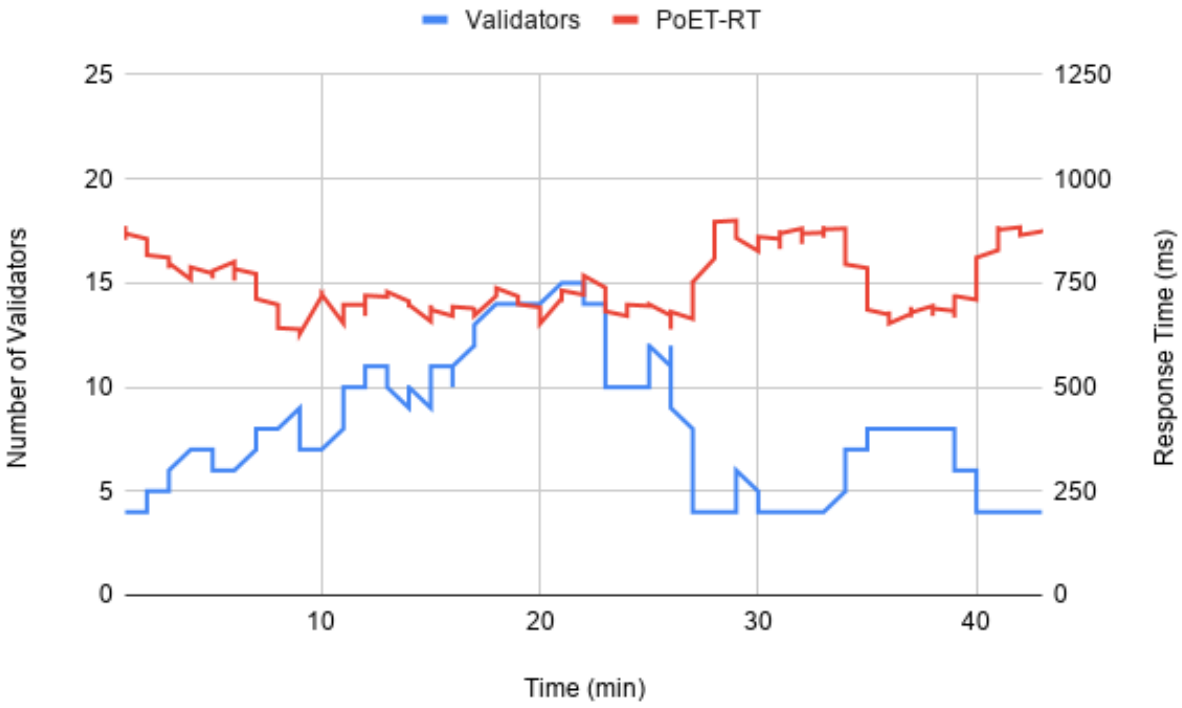


Figure 9.9 PoET Response Time VS Number of Validators

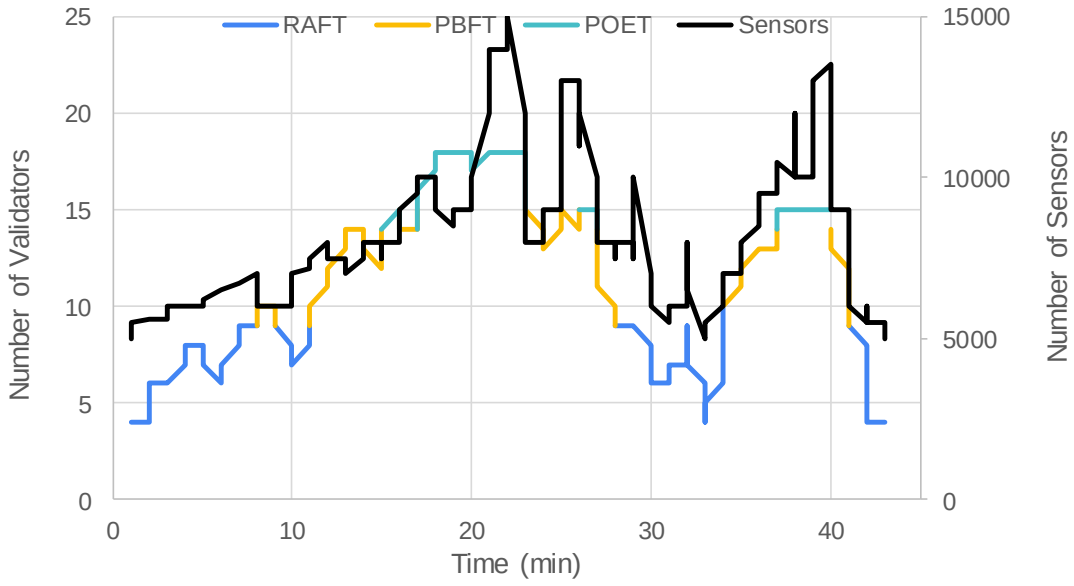


Figure 9.12 Number of validators VS number of sensors. The change in color indicates a switch of consensus algorithm by the self-adaptive mechanism.

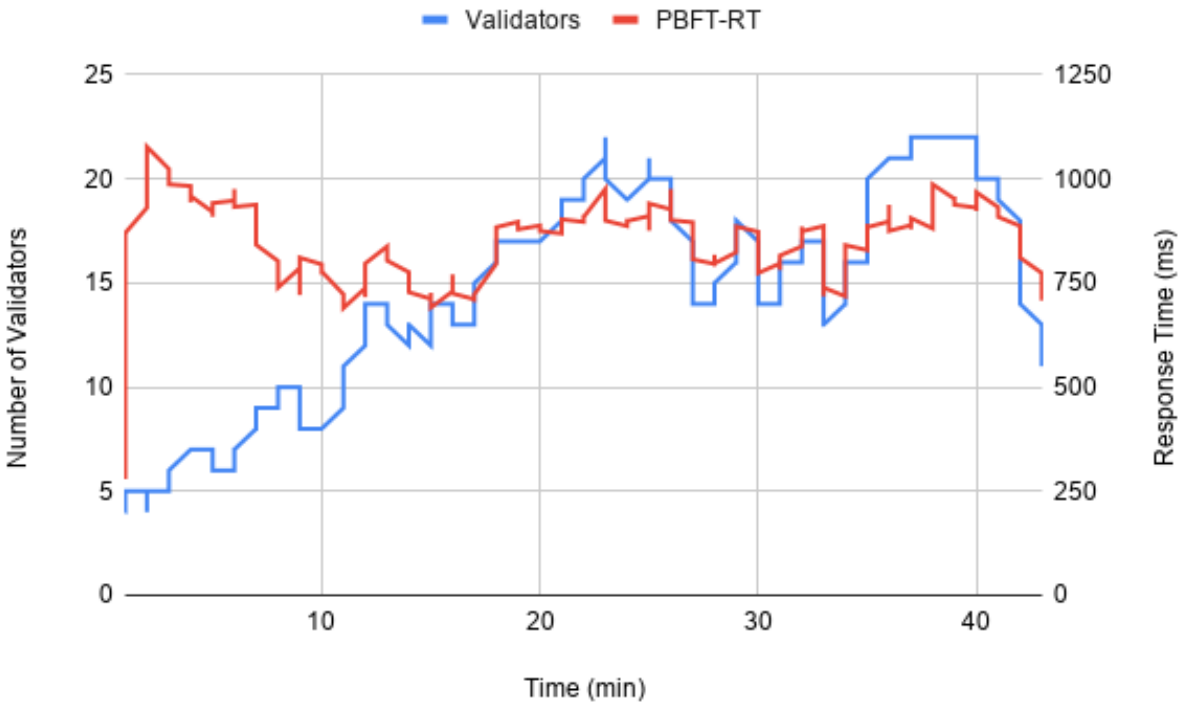


Figure 9.10 PBFT Response Time VS Number of Validators

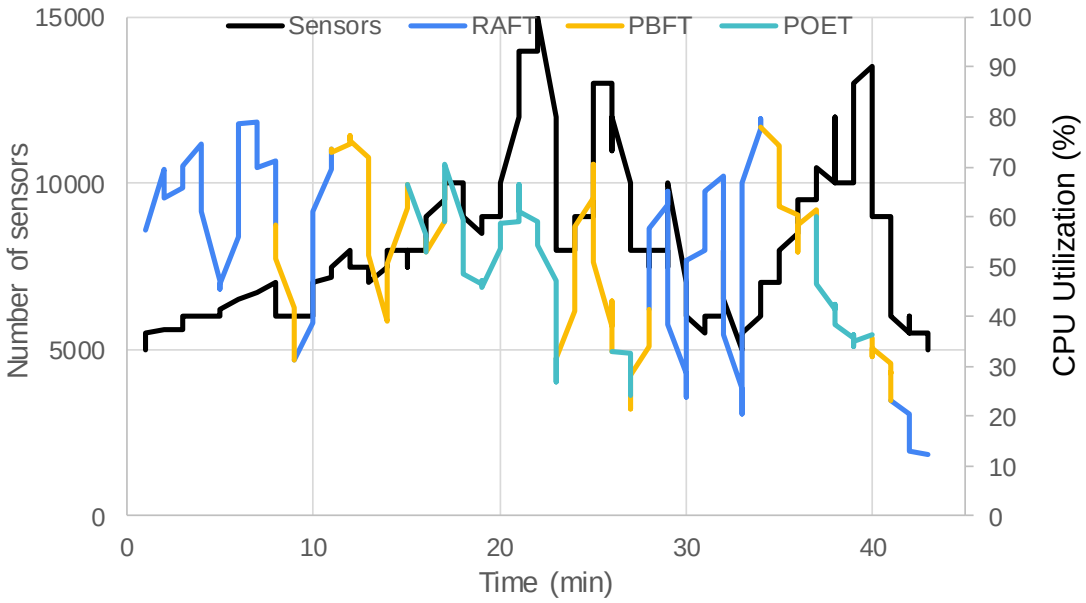


Figure 9.13 CPU Utilization VS Number of Sensors. The change in color indicates a switch of consensus algorithm by the self-adaptive mechanism.

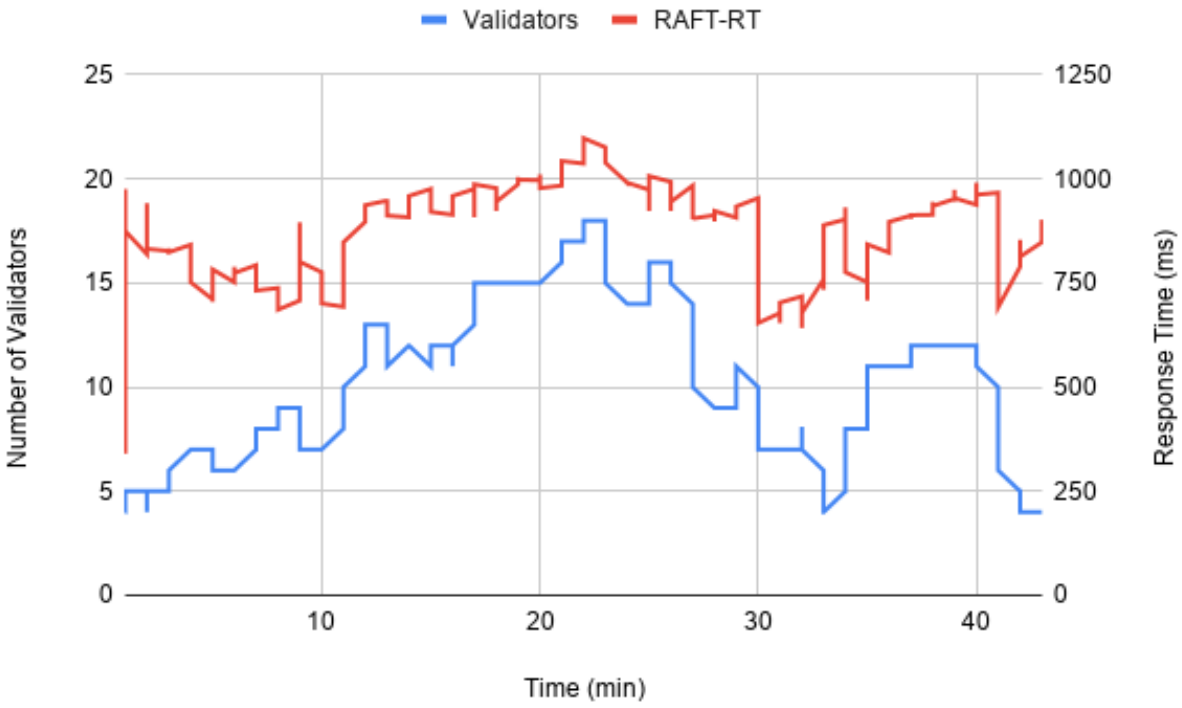


Figure 9.11 RAFT Response Time VS Number of Validators

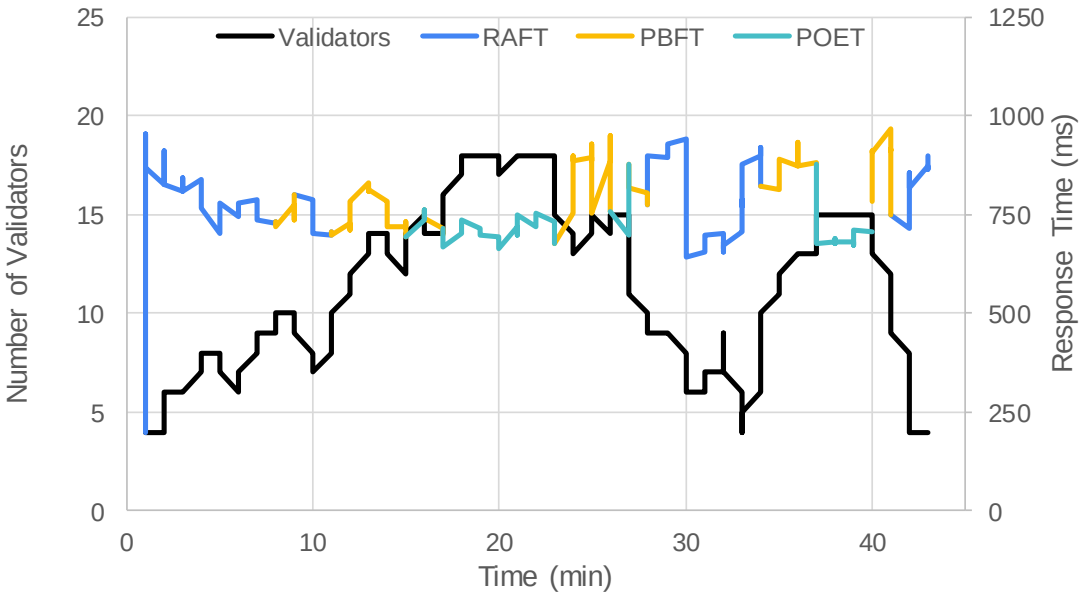


Figure 9.14 Response Time VS Number of Validators. The change in color indicates a switch of consensus algorithm by the self-adaptive mechanism.

The findings of this study helped us to design our self-adaptive mechanism to dynamically reconfigure the consensus protocol of Hyperledger Sawtooth at run-time. When we have a small-sized network and a small number of validators (less than 9) is sufficient, our MAPE-k tool picks the robust PBFT algorithm. For medium-sized networks where less than 14 validators are necessary, MAPE-k picks the Raft algorithm. Finally, when the required validators exceed 14 and we need to maintain good performance, MAPE-k recommends the use of PoET (see Algorithm 2).

9.13 Evaluation of the Self-Adaptive mechanism

When testing our self-adaptive mechanism, we enable in the planning module the dynamic reconfiguration of the consensus protocol (lines 9-19 in Algorithm 2) on top of the simple scaler (lines 5-8 in Algorithm 2). By default our MAPE-k system starts with Raft as the consensus algorithm and changes based on the number of validator nodes deployed in the Sawtooth system. Figure 9.12 shows the number of validator nodes (blue for Raft, yellow for PBFT and green for PoET) deployed as the response to the fluctuation in the number of sensors (black line). As it can be seen, the line for valdiators is colored differently for each consensus algorithm and a change in color indicates a decision to switch algorithms. For example at time 8 minutes, the consensus has been changed from Raft (blue) to PBFT (yellow). In addition, Figures 9.13 and 9.14 show the CPU consumption and the response time of the system, when of our MAPE-k system is deployed. Finally, Table 9.2 summarizes the results of this experiment.

