A Simulation Framework for Evaluating the Performance of Blockchain-based IoT Ecosystems

Adel Mohammed Albshri

Department of Computing

Newcastle University

This dissertation is submitted for the degree of

Doctor of Philosophy

School of Computing

Thursday 5th September, 2024

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Adel Mohammed Albshri

Thursday 5th September, 2024

Acknowledgements

First, I would like to express my gratitude to Allah (God) for bestowing on me the fortitude, resilience, wisdom, and the opportunity to successfully complete my research. Without His divine blessings, this achievement would not have been attainable.

I would like to express my sincere thanks to my supervisors, Dr. Ellis Solaiman, Prof. Rajiv Ranjan, and Dr. Karan Mitra. The research work presented in this thesis would not have been possible without their exceptional supervision, guidance, encouragement, invaluable support, and constructive feedback. I also would like to take the opportunity to thank the members of my examining committee, Prof. Martin White and Dr. Charles Morisset.

I extend my sincere thanks and appreciation to Dr. Ali Alzubaidi for his invaluable contributions to my PhD journey, including his insights, fruitful discussions, constructive reviews, and supportive feedback. Additionally, I express my gratitude to my colleague, Dr. Bakri Awaji, for his support during the initial phases of my journey. My thanks and appreciation also go to everyone who has contributed to my PhD journey.

I am deeply grateful to the University of Jeddah, Saudi Arabia, for granting me the opportunity to pursue my doctoral studies and for their generous financial support. I also extend my sincere appreciation to Newcastle University for their unwavering support throughout my academic journey, including generous funding for various research experiments and providing the necessary equipment, all of which were crucial to the successful completion of this study.

Finally, I would like to express my appreciation to my wife, Atheer, for her constant support during this journey. Her invaluable assistance was crucial in enabling me to complete this dissertation. To my daughters, Shaden and Sula, I extend my profound love and gratitude

for their understanding and encouragement. I also cannot express enough my appreciation and gratitude for all the support and encouragement I received from my parents and siblings.

Abstract

Recently, it has been appealing to integrate Blockchain with IoT in several domains, such as healthcare and smart cities. This integration facilitates the decentralized processing of IoT data, enhancing cybersecurity by ensuring data integrity, preventing tampering, and strengthening privacy through decentralized trust mechanisms and resilient security measures. These features create a secure and reliable environment, mitigating potential cyber threats while ensuring non-repudiation and higher availability. However, Blockchain performance is questionable when handling massive data sets generated by complex and heterogeneous IoT applications. Thus, whether the Blockchain performance meets expectations will significantly influence the overall viability of integration. Therefore, it is crucial to evaluate the feasibility of integrating IoT and Blockchain and examine the technology readiness level before the production stage. This thesis addresses this matter by extensively investigating approaches to the performance evaluation of Blockchain-based IoT solutions. Firstly, it systematically reviews existing Blockchain simulators and identifies their strengths and limitations. Secondly, due to the lack of existing blockchain simulators specifically tailored for IoT, this thesis contributes a novel blockchain-based IoT simulator which enables investigation of blockchain performance based on adaptable design configuration choices of IoT infrastructure. The simulator benefits from lessons learnt about the strengths and limitations of existing works and considers various design requirements and views collected through questioners and focus groups of domain experts. Third, the thesis recognises the shortcomings of blockchain simulators, such as support for smart contracts. Therefore, it contributes a middleware that leverages IoT simulators to benchmark real blockchain platforms' performance, namely Hyperledger Fabric. It resolves challenges related to integrating distinctive environments:

simulated IoT models with real Blockchain ecosystems. Lastly, this thesis employs Machine Learning (ML) techniques for predicting blockchain performance based on predetermined configurations. Contrariwise, it also utilises ML techniques to recommend the optimal configurations for achieving the desired level of blockchain performance.

Table of contents

List of figures

List of tables

Nomenclature

Acronyms / Abbreviations

ABC Artificial Bee Colony

ACO Ant Colony Optimization

AI Artificial Intelligence

APIs Application Programming Interfaces

CA Certificate Authority

CLOC Count Lines of Code

CLOC Count Lines of Code

CoAP Constrained Application Protocol

CRUD Create, Read, Update, and Delete

DAG Directed Acyclic Graph

DApp Decentralised Applications

DDoS Distributed Denial of Service Attacks

DeFi Decentralised Finance

DLT Distributed Ledger Technology

EVM Ethereum Virtual Machine

GQM Goal Question Metrics

GUI Graphical User Interface

GWO Grey Wolf Optimization

HHO Harris Hawk Optimization

HLF Hyperledger Fabric

HLF Hyperledger Fabric

IED Intelligent Electronic Devices

IoT Internet of Things

IoTSP IoT Service Provider

ISO Improved Salp Optimization

kNN k Nearest Neighbour

ML Machine Learning

MQTT Message Queuing Telemetry Transport

P2P Peer-to-Peer

PKI Public key Infrastructure

PLC Programmable Logic Controllers

PoA Proof of Authority

PoA Proof of Authority

PoET Proof of Elapsed Time

PoS Proof of Stake

PoS Proof-of-Stake

PoW Proof of Work

PSO Particle Swarm Optimization

RFID Radio Frequency Identification

RMSE Root Mean Square Error

RST Rough Set Theory

RTU Remote Terminal Units

SCADA Supervisory Control and Data Acquisition

SDKs Software Development Kits

SO Salp Optimization

SPOF Single Point of Failure

SPSS Statistical Package for the Social Sciences

SVM Support Vector Machine

TPS Transactions Per Second

TPS Transactions Per second

TRL Technology Readiness Level

URI Uniform Resource Identifier

ZKP Zero Knowledge Proof

Chapter 1

Introduction

1.1 Introduction

Recently, there has been an increasing interest in the convergence between IoT and Blockchain to unlock the potential of both paradigms [1]. The Internet of Things (IoT) has emerged as a transformative technology, connecting billions of devices and enabling innovative applications in various domains, such as healthcare, transportation, manufacturing, and smart cities [2]. IoT devices, equipped with sensors and communication capabilities, generate large amounts of data that can be analysed to derive valuable insights and optimise processes. However, as the number of IoT devices grows exponentially, concerns about security, privacy, and scalability have become increasingly prominent [3]. Traditional centralised architectures, where all data and control flow through a single point, pose significant risks and limitations for IoT systems. Centralised servers are vulnerable to attacks, single points of failure, and performance bottlenecks, which can compromise the integrity, availability, and efficiency of IoT applications [4].

Several studies foresee the opportunities of leveraging blockchain technology to mitigate IoT issues and even revisit the way of thinking about IoT solutions [5]. This is due to several features associated with blockchain technology, such as decentralisation, high availability, mitigation of single points of failure, immutability, transparency, and traceability

[6]. Blockchain enables secure data sharing among untrusted parties, prevents tampering with stored data, and provides a verifiable audit trail for IoT transactions [7].

Despite the promising opportunities of integrating blockchain with IoT, there are several challenges and complexities along the way. For instance, IoT devices are often resource-constrained, with limited processing power, storage capacity, and bandwidth [8]. Participating in blockchain consensus mechanisms and maintaining a full copy of the ledger can be computationally intensive and energy-consuming, which may not be feasible for many IoT devices [9]. More importantly, the decentralised nature of blockchain can result in performance overhead and scalability issues, especially when dealing with high transaction volumes and real-time data processing requirements [10]. To fully realise the potential of blockchain IoT integration, it is crucial to thoroughly investigate the design choices, configuration trade-offs, and performance implications of different blockchain architectures and consensus algorithms in the context of IoT systems.

The use of simulation for the early evaluation of emerging technologies, such as Blockchain and IoT, is crucial for ensuring the planning, design and development of proactive solutions [11]. In particular, simulation can help identify points of strength and limitations, as well as analyse the requirements for achieving a desired performance target. Even in the post-production stage, simulation enables the reproduction of issues, the tracking of root causes, and the proposal of an optimised solution.

As opposed to testing in a real-world environment, simulation can help with all of the above with minimum risk, less cost possible, and mitigated ethical concerns, particularly for critical systems [12]. Furthermore, simulation can assist researchers and enthusiasts in experimenting with large-scale, heterogeneous, and complex systems such as Blockchain and IoT, which are difficult to access in many cases. In the context of blockchain-based IoT, simulation allows modelling and exploring various scenarios, testing different configurations, and assessing the performance and associated trade-offs of blockchain-based IoT applications. The simulation also provides insight into the behaviour and interactions of the IoT and Blockchain components under various conditions.

Despite the importance of simulation for blockchain-based IoT solutions, there is currently a lack of comprehensive and flexible simulation frameworks specifically designed for this domain [13]. Existing simulation tools often focus on either IoT or blockchain independently, lacking the necessary integration and extensibility to capture the unique characteristics and requirements of blockchain IoT systems [14]. Moreover, many simulation frameworks are limited in their ability to model diverse IoT scenarios, blockchain architectures, and consensus algorithms, hindering the exploration of novel solutions and comparative analysis [14, 11].

To address these limitations, this thesis aims to provide a simulation framework that covers various aspects of evaluating the performance of blockchain-based IoT systems. First, it systematically reviews Blockchain and IoT simulation as of today's practice. Second, it proposes a modular and extensible simulator for evaluating blockchain-based IoT systems. Third, it proposes a middleware architecture to integrate IoT simulators with real blockchain platforms. Finally, it incorporates machine learning (ML) techniques to predict overall performance and recommend optimal configuration parameters to achieve a desired performance target.

1.2 Research Aim, Questions and Objectives

Estimating the performance of a Blockchain-based IoT Ecosystem is a challenging task. While several studies have focused on simulating either Blockchain or IoT separately, no extensive study has considered integrating both technologies. This thesis aims to tackle the challenges associated with evaluating the performance of a Blockchain-based IoT Ecosystem by proposing and implementing various approaches. To achieve this, the thesis seeks to answer the following research questions:

1. ***RQ1***. *What techniques and configurations are used in current blockchain simulators?*

   Exploring the current state-of-the-art Blockchain simulation assists in understanding the subject and learning from existing knowledge, lessons, and experience. Given that there has not been a systematic review of Blockchain simulation, and to properly

answer this question, this thesis holds the objective of conducting a systematic review of Blockchain simulations, which is useful for

(a) Exploring and understanding the subject of blockchain simulation.

(b) Categorising existing simulators based on their modelling approaches, types of simulated blockchain platforms, and supported blockchain features.

(c) Identifying the capabilities and shortcomings of existing blockchain simulators in terms of performance evaluation.

(d) Analysing the usability of existing Blockchain simulators for IoT applications.

(e) Investigating areas where more research and development work is needed to advance the state-of-the-art.

2. ***RQ2***. *Given the lack of existing simulation frameworks for evaluating the performance of Blockchain-based IoT ecosystems, what is required to bridge the gap?*

   There has been a growing interest in utilising Blockchain for IoT applications. While there are simulators for each, the systematic review reveals that no existing framework considers both in a unified simulator for performance evaluation purposes. Therefore, this thesis holds the following objectives in this regard (see Figure 1.1 for a visual illustration):

   (a) Gathering key requirements that must be satisfied in the simulator dedicated to evaluating blockchain-based IoT ecosystems. These requirements are analysed to determine key features, capabilities, modelling approach, critical performance aspects, configuration options, parameter fine-tuning process, and evaluation output. The ultimate goal is a simulator that can capture the characteristics of a real platform that integrates an IoT ecosystem with Blockchain, mimics its behaviour, and produces reasonably identical performance metrics. In order to realise a proper simulation framework design and implementation, there are various sources considered for the requirements-gathering process, which are:

i. Examining the systematic review findings with a focus on relevant existing simulators in terms of their capabilities and shortcomings, shortcomings, and what lessons are learnt from their experience.

ii. Questioners to consider key requirements proposed by subject-matter experts and ensure quality coverage.

iii. Interviews with experts who express their desire to participate in further discussion and analysis of the requirements.

(b) Designing a simulation framework for evaluating the performance of blockchain-based IoT ecosystems. This step considers the outputs of the requirements gathering and analysis procedures.

(c) Validating the simulation framework design by holding a workshop that involves subject-matter experts. The workshop is intended to present the proposed simulation framework design to the select experts, consider their feedback on areas of strengths and weaknesses via questionnaires, and conduct a focus group for open discussions on the matter.

(d) Refining the simulation framework design to consider the outcomes of the validation process and any areas of improvement.

(e) Materialising a reference implementation for the simulation framework design.

(f) Validating and evaluating the reference implementation to ensure whether it meets the goals of framework design and can capture by conducting the following:

i. Carrying out a goal-oriented approach for measuring the software implementation quality, namely GQM, short for Goal, Question, Metric. It helps confirm whether the simulation framework implementation aligns with the intended design goals.

ii. Suggesting a use case scenario about a blockchain-based IoT ecosystem. The use case scenario is intended for simulator demonstration, evaluation, and validation.

iii. Implementing the use case scenario in real settings (A real blockchain platform and a simplified IoT system).

iv. Benchmarking the real implementation to obtain performance indicators to be utilised for comparison purposes.

v. Implementing the use case scenario using the proposed simulator to produce the same performance indicators.

vi. Conducting a statistical analysis to validate whether the simulator performance evaluation corresponds to their counterparts produced by the simulator.

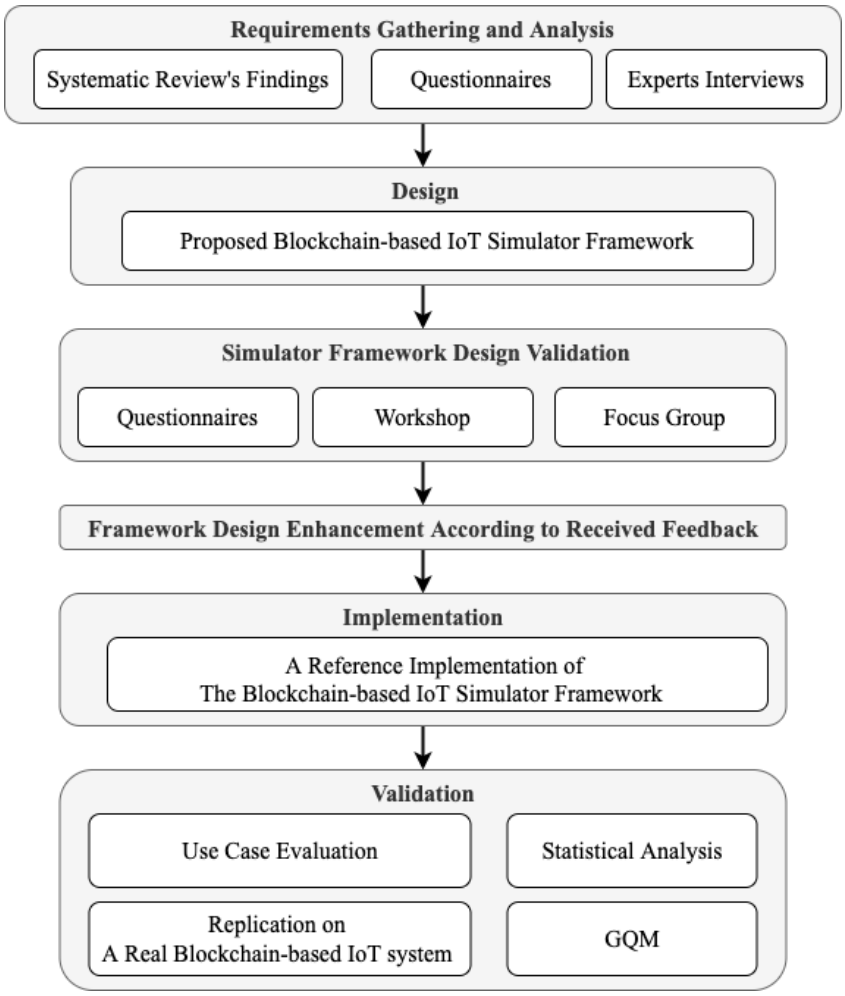


Fig. 1.1 The Research Methodology for Realising a Blockchain-based IoT Simulator

3. ***RQ3.*** *How feasible is it to utilise IoT simulators as workload generators to benchmark the performance of real blockchain platforms?*

   For performance evaluation purposes, not all blockchain features (i.e. smart contracts) can be perfectly simulated for every scenario. A set of real Blockchain platforms exists that are open source, accessible, and can be deployed to scalable cloud computing resources. However, large-scale IoT infrastructures are not easily accessible for research and development purposes. To answer this question, this thesis proposes a middleware architecture that addresses the gap between IoT simulators and real blockchain platforms for performance benchmarking purposes. The following objectives are set for this matter:

   (a) Design a middleware architecture that addresses the challenges associated with the connection and interaction between blockchain platforms and IoT simulators

   (b) Ensure interoperability and data synchronisation between the two distinctive environments.

   (c) Provide a reference implementation that demonstrates the middleware architecture.

   (d) Validate correct operations by utilising the implemented middleware for benchmarking an IoT simulator with a real blockchain platform.

4. ***RQ4****. How do machine learning techniques assist in predicting blockchain performance metrics and identifying optimal configuration parameters?*

   Blockchain infrastructure is heterogeneous and complex. Numerous factors influencing the overall blockchain performance must be considered before an organisation investigates blockchain viability before production. While simulation can help investigate the performance of blockchain-based IoT applications, there would be multiple trial and error processes until the right configuration parameters are determined for optimal blockchain performance. To answer this question, this thesis holds the following objectives:

(a) Investigate proper machine learning techniques for prediction and recommendations.

(b) Design and implement ML models to predicate the overall blockchain performance (i.e., throughput, latency, etc.) based on predefined configuration parameters.

(c) Design and implement ML models to recommend optimal configuration parameters to achieve desired blockchain performance.

Considering the thesis aim, research questions, and associated objectives, the thesis has proposed and implemented various approaches, as outlined in Section 1.3. These approaches include developing a novel simulation framework that integrates both Blockchain and IoT technologies (discussed in Chapter 5), designing a middleware architecture to enhance benchmarking capabilities (discussed in Chapter 6) and utilizing machine learning models to predict performance metrics and optimize configurations (discussed in Chapter 7). The findings from these efforts demonstrate that the proposed approaches have successfully achieved the task of estimating and evaluating the performance of a Blockchain-based IoT Ecosystem. Consequently, these contributions have successfully achieved the research aim and addressed the research questions and objectives, as summarised in Chapter 8.

1.3 Primary Contributions

The primary contributions of this research are as follows:

1. The first primary contribution is a systematic mapping study on blockchain simulation. It addresses the first research question (RQ1) and is included in Chapter 3. This contribution is published in a research work[13] that extensively and systematically surveys the domain of Blockchain simulations. It critically classifies existing blockchain simulation studies based on their features, capabilities, and limitations. Moreover, it delves into the implementation of existing open-source blockchain simulators to analyse their supported configuration parameters and performance evaluation metrics. It critically

compares their code quality, flexibility, reliability, and maintainability. Furthermore, it investigates whether and how these simulators are scientifically validated.

2. The second primary contribution is a simulation framework for evaluating the performance of blockchain-based IoT ecosystems. It addresses the second research question (RQ2). It is covered in two Chapters 4 and 5. The contribution of the simulation framework is the first in the literature to combine IoT and Blockchain in a unified simulation tool for performance evaluation purposes. The simulation framework undergoes a rigorous evaluation and validation procedure, including requirements gathering and analysis, conceptual design, implementation and validation, as visualised in Figure 1.1. Two published scientific papers [14, 15] is associated with this contribution with regard to requirements gathering and analysis, as well as design conceptualisation and validation. A third paper about the implementation and experiments has also been prepared and submitted to a journal.

3. The third primary contribution is a middleware architecture that enables utilising IoT simulator for benchmarking real blockchain platforms. It addresses the third research question (RQ3) and is included in Chapter 6. It resolves major challenges associated with the distinction between simulated execution environments (IoT simulators) and real execution environments (Blockchain platforms). Therefore, it enables connectivity and communication between both ends to enable utilising IoT simulators (i.e. IoTsim-Osmosis [16]) as workload generators to benchmark the performance of a real blockchain platform (i.e. Hyperledger Fabric [17]. A scientific paper has been prepared and submitted to a journal that elaborates on the design of middleware architecture, its implementation, and the validation process.

4. The fourth primary contribution is designing and implementing machine learning models trained by incorporating large data sets generated from the proposed blockchain-based IoT simulator. It addresses the fourth research question (RQ4) and is included in Chapter 7. Two purposes are considered in this regard as follows. First, employing the k-nearest neighbours (*k*NN) method for predicting the overall blockchain performance

based on various elements such as the number of nodes, miners, and transactions. Second, an enhanced version of the Salp Optimization (ISO) algorithm incorporating rough set theory is introduced to handle performance uncertainty and identify optimal configuration parameters for achieving a target blockchain performance. This contribution has been published at a scientific conference [18].

1.4 List of Publications

The research findings from chapters 3, 4, 5 and 7 have been published in international conferences and journals. The primary researcher did the majority of the work, including the idea, running the experiments, and writing the paper. The co-authors contributed significantly through discussions, paper revisions, editing, and comments.

Conference Papers

1. Albshri, A., Alzubaidi, A., Awaji, B., and Solaiman, E. (2022). **Blockchain Simulators: A Systematic Mapping Study**. In 2022 IEEE International Conference on Services Computing **(SCC)** (pp. 284-294). (This work covers chapter 3).

   Authors Contributions: Designing and developing the framework for the systematic mapping study on blockchain simulators was carried out by A. Albshri. A. Albshri led the data collection and analysis. A. Albshri designed and executed the literature review and data extraction; A. Albshri analyzed the data. A. Albshri drafted the manuscript, then reviewed and edited it by A. Albshri, E. Solaiman, A. Alzubaidi, and B. Awaji; E. Solaiman is the lead supervisor of the project.

2. Albshri, A., Awaji, B., and Solaiman, E. (2022). **Investigating the Requirement of Building Blockchain Simulator for IoT Applications**. In 2022 IEEE International Conference on Smart Internet of Things **(SmartIoT)** (pp. 232-240). (This work covers chapter 4).

Authors Contributions: Designing and collecting data to investigate the requirements for building a blockchain simulator for IoT Applications was carried out by A. Albshri; A. Albshri developed the methodology; A. Albshri designed and executed the data collection and analysis. A. Albshri drafted the manuscript, then reviewed and edited it by A. Albshri, E. Solaiman, and B. Awaji; E. Solaiman is the project's lead supervisor.

3. Albshri, A., Alzubaidi, A., and Solaiman, E. (2022). **A Model-Based Machine Learning Approach for Assessing the Performance of Blockchain Applications**. In 2023 IEEE International Conference on Smart Internet of Things **(SmartIoT)** (pp. 46-55). (This work covers chapter 7).

   Authors Contributions: A. Albshri designed and developed the machine learning models for assessing blockchain performance; A. Albshri developed the methodology; A. Albshri led the data collection. A. Albshri implemented the ML models; A. Albshri analyzed and evaluated the results. A. Albshri drafted the manuscript, then reviewed and edited it by A. Albshri, E. Solaiman, and A. Alzubaidi; E. Solaiman is the project's lead supervisor.

Journal Papers

1. Albshri, A., Alzubaidi, A., Alharby, M., Awaji, B., Mitra, K., & Solaiman, E. (2023). **A conceptual architecture for simulating blockchain-based IoT ecosystems**. Journal of Cloud Computing, 12(1), 1-26. (This work covers chapter 4 and 5).

   Authors Contributions: Designing and developing the conceptual architecture for simulating blockchain-based IoT ecosystems was carried out by A. Albshri; A. Albshri and A. Alzubaidi developed the methodology; A. Albshri evaluated the conceptual architecture. A. Albshri drafted the manuscript, then reviewed and edited by A. Albshri, E. Solaiman, A. Alzubaidi, M. Alharby, K. Mitra, and B. Awaji; E. Solaiman is the project's lead supervisor.

Under Review

1. The paper entitled ***"SimBlockLink: A Middleware for Linking IoT Simulations with Real Blockchain Platforms for Enhanced Performance Evaluation"***, by Alzubaidi, A., Albshri, A., Mitra, K., Ranjan, R., & Solaiman, E., has been submitted to the Journal of Blockchain: Research and Applications. It is relevant to the chapter 6.

   Authors Contributions: Designing and developing the SimBlockLink middleware architecture for linking IoT simulations with real blockchain platforms was carried out by A. Alzubaidi and A. Albshri; A. Albshri developed the methodology; A. Albshri developed the conceptual framework and middleware implementation; A. Albshri evaluated the SimBlockLink middleware architecture; A. Albshri drafted the manuscript; then, reviewed and edited by A. Albshri, A. Alzubaidi, E. Solaiman and K. Mitra; E. Solaiman is the project's lead supervisor.

   My Contribution Scope: The collaborative work in Chapter 6 focuses on improving the design and implementation of an existing tool previously developed by Alzubaidi. The improved version of this tool has been specifically adapted to serve the purpose of my thesis, which focuses on blockchain performance benchmarking using IoT simulators. In contrast, the previous version of the tool was primarily focused on a different purpose: examining SLA compliance in a multi-cloud setting within the context of IoT. While this chapter required a different design, implementation, and results analysis, the tool has been significantly improved, particularly in terms of the transaction submission mechanism and compatibility with recent blockchain advancements. Consequently, the tool was redesigned, reimplemented, and validated to serve the purpose of my work.

Collaborative Research

In addition to the publications mentioned above, I have co-authored several research papers in the Blockchain field. However, these papers have not directly contributed to this thesis. The following is a list of those papers:

1. Awaji, B., Solaiman, E., & Albshri, A. (2020). **Blockchain-based applications in higher education: A systematic mapping study**. In Proceedings of the 5th International Conference on Information and Education Innovations (pp. 96-104).

   Co-Author Role: A. Albshri's role in this paper involved collecting data from various sources. Additionally, A. Albshri collaborated with B. Awaj to analyze and classify the papers based on pre-specified factors and contributed to the manuscript's review and editing.

2. Awaji, B., Solaiman, E., & Albshri, A. (2022). **Development and Evaluation of a Trusted Achievement Record of Accomplishments for Students in Higher Education Using Blockchain**. In International Conference on Computer Supported Education (pp. 100-124).

   Co-Author Role: A. Albshri's contributions to this paper included designing the methodology in collaboration with B. Awaj, as well as reviewing and editing the manuscript.

1.5 Thesis Structure

This thesis primarily consists of the contributions outlined in Section 1.3. Subsequent sections provide an overview of the thesis structure and associate each contribution with its corresponding chapter.

- **Chapter 2** provides an overview of blockchain technology, its integration with the Internet of Things (IoT), and evaluation methods for blockchain-based IoT applications.
- **Chapter 3** provides a comparative analysis of blockchain simulators, focusing on their features, code quality, and the validation of metrics.
- **Chapter 4** analyzes expert opinions on the necessity and design of a simulation framework for blockchain-based IoT ecosystems, incorporating two studies and interviews to understand associated opportunities and challenges.

- **Chapter 5** proposes an architecture for simulating blockchain-based IoT systems, including design, implementation, and performance evaluation through statistical analyses.
- **Chapter 6** develops a middleware to connect IoT simulators with blockchain platforms, detailing its design, implementation, and benchmarking capabilities for evaluating blockchain performance under simulated IoT workloads.
- **Chapter 7** introduces machine learning models for predicting blockchain performance metrics and optimizing blockchain configurations.
- **Chapter 8** summarizes the thesis's contributions and limitations and suggests directions for future research.
- **Appendix A** provides the technical details for the reference implementation of the Blockchain-based IoT simulation framework proposed in Chapter 5.
- **Appendix B** illustrates the user manual for the implemented simulator.
- **Appendix C** provides the technical details for conducting the comparative analysis between the implemented simulator and a real blockchain platform to examine the accuracy of the former's performance evaluation in terms of transactions throughput and latency.

Chapter 2

Background

Summary

This chapter provides a background for the thesis, covering the main aspects of it. First, it provides an overview of blockchain technology and highlights its characteristics, such as types of blockchain networks, consensus protocols, smart contracts, and transaction execution 2.1. Next, it provides an overview of the Internet of Things (IoT) concept and its architecture 2.2. The chapter then sheds light on a set of challenges associated with IoT 2.3. The following section delves into the motivations behind integrating these two technologies, highlighting the benefits and challenges associated with their convergence. It further classifies the motivations based on system functionality by distinguishing between essential and non-essential features. Finally, it explores four key evaluation approaches for blockchain-based IoT systems 2.4: Benchmarking, Simulation, Middleware, and Model-based machine learning.

2.1 Blockchain Technology

This section overviews the fundamentals of blockchain technology, its key characteristics, and potential applications in various industries.

2.1.1 An Overview of Blockchain Technology

In centralised systems, a single authority usually processes transactions between parties. With this model, various challenges must be dealt with, including security vulnerabilities (e.g., unauthorised access), reliability concerns (e.g., single point of failure), trust levels, and misconduct acts. Blockchain technology was introduced with the promise of overcoming these challenges by allowing untrusted parties to interact in a distributed manner without the need for a third party. Unlike centralised systems, a blockchain system is essentially a decentralised system that processes transactions by an interconnected network of independent computing nodes. Every processed transaction is immutably recorded within blocks on a replicated ledger over participating nodes.

The participating nodes are responsible for maintaining the validity of the commonly shared ledger by collecting transactions from a pool of pending transactions, validating the picked transactions, and appending them to new blocks. Each validating node proposes its block of transactions by propagating it to other peers for consensus on the validity of the block's structure and content. Ultimately, a successfully validated block will end up appended to the ledger of each participating node.

Blockchain derives its name from its ledger structure, which organises its transactions into a chain of blocks, as depicted in Figure 2.1. Each block is uniquely identified by a cryptographic hash and linked to its predecessor, forming a series of interlinked blocks. Once a block is generated and appended to the blockchain ledger, misbehaving nodes find it difficult to alter transactions without rewriting all proceeding blocks. This property makes the blockchain system resistant to tampering and the double-spending problem [19]. The unique blockchain data structure promotes transparency and traceability while ensuring high availability through data replication across participating nodes.

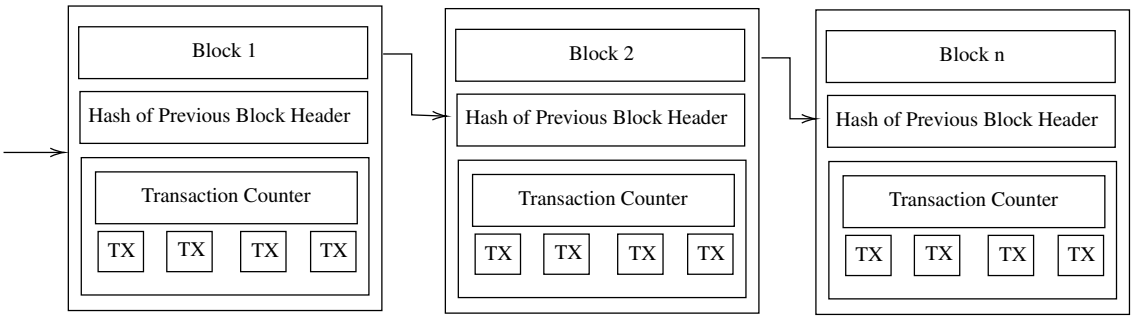


Fig. 2.1 A chain of blocks

Bitcoin [20] is the first and most well-known Blockchain example, which solely serves the purpose of being a decentralised cash system over a peer-to-peer (P2P) network. Bitcoin lacks the openness and programming capabilities needed to develop further decentralised applications beyond financial use cases. Consequently, several blockchains emerged to remedy this shortcoming by offering agnostic blockchain infrastructure and smart contracts. Hyperledger Fabric [17], Ethereum [21], and Quorum [22] are common examples of Blockchain platforms for Decentralised Applications (DApp). The concept of smart contracts plays an essential role in these blockchain platforms, which is a computer programme that can be incorporated into the blockchain to provide unified instructions to the participating nodes about processing and recording transactions and their payload data. Consequently, it has been possible to seamlessly integrate blockchain with various technologies such as the Internet of Things (IoT) and Cloud computing to serve a multitude of decentralised applications in several domains such as education, healthcare, supply chain, manufacturing, and others.

2.1.2 Blockchain Characteristics

2.1.2.1 Decentralisation and Consensus Mechanisms

Blockchain is generally considered a decentralised peer-to-peer (P2P) network system and infrastructure. Contrary to centralised systems, no single author can manage the ledger, process transactions, and decide their validity and finality [23]. Therefore, Blockchain, being a decentralised system, usually employs consensus mechanisms to compensate for the lack of a single authority to help validating nodes reach a common decision on transactions'

validity and finality [24]. Given the multitude of validating nodes, consensus mechanisms are important to guarantee data integrity and consistency across the blockchain network [23].

Several studies have explored various consensus protocols utilized in different blockchain networks, including those discussed by [25] and [26]. For instance, Bitcoin employs Proof-of-Work (PoW) as its consensus mechanism. Although Ethereum used to adopt the same consensus protocol, it has recently transitioned towards a more lightweight protocol known as Proof-of-Stake (PoS). Furthermore, various Ethereum-based independent networks have adopted different consensus protocols, such as Proof of Authority (PoA) and Proof of Elapsed Time (PoET). Hyperledger Fabric, which operates as a consortium and a permissioned network, follows a modular approach to consensus mechanisms, allowing the adoption of any protocol in theory and by design [17]. Currently, the recommended and supported consensus protocol for Hyperledger Fabric is Raft.

2.1.2.2 Types of Blockchain Networks

While most blockchain platforms emphasise the concept of decentralisation, they can be categorised into three main types of network openness: public, consortium and private [27]. Each type has its own unique characteristics, advantages, and disadvantages, as illustrated in Table 2.1.

1. ***Public Blockchains***: also interchangeably referred to as permissionless blockchains, which enables any node to join the network, view the ledger and participate in transactions processing and block validation. Public blockchain networks are characterised by a high degree of decentralisation and transparency, where all users possess the same privilege levels. Several open blockchain platforms, such as Bitcoin [20] and Ethereum [21], encourage validating nodes to participate in the network by providing incentives in the form of cryptocurrencies or tokens. Public blockchains are particularly distinguished by their high degree of inclusiveness and transparency. They also exhibit outstanding resistance to single points of failure, partially due to the extensive participant base. Nonetheless, they come with drawbacks such as privacy concerns, limited transaction processing capacity, extended delays, and substantial energy usage.

These disadvantages are mainly due to the large number of validating nodes and reliance on heavyweight consensus protocols (e.g., Proof-of-Work). Given their open nature, public blockchain networks follow an economic model in which clients hold the responsibility for covering transaction costs, which can be seen in many cases as a disadvantage.

2. ***Consortium Blockchains***: also interchangeably referred to as permissioned blockchains, only enables a predetermined set of authorised and authenticated nodes to participate in transaction processing and block validations. Typically, these nodes are under the control of organisations that are part of a consortium. Positioned as a blend between public and private blockchains, consortium blockchains offer a level of decentralisation while retaining control and privacy features, as exemplified by Hyperledger Fabric [17]. The key benefits of consortium blockchains include increased transaction processing capacity, reduced latency, and improved privacy compared to public blockchains. These advantages are mainly due to the limited number of participating nodes and reliance on lightweight consensus mechanisms (e.g. RAFT, a crash-fault-tolerant protocol). Moreover, consortium Blockchains present a structured governance framework, allowing members to establish and adhere to enforced rules and guidelines. Unlike public blockchain networks, participating organisations are responsible for covering the cost of running and operating the underlying infrastructure and transaction processing.

3. ***Private blockchains***: Also referred to as permissioned blockchains, private blockchains are under the full control of a single organisation. Private blockchains are the least network in terms of decentralisation. It can be any public or consortium blockchain platform, but with the exception that participation is limited to nodes under a single authority. This limited form of decentralisation could enable higher transaction processing capacity, minimal delay, and the ability to exercise complete authority over the network and its members. Furthermore, private blockchains can be useful for achieving a higher confidentiality level by enabling ledger access restrictions. Nevertheless,

private blockchains can be seen as redundant since node participation is limited to a single authority.

The selection among public, consortium, and private blockchains depends on their suitability to various use cases and requirements. Public blockchains are well-suited for applications that emphasise decentralisation, transparency, and security, such as cryptocurrencies and decentralised finance (DeFi) applications. Private blockchains are most suitable for applications that prioritise efficiency, confidentiality, and governance, such as internal record-keeping and asset tracking within a single entity. Consortium blockchains are optimal for applications that necessitate a trade-off between decentralisation, efficiency, and confidentiality, such as supply chain management and cross-border payments.

Table 2.1 Comparison of the key features of public, consortium, and private blockchains.

Feature	Public Blockchain	Consortium Blockchain	Private Blockchain
Access	Permissionless	Permissioned	Permissioned
Decentralisation	High	Medium	Low
Transparency	High	Medium	Low
Performance	Low	Medium	High
Privacy	Low	Medium	High
Governance	Decentralised	Partially centralised	Centralised
Examples	Bitcoin, Ethereum	Hyperledger Fabric	Any Blockchain platform

2.1.2.3 Smart Contracts

Following the introduction of Bitcoin, several generic blockchain platforms were developed to utilise blockchain features beyond financial use cases [28]. These generic blockchain platforms mainly rely on the concept of smart contracts to express the logic of transaction execution. The concept of smart contracts is not a new concept. Szabo (1994) [29] first introduced the concept, defining it as "*a computerised transaction protocol that executes the terms and conditions of a contractual agreement*", thereby enabling self-enforcement without the need for trusted intermediaries.

With the advent of blockchain technology, this concept has materialised to existence as an electronic program stored on the chain, intended to act as a trusted intermediary code of

conduct between participants, even in the light of no mutual trust. To provide a practical definition, smart contacts, in the context of blockchain, are programmed instructions that describe how validating nodes must execute transactions. Ethereum and Hyperledger Fabric, for instance, enable the deployment and execution of smart contracts. The former introduces a new programming language called Solidity for developing smart contracts, while the latter provides smart contract development frameworks based on existing programming languages such as JavaScript, Golang, and Java.

It is worth noting that once the smart contract is deployed, it is difficult to upgrade the smart contract. Different Blockchain platforms have different degrees of immutability in smart contracts, ranging from full immutability, as would be the case with Ethereum, to partial immutability, as would be the case with Hyperledger Fabric. This feature is important for countering possible cyberattacks or misconduct. The latter allows smart contract upgradability under rigorous governance procedures for approving a new smart contract version. In any case, and as Figure 2.2 illustrates, each validating node deploys a replica of the smart contract to execute a block of pending transactions. Provided that the outcome of the transaction execution is validated and agreed upon, the outcomes are immutability recorded on the ledger.

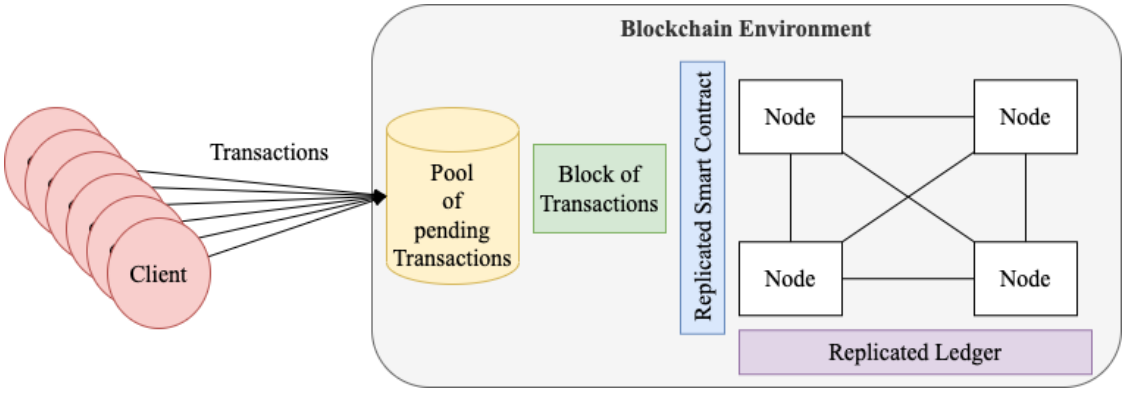


Fig. 2.2 Validating nodes execute pending transactions via their replica of a smart contract and record the outcome into their copy of the ledger

2.2 Internet of Things (IoT)

This section provides an overview of the concept of the Internet of Things (IoT), as well as its architecture and components.

2.2.1 An Overview of IoT

The Internet of Things (IoT) concept is established around the idea of enabling virtually anything to connect to the Internet and actively communicate meaningful information [30]. It facilitates the creation of intelligent systems by processing and analysing vast amounts of data collected from sensors and devices and actively responding to changing factors. IoT promises the creation of unprecedented business value with less human intervention, improved productivity, and effective decision-making [31]. Technically, IoT is a heterogeneous network of interconnected devices, objects, and gadgets that are usually featured with capabilities such as sensing, actuation, computing resources, connectivity, and communication [32]. One can think of a wide range of applications that can benefit from the IoT concepts, including smart homes [33], smart cities [34], Telemedicine [35], and smart grids [36].

2.2.1.1 IoT Architecture

Several proposed IoT architectures exist in the literature, such as Al-Fuqaha et al.[37] and Bouakouk et al.[38]. This thesis conveniently adopts the architecture proposed by Xu et al.[39]. which is composed of four layers, mainly perception, networking, service and application, as shown in Figure 2.3.

1. Perception Layer: The perception layer is directly associated with the physical world. In this layer, sensors are used to collect environmental information, while actuators are used to respond to received data or changing factors [40]. The perception layer accommodates the following:

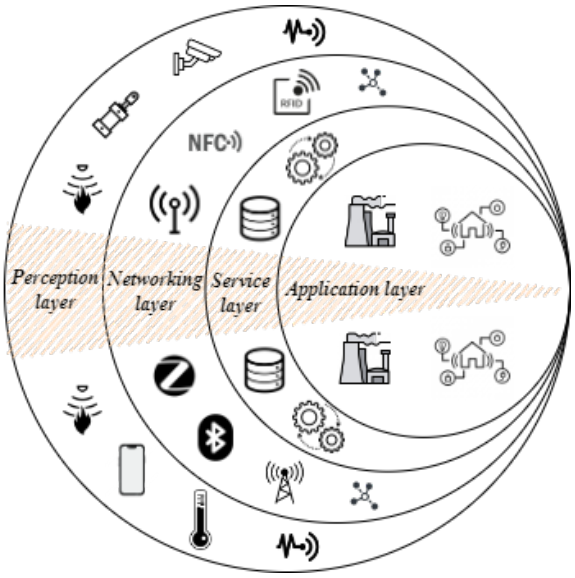


Fig. 2.3 IoT architecture

(a) Any things attached to industrial controllers such as Programmable Logic Controllers (PLC), Remote Terminal Units (RTU), Intelligent Electronic Devices (IED), or even commercial controllers such as Raspberry Pi and Arduino.

(b) Sensors collect data from their surroundings, such as temperature, humidity, pressure, smoke, fire, position, posture, etc.

(c) Actuators that can respond to events and take actions [40] whether they are electronic (e.g., relays), pneumatic (e.g., rod-style cylindrical) or hydraulic (e.g., hydraulic rotating valves).

2. Networking Layer: The network layer is responsible for connecting things to the Internet to enable the transmission of their collected data from the perception layer to their destination. There are various networks, connectivity, communication protocols, and technologies involved. Some of them are specifically suited for lightweight IoT devices, while others are well-established and not only used for IoT purposes but can be found elsewhere. For instance, Well-established connectivity protocols and technologies in the network layer can involve Wireless networks (e.g. Wi-Fi and Bluetooth), Wired networks (e.g. Ethernet), and Cellular networks (e.g. 3G, 4G or 5G). The Hypertext Transfer Protocol (HTTP) is a well-established communication and messaging proto-

col that is still in use for IoT purposes. On the other hand, lightweight connectivity protocols and technologies that are well-suited for constrained-resources IoT devices include, but are not limited to, Low-Power Wide-Area Networks NB-IoT, LoRaWAN and Sigfox, Radio Frequency Identification (RFID), and Near Field Communication (NFC) [41]. Regarding lightweight communication and messaging protocols, Message Queuing Telemetry Transport (MQTT) and Constrained Application Protocol (CoAP) are some examples that can suit IoT devices in most cases [37].

3. Service Layer: The service layer is responsible for processing and managing the data collected from the perception layer and transmitted through the network layer. This layer handles various tasks, such as things management, data storage, data analysis, and decision-making. For instance, devices collected data sets from their attached sensor. Given proper authentication and authorisation from the device management component at the service layer, devices can send data for long-term storage, which can then be analysed to extract insights and make informed decisions [40]. Moreover, the service layer is also concerned with business logic control, configuration, updates, monitoring, and security [40].

4. Application layer: This layer represents the generated value from connected things to the internet. The generated value can be in the form of an emerging application or an enhanced experience. Applications can be divided into two categories, which are critical applications (e.g. industrial) of life quality applications (e.g. smart home). Either way, the application layer interfaces with end users and other machines for interaction and communication with the underlying IoT ecosystem. For instance, it can provide a user-friendly interface, whether desktop, web, or mobile Graphical User Interface (GUI), allowing users to control IoT devices and access visualised data remotely. Dashboards and visualisation tools can provide interactive and intuitive representations of data, which can be particularly beneficial for monitoring and alerting purposes [42]. Application Programming Interfaces (APIs) and Software Development Kits (SDKs) are also related to the application layer since they enable other machines

and applications to integrate and extend the underlying IoT system and communicate with it. Several domains, such as healthcare, smart cities, education, agriculture, and others, can take advantage of the underlying IoT system to innovate an unprecedented application or enhance traditional systems by introducing an enhanced experience [43].

2.3 Integration of Blockchain with IoT

This section highlights areas where the integration of blockchain and IoT can be appealing and explains the benefits and challenges associated with their convergence. It also differentiates between motivations that are essential for system functionality and those that are beneficial but not strictly necessary.

2.3.1 Motivation for Integrating Blockchain with IoT

Several challenges associated with typical IoT systems call for revisiting current models and incorporating revolutionary and effective technologies and methods. When considering the features of Blockchain technologies, they present an appealing opportunity to utilise Blockchain technology to overcome challenges of IoT or simply for creativity and innovation grounds [44]. For example, the following are some of the IoT challenges that can possibly be addressed by blockchain features:

1. **Reduced Latency**: In a centralised system, data sets are usually persisted in a central data storage. Hence, undesirable delays can occur when sending collected data from geographically dispersed devices to a central hub or requesting data [45]. Distributed nodes can process and store data locally, which may help in reducing latency and enabling faster response for real-time applications such as traffic management or smart grids [45]. Fog Computing, which is an IoT extension, is distributed by nature and can directly benefit from the distributed nature of Blockchain technology [46]. With Blockchain, Fog nodes can benefit from the local ledger to mitigate the need for mobile IoT devices for constant communication with the distant cloud, minimizing delays caused by network congestion or geographical distance [46].

2. **High Availability**: It is a vital aspect for critical IoT systems, such as manufacturing and smart grids, to remain functional whenever needed [32]. Any downtime or interruption in the availability of any part (e.g. sensors, devices, networks, servers, and applications) of such heterogeneous and complex IoT systems can hinder effective operations and lead to significant life and financial losses. Most traditional IoT systems are heavily dependent on centralised servers, in which a failure at the latter causes the entire ecosystem to degrade or, worse, completely get out of service [47]. This case is extremely undesirable in critical systems that require high reliability and dependability.

   The adoption of a highly available system approach (i.e., redundancy) can positively enhance the ability of IoT systems to operate continuously. One of the essential advantages of blockchain is its ability to mitigate the risk of a single failure point, which sounds appealing for resolving such issues in IoT systems [48]. In Blockchain, all validating nodes can continue to validate transactions and record their outcomes, even if some nodes are out of service. Once absent nodes are available, they can synchronise their ledger and carry on their operations from that point [49].

3. **Trust and Accountability**: A typical IoT application involves multiple stakeholders and various participants to deliver its aims and goals [50]. For instance, an IoT-based shipment tracking system can involve suppliers, various international customs, cargo companies, warehouses, delivery companies, etc. As to which party to trust for transaction validation and data integrity, it can be problematic. Independent third-party entities may sound appealing, but there are several issues associated with them, such as cost, single point of failure, and potential misconduct [51]. Blockchain removes the need for a single authority or a third party. It offers a high level of trust because the majority of participants must come to a consensus on transaction validity based on preset rules and procedures [51]. Transactions transparency, traceability, and immutability features can provide an out-of-the-box accountability tool, which prevents the dilemma of finger-pointing where each involved party refuses to be responsible for a failure, given the lack of hard evidence.

4. **Security and Resilience**: IoT applications rely heavily on generated data for decision-making and delivering the intended business value. Hence, it is important to maintain the integrity of the data, whether it is in transit or rest, as well as the security of the overall IoT system from harmful actions [32]. Each IoT node can present a potential failure point when exposed to cyberattacks such as distributed denial of service (DDoS) attacks, injection attacks, and other security threats [52]. Additionally, the reliance on centralised systems, such as a single cloud provider or Supervisory Control and Data Acquisition (SCADA) systems, can introduce a single point of failure when being vulnerable to attacks [53]. In such an event, the overall IoT system can negatively diverge from the intended behaviour.

   Blockchain is infamous for its security-by-design feature, employing well-established cryptography algorithms and hashing techniques wherever possible at the journey of transactions [54]. To begin with, the Zero-trust architecture is adopted in the vast majority of blockchain platforms [55]. A public key infrastructure (PKI) is also essential for sending, processing, and persisting transactions in these blockchain platforms. The data organisation into a chain of hashed blocks and transactions consolidates immutability, making it difficult for any blockchain node to tamper with the data in their local ledgers maliciously. Even if a node manages to execute malicious actions or is compromised for any reason, several measures are in place to tackle the issue, such as the distribution nature of the blockchain network, the mitigation of points of a single failure, replication and consensus mechanism. These are some of the reasons why blockchain can be considered for securing IoT networks and preserving data integrity.

2.3.2 Challenges Associated with the Convergence between Blockchain and IoT

Despite the opportunities of integrating Blockchain with IoT, several challenges and points of consideration must be accounted for when designing a blockchain-based IoT solution, which

arises from the complexities of both technologies and the difficulties in implementation. The following are examples of which:

1. **Resource Constraints**: Many IoT devices are often battery-powered and have limited energy resources that make energy efficiency a critical factor in the design of blockchain-based IoT architectures [10]. On the other hand, several blockchain platforms employ heavyweight consensus protocols (i.e. Proof of Work (PoW)) and security measures (i.e. PKI and Zero Knowledge Proof (ZKP)), which require high computational resources to cope with [56]. Furthermore, several IoT devices are limited in their capabilities, such as processing, storage, connectivity and communication protocols [10]. Consider a scenario where IoT devices must continuously operate within the blockchain network, which calls for the need to address associated challenges given the lack of sufficient IoT capabilities.

2. **Scalability**: Many IoT applications can generate a massive amount of datasets from an enormous number of IoT devices, such as in healthcare and power grid. Unlike centralised systems, scalability is a major challenge in integrating blockchain with IoT, given the level of distribution and decentralisation of most blockchain platforms. This is particularly imminent with blockchain platforms that depend on heavyweight consensus mechanisms (e.g., PoW), which severely impacts the overall performance and limits transaction throughput [10]. This fact defeats the purpose of utilising blockchain to reduce latency in the overall system, even if it succeeds in reducing latency between IoT devices and local blockchain nodes.

3. **Integration Complexity**: Both IoT and Blockchain are complex and heterogeneous, each with its own set of protocols, standards, and architectures. This complexity presents several challenges that can hinder research and development (R&D) efforts in experimenting with blockchain-based IoT solutions [14]. While Blockchain platforms are mostly accessible thanks to their open-source nature and scalable cloud services, a significant obstacle lies in accessing sufficiently actual IoT infrastructure for experimental purposes. Implementing a real-world IoT system to assess the viability

and effectiveness of a blockchain-driven approach can be laborious and demanding in terms of resources, necessitating the setup of the tangible quantity of equipment, acceptable infrastructure scale, and the combination of different elements and technologies [57]. Additionally, organisations may encounter challenges related to funding, understaffing, or technical knowledge to conduct practical testing and assessment of Blockchain-based IoT solutions [15] before the production stage.

2.3.3 Essential and Non-essential Motivations for Integrating Blockchain with IoT

When considering the integration of blockchain with IoT, it is important to differentiate between motivations (e.g. features) that are essential for system functionality and those that are beneficial but not strictly necessary. This section highlights and distinguishes between the motivations discussed in Sections 2.3.1 and 2.3.2.

Essential Motivations

These motivations address critical challenges and are often required to ensure the effective operation of IoT systems.

1. **High Availability**: Critical IoT systems like manufacturing and smart grids need continuous operation. Blockchain's decentralization prevents single points of failure, ensuring system functionality despite node failures.

2. **Security and Resilience**: IoT systems require data integrity and protection from cyberattacks. Blockchain's cryptography and hashing ensure data immutability and system reliability, safeguarding IoT networks.

3. **Trust and Accountability**: Blockchain's consensus mechanisms provide a transparent, traceable, and immutable transaction record, essential for multi-stakeholder IoT applications like supply chain management.

4. **Scalability**: IoT applications generate vast data, posing scalability challenges. Blockchain can manage this, but heavy consensus mechanisms like PoW can impact performance.

Non-essential Motivations

These motivations offer substantial benefits and opportunities for enhancement but are not always critical for the core functionality of IoT systems.

1. **Reduced Latency**: Reducing latency enhances real-time performance but isn't always critical. Blockchain can improve IoT performance by processing data locally, reducing delays from network congestion or distance.

2. **Resource Constraints**: IoT devices often have limited power and processing. Addressing these constraints is important but not always critical. Blockchain needs significant resources, but lightweight consensus or task offloading addresses blockchain's high computational needs.

2.4 Performance Evaluation Approaches

Evaluating the performance of Blockchain-based IoT systems is essential for assessing their viability and effectiveness. Performance evaluation may take place in the early stages or after the production stage. Given the complexity of both such systems and the difficulty of accessing a large and sufficient IoT infrastructure, this thesis is more focused on the early stages of performance evaluation. The performance evaluation must account for the unique characteristics and requirements of each of the underlying technologies and the convergence between them [58]. The distributed nature of blockchain networks, combined with the heterogeneity and resource constraints of IoT devices, necessitates the design of experiments that can capture the complex interactions and performance trade-offs between the various components. This involves the selection of appropriate metrics, such as transaction throughput, latency, and energy consumption, as well as the design of realistic workloads and network properties that can practically reflect real-world deployments. The remainder of this section provides an overview of key performance evaluation approaches employed in this thesis, which are Benchmarking 2.4.1, Simulation 2.4.2, Middleware 2.4.3, and Model-based machine learning 2.4.4.

2.4.1 Benchmarking Approaches

Benchmarking is a methodical process that involves conducting standardised and reproducible evaluations of blockchain-IoT systems, comparing their performance against established baselines and other state-of-the-art solutions [59, 60]. The benchmarking approach can enable a more objective and comparable assessment of different blockchain-IoT implementations, allowing researchers to identify strengths, weaknesses, and areas for improvement. The output of a benchmarking process can help improve the system design by identifying performance bottlenecks [61].

Key properties and requirements must be met to ensure effective benchmarking, including repeatability, comparability, and relevance [62]. Benchmarks must be developed using a rigorous scientific process based on well-established and acknowledged metrics that are practical and widely applicable to research and real-world contexts [63]. For instance, the Hyperledger Performance and Scale Working Group [64] developed an evaluation framework for assessing key metrics of blockchain performance, such as throughput, latency, rates of success/fail rates, and others. IoT-wise, there have been several studies that propose a performance evaluation framework for typical IoT architecture layers, such as Cloudrank-d [65] for cloud performance and EdgeBench [66] for edge layer performance [67]. Moreover, the IEEE Standardisation Association introduced a framework for the convergence between Blockchain and IoT [68].