As it can be seen, our adaptive consensus protocol performs exceptionally well compared to Raft and PBFT, while its performance is comparable to that of PoET, 11 validators on average against the 10 of PoET, 50% average CPU consumption against 45% of PoET and 784 ms response time compared to 759 ms of PoET. The additional benefit, as it is obvious from the graphs, is that PoET is deployed at higher traffics with a large number of validators, exactly when it is the most powerful.

Table 9.2 Summary of experimental results for self-adaptive system

Algorithm	SA
Avg Validators	10.98
Max Validators	18
Avg CPU (st.dev.)	49.96% (22.57)
Avg Response Time (ms) (st.dev.)	784.22 (97.49)

9.14 Threads to Validity

In this study, we have designed a self-adaptive mechanism that can efficiently choose and deploy a different consensus algorithm at run-time. To evaluate the system, we have simulated a virtual IoT lab environment that aims to store its generated data in the Blockchain database. For our future study, we plan to use real IoT devices that will produce and send their actual data to the Blockchain server to see how it impacts the network. In this study, we experimented with synthetic data to better control the environment. There can exist other parameters that can impact performance, such as such as messaging format (JSON vs. Protobug), communication protocols (such as GRPC), security, or any other unique workload characteristics of IoT devices. It is our intention to include these factors in other future studies.

Another threat to validity could be the number of considered consensus protocols. We specifically chose to focus on PBFT, PoET and Raft, because these are all supported by Hyperledger Sawtooth. By using a single Blockchain platform, thus a homogeneous hosting platform, we eliminated the impact of other factors and studied only the impact of the consensus protocol. Naturally, our intention is to study more algorithms in the future.

To conduct study, we have used the suggested default configuration of consensus protocols in the Hyperledger Sawtooth network. It is expected that special configurations may have render an algorithm much more efficient than we reported in our study. By keeping the same default configurations for all algorithms, we made sure that the comparisons were fair and under the same given conditions for all algorithms. The exploration of different configurations is another possible future step of this work.

Lastly, for our experiment , we have used a single VM, which implies the network I/O is negligible. For future studies, we plan to use different VMs for the nodes and see the effect of the network I/O on the performance.

9.15 Conclusion

Although Blockchain is an emerging technology among practitioners to improve security issues for distributed systems, it could also prove inefficient in terms of bandwidth and cost, especially for IoT systems. In this study, we first compared three popular consensus algorithms for Blockchain, namely PBFT, Raft and PoET, to see whether there is a single best with respect to time and computation overheads or if there are trade-offs between the three consensus protocols in Hyperledger Sawtooth. From the empirical analysis, we have observed that PoET is the most efficient consensus algorithm comparably to RAFT and PBFT, but

probably not as robust and secure for small networks. Conversely, the performance of Raft and PBFT deteriorates for larger-sized networks.

Based on our performance evaluation and theoretical security analysis of the consensus protocols, we have designed a novel self-adaptive mechanism to dynamically and automatically choose and deploy the right consensus algorithm at run-time. Our results demonstrate that our self-adaptive mechanism has better CPU utilization comparably to PBFT and RAFT, and average Response Time is also better than both RAFT and PBFT. In theory, our adaptive consensus protocol addresses the security limitations of PoET for small networks while maintaining the same performance standards for larger networks. In the future, we plan to extend our study to also include these security concerns in a concrete manner.

In this study, we have focused on reducing the cost and assessing the security based on the number of nodes. For future research, we are planning to explore how the framework can be used if the optimization target changes regarding other security aspects (e.g., the ones related to Blockchain forks). Moreover, another interesting subject to explore for future studies is to analyze algorithms' performance concerning the number or frequency of the transactions and possibly the characteristics of these transactions.

CHAPTER 10 ARTICLE 7: INTELLICHAIN: AN INTELLIGENT AND SCALABLE FRAMEWORK FOR DECENTRALIZED APPLICATIONS ON PUBLIC BLOCKCHAIN TECHNOLOGIES: AN NFT MARKETPLACE CASE STUDY

10.1 Paper Information

Title

IntelliChain: An Intelligent and Scalable Framework for Decentralized Applications on Public Blockchain Technologies: An NFT Marketplace Case Study

Authors

Mohammadreza Rasolroveicy, and Marios Fokaefs

Date of submission: 2022/11/14

Submitted to: IEEE Transactions on Emerging Topics in Computing

Chapter Overview: This Chapter presents *IntelliChain*, a self-adaptive framework for public blockchains.

IntelliChain consists of two main features. First, IntelliChain minimizes errors and transaction retires and increases throughput by accurately predicting the required transaction fee for various public blockchains. Second, it allows to dynamically transition between public blockchain platforms depending on the cost of recording transactions and the number of errors based on pending transactions and block size in blockchain gas stations.

IntelliChain currently supports three public platforms, including Avalanche, Polygon, and Fantom. In IntelliChain we used different machine learning and neural network models to predict the blockchain transaction fees and errors. Based on the dataset, we picked the fittest model. Our empirical results have shown that IntelliChain is faster by two hours and has 7 times fewer errors in comparison with the original configurations.

This study covers two of the five dimensions of blockchain performance we explore in this thesis, namely consensus, and comparison of public blockchains.

10.2 Abstract

Non-fungible tokens (NFTs) have been attracting the interest of both technical and non-technical parties, including collectors and traders, among others. The amount of transactions

in NFTs surpassed $50 billion in 2022. Blockchain technology's advantages as a distributed, immutable, and transparent database make it ideal for verifying the ownership of digital goods created by their producers. On the other hand, high computation and transaction costs are known disadvantages of public blockchain networks like Ethereum V1, used in NFT marketplaces. To address these inefficiencies, other public blockchain systems have emerged as replacements for NFT marketplaces, each with its own unique properties. When planning such an NFT exchange, it is vital, but not trivial, to select the most appropriate public blockchain platform. In this work, we make two contributions to support this decision. In this paper, we present *IntelliChain*, a self-adaptive framework that can predict the best optimal transaction fees (also known as gas fee) for blockchain to reduce the errors and also an ability to switch the public blockchain-based dynamic needs such as transaction fees and stability of the network.

10.3 Introduction

Non-fungible tokens (NFTs) have been experiencing an enormous momentum among traders and digital asset collectors since early 2021. An NFT is an encrypted token representing a digital asset that can be traded or exchanged [31]. NFTs are cryptographic assets with distinct identification that can be neither exchanged nor traded equivalently. Therefore, NFTs are distinguishable, and their ownership can be traced and tracked [31]. Digital assets can be tokenized through smart contracts on the blockchain network to assign the original ownership and manage the transferability of an NFT ownership [31]. Different kinds of intangible digital assets can be tokenized, such as pieces of music, digital art, book, or movie. Tokenizing digital assets enables traders and collectors to buy, sell, and trade their NFTs more efficiently to reduce the possibility of alteration and Forgery.

Smart contracts are simply a piece of code permanently stored on the blockchain and can be automatically executed when predetermined conditions are met [79]. The most important feature of a smart contract is that the code can be executed and enforced among untrusted parties without the involvement of a trusted party. Transactions of smart contracts are permanently stored, replicated, and updated in an immutable and transparent blockchain as a distributed and decentralized database [230]. The ability to deploy smart contracts in Ethereum has attracted several sectors including government, energy, the Internet of Things, health, and art, among others [231].

The emergence of NFTs in 2017 caused Ethereum, a public permissionless blockchain platform, to increase its popularity as it became the platform of choice for artists and digital content creators to store their digital assets on an immutable Ethereum ledger and demonstrate the

proof of ownership and authenticity of their products to their clients.

There are different standards for NFTs in the Ethereum community such as ERC721 [1] and ERC1155 [2]. Minting [31] an NFT means that a unique token has been created in a unique smart contract as a transaction. Minting also involves recording information on the digital asset represented by the NFT on the blockchain, including the original creator, timestamp and unique token ID, and a URL to the digital asset metadata. The metadata can include names, descriptions, and image/video/music URLs in a decentralized storage such as IPFS (InterPlanetary File System) [80]. After minting the NFT, it is possible to transfer the NFT ownership to someone else. In the blockchain, the history of NFT ownership is transparent and traceable. In order to mint and store an NFT in the blockchain, a transaction fee needs to be paid to the host blockchain, which then will be distributed among the nodes in the blockchain as compensation for the time and computational resources that they used for minting the NFT. The transaction fees for the minting and for storing the NFT in the blockchain should be paid in the native cryptocurrency of the blockchain. Therefore, the NFT creators must own a crypto wallet to pay these fees. Moreover, transferring ownership and intellectual rights of an NFT through a smart contract requires initiating another transaction in the blockchain, which needs to be paid once again through the NFT owner's crypto wallet. All the transactions on the blockchain are immutable, time-stamped, and transparent.

The traditional Ethereum V1 network employs a Proof of Work (PoW) consensus mechanism to maintain the security and integrity of its peer-to-peer network, resulting in high transaction fees, processing time, and latency overheads for real-time applications that need a considerable quantity of data [16,232]. Furthermore, despite being one of the most decentralized and secure blockchain platforms with smart contract capacity, it has been criticized for using 104.12 TW/h, which is four times the usage in the entire country of Portugal [3] [4]. For these reasons, several alternative public platforms with distinct characteristics in terms of architecture, novel consensus algorithm, availability, and performance, such as Polygon [5], Fantom [6], and Avalanche [7] have been proposed to overcome the limitations mentioned above in the Ethereum network. The majority of these platforms are forks of the leading Ethereum network, with merely the consensus protocol replaced by more cost-effective algorithms.

In the context of NFT, the two most important shortcomings of current blockchain platforms

[1] https://eips.ethereum.org/EIPS/eip-721
[2] https://eips.ethereum.org/EIPS/eip-1155
[3] https://tinyurl.com/22mzwf8z
[4] https://statista.com/statistics/280704/world-power-consumption/
[5] https://polygon.technology/
[6] https://fantom.foundation/
[7] https://avax.network/

are the limited scalability and the unpredictable and highly volatile transaction fees. Scalability is mainly bounded by the deployed consensus protocol, where multiple nodes are usually required to validate every single transaction effectively precluding to mint large quantities of NFTs and transferring them fast [162]. Moreover, the request for either minting an NFT or transferring its ownership could fail due to limited resources in the network, low transaction fees for transactions, or miner node failures. On the other hand, transaction fees follow the law of supply (of computation resources) and demand (of minting requests for NFTs). As such, they can be largely volatile making the decision on how or where to mint an NFT a challenging task. In the presence of multiple public blockchain platforms with different performance characteristics, this can be especially challenging.

In our previous work [85], we studied the performance of 3 different public application platforms, including Fantom, Polygon, and Avalanche networks. We developed a prototype NFT marketplace and simulated a workload of minting and transferring 1000 unique NFTs to 1000 individual crypto wallets. We monitored the marketplace and gathered data for the duration of completing both minting and transferring NFTs, the number of errors per platform, the number of errors per minute, throughput per minute, and the transaction fee obtained from the official gas stations for each blockchain network. Next, we trained and compared different machine learning models to predict the gas price and throughput and identify the important parameters that could affect the cost and the throughput. Compared to our previous work, *IntelliChain* is a self-adaptive prediction mechanism for minting and trading NFTs on the public blockchain. *IntelliChain* predicts the blockchain transaction fee for different public blockchain platforms to reduce errors and retries of transactions. Furthermore, it can dynamically change the public blockchain for minting NFTs based on transaction fees and errors.

Our main contributions in this work are:

- **A prototype NFT marketplace**: The main focus for such a marketplace is to be scalable and efficient in managing to mint and transfer a large amount of NFTs in a short time, and compatible with a number of underlying technologies.
- **A comparative study of public blockchain platforms**: The objective of this study is to find which public blockchain is the most efficient, the most cost-efficient, and the most reliable one in the context of an NFT marketplace.
- **Two predictive models for transaction fee and error**: These models aim to improve the cost-efficiency and reliability of the NFT marketplace by guiding the

decisions around submitting the right fee for a transaction to the write public blockchain platform.

- **A self-adaptive mechanism to dynamically switch between public blockchain platforms**: This mechanism aims at improving the flexibility of the marketplace and allowing for even more improved efficiency and reliability.

The rest of the paper is organized as follows. In Section 10.4, we outline the related literature and current studies on NFT marketplaces. In Section 10.5, we present the architecture of our proposed NFT marketplace. Section 10.6 discusses a comparative study on public blockchain technologies. Section 10.7 presents the methodology and results for the predictive models for error and transaction fees. Section 10.8 presents a self-adaptive decision-support system for intelligently switching public blockchain. Finally, Section 10.9 concludes this work.

10.4 Related Work

10.4.1 Blockchain and NFTs:

OpenSea [8] and Rarible [9] are the most well-known NFT marketplaces that are based on the public blockchain technologies such as Ethereum and Polygon. Opensea itself recorded over 5 billion USD worth of sales between by January 2022 [10]. The two platforms provide a decentralized infrastructure for digital asset traders and creators that requires a crypto wallet [162]. This kind of marketplace has some limitations. First, it is required that digital asset creators be familiar with a crypto wallet and cryptocurrency, and they need to pay the transaction fees in cryptocurrency, which is a cumbersome process for many basic users. Second, the creators need to individually mint their digital assets one by one, requiring a significant amount of time. They would also need to reveal what is the exact NFT that users want to buy, but some sellers want to surprise their buyers (e.g, surprise gift boxes as part of a marketing campaign). Third, it is hard to randomize the NFTs' serial numbers to the buyers, as all the NFTs are already there, and the users can choose which serial number they want to acquire. Finally, there is limited support for different public blockchain networks.

Flow blockchain [11] is another notable project designed for some sports teams of national or international competitions such as UFC, NBA, and NFL. Flow is a closed blockchain marketplace, which is only designed for some specific digital artists such as NBA teams who

[8] https://opensea.io/
[9] https://rarible.com/
[10] https://hypebeast.com/2022/2/opensea-new-record-nft-sales-january-2022
[11] https://www.onflow.org/technical-paper

have a contract with Flow blockchain and not everyone is allowed to mint and sell their NFTs, as in OpenSea or Rarible. This imposes certain limitations. First, it is based on a private blockchain, which reduces the decentralization of the service and imposes a central trusted authority, which contradicts the purpose of blockchain. Second, this permissioned blockchain only has 37 validators, which may jeopardize the platform's availability and integrity of the data. Chiliz [165] is another permissioned NFT marketplace. In general, permissioned marketplaces imply the control of the data and of the verification process by one or more parties playing the role of a central authority, which is against the objectives of blockchain.

Binance Smart Chain [12] and Crypto.com [13] are proprietary NFT marketplaces, based on specific blockchain technologies, which imply that traders and creators trust these companies to always keep their blockchain platform available for the users. These two platforms both are running based on central authority and trust among the peers. This is against two of the main characteristics of blockchain which are decentralization and trustless [233].

10.5 Prototype NFT Marketplace

Based on our experience working on the development of NFT marketplaces in an industrial setting, a scalable and cost-efficient NFT marketplace based on the public blockchain should have the following properties/functionalities:

1. It should follow the "lazy minting" approach. In lazy minting, an NFT is minted only if and when it is sold. In practive[14], when a creator submits the NFT to the blockchain, a "voucher" is created, which is then redeemed by a buyer at which point the NFT is minted and only then the mint and transfer fees are paid by the creator and the buyer. In principle, this should reduce unnecessary gas consumption and transaction fees, in case of a lack of buyers, on the blockchain.

2. It should scale to process thousands of NFTs, including both minting and transferring. This scale comes from a specific use case, particularly popular in proprietary marketplaces, where the owner opens the marketplace for a specific period, during which thousands of sales take place simultaneously. The NFTs are already in the marketplace when it opens, but as per the lazy minting, they are minted as soon as the orders have been successfully completed and the minted NFTs should be transferred to the buyer's crypto wallet. While marketplaces, like OpenSea, are always open and a variable number

[12] `https://bnbchain.org`
[13] `https://cronos.org/`
[14] `https://nftschool.dev/tutorial/lazy-minting/`

of NFTs may be minted and transferred at any given time, with potential periods of low workload, the use case we present here works more in spurts of high workload, putting a particular strain in the system's capacity and performance.