Benchmarks can be categorised into two types: microbenchmarks and macrobenchmarks [69]. Microbenchmarks focus on evaluating small, specific parts of a software system, while macrobenchmarks assess the performance of larger, complex systems, often simulating real-world scenarios. Others may also classify benchmarks based on workload characteristics or the types of metrics used, such as latency, throughput, and utilisation [61]. Several tools have been developed for benchmarking blockchain technologies such as Blockbench [70], and Hyperledger Caliper [71]. The former is a benchmarking framework for private blockchains that evaluates key performance metrics across four layers of blockchain abstraction: consensus, data model, execution engine, and application. The latter is an evaluation framework

designed to stress the performance of several blockchain platforms, including Hyperledger Fabric, Ethereum, and others.

Several studies have focused on evaluating the performance of blockchain technologies in different contexts. Rouhani and Deters [72] examined Ethereum's efficiency on a private blockchain, comparing Ethereum clients Geth (PoW-based) and Parity (PoA-based). Baliga et al. [73] carried out an experimental study on Hyperledger Fabric (HLF) version 1.0, using Caliper to benchmark transactional parameters and chaincodes (smart contracts), focussing on their impact on latency and performance in micro-workloads. Similarly, Mazzoni et al. [74] conducted a study on the ConsenSys Quorum blockchain using Hyperledger Caliper to benchmark throughput and latency with a focus on the Raft, Clique PoA, and IBFT consensus algorithms under varying network sizes and transaction loads. Expanding the scope, Chen et al. [75] investigated the performance of private Ethereum blockchain networks in IoT contexts, deploying these networks on an IoT testbed and Google Cloud and examining the latency of processing blocks and their transactions. The authors demonstrated the feasibility of using private Ethereum networks for IoT applications, highlighting the impact of network size, transaction complexity, and hardware resources on overall performance. Lastly, Ferreira et al. [76] focused on the performance of blockchain hash functions, particularly in IoT environments, testing various cryptographic functions such as MD5 and SHA series. The authors evaluated the computational cost and energy consumption of different hash functions in resource-constrained IoT devices, identifying the most suitable algorithms for the integration of blockchain with IoT.

Despite the benefits of the benchmarking approaches, they are largely based on a trial-and-error process and do not guarantee an automated discovery of optimal performance. Moreover, it is essential to recognise that most existing blockchain benchmarking tools treat systems under tests as black boxes and may not capture all the nuances and complexities of real-world IoT. To date, no blockchain benchmarking tool is specifically designed to consider IoT applications. Therefore, the selection of appropriate benchmarks and metrics can be challenging, as different IoT applications may have varying requirements and priorities. It is most likely that users of blockchain benchmarking tools, such as Hyperledger Caliper, must

express the IoT logic in the form of a programming code, which may introduce an unpleasant learning curve.

2.4.2 Simulation Approchaes

Experimenting with a system in real life can be, in several cases, impossible, impractical, or very costly [77]. Simulation is a common practice for mimicking the actual system's behaviour and evaluating systems performance in a controlled and reproducible manner[78]. Simulation requires modelling the system under test, whether it is already in existence or still under design. Simulation modelling can be defined as a method for creating an abstract representation of the system composed of mathematical formulas, structural relationships, and logical relationships [78]. Modelling can be classified as deterministic, stochastic, static or dynamic [79]. The latter can further be classified as continuous or discrete [80].

Simulation allows experimenting without interrupting an actual system or having to implement a corresponding system from scratch. Simulation can be used to describe and predict how various conditions or scenarios will affect the behaviour of the system, as well as to test new policies and design alternatives. Simulation can be considered a viable option for performance evaluation, given the support for modularity and a wide range of configurations [77]. However, developing accurate and realistic simulation models of such complex and heterogeneous systems can be challenging, as they require a deep understanding of the underlying technologies and their interactions. Additionally, simulation results may not always reflect the performance and behaviour of actual physical systems, and validation against real-world data is essential to ensure the accuracy and reliability of simulation-based evaluations. Several simulation studies have attempted to model IoT systems and blockchain platforms.

The remainder of this section reviews relevant IoT simulators, as detailed in Table 2.2. Chapter 3 presents a comprehensive review and analysis of existing blockchain simulators.

2.4.2.1 IoT Simulation

IoT simulation can include modelling end-to-end IoT architecture including, but not limited to, sensors, devices, smart things, connectivity technologies, communication protocols, platforms, cloud infrastructure, and applications. Simulated IoT models enable testing and analysis of various scenarios and configurations without the need for expensive and time-consuming deployment of physical IoT equipment, software, and other resources [81]. Numerous research efforts have been dedicated to simulating different Internet of Things (IoT) aspects, as detailed in Table 2.2.

For instance, *SimIoT* simulator, proposed in [82], is designed to simulate IoT devices and their interaction with cloud environments, facilitating the experimentation and evaluation of resource management techniques in IoT cloud systems. Similarly, the *Edge-Fog Cloud* architecture in [83] focuses on the distributed processing of IoT data using edge devices and fog nodes, with the aim of reducing latency and network congestion. Furthermore, the *IoTSim-Edge* simulator [84] and the *iFogSim* simulator [85] extend the capabilities of the *CloudSim* simulator [86] to model IoT and edge computing environments. *IoTSim-Edge* emphasises capturing the complexity of IoT and edge infrastructure, while *iFogSim* focuses on evaluating resource management policies in fog computing environments based on various metrics. Additionally, *EdgeCloudSim* [87] and *EdgeNetworkCloudSim* [88] offer advanced modelling and performance evaluation for edge computing systems and the placement of service chains in edge cloud environments, respectively. Both simulators extend existing frameworks to capture the specifics of edge computing. *MyiFogSim* [89], a further extension of *iFogSim* [85], targets virtual machine migration in fog computing environments, with a focus on user mobility and maintaining service quality. *FogNetSim++* [90] stands out for modelling distributed fog computing environments, offering unique features such as heterogeneous devices, mobility models, fog node scheduling algorithms, and handover mechanisms. Lastly, *IoTSim-Osmosis* [16] extends existing simulators to model IoT applications over an edge-cloud continuum, with the aim of testing and confirming osmotic computing principles and algorithms in heterogeneous environments.

Table 2.2 Comparison of features and capabilities of different simulators for modelling and simulation IoT applications.

Simulator/ Feature	**SimIoT [82]**	**Edge-Fog [91]**	**IoTSim-Edge [84]**	**iFogSim [85]**	**EdgeCloudSim [87]**	**EdgeNetworkCloudSim [88]**	**MyiFogSim [89]**	**FogNetSim++ [90]**	**IoTsim-Osmosis [16]**
Simulation Purpose	IoT Devices and Cloud	IoT Devices and Edge/Fog	IoT Devices and Edge	IoT Devices, Edge/Fog, and Cloud	IoT Devices and Edge	Cloud and Edge	IoT Devices, Edge/Fog, and Cloud	IoT Devices and Edge/Fog	IoT Devices, Edge/Fog, and Cloud
Source Code	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Simulation Type	Discrete event	N/A	Discrete event	Discrete event	Discrete event	Discrete event	Discrete event	Discrete event	Discrete event
Programming Language	SimJava	Python	Java	Java	Java	Java	Java	C++	Java
Based on	[92]	N/A	[86]	[86]	[86]	[88]	[85]	N/A	[86]
Cloud	Yes	No	No	Yes	No	Yes	Yes	No	Yes
SDN/SDWAN	No	No	No	No	No	No	No	No	Yes
Network	No	No	Yes	Yes	No	No	No	Yes	Yes
Edge/Fog	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Physical Things	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
Blockchain Interface	No	No	No	No	No	No	No	No	No
Monitoring	No	No	No	No	No	No	No	No	No

2.4.2.2 Blockchain Simulation

Several blockchain simulation studies have been proposed to capture various aspects of blockchain, such as blockchain networks, data structure, consensus mechanisms, transaction processing and validation, and others [13]. Chapter 3 provides a comprehensive review and analysis of existing blockchain simulators, which is self-sufficient in this regard.

2.4.2.3 The Lack of Blockchain-based IoT Simulators

Despite the extensive efforts to develop simulators for both blockchain and IoT systems, there remains a notable gap in the research landscape: To our knowledge, no existing simulators specifically focus on the integration of these two pivotal technologies. To address this gap,

Chapter 5 introduces a novel simulation framework designed to simulate the integration of blockchain with IoT systems.

2.4.3 Middleware Approaches

Chapter 6 presents a novel middleware approach for utilising existing IoT simulators to evaluate the performance of real blockchain platforms. The proposed middleware plays a crucial role in enabling seamless communication and interaction between simulated IoT models and blockchain networks. It addresses the challenges of heterogeneity, scalability, and security aspects by providing abstractions and services that facilitate the integration of simulated large-scale IoT systems with a real blockchain platform. It resolves issues related to the distinction between simulated IoT environments and real blockchain platforms, such as synchronisation and the problem of race conditions. This section overviews the middleware concept and reviews related works.

2.4.3.1 Overview of Middleware Concept

The concept of Middleware plays a crucial role in complex systems by serving as an intermediary between disparate components and bridging the gap between them [93]. The concept of middleware abstracts the complexity of different technologies, platforms, and protocols, which allows developers to focus on business logic rather than delving into unnecessary concerns associated with integration and compatibility. The middleware can manage and facilitate seamless interactions among diverse software applications or components, often across distributed networks [94]. It also incorporates features that improve scalability and performance, such as load balancing, caching, and failover mechanisms, while ensuring robust security measures, including authentication, authorisation, and encryption [94]. In environments characterised by complex transactional processes, such as IoT and Blockchain, the middleware concept is indispensable to maintain data consistency and integrity [93].

2.4.3.2 Related Middleware works

Some existing studies propose a middleware approach in the context of Blockchain and IoT to address a wide range of needs. For example, Tapas *et al.* [95] propose a middleware that focusses primarily on securing communication between IoT applications and the Ethereum-like blockchain platform. Similarly, Samaniego and Deters [96] developed a middleware that operates a mining-based blockchain to achieve a zero-trust IoT security management mechanism. Danish [97] suggests a middleware that considers IoT requirements to select the optimal storage strategy, whether data should be stored on-chain or off-chain within a cloud-based storage. Wang *et al.* [98] introduces a middleware that synchronises data between real IoT applications and Hyperledger Fabric while ensuring coherence and consistency. However, it exclusively focuses on SQL queries and lacks a rigorous correctness examination that should have considered failure recovery and concurrent execution. Similarly, Zhou *et al.* [99] propose a similar middleware approach but concentrate only on synchronising read queries from Hyperledger Fabric to a conventional database, omitting write operations to the ledger.

Despite the various middleware solutions presented in the literature for blockchain and IoT, the majority assume access to real IoT infrastructure and Blockchain platforms for specific purposes such as security and storage. Furthermore, several existing works do not consider experimenting with a large-scale IoT system, most likely because of the lack of access to extensive IoT infrastructure. Therefore, in Chapter 6, we present and develop an intermediary solution that bridges the simulated Internet of Things (IoT) and real blockchain technology for performance evaluation purposes.

2.4.4 Machine Learning (ML) for Performance Evaluation

Machine Learning (ML) algorithms can be broadly classified into three main categories: supervised, unsupervised, and reinforcement learning techniques. Supervised learning algorithms learn from labelled data, unsupervised learning algorithms discover hidden patterns in unlabelled data, and reinforcement learning algorithms learn by interacting with an environment [100]. ML techniques have demonstrated effectiveness and robustness for

system evaluation purposes [101]. The process of training an ML model starts by collecting data from the system under study, using these data to train the ML model, and then using the model for:

1. Predicting the system performance under different conditions.
2. Identifying performance areas of strengths and bottlenecks
3. Recommending procedures to enhance the overall performance.

Despite the facilitation offered by simulation and benchmarking tools, both rely on predetermined parameters to model the behaviour of the blockchain system. Consequently, they can be limited in achieving the best possible performance due to the difficulty in determining the optimal values for the configuration parameters. On the other hand, model-based machine learning can play a vital role in optimising the performance and efficiency of the integrated architecture. By leveraging the predictive capability of machine learning models, it is possible to explore a wide range of design options and configurations. This approach enables the identification of optimal system parameters, such as block size, transaction rate, and consensus algorithm, that maximise the performance and scalability of the blockchain-based IoT system.

In their quest for optimal performance, some studies have focused on characterising the performance features of existing blockchain platforms under varying workloads and supported consensus algorithms. In doing so, they aim to reveal the maximum attainable performance in terms of throughput and latency characteristics. For example, Rouhani and Deters [102] conducted a comprehensive performance analysis of Ethereum in which they evaluated the two most widely used Ethereum Virtual Machines (EVM), which are Proof-of-Work (PoW)-based Geth and Proof-of-Authority (PoA)-based Parity. Similarly, an in-depth analysis of the performance of the Quorum blockchain was performed by Baliga et al. [73]. These studies have provided information on performance approximations under a given set of conditions. Nevertheless, there has not been an approach that combines ML techniques with the simulation of Blockchain-based IoT for performance evaluation, which

can benefit from the large data sets generated from the IoT simulation to achieve reliable and more accurate results.

2.5 Conclusion

The background chapter provides an overview of the knowledge body required to cover the concept of Blockchain-based IoT simulation proposed for performance evaluation. It unpacks the main keywords in the thesis topic, which are IoT, Blockchain, Simulation, and performance evaluation, by introducing them to the reader from the perspective of this thesis and highlighting their definition, characteristics, importance, relevance, and examples. The integration of IoT with Blockchain is highlighted in terms of applicability and associated advantages. To motivate and justify the need for a simulation framework for evaluating Blockchain-based IoT systems, this chapter delves into the challenges of evaluating the viability and the overall performance of such complex and heterogeneous systems. Then, it overviews various performance evaluation approaches this thesis considers, which are benchmarking methods, simulation, middleware, and machine learning. Finally, this chapter presented the general methodology outlook, which this thesis follows in conducting its tasks, starting with a systematic study of Blockchain simulators with regard to IoT, the simulation architecture and implementation, the middleware approach for connecting IoT simulators with real blockchain platforms and lastly the machine learning approaches for predicting the overall performance of blockchain-bases IoT systems and recommending optimal values for configuration parameters for achieving a target performance. The following chapters delve into the details of conducting the methodology tasks and presenting their outcomes.

Chapter 3

Blockchain Simulators: A Systematic Mapping Study

Summary

Blockchain technology has gained significant interest from researchers, government, and industry due to its potential to revolutionize various domains. However, the lack of tools for evaluating proposed blockchain solutions and their applications limits their real-world implementation. Organisations are hesitant to adopt blockchain unless they are assured of its benefits, as deploying blockchain systems can be complicated and require substantial resources.

To address this issue, researchers have developed various blockchain simulators that enable designing, testing, and analysing blockchain solutions before actual implementation. The quality and utility of these simulators depend on factors such as their usability, reliability, efficiency, and ability to vary the system parameters.

This chapter presents a systematic mapping study that provides an in-depth review of the state-of-the-art blockchain simulation, comparing their features, capabilities, supported platforms, and consensus mechanisms; examines the configuration parameters and performance metrics covered by each simulator to evaluate the source code quality of publicly available simulators; to determine which performance metrics are scientifically validated in the studies

proposing each simulator; and discuss the current limitations and future research directions in the field of blockchain simulation.

This chapter is organised as follows: Section 3.1 provides a brief introduction to this chapter and highlights the challenges it seeks to address. The Research Question (RQ), the contributions of this chapter, and its relevance to the published paper are explained in Section 3.2. Section 3.3 reviews the literature with regard to similar systematic studies. The research methodology is given in Section 3.4. Section 3.5 outlines the results of the systemic mapping. Section 3.6 provides a detailed discussion about the current simulators and their limits. Finally, Section 3.7 concludes the chapter.

3.1 Introduction

Traditionally, transactions and exchanges between parties have typically been carried out within a centralised structure, which requires the contribution of a third party (e.g. a bank). The hurdle is that this manner of transaction relies mainly on the third party, and if the party encounters a failure, the system completely stops. This problem is commonly known as a single point of failure (SPOF) [103]. What is more, high fees are often associated with third parties. Blockchain has arisen to handle these issues (both SPOF and high fees) by permitting nodes (parties) to associate with one another in a decentralized (aka distributed) way without the contribution of a third party. This is why this technology has gained researchers' attention and enthusiasm in recent years. Formally, blockchain uses a distributed/shared database that logs all the executed transactions within a network [104]. In other words, through the use of a distributed ledger, blockchain makes all transactions available and verifiable by all nodes. Interestingly, the involved nodes are transparent to the chain state update. In particular, thanks to the decentralized nature, nodes can transparently view all transactions occurring at a given time. Each node has its own copy of the chain that is updated following every newly confirmed block.

In the beginning, blockchain was designed for handling the exchange of a digital currency – referred to as Bitcoin [20] – in a peer-to-peer network. Following the success of Bitcoin

technology, a number of other blockchain solutions emerged, such as Ethereum in July 2015 and Hyperledger in December 2015. Since then, these solutions have been applied to various application domains, such as the Internet of Things (IoT) [105]. Specifically, Bitcoin networks offered money-related exchanges through the utilisation of the eponymous tokens: bitcoins. The tokens subsequently developed an immense financial worth [106]. Other blockchains like Ethereum [107] permitted code to be executed within the blockchain system, granting flexibility to the transactions and exchanges, commonly known as smart contracts [108]. By and large, the reason behind the success of blockchain technology is its wide range of merits. Firstly, blockchain creates immutable ledgers, which by nature are unable to be changed or altered. Once a transaction is created and registered, it cannot be altered [109]. Secondly, a vital characteristic of blockchain is its reliance on decentralized control, in which the resources of all nodes involved are used to overcome the issue of SPOF. Thirdly, blockchain can efficiently protect users' identities. Fourthly, blockchain technology has stronger security due to the mitigation of the SPOF issue [108]. Finally, blockchain enables participating nodes to collaboratively process transactions in a timely manner [110].

From a practical perspective, like other systems, experimenting with in-production blockchain systems can cause unnecessary costs, safety threats, resource consumption, and environmental issues [111]. In order to limit possible faults and unexpected failures and to identify bottlenecks, simulations are commonly employed, using various design setups before the implementation of the actual design or when making amendments to existing systems. Similar to emulation, simulation is of high value when tackling complicated tasks in a complicated environment [112].

3.2 Research Questions, Contributions, and Relevance to Published Work

Section 3.2.1 provides a detailed explanation of the Research Question (RQ) and highlights the contributions associated with this chapter. Section 3.2.2 clarifies the relevance of this chapter to the published paper, as outlined by the publications listed in Section 1.4.

3.2.1 Research Question and Contribution

Research Question 1 (RQ1): What techniques and configurations are used in current blockchain simulators? To answer this question, this chapter contributes a comprehensive systematic mapping study that follows the process used in [113]. It critically classifies existing blockchain simulation studies based on their features, capabilities, and limitations. Moreover, it delves into the implementation of existing open-source blockchain simulators to analyse their supported configuration parameters and performance evaluation metrics. It critically compares their code quality, flexibility, reliability, and maintainability. Furthermore, it investigates whether and how these simulators are scientifically validated.

3.2.2 Relevance of the Chapter to the Published Paper

This chapter corresponds to the work published in [13], which provided a review and categorization of blockchain simulators from different aspects. While closely aligned with the original publication, this chapter extends the discussion by analyzing the importance of parameters supported by several simulators, such as P1, P11, M8, and M14, highlighting their relevance to the field.

3.3 Related work

Since 2017 blockchain has gained great attention from the researcher. Therefore, several survey article has recapitulated some of blockchain features, below are some attempt in this direction.

An early attempt to cover the blockchain aspect is carried out by Anilkumar et al. [114]. The main focus of this review article is to cover the popular and noteworthy blockchain-based platforms. Specifically, a brief description of Ethereum, IBM OBC, Intel Sawtooth Lake, BlockStream Sidechain Elements and Eris is given. Moreover, to solidify their notes, the characteristics of these platforms (like usability, currency, and security) are defined. Finally, a small number of simulation platforms for Ethereum are listed. Despite being an informative

review, several aspects are missing, including the recent simulators, evaluation metrics and the set of configured inputs. More recently, Wan et al. [115] provide a review article that sums up blockchain technology and its frontier operations. The main focus of this review is to shed light on the main design principles of blockchain. Specifically, they elaborate on how the data is handled, how the block is validated and how to solve conflict among users. Then, a detailed comparison between the different types of blockchains (public, private) is given. Moving on to the consensus layer, a brief description of PoW and PoS is given. Additionally, they elaborated on the differences between Bitcoin networks and Ethereum networks. Finally, they emphasise the importance of blockchain simulators. In this direction, a brief description of the different simulation models (discrete event, stochastic), with some little examples, is given. However, a comprehensive discussion about the detailed description and features of the simulator is lacking. Moreover, no words are given about how to validate/evaluate blockchain systems.

To cover such a gap, Smetanin et al. [112] provide a review of the state-of-the-art evaluation approaches for blockchain systems. The metrics used for evaluating a blockchain system are reviewed. The lower and upper bound of each metric is given. Additionally, they reviewed the set of challenges associated with each metric. Also, the academic activities for evaluating blockchain, like queuing models, Markov processes, Markov Decision Processes and random walk, are deeply discussed. Then, a discussion about 7 simulators is given. Finally, the main challenges facing these simulators, like lack of adoption and lack of standardization, are listed. By and large, a large number of simulators in the literature are missing in the review. Moreover, no focus is given to the set of configured inputs and outputs for each simulator.

Another direction focusing on distributed ledger technology (DLT) is carried out by Smetanin et al. [116]. They reviewed the tools and instruments used for evaluating DLT. Accordingly, they begin by deeply introducing DLT concepts. Additionally, a detailed analysis of the system behaviour of DLT is covered. Also, the main methodologies used for DLT are reviewed. With no exception, a wide set of simulators designed for simulating DLT

in the literature are reviewed. Finally, the future directions are discussed. The hurdle is that the only focus here is on DLT, which neglects other technologies.

Returning back to the traditional blockchain concept, another review paper is published by Fan et al. [117]. The main focus here is the empirical analysis of blockchain performance and analytical modelling. In the former, the experimental analysis and simulation are reviewed, while in the latter, the stochastic models applied to performance evaluation are investigated. Furthermore, the blockchain benchmarking tools with respect to the layers are deeply discussed. Finally, experimental analysis between blockchain systems is carried out. The lack of focus on configuring inputs and outputs, however, is the only limitation.

Paulavicius et al. [118] provide a systematic review and empirical analysis of blockchain simulators. They begin by introducing DLT with a detailed description of its layer of abstraction. Then, the existing simulators in the literature are listed with comparative analysis. This includes focusing on the model type of the simulator, language/framework used and availability of source code. The interesting point is that they gave a complete picture of the input/output parameters for each layer. Unfortunately, they did not shed light on each simulator's evaluation/validation aspects.

To the best of our knowledge, there is no extensive survey given so far for blockchain simulators. Therefore, this work addresses that gap by systematically analyzing the configuration parameters (inputs) and produced metrics (outputs) supported by each simulator. Furthermore, it investigates which metrics supported by each simulator are scientifically validated/evaluated. Moreover, code quality comparison is carried out to assess the source code of the covered simulators.

3.4 Research Methodology

This chapter conducts a systematic mapping study [113] with the aim of investigating studies pertaining to blockchain simulators. The reason for adopting a systematic mapping method in this chapter is to go beyond the shallow description of existing blockchain simulators. That is, a systematic mapping review not only helps narrow down the subject exploration to

specific questions but also provides analytical methods that critically examine the literature on blockchain simulators. The findings of this study will also enable us to identify and map important research directions. Figure 3.1 shows the five steps of systematic mapping used in this study.

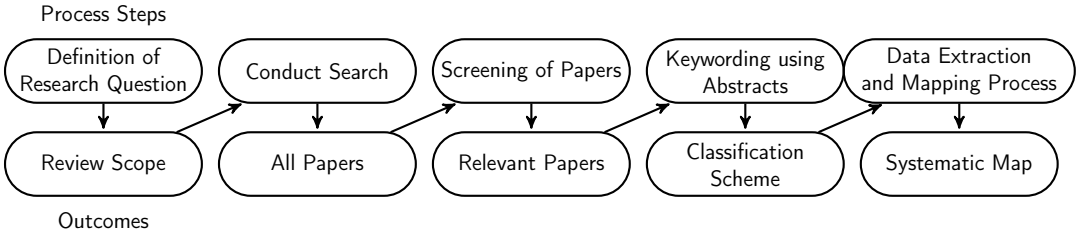


Fig. 3.1 Steps of the systematic mapping study.

3.4.1 Systematic Mapping study Questions

This chapter aims to answer the following questions:

RQ1. What techniques and configurations are used in current blockchain simulators?

RQ2. Which metrics supported by existing blockchain simulators are scientifically validated/evaluated?

RQ3. What are the limitations of the current simulators?

3.4.2 Performing the Literature Search

In this stage, the recent scientific papers and articles relevant to the research topic (blockchain simulators) are identified. For this purpose, the term “blockchain simulator" is used as the keyword to search scientific databases. To specify the search, the query execution ensures the existence of both “blockchain" and “simulator" in the title or abstract. Furthermore, four highly reputable scientific databases were chosen: ACM Digital Library, IEEE Explore, Springer, and Scopus. For precise, accurate, and up-to-date results, high-quality articles published in books, journals, conferences, symposiums, and workshops were selected.

3.4.3 Searching for Relevant Studies

In this stage, studies related to our research questions were searched. We utilised the same searching strategy as in [119]. Specifically, we eliminated all the papers that were irrelevant to the topic based on their titles. If we were uncertain about a paper, we skimmed its abstract. Generally, we utilised exclusion criteria to filter out the search results, by which non-English papers, grey literature or newsletters, and papers with no full text were eliminated from the search.

3.4.4 Searching Abstracts for Keywords

In this stage, keywords were used to classify the relevant papers. We utilised the same keyword process as in [118]. Firstly, we read the abstract of each paper to spot the most significant keywords and their primary contributions. Secondly, we used these extracted keywords to classify the paper. Once all papers were classified, they were investigated, and if needed, switches between classifications were made.

3.4.5 Data Extraction and Mapping Processes

In this stage, the required information was gathered from the papers according to their relevance to the above-stated research questions. Thus, we gathered different data items from each study, which, in turn, highlighted the objectives and contributions of the studies.

3.5 Study Results

This section is designed mainly to outline the results of the systematic mapping study carried out on blockchain simulators. The results of searching and screening for relevant papers are discussed. Afterwards, the resulting classification is given.

3.5.1 Searching and Screening Results

As discussed above in Section 3.4, searching and screening are two steps in the systematic mapping study. In the searching phase, we searched for all papers using the keyword 'blockchain simulator' in different scientific databases, as stated above. The search returned 259 papers in total (as of 7 January 2022). In the screening phase, upon investigating the title and abstract of the papers, we excluded 209 irrelevant papers. These excluded papers are those whose main contribution is not focused on simulating the blockchain. The reason behind the high number of eliminated papers is twofold. Firstly, many papers were irrelevant to our study since our focus was to explore blockchain simulators from a technical perspective. Secondly, some of the excluded papers were about general aspects of blockchains, with no contributions related to our pre-defined research questions. After that, duplicate papers, specifically 23, were removed, resulting in 27 final papers. Finally, we excluded seven papers that were relevant to specific applications; i.e. they provided no useful information on simulation for general blockchains. As a consequence, we ended this phase with 20 papers on which to carry out our systematic mapping study.

3.5.2 Taxonomy for Blockchain Simulators

Following the *keywording strategy* discussed in Section 3.4, we characterised the simulators according to several criteria:

1. **Overview of blockchain simulators**: This represents further information related to the identified blockchain simulators in Table 3.1.

2. **Comparative analysis**: This critically compares existing simulators based on their available open-source implementation. The comparison is focused on their supported configuration parameters (inputs) and generated metrics (outputs) as per Table 3.3.

3. **Code Quality**: This provides a general report about the source code, including a number of bugs, code smell, and security hotspot, as shown in Table 3.4.

4. **Scientifically Validated/Evaluated Metrics**: This is to represent which supported metrics by each simulator are scientifically validated/evaluated in their corresponding papers, as shown in Table 3.5.

3.5.3 Overview of Blockchain Simulators

The systematic review resulted in 20 relevant blockchain simulators. Table 3.1 lists and examines them based on the following criteria:

1. **Code availability**: Reflects if the source code of the simulator is publicly available on GitHub.

2. **Programming language and library**: Reflects the programming language and the libraries used for coding the simulator.

3. **Core of the simulator**: Reflects if the simulator inherits a base simulator or is built from scratch.

4. **Purpose and objective**: Reflects if the simulator is designed for assessing performance and/or security.

5. **Blockchain platform**: Reflects the type/platform of the simulated blockchain, i.e. Bitcoin, Ethereum, and IOTA.

6. **Consensus algorithm**: Reflects the implemented consensus algorithm in the simulator.

This section provides a summary of each blockchain simulator as follows. An early attempt in 2015 was carried out by Miller and Jansen [120], who proposed a *discrete event* Shadow-Bitcoin simulator. Its main focus was to simulate Bitcoin networks. This simulator utilises the concept of shadowing, which permits the use of parallel processing. Accordingly, the simulator has the ability to provide insights about the simulated multi-threading application in a scalable manner. The shadow-Bitcoin simulator is recognised as a dynamic and stochastic simulator that was developed using pure Python. The code is publicly available on GitHub. PoW consensus algorithm is implemented in this simulator.

The simulator is able to focus on transaction propagation and provide insights about the system's performance and security.

Wang and Kin [121] propose a blockchain simulator named FastChain, which extends Shadow-Bitcoin to support evaluating the correlation between throughput, block propagation time, and bandwidth-informed neighbour selection algorithms. FastChain facilitates the tuning of several parameters that influence the blockchain's performance with regard to block rate and throughput.

A similar attempt is carried out by Stoykov et al. [122], who proposed a *discrete-event* and dynamic simulator named *VIBES*, which stands for Visualisations of Interactive, Blockchain, Extended Simulations. It is coded in Scala with the aim of enabling empirical insights and analytics about the blockchain performance under various parameters, such as network topology and area size, are simulated, with the aims of predicting the total processing time, the total number of transactions processed, throughput (transactions per second), block propagation delay. While VIBES focuses on simulating Bitcoin-style blockchain networks, Ethereum is out of its scope. Following this, Deshpande et al. [123] proposed *ethernet VIBES* (eVIBES), a further improvement of VIBES, to mimic the behaviour of the Ethereum network. It depends on a reactive manifesto model using an orchestrator and reducer in its core. The orchestrator is used to control the simulator, which receives the parameter settings from the user and feeds them to the Ethereum network.

Another *discrete-event*, dynamic and stochastic simulator for simulating generic blockchain is proposed by Piriou and Dumas [124]. The authors find that the previous simulators depend on applying consensus algorithms in a sequential manner, which may result in a double-spending attack. Accordingly, they mainly contribute a generic blockchain-style simulator with a focus on distributed consensus protocols, namely, PoW and PoS. The source code is written in Python and utilises the PyCATSHOO library, which allows for dealing with a large number of parameters of interest. The simulator sheds light on the impact of various parameters on the overall performance. This is done through the integration with the Monte Carlo simulation, which is known for its ability to check dynamic behaviour.

Wang et al. [125] also proposed a *discrete-event* and stochastic simulator that has the ability to simulate the complex and dynamic behaviour of the Bitcoin blockchain network. The main aim of this simulator is to evaluate the blockchain performance by setting various parameters such as simulation time, number of nodes, mining time, block size and transaction size.

Aside from traditional simulations, Memon et al. [126] proposed a *queuing* blockchain simulator using the M/M/n/L queuing system. This simulator is coded in Java and was designed with the aim of simulating PoW-based mining operations, which are known to be costly and time-consuming tasks.

BlockSim is one of the well-established simulators, which is initially proposed by Alharby and Moorsel [127]; and has been further developed in [128]. The source code, written in pure Python, is publicly available on GitHub. This simulator aims to mimic the implementations of public blockchains (Bitcoin and Ethereum) using PoW. Like other *discrete-event* simulators, it enables testing the influence of various configurations on the overall blockchain's performance. This is done via two different modes of simulation, which are *full transactions* and *light transactions* techniques. The former emulates a realistic blockchain network and records detailed logs of a typical transaction journey. According to the simulator's authors, this technique is time and resources-intensive; however, it provides an in-depth insight into the network latency measurement. On the other hand, the latter simulates the blockchain network's behaviour by employing a single transaction pool and omits several transaction details, which is, according to its authors, more effective in terms of time and computing consumption; however, it can be useful for other purposes other than latency measurement such as transactions throughput and execution cost.

Further, Polge et al. [129] appraise *BlockSim* [127] simulator's performance with Bitcoin. However, as per their study, most of the existing simulators lack several important features, such as extensibility and failure to cover all aspects/metrics. Therefore, an extended version referred to as BlockPerf is proposed to alleviate BlockSim issues. It is also written in Python with the aim to realise a stochastic, dynamic and *discrete event* simulator depending on PoW consensus protocol. Similarly, Agrawal et al. [130] state that *BlockSim* [127] is

restricted to simulating blockchain networks (either Bitcoin or Ethereum) over a single CPU, which results in bottleneck problems. Therefore, *BlockSim-Net*, another extended version of BlockSim, is proposed as a distributed simulator. What distinguishes BlockSim-net from the traditional *BlockSim* is the ability to focus on the actual propagation of blocks. Furthermore, it is useful for the assessment of blockchain application security (such as a selfish mining attack on PoW).

Another *discrete-event* blockchain simulator is proposed by Faria and Correia [131]; also named as BlockSim. Not to be confused with other *BlockSim* simulators in [127], it is also coded in Python and leverages the SimPy 3 library. Unlike the other *BlockSim*, it simulates blockchain networks over specific intervals. Thus making it a *discrete-event*, stochastic and a *dynamic* simulator.

Fattahi et al. [132] have stated that BlockSim, proposed by Faria and Correia [131], is a reliable simulator able to evaluate blockchains. However, it does not simulate some real features, such as Merkle tree transactions. Therefore, they proposed an extended version of BlockSim, referred to as SIMulator, for application to blockchains (SIMBA). Similar to its forerunner, it is written in Python and uses SimPy 3 features. Also similar to BlockSim [131], it is a *stochastic* and *discrete-event* simulator.

Pandey et al. [133] also proposed another blockchain simulator named BlockSIM, with "sim" capitalised. BlockSIM. It is a *stochastic discrete-event* and *dynamic* simulator written in Python using the SimPy library. It facilitates evaluating the performance of Ethereum and Hyperledger blockchain networks and supports both the PoW and PoA protocols. More recently, in 2020, Alsahan et al. [134] extended this work and proposed a local Bitcoin simulator that has the ability to enable fast simulation for large-scale networks without affecting the mining process quality. It is a *virtualization* based simulator with the ability to model different network topologies.

Wang et al. [135] proposed a ChainSim simulator that evaluates peer-to-peer blockchain networks. The simulator's main aim is to alleviate the burden of computing resources and the financial cost needed for deploying and experimenting with blockchain systems. Such

a simulator has the ability to simulate blockchain-based applications with thousands of involved nodes.

Another direction is to simulate concurrent operations within blockchain networks. To do so, Gouda et al. [136] propose the Blockeval simulator, which uses deep learning algorithms to allow the simulation of scalable blockchain systems. This makes Blockeval's a modular simulator for assessing the performance of private blockchain networks. Moreover, it can elaborate on the metrics used for assessing the system. Blockeval's main contribution is twofold. Firstly, it can be used to assess the scalability of the proposed blockchain system. Secondly, it can analyse the security of the proposed blockchain system.

Another attempt that was modelled with both PoW and PoS protocols was carried out by Aoki et al. [137]. The simulator, the SimBlock, was written in Java. It can be differentiated from its peers by its ability to investigate blockchain performance with different node behaviours. SimBlock is considered as a *stochastic, dynamic*, and discrete-event simulator that focuses on modelling block generation and message transmissions.

This work has been further extended by Basile et al. [138]. They state that SimBlock fails to simulate the block mining process; thus, they address this limitation in their work. Banno and Shudo [139] also extended SimBlock to support the simulation of thousands of nodes and to improve the neighbour selection strategy. Moreover, it enables the assessment of the influence of relay networks on overall performance.

Beyond typical blockchain data structure, Zander et al. [140] proposed the DAGsim simulator that uses a *Directed Acyclic Graph* (DAG) approach to represent a scalable distributed ledger, which allows simulating a vast amount of transactions over a large number of nodes. This simulator is influenced by the philosophy of a DLT network called IoTA [141]. An interesting feature provided by DAGsim is the support of modelling and experimenting with malicious nodes. The code is written in Python with the ability to run in $O(n^2)$, where n is the number of nodes. This made DAGsim an *agent based* stochastic and dynamic simulator. The final output of the simulator is a DAG representing the structure of all transactions.

By and large, there are four simulation models found in the literature of blockchain simulators, namely, *stochastic, dynamic, discrete event, virtualization* models. Bear in

mind that all the simulators are discrete-event except the queuing model simulation in [126], the agent-based simulation in [140], and the virtualization-based modelling in [134]. We also find that existing blockchain simulators do not support consensus mechanisms other than PoW, PoA and PoS; among them, PoW is the only protocol implemented in all simulators. As regards PoS and PoA, there is only one simulator that implements each of them. Another interesting observation is that Bitcoin is the most popular blockchain network, being implemented in 15 out of the 21 covered simulators.

Table 3.1 A summary of blockchain simulators. Each row represents a separate simulator, while the columns represent the features. Note that all the simulators are stochastic dynamic simulators and characterized as being discrete-event except the three simulators, namely: Modelling by queuing theory simulation [126] that is queuing model, DAGsim [140] that is agent-based, and Local Bitcoin [134] that is virtualization based.

Simulator	Year	GitHub Code	Prog. Lang.	Library	Core	PRF.	Security	Platform	Consensus
Shadow-Bitcoin [120]	2015	✓[1]	Python	N/A	Shadow	✓	✓	Bitcoin	PoW
VIBES [122]	2017	✓[2]	Scala	N/A	N/A	✓	✓	Bitcoin	PoW
Stochastic Blockchain Models [124]	2018	N/A	Python	PyCATSHOO	N/A	✓	✓	Generic	PoW/PoS
Behavior and Quality of Blockchain [125]	2018	N/A	Python	SimPy	N/A	✓	✗	Bitcoin	PoW
eVIBES [123]	2018	✓[3]	Scala	N/A	[122]	✓	✗	Ethereum	PoW
Modeling by Queuing Theory [126]	2018	N/A	Java	N/A	[142]	✓	✗	Bitcoin	PoW
BlockSIM [133]	2019	✓[4]	Python	SimPy 3.0	N/A	✗	✗	Ethereum/ Hyperledger	PoW/PoA
DAGsim [140]	2019	N/A	Python	N/A	N/A	✓	✗	IOTA	IOTA
BlockSim [127]	2019	✓[5]	Python	N/A	N/A	✓	✗	Bitcoin/ Ethereum	PoW
FastChain [121]	2019	N/A	N/A	N/A	[120]	✓	✗	Bitcoin	PoW
simBlock [137]	2019	✓[6]	Java	N/A	N/A	✓	✗	Bitcoin	PoW/PoS
Blocksim [131]	2019	✓[7]	Python	SimPy 3.0	N/A	✓	✗	Bitcoin/ Ethereum	PoW
Ext-simblock [139]	2019	N/A	Java	N/A	[137]	✓	✗	Bitcoin	PoW
BlockSim-Net [130]	2020	N/A	Python	N/A	[127]	✓	✗	Bitcoin/ Ethereum	PoW
ChainSim [135]	2020	N/A	Python	N/A	[127]	✓	✗	Bitcoin/ Ethereum	PoW
Local Bitcoin Network [134]	2020	✓[8]	Python	N/A	N/A	✓	✗	Bitcoin	PoW
SIMBA [132]	2020	✓[9]	Python	SimPy 3.0	[131]	✓	✗	Bitcoin	PoW
Ext 2-simBlock [138]	2021	N/A	Java	N/A	[137]	✓	✗	Bitcoin	PoW
BlockPerf [129]	2021	✓[10]	Python	N/A	[127]	✓	✗	Bitcoin	PoW
BlockEval [136]	2021	✓[11]	Python	SimPy	N/A	✓	✗	Bitcoin	PoW

[1] https://github.com/shadow/shadow-plugin-bitcoin
[2] https://github.com/i13-msrg/vibes
[3] https://github.com/i13-msrg/evibes
[4] https://github.com/RoseBay-Consulting/BlockSim
[5] https://github.com/maher243/BlockSim
[6] https://github.com/dsg-titech/simblock
[7] https://github.com/carlosfaria94/blocksim
[8] https://github.com/noureddinel/core-bitcoin-net-simulator
[9] https://github.com/nyit-vancouver/SIMBA
[10] https://github.com/Deadlyelder/BlockPerf
[11] https://github.com/deepakgouda/BlockEval

3.5.4 Comparative Analysis

After searching the internet for blockchain simulators and noting their main design principles, we focused on the operational range of each simulator. Specifically, we study the set of supported configuration parameters (inputs) and provided metrics (outputs) by each simulator. A brief description of each is given in Table 3.2. To this end, no existing blockchain simulator can support all configurations parameters (P) and produced metrics (M). Table 3.2 highlights these parameters/metrics and their association with each of the blockchain layers. By inspecting Table 3.3, we can notice that out of the 16 parameters, the least number of implemented parameters is 5, which is the case with Local Bitcoin [134]. In other words, at least 40% of the parameters are implemented. On the other hand, at most, about 81% of the parameters are implemented, which is the case with both BlockSim [131] and BlockPerf [129]; i.e. 13 parameters. Similarly, not all the metrics are supported by all simulators. The least number of supported metrics is 4, which represents 25% of the metrics as is with BlockSIM [133]. By contrast, at most 88% of the parameters are implemented, which is the case with BlockPerf [129]; i.e. 13 parameters. Accordingly, we can notice that BlockPerf [129] is the richest simulator with both parameters and metrics.

Table 3.2 The definition of the parameters and metrics used with respect to blockchain layers.

Layer	P/M	Definition	P/M	Definition
Network Layer	**(P1)** Total number of nodes	number of involved nodes	**(P7)** Payload transaction size (unit: Megabyte)	Maximum transaction size
	(P1.1) Regions of nodes (unit: geographical)	Geographical location of each node	**(P8)** Block size (unit: Megabyte)	Maximum block size configured
	(P2) Total number of connections	Number of possible connections	**(M1)** Average block size	Average block sizes
	(P3) Average block propagation delay (unit: seconds)	Average time delayed in the propagation process of each block	**(M2)** Average block propagation time	Average time taken to propagate blocks
	(P4) Average transaction propagation delay (unit: seconds)	Average time delayed in the propagation process of each transaction	**(M3)** Average transaction propagation time	Average time taken by the simulator to propagate transaction
	(P5) Average bandwidth (unit: bits per second)	Bandwidth assumed for the simulated network	**(M4)** Throughput (unit: Tx/second)	Throughput taken to the end of the simulations
	(P6) Average latency (unit: seconds)	Average latency assumed for the simulated network		
Data layer	**(P9)** Generate random transactions (unit: Integer of Tx per second)	Automatically generated transactions	**(M6)** Security	Security assessement
	(M5) Chain of block	Resulting chain		
Consensus layer	**(P10)** Average mining power (Hash Rate)	Average used mining power	**(M7)** Average block interval	Average time for the blocks to accept transactions
	(P11) PoW consensus Algorithm	Ability to implement PoW consensus algorithm	**(M8)** Number of generated blocks	Total number of generated blocks
	(P12) Other consensus Algorithm	Ability to implement other consensus algorithms than PoW	**(M9)** Number of mined blocks	Total number of mined blocks
	(P13) Average transaction fee (unit: cryptocurrency)	Average transaction fees	**(M10)** Rate of orphan blocks (unit: percentage)	Percentage of the orphan blocks
	(P14) Block Interval (unit: seconds)	Average time for creating a block	**(M11)** Fork Resolution	Determine forks occurred as protocol change
Incentive layer	**(P15)** Reward for mining a new block (unit: cryptocurrency)	Amount of reward configured	**(M12)** Reward for a miner (unit: cryptocurrency)	Amount of reward consumed
Execution layer	**(M13)** Time of executing a contract (unit: seconds)	Time taken to execute a contract	**(M14)** Validation of contract and execution time	How the simulator validates the contract
Application layer	**(P16)** Simulation run time (unit: seconds)	Configured simulation time	**(M16)** Simulation time (unit: seconds)	Represents the actual time taken by the simulator
	(M15) Resources usage	How the simulator keeps track of the resource usage/utilization		

Table 3.3 Set of parameters available in each simulator. For a detailed description of the parameters, refer to Subsection 3.5.4. The sign ● means that the parameter is available in the simulator, while the sign ○ means that the parameter is not available in the simulator. The last row represents the total number of simulators supporting a particular parameter. Similarly, the last column represents the total number of parameters supported by a particular simulator. The bold values represent the maximum values, and the underlined values represent the minimum values.

Simulator	**Parameters**																	**Total**
	P1	P1.1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	
Shadow-Bitcoin [120]	●	●	○	○	○	●	○	○	○	●	○	●	○	○	○	●	○	6
VIBES [122]	●	○	●	●	●	○	○	●	●	●	●	●	○	○	○	●	○	10
eVIBES [123]	●	○	○	○	○	○	○	●	○	●	●	●	○	○	○	●	●	6
BlockSIM [133]	●	○	○	○	○	○	○	●	●	●	●	●	○	○	●	○	●	8
BlockSim [127]	●	○	○	●	●	○	○	●	●	●	●	●	○	●	●	●	●	12
SimBlock [137]	●	●	●	○	○	●	●	○	●	○	●	●	●	●	●	○	●	12
BlockSim [131]	●	●	●	●	○	●	●	●	●	●	●	●	○	●	●	○	○	**13**
Local Bitcoin [134]	●	○	○	●	○	○	○	○	○	○	●	●	○	○	○	○	●	5
SIMBA [132]	●	●	●	●	○	●	●	●	●	●	●	●	○	○	●	○	○	12
BlockPerf [129]	●	●	○	●	●	○	○	●	●	●	●	●	○	●	●	●	●	**13**
BlockEval [136]	●	●	●	●	○	○	○	●	●	●	○	●	○	○	●	●	●	11
Total	**11**	6	5	7	3	4	3	8	8	9	9	**11**	1	4	7	6	7	

Simulator	**Metrics**																**Total**
	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M13	M14	M15	M16	
Shadow-Bitcoin [120]	○	○	○	●	○	●	●	○	●	○	○	●	●	○	●	●	8
VIBES [122]	●	●	●	●	●	●	●	●	●	●	○	○	○	○	○	●	11
eVIBES [123]	●	●	○	●	○	○	●	●	●	●	●	●	○	●	○	●	11
BlockSIM [133]	●	○	○	●	○	○	●	●	○	○	○	○	○	○	○	○	4
BlockSim [127]	●	○	●	●	●	○	○	●	●	●	●	●	○	○	○	●	10
SimBlock [137]	●	●	○	○	●	○	●	●	○	●	○	○	○	○	○	●	7
BlockSim [131]	●	●	●	●	○	○	○	●	●	○	●	○	○	○	○	○	7
Local Bitcoin [134]	○	○	○	●	○	○	○	●	●	○	●	●	○	○	●	○	6
SIMBA [132]	●	●	○	●	○	○	○	●	●	○	●	○	○	○	○	●	7
BlockPerf [129]	●	●	●	●	●	○	○	●	●	●	●	●	●	●	●	●	**14**
BlockEval [136]	●	●	○	○	●	○	○	●	●	●	●	●	○	○	○	●	9
Total	9	7	4	9	5	2	5	**10**	9	6	7	6	2	2	3	8	

3.5.5 Code Quality

To solidify the view about the simulators, we assess their source code from different aspects, which will help researchers determine the future trends and modifications needed for each simulator. Below is a detailed description of the aspects used.

1. **Lines of Code**: The number of code lines.
2. **Comments(%)**: The percentage of commented lines.
3. **Duplication(%)**: The percentage of duplicated lines.
4. **Files**: The number of code files.
5. **Bugs**: The number of bugs.
6. **Code Smells**: The complexity degree of understanding the code.
7. **Security Hotspots**: The number of source code parts that need a major overhaul from the security aspect.

We have used the Sonarqube [143] and the Count Lines of Code (CLOC) [144] tool to assess the codes of the simulators. An interesting point about such a tool is its ability to shed light on the previous aspects mentioned above. A detailed description of this comparison is shown in Table 3.4. With a quick skimming of the table, we can notice that eVIBES [123] has the largest number of lines while being Bug-free. On the other hand, local Bitcoin [134] has the least number of lines while also being Bug-free. Another interesting point is that six simulators out of 11 are Bug-free. Despite being the most reputable, *BlockSim* [127] has the largest number of bugs with about 6% code duplication.

Table 3.4 Evaluating aspects of each simulator using Sonarqube and CLOC tool.

Simulator	Evaluation Aspect						
	Lines of Code	**Comments** (%)	**Duplication** (%)	**Files**	**Bugs**	**Code Smells**	**Security Hotspots**
Shadow-Bitcoin [120]	1218	12	0	17	4	117	12
VIBES [122]	20773	1.8	0	118	2	19	2
eVIBES [123]	25909	5	0	166	0	53	0
BlockSIM [133]	712	22	0	35	0	28	4
BlockSim [127]	1730	18	5.8	2.9	65	215	11
SimBlock [137]	2487	53	2.4	37	6	128	9
BlockSim [131]	1721	18	0	27	0	28	4
Local Bitcoin [134]	323	13	15	6	0	4	36
SIMBA [132]	2284	15	3	36	0	56	4
BlockPerf [129]	1668	4	0	26	0	26	6
BlockEval [136]	2761	76	0	20	1	71	1

3.5.6 Scientifically Validated/Evaluated Metrics

For the picture to be complete, we also shed light on the scientifically validated/evaluated metrics in the corresponding paper for each simulator. A summary of this is shown in Table 3.5. According to the reviewed simulators, there are 15 validation/evaluation metrics; a brief description of each is given below.

1. **Block propagation time**: The time taken from sending to receiving a block.
2. **Transaction propagation time**: The time taken from sending to receiving a transaction.
3. **Average of block size**: The average block size generated during the simulation period.
4. **Transaction throughput**: The rate at which a set of valid committed transactions occurs in a defined time period.
5. **Network delay**: The total delay in the network.
6. **Number of generated blocks**: The total number of the generated blocks during a simulation period.

7. **Number of valid blocks**: The total number of valid blocks created during the simulation.

8. **Block verification time**: The average time taken to verify the generated blocks during the simulation period.

9. **Uncle or stale blocks**: The number of generated uncle/stale blocks during a simulation period.

10. **Fork resolution**: Reflects whether a fork has ever occurred during the simulation.

11. **Pending transaction**: The simulator's ability to keep track of the transaction number while awaiting confirmation.

12. **Mining difficulty**: The difficulty assigned for each transaction.

13. **Mining reward**: The amount of the used rewards for the mining processes that occurred throughout the simulation.

14. **Processing speed**: Tthe average time the simulator takes to carry out a specific task during the simulation period.

15. **System stability**: Eventual consistency achieved over time by participating blockchain network nodes regarding the ledger replicas.

Having mentioned the previous metrics, we also focused on the distribution of these metrics over the blockchain layers. Table 3.5 shows the set of metrics associated with each layer. Moreover, from the table, we can see which metrics are implemented for each simulator. On closer inspection, we notice that the majority of the metrics (7 out of 15) are focusing on the consensus layer. On the other hand, the incentive layer has the least share of the metrics (only one metric). Also, there are nine simulators focusing on validating and evaluating the network layer. On the consensus layer, we can notice that only seven simulators validate such a layer. With the least attention, the incentive layer is validated in one simulator only. From another viewpoint, with six validation/evaluation metrics, VIBES [122] comes out at

the top, whereas shadow-Bitcoin [120] has the least number of validation/evaluation metrics (1 metric).

Table 3.5 The scientifically validated/evaluated metrics of each simulator with respect to different layers. The signs ● and ○ depicts the available and missing metrics, respectively. The last row represents the total number of simulators used in a particular metric. Similarly, the last column represents the total number of metrics used by a particular simulator. The bold values represent the maximum values, and the underlined values represent the minimum values.

Simulator	**Approach used**															**Total**
	Network layer					**Consensus layer**							**Incentive layer**	**General**		
	Block propagation time	Transaction propagation time	Average of block size	Transaction throughput	Network delay	Number of generated block	Number of mined block	Block verification time	Uncle or stale blocks	Fork resolution	Pending transaction	Mining difficulty	Mining reward	Processing speed	System stability	
Shadow-Bitcoin [120]	○	●	○	○	○	○	○	○	○	○	○	○	○	○	○	1
VIBES [122]	●	○	●	●	○	●	○	○	●	○	○	○	○	●	○	**6**
eVIBES [123]	○	○	○	○	○	●	○	●	○	○	○	○	○	○	○	2
BlockSIM [133]	○	○	○	●	○	○	○	○	○	○	○	○	○	○	●	2
BlockSim [127]	○	○	○	●	○	●	○	○	●	○	○	○	○	○	○	3
SimBlock [137]	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	1
BlockSim [131]	●	●	○	○	○	○	○	○	○	○	○	○	○	○	○	2
Local Bitcoin [134]	○	○	○	○	●	○	○	○	○	○	○	●	○	○	○	2
SIMBA [132]	●	○	●	○	○	○	○	●	○	○	○	○	○	○	○	3
BlockPerf [129]	○	○	●	●	○	○	●	○	●	○	○	○	●	○	○	5
BlockEval [136]	○	○	○	○	○	○	○	○	○	●	●	○	○	○	○	2
Total	**4**	2	3	**4**	1	3	1	2	3	1	1	1	1	1	1	

3.6 Discussion

According to the results obtained in Section 3.5, this section is dedicated mainly to outlining proposed solutions to the predefined research questions stated in Section 3.4.

RQ1. What techniques and configurations are used in current blockchain simulators?

The systematic mapping study reveals that 11 out of 20 blockchain simulation studies publicly provide their source code. The majority of existing simulators are stochastic, which is immensely complex and requires in-depth statistical capabilities to ensure realistic outcomes. Additionally, several blockchain simulators are dynamic, aligning well with the evolving nature of blockchain networks. For example, dynamic simulators allow for the investigation of blockchain behaviour under varying conditions, such as changes in the number of miners (nodes) over time.

Regarding configuration parameters, existing blockchain simulators show significant variation in their focus on certain parameters (refer to Table 3.3). Notably, BlockSim [131], and BlockPerf [129] emerge as the most comprehensive, supporting 13 configuration parameters each. In contrast, Local Bitcoin [134] supports the fewest parameters, controlling only five. The total number of nodes (P1) and Proof of Work (PoW) consensus algorithm (P11) are consistently included across simulators, underscoring their importance in evaluating network scalability, security, and performance.

The consistent inclusion of the total number of nodes (P1) across simulators highlights its foundational role in blockchain simulations for assessing network scalability, decentralization, and robustness within blockchain environments. Simulators that incorporate this parameter facilitate meaningful comparisons and benchmarking across diverse blockchain settings. The extensive support for PoW (P11) reflects its central role in securing blockchain networks, especially those similar to Bitcoin. PoW requires computational effort to validate transactions, making it a key factor in performance evaluations. However, the limited support for alternative consensus algorithms (P12) in existing simulators reveals a significant gap in current simulation capabilities. This gap is important because it restricts the ability to explore and test newer, potentially more efficient consensus mechanisms that could address some of the limitations of PoW, such as high energy consumption and slower transaction times.