3. It should use a public blockchain that prioritizes throughput and cost. In this context, throughput corresponds to successful requests that include both minting and transfer for the NFT. For the use case described before, throughput is important given that the transaction period is limited, thus the more transactions we can get through, the higher the chance of completing the goal. Given that throughput and cost in a public blockchain depend on supply and demand, and how busy the network is, both metrics are volatile, which means that the marketplace should be able to choose the optimal network in the presence of many.

4. Finally, both creators and traders should be able to use the system and trade NFTs regardless of knowledge of blockchain, solidity smart-contract programming, having a crypto wallet, and cryptocurrencies.

The proposed architecture for the prototype NFT marketplace is shown in Figure 10.1. A web application acts as the front-end, based on `Node.js`, where interested customers can submit an order to purchase an NFT. Marking the order as "pending" indicates that it has been received and is being processed. The event processor then selects the transaction and adds it to the FIFO queue. Finally, the mint request is sent to the blockchain smart contract using a pool of RPC (Remote Procedure Call) nodes (see Section 10.6.1).

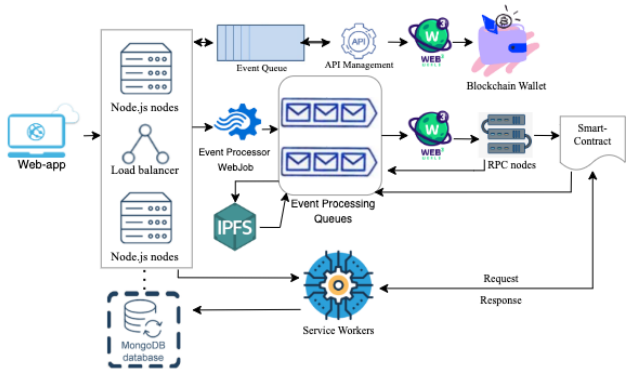


Figure 10.1 Schematic components of the framework for an NFT dApp

At this time, the miner has interacted with the smart contract and performs the minting of the requested NFT. If it is successful, a transaction hash is received and the NFT status changes to "Processing" in the database. A running worker constantly checks the status of

the transactions and in case of failures, it reverts them back to the queue. During testing, it was observed that a transaction hash may be received from the blockchain but the transaction would have never been minted. The system has a 60-minute timeout to verify these cases and in case of failure, we will have to remint the transactions that have not succeeded. At the same time, the worker is also responsible for finding out the unique token ID of the NFT in the blockchain and storing it in the database. A successfully minted NFT then has its ownership transferred to the buyer. The transfer process is similar to that of minting. After a confirmed transaction hash for the NFT has been received, the buyer will receive an update with confirmed details along with their crypto Blockchain wallet. The database captures events for both minting and transfers and marks the status as transferred.

The prototype eliminates the need for digital asset creators to deploy the smart contract or specify a crypto wallet and pay the transaction fees in cryptocurrency for every transaction. In contrast, after signing up, customers have the option of either providing the address of their cryptocurrency wallet or having the application generate one for them automatically. The deployment of smart contracts and the minting and transfer of NFTs is also taken care of by the application.

10.6 A Comparative Study of Public Blockchain

As it has been mentioned, due to the limitations of Ethereum, several other solutions have emerged to provide an alternative for the public blockchain to back an NFT marketplace. This poses an important strategic question for these marketplaces: what is the best alternative? To be able to answer this question, it is necessary to study the performance and the costs of each alternative under a workload similar to the use case we described before.

To this end, we conducted a study to compare three public blockchain platforms used in NFT minting, including Polygon Matic, Avalanche, and Fantom. The three platforms were selected, among multiple alternatives, based on the following criteria.

1. The platform should support solidity smart contracts, the language used for Ethereum smart contracts. This was done for consistency and conformity reasons to allow for a better comparison. The three platforms are all descended from Ethereum so they support solidity natively.

2. The platform should provide access to its gas station. Among other things, a gas station provides a front-end interface for the online monitoring of the blockchain platform. The gas station provides data such as transaction fees, the number of pending transactions

in the queue, and the average block (transaction) size. Other parameters can affect transaction fees and throughput, but these data are unavailable at public blockchain gas stations.

3. The public blockchain should provide a crypto faucet for obtaining test cryptocurrency. The cryptocurrency is necessary to run experiments with an actual public blockchain. However, given the scale of our experiments and the purpose of our study, we only need a specific amount of cryptocurrency for each platform. A faucet is an application that exchanges cryptocurrency for the completion of small tasks.

10.6.1 Study Infrastructure

Figure 10.2 shows the general architecture of the system used in the comparison study. Here, we outline the components of the system and their function in the study.

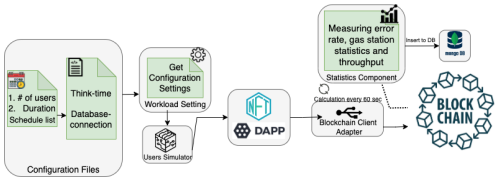


Figure 10.2 Schematic components of workload design

Workload Generator

To exercise the three blockchain networks and evaluate their performance and costs of minting and transferring NFTs, we designed a custom workload generator that simulates concurrent and real-time users buying the NFTs. The workload is that of the use case described before. One thousand NFTs are available for minting and transfer in the blockchain. Once the marketplace is open (the time the experiment begins), the users, simulated by the workload generator, start placing orders for acquiring NFTs. Batches of requests are sent simultaneously at frequent intervals. As soon as, an order is placed, the mint and transfer request for a particular NFT goes to the mint and transfer queues in order to be processed. The workload generator stops sending orders when all NFTs have been successfully minted and transferred.

Subject Application

The prototype decentralized application described in Section 10.5 is used as the subject application to evaluate the three blockchain platforms. The experiment implementation of

the prototype is based on the ERC1155 standard for NFT. An API to allow users to sign up for the marketplace was developed and connected directly to the workload generator. At the registration, a web3 framework is used to generate a unique crypto wallet for each user. Several other APIs were developed to implement the backend REST communication, store NFT metadata in IPFS, and to mint and tokenize the NFT. Two FIFO queues are used to handle the simultaneous orders. Once an order is submitted it enters the queue for minting NFTs. Once an NFT has been minted, it enters the queue to be transferred to the final buyer. The queues have been implemented using the Amazon SQS service[15]. If a mint or transfer request fails, due to the unavailability of blockchain or insufficient funds, the request reenters the respective queue as per the fallback strategy. According to the lazy minting strategy, only confirmed sold NFTs are minted and transferred. This confirmation happens at the Node.js frontend, at which point the mint event is triggered through the proper queue.

Client adapter

To allow for the actual submission to each of the studied blockchain platforms, we created a "client adapter" for each one of them. This component is necessary since each platform has its own API to accept requests. In practice, the adapter submits the smart contract to the blockchain (one for each different platform) and contains the admin crypto wallet, specified in the cryptocurrency of each blockchain platform, and is responsible for handling all transactions in this cryptocurrency. The adapter is an additional component that needs to be developed to use a public blockchain, but it keeps the prototype NFT marketplace generic and independent of the underlying public blockchain, allowing, as we will discuss later, to dynamically switch between the alternative platforms. In the prototype marketplace, the Node.js server pulls requests from the queues and then submits the requests to RPC nodes specified in the adapter.

Due to the limitation of public free RPC nodes [16] for the number of simultaneous transactions per minute, we implemented a dedicated node. This dedicated RPC node acts as an intermediary between the blockchain and the web application, i.e., the marketplace. While the RPC node is considered one of the miners/validators in the blockchain network, it does not do any real validation based on a consensus protocol, but it is rather used to forward the request seamlessly to the blockchain and replicates the request among other nodes of the network for them to validate and store the transaction [234]. In case of failure of the RPC node, a fallback strategy is in place, according to which the request reverts back to the queue

[15] `https://aws.amazon.com/sqs/`

[16] `https://alchemy.com/overviews/rpc-node`

to be reprocessed. Even if the node is available, the mint request may still fail because of an insufficient transaction fee. As it has been mentioned, the transactions are volatile and the fee that was submitted with the request may become insufficient between the time the request was submitted and the time it is processed. In such a case, the mint request reverts back to the queue until the submitted transaction fee is enough to be processed. This fallback, in either case, increases the reliability of the system as it decreases the number of lost or completely failed transactions.

The client adapter is also responsible for submitting the amount of gas (i.e., the transaction fee) that the user desires to pay for the transaction, through the admin crypto wallet. It constantly checks the suggested gas price from the official gas station of the blockchain. However, the blockchain will not always accept the recommended amount from the gas station, because conditions change constantly. Therefore the transaction will be declined with different errors that will be discussed later. In case of any type of error, we store the error type in the database and resend the transaction to the queue to retry. The admin wallet in the client adapter handles all transactions, including process fees for minting and transferring NFTs. The user crypto wallets created by the workload generator are only used to transfer the successfully minted NFTs to the buyers, but perform no other transaction in cryptocurrency.

Blockchain platforms

Table 10.1 Details of the studied blockchain platforms.

	Consensus	Validators
Avalanche	DAG	1567
Polygon	PoS	80
Fantom	DAG-aBFT	89

The three blockchain platforms compared in this study are Polygon, Fantom, and Avalance, and their details are presented in Table 10.1. As shown, the Avalanche network, which uses the DAG (Directed Acyclic Graphs) [17] technology for the consensus, is the most decentralized network, in terms of the number of validators with 1567 nodes, among the three platforms. The DAG technology has been improved by Avalanche to be more resilient and scalable, it does not require any specialized hardware for processing, and it can withstand "51% cyberattacks", in which 51% of the network's nodes are compromised. Additionally, Proof of Stake (PoS) has been included, which requires a financial contribution to function as a validator on the Avalanche platform. In PoS, the more money a node has, the greater its voting power in the

[17] `https://docs.avax.network/overview/getting-started/avalanche-consensus/`

consensus process. The compensation that the validators get will depend on how much money they have contributed or "staked".

The Polygon network is also based on The Proof of Stake [18] protocol, and it is now available for everyone to join as a miner. At present, Polygon has 80 validators, and to become one it requires specialized hardware with certain specifications for producing blocks and contributing to the consensus. A minimum contribution will be needed to qualify as a validator. In PoS, the system will function reliably if at least 2/3 of the validators are trustworthy.

The Fantom blockchain is based on the Lachesis consensus protocol [19]. The Lachesis consensus combines the DAG and Asynchronous Byzantine Fault Tolerance (aBFT) protocols. The efficiency, security, and latency concerns are prioritized to be improved. The aBFT is a modified form of Practical Byzantine Fault Tolerance (PBFT) that minimizes the number of exchanges needed between nodes to achieve an agreement. As opposed to PBFT, the aBFT algorithm allows for the simultaneous loss or delay of communications, which leads to a speedier network. Each event block in the DAG-aBFT consensus process is recorded by the nodes and comprises several transactions. Peers regularly discuss the previous two to three occurrences and affirm or deny them. Although the DAG-aBFT minimises the number of exchanges needed, its consistency still depends on peer exchange, and the more nodes we have, the more messages will be expected to be exchanged, which impacts the system's latency. There are currently 89 active validators on the Fantom network. In the Fantom blockchain, the network performs reliably as long as two thirds of the nodes are trustworthy.

Monitoring

The monitoring component is a vital part of this study because we need to get data from different parts of the system at the same time.

The following elements make up the monitoring component:

1. Blockchain gas station scraper: It gathers data directly from the gas station as it is updated. The data includes the number of pending transactions, the average block size, and the average utilization of the network as a percentage. Every data pull from the scraper is timestamped.

2. Blockchain gas station API: The API can return the anticipated gas limit and gas pricing for each platform. Each gas station of the three platforms provides such an API.

[18] `https://polygon.technology/solutions/polygon-pos`
[19] `https://fantom.foundation/lachesis-consensus-algorithm`

3. Blockchain transaction recorder: This is a script in the Node.js server to record all live transactions for minting and transferring on a certain smart contract and save the results in a MongoDB database. A function to determine throughput as the number of transactions per minute has been developed separately.

4. "Retry-and-error" counter: This is a script hosted in the RPC node which counts how many transactions had to be retried. The script also records the type of error in the database.

10.6.2 Results of comparative study

Table 10.2 Comparison results for Polygon, Fantom and Avalanche networks

	Avalanche	Polygon	Fantom
Total Time (min)	1179	953	1115
Avg TX fee (gwei)	86.41	51.95	337.17
Avg Mint Fee (USD)	$0.56	$0.02	$0.05
Transfer Fee (USD)	$0.15	$0.00	$0.01
Avg Throughput/min	1.69	2.09	1.77
Total Errors	230	138	83

The results of the comparison between Avalanche, Polygon, and Fantom networks are shown in Table 10.2. As mentioned the workload consisted of 2000 transactions, 1000 for minting NFTs and 1000 for transferring them. The Polygon network processed 2000 transactions in 953 minutes (16 hours), making it the quickest in terms of the overall time required to mint and transmit 1000 NFTs. For Avalanche, the equivalent completion time was 1179 minutes (>19 hours). Lastly, for the Fantom blockchain, the total time was 1115 minutes (>18 hours). The least expensive average gas value among the three is 51.95 gwei[20] for Polygon, followed by 86.41 gwei for Avalanche, and 337.17 gwei for Fantom. Avalanche is the most pricey among the three blockchain systems when we consider the conversion of minting costs into US dollars. This is due to the fact that the currency conversion rate also affects the real price that we pay for a transaction. According to Coinmarketcap [21], one Avalanche token is worth $16.03 USD, ranking #14; one Polygon Matic coin is worth $0.78 USD, ranking #12; and one Fantom token is worth $0.20, ranking #67.

[20] "Gwei is a denomination of the cryptocurrency ether (ETH), the digital coin used on the Ethereum network." `https://investopedia.com/terms/g/gwei-ethereum.asp`

[21] `https://coinmarketcap.com/lastaccessedon12November2022`

In contrast to Polygon and Avalanche, Fantom's transaction fee is six times higher than that of the Polygon network and the value of one Fantom crypto is half of Polygon Matic Crypto token. However, we continue to pay 2.5 more for minting in Fantom, which amounts to $0.02 in the Polygon network and $0.05 on average for Fantom. It can be surprising that an NFT transfer costs roughly $0.00001 on the Polygon network, next to nothing, whereas in the Fantom network costs $0.01, or 1000 times more. However, this is a conscious and strategic decision between the Polygon peers. The Polygon network's validators unilaterally decided that it would be almost free for individuals to transfer and trade existing NFTs on their network and that they would not charge validations to incentivize users and enhance the popularity of the Polygon network for NFT markets. Nevertheless, the transaction fee is not entirely free to avoid network spamming.

With respect to throughput, we measured the completion of minting or transferring unique NFTs every minute. The throughput for both Avalanche and Polygon networks was four transactions per minute, whereas Fantom was limited to three per minute. It is impossible to achieve much higher throughput rates than this because doing so would be interpreted as a DDoS attack on the network.

Finally, we recorded the type and number of errors per minute that we could observe while processing the transactions in the back-end. Eventually, the following errors occurred during the study.

- "Replacement transaction underpriced" or "Out of gas" : This error occurs more often compared to others. It occurs when the amount of gas that was originally submitted for paying the transaction is declined by the blockchain. This happens when the gas amount changes quickly in the gas station.
- "The gas limit required for your transaction is higher than the block gas limit.": This happens when suddenly the transaction fee drops significantly, and the transaction is rejected.

Based on our experimental results, there is no single public blockchain that is the best with the respect to the level of decentralization, throughput, error rate, and costs.

10.7 Prediction of Transaction fee and Error

As it was seen during the design of the prototype NFT marketplace and the comparative study for public blockchain, the main challenge is related to errors. Errors result in retries

or failed altogether transactions if the marketplace does not have the fallback queues, which in turn reduces the throughput, i.e., successfully completed transactions, and increases the total time for minting and transferring NFTs. As it has been discussed, one important reason for errors is the insufficient transaction fee declared by the user during the request. If this fee is lower than what the blockchain expects, so that the effort of the validators is properly rewarded, the transaction fails. One naïve approach to address this challenge is to provide a transaction fee with an overhead, in other words, a slightly inflated fee to increase the probability of a successfully completed transaction. In the long term, this may result in increased costs, potentially unnecessarily.