By inspecting the generated metrics by each simulator, we find that the number of generated blocks (M8) is widely supported, except by Shadow-Bitcoin [120], indicating its

importance for measuring network throughput and latency. This metric reflects the network's ability to process transactions efficiently, a crucial aspect of blockchain performance. On the other hand, contract validation and execution time (M14) are the least supported metric, with only eVIBES [123] and BlockPerf [129] including it. Given the increasing integration of smart contracts into blockchain platforms, understanding the performance implications of M14 is essential for optimizing blockchain usability and efficiency.

RQ2. Which metrics supported by existing blockchain simulators are scientifically validated/evaluated?

To better understand the system's behaviour, a set of validation/evaluation metrics is needed to assess the overall system's performance. Generally, blockchain systems can be judged from different viewpoints as follows.

1. **Usability and reliability**: *Is the system ready for being implemented in a real-world situation?* This is related to assessing the deployed network. The network can be assessed using two metrics: volume of P2P traffic and packet loss. The former represents the network's ability to perform under elevated traffic, while the latter represents the ratio of lost packets. In view of this metric, the system is usable if it is able to exchange a large amount of traffic with the fewest lost packets.

2. **Functional testing**: *Is the system able to provide promising results?* This is related to assessing the blockchain itself. Blockchain can be assessed using three metrics: transaction throughput, latency, and finality time. The transaction throughput represents the amount of successfully committed transactions per second. Blockchain is successful if it is able to provide high transaction throughput, especially in the case of permissionless blockchain. The latency represents the time taken for the effect of the transaction to be reflected; it should be minimal. Finally, the finality time represents the transaction's time to be committed. This metric is highly important as if wrongly adjusted, it decreases the system's efficiency.

3. **Resource testing**: *Are the involved nodes operating properly?* This is related to assessing the involved nodes. Theoretically, this is assessed using the resource metrics, which represent the computational power (CPU/GPU, memory, storage capacity, connectivity, and cache ratio) of the nodes. This metric is of high importance, as low resources can incur a significant negative impact on the chain.

   The systematic mapping study reveals that there is no simulator able to assess the system's performance from all the different viewpoints. From the source code viewpoint, existing blockchain simulators support multiple metrics. However, their corresponding papers do not validate/evaluate all of them. Again, the problem is not with the simulators themselves but with the target application. Focusing on the network layer, all simulators except eVIBES [123] and BlockEval [136] are concerned with relevant metrics. VIBES [122] implements 3 out of the five network metrics. BlockSim [131], SIMBA [132] and BlockPerf [129] implement only 2 out of them. Shadow-Bitcoin [120] implements only one network metric.

   On the consensus layer, seven simulators implement associated metrics. VIBES [122], eVIBES [123], *BlockSim* [127], BlockPerf [129], and BlockEval [136] implement only 2 out of the consensus metrics. Local Bitcoin [134] and SIMBA [132] implement only one metric related to the consensus layer.

   On the incentive layer, the mining reward metric is only supported by BlockPerf [129]. Regarding the general metrics, the processing speed metric is only implemented in VIBES [122], while the system stability metric is only supported by BlockSIM [133].

RQ3. What are the limitations of the current simulators?

The limitations of the covered simulators can be expressed by the following viewpoints.

1. **Usability**: Great headway has been made in the field of simulating blockchain, but the work is limited. The usability of such simulators is hindered by the fact that there are a large number of parameters that need to be adjusted (such as the simulation scenario and the execution environment); this necessitates a deep understanding of blockchain technology. Furthermore, most of the existing simulators require coding

skills and/or knowledge of command-line interfaces. However, some effort has been made to mitigate this issue through the use of web interfaces, as in VIBES [122] and eVIBES [123].

2. **Availability and Scalability**: the majority of the simulators virtually run multiple blockchain nodes on a single machine, which naturally suffers from limited resources. Thus, it can be challenging to generalise the outcomes of simulated blockchain models on real-world blockchain networks (i.e. resource usage and energy consumption). Moreover, there is no focus on node behaviour under different sources of power and resources, i.e. blockchain running on custom ASIC-based computers. Additionally, to the best of our knowledge, the majority of existing simulators do not draw much attention to the consensus layer; and many of them solely focus on PoW algorithms while neglecting others, such as PoS, PoA, Raft, and others.

3. **Applicability**: the majority of simulators are predominantly targeted for financial applications, such as Bitcoin. However, no generalisation has been made outside the field of finance. The hurdle is that there is no simulation to support the integration of blockchain with other technologies. For instance, none of the covered simulators are specifically tailored to experiment with the intersection of blockchain with other domains such as IoT, Cloud, Cybersecurity, Supply Chain, and others.

3.7 Conclusion

Modelling and simulation have been useful in several disciplines, and blockchain is no exception. Such practice allows for experimenting with complex systems, such as blockchain systems, with minimum cost and effort. The presented systematic study mainly investigates blockchain simulators, their features and capabilities. The results showed that there are 20 simulators dedicated to this purpose. We focused on 11 simulators whose source code is publicly available. Additionally, it highlights scientifically validated/evaluated metrics supported by each simulator.

Most existing simulators support stochastic, dynamic and discrete event modelling approaches. We find that the majority of existing blockchain simulators support dynamic modelling, which aligns well with the nature of blockchain networks. Regarding the evaluation/validation process, not all simulators are interested in the same collection of evaluation metrics or blockchain layers. Moreover, we find that not all supported metrics by each simulator are scientifically validated/evaluated in their corresponding papers.

To date, no blockchain simulator can comprehensively cover all blockchain facets. Moreover, existing blockchain simulators are unsuitable for supporting other technologies, such as cloud and IoT. Blockchain simulation is generally still in its infancy, and further research is needed in this direction. The next chapters address the lack of existing Blockchain simulation for IoT purposes by introducing a simulation framework that considers complex architectures where Blockchain and IoT paradigms are integrated. The simulation framework focuses on evaluating the performance of modelled Blockchain-based IoT scenarios in various ways, including pure simulation, middleware that connects the simulator with real blockchain platforms, and machine learning techniques for performance prediction, as well as planning configuration parameters to produce a target performance level.

Chapter 4

Investigating the Requirement of Building Blockchain-based IoT Simulation

Summary

The Internet of Things (IoT) has enabled the management of a vast number of smart devices. Still, existing IoT architectures tend to depend on centralised models, which are at risk of single points of failure and security limitations. Blockchain technology offers a potential solution to improve IoT architectures and realise unprecedented opportunities. However, both blockchain and IoT are complex, making it difficult to assess the performance of IoT systems integrated with blockchain, especially in the presence of highly heterogeneous devices. Therefore, simulation tools are needed to enable the modelling and evaluation of such systems prior to real-world deployment.

This chapter is the first part of this thesis effort to realise a simulation tool for blockchain-based IoT systems with a focus on performance evaluation purposes. This chapter aims to gather and analyse the requirements for the simulation tool based on perspective and feedback from subject-matter experts. A two-part study was conducted to gather the requirements for designing and implementing a blockchain-based IoT simulator. Firstly, a questionnaire was used to confirm the benefits of designing a simulator to analyse the performance of IoT integrated with blockchain. Secondly, participants were interviewed to understand the key

challenges they face with blockchain-based IoT during performance evaluation, along with features required for a simulator for such cases, and to gather insights into how blockchain can benefit IoT. This chapter paves the way for the next Chapter 5 to facilitate the design and implementation of the simulator.

The remainder of this chapter is organised as follows: Section 4.1 provides a brief introduction to this chapter and highlights the challenges it seeks to address. Section 4.2 explains the Research Question (RQ), highlights the chapter's contributions, and its relevance to the published paper. Section 4.3 presents the objectives of the study. The methods employed in this study are detailed in Section 4.4. The results are presented in Section 4.5 and then discussed in Section 4.6. Section 4.7 presents the study's proposals and the requirements specification for a blockchain-based IoT simulator. Section 4.8 concludes the chapter.

4.1 Introduction

The Internet of Things (IoT) constitutes a vast network of interconnected devices, including sensors, actuators, smart TVs, and smart cars. These devices can communicate and share data with one another or with users. Several issues are associated with the centralisation of most IoT architectures, as highlighted by Section 2.3. Several Blockchain features, such as those in Section 2.1.2, invite the utilisation of Blockchain to decentralised IoT architectures. However, IoT and Blockchain technologies are highly complex and heterogeneous, not to mention the integration of both (see Section 2.3.2 for further details). Simulators investigate a system's parameters and behaviour [145], which can be useful for complex systems that must be examined before actual deployment [146, 147]. Blockchain and IoT are good examples that can benefit from simulation because blockchain and IoT consist of various interconnected layers [148, 149]. Simulators can substantially reduce the financial costs needed to deploy real blockchain or IoT systems. Also, simulators allow investigation of a system's performance under different configuration setups. Nevertheless, the systematic mapping study, in Chapter 3, reveals that no existing simulator specifically considers the

combination of IoT and Blockchain for performance evaluation purposes. Therefore, this thesis ridges the gap by contributing a blockchain-based IoT simulator for performance evaluation purposes, which can benefit the industry and academia.

4.2 Research Questions, Contributions, and Relevance to Published Work

Section 4.2.1 provides a detailed explanation of the Research Question (RQ) and highlights the contributions associated with this chapter. Section 4.2.2 clarifies the relevance of this chapter to the published paper, as outlined by the publications listed in Section 1.4.

4.2.1 Research Question and Contribution

Research Question 2 (RQ2): Given the lack of existing simulation frameworks for evaluating the performance of Blockchain-based IoT ecosystems, what is required to bridge the gap? To answer this question, this thesis contributes a simulation framework for evaluating the performance of blockchain-based IoT ecosystems. The contribution of the simulation framework is the first in the literature to combine IoT and Blockchain in a unified simulation tool for performance evaluation purposes. This chapter specifically undertakes the first step for realising the simulator by gathering and analysing relevant requirements based on subject-matter experts. This step is essential for conducting the next chapter, Chapter 5, which, in turn, designs and builds the simulator based on the outcomes of this chapter.

4.2.2 Relevance of the Chapter to the Published Paper

This chapter is closely aligned with the corresponding publication [14], which explores the requirements for developing a simulator to assess the performance of IoT integrated with blockchain. It addresses the primary challenges encountered in blockchain-based IoT and examines how blockchain technology can enhance IoT systems. While it covers the same key topics and presents similar information as the publication, this chapter goes further by

expanding on the methodology with additional details. Moreover, it incorporates a thematic analysis of interview data and outlines specific requirements for designing and developing a simulation framework to evaluate the performance of blockchain-IoT integration.

4.3 Objectives

This chapter aims to obtain the opinions and perspectives of research participants regarding blockchain's potential contributions to IoT. For example, enabling IoT data transparency and security. Once the participants' thoughts are gathered, the proposed system's requirements and the required tools and mechanisms can be established. This process is described in relation to a number of objectives, as follows:

1. To gather the required information from experts in the field regarding:
    (a) The usage of IoT in our daily life.
    (b) The most commonly used blockchain types.
    (c) The IoT data that should be stored on the blockchain.
    (d) The consensus algorithms required for the simulator.
    (e) The users' needs as regards the blockchain log.
    (f) The possibility of using IoT nodes as blockchain nodes.
2. To provide analytical information regarding:
    (a) Participants' opinions about having an integrated blockchain IoT simulator.
    (b) Participants' opinions on modelling various types of blockchain in the simulator.
3. To design a simulator to validate the integrated blockchain IoT systems.

4.4 Method

This study employs a sequential explanatory design [150], starting with a questionnaire (quantitative) phase followed by an interview (qualitative) phase to explore the perspec-

tives of professionals and researchers on developing a blockchain-based IoT simulation framework. The following subsections outline the approaches and procedures used in the study. Section 4.4.1 details participants' identification, selection, and recruitment, along with a summary of their number and demographic characteristics. Section 4.4.2 describes the methods and procedures utilized for collecting and analyzing both questionnaire and interview data.

4.4.1 Participants

Participants were selected from a pool of individuals who met the criteria for involvement in this study through a multi-stage process detailed in the following subsections.

4.4.1.1 Identifying Potential Participants

The study targeted individuals with relevant expertise and experience in the Blockchain and IoT fields by identifying several key sources. These included professional academic networks like ResearchGate, specialized research groups focused on Blockchain and IoT, both within and outside the researcher's educational institution, and scholarly publications related to these fields. During this stage, we gathered contact information, including email addresses that facilitate subsequent communication and invitations to participate in the study.

4.4.1.2 Selection Criteria

After identifying potential participants in the previous step, we established specific selection criteria. These criteria included relevant experience for practical and professional work in blockchain and IoT and a review of their publications in these fields.

4.4.1.3 Recruitment Approach

1. **Online Questionnaire**: The selected participants were distributed via email to complete an online questionnaire. The email included an overview of the study's objectives, their expertise's importance, an estimated completion time, and a link to the question-

naire. Out of the 42 individuals invited, 25 completed the questionnaire. However, seven participants were excluded during the analysis stage due to their low familiarity with blockchain and IoT, as the study targeted participants with expertise. As a result, 18 of the 25 participants' responses were included in this study.

2. **Follow-up Interviews**: The selected participants for follow-up interviews received email invitations that outlined the study's purpose, emphasized the importance of their expertise in Blockchain and IoT, and specified the expected time commitment. Ten participants were chosen for follow-up interviews based on several criteria, including a high level of familiarity with the field as determined by quantitative analysis results, as well as demographic factors such as academic level, experience, and areas of interest. Participants' willingness and availability for interviews were also considered in the selection process. Then, follow-up emails were sent to confirm participation. Ultimately, six out of the ten participants who agreed to participate were interviewed.

4.4.1.4 Demographics

The participants' demographic, including their academic levels, professional backgrounds, and research interests and experiences, is summarized in Table 4.1.

Table 4.1 Participant Demographics

Participant	Academic Level	Professional Background	Interest	Experiences
1	PhD	Academia	Blockchain-based SLA in the context of IoT	4
2	PhD	Academia	IoV, Blockchain and IoT	4
3	PhD	Academia	IoT, Blockchain and AI	3
4	PhD	Academia	Data management and blockchain	2
5	Master	Industry	Remote health monitoring using IoT and blockchain	1
6	PhD	Academia	Blockchain, Formal modelling and AI	2
7	Master	Industry	Blockchain, supply chain management and IoT	4
8	PhD	Academia	Decentralized networks for IoT	1
9	PhD	Academia	Blockchain with IoT scalability	3
10	Master's	Industry	Blockchain for smart city	3
11	PhD	Academia	IoT and blockchain	3
12	PhD	Academia	Blockchain-based IoT	1
13	Master's	Industry	Blockchain, AI and IoT	3
14	PhD	Academia	Blockchain integration with AI	2
15	PhD	Academia	IoT and blockchain	2
16	Master	Industry	IoT security using blockchain	1
17	PhD	Academia	Blockchain, IoT and Cloud	3
18	PhD	Academia	Blockchain, IoT	4

4.4.2 Data Collection and Analysis Methods

This section describes the methods and procedures utilized for collecting and analyzing both Questionnaire 4.4.2.1 and Interviews 4.4.2.2.

4.4.2.1 Questionnaire

The questionnaire was designed based on the study's objectives outlined in section 4.3 and was subsequently administered via the SurveyMonkey platform. The questions were specifically crafted to explore participants' perceptions and attitudes towards various aspects of blockchain technology, with a particular focus on simulating blockchain-based IoT. The questionnaire comprised nine closed-ended questions, utilizing a Likert scale ranging from 1 ('strongly disagree') to 5 ('strongly agree').

Prior to distributing the questionnaire to the primary participants, a pilot test was conducted with a small sample group. This preliminary step aimed to assess the questionnaire's clarity, relevance, and ease of completion. Based on the feedback from the pilot test, refinements were made to ensure the questionnaire was well-suited to the target participants, as discussed in section 4.4.1.

Following data collection, a statistical analysis was carried out using the Statistical Package for the Social Sciences (SPSS). This analysis included a reliability test through Cronbach's Alpha, which yielded a value of 0.796, indicating a high level of internal consistency among the questions.

Furthermore, the findings were presented through descriptive analysis, with graphical representations employed to illustrate response distributions and highlight key findings as provided in section 4.5.1.

4.4.2.2 Interviews

The 6Ps framework by Oates [151] was utilized to design and implement the follow-up interview, ensuring the quality and consistency of the process throughout the study. This framework encompasses six key elements: purpose, paradigm, process, participants, product, and presentation. The procedure for applying this framework is detailed below, with a summary in Table 4.2.

Table 4.2 Summary of the 6Ps Framework

6Ps	Scope	Question	Procedure
Purpose	**Motivation**	*Why* was this interview conducted?	- Gathered expert insights, challenges, and requirements for a Blockchain-IoT simulator
Paradigm	**Data**	*What* method was used?	-Qualitative, -Semi-structured interview
Process	**Function**	*How* was the data analyzed?	- Thematic analysis
Participants	**People**	*Who* was involved?	- Participants, Blockchain, and IoT
Product	**Results**	*What* were the outcomes?	- Challenges, Benefits, Requirements, Recommendations
Presentations	**Place & Time**	*Where* and *when* were the interviews conducted?	- Conducted via Zoom, Scheduled based on participants' availability, Approximately 60 minutes

1. **Purpose**: This study aimed to collect information concerning their opinions on using IoT-based Blockchain, as well as comprehending their requirements for a simulation environment to assess blockchain-based IoT. The participants were asked to answer the following questions:

   (a) **What are the major challenges you face when dealing with blockchain-based IoT for any evaluation purposes?**

   (b) **Which features make blockchain suitable for the IoT?**

   (c) **What are the anticipated outcomes of utilising blockchain within the IoT?**

2. **Paradigm**: The study adopted a qualitative research method to explore the insights of experts in the field of Blockchain and IoT technologies to identify the key challenges, benefits, and specific requirements for designing a simulator to evaluate integrated Blockchain-IoT systems' performance. It relied on semi-structured interviews, which were well-suited for capturing viewpoints and detailed explanations that quantitative methods may overlook.

3. **Process**: The research employed a semi-structured interview methodology to gather qualitative data. The interview guide was developed based on the study's objectives,

including key questions that addressed challenges, opportunities, and requirements for developing a simulation framework related to Blockchain and IoT.

Each interview began with an introduction explaining the study's purpose. The researcher then followed the prepared guide (e.g., motivation scenario, challenges of integration, etc.) to allow flexibility for participants to elaborate on relevant topics. Interviews were recorded with participants' consent for accurate transcription and analysis.

After each interview, the recordings were transcribed verbatim, and thematic analysis [152] was applied to identify significant patterns and themes. The data coding process involved open coding, followed by axial coding to connect related themes, as explained in Section 4.4.2.3.

4. **Participants**: The study included six participants with experience in Blockchain and IoT technologies. The participants were chosen based on the criteria, as illustrated in 4.4.1.

5. **Product**: The research produced several key outcomes, providing valuable insights into integrating Blockchain and IoT technologies. The primary products of the study included identifying challenges in using Blockchain with IoT, requirements for designing a simulation environment, and recommendations, as presented in section 4.7.

6. **Presentations**: The interviews were conducted via the Zoom platform, allowing experts in different geographic regions to participate remotely. The sessions were scheduled to accommodate the participants' time zones at various times based on their availability. Each interview lasted approximately 60 minutes, providing sufficient time for in-depth discussions.

4.4.2.3 Thematic Analysis Procedure

The thematic analysis was conducted following a systematic process that involved several key steps: familiarization with the data, generating initial codes, searching for themes, reviewing themes, defining and naming themes, and finally, writing the report.

1. **Familiarizing with the Data**

   The first step in the thematic analysis involved familiarizing ourselves with the data gathered from the previous step 4.4.2.2. This process began with repeated reading of the interview transcripts and the questionnaire responses. The aim was to immerse in the data to develop a deep understanding of the content and context. During this phase, we took notes on initial thoughts and potential patterns that appeared significant.

2. **Generating Initial Codes**

   Once familiarized with the data, the next step was to generate initial codes. Coding involves identifying and labelling segments of the data that are relevant to the research questions. Each code represents a specific feature of the data that appeared interesting or meaningful. In this study, coding was applied systematically across the entire data set with cross-references to the data sources and identifiers, as shown in Table 4.3.

Table 4.3 Summary of Generating Initial Codes.

RQ	***Initial Round***	***Final Round***	**Quote of Participants Response**
	Initial Code	**Final Code**	
RQ1	**C1**: Difficulty evaluating performance due to cost	**C1**: Costly real-world performance measurement	*"The obstacle lies in investigating the performance and cost of these technologies."*
	C2: Difficulty in monitoring system performance	**C2**: System monitoring difficulty	*"...is the difficulty of monitoring systems' performance."*
	C3: Heterogeneity and mobility of IoT Devices	**C3**: System heterogeneity and mobility	*"..blockchain-based IoT challenges is system evaluation because of the heterogeneity and mobility of IoT devices."*
	C4: Difficulty obtaining performance statistics	**C4**: Difficulty obtaining performance statistics	*"...challenge is how to obtain various statistics about the system..."*
	C5: Complexity of Blockchain and IoT	**C5**: Complexity measuring performance	*"...blockchain and the IoT, it is difficult to measure performance due to the complexity of both technologies. "*

Continued on next page

Table 4.3 – continued from previous page

	C6: Focus on either IoT or blockchain	**C6**: Single-focus simulators	*"...there are many proposed simulators for Blockchain and IoT in the literature; however, each simulator either focuses on IoT or blockchain."*
	C7: Lack of features supported in Blockchain or IoT simulator	**C7**: Lack of feature coverage	*"... use a cloud simulator to evaluate the system. Having a Blockchain simulator with IoT features... "*
	C8: Make decisions based on exploring the performances	**C8**: Decision-making	*"The simulation metrics give me an indicator about the proposed system to make decisions."*
	C9: Simulate IoT with Blockchain	**C9**: Multi-Discipline simulation	*"... having a multi-discipline simulator..."*
	C10: Test from diverse aspects	**C10**: Evaluation from multiple aspects	*"... assess the system from different viewpoints..."*
	C11: Facilitate the evaluation process	**C11**: Facilitate the evaluation process	*"Blockchain simulator with IoT features that can track every transaction and system throughput will ease my tasks."*
	C12: Ability to customize IoT and Blockchain configurations	**C12**: Configuration flexibility	*"... configure the number of IoT devices and protocols used while at the same time determining the size of transactions, either for blockchain or the IoT (end to end)."*
	C13: Transaction latency	**C13**: Performance metrics and statistics	*" Simulator supports more than one measure, such as latency, throughput, total time, along with the number of blocks created... ."* *"...like the number of generated transactions, number of blocks and time of confirmation for block and transaction."*
	C14: Throughput		
	C15: Number of transactions		
	C16: Number of blocks		
	C17: Confirmation times for transaction and block		
RQ2	**C18**: Secure management of data	**C14**: Handling of sensitive and important Data	*"Healthcare data is of high importance and needs to be securely handled. "* *"Not all IoT data are of high importance, but there is still a need to secure sensitive data and enhance privacy. "*
	C19: Data privacy without third party	**C15**: Data privacy without third party involvement	*".....the danger of breaking data privacy and policies.....blockchain's traceability can help in these situations."*
	C20: No third-party involved	**C16**: Elimination of Third-Party management	*"Blockchain is a strong fit... because of its features (for example, decentralization) that dispense a third party to manage data."*
	C21: Single Points of Failure	**C17**: Preventing Single Points of Failure	*" Decentralization... can prevent a single point of failure and bottlenecks from occurring."*
	C22: Immutability and Authenticity	**C18**: Immutability and Authenticity	*"IoT data can be immutable and distributed... ensure the data's authenticity and that it will never be tampered with. "*
	C23: Data transfer	**C19**: Reliable data transfer	*"Blockchain benefits the IoT by ensuring reliable data transfer."*
	C24:Data storage	**C20**: Selective data storage	*"Data storage is a crucial metric... I prefer storing only the most important data."*

Continued on next page

Table 4.3 – continued from previous page

	C25: Processing	**C21**: Limited processing capabilities	*" Due to the limited processing capabilities of IoT devices, third-party service providers are generally used to process additional data."*
	C26:Traceability	**C22**: Traceability	*"Blockchain's traceability can help in these situations."*
RQ3	**C27**: Device management and access control of IoT components	**C23**: Identity management and access control	*" .....large number of devices expected to be connected.... blockchain would alleviate security issues...identity management and access to the IoT ..."* *"....Due to the limited processing capabilities of IoT devices, third-party service providers are generally used to process additional data.."*
	C28:Privacy	**C24**: Privacy concerns	*" ....IoT issues related to privacy."*
	C29: Data generated by the trusted device in Blockchain	**C25**: Enhanced data integrity and privacy	*"..every single device can be identified using a permissioned blockchain network... ensuring immutability... data is generated by an identified device (trusted). "* *"..blockchain is a promising choice when it comes to ensuring privacy. "*
	C30: Improved IoT security and trust	**C26**: Enhanced security and trust	*"...blockchain would alleviate security issues... identity management and access to the IoT should be more secure and trusted. "* *"...blockchain would provide the IoT developers with more secure solutions due to its features."*
	C31: Optimized IoT access policies and control mechanisms	**C27**: Enhanced access control and policy definition	*"..blockchain can define a set of policies needed to control IoT data access."* *"..using a reliable tool for controlling data access."*
	C32: Traceability and transparency	**C28**: Enhanced traceability and transparency	*"...blockchain allows for the traceability of all recorded data stored...transparency in supply chains.. shared with ...stakeholders"*

3. **Searching for Themes**

   Following the coding process, the next step was to search for themes. A theme captures something important about the data in relation to the research question and represents a patterned response or meaning within the data set. In this study, related codes were grouped together to form broader themes, as presented in Table 4.4. For example, codes related to "Scalability Issues" and "Data Management Challenges" were grouped under the theme "Challenges in Blockchain-IoT Integration."

Table 4.4 Summary of Themes Categorization.

RQ	Code	Theme Category
RQ1	**C1**: Costly real-world performance measurement	**Challenges**
	C2: System monitoring difficulty	
	C3: System heterogeneity and mobility	
	C4: Difficulty obtaining performance statistics	
	C5: Complexity measuring performance	
	C6: Single-focus simulators	**Simulators' Limits**
	C7: Lack of feature coverage	
	C8: Decision-making	**Need for a Simulator**
	C9: Multi-Discipline simulation	
	C10: Evaluation from multiple aspects	
	C11: Facilitate the evaluation process	
	C12: Configuration flexibility	**Features Needed**
	C13: Performance metrics and statistics	

RQ	Code	Theme Category
RQ2	**C14**: Handling of Sensitive and Important Data	**Security and Privacy**
	C15: Data privacy without third party involvement	
	C16: Elimination of Third-Party Management	**Decentralization**
	C17: Preventing Single Points of Failure	
	C18: Immutability and Authenticity	**Data Integrity and Reliability**
	C19: Reliable Data Transfer	
	C20: Selective Data Storage	**Scalability and Efficiency**
	C21: Limited Processing Capabilities	
	C22: Traceability	**Traceability and Transparency**

RQ	Code	Theme Category
RQ3	**C23**: Identity management and access control	**IoT Security Challenges**
	C24: Privacy concerns	
	C25: Enhanced data integrity and privacy	**Blockchain Anticipated Outcomes as a Mitigation**
	C26: Enhanced security and trust	
	C27: Enhanced access control and policy definition	
	C28: Enhanced traceability and transparency	

4. **Reviewing Themes**

   Once the initial themes were identified, they were reviewed to ensure they accurately represented the data. This step involved two levels of review: the first level focused

on the coherence of data within each theme, and the second level involved assessing the validity of the themes in relation to the entire data set. Some themes were merged, refined, or discarded based on their relevance and clarity.

5. **Defining and Naming Themes**

   After reviewing the themes, the next step was to define and name them. Each theme was clearly defined to articulate its essence and scope. The themes were also given descriptive names that succinctly captured their core meaning, as shown in Table 4.5. For example, the theme "Challenges of Evaluating Blockchain-based IoT " was defined as encompassing all discussions related to the difficulties of evaluating blockchain with IoT, including system monitoring difficulty and complexity measuring performance.