To overcome this challenge, we propose an intelligent predictive model, first, to predict the optimal transaction fee to complete a request and, second, to estimate the expected errors given this transaction fee to confirm the probability of success for a given blockchain platform. To this end, we designed a pipeline of two models, where the result of the fee prediction model is provided as input for the error prediction model. We experimented with a variety of machine learning (ML) models, such as eXtreme Gradient Boosting (XGB) [177], Random Forest (RF) [178], Decision Tree [179], Linear Support Vector Machine [235] (LSVM) for discrete outcomes, and linear regression [236] for continuous outcomes, and neural network (NN) models, including Long Short-Term Memory (LSTM) [237], Gated Recurrent Unit (GRU) [238], and Bi-directional Long Short-Term Memory (Bi-LSTM) [239], for both predictions. The ML models were trained using the Scikit-Learn library [176] in Python 3.6 [175], while the NN models were trained using the Keras high-level neural network library [240] with TensorFlow [241] again in Python 3.6. In order to train and test the models, we split the data into 75% training and 25% testing data and we used 10-fold cross-validation on the training data.

10.7.1 Predicting the blockchain transaction fees

The first model of IntelliChain is used to predict the required transaction fee for minting/trading an NFT on a public blockchain by reducing the probability of rejection by the blockchain for the reason of insufficient gas fees. The data we used as input for this model included the average number of pending transactions in the public blockchain, the average block size, and the utilization of the blockchain network. Utilization was only available in the gas stations of Avalanche and Polygon and as a consequence, it was not considered for the Fantom model.

The output of this model is the optimal transaction fee to reduce the errors, which is a continuous variable. As a result, we used the R^2 score, the Root Mean Square Error

(RMSE), and the Mean Absolute Error (MAE) to evaluate the results of the models [181, 242]. Hyperparameter tuning was performed during the training of the models to optimize their performance, but its impact was not significant in the final results.

NN models are better suited for large and complex datasets [243]. Although the initial intention was to use the data from our comparative study (Table 10.2), this was not sufficient. Given that the input data necessary for this model is readily in the gas stations, we scraped the gas stations in general, not necessarily data from our experiments, we obtained 80,000 data entries for each public blockchain. We used this data to train both ML and NN models for the transaction fee.

We configured NN models for 500 epochs and 50 batch sizes. Additionally, we tried various optimizers, such as Stochastic Gradient Descent methods (SGDs) [244], Adaptive Gradient Algorithm (AdaGrad) [245] and Adaptive Moment Estimation (Adam) [246]. Based on our results, Adam optimizer was the best among these candidates. Lastly, we added early stopping to dropout layers so that they would not converge too quickly. This helped us avoid over-fitting and make the models work better.

Results for prediction of transaction fees

Table 10.3 Summary of the results for RMSE, MAE and R2 score for transactions fee prediction among different Machine Learning (RF, XGB, DT, and LR) and Neural Network (LSTM, Bi-LSTM, and GRU) models

		ML				NN		
		RF	XGB	DT	LR	LSTM	Bi-LSTM	GRU
Avalanche	RMSE	11.89	14.97	16.35	20.61	7.25	4.04	9.27
	MAE	7.77	8.44	8.78	13.36	4.79	2.05	5.79
	R2 Score	0.86	0.77	0.73	0.57	0.94	0.98	0.9
Polygon	RMSE	8.21	11.8	8.64	13.91	8.12	4.92	10.67
	MAE	5.44	7.07	5.12	8.18	5.07	1.9	6.69
	R2 Score	0.93	0.85	0.92	0.79	0.97	0.99	0.96
Fantom	RMSE	243.13	424	272.95	378.98	773.76	736.97	742.97
	MAE	50.55	68.5	53.38	75.33	266.99	219.32	242.94
	R2 Score	0.57	-0.32	0.45	-0.06	0.27	-0.57	-0.55

Table 10.3 summarizes the results for the seven models to predict transaction fees. Random Forest is the best model among the four ML algorithms across all three evaluation metrics and between all three public blockchain platforms. More specifically, RF had a R^2 score of

0.86, 0.93, and 0.57 for Avalanche, Polygon, and Fantom respectively. On the other end, NN models performed even better than ML models, with Bi-LSTM being generally the better of the three with a R^2 score of 0.98, 0.99, and -0.57 for Avalanche, Polygon, and Fantom respectively. In general, it looks like all models had a harder time accurately predicting the transaction fee for Fantom. This is even more evident by the significant difference in the RMSE and MAE errors between Fantom and the other two platforms. There can exist many reasons for this challenge, including Fantom's exceptionally low cost, which may cause scaling problems for the prediction, or the lack of utilization as input. To determine and prove the actual reasons for this deviation is out of the scope of this work, but we plan to further investigate this phenomenon in our future work.

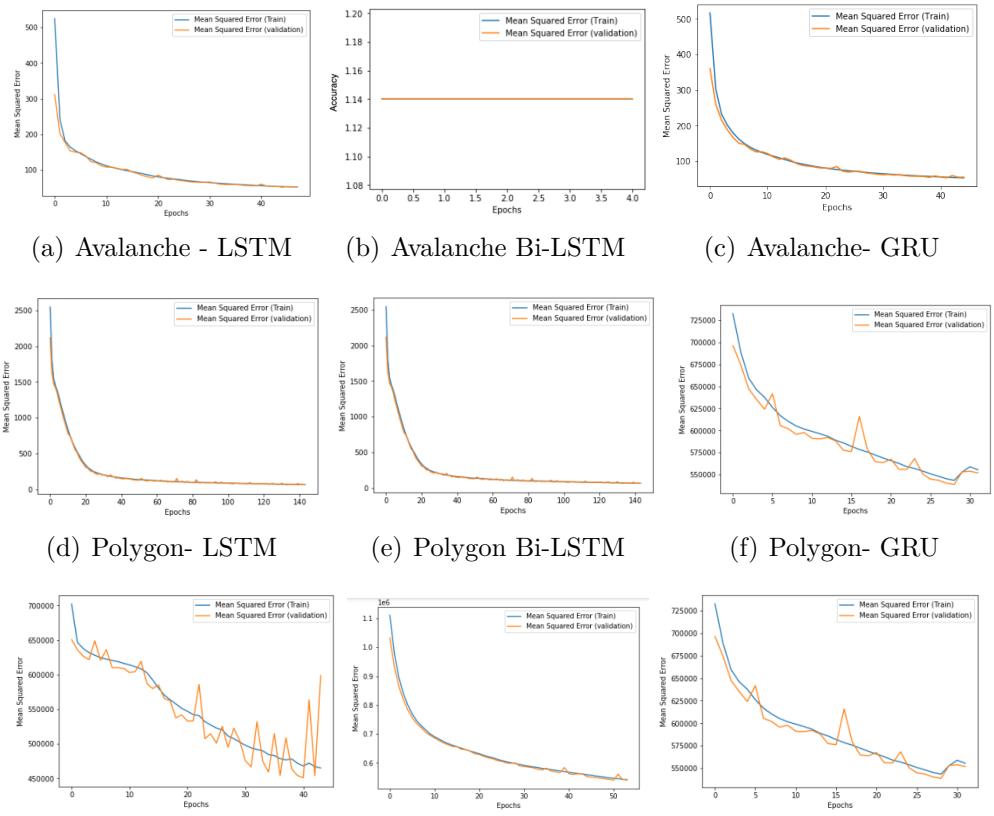
Figure 10.3 graphically presents the progression of the MSE metric for the different epochs of training for the NN models. To begin with the Avalanche platform with three different models, we can observe that although Bi-LSTM experienced the highest MSE at the beginning, over time, the MSE for training and validation data becomes similar. In Polygon, both LSTM and Bi-LSTM models almost performed similarly, with the same number of epochs; however, the GRU model experienced some fluctuations with different epochs. For Fantom, it is interesting to see that, although we received a negative R^2 score for Bi-LSTM, the MSE for training and validation data performed is quite similar. Finally, both LSTM and GRU show significant fluctuation in validation data.

Eventually, based on the final results of our experiments, we decided to use Bi-LSTM to predict transaction fees in Avalanche and Polygon and RF for Fantom.

10.7.2 Predicting the blockchain errors per minute

The next step, after predicting the optimal transaction fee is to confirm that this fee would indeed result in reduced errors. To do this, we train another model to predict the number of errors given the transaction fee as input. In addition, the model receives as input the average number of pending transactions, the average block size, the utilization (only for Avalanche and Polygon), and the throughput of the blockchain platform as monitored by our Node.js server (see Section 10.6.1). In this model, the output is the number of errors returned as a discrete variable with possible values ranging between 1 and 4. The reason for this range is that public blockchain platforms usually do not allow us to send more than 3 or 4 requests within a minute as this will be perceived as a DDoS attack. Consequently, it is not possible to receive more than 4 errors. Moreover, it is only possible to receive an integer number of errors and not fractional. Therefore, the output is modeled as a discrete variable between 1 and 4.

Given that the errors are monitored only by our Node.js server and are not available in the gas



(a) Avalanche - LSTM (b) Avalanche Bi-LSTM (c) Avalanche- GRU

(d) Polygon- LSTM (e) Polygon Bi-LSTM (f) Polygon- GRU

(g) Fantom- LSTM (h) Fantom- Bi-LSTM (i) Fantom- GRU

Figure 10.3 The progression of MSE between training epochs for the prediction of transaction fee in GRU, LSTM and Bi-LSTM models

station, we had to rely on data from our comparative study. To overcome this, we used data augmentation [247] to generate additional synthetic data up to 80,000 entries. Oh et al. [247] proposed a general-purpose time-series data augmentation method based on interpolation, which was shown to generally improve accuracy, and this was why we selected this method. On this synthetically augmented data, we trained the same ML and NN models, as in the transaction fee prediction model, except for linear regression. Since this type of model is more appropriate for continuous outcomes, we used the linear support vector machine (LSVM) model as a linear alternative for discrete outcomes. All models were tested separately on the 25% of the augmented dataset and on the original dataset from the comparative study (Table 10.2). The latter was done to confirm the accuracy of the model on the actual data, even if it was a small dataset. With respect to the evaluation metrics, due to the fact that the error is a discrete output as opposed to the fee being a continuous output, we used different evaluation metrics. More specifically, we used precision and recall, the F1-score, as the harmonized average of precision and recall, and accuracy, as the total percentage of correct predictions [248].

Results for prediction of transaction errors

Table 10.4 Results on training and testing on augmented data for error prediction models

		ML				NN		
		RF	XGB	DT	LSVM	LSTM	Bi-LSTM	GRU
Avalanche	Accuracy	0.64	0.62	0.55	0.66	0.88	0.12	0.83
	Precision	0.33	0.31	0.3	0.16	0.92	0.03	0.94
	Recall	0.27	0.28	0.3	0.25	0.9	0.42	0.85
	F1-Score	0.29	0.29	0.3	0.19	0.91	0.06	0.89
Polygon	Accuracy	0.67	0.67	0.61	0.1	0.98	0.99	0.98
	Precision	0.28	0.35	0.35	0.06	0.99	1	1
	Recall	0.26	0.28	0.3	0.23	1	1	0.99
	F1-Score	0.26	0.31	0.32	0.09	0.99	1	0.99
Fantom	Accuracy	0.63	0.62	0.52	0.61	0.7	0.8	0.76
	Precision	0.22	0.29	0.28	0.2	0.97	0.93	0.94
	Recall	0.25	0.27	0.29	0.24	0.7	0.82	0.77
	F1-Score	0.23	0.27	0.28	0.21	0.82	0.87	0.85

Table 10.4 shows the results for the seven models when tested on 25% of the augmented data. As it can be seen, NN models are again generally better than ML models. Between the ML

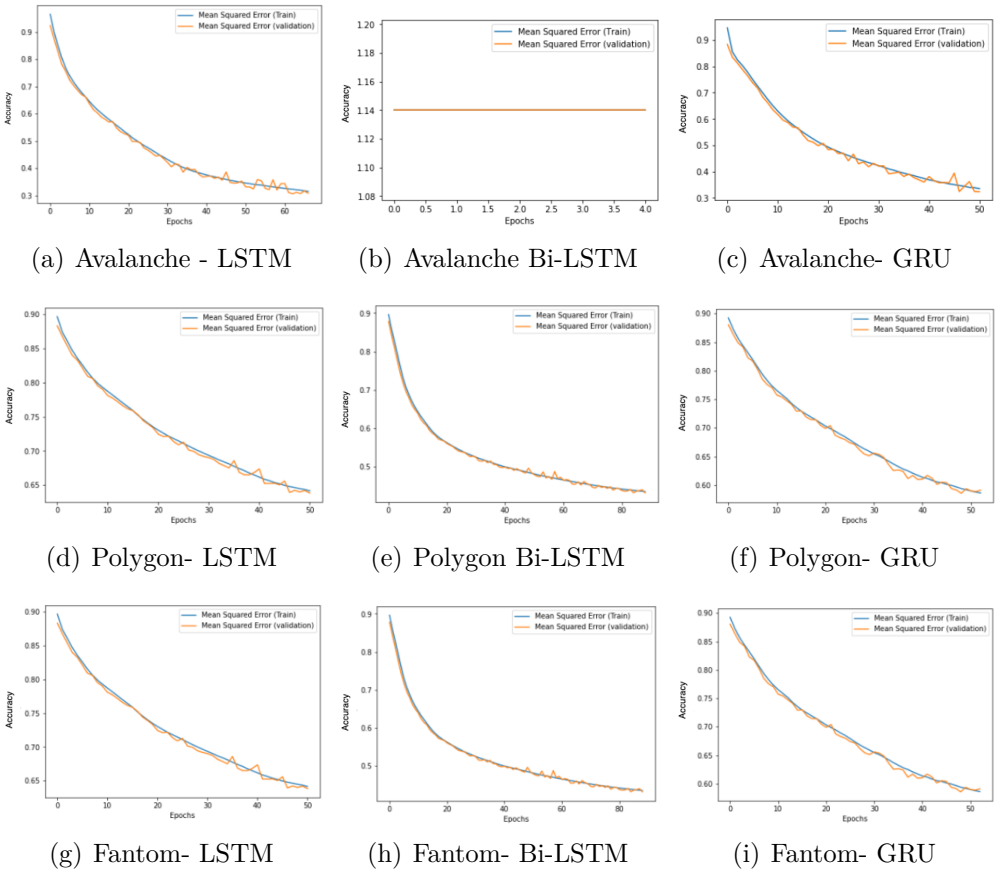
Table 10.5 Results for testing the error prediction model on the original data of Table 10.2

		ML				NN		
		RF	XGB	DT	LSVM	LSTM	Bi-LSTM	GRU
Avalanche	Accurancy	0.59	0.58	0.52	0.59	0.66	0.31	0.65
	Precision	0.32	0.25	0.29	0.17	0.9	0.4	0.92
	Recall	0.3	0.25	0.3	0.23	0.73	0.48	0.71
	F1-Score	0.3	0.25	0.29	0.19	0.81	0.48	0.8
Polygon	Accurancy	0.58	0.58	0.52	0.58	0.73	0.73	0.73
	Precision	0.24	0.24	0.29	0.24	1	1	1
	Recall	0.25	0.25	0.3	0.25	0.74	0.74	0.74
	F1-Score	0.25	0.25	0.29	0.25	0.85	0.85	0.85
Fantom	Accurancy	0.53	0.62	0.53	0.66	0.67	0.67	0.67
	Precision	0.27	0.28	0.28	0.16	1	1	1
	Recall	0.28	0.27	0.28	0.25	0.67	0.67	0.67
	F1-Score	0.28	0.27	0.28	0.19	0.8	0.8	0.8

models, RF is once again the better model with respect to accuracy and F1-score. However, in this case, the differences between the four models are not as prominent as in the case of the transaction fee. Between the NN models, LSTM is the better model across all metrics for Avalanche and Polygon, and Bi-LSTM is better for Fantom. However, Bi-LSTM shows bad performance for the Avalanche model, for a reason, we will explain below. On the other hand, the Fantom models are generally good compared to the transaction fee models, which may provide an additional indication that the scale of the fee is an issue. Table 10.5 shows the evaluation results on the original data from the comparison study. It can be seen that, even though the performance degrades across all models, the NN models remain accurate to a satisfactory level with accuracy above 65% and an F1-score of above 80%.