Table 4.5 Summary of Defining Themes.

Code	Theme Category	Theme	Theme Description
C1: Costly real-world performance measurement	**Challenges**	**Challenges of Evaluating Blockchain-based IoT**	This theme explores participants' perspectives on the challenges and limitations of evaluating the performance of blockchain-based IoT.
C2: System monitoring difficulty			
C3: System heterogeneity and mobility			
C4: Difficulty obtaining performance statistics			
C5: Complexity measuring performance			
C6: Single-focus simulators	**Simulators' Limits**		
C7: Lack of feature coverage			
C8: Decision-making	**Need for a Simulator**	**Need and Core Features of a Simulator for Blockchain-based IoT Performance Evaluation**	This theme explores whether participants are interested in a simulator capable of evaluating Blockchain-based IoT performance and identifies the key features and capabilities that such a simulator should encompass.
C9: Multi-Discipline simulation			
C10: Evaluation from multiple aspects			
C11: Facilitate the evaluation process			
C12: Configuration flexibility	**Features Needed**		
C13: Performance metrics and statistics			

Continued on next page

Table 4.5 – continued from previous page

C14: Handling of sensitive and important data	**Security and Privacy**	**Key Features Enhancing Blockchain Integration with IoT**	This theme explores the specific features of blockchain that make it particularly well-suited for integration with IoT systems.
C15: Data privacy without third party involvement			
C16: Elimination of Third-Party management	**Decentralization**		
C17: Preventing Single Points of Failure			
C18: Immutability and authenticity	**Data Integrity and Reliability**		
C19: Reliable data transfer			
C20: Selective data storage	**Scalability and Efficiency**		
C21: Limited processing capabilities			
C22: Traceability	**Traceability and Transparency**		
C23: Identity management and access control	**IoT Security Challenges**	**Feasibility of Blockchain Integration with IoT for Addressing Security Challenges and Mitigating Potential Risks**	This theme explores the practicality and viability of using blockchain to address security challenges within IoT systems.
C24: Privacy concerns			
C25: Enhanced data integrity and privacy	**Blockchain Anticipated Outcomes as a Mitigation**		
C26: Enhanced security and trust			
C27: Enhanced access control and policy definition			
C28: Enhanced traceability and transparency			

6. **Producing the Report**

   The final step in the thematic analysis was writing the report. The report presented the themes in a structure supported by relevant data extracts. Each theme was discussed in detail, with a focus on how it addressed the research questions and what it revealed about the challenges, benefits, and anticipated outcomes of integrating blockchain and IoT technologies, such as the mitigation of IoT security issues, as illustrated in section 4.5.2.

4.5 Findings

The study's findings are organized as follows: Section 4.5.1 presents the questionnaire results, and Section 4.5.2 presents the interview results.

4.5.1 Questionnaire

The questionnaire for this study was designed following the structured procedures outlined in Section 4.4.2.1. Moreover, we crafted the questions based on the study's objectives 4.3, with each question clearly aligned with a predefined objective, as shown in Table 4.6. This resulted in a questionnaire with nine closed-ended questions formatted on a Likert scale. A total of 25 selected participants initially responded to the questionnaire, which was distributed to the selected during the analysis using the approach described in Section 4.4.1. However, we found that seven out of the 25 participants had low or moderately low familiarity with blockchain and IoT technology during the analysis of participants' responses. Given the study's focus on participants with relevant experience, we excluded these individuals from the final analysis. Consequently, the final analysis focuses on the responses of 18 participants who demonstrated moderate to high familiarity with both IoT and blockchain.

Table 4.6 Matching the questionnaire questions to the predefined objectives.

Question	**Objective 1**						**Objective 2**		**Objective 3**
	a	b	c	d	e	f	a	b	
Q1	✓								
Q2		✓							
Q3	✓	✓							
Q4							✓		✓
Q5			✓						✓
Q6				✓					✓
Q7					✓				✓
Q8						✓			✓
Q9								✓	✓

The questionnaire begins by asking questions to determine the participants' familiarity with the IoT. Specifically, the participants were asked, *"To what extent are you familiar*

with IoT?" In this case, 18 answers presented in Figure 4.1 were received from participants regarding their familiarity. The figure shows that the majority (eight participants, 44%) are moderately aware of the IoT, while five participants (28%) have moderately high familiarity with the IoT. Moreover, five participants (28%) are highly aware of the IoT. Accordingly, the selected participants are a good fit because the majority (moderate and higher) are aware of the IoT.

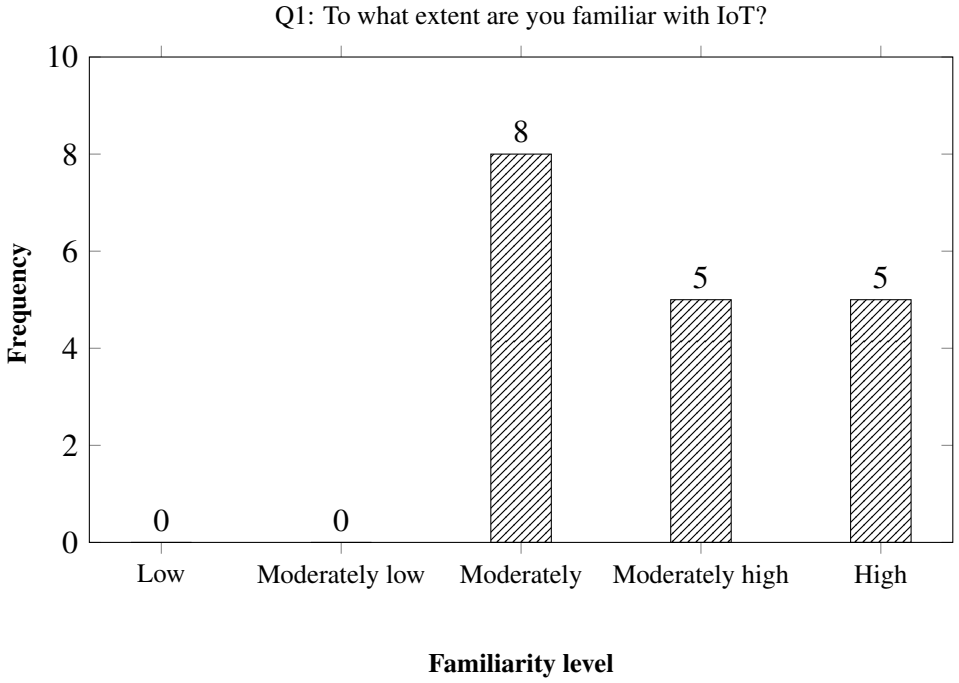


Fig. 4.1 Participants' familiarity with the IoT.

As we are discussing two technologies, we also examined their familiarity with blockchain to feel more confident about the participants' answers. Thus, the participants were asked, *"To what extent are you familiar with blockchain?"* In this instance, 18 responses, presented in Figure 4.2, were received from the respective participants regarding their familiarity. The figure suggests that the majority (six participants, 33%) possess moderately high awareness of blockchain, while seven participants (39%) are very familiar with blockchain. Moreover, five participants (28%) are moderately aware of blockchain.

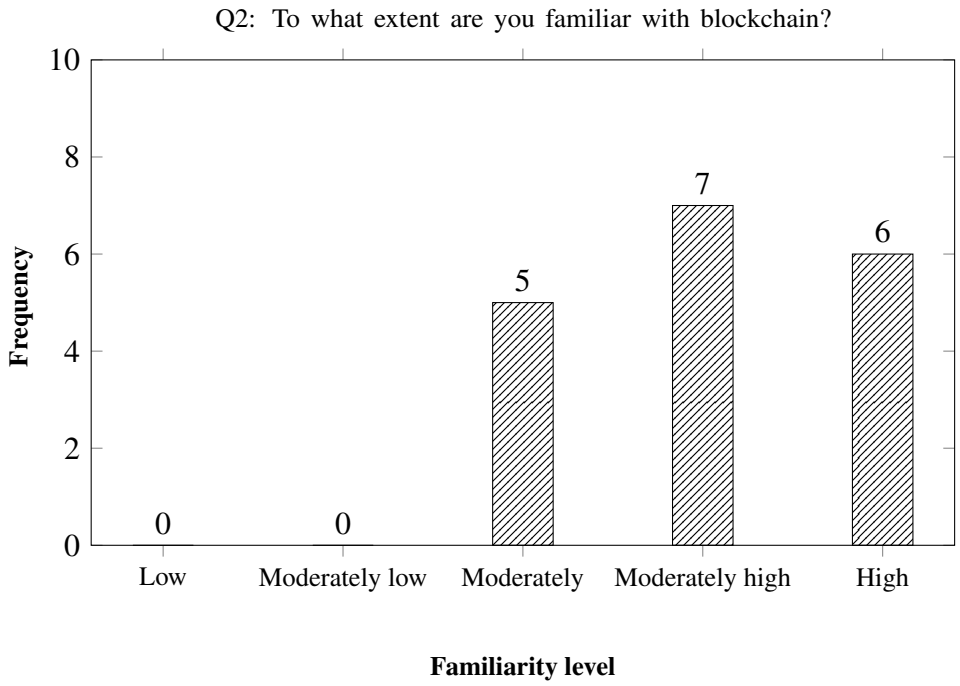


Fig. 4.2 Participants' familiarity with Blockchain.

Similar to the participants' familiarity with the IoT, the selected participants are a good fit, given that the majority are aware of blockchain. Subsequently, the participants were asked *"if they believe that there will be an expansion of blockchain with IoT in the future."* All 18 participants responded to this question, with their responses presented in Figure 4.3. It was established that the majority (eight participants, 45%) highly agreed with this point. Moreover, six participants (33%) expressed moderately high agreement with the idea. A minority (four, 22%) either moderately or completely disagreed.

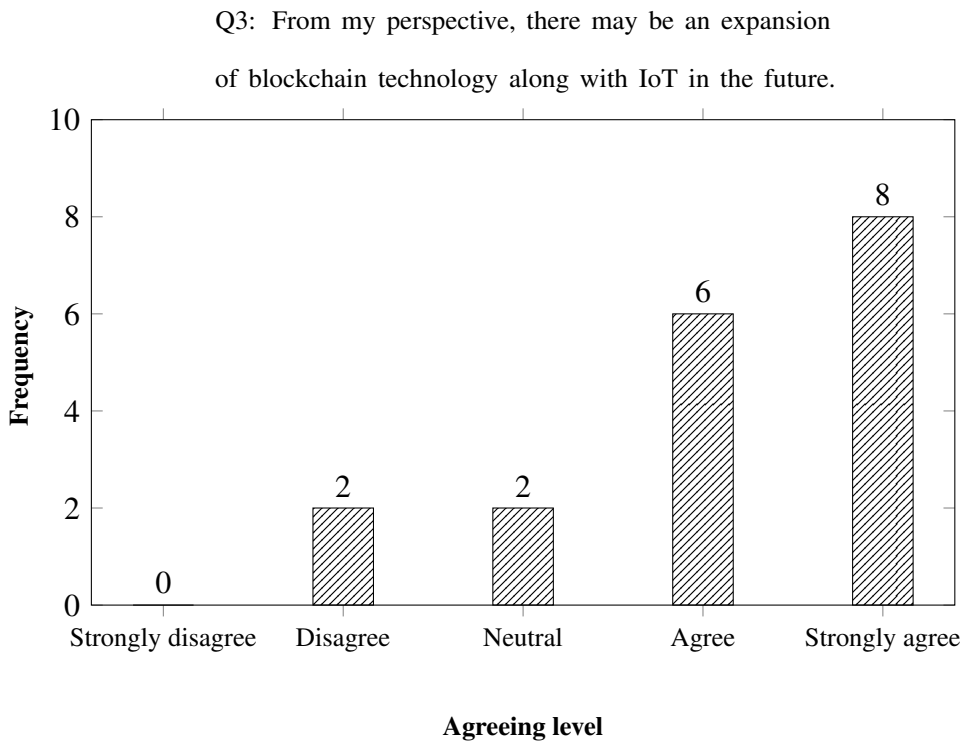


Fig. 4.3 Participants' thoughts about the IoT's integration with blockchain.

Following this, participants were asked, *"What are your thoughts regarding the need to have an IoT blockchain simulator for helping developers with adjusting the system's configurations?"* All 18 participants provided their responses, summarised in Figure 4.4. As is apparent from this figure, nine participants (50%) strongly agreed with this notion, while eight participants (44%) agreed with this concept. Only one participant (6%) expressed neutrality, and none disagreed.

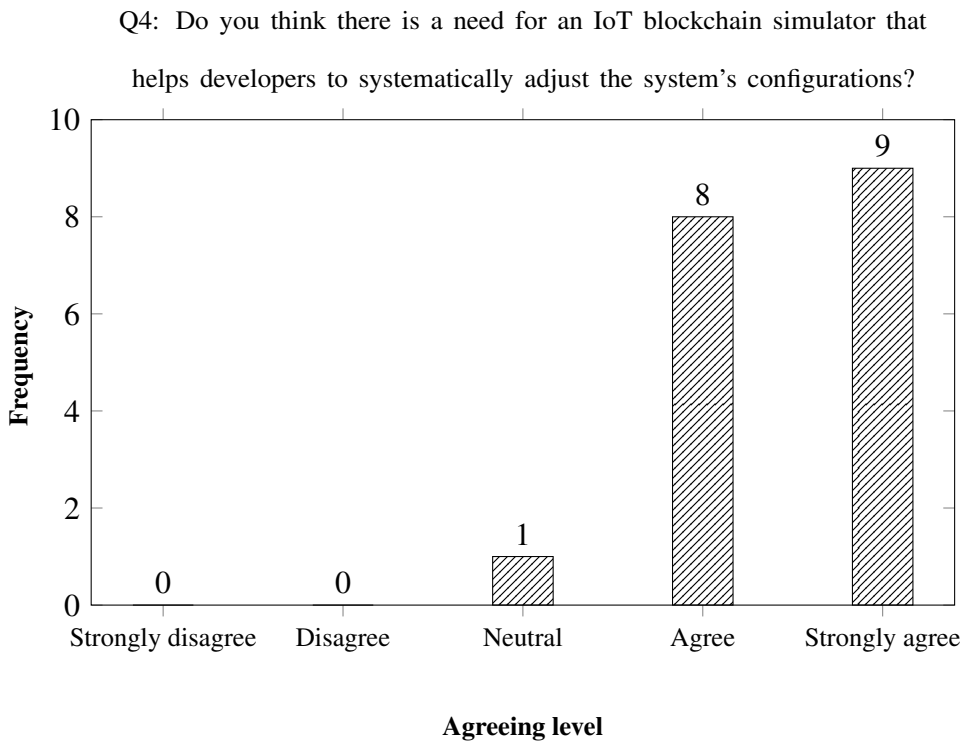


Fig. 4.4 Participants' thoughts about having an integrated IoT blockchain simulator.

Given that the participants are domain experts, we took the opportunity to gather their perspectives on storing IoT data in blockchain. The participants were asked, "Do you agree that all IoT data should be stored in the blockchain?" The responses, as shown in Figure 4.5, reveal that the majority disagreed with this statement, with 13 (72%) participants either disagreeing or strongly disagreeing. This outcome may reflect the varying scenarios in which IoT and blockchain are utilized. Additionally, three participants (17%) expressed neutrality, while two participants (11%) agreed or strongly agreed with the statement.

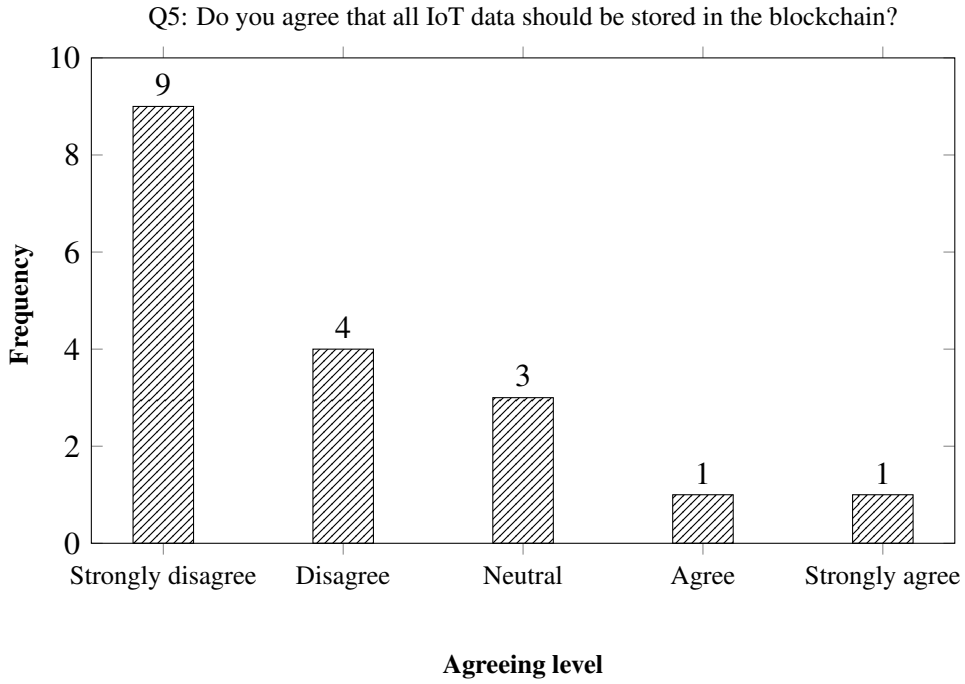


Fig. 4.5 Participants' thoughts about storing all of the IoT data in the blockchain.

Consensus algorithms are of considerable importance in blockchain because they are used to reach a common agreement (consensus) on the current state of the ledger data. They also enable unknown peers to be trusted in a distributed computing environment. Thus, there is a need to establish participants' needs relating to this. Accordingly, the participants were asked, *"What are your thoughts on having multiple consensus algorithms in the simulator?"* The participants' responses to this question are summarised in Figure 4.6. Considering the data more closely, it is apparent that the majority (eight participants, 44%) agreed with this notion. Furthermore, five participants (28%) strongly agreed with the idea. Meanwhile, two participants (11%) were moderately supportive, and three participants (17%) either disagreed or strongly disagreed.

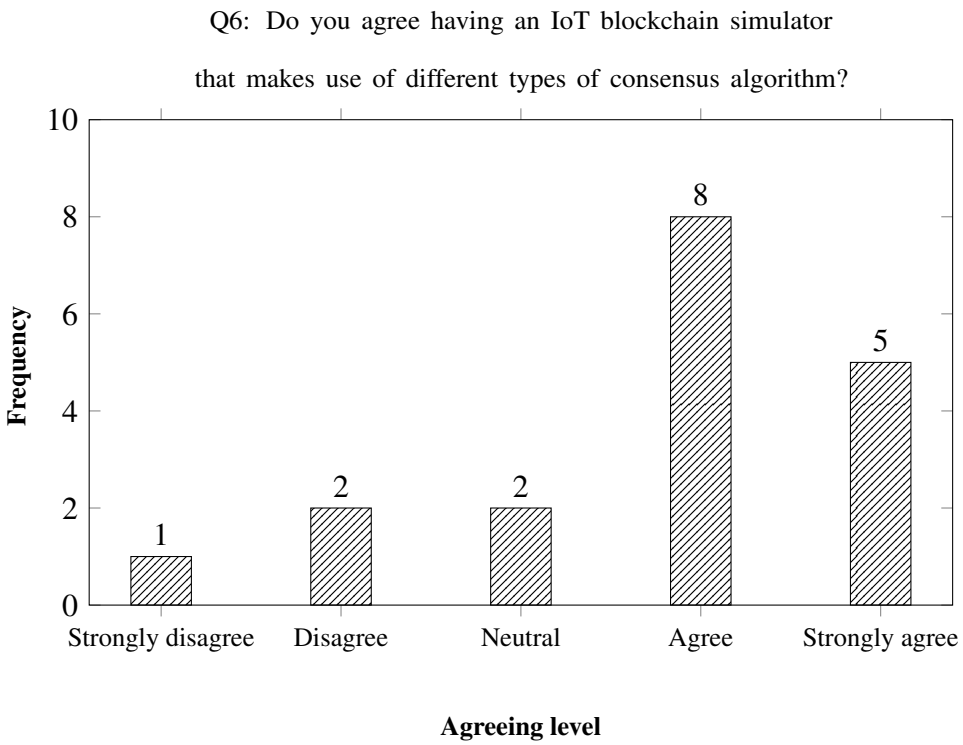


Fig. 4.6 Participants' thoughts about having multiple consensus algorithms in the simulator.

Considering blockchain in greater depth, it is essential to determine the participants' perspectives regarding investigating the log. This is crucial because it provides the opportunity to compute system latency and throughput. Accordingly, the participants were asked for their opinions concerning investigating the log file. The participants' responses to this question are presented in Figure 4.7. The significant point is that the majority (12 participants) either strongly agreed (22%) or agreed (44%) with this idea. Additionally, four participants (23%) remained neutral on the statement. Meanwhile, two participants (11.11%) expressed varying levels of disagreement.

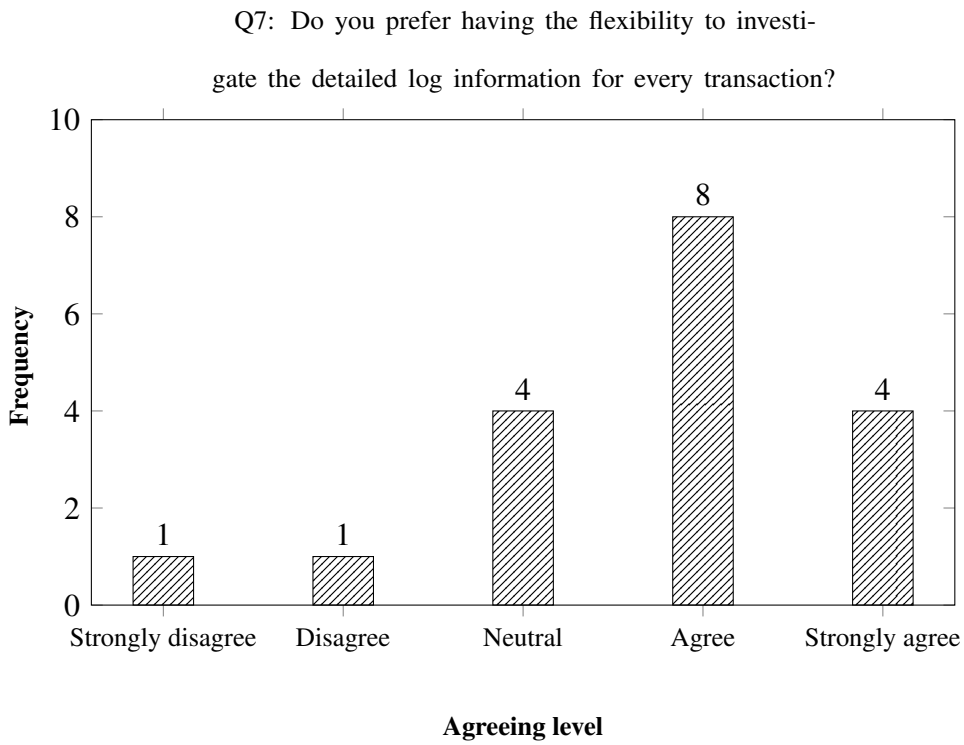


Fig. 4.7 Participants' thoughts about the ability to investigate the log.

Subsequently, the participants were asked about using IoT devices as blockchain nodes. Their responses, as shown in Figure 4.8, reflect an overall positive attitude toward this idea. Ultimately, most participants either strongly agreed (seven participants, 39%) or agreed (six, 33%) with the statement. In contrast, four participants (22%) either strongly disagreed or disagreed with this notion. Finally, one participant (6%) expressed neutrality regarding this notion.

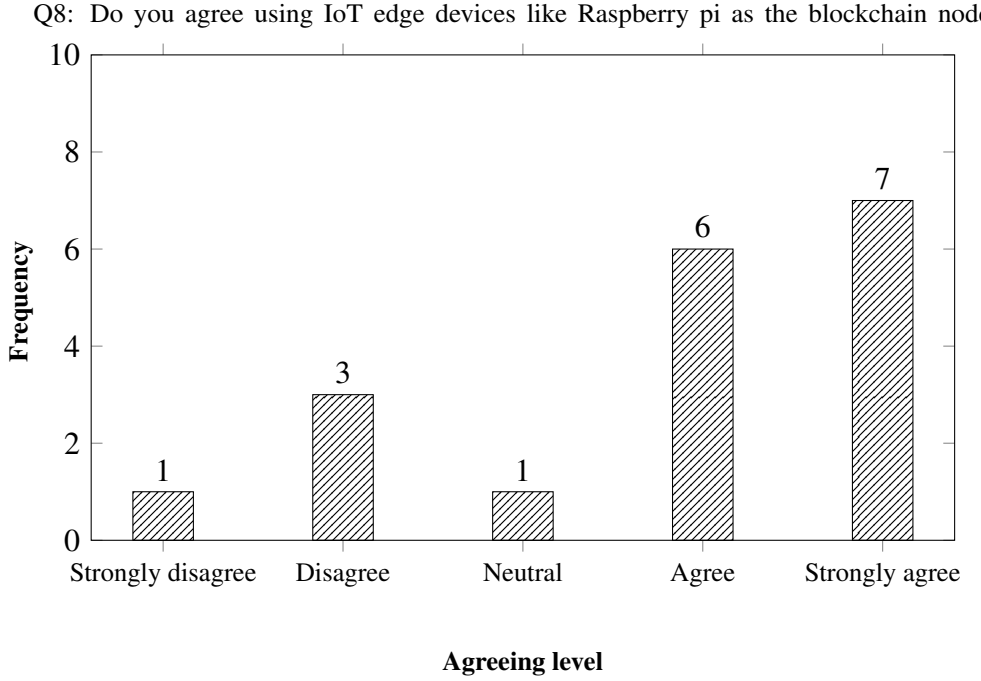


Fig. 4.8 Participants' thoughts about using IoT edge devices as blockchain nodes.

Finally, given that there are numerous types of blockchain, there is a need to comprehend if it is essential to have a simulator that can model the diverse types. Accordingly, the participants were asked about this, and their responses to this question are presented in Figure 4.9. According to the participants' perspectives, the majority (nine participants, 50%) are neutral towards this. However, three participants (17%) agreed, while two participants (11%) strongly agreed. Meanwhile, four participants (22%) either strongly disagreed or disagreed with this notion.

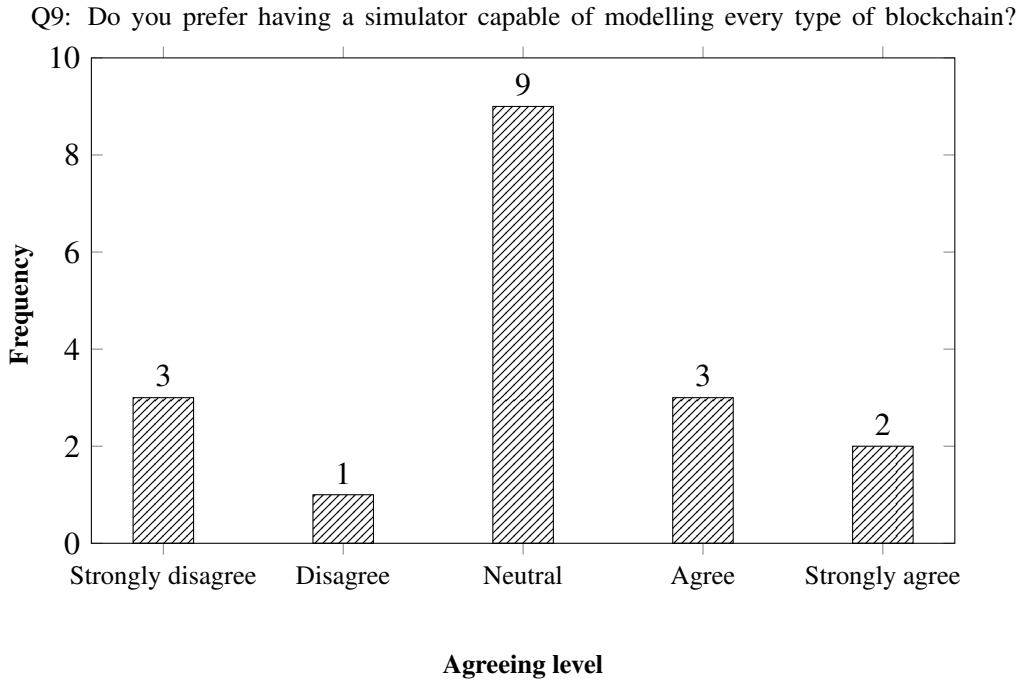


Fig. 4.9 Participants' thoughts about modelling different blockchain types in the simulator.