Figure 10.4 shows the progression of the accuracy validation loss across the training epochs in the three blockchain platforms for the LSTM, Bi-LSTM, and GRU models. In the Avalanche network, the LSTM model has the highest similarity between training and validation data with respect to accuracy with the GRU model following. However, for the Bi-LSTM model, there is a single solid line. This is because the validation actually stopped in the first three epochs as the validation loss was not converging, also known as early stopping. , which means that the validation loss is not converging. This is also known as early stopping. This was the reason why Bi-LSTM significantly underperformed for the Avalanche platform, but the underlying reason is currently unknown. On the other hand, for both Polygon and Fantom blockchain platforms, Bi-LSTM has the best similarity for both training and validation datasets with

respect to accuracy.



(a) Avalanche - LSTM (b) Avalanche Bi-LSTM (c) Avalanche- GRU

(d) Polygon- LSTM (e) Polygon Bi-LSTM (f) Polygon- GRU

(g) Fantom- LSTM (h) Fantom- Bi-LSTM (i) Fantom- GRU

Figure 10.4 The progression of accuracy between training epochs for the prediction of error in GRU, LSTM and Bi-LSTM models

Based on the combined evaluation results, we eventually selected the LSTM model for Avalanche and the Bi-LSTM model for Polygon and Fantom.

10.8 Self-adaptive NFT Marketplace with Dynamic Public Blockchain Selection

The results of the previous experiments showed that we have an accurate enough method to predict both the optimal transaction fee and confirm the number of errors it may produce in each of the studied blockchain platforms. To further validate these models we tested them in practice for each of the three respective blockchain platforms we study in this work. More specifically, we repeated the experiments described in Section 10.6, but this time we equipped the Node.js server with the transaction fee and error prediction models to determine the optimal transaction fee to submit for each request to the public blockchain. Then, we ran the same experiments for the three platforms and we measured the total execution time, the

Table 10.6 Comparison results for Polygon, Fantom and Avalanche networks before and after using predicted transaction fee and, using the self-adaptive framework

	Before prediction			After prediction			After prediction
	Avalanche	Polygon	Fantom	Avalanche	Polygon	Fantom	Self-adaptive
Total Time (min)	1179	953	1115	876	890	1011	843
Avg TX fee (gwei)	86.41	51.95	337.17	52.31	24.95	3.5	21.76
Avg Mint Fee (USD)	$0.56	$0.02	$0.05	$0.11	$0.02	$0.001	$0.001
Transfer Fee (USD)	$0.15	$0.001	$0.01	$0.02	$0.001	$0.00	$0.0007
Avg Throughput/min	1.69	2.09	1.77	2.81	2.75	2.11	3.04
Total Errors	230	138	83	81	41	75	18

average transaction fee, the average mint and transfer fees, the average throughput, and the total number of errors encountered.

The first part of Table 10.6 shows the results of the two experiments, before and after the use of the prediction models, for the three platforms. The data for the before the part is the same as in Table 10.2. As it can be seen, the most notable difference is in terms of errors, where our method managed to reduce the total number of errors by 149, 97, and 8 for Avalanche, Polygon, and Fantom respectively. As this was the main objective of our models, we can consider that the proposed method is successful in reducing the number of errors. As a consequence of reducing the errors, we can also observe that the platforms spend much less time in retries and more productive time in completing transactions. More specifically, the use of prediction reduced the total time by 303 minutes (almost 5 hours) for Avalanche, 63 minutes (or an hour) for Polygon, and 104 minutes (one hour) for Fantom. Even for the previously fastest platform, Polygon, our method found one additional hour to further improve its performance. Eventually, the predictive models transformed the least efficient platform, Avalanche, into the most efficient one. Consequently, as the number of total transactions was constant, but the total time was reduced, the average throughput was increased for all three platforms. Finally, we can see that the prediction models managed to also reduce the transaction fees, especially for Avalanche, which was the most expensive of the three platforms. As it was argued before, the ability to accurately predict the optimal fee helps us to avoid any intentional overhead in the hope of getting the transaction accepted.

It is important to note that with respect to transaction fees and time, the current state of the blockchain (traffic and availability of validators/resources) plays an important role in the performance of the blockchain. As it was impossible to run both experiments at exactly the same time without affecting one another, it is possible that the experiment after the application of the predictive models was conducted under more favorable conditions. This can be considered a threat to the validity of our study, it could potentially be mitigated by

the repetition of the experiment multiple times and by considering the average of all results that would reduce the impact of such sources of variability. However, this repetition can be expensive and time-consuming and could not be completed in the context of this study. It is in our future plans to continue validating the models in industrial contexts with our partner to further solidify the results.

It is clear that the three platforms, even after the application of the predictive model, differ with respect to latency, throughput, cost, and error rate and there is no clear best platform for the case study of an NFT marketplace. Furthermore, the three platforms have other differences that we do not directly address in this study, namely security and integrity. As it was shown in Table 10.1, Avalanche, for example, uses a large number of validators, which may slow it down, but it increases its integrity and makes it more difficult for malicious actors to compromise the data or the platform itself. Even after improving the performance of the three platforms with the predictive models, it is clear that sticking with a single blockchain platform for the entire duration of minting and transferring the NFTs may be suboptimal given the changing conditions of the platforms.

For these reasons, we further propose the development of a self-adaptive mechanism for dynamically switching between public blockchain platforms at runtime. For this self-adaptive system, IntelliChain follows the MAPE architecture [107] and has the following four components: 1) the *monitoring* component, which is responsible for continuously retrieving the data from the public blockchain gas station, 2) the *analysis* component, which pools data from monitoring and every five minutes uses the predictive models to estimate the fee that will likely cause no errors for every alternative platform, 3) the *planning* component compares the analysis results and selects the best public blockchain, and 4) the *execution* component, which is responsible to change the Node.js application and transfer the workload the nominated public blockchain through the respective RPC node. If two platforms are found to return the same number of errors, then the planner selects the one with the lowest predicted transaction fee. As a result, the MAPE system in IntelliChain chooses the optimal public blockchain in terms of both reliability and cost efficiency.

Table 10.6 illustrates the results of our self-adaptive predictive public blockchain which decided to change the public blockchain based on the data we scrap from three different public blockchain technologies. As it is shown 10.6, the total time for minting and transferring 1000 NFTs was reduced to 843 and the throughput increased to 3.04 transactions per minute. This was an improvement of about half an hour and 0.2 transactions per minute from the best case after the prediction and about two hours and 1 transaction per minute from the best case before the prediction. This improvement was achieved mainly by the significant reduction of

the errors, a total of 18 for the self-adaptive system, compared to 41 after prediction and 83 before prediction. Finally, the transaction fee was also reduced significantly. The Fantom cost remained lower, but as we have discussed this is a choice by design for this platform. In total, IntelliChain minted 716 NFTs on Fantom, 248 NFTs on Polygon, and 36 NFTs on Avalanche. Fantom was the preferred platform in most cases as the cheapest alternative with Polygon as the second choice as the fastest one. Although Avalanche was chosen only in a few instances, it could be the case that it is a reliable and fairly cheap and fast alternative for smaller workloads. This distribution of the transaction generally reinforces the conclusion that no single platform is the best under all conditions for all transactions.

10.9 Conclusion and Future Works

As the number of public blockchain technologies increases, it is vital for practitioners and researchers to find the most appropriate network for their decentralized applications. There are different criteria for selecting a public blockchain such as public acceptability, robustness, transaction fees, and throughput. In this paper, we studied three popular public blockchain technologies and identified the performance and costs of these three platforms in a case study of an NFT decentralized application. Moreover, in this paper, we propose *IntelliChain*. On the one hand, it accurately predicts the required transaction cost for various open blockchain systems, hence minimizing errors, transaction retries, and throughput.

On the other hand, depending on the cost of recording transactions on the blockchain and the number of errors based on pending transactions and block size in gas stations, it enables the NFT marketplace to dynamically transition between public blockchain platforms. For future work, we would like to increase the number of public blockchain technologies with various smart-contract programming languages and also consider more metrics for our predictions such as the blockchain gas limit and usage by the transaction to be recorded, exploring pool size of pending transactions, validation time of a transaction in the blockchain and, response time from the blockchain. Blockchain technologies are still in their infancy and the results of the experiments on public blockchain platforms might be altered in the future.

CHAPTER 11 GENERAL DISCUSSION

We emphasize the far-reaching impact of our work in our area and industry, and we detail its unique contributions, and findings in this chapter.

11.1 Reviewing milestones

In this section, we further discuss the objectives highlighted in Chapter 3.

An extendable benchmarking platform for blockchain technologies: Blockchain's reassuring promise to maintain the integrity and immutability of data is increasing its popularity among practitioners. Ethereum, the conventional and most popular public blockchain platform, is built on Proof of Work. The proof of Work algorithm seeks to ensure data consistency and integrity by solving a mathematical challenge to add a new record to the ledger. When a new record is added to the ledger, the proof of Work method uses a mathematical puzzling to ensure the record's validity. As a result, there will be considerable delays and time overheads, and the cost of recording a transaction on the Ethereum ledger will rise. There have been several proposals for alternative blockchain technology and consensus protocol to improve scalability and performance. Our intention in this dissertation is to find the most appropriate blockchain technology for real-time applications depending on the workload. To this end, we proposed *BlockCompass* to emulate different blockchain technologies and simulate continuous and fluctuating workloads.

Existing solutions for benchmarking blockchain technology concentrate on analyzing throughput by delivering a batch of transactions. BlockCompass, on the other hand, is intended to examine these technologies' throughput, error rate, and resource consumption by generating a transactional workload that allows academics and practitioners to study over a prolonged period. This method can predict when the blockchain platform will become overloaded and cannot be restarted. For example, we used BlockCompass to make a comparion between the Sawtooth platform with PBFT consensus and Hyperledger Fabric with Raft consensus. Although both Sawtooth and Fabric employ the same consensus technique, Sawtooth has up to 50% packet loss when the number of users increase. For Fabric, we always had 0% for the same number of users. The same thing occurred with response time. When we added additional users to the system, the Sawtooth response time became five times slower than the Fabric blockchain.

To gauge BlockCompass's quality and usability, we conducted survey research and compared

it to Blockbench, an already available blockchain benchmarking tool. Overall, participants were more satisfied with BlockCompass than Blockbench regarding their experiences using benchmark tools, the clarity of documentation, the presentation of final data, and the ease with which trials could be done. BlockCompass often beats Blockbench regarding the time required to complete the problems and the setup, reporting, and documentation. BlockCompass is an extendable tool which means that it is designed to easily add new blockchain platforms and workloads so that practitioners can choose the most appropriate blockchain technology for their decentralized application.

Performance study of blockchain technologies: In this part of our study, we contrasted NoSQL distributed databases with popular private blockchain systems. Regarding response time and resource consumption, we discovered that Hyperledger Sawtooth with PoET consensus protocol beats all our subject tests. The key reason is that the PoET validates the blocks without any mathematical computations. PoET chooses a leader to append the block by random waiting time through Intel SGX. Additionally, PoET may also handle transactions in parallel. A system with PoET architecture and a limited number of validators (fewer than 16) may, nonetheless, be compromised based on the theoretical assertion.

Then, in the following research article, we tried to identify the difference between public and private blockchain technologies on real-time and continuous workloads. This study showed us that the public blockchain could more efficiently handle large transactions with low arrival rates. On the other hand, private blockchain technologies like Ethereum PoA and Hyperledger Fabric are better at handling large numbers of small transactions and cost less.

In the third article, we evaluated the integration of Non-Fungible Tokens on the public blockchain. NFTs have been receiving much attention recently, especially on the Ethereum blockchain. Although the Ethereum blockchain can guarantee immutability and integrity, using the Ethereum blockchain is expensive and may impose a significant computational overhead. Several public blockchain platforms with different consensus protocols have recently proposed to cut down on the high transaction fees and delays in Ethereum with POW. Therefore, it is crucial to choose the best public blockchain platform. In this work, we compared the cost and performance of Fantom, Avalanche, and Polygon, three public blockchain platforms, in a use case for minting and trading NFTs. We observed that at the time of our study, Polygon blockchain was the most efficient platform compared to Avalanche and Fantom.

Self-adaptive systems in blockchain: We used two separate case studies in this part of the dissertation. We believe that implementing self-adaptive mechanisms would enhance blockchain's performance, integrity and efficiency. In first research article, we leveraged self-adaptive to dynamically change the consensus protocol to temporarily to mitigate the

impact of the DDoS attack until a more drastic solution is prepared. We suggest using more complex self-adaptive systems in future research to capture a range of concerns and realistic characteristics including manual algorithm configurations. What would happen, for instance, if we had a choice between utilising greater CPU power or minimising packet loss in certain circumstances.

In the second research article, we proposed our self-adaptive architecture to change the consensus mechanism dynamically when the decentralized application is subjected to a continuous and fluctuating workload.

This framework intends to improve the quality of service in terms of CPU utilization which corresponds to the resource costs and also response time. We have shown that our self-adaptive mechanism has better CPU utilization and response time comparably to PBFT and RAFT. However, PoET still remains the best in terms of CPU consumption and response time. Theoretically, our adaptive consensus protocol addresses PoET's security limitations for small networks (those with fewer than 16 validators) while maintaining PoET's performance requirements for larger networks.

Scalable and efficient decentralized framework: In this part of the dissertation, we presented *IntelliChain*, a scalable and intelligent framework to improve performance and scalability in decentralized application with the case study of public blockchain. We studied three popular public blockchain technologies and identified the the performance and costs of these three platforms in a case study of an NFT decentralized application. We simulated an NFT marketplace with 1000 unique NFTs to mint and transfer to 1000 unique users. We identified the throughput per minute, total time for completing the minting and transferring process, transaction fees and number of errors per minute.

The main goal o IntelliChain in this paper is on the one hand, it accurately predicts the required transaction cost for various open blockchain systems, hence minimizing errors, transaction retries, and throughput.

On the other hand, depending on the cost of recording transactions on the blockchain and the number of errors based on pending transactions and block size in gas stations, it enables the NFT marketplace to dynamically transition between public blockchain platforms. Using IntelliChain can improve the throughput and decrease the error and retry in the system.

For future work, we would like to increase the number of public blockchain technologies with various smart-contract programming languages and also consider more metrics for our predictions such as the blockchain gas limit and usage by the transaction to be recorded, exploring pool size of pending transactions, validation time of a transaction in the blockchain

and, response time from the blockchain. The blockchain technologies are still in their infancy and the results of the experiments on public blockchain platforms might be altered in the future. We also recommend to increase the number of transactions on public blockchain for building a better and more efficient model for predicting errors. This study aims to shed light on leveraging self-adaptive systems in using blockchain technologies.

CHAPTER 12 CONCLUSION

12.1 Summary of Works

Although blockchain technologies are still in their infancy, they are gaining wide popularity in a wide range of applications and industries— trust serves as the primary driver of adoption. Enhanced data integrity, smart contracts, immutability, traceability, and disintermediation are the five areas where these technologies provide value to blockchain technology integration. The blockchain principle is built around a consensus protocol that secures the network against manipulations. This means that when we store data in the network, we can ensure that our data is almost immutable and tractable.

Despite the ability of blockchain to guarantee the integrity of data through consensus, it is also true that this consensus protocol can be a performance and resource bottleneck, which is contradicted for real-time applications. As discussed in related literature, multiple benchmarking tools have been proposed, but those are not focused on identifying blockchain performance at run-time with continuous and fluctuating users. Monitoring and analyzing blockchain platforms can help us identify which blockchain technology is helpful for the specific workload and decentralized application. The main goal of this dissertation is to guide practitioners on how to design a cost-efficient decentralized application on the blockchain.

In the first part of this dissertation, we presented *BlockCompass*. BlockCompass is an extendable benchmarking tool that can automatically simulate workloads and emulate blockchain technologies. We designed a live monitoring dashboard that can show us in real-time when and with how many users the system has been saturated or with how many users the throughput or integrity of the system has deteriorated. BlockCompass currently supports the following metrics Emit rate, throughput, error rate, response time, resource utilization such as CPU, memory, and network, and scalability of the blockchain technology.

BlockCompass currently supports Ethereum PoA, Ethereum PoW, Hyperledger Fabric, and Hyperledger Sawtooth. However, it is well documented how the tool can be easily extended for newer blockchain technologies. At the time of writing this dissertation, we are collaborating to add Polygon and Algorand blockchain networks to the BlockCompass platform. Adding more blockchain technologies and consensus protocols in BlockCompass would help decision-makers and practitioners emulate their desired blockchain platforms and evaluate the target platform with different workloads.