4.5.2 Interviews

The interviews were structured using Oates's 6Ps framework to ensure consistency and quality in data collection. The framework's elements(e.g. Purpose, Paradigm, Process, Participants, Product, and Presentation) guide the study, as outlined in Section 4.4.2.2. We analysed the data collected using a structured thematic analysis approach, which involved familiarisation with the data, coding, theme identification, and thematic refinement, as detailed in Section 4.4.2.3.

Overview of Themes

The thematic analysis revealed four primary themes that encapsulate the core challenges, needs, and opportunities within the domain of blockchain-based IoT systems and their feasibility, as depicted in Figure 4.10.

The first theme, *Challenges of Evaluating Blockchain-based IoT*, explores the various obstacles practitioners face when assessing the performance of these integrated systems,

including cost, complexity, difficulties in monitoring and collecting data for evaluation purposes and the shortcomings of current simulators, mainly their narrow focus and lack of features necessary for evaluating both blockchain and IoT together. The second theme, *Need and Core Features of a Simulator for Blockchain-based IoT Performance Evaluation*, emphasizes the necessity of advanced simulation tools to support multi-disciplinary evaluations, facilitate decision-making, and provide flexible configurations and detailed performance metrics. The third theme, *Key Features Enhancing Blockchain Integration with IoT*, delves into the essential characteristics required for successful integration, such as security, privacy, decentralization, data integrity, and scalability. Finally, the fifth theme, *Feasibility of Blockchain Integration with IoT for Addressing Security Challenges and Mitigating Potential Risks*, assesses how blockchain can address IoT security challenges, improve efficiency, and mitigate risks through features like enhanced data integrity, decentralized management, and transparent traceability.

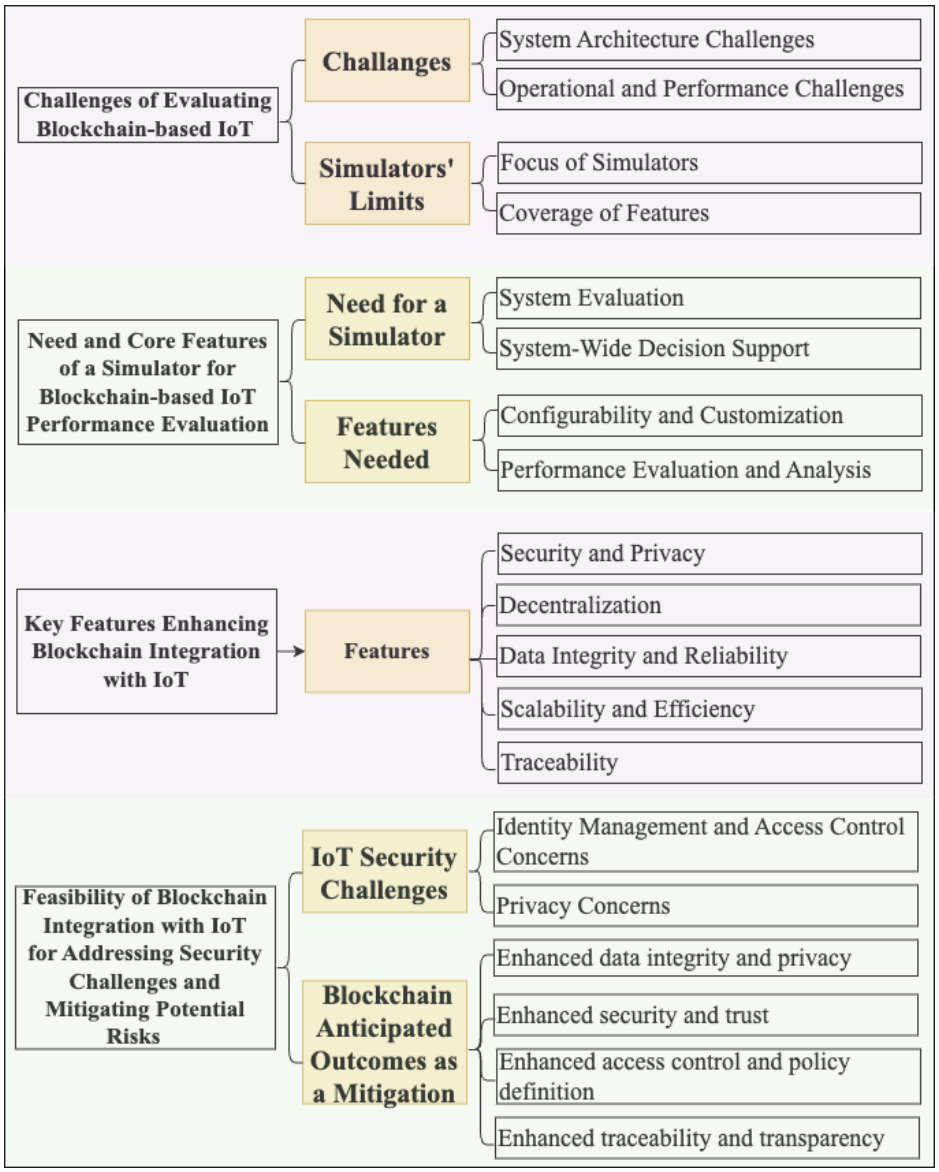


Fig. 4.10 Overview of Key Themes Identified in the Analysis.

4.5.2.1 Theme 1: Challenges of Evaluating Blockchain-based IoT

This theme captures participants' views on the complexities and difficulties in evaluating the performance of blockchain-based IoT systems and the limitations of existing simulators.

- **Challenges**

> *This sub-theme delves into the various challenges associated with assessing the performance of integrating blockchain technology with IoT. The challenges are categorized according to system architectural challenges and operational performance challenges.*

  - **System Architecture Challenges**: Participants highlighted the difficulty of monitoring blockchain-based IoT systems due to their decentralized and distributed nature. Data is generated and processed across numerous devices, often in real-time, complicating data collection and performance evaluation. As one participant noted, *"The difficulty lies in monitoring systems' performance due to decentralization"*.

    The heterogeneity and mobility of IoT devices add another layer of complexity. The diverse range of devices, each with varying capabilities and interconnectivity across different networks makes performance evaluation even more challenging. A participant remarked, *"Evaluating system performance is difficult because of the heterogeneity and mobility of IoT devices"*.

  - **Operational and Performance Challenges**: Participants frequently cited the high costs associated with real-world performance measurements. Setting up and maintaining a blockchain-based IoT system is resource-intensive, both financially and in terms of the specialized hardware and software required. As one participant explained, *"The challenge lies in investigating the performance and cost of these technologies"*.

    Another challenge is obtaining performance statistics. The decentralized nature of blockchain-based IoT systems complicates data collection, making it difficult to gather the necessary statistics for thorough evaluations. One participant stated, *"The challenge is obtaining various statistics about the system"*.

    The inherent complexity of measuring performance in these systems was also emphasized. The interactions between blockchain protocols and IoT devices

involve multiple factors that influence system performance, making the evaluation process difficult. A participant explained, *"It's difficult to measure performance due to the complexity of both technologies"*.

- **Simulators' Limits**

> *This sub-theme highlights participants' perspectives on the limitations of current simulators in evaluating blockchain and IoT integration. These limitations are categorized according to the simulator's objective (a.k.a. focus) and the range of supporting features.*

  - **Focus of Simulators**: Participants noted that many current simulators are designed with a narrow focus, either on blockchain or IoT, but not both. This limitation prevents a holistic evaluation of integrated systems, as these simulators cannot fully capture the interactions between the two technologies. One participant commented, *"Many simulators focus on either IoT or blockchain, but not both"*.
  - **Coverage of Features**: Participants also pointed out that existing simulators often lack the necessary features to evaluate all aspects of blockchain-IoT integration effectively. As one participant suggested, *"A simulator with both blockchain and IoT features is needed to evaluate the system comprehensively"*.

4.5.2.2 Theme 2: Need and Core Features of a Simulator for Blockchain-based IoT Performance Evaluation

Given the substantial challenges in evaluating blockchain-IoT systems, as discussed in Theme 1 4.5.2.1, this theme explores whether needed for a simulator is capable of addressing the challenges identified in the previous theme and also identifies the key features and capabilities that such a simulator should encompass.

- **Need for a Simulator**

> *This sub-theme highlights participants' perspectives on the importance of utilizing simulators during the evaluation phase, particularly in the context of Blockchain and IoT. These insights are categorized according to system evaluation and system-wide decision support.*

  - **System Evaluation**: Participants underscored the challenge of evaluating system performance across different scenarios, emphasizing that a simulator could provide valuable insights by allowing for evaluations from multiple perspectives. One participant explained, *"A simulator helps assess the system from different viewpoints, which is essential for understanding its performance"*.

    Additionally, evaluating Blockchain-based IoT applications in the real world is often resource-intensive and time-consuming. Thus, simulators can overcome challenges by facilitating the evaluation process through modelling and simulating Blockchain-based IoT. As noted by one participant, *"Blockchain simulator with IoT features that can track every transaction and system throughput will ease my tasks"*.

  - **System-Wide Decision Support**: Participants also highlighted the limitations of existing tools in capturing the complex dynamics of blockchain-IoT systems. They emphasized the need for a simulator that can model and simulate these integrations, facilitating more informed decision-making. One participant remarked, *"The simulation metrics give me an indicator about the proposed system to make decisions"*.

    The participants also pointed out that current simulators are often limited in scope, focusing on either blockchain or IoT, but not both. As one participant noted, *"there are many proposed simulators for Blockchain and IoT in the literature; however, each simulator either focuses on IoT or blockchain"* and another participant remarked, *"having a multi-discipline simulator"*.

- **Features Needed**

> *This sub-theme delves into the specific features and requirements that participants believe are essential for a simulator. These features and requirements are categorized according to configurability and customization in the simulator and features supported by the simulator.*

  - **Configurability and Customization**: Participants emphasized the necessity of having a flexible and configurable simulator. This flexibility allows for the simulation of specific scenarios, making performance evaluations more accurate and meaningful. As one participant mentioned,*"configure the number of IoT devices and protocols used while at the same time determining the size of transactions, either for blockchain or the IoT (end to end)"*.
  - **Features supported by the simulator**: Participants highlighted the importance of a simulator that can generate performance metrics for understanding system behaviour under various conditions. As one participant explained,*"The simulator should support multiple measures, such as latency, throughput, total time, and the number of blocks created"* and another participant noted,*"like the number of generated transactions, number of blocks and time of confirmation for block and transaction"*.

4.5.2.3 Theme 3: Key Features Enhancing Blockchain Integration with IoT

This theme explores the specific features of blockchain that make it particularly well-suited for integration with IoT systems.

- **Features**

> *This sub-theme highlights the primary attributes of blockchain that facilitate its integration with IoT, such as security through cryptography, decentralization to eliminate single points of failure, and immutable data to ensure integrity and trust.*

  - **Security and Privacy** : Participants consistently highlighted security and privacy as key reasons for integrating blockchain with IoT. Blockchain's decentralized structure enables secure management of sensitive IoT data without relying on third parties. As one participant noted, "Healthcare data is of high importance and needs to be securely handled", and another participant remarked,*"Blockchain is a strong fit for IoT because of its features (for example, decentralization) that dispense a third party to manage data"*.
  - **Decentralization**: Participants also emphasized decentralization, noting that it reduces reliance on centralized entities and prevents potential vulnerabilities. One participant remarked,*"Decentralization can prevent a single point of failure and bottlenecks from occurring"*.
  - **Data Integrity and Reliability**: Blockchain's immutability ensures that data recorded cannot be altered, which is important for maintaining data integrity within IoT systems. One participant shared,*"IoT data can be immutable and distributed to ensure the data's authenticity and that it will never be tampered with"* and a participant noted,*"Blockchain benefits the IoT by ensuring reliable data transfer"*.
  - **Traceability**: The traceability offered by blockchain provides a transparent record of all transactions within an IoT system, which participants saw as a significant advantage. *"Blockchain's traceability can help monitor and verify data exchanges"*, one participant mentioned.
  - **Scalability and Efficiency**: Participants also discussed blockchain's scalability concerns, particularly with IoT's massive data generation. It was noted that

blockchain's capability to handle large numbers of IoT data must be considered. As one participant stated, *"Data storage is critical; only the most important data should be stored"*.

4.5.2.4 Theme 4: Feasibility of Blockchain Integration with IoT for Addressing Security Challenges and Mitigating Potential Risks

This theme explores the practicality and viability of using blockchain to address security challenges within IoT systems.

- **IoT Security Challenges**

> *This sub-theme highlights the key security concerns within IoT systems, as identified by participants. These concerns include identity management, access control, and data privacy issues, which are crucial due to the vast number of connected devices and the limitations of IoT technology.*

  - **Identity Management and Access Control Concerns**: Participants emphasized the difficulties in managing identities and access controls in IoT ecosystems, where a large number of connected devices create significant vulnerabilities. One participant noted, *"A large number of devices are expected to be connected that may cause security issues like managing identity and access to the IoT"*.
  - **Privacy Concerns**: Privacy is another major issue in IoT systems, primarily due to the vast amounts of data generated and the frequent need to outsource data processing due to the limited capabilities of IoT devices. One participant stated, *"Due to the limited processing capabilities of IoT devices, third-party service providers are used to process additional data, which may cause privacy issues"*.

- **Blockchain Anticipated Outcomes as a Mitigation**

> *While the previous sub-theme illustrated participants' concerns about IoT security challenges, this sub-them focuses on the anticipated impacts of blockchain technology as mitigation by enhancing data integrity and privacy, strengthening security and trust, improving access control and policy definition, and increasing traceability and transparency.*

  - **Enhanced data integrity and privacy**: Participants anticipated that blockchain could significantly enhance data integrity and privacy within IoT systems by providing a trusted and immutable data record. As one participant mentioned,*"Every single device can be identified management in a permissioned blockchain network for ensuring data generated by an identified device (trusted)"*.
  - **Enhanced security and trust**: Blockchain's decentralized nature is expected to bolster security and trust in IoT systems by addressing common vulnerabilities, such as unauthorized access and data breaches. One participant remarked,*"Blockchain would alleviate security issues of identity management and access control for making it more secure and trusted"*.
  - **Enhanced access control and policy definition**: Participants emphasized the ability of blockchain to define and enforce access control policies to address identification management and privacy issues in IoT systems, which often involve multiple stakeholders with varying access needs. As one participant explained,*"Blockchain can define a set of policies needed to control IoT data access"*.
  - **Enhanced traceability and transparency**: Participants also emphasized the importance of blockchain's traceability and transparency features in enhancing IoT security. Blockchain's ability to provide a transparent and traceable record of all data transactions within an IoT system is seen as a significant advantage. For example, one participant noted in the context of supply chains,*"Blockchain*

allows for the traceability of all recorded data stored, ensuring transparency in supply chains as it is shared with various stakeholders".

4.6 Discussion

4.6.1 Questionnaire

As previously stated, we analyzed the responses of 18 participants who completed the questionnaire and demonstrated moderate to high familiarity with both IoT and blockchain, as indicated by their answers to Questions 4.1 and 4.2, respectively. To begin with, the participants generally view blockchain as a solution to mitigate the challenges facing IoT, as reflected in Question 4.3. This belief is consistent with the responses to the first two questions, which underscore the increasing importance of IoT applications for individuals and organizations in the coming years despite current challenges such as trust, reliability, security, and performance.

In light of these challenges, blockchain technology offers several features, particularly in the realm of security, that can address these concerns. Furthermore, the participants expressed a positive attitude toward the idea of an integrated simulator to mimic the behaviour of IoT and blockchain technologies, as illustrated in Question 4.4. There is a clear indication that a simulation tool is needed to support the modelling of applications that integrate blockchain with IoT. However, most participants opposed this idea when considering the storage of sensed or gathered data on the blockchain, as shown by their responses to Question 4.5. This opposition is primarily due to IoT's vast amount of data per second and blockchain's acknowledged scalability issues. As a result, specialists may reject this approach to avoid potential blockchain failures.

On the other hand, most participants agreed on having multiple consensus algorithms that are supported by the simulator, as depicted in Question 4.6. This agreement arises from the need for developers to evaluate their design proposals under different configurations, especially considering how various consensus algorithms address scalability issues. As design proposals evolve, the consensus algorithm requirements may shift, necessitating

lightweight algorithms, which explains the need for different consensus algorithms in the simulator.

Another significant finding is that most participants prefer capabilities for in-depth investigation of blockchain logs, as reflected in Question 4.7. This preference highlights the need for deep transaction tracking to identify potential delays and other performance issues. Additionally, there is substantial agreement on using IoT devices as blockchain nodes, as established by Question 4.8. However, it is impractical for IoT devices to participate as blockchain nodes due to their limited computational power and energy constraints. Blockchain nodes are required to perform complex cryptographic operations, such as mining and transaction validation, which demand substantial processing capabilities and energy resources.

Lastly, in Question 4.9, participants did not favour having a simulator that includes various blockchain types. The main reason is that each blockchain type has distinct features, and implementing all of these features in a single simulator could increase complexity, thereby limiting its applicability.

4.6.2 Interview

The thematic analysis provided valuable insights into the challenges and opportunities surrounding blockchain-IoT integration. A recurring theme was the significant challenge of assessing system performance. Participants expressed concerns about the difficulty of evaluating blockchain-based IoT systems before real-world deployment, largely due to the limitations of current simulators. These tools often focus on isolated aspects of the system, failing to provide a comprehensive evaluation. This gap highlights the need for more advanced simulation tools that can analyze blockchain-IoT systems from multiple perspectives, a requirement that remains underexplored in existing literature.

Despite these challenges, participants were optimistic about blockchain's potential to address critical IoT issues, particularly in data security and privacy. Blockchain's tamper-proof nature emerged as a significant advantage, ensuring that IoT data remains secure and unaltered. Additionally, blockchain's decentralized architecture reduces the reliance on third-party servers, aligning well with the distributed nature of IoT systems. This decentralization

offers a robust solution for managing and securing the vast amounts of data generated by IoT devices.

However, concerns about blockchain's scalability persisted. The massive data generation typical of IoT systems could overwhelm blockchain networks, making it impractical to store all data on-chain. Some participants suggested a selective approach, where only the most critical data is stored on the blockchain, while less essential information is handled through other means. While this strategy could mitigate scalability issues, it raises new questions about determining the value of data for blockchain storage.

4.7 Recommendations

The results presented in the previous sections have highlighted the significant challenges and opportunities surrounding integrating blockchain with IoT systems. A recurring theme in the discussions was the need for more advanced simulation tools to evaluate blockchain-based IoT systems. Participants emphasized that current simulators often fall short by focusing on only limited aspects, leaving critical perspectives unexplored. On this basis, we recommend greater research and exploration into the design and development of a blockchain-based IoT simulator. These simulators should be capable of analyzing systems from multiple angles, including performance, security, and scalability. Given the lack of such advanced tools in the literature, this presents a crucial opportunity for researchers to address this gap and enhance the evaluation process for blockchain-IoT integration.

Additionally, while participants recognized blockchain's potential to enhance IoT security and privacy, concerns were raised about its scalability in handling the massive data generated by IoT devices. To address this, we recommend further investigation into selective data storage strategies that prioritize the most critical information for blockchain storage while managing less essential data through alternative methods.

4.7.1 Requirements Specification

In response to the challenges and opportunities identified in integrating blockchain with IoT systems, particularly the need for more advanced simulation tools, we propose a set of essential features that should be incorporated into a blockchain-based IoT simulation framework. These features, derived from this study and Chapter 3, are for building a simulation framework for evaluating the performance of Blockchain and IoT to address the complexities of system performance evaluation.

The following Table 4.7 provides a breakdown of these features. These features are categorized into two main sections: **Configuration Parameters (Inputs)** and **Performance Metrics (Outputs)**. These categories address specific areas, including network configuration, transaction configuration, block configuration, consensus mechanisms, IoT configurations, block metrics, and transaction metrics. Each feature is accompanied by a description outlining its function in the simulation, along with its role and the rationale behind its inclusion. This table provides a requirements specification guide utilised in Chapter 5 for designing and building a simulation tool to evaluate blockchain-based IoT.

Table 4.7 Requirements Specification for Blockchain-based IoT Simulation.

	Feature		Description	Role	Rational
Configuration Parameters (Inputs)	**Network Configuration**	**F1**: Node	Defines the total number of nodes that function as blockchain nodes.	To store the ledger and validate all transactions and blocks.	These parameters facilitate the simulation of a blockchain-based IoT network and the analysis of its performance under various conditions.
		F2: Miner Node	Defines the total number of miner nodes that function as blockchain miner nodes.	To perform the mining process to validate transactions and add new blocks to the blockchain.	
	Transaction Configuration	**F3**: Transaction Delay	Specify the average transaction propagation time.	To control how long a transaction takes to propagate across the network.	This parameter allows users to evaluate how delay levels affect the blockchain's ability to process transactions.
		F4: Max Transaction Size	Specify the maximum size of a transaction.	To specify both maximum and minimum transaction sizes for flexibility in evaluating different transaction scenarios.	These parameters allow users to evaluate how transaction size impacts key aspects of blockchain network performance, such as block propagation time and processing speed.
		F5: Min Transaction Size	Specify the minimum size of a transaction.		

Continued on next page

Table 4.7 – continued from previous page

		Feature	Description	Role	Rational
	Block Configuration	**F6**: Block Interval	Defining the time interval between the creation of consecutive blocks.	To control the timing of new block creation.	This parameter enables users to analyze how varying the interval between blocks impacts blockchain performance, particularly regarding transaction execution speed, block creation timing, and overall network stability under different conditions.
		F7: Max Block Size	Specify the maximum size of a block.	To control the number of transactions that can be included in a single block.	This parameter allows users to investigate how block size is able to include the number of transactions and its subsequent impact on network throughput, transaction processing efficiency, and overall latency.
	Consensus Configuration	**F8**: Consensus Type	Select the consensus mechanism (e.g., Proof of Work, Raft etc.) implemented in the proposed simulator.	This parameter controls the method by which blockchain nodes reach agreement on the validity of transactions and the creation of new blocks.	This parameter allows users to simulate different consensus mechanisms to evaluate their impact on network performance, security, and energy consumption.
	IoT Configurations	**F9**: IoT Specifications	Define the properties of IoT, including devices, edge, network, and cloud components.	To model the entire IoT ecosystem with different configurations	This parameter allows users to model IoT environments where data is generated, processed at the edge, transmitted over networks, and processed in the cloud.
		F10: Data Generation Rate	Specify the rate at which IoT devices generate data.	To simulate the load on the blockchain network.	This parameter allows users to evaluate the system's scalability and ability to handle various loads of IoT data.
Performance Metrics (Outputs)	Block Metrics	**M1**: Total Blocks	The total number of blocks created during the simulation.	During the simulation, miner nodes create blocks at intervals determined by the block interval. The total number of blocks is calculated by dividing the total simulation time by the block interval.	To provide a baseline measure of the number of blocks created during the simulation, which helps in evaluating the blockchain's performance and scalability.
		M2: Blocks with Txs	The number of blocks that include one or more transactions.	During the simulation, blocks are filled with transactions based on the transaction generation rate and block size. The metric is calculated by counting the number of blocks that include one or more transactions.	To measure the network's ability to fill blocks with transactions, which indicates the network's transaction processing capability.
		M3: Blocks without Txs	The number of blocks that do not contain any transactions.	During the simulation, blocks that do not contain any transactions are identified. The metric is calculated by subtracting blocks with transactions from the total number of blocks.	To identify blocks that were created without transactions, highlighting inefficiencies in block utilization.
		M4: Avg. Block Size	The average size of the blocks generated during the simulation.	During the simulation, each block takes a certain amount of time to propagate across the network. The average block propagation time is calculated by averaging the time it takes for blocks to propagate through the network.	To determine how well transactions are packed into blocks, which affects network throughput and efficiency.

Continued on next page

Table 4.7 – continued from previous page

	Feature		Description	Role	Rational
		M5: Avg. Block Propagation Time	The average time it takes for a block to propagate across the network	During the simulation, each block takes a certain amount of time to propagate across the network. The average block propagation time is calculated by averaging the time it takes for blocks to propagate through the network.	To assess the time it takes for blocks to propagate through the network, which indicates synchronization and performance across nodes.
	Transaction Metrics	**M6**: Total Txs	The total number of transactions generated and processed during the simulation.	During the simulation, each transaction is tracked to determine if it has been processed. The total number of transactions is calculated by summing all processed transactions.	To measure the total number of transactions processed by the network, providing insights into overall system capacity and efficiency.
		M7: Pending Txs	The number of unconfirmed or unprocessed transactions at the end of the simulation.	During the simulation, transactions that are not processed into blocks are tracked. The metric is calculated by subtracting processed transactions from the total generated transactions.	To track the number of unprocessed transactions at the end of the simulation, identifying potential bottlenecks in the transaction processing.
		M8: Avg. Txs per Block	The average number of transactions included in each block.	During the simulation, block size and transaction size dictate how many transactions can fit into each block. The average number of transactions per block is calculated by dividing the total transactions by the number of blocks with transactions.	To measure the average number of transactions included in each block, which helps evaluate the efficiency of block utilization.
		M9: Avg Txs Size	The average size of the transactions processed during the simulation.	During the simulation, transactions are generated within the specified size range. The average transaction size is calculated by summing all transaction sizes and dividing by the total number of transactions.	To determine the average size of transactions, which influences network performance and the number of transactions that can fit into each block.
		M10: Txs Latency	The average time it takes for a transaction to be confirmed after it is created.	During the simulation, the time from transaction creation to confirmation is measured. The average transaction latency is calculated by averaging the latencies of all transactions.	To measure the time it takes for a transaction to be confirmed to evaluate the speed and responsiveness of the blockchain system.
		M11: Txs Throughput	The rate at which transactions are processed by the network	During the simulation, the rate at which transactions are processed is determined. Transaction throughput is calculated by dividing the total number of transactions by the total simulation time.	To assess the network's ability to process transactions over time, providing a key measure of system performance and scalability.

4.8 Conclusion

IoT systems are becoming increasingly widespread, but current centralized architectures face inherent limitations and challenges. Blockchain technology has the potential to address these issues and unlock new opportunities for IoT applications. However, the lack of a credible

simulator for evaluating the effectiveness of blockchain as a solution to IoT problems hinders the ability to assess and validate such integrated systems. Two studies were conducted to understand the requirements for a blockchain-based IoT simulator: a questionnaire and interviews with experienced participants in the domain of Blockchain and IoT. The results demonstrated the participants' familiarity with both technologies and their strong belief that blockchain can mitigate several key issues faced by IoT systems and performance evaluation. Importantly, the findings highlighted the necessity for a simulator software capable of replicating the behaviour of IoT applications combined with blockchain technology. This chapter's requirements gathering and analysis outcomes play a vital role in the next chapter. These outcomes are considered for designing, implementing, and validating the simulation framework for evaluating the performance of Blockchain-based IoT ecosystems, proposed in Chapter 5.