In the second part of this dissertation, we studied blockchain technologies in three dimensions.

The intention is to guide practitioners on which blockchain technology suits a specific workload. This part consists of three research projects. We began comparing the performance of various private blockchain technologies against a conventional distributed database. The result helped us identify the throughput and resource utilization in some of the most popular private technologies and guide researchers and practitioners on what platform performs better in a long period of stress testing with many users. Next, we explored a different dimension: the difference in performance between public and private blockchain technologies. This research helped us identify the difference between these technologies in costs, throughput, and performance. We showed which blockchain is more suitable for which kind of workload.

In the third research project of this part, we presented a study of applying ML techniques to learn and predict different aspects related to the transactions for NFT platforms, which is to provide readers with complete results and assist in decision-making to have perspective on the best public blockchain to mint and trade for NFT.

The third part of this architecture is to introduce the benefits of self-adaptive systems to improve the system's quality and performance using MAPE architecture. In the first section of this study, we first studied the impact of DDoS attacks on different consensus protocols; then, we proposed self-adaptive architecture to temporarily reduce the impact of DDoS attacks on the system's integrity. In the second part, we proposed a self-adaptive framework to dynamically change the consensus protocol based on the number of nodes and CPU utilization. Our self-adaptive approach can address the security limitations in the PoET framework when running with fewer nodes and performs better comparably to PBFT and Raft consensus algorithms.

In the fourth part of this study, we proposed *IntelliChain*, a scalable framework to improve the throughput in decentralized applications, specifically NFTs. IntelliChain has two goals, first, accurately predict the amount of transaction fee to mint an NFT on a public blockchain to avoid errors and retries due to inappropriate amounts of proposed transaction fees on a public blockchain. Furthermore, it dynamically changes the public blockchain by predicting which public blockchain will have zero errors. In case all public blockchains are predicted to have zero errors, IntelliChain selects the most cost-efficient public blockchain. We have shown that using self-adaptive systems for having multiple choice for minting NFTs can improve throughput and reduce the number of errors and retries. In our comparative studies, we have observed that there is no single optimal public blockchain, and each has some straights and weaknesses in terms of costs, throughput, and decentralization. We recommend employing self-adaptive networks to improve the performance and efficiency of decentralized applications.

12.2 Future Research

BlockCompass currently supports a few private blockchain technologies, including Ethereum PoW, Ethereum PoA, Hyperledger Fabric Raft, and Hyperledger Sawtooth (Raft, PoET, and PBFT). We designed BlockCompass to be easily extendable, and new drivers can be added to the repository. It would be essential to cover the novel blockchain technologies as much as possible to understand the differences better. In terms of workload, we currently support closed workloads, which means that we start sending fluctuating and continuous requests from multiple sources and wait for a response from the server on whether the requests were successful. For future work, we also recommend another type of workload, such as open or batch workload, so that the practitioners can evaluate the target platform. In BlockCompass, we currently measure throughput, response time, resource utilization, and emit rate. We recommend adding more metrics such as the security of the target blockchain based on the malicious attacks, evaluation of energy and electricity consumption of each consensus algorithm and blockchain platform, and measuring the confirmation time by a consensus algorithm. We also recommend adding different case studies to BlockCompass workloads. We used key-value and NFT minting/trading in our previous experiments. The more options we will add, the more interesting it will be for practitioners and academia to evaluate the efficiency and performance of the target blockchain networks.

In this thesis, we also presented *IntelliChain*, a self-adaptive framework to predict the blockchain transaction fee and dynamically change the public blockchain based on the current situation and metrics in public blockchain gas stations. IntelliChain currently supports three major public blockchain technologies, and we should add more blockchain technologies and the ability to choose the criteria for specific decentralized applications. For example, security and decentralization are essential factors. In this case, IntelliChain should only choose those with the highest level of decentralization. If the transaction fee and throughput are essential factors, we should only select the ones that meet these criteria. IntelliChain uses the data from these three public blockchain gas stations to predict the errors and choose a public blockchain platform. For future studies, it would be better to include other criteria such as transaction explorers, the number of current validators, the transaction gas limit, confirmation time on a blockchain, number of rejections in a certain period. This will help to improve the prediction and efficiency of the IntelliChain framework. IntelliChain is currently designed to integrate NFT decentralized applications, and for future works, we should consider a more generalized situation for all decentralized applications. The main difference is that the payload for NFTs is always constant, and for the other decentralized applications, the payload should change, and we should also take the payload size into account.

REFERENCES

[1] A. A. Monrat, O. Schelén, and K. Andersson, "A survey of blockchain from the perspectives of applications, challenges, and opportunities," *IEEE Access*, vol. 7, pp. 117134–117151, 2019.

[2] M. Li, S. Shao, Q. Ye, G. Xu, and G. Q. Huang, "Blockchain-enabled logistics finance execution platform for capital-constrained e-commerce retail," *Robotics and Computer-Integrated Manufacturing*, vol. 65, p. 101962, 2020.

[3] J. Chen, T. Cai, W. He, L. Chen, G. Zhao, W. Zou, and L. Guo, "A blockchain-driven supply chain finance application for auto retail industry," *Entropy*, vol. 22, no. 1, p. 95, 2020.

[4] R. Wang, K. Ye, T. Meng, and C.-Z. Xu, "Performance evaluation on blockchain systems: A case study on ethereum, fabric, sawtooth and fisco-bcos," in *International Conference on Services Computing*, pp. 120–134, Springer, 2020.

[5] T. T. A. Dinh, J. Wang, G. Chen, R. Liu, B. C. Ooi, and K.-L. Tan, "Blockbench: A framework for analyzing private blockchains," in *Proceedings of the 2017 ACM International Conference on Management of Data*, pp. 1085–1100, 2017.

[6] T. T. A. Dinh, R. Liu, M. Zhang, G. Chen, B. C. Ooi, and J. Wang, "Untangling blockchain: A data processing view of blockchain systems," *IEEE transactions on knowledge and data engineering*, vol. 30, no. 7, pp. 1366–1385, 2018.

[7] S. Bano, M. Al-Bassam, and G. Danezis, "The road to scalable blockchain designs," *USENIX; login: magazine*, 2017.

[8] Z. Ahmed, S. M. Danish, H. K. Qureshi, and M. Lestas, "Protecting iots from mirai botnet attacks using blockchains," in *2019 IEEE 24th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, pp. 1–6, IEEE, 2019.

[9] S. Wang, L. Ouyang, Y. Yuan, X. Ni, X. Han, and F.-Y. Wang, "Blockchain-enabled smart contracts: architecture, applications, and future trends," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 11, pp. 2266–2277, 2019.

[10] W. JI, "Blockbench: A framework for analyzing private blockchains," 2017.

[11] D. Vujičić, D. Jagodić, and S. Ranđić, "Blockchain technology, bitcoin, and ethereum: A brief overview," in *2018 17th international symposium infoteh-jahorina (infoteh)*, pp. 1–6, IEEE, 2018.

[12] Q. Zhou, H. Huang, Z. Zheng, and J. Bian, "Solutions to scalability of blockchain: A survey," *IEEE Access*, vol. 8, pp. 16440–16455, 2020.

[13] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, *et al.*, "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *Proceedings of the thirteenth EuroSys conference*, pp. 1–15, 2018.

[14] M. Schäffer, M. Di Angelo, and G. Salzer, "Performance and scalability of private ethereum blockchains," in *International Conference on Business Process Management*, pp. 103–118, Springer, 2019.

[15] B. Ampel, M. Patton, and H. Chen, "Performance modeling of hyperledger sawtooth blockchain," in *2019 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pp. 59–61, IEEE, 2019.

[16] A. Dorri, S. S. Kanhere, R. Jurdak, and P. Gauravaram, "Blockchain for iot security and privacy: The case study of a smart home," in *2017 IEEE international conference on pervasive computing and communications workshops (PerCom workshops)*, pp. 618–623, IEEE, 2017.

[17] D. Khan, L. T. Jung, and M. A. Hashmani, "Systematic literature review of challenges in blockchain scalability," *Applied Sciences*, vol. 11, no. 20, p. 9372, 2021.

[18] M. Vukolić, "The quest for scalable blockchain fabric: Proof-of-work vs. bft replication," in *International workshop on open problems in network security*, pp. 112–125, Springer, 2015.

[19] L. Chen, L. Xu, N. Shah, Z. Gao, Y. Lu, and W. Shi, "On security analysis of proof-of-elapsed-time (poet)," in *International Symposium on Stabilization, Safety, and Security of Distributed Systems*, pp. 282–297, Springer, 2017.

[20] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *Annual International Cryptology Conference*, pp. 357–388, Springer, 2017.

[21] M. Castro, B. Liskov, *et al.*, "Practical byzantine fault tolerance," in *OSDI*, vol. 99, pp. 173–186, 1999.

[22] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *2014 {USENIX} Annual Technical Conference ({USENIX}{ATC} 14)*, pp. 305–319, 2014.

[23] Hyeperledger, "Hyperledger Foundation, 2021." `https://hyperledger.github.io/caliper/`, 2021. Accessed: 2020-11-02.

[24] D. Saingre, T. Ledoux, and J.-M. Menaud, "Bctmark: a framework for benchmarking blockchain technologies," in *2020 IEEE/ACS 17th International Conference on Computer Systems and Applications (AICCSA)*, pp. 1–8, IEEE, 2020.

[25] S. Kounev, K.-D. Lange, and J. von Kistowski, *Systems Benchmarking*. Springer, 2020.

[26] F. Calvão, "Crypto-miners: Digital labor and the power of blockchain technology," *Economic Anthropology*, vol. 6, no. 1, pp. 123–134, 2019.

[27] A. Boudguiga, N. Bouzerna, L. Granboulan, A. Olivereau, F. Quesnel, A. Roger, and R. Sirdey, "Towards better availability and accountability for iot updates by means of a blockchain," in *2017 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pp. 50–58, IEEE, 2017.

[28] Ethereum, "Ethereum PoA, 2021." `https://ethereum.org/en/developers/docs/consensus-mechanisms/pow/#top`, 2021. Accessed: 2021-05-02.

[29] avalanche, "avalanche-consensus, 2022." `https://docs.avax.network/overview/getting-started/avalanche-consensus/`, 2022. Accessed: 2022-04-02.

[30] polygon, "polygon-consensus, 2022." `https://polygon.technology/solutions/polygon-pos/`, 2021. Accessed: 2021-11-02.

[31] Q. Wang, R. Li, Q. Wang, and S. Chen, "Non-fungible token (nft): Overview, evaluation, opportunities and challenges," *arXiv preprint arXiv:2105.07447*, 2021.

[32] Hyeperledger, "Hyperledger Caliper, 2019." `https://www.hyperledger.org/blog/2018/03/19/measuring-blockchain-performance-with-hyperledger-caliper`, 2019. Accessed: 2019-11-02.

[33] B. Schroeder, A. Wierman, and M. Harchol-Balter, "Open versus closed: A cautionary tale," USENIX, 2006.

[34] H. Fabric, "Hypeledger Fabric, 2021." https://github.com/hyperledger/fabric/releases, 2021. Accessed: 2021-05-02.

[35] " Sawtooth Dynamic Consensus, ." https://sawtooth.hyperledger.org/docs/core/releases/latest/sysadmin_guide/about_dynamic_consensus.html, 2020. Accessed: 2020-05-20.

[36] BigchainDB, "Hyperledger, 2019." https://www.bigchaindb.com/, 2019. Accessed: 2019-11-02.

[37] Hyperledger, "Hyperledger burrow Hyperledger."

[38] Fantom, "Fantom, 2021." https://fantom.foundation/, 2021. Accessed: 2021-11-02.

[39] Polygon, " Polygon, 2021." https://polygon.technology/, 2021. Accessed: 2021-05-20.

[40] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[41] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, 2014.

[42] R. Kamath, "Food traceability on blockchain: Walmart's pork and mango pilots with ibm," *The Journal of the British Blockchain Association*, vol. 1, no. 1, p. 3712, 2018.

[43] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 4, no. 3, pp. 382–401, 1982.

[44] A. M. Antonopoulos, *Mastering Bitcoin: unlocking digital cryptocurrencies*. " O'Reilly Media, Inc.", 2014.

[45] D. Mingxiao, M. Xiaofeng, Z. Zhe, W. Xiangwei, and C. Qijun, "A review on consensus algorithm of blockchain," in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 2567–2572, IEEE, 2017.

[46] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," in *2017 IEEE International Congress on Big Data (BigData Congress)*, pp. 557–564, IEEE, 2017.

[47] S. Nakamoto, "Bitcoin whitepaper," *URL: https://bitcoin. org/bitcoin. pdf*, 2008.

[48] A. Baliga, "Understanding blockchain consensus models," in *Persistent*, 2017.

[49] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[50] J. R. Douceur, "The sybil attack," in *International workshop on peer-to-peer systems*, pp. 251–260, Springer, 2002.

[51] A. Zamyatin, N. Stifter, A. Judmayer, P. Schindler, E. Weippl, and W. J. Knottenbelt, "A wild velvet fork appears! inclusive blockchain protocol changes in practice," in *International Conference on Financial Cryptography and Data Security*, pp. 31–42, Springer, 2018.

[52] Y. Sompolinsky and A. Zohar, "Accelerating bitcoin's transaction processing," *Fast money grows on trees, not chains*, 2013.

[53] Y. N. Aung and T. Tantidham, "Review of ethereum: Smart home case study," in *2017 2nd International Conference on Information Technology (INCIT)*, pp. 1–4, IEEE, 2017.

[54] I. Makhdoom, M. Abolhasan, H. Abbas, and W. Ni, "Blockchain's adoption in iot: The challenges, and a way forward," *Journal of Network and Computer Applications*, 2018.

[55] F. Saleh, "Blockchain without waste: Proof-of-stake," *Available at SSRN 3183935*, 2019.

[56] K. Olson, M. Bowman, J. Mitchell, S. Amundson, D. Middleton, and C. Montgomery, "Sawtooth: An introduction," *The Linux Foundation, Jan*, 2018.

[57] Tendermint, "Byzantine Consensus Algorithm in Tendermint, 2019." `https://docs.tendermint.com/master/spec/consensus/consensus.html`, 2019. Accessed: 2019-11-02.

[58] D. Huang, X. Ma, and S. Zhang, "Performance analysis of the raft consensus algorithm for private blockchains," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2019.

[59] H. Sukhwani, J. M. Martínez, X. Chang, K. S. Trivedi, and A. Rindos, "Performance modeling of pbft consensus process for permissioned blockchain network (hyperledger fabric)," in *2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS)*, pp. 253–255, IEEE, 2017.

[60] linux foundation, "Linux Fondation, 2019." `https://www.linuxfoundation.org/`, 2019. Accessed: 2019-11-02.

[61] Hyperledger, "Hyperledger, 2019." https://www.hyperledger.org/, 2019. Accessed: 2019-11-02.

[62] Hyperledger, "Hyperledger burrow Hyperledger."

[63] J. Passerat-Palmbach, T. Farnan, R. Miller, M. S. Gross, H. L. Flannery, and B. Gleim, "A blockchain-orchestrated federated learning architecture for healthcare consortia," *arXiv preprint arXiv:1910.12603*, 2019.

[64] Hyeperledger, "Hyperledger INDY, 2019." https://www.hyperledger.org/projects/hyperledger-indy, 2019. Accessed: 2019-11-02.

[65] Hyeperledger, "Hyperledger IROHA, 2019." https://www.hyperledger.org/projects/iroha, 2019. Accessed: 2019-11-02.

[66] J. Kwon, "Tendermint: Consensus without mining," *Draft v. 0.6, fall*, vol. 1, p. 11, 2014.

[67] M. Castro and B. Liskov, "Practical byzantine fault tolerance and proactive recovery," *ACM Transactions on Computer Systems (TOCS)*, vol. 20, no. 4, pp. 398–461, 2002.

[68] H. Fabric, "Hyperledger Fabric Consensus, 2019." https://hyperledger-fabric.readthedocs.io/en/release-1.4/whatsnew.html, 2019. Accessed: 2019-11-02.

[69] MONAX, "MONAX, 2019." https://monax.io/, 2019. Accessed: 2019-11-02.

[70] INTEL, "INTEL, 2019." https://www.intel.com, 2019. Accessed: 2019-11-02.

[71] C. Dannen, *Introducing Ethereum and solidity*, vol. 318. Springer, 2017.

[72] V.-O. Ossip, "Ethereum blockchain and hyperledger burrow blockchain comparative analysis," tech. rep., Tech. Rep.

[73] K. W. Christopher Jr, K. D. Huynh, V. M. Roarabaugh, and T. C. Waldron III, "Method and system for utilizing benign fault occurrence to measure interrupt-blocking times," Apr. 5 1994. US Patent 5,301,312.

[74] C. Mohan, "Blockchains and databases: A new era in distributed computing," in *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pp. 1739–1740, IEEE, 2018.

[75] F. Tian, "A supply chain traceability system for food safety based on haccp, blockchain & internet of things," in *2017 International Conference on Service Systems and Service Management*, pp. 1–6, IEEE, 2017.

[76] L. Aniello, R. Baldoni, E. Gaetani, F. Lombardi, A. Margheri, and V. Sassone, "A prototype evaluation of a tamper-resistant high performance blockchain-based transaction log for a distributed database," in *2017 13th European Dependable Computing Conference (EDCC)*, pp. 151–154, IEEE, 2017.

[77] T. Min and W. Cai, "Portrait of decentralized application users: an overview based on large-scale ethereum data," *CCF Transactions on Pervasive Computing and Interaction*, pp. 1–18, 2022.

[78] W. Cai, Z. Wang, J. B. Ernst, Z. Hong, C. Feng, and V. C. Leung, "Decentralized applications: The blockchain-empowered software system," *IEEE Access*, vol. 6, pp. 53019–53033, 2018.

[79] IBM, "SmartContract, 2021." `https://www.ibm.com/topics/smart-contracts`, 2022. Accessed: 2022-09-02.

[80] J. Benet, "Ipfs-content addressed, versioned, p2p file system," *arXiv preprint arXiv:1407.3561*, 2014.

[81] J. Arcenegui, R. Arjona, R. Román, and I. Baturone, "Secure combination of iot and blockchain by physically binding iot devices to smart non-fungible tokens using pufs," *Sensors*, vol. 21, no. 9, p. 3119, 2021.

[82] S. Barrington, "The role of metadata in non-fungible tokens: Marketplace analysis and collection organization," *arXiv preprint arXiv:2209.14395*, 2022.

[83] Sofi, "Minting an NFT, 2021." `https://www.sofi.com/learn/content/what-is-nft-minting/`, 2021. Accessed: 2021-11-02.

[84] M. Rae, "Analyzing the nft mania: Is a jpg worth millions?," in *SAGE Business Cases*, SAGE Publications: SAGE Business Cases Originals, 2021.

[85] M. Rasolroveicy and M. Fokaefs, "Performance and cost evaluation of public blockchain: An nft marketplace case study," in *2022 4th Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS)*, pp. 79–86, IEEE, 2022.

[86] Q. Nasir, I. A. Qasse, M. Abu Talib, and A. B. Nassif, "Performance analysis of hyperledger fabric platforms," *Security and Communication Networks*, vol. 2018, 2018.

[87] Y. Hao, Y. Li, X. Dong, L. Fang, and P. Chen, "Performance analysis of consensus algorithm in private blockchain," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 280–285, IEEE, 2018.

[88] S. Pongnumkul, C. Siripanpornchana, and S. Thajchayapong, "Performance analysis of private blockchain platforms in varying workloads," in *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, pp. 1–6, IEEE, 2017.

[89] M. Kuzlu, M. Pipattanasomporn, L. Gurses, and S. Rahman, "Performance analysis of a hyperledger fabric blockchain framework: Throughput, latency and scalability," in *2019 IEEE International Conference on Blockchain (Blockchain)*, pp. 536–540, IEEE, 2019.

[90] C. Saraf and S. Sabadra, "Blockchain platforms: A compendium," in *2018 IEEE International Conference on Innovative Research and Development (ICIRD)*, pp. 1–6, IEEE, 2018.

[91] F. Österlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-level sensor network simulation with cooja," in *First IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp 2006)*, 2006.

[92] S. Alrubei, J. Rigelsford, C. Willis, and E. Ball, "Ethereum blockchain for securing the internet of things: Practical implementation and performance evaluation," in *2019 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, pp. 1–5, IEEE, 2019.

[93] O. Novo, "Scalable access management in iot using blockchain: a performance evaluation," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4694–4701, 2018.

[94] M. Birim, H. E. Ari, and E. Karaarslan, "Gohammer blockchain performance test tool," *Journal of Emerging Computer Technologies*, vol. 1, no. 2, pp. 31–33, 2021.

[95] G. Shapiro, C. Natoli, and V. Gramoli, "The performance of byzantine fault tolerant blockchains," in *2020 IEEE 19th International Symposium on Network Computing and Applications (NCA)*, pp. 1–8, IEEE, 2020.

[96] A. Baliga, N. Solanki, S. Verekar, A. Pednekar, P. Kamat, and S. Chatterjee, "Performance characterization of hyperledger fabric," in *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*, pp. 65–74, IEEE, 2018.

[97] M. Mazzoni, A. Corradi, and V. Di Nicola, "Performance evaluation of permissioned blockchains for financial applications: The consensys quorum case study," *Blockchain: Research and applications*, vol. 3, no. 1, p. 100026, 2022.

[98] D. Gupta, L. Perronne, and S. Bouchenak, "Bft-bench: Towards a practical evaluation of robustness and effectiveness of bft protocols," in *IFIP International Conference on Distributed Applications and Interoperable Systems*, pp. 115–128, Springer, 2016.

[99] M. Di Pierro, "What is the blockchain?," *Computing in Science & Engineering*, vol. 19, no. 5, pp. 92–95, 2017.

[100] M. Rasolroveicy and M. Fokaefs, "Performance evaluation of distributed ledger technologies for iot data registry: A comparative study," in *2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*, pp. 137–144, IEEE, 2020.

[101] M. Rasolroveicy, W. Haouari, and M. Fokaefs, "Public or private? a techno-economic analysis of blockchain," in *Proceedings of the 31st Annual International Conference on Computer Science and Software Engineering*, pp. 83–92, 2021.

[102] Avalanche, "Avalanche, 2021." `https://www.avax.network/`, 2021. Accessed: 2021-11-02.

[103] M. R. M. . F. M., "Impact of ddos attacks on the performance of blockchain consensus as an iot data registry: An empirical study." CASCON, the 32th Annual International Conference on Computer Science and Software Engineering, 2022.

[104] R. Singh, S. Tanwar, and T. P. Sharma, "Utilization of blockchain for mitigating the distributed denial of service attacks," *Security and Privacy*, vol. 3, no. 3, p. e96, 2020.

[105] S. Wani, M. Imthiyas, H. Almohamedh, K. M. Alhamed, S. Almotairi, and Y. Gulzar, "Distributed denial of service (ddos) mitigation using blockchain—a comprehensive insight," *Symmetry*, vol. 13, no. 2, p. 227, 2021.

[106] R. F. Ibrahim, Q. Abu Al-Haija, and A. Ahmad, "Ddos attack prevention for internet of thing devices using ethereum blockchain technology," *Sensors*, vol. 22, no. 18, p. 6806, 2022.

[107] P. Horn, "Autonomic computing: Ibm's perspective on the state of information technology, ibm corporation," 2001.

[108] M. Rasolroveicy and M. Fokaefs, "Dynamic reconfiguration of consensus protocol for iot data registry on blockchain," in *Proceedings of the 30th Annual International Conference on Computer Science and Software Engineering*, pp. 227–236, 2020.

[109] K. Wang, M. Liu, X. Jiang, C. Yang, and H. Zhang, "A novel vehicle blockchain model based on hyperledger fabric for vehicle supply chain management," in *International Conference on Blockchain and Trustworthy Systems*, pp. 732–739, Springer, 2019.

[110] M. Dabbagh, M. Kakavand, M. Tahir, and A. Amphawan, "Performance analysis of blockchain platforms: Empirical evaluation of hyperledger fabric and ethereum," in *2020 IEEE 2nd International Conference on Artificial Intelligence in Engineering and Technology (IICAIET)*, pp. 1–6, IEEE, 2020.

[111] geth, "Geth private network, 2019." `https://geth.ethereum.org/docs/interface/private-network`, 2019. Accessed: 2021-06-10.

[112] P. Ekparinya, V. Gramoli, and G. Jourjon, "The attack of the clones against proof-of-authority," *arXiv preprint arXiv:1902.10244*, 2019.

[113] B. academy, "poaexplained 2022," 2022.

[114] D. Ongaro and J. Ousterhout, "The raft consensus algorithm," *Lecture Notes CS*, vol. 190, 2015.

[115] E. Go, "Ethereum Go 2021," 2021.

[116] H. Sawtooth, "Hyperledger sawtooth 1.2 (chime)." `https://sawtooth.hyperledger.org/release/chime/`, 2019. Accessed: 2022-01-10.

[117] G. D. Ruxton, "The unequal variance t-test is an underused alternative to student's t-test and the mann–whitney u test," *Behavioral Ecology*, vol. 17, no. 4, pp. 688–690, 2006.

[118] P. E. McKnight and J. Najab, "Mann-whitney u test," *The Corsini encyclopedia of psychology*, pp. 1–1, 2010.

[119] P. Royston, "Approximating the shapiro-wilk w-test for non-normality," *Statistics and computing*, vol. 2, no. 3, pp. 117–119, 1992.

[120] E. W. Weisstein, "Bonferroni correction," *https://mathworld. wolfram. com/*, 2004.

[121] F. Xia, L. T. Yang, L. Wang, and A. Vinel, "Internet of things," *International Journal of Communication Systems*, vol. 25, no. 9, p. 1101, 2012.

[122] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "Ddos in the iot: Mirai and other botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017.

[123] H. Sukhwani, "Performance modeling & analysis of hyperledger fabric (permissioned blockchain network)," *Duke University: Duke, UK*, 2018.

[124] K. Sultan, U. Ruhi, and R. Lakhani, "Conceptualizing blockchains: Characteristics & applications," *arXiv preprint arXiv:1806.03693*, 2018.

[125] K. Chodorow, *MongoDB: the definitive guide: powerful and scalable data storage.* " O'Reilly Media, Inc.", 2013.

[126] G. Papadodimas, G. Palaiokrasas, A. Litke, and T. Varvarigou, "Implementation of smart contracts for blockchain based iot applications," in *2018 9th International Conference on the Network of the Future (NOF)*, pp. 60–67, IEEE, 2018.

[127] R. Paul, P. Baidya, S. Sau, K. Maity, S. Maity, and S. B. Mandal, "Iot based secure smart city architecture using blockchain," in *2018 2nd International Conference on Data Science and Business Analytics (ICDSBA)*, pp. 215–220, IEEE, 2018.

[128] C. Gorenflo, S. Lee, L. Golab, and S. Keshav, "Fastfabric: Scaling hyperledger fabric to 20,000 transactions per second," *arXiv preprint arXiv:1901.00910*, 2019.

[129] A. Banafa, "Iot and blockchain convergence: benefits and challenges," *IEEE Internet of Things*, 2017.

[130] N. Komninos, E. Philippou, and A. Pitsillides, "Survey in smart grid and smart home security: Issues, challenges and countermeasures," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 1933–1954, 2014.

[131] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm (extended version)," 2013.

[132] A. Choudhury, G. Garimella, A. Patra, D. Ravi, and P. Sarkar, "Crash-tolerant consensus in directed graph revisited," in *International Colloquium on Structural Information and Communication Complexity*, pp. 55–71, Springer, 2018.

[133] M. S. Arbabi and M. Shajari, "Decentralized and secure delivery network of iot update files based on ethereum smart contracts and blockchain technology," in *Proceedings of the 29th Annual International Conference on Computer Science and Software Engineering*, pp. 110–119, IBM Corp., 2019.

[134] V. Kaushal and A. G. Bala, *Autonomic Fault Tolerance Using HAProxy in Cloud Enviorment*. PhD thesis, 2011.

[135] B. Ramprasad, M. Fokaefs, J. Mukherjee, and M. Litoiu, "Emu-iot-a virtual internet of things lab," in *2019 IEEE International Conference on Autonomic Computing (ICAC)*, pp. 73–83, IEEE, 2019.

[136] D. R. API, "Docker Remote API, 2019." `https://docs.docker.com/engine/api/v1.24/`, 2019. Accessed: 2019-11-02.

[137] N. Andola, M. Gogoi, S. Venkatesan, S. Verma, *et al.*, "Vulnerabilities on hyperledger fabric," *Pervasive and Mobile Computing*, vol. 59, p. 101050, 2019.

[138] M. Warkentin and C. Orgeron, "Using the security triad to assess blockchain technology in public sector applications," *International Journal of Information Management*, vol. 52, p. 102090, 2020.

[139] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, *et al.*, "Understanding the mirai botnet," in *26th {USENIX} Security Symposium ({USENIX} Security 17)*, pp. 1093–1110, 2017.

[140] S. Khezr, M. Moniruzzaman, A. Yassine, and R. Benlamri, "Blockchain technology in healthcare: A comprehensive review and directions for future research," *Applied sciences*, vol. 9, no. 9, p. 1736, 2019.

[141] P. Szilágyi, "Ethereum PoA, 2021." `https://eips.ethereum.org/EIPS/eip-225`, 2021. Accessed: 2021-05-02.

[142] H. Malik, A. Manzoor, M. Ylianttila, and M. Liyanage, "Performance analysis of blockchain based smart grids with ethereum and hyperledger implementations," in *2019 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, pp. 1–5, IEEE, 2019.

[143] A. Aswin and B. Kuriakose, "An analogical study of hyperledger fabric and ethereum," in *Intelligent Communication Technologies and Virtual Mobile Networks*, pp. 412–420, Springer, 2019.

[144] C. Nedelcu, *Nginx HTTP Server*. Packt Publishing Ltd, 2015.

[145] " Ethereum Raw JSON RP, ." `https://documenter.getpostman.com/view/4117254/ethereum-json-rpc/RVu7CT5J`, 2021. Accessed: 2021-06-14.

[146] Ethereum, "Ethereum Dimensions 2021," 2021. Accessed: 2021-05-02.

[147] 2miners, "poaexplained 2022," 2022.

[148] K. Iyer and C. Dannen, "The ethereum development environment," in *Building games with ethereum smart contracts*, pp. 19–36, Springer, 2018.

[149] " Hyperledger Fabric docs, ." `https://hyperledger-fabric.readthedocs.io/en/release-2.3/`, 2021. Accessed: 2021-06-14.

[150] " Raft Consensus, ." `https://sawtooth.hyperledger.org/docs/raft/nightly/master/introduction.html`, 2021. Accessed: 2021-01-20.

[151] readthedocs.io, "web3 explanation 2021," 2021.

[152] E. J. RPC, "Ethereum JSON RPC 2021," 2021.

[153] S. Sharkey and H. Tewari, "Alt-pow: an alternative proof-of-work mechanism," in *2019 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPCON)*, pp. 11–18, IEEE, 2019.

[154] Binance, "Binance Smart Chain, 2021." `https://www.binance.org/en/smartChain`, 2021. Accessed: 2021-11-02.

[155] Tron, "Tron 2021," 2021.

[156] P. R. Nair and D. R. Dorai, "Evaluation of performance and security of proof of work and proof of stake using blockchain," in *2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*, pp. 279–283, IEEE, 2021.

[157] Hyeperledger, "caricatureart , 2021." `https://www.caricatureart.com/why-authenticity-is-important-in-art/`, 2021. Accessed: 2021-05-02.

[158] S. Namasudra and G. C. Deka, *Applications of blockchain in healthcare*. Springer, 2021.

[159] D. Tanana, "Avalanche blockchain protocol for distributed computing security," in *2019 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*, pp. 1–3, IEEE, 2019.

[160] H. Tang, Y. Shi, and P. Dong, "Public blockchain evaluation using entropy and topsis," *Expert Systems with Applications*, vol. 117, pp. 204–210, 2019.

[161] opensea, "opensea, 2022." https://opensea.io/, 2022. Accessed: 2022-04-02.

[162] J. Potts and E. Rennie, "Web3 and the creative industries: how blockchains are reshaping business models," in *A Research Agenda for Creative Industries*, Edward Elgar Publishing, 2019.

[163] flow, "Flow, 2021." https://www.onflow.org/technical-paper, 2021. Accessed: 2021-11-02.

[164] Flow, "Flow-nodes, 2022." https://github.com/onflow/flow/blob/master/nodeoperators/NodeOperatorList.md, 2022. Accessed: 2022-04-02.

[165] chiliz, "chiliz, 2021." https://www.chiliz.com/docs/CHZ_whitepaper.pdf, 2021. Accessed: 2021-11-02.

[166] Z. Wang, L. Yang, Q. Wang, D. Liu, Z. Xu, and S. Liu, "Artchain: blockchain-enabled platform for art marketplace," in *2019 IEEE International Conference on Blockchain (Blockchain)*, pp. 447–454, IEEE, 2019.

[167] P. Rodrigo, J. Pouwelse, and M. d. Vos, "Unicon: Universal and scalable infrastructure for digital asset management," in *Proceedings of the 2nd International Workshop on Distributed Infrastructure for Common Good*, pp. 5–10, 2021.

[168] P. Otte, M. de Vos, and J. Pouwelse, "Trustchain: A sybil-resistant scalable blockchain," *Future Generation Computer Systems*, vol. 107, pp. 770–780, 2020.

[169] metamask, "Crypto Wallet web3, 2021." https://metamask.io/, 2021. Accessed: 2021-11-02.

[170] nftschool, "NFT school, 2022." https://nftschool.dev/tutorial/lazy-minting/, 2022. Accessed: 2022-04-02.

[171] moralis, "RPCnode, 2022." https://moralis.io/ethereum-rpc-nodes-what-they-are-and-why-you-shouldnt-use-them/, 2022. Accessed: 2022-04-02.

[172] Ethereum, "erc1155, 2022." https://eips.ethereum.org/EIPS/eip-1155, 2022. Accessed: 2022-04-02.

[173] polygon po, "polygon-po, 2021." https://polygon.technology/solutions/polygon-pos/, 2021. Accessed: 2021-11-02.

[174] lachesis, "lachesis, 2022." https://www.fantom.foundation/lachesis-consensus-algorithm/, 2022. Accessed: 2022-05-02.

[175] R. G. Van and F. Drake, "Python 3 reference manual," *Scotts Valley, CA: CreateSpace*, vol. 10, p. 1593511, 2009.

[176] O. Kramer, "Scikit-learn," in *Machine learning for evolution strategies*, pp. 45–53, Springer, 2016.

[177] T. Chen, T. He, M. Benesty, V. Khotilovich, Y. Tang, H. Cho, K. Chen, *et al.*, "Xgboost: extreme gradient boosting," *R package version 0.4-2*, vol. 1, no. 4, pp. 1–4, 2015.

[178] M. Pal, "Random forest classifier for remote sensing classification," *International journal of remote sensing*, vol. 26, no. 1, pp. 217–222, 2005.

[179] A. J. Myles, R. N. Feudale, Y. Liu, N. A. Woody, and S. D. Brown, "An introduction to decision tree modeling," *Journal of Chemometrics: A Journal of the Chemometrics Society*, vol. 18, no. 6, pp. 275–285, 2004.

[180] G. A. Seber and A. J. Lee, *Linear regression analysis*. John Wiley & Sons, 2012.

[181] T. Chai and R. R. Draxler, "Root mean square error (rmse) or mean absolute error (mae)?–arguments against avoiding rmse in the literature," *Geoscientific model development*, vol. 7, no. 3, pp. 1247–1250, 2014.

[182] medium, "MAE and RMSE, 2021." https://medium.com/human-in-a-machine-world/mae-and-rmse-which-metric-is-better-e60ac3bde13d, 2021. Accessed: 2021-11-02.

[183] B. H. Menze, B. M. Kelm, R. Masuch, U. Himmelreich, P. Bachert, W. Petrich, and F. A. Hamprecht, "A comparison of random forest and its gini importance with standard chemometric methods for the feature selection and classification of spectral data," *BMC bioinformatics*, vol. 10, no. 1, pp. 1–16, 2009.

[184] towardsdatascience, "Gini Index, 2022." https://tinyurl.com/2p998va7, 2022. Accessed: 2022-04-02.

[185] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[186] J. Howell, "Number of connected iot devices will surge to 125 billion by 2030," *IHS markit says*, 2017.

[187] R. Vishwakarma and A. K. Jain, "A survey of ddos attacking techniques and defence mechanisms in the iot network," *Telecommunication Systems*, vol. 73, no. 1, pp. 3–25, 2020.

[188] Q. Shafi and A. Basit, "Ddos botnet prevention using blockchain in software defined internet of things," in *2019 16th International Bhurban Conference on Applied Sciences and Technology (IBCAST)*, pp. 624–628, IEEE, 2019.

[189] U. Javaid, A. K. Siang, M. N. Aman, and B. Sikdar, "Mitigating Iot device based ddos attacks using blockchain," in *Proceedings of the 1st Workshop on Cryptocurrencies and Blockchains for Distributed Systems*, pp. 71–76, 2018.

[190] N. Kshetri, "Can blockchain strengthen the internet of things?," *IT professional*, vol. 19, no. 4, pp. 68–72, 2017.

[191] P. Szalachowski, D. Reijsbergen, I. Homoliak, and S. Sun, "{StrongChain}: Transparent and collaborative {Proof-of-Work} consensus," in *28th USENIX Security Symposium (USENIX Security 19)*, pp. 819–836, 2019.

[192] S. Schorradt, E. Bajramovic, and F. Freiling, "On the feasibility of secure logging for industrial control systems using blockchain," in *Proceedings of the Third Central European Cybersecurity Conference*, pp. 1–6, 2019.

[193] S. Sun, R. Du, S. Chen, and W. Li, "Blockchain-based iot access control system: towards security, lightweight, and cross-domain," *IEEE Access*, vol. 9, pp. 36868–36878, 2021.

[194] S. A. Ahmed, "A performance study of hyperledger fabric in a smart home and iot environment," Master's thesis, 2019.

[195] M. Sabt, M. Achemlal, and A. Bouabdallah, "Trusted execution environment: what it is, and what it is not," in *2015 IEEE Trustcom/BigDataSE/ISPA*, vol. 1, pp. 57–64, IEEE, 2015.

[196] H. Howard, "Arc: analysis of raft consensus," tech. rep., University of Cambridge, Computer Laboratory, 2014.

[197] V. Dhillon, D. Metcalf, and M. Hooper, "The hyperledger project," in *Blockchain enabled applications*, pp. 139–149, Springer, 2017.

[198] M. Grinberg, *Flask web development: developing web applications with python.* " O'Reilly Media, Inc.", 2018.

[199] C. Nedelcu, *Nginx HTTP Server: Adopt Nginx for Your Web Applications to Make the Most of Your Infrastructure and Serve Pages Faster Than Ever*. Packt Publishing Ltd, 2010.

[200] S. R. Zeebaree, K. Jacksi, and R. R. Zebari, "Impact analysis of syn flood ddos attack on haproxy and nlb cluster-based web servers," *Indones. J. Electr. Eng. Comput. Sci*, vol. 19, no. 1, pp. 510–517, 2020.

[201] Z. Shi, H. Zhou, Y. Hu, S. Jayachander, C. de Laat, and Z. Zhao, "Operating permissioned blockchain in clouds: A performance study of hyperledger sawtooth," in *2019 18th International Symposium on Parallel and Distributed Computing (ISPDC)*, pp. 50–57, IEEE, 2019.

[202] L. Liang, K. Zheng, Q. Sheng, and X. Huang, "A denial of service attack method for an iot system," in *2016 8th international conference on Information Technology in Medicine and Education (ITME)*, pp. 360–364, IEEE, 2016.

[203] W. Eddy *et al.*, "Tcp syn flooding attacks and common mitigations," tech. rep., 2007.

[204] J. Turnbull, *The Docker Book: Containerization is the new virtualization*. James Turnbull, 2014.

[205] M. P. Fay and M. A. Proschan, "Wilcoxon-mann-whitney or t-test? on assumptions for hypothesis tests and multiple interpretations of decision rules," *Statistics surveys*, vol. 4, p. 1, 2010.

[206] N. Smalheiser, *Data literacy: How to make your experiments robust and reproducible chapter 12*. Academic Press, 2017.

[207] G. Macbeth, E. Razumiejczyk, and R. D. Ledesma, "Cliff's delta calculator: A non-parametric effect size program for two groups of observations," *Universitas Psychologica*, vol. 10, no. 2, pp. 545–555, 2011.

[208] B. Kitchenham, L. Madeyski, D. Budgen, J. Keung, P. Brereton, S. Charters, S. Gibbs, and A. Pohthong, "Robust statistical methods for empirical software engineering," *Empirical Software Engineering*, vol. 22, no. 2, pp. 579–630, 2017.

[209] M. Sachdeva, K. Kumar, G. Singh, and K. Singh, "Performance analysis of web service under ddos attacks," in *2009 IEEE International Advance Computing Conference*, pp. 1002–1007, IEEE, 2009.

[210] S. J. Alsunaidi and F. A. Alhaidari, "A survey of consensus algorithms for blockchain technology," in *2019 International Conference on Computer and Information Sciences (ICCIS)*, pp. 1–6, IEEE, 2019.

[211] J. Romano, J. D. Kromrey, J. Coraggio, and J. Skowronek, "Appropriate statistics for ordinal level data: Should we really be using t-test and cohen'sd for evaluating group differences on the nsse and other surveys," in *annual meeting of the Florida Association of Institutional Research*, vol. 177, 2006.

[212] M. A. Alsmirat, Y. Jararweh, I. Obaidat, and B. B. Gupta, "Internet of surveillance: a cloud supported large-scale wireless surveillance system," *The Journal of Supercomputing*, vol. 73, no. 3, pp. 973–992, 2017.

[213] R. Casado-Vara, P. Chamoso, F. De la Prieta, J. Prieto, and J. M. Corchado, "Non-linear adaptive closed-loop control system for improved efficiency in iot-blockchain management," *Information Fusion*, vol. 49, pp. 227–239, 2019.

[214] M. Bastiaan, "Preventing the 51%-attack: a stochastic analysis of two phase proof of work in bitcoin," in *Availab le at http://referaat. cs. utwente. nl/conference/22/paper/7473/preventingthe-51-attack-a-stochasticanalysis-oftwo-phase-proof-of-work-in-bitcoin. pdf*, 2015.

[215] " DevMode Consensus, ." `https://sawtooth.hyperledger.org/docs/core/nightly/1-2/sysadmin_guide/about_dynamic_consensus.html#devmode-consensus-label`, 2020. Accessed: 2020-05-20.

[216] " Sawtooth Documentation for Supporting Different Consensus Protocols, ." `https://sawtooth.hyperledger.org/faq/consensus//`, 2021. Accessed: 2021-01-20.

[217] W. Liang, M. Tang, J. Long, X. Peng, J. Xu, and K.-C. Li, "A secure fabric blockchain-based data transmission technique for industrial internet-of-things," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3582–3592, 2019.

[218] M. Alaslani, F. Nawab, and B. Shihada, "Blockchain in iot systems: End-to-end delay evaluation," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8332–8344, 2019.

[219] L. Zhao and J. Yu, "Evaluating dag-based blockchains for iot," in *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pp. 507–513, IEEE, 2019.

[220] S. Popov, "The tangle," *cit. on*, p. 131, 2016.

[221] M. Divya and N. B. Biradar, "Iota-next generation block chain," *International Journal Of Engineering And Computer Science*, vol. 7, no. 04, pp. 23823–23826, 2018.

[222] Y. Li, B. Cao, M. Peng, L. Zhang, L. Zhang, D. Feng, and J. Yu, "Direct acyclic graph-based ledger for internet of things: Performance and security analysis," *IEEE/ACM Transactions on Networking*, 2020.

[223] N. Alzahrani and N. Bulusu, "A new product anti-counterfeiting blockchain using a truly decentralized dynamic consensus protocol," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 12, p. e5232, 2020.

[224] A. Varga, "Omnet++," in *Modeling and tools for network simulation*, pp. 35–59, Springer, 2010.

[225] S. Liaskos, B. Wang, and N. Alimohammadi, "Blockchain networks as adaptive systems," in *2019 IEEE/ACM 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, pp. 139–145, IEEE, 2019.

[226] R. Casado-Vara, F. De la Prieta, S. Rodriguez, I. Sitton, J. L. Calvo-Rolle, G. K. Venayagamoorthy, P. Vega, and J. Prieto, "Adaptive fault-tolerant tracking control algorithm for iot systems: Smart building case study," in *International Workshop on Soft Computing Models in Industrial and Environmental Applications*, pp. 481–490, Springer, 2019.

[227] G. Hovland and J. Kucera, "Nonlinear feedback control and stability analysis of a proof-of-work blockchain," 2017.

[228] " PoET Consensus, ." `https://sawtooth.hyperledger.org/docs/core/releases/latest/architecture/poet.html`, 2020. Accessed: 2020-05-20.

[229] M. Fokaefs, C. Barna, R. Veleda, M. Litoiu, J. Wigglesworth, and R. Mateescu, "Enabling devops for containerized data-intensive applications: an exploratory study," in *Proceedings of the 26th Annual International Conference on Computer Science and Software Engineering*, pp. 138–148, 2016.

[230] Z. Zheng, S. Xie, H.-N. Dai, W. Chen, X. Chen, J. Weng, and M. Imran, "An overview on smart contracts: Challenges, advances and platforms," *Future Generation Computer Systems*, vol. 105, pp. 475–491, 2020.

[231] A. Khatoon, "A blockchain-based smart contract system for healthcare management," *Electronics*, vol. 9, no. 1, p. 94, 2020.

[232] F. Wilhelmi, S. Barrachina-Muñoz, and P. Dini, "End-to-end latency analysis and optimal block size of proof-of-work blockchain applications," *arXiv preprint arXiv:2202.01497*, 2022.

[233] P. De Filippi, "Blockchain technology and decentralized governance: the pitfalls of a trustless dream," *De Filippi, P.(2019). Blockchain Technology and Decentralized Governance: The Pitfalls of a Trustless Dream. Decentralized Thriving: Governance and Community on the Web*, vol. 3, 2019.

[234] M. T. Hammi, P. Bellot, and A. Serhrouchni, "Bctrust: A decentralized authentication blockchain-based mechanism," in *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–6, IEEE, 2018.

[235] X. Wang, Z. Yang, X. Chen, and W. Liu, "Distributed inference for linear support vector machine," *Journal of machine learning research*, vol. 20, 2019.

[236] D. C. Montgomery, E. A. Peck, and G. G. Vining, *Introduction to linear regression analysis*. John Wiley & Sons, 2021.

[237] A. Graves, "Long short-term memory," *Supervised sequence labelling with recurrent neural networks*, pp. 37–45, 2012.

[238] Y. Wang, W. Liao, and Y. Chang, "Gated recurrent unit network-based short-term photovoltaic forecasting," *Energies*, vol. 11, no. 8, p. 2163, 2018.

[239] J. Wang, F. Hu, and L. Li, "Deep bi-directional long short-term memory model for short-term traffic flow prediction," in *International conference on neural information processing*, pp. 306–316, Springer, 2017.

[240] A. Gulli and S. Pal, *Deep learning with Keras*. Packt Publishing Ltd, 2017.

[241] M. Abadi, "Tensorflow: learning functions at scale," in *Proceedings of the 21st ACM SIGPLAN International Conference on Functional Programming*, pp. 1–1, 2016.

[242] A. Ash and M. Shwartz, "R2: a useful measure of model performance when predicting a dichotomous outcome," *Statistics in medicine*, vol. 18, no. 4, pp. 375–384, 1999.

[243] S. S. Sepasgozar and S. Pierre, "Network traffic prediction model considering road traffic parameters using artificial intelligence methods in vanet," *IEEE Access*, vol. 10, pp. 8227–8242, 2022.

[244] L. Bottou, "Stochastic gradient descent tricks," in *Neural networks: Tricks of the trade*, pp. 421–436, Springer, 2012.

[245] Q. Li, C. Tai, and E. Weinan, "Stochastic modified equations and adaptive stochastic gradient algorithms," in *International Conference on Machine Learning*, pp. 2101–2110, PMLR, 2017.

[246] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[247] C. Oh, S. Han, and J. Jeong, "Time-series data augmentation based on interpolation," *Procedia Computer Science*, vol. 175, pp. 64–71, 2020.

[248] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.