Chapter 5

Blockchain-based IoT Simulation Framework

Summary

Simulation can offer an opportunity to develop a cost-effective approach for evaluating the performance of blockchain-based IoT solutions. The findings of the literature study reveal simulators that independently target either IoT or Blockchain (refer to Section 2.4.2.1 and Chapter 3; respectively). However, none provides comprehensive support for modelling blockchain-enabled IoT systems for performance evaluation. To cover this gap, this chapter complements the effort conducted in the previous chapter, Chapter 4, for proposing a novel simulation architecture. It is mainly based on the outcome of the previous chapter with regard to simulation framework requirements gathering and analysis. This chapter provides a conceptual design that realises a blockchain-based IoT simulation framework for performance evaluation. To provide a reference implementation, it extends a well-established IoT simulator (namely: IoTSim-Osmosis [16]) to support blockchain simulation and performance evaluation functionalities. The conceptual design and the implementation of the proposed simulator architecture are rigorously evaluated through three types of evaluation: 1) the validity of the conceptual model is assessed through experts' feedback via questioners and a focus group session; 2) the implemented simulator accuracy is validated against a

real-world blockchain platform (namely: Quorum blockchain platform); 3) the quality of the implementation is evaluated by IoT, blockchain, and simulation experts using the goal question metrics (GQM) approach. The results demonstrate that the proposed simulator is able to reflect realistic blockchain-based IoT applications reasonably and provides a useful tool for performance evaluation purposes.

The rest of this chapter is organised as follows: Section 5.1 provides a brief introduction to this chapter and highlights the challenges it seeks to address. The Research Question (RQ), the contributions of this chapter, and its relevance to the published paper are explained in Section 5.2. The proposed conceptual simulation architecture is given in Section 5.3. The implementation of the simulator is highlighted in section 5.3.3 and technically described in Appendix A. Evaluation and analysis tasks for validating the proposed simulator design and implementation are presented in Section 5.4. Finally, Section 5.5 concludes the chapter, summarising the key contributions, findings, and areas of future improvement.

5.1 Introduction

The Internet of Things (IoT) has revolutionized various industries by connecting billions of devices, leading to innovations in healthcare, transportation, manufacturing, and smart cities. However, the centralized architectures typical of IoT systems introduce significant challenges, including security vulnerabilities, inefficiencies, and the risk of single points of failure [3]. These issues, discussed in Section 2.3, necessitate new solutions to ensure the continued growth and reliability of IoT applications.

Blockchain technology has emerged as a promising solution to these challenges by enhancing security, transparency, and efficiency across IoT applications [5]. As outlined in Section 2.3.1, blockchain's decentralized nature addresses the limitations of traditional IoT systems, offering benefits that have already been demonstrated in domains such as healthcare [153], supply chain management [154], and logistics [155].

The integration of IoT and blockchain technologies holds transformative potential across multiple sectors. For example, in the energy sector, IoT devices like smart meters optimize

energy distribution within smart grids. However, centralized control systems can be vulnerable to attacks and inefficiencies. Blockchain mitigates these risks by enabling decentralized energy markets, where transactions are securely recorded on an immutable ledger, reducing the need for intermediaries and enhancing trust among participants.

Similarly, in agriculture, IoT monitors crop health, soil conditions, and weather patterns, but ensuring the traceability and authenticity of products remains challenging. Blockchain addresses this by securely recording the journey of agricultural products, enhancing food safety and reducing fraud. This ensures that data collected by IoT sensors is reliable and tamper-proof, ultimately improving trust in agricultural supply chains.

In smart cities, IoT devices manage urban infrastructure, from traffic lights to waste management systems. However, centralized management can lead to inefficiencies and security risks. Blockchain's decentralized approach ensures that data collected by IoT devices is securely recorded and shared, leading to more efficient and secure city management.

While integrating IoT and blockchain enhances security, transparency, and efficiency, it also introduces performance challenges. Evaluating the performance of blockchain-based IoT applications is crucial in the early stages of development to assess the feasibility and identify potential risks [14]. However, real-world evaluations are often impractical due to the complexity and scale of IoT systems [15]. Simulation tools offer a cost-effective and flexible alternative, enabling researchers to model and analyze these systems under various conditions [14].

Existing IoT simulators, such as IoTSim-Osmosis [16], and blockchain simulators, like BlockSim [128] and SimBlock [137], provide valuable insights but do not focus on the performance evaluation of blockchain-based IoT systems. To address this gap, this chapter proposes a novel simulation architecture that extends IoTSim-Osmosis to include blockchain simulation functionalities, focusing on performance evaluation. The design and implementation of this architecture are rigorously evaluated, demonstrating its viability as a tool for advancing blockchain-IoT integration.

5.2 Research Questions, Contributions, and Relevance to Published Work

Section 5.2.1 provides a detailed explanation of the Research Question (RQ) and highlights the contributions associated with this chapter. Section 5.2.2 clarifies the relevance of this chapter to the published paper, as outlined by the publications listed in Section 1.4.

5.2.1 Research Question and Contribution

Research Question 2 (RQ2): Given the lack of existing simulation frameworks for evaluating the performance of Blockchain-based IoT ecosystems, what is required to bridge the gap? To answer this question, this thesis contributes a simulation framework for evaluating the performance of blockchain-based IoT ecosystems. The contribution of the simulation framework is the first in the literature to combine IoT and Blockchain in a unified simulation tool for performance evaluation purposes. The previous chapter, Chapter 4, took the first step for realising the simulation framework by gathering and analysing relevant requirements. This chapter specifically complements the efforts of the previous chapter by designing and implementing a novel simulation framework that underwent rigorous evaluation and validation procedure for its conceptual design and implementation, as visualised in Figure 1.1.

5.2.2 Relevance of the Chapter to the Published Paper

This chapter aligns with the relevant publication [15], which expands on the work discussed in Chapter 4. It encompasses the same essential subjects, presents similar information and data, and arrives at comparable conclusions as the publication. However, certain sections of this chapter, including the Reference Implementation, Implementation Accuracy Evaluation, and Code Quality Evaluation, have not been published.

5.3 Conceptual Architecture

5.3.1 Research Problem

Assessing the viability and effectiveness of a blockchain-based IoT application on a large scale poses considerable challenges. Consider a scenario of a blockchain-based IoT ecosystem as presented by Alzubaidi et al. [156], where a firefighting station outsources the deployment and operation of its IoT infrastructure to a specialised IoT service provider (IoTSP). In this scenario, the IoTSP is responsible for immediately reporting fire alerts to the firefighting station. However, trust issues may arise if a fire occurs without being reported, either due to the IoTSP's failure to report the incident or the unavailability of the firefighting station's system. To resolve potential disputes, Blockchain employed communicating fire alerts through a shared ledger (see Figure 5.1). Evaluating such a system using real deployment settings faces several hurdles, as follows:

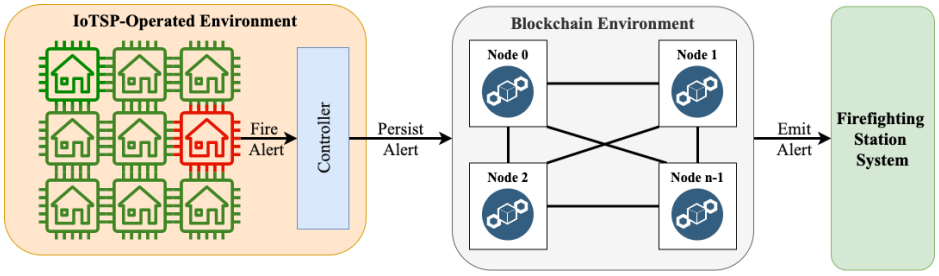


Fig. 5.1 Motivating Blockchain-based IoT Scenario: A firefighting station and IoT service provider (IoTSP) engage in an SLA where the conformance of the IoTSP is measured based on monitoring logs stored on a shared blockchain ledger.

1. **Complexity**: Setting up a large-scale IoT infrastructure solely for experimental purposes is highly complex and requires extensive planning, coordination and technical expertise.

2. **Resource limitations**: The lack of sufficient IoT devices and resources can hinder the ability to perform comprehensive tests and experiments, limiting the scope and validity of the results.

3. **Personnel shortages**: The shortage of technical personnel to manage the deployment process can lead to delays, errors, and inconsistencies in the experimental setup, affecting the reliability of the findings.

4. **High costs**: Real-world testing of large-scale blockchain-based IoT deployments can be expensive, considering the costs associated with hardware, platforms, underlying infrastructure, and workforce.

Simulation plays an important role in the early assessment of new technologies such as blockchain and IoT. In particular, simulation helps pinpoint strengths and weaknesses and determines the necessary steps to reach performance goals. Despite the importance of simulation for blockchain-based IoT solutions, there is currently a lack of flexible simulation frameworks specifically designed for evaluating the performance of blockchain-based IoT solutions [13]. Existing simulation tools often focus on either IoT or blockchain independently and lack the necessary integration and extensibility to capture the unique characteristics and requirements of blockchain-based IoT systems [14]. Moreover, many simulation frameworks are limited in their ability to model various IoT scenarios, blockchain architectures, and consensus algorithms, hindering the exploration of novel solutions and comparative analysis [14, 11].

5.3.2 Proposed Architecture

This section delineates the conceptual framework of the proposed simulator that integrates Blockchain technology within the Internet of Things (IoT) environments, as illustrated in Figure 5.2. This conceptual architecture aims to provide a comprehensive and modular approach to simulating Blockchain-based IoT ecosystems, allowing customisation and evaluation of various scenarios. The architecture is segmented into four main components: the Configurator 5.3.2.1, the Generator 5.3.2.2, the Simulation Core 5.3.2.3 and the Reporter 5.3.2.4.

5.3.2.1 Configurator

The Configurator module is responsible for establishing the simulation parameters from both the IoT and Blockchain perspectives. It serves as the starting point for specifying various simulation parameters as follows:

1. **Blockchain**: Table 5.1 summarises most important Blockchain configuration parameters.

2. **IoT**: Typical IoT configuration parameters already provided by IoTsim-Osmosis [16] such as the specifications of devices, edge, network, and cloud components.

3. **Simulation Settings**: Examples of which are the duration of simulation and the number of simulation runs.

Table 5.1 Summary of Configuration parameters.

Type	Parameter	Description
Participating Nodes	N_{n}	Total number of nodes
	M_{n}	Total number of miners
Transaction	Tx_{delay}	Average transaction propagation delay in seconds
	Tx_{maxSize}	Maximum transaction size in megabytes (MB)
	Tx_{minSize}	Minimum transaction size in megabytes (MB)
Block	B_{interval}	Average of the time interval between the creation of consecutive blocks
	B_{maxSize}	Maximum block size in megabytes (MB)
Consensus	C_{type}	The consensus algorithm used in the simulation (e.g., "raft" or "PoW")
IoT specifications	IoT	Properties of IoTsim-Osmosis [16] such as devices, edge, network, and cloud components
Simulation	S_{simTime}	Simulation time in seconds
	S_{runTime}	The number of times the simulation is run

On the IoT side, the module leverages IoTSim-Osmosis [16], a sophisticated extension of the CloudSim [86] framework that facilitates the customisation of the IoT environment. IoTSim-Osmosis allows the specification of various components of the IoT architecture in detail, including sensors, actuators, devices, edge computing units, network topology, data centres, and computing resources.

In terms of the Blockchain dimension, the Configurator module provides tools for setting up the Blockchain network. This includes defining essential network parameters such as the number of nodes (miners or validators, based on the simulated IoT topology), block size, block generation difficulty, transaction size, transaction delay, and the choice of consensus mechanism (e.g., Proof of Work, Raft, etc.). Additionally, the module allows for customisation of simulation-specific parameters, such as the number of simulators to be deployed concurrently.

5.3.2.2 Generator

Based on the specified settings outlined in Section 5.3.2.1, the generator module within the proposed framework is tasked with establishing the essential infrastructure for both the IoT application and the blockchain network. Utilizing the parameters established by the configurator module, the generator module generates the requisite components and links for the IoT architecture and the blockchain network. For instance, it can generate the essential sensor nodes and edge units for the IoT structure, along with the data transmission and reception protocols. Additionally, it can establish the participating nodes in the blockchain network, such as miners or validators, and configure parameters (e.g. block settings, transaction settings, and the chosen consensus algorithm).

5.3.2.3 Simulation Core

The simulation core in the proposed conceptual model typically consists of several main components that work together to simulate the operation of the system. These components include:

1. **The transaction factory and workload feeder**: are components in a simulation environment for a blockchain-based IoT system. The transaction factory is responsible for generating transactions based on the data collected from the workload feeder, while the workload feeder manages the flow of transactions and ensures that they are processed efficiently and accurately. The transaction factory follows a specific process to create and broadcast transactions, including:

(a) **Construct a Transactions Structure**: The transaction factory prepares the format of the transactions to match the structure required by the blockchain network. This includes defining the data structure, the required fields for the transactions, and any other requirements or constraints.

(b) **Broadcast transactions to miner nodes**: Once a transaction structure is constructed, it then broadcasts the prepared transactions to all network nodes to inform them of the new transactions.

(c) **Appending the transactions to the transaction pool**: A collection of pending transactions that are waiting to be added to the blockchain.

The process of generating and managing transactions is typically repeated until the workload feeder no longer feeds transactions into the system.

In a blockchain network, the miner nodes are responsible for creating and adding blocks of transactions to the blockchain. When a miner receives transactions in its transaction pool, it typically tries to create a block by selecting a subset of the transactions from the pool and adding them to a new block. The process of creating a block is often referred to as an “event" because it represents a significant event in the operation of the blockchain network. To create a block, a miner typically must perform a consensus algorithm, such as a proof of work, which involves using cryptographic algorithms to demonstrate the work that has been done to validate and include the transactions in the block. In a simulation environment, the aim may be to simulate the process of creating blocks and adding them to the blockchain to test and evaluate the performance of the network and the mining nodes. In the conceptual model, we create a Block Factory component.

2. **The block factory**: is a component of a simulation environment for a blockchain-based IoT system. It is responsible for simulating the process of creating blocks and adding them to the blockchain. The block factory follows a specific process to create and execute transactions, including:

(a) **Invoking and Executing Transactions**: The miner selects a subset of pending transactions from the transaction pool based on certain criteria, such as the time the transactions were created, the gas price associated with them or the order in which they were received.

(b) **Append Transactions to Next Block**: When a miner node receives transactions in its transaction pool, it will typically try to create a block by selecting a subset of the transactions from the pool and adding them to a new block. This process is known as “appending" the transactions to the block, as it involves adding the transactions to the block and preparing them for inclusion in the blockchain.

(c) **Constructing block and append it to the local blockchain**: The block has been created with its set of transactions.

(d) **Append Block to local Blockchain**: Once the block has been constructed, it is ready to be appended to the local copy of the blockchain. This involves adding the block to the end of the local copy of the blockchain and updating the local copy to reflect the new block.

(e) **Broadcast the block to other nodes**: The miner broadcasts the newly added block to all other nodes in the network in order to inform them of the new block and update their copies of the blockchain.

Once a block has been broadcast to the blockchain network, it becomes the responsibility of the block receivers to validate the block and decide whether to accept it and add it to their copy of the blockchain. The process of validating a block involves verifying that the block meets all the requirements and standards of the blockchain network. This may include checking the block header to ensure that it includes a valid reference to the previous block in the blockchain and verifying the transactions contained in the block to ensure that they are valid and properly formatted. In the conceptual model, we create the Received Blocks component.

3. **Received Blocks**: A component of a simulation environment for a blockchain-based IoT system. It is responsible for receiving blocks that have been broadcasted to the

network and deciding whether to accept them and add them to the local copy of the blockchain. The received blocks component typically follows a specific process when receiving a new block, which may include the following steps:

(a) **Check Validity of Received Block**: When receiving a new block, one of the key tasks of the Received Blocks component is to check the validity of the received block.

(b) **Updating and Append it to the Local Blockchain**: If the received block is deemed valid, the next step in the process is to update the local copy of the blockchain and add the received block to it. This involves adding the received block to the end of the local copy of the blockchain and updating the local copy to reflect the new block.

(c) **Updating the transaction pool**: Once the new block has been added to the local blockchain, the node will update the transaction pool by removing the transactions that were included in the block. This leaves the transaction pool with only the transactions that have not yet been included in a block, allowing the node to continue the process of verifying and adding new transactions to the blockchain.

5.3.2.4 Reporter

The benchmark report is an important part of the simulation process, as it provides detailed information about the performance and viability of a blockchain-based IoT system. In our proposed conceptual model, once the simulation is finished, the simulator will prepare the benchmark report as an Excel file, which consists of several sheets, each of which provides specific information about different aspects of the system, as shown below

1. **Configuration**: This provides information on the parameters used to carry out the experiment, such as the type and number of nodes, the blockchain protocol used, and any other relevant system parameters.

2. **Overall result**: A benchmark report provides a summary of the overall performance of a blockchain-based IoT. This includes a range of statistics that can be useful for understanding the system's performance and identifying any issues or opportunities for improvement. Some examples of the types of statistics that be included in the "Overall result" section include:

   (a) **Total number of blocks**: This is the total number of blocks that were added to the blockchain during the simulation.

   (b) **Total number of blocks, including transactions**: This is the total number of blocks that contain at least one transaction.

   (c) **Total number of blocks without transactions**: This is the total number of blocks that did not contain any transactions.

   (d) **Average block size**: This is the average size of the blocks on the blockchain.

   (e) **Total number of transactions**: This is the total number of transactions that were processed during the simulation.

   (f) **Average number of transactions per block**: This is the average number of transactions included in each block.

   (g) **Average transaction inclusion time**: This is the average time it took for a transaction to be included in a block.

   (h) **Average transaction size**: This is the average size of the transactions processed during the simulation.

   (i) **Total number of pending transactions**: This is the total number of transactions that were waiting to be processed at the end of the simulation.

   (j) **Average block propagation time**: This is the average time it takes for a block to propagate to all nodes on the network.

   (k) **Average transaction latency**: This is the average time it takes for a transaction to be processed and added to the blockchain.

(l) **Transaction execution**: This is the percentage of transactions that were successfully processed during the simulation.

(m) **Transaction throughput**: This is the number of transactions that were processed per second.

3. **Blocks overview**: A benchmark report provides details about the individual blocks that were added to the blockchain during the simulation. This includes information such as block ID, previous block ID, block depth, block timestamp, block size, number of transactions, and minter.

4. **Transactions latency overview**: A benchmark report provides details on the latency for each transaction in a blockchain-based IoT, including the transaction ID, the creation time, the confirmation time, and the transaction latency.

5. **Pending Transactions overview**: A benchmark report documents transactions not executed during the simulation, if any at all.

6. **Statistic**: A benchmark report provides statistical information about the performance of a blockchain-based IoT system. Specifically, it provides details about the distribution of block time and block latency, including the minimum, maximum, mean, and standard deviation of these metrics.

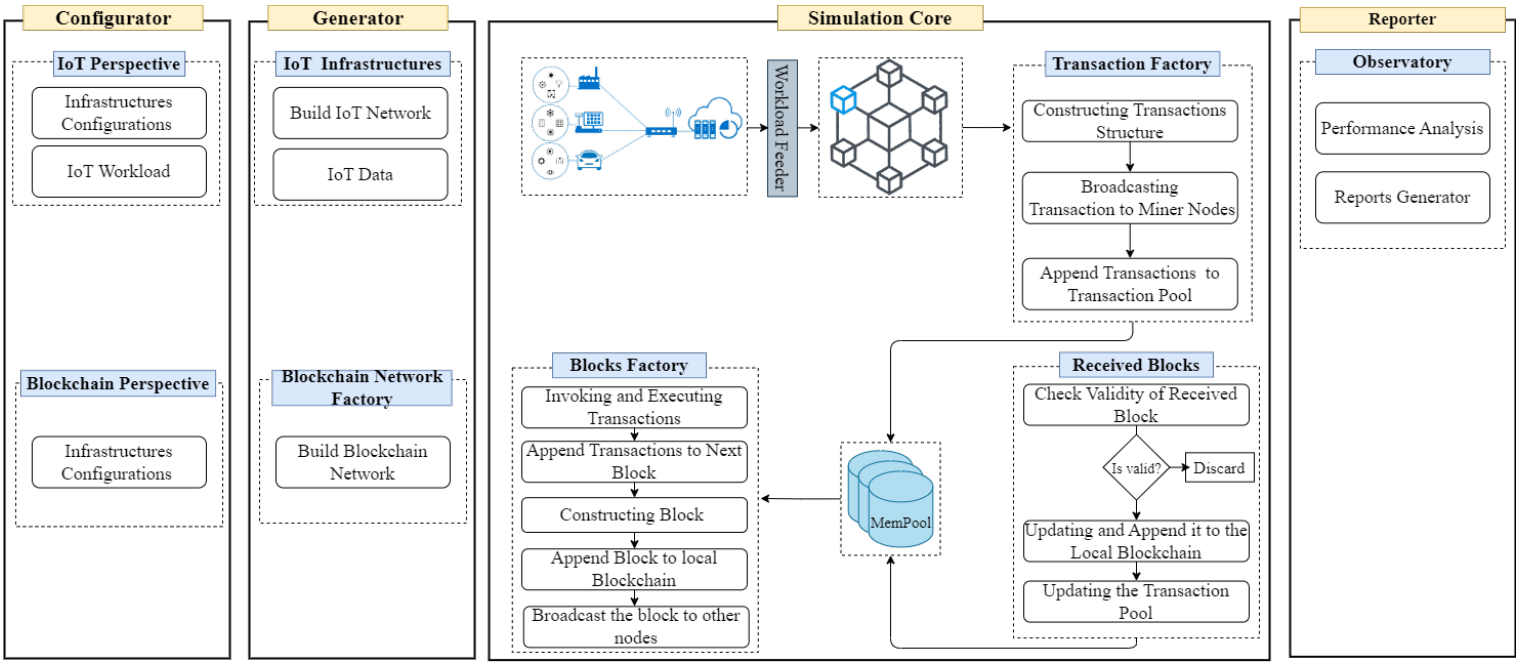


Fig. 5.2 An overview of the Conceptual Model for Simulating Blockchain-based IoT Ecosystems.

5.3.3 Reference Implementation

This thesis provides a Java-based reference implementation for the proposed Blockchain-based IoT simulator, which materialises the conceptual design explained above in section 5.3.2. Appendix A delves further into the technical details of the implementation, which can also be accessed at a public GitHub repository [1]. For clarity, the code of reference implementation is annotated with self-explanatory comments and follows the best practice of utilising naming conventions. Appendix B also provides a user manual to facilitate the usage of the simulator. Java is the main programming language used to implement the simulation architecture. The selection of Java is to align with the programming language used for implementing IoTsim-Osmosis. The reference implementation extends the IoTsim-Osmosis simulator [16] and builds upon it to support blockchain-specific parameters, functionalities, and performance metrics.

[1] https://github.com/AlbshriAdel/BlockSimOsmosis

Noteworthy mentions that the simulation supports two consensus mechanism alternatives: Raft and PoW (Proof-of-Work). The implementation also involves the components necessary for the integration between IoT and Blockchain, as well as monitoring and performance evaluation measurements. The implementation also employs a library named *ExcelWriter* for producing and reporting a benchmark report in Excel format, organising data into distinct sheets for the configuration parameters and overall evaluation outcomes (throughput, latency, etc.), as discussed in Section 5.3.2.4. Appendix A.2.5 further illustrates this library. The generated reports also provide details of the most important simulation events to help explain phenomena and recognise patterns, such as a view of the blockchain ledger, transaction contents, transaction pool, and other statistical findings. This library also assists in preparing the generated from both the simulator and a real blockchain performance benchmark process to conduct a comparative analysis to ensure the simulator's accuracy. Section 5.4.2 describes the comparative analysis, and Appendix C.2.6 provides a visual illustration of the process.

5.4 Evaluation

The proposed simulator architecture underwent a rigorous evaluation process to assess the following:

1. The validity of the architecture conceptual design, as presented in Section 5.4.1.
2. The precision of the simulator implementation against a real blockchain platform, as presented in Section 5.4.2.
3. The quality of the simulator implementation, as presented in Section 5.4.3.

5.4.1 Conceptual Design Evaluation

The proposed simulator conceptual design was evaluated through a combination of questionnaire responses and a focus group interview with subject-matter experts. The following describes the evaluation process of the simulator conceptual design.

5.4.1.1 Participant Recruitment and Selection

This section outlines how participants were recruited and selected for the study, including identifying potential participants, selection criteria, and the approach adopted. It also provides participant demographics based on academic levels, professional backgrounds, and research interests.

1. **Identifying Potential Participants**: Potential participants were identified through professional academic networks (e.g., ResearchGate), research groups specializing in Blockchain and IoT both within and outside my educational institution, and scholarly publications in IoT and Blockchain.

2. **Selection Criteria**: To ensure the suitability of participants for this study, we established selection criteria based on the following factors:

   (a) Knowledge and experience in blockchain and IoT technologies.

   (b) Publications or projects related to blockchain and IoT.

   (c) Willingness and availability to take part in the focus group discussion.

3. **Recruitment Approach**: A targeted sampling approach was implemented to recruit participants for the study. Initially, 15 potential candidates were sent invitations via email, outlining the study's purpose, the importance of their expertise, and the expected time commitment. Follow-up emails were then sent to address any queries and confirm participation. Ultimately, 10 out of the 15 invited individuals agreed to participate.

4. **Participant Demographics**: The study gathered demographic information from participants, including their academic level, professional background, and research interests, summarized in Table 5.2. The focus group comprised 10 participants with diverse academic and professional backgrounds. Most participants held doctoral degrees (8), while the remaining 2 had master's degrees. Nine participants were affiliated with academic institutions, and one was from the industry sector. The participants' research interests included blockchain technology and its varied applications across different domains in the Internet of Things (IoT).

Table 5.2 Participant Demographics.

Participant	Academic Level	Professional Background	Research Interest
1	PhD	Academia	Blockchain technology and IoT applications
2	PhD	Academia	Blockchain performance
3	PhD	Academia	Blockchain-based SLA in the context of IoT
4	PhD	Academia	Data privacy in the context of IoT via blockchain
5	Master's	Academia	Allocating Cloud resources & blockchain
6	PhD	Academia	Optimisation blockchain with the Internet of Vehicles (IoV)
7	PhD	Academia	IoT and Blockchain
8	PhD	Academia	Research related to IoT, Cloud and Blockchain
9	PhD	Academia	IoT data management and blockchain
10	Master's	Industry	emote health monitoring using IoT and blockchain

5.4.1.2 Procedure

A focus group session was held involving subject-matter experts to identify and examine the challenges of implementing and evaluating the conceptual model. The session began with a presentation on the challenges of merging Blockchain with IoT, including issues related to scalability, security, and interoperability, as well as the limitations of existing simulators for Blockchain and IoT. An overview of the IoTSim-Osmosis simulator [16] was provided to give insight into the simulator planned for extending blockchain features. A real-world scenario involving a fire station 5.1 was shared to depict potential integration and deployment challenges. Subsequently, the conceptual model was presented, and participants were asked to complete a questionnaire to evaluate its clarity, relevance, and practicability. The questionnaire featured four closed-ended questions, answered via a five-point Likert scale, aimed at collating participants' perspectives on implementing and evaluating the conceptual model. Additionally, two open-ended questions were included to allow for feedback and suggestions on improving the conceptual design.

1. ***Questionnaire***

   (a) **To what extent are you satisfied with the conceptual model?**

   (b) **To what extent are you satisfied with the conceptual model's generality?**

(c) **Assuming that IoTsim-Osmosis is the base IoT simulator for the conceptual model, to what extent do you agree that it covers your requirements?**

i. ease of use

ii. configurability

iii. extensibility

iv. maintainability

v. network topology

(d) **To what extent does the blockchain part cover your requirements?**

2. ***Focus Group***

(a) **What are your overall thoughts on the conceptual model for the blockchain simulator?**

(b) **Do you believe the conceptual model for the blockchain simulator is comprehensive and well-designed, or are there any areas that you feel need further improvement or refinement?**

These questions aim to achieve these predefined goals as follows:

1. To evaluate the inclusiveness and quality of the conceptual model.
2. Determine to what extent the IoTsim-Osmosis simulator meets the requirements of the participants.
3. Assess the effectiveness of the conceptual model's blockchain component in meeting the participants' needs.
4. Identify areas of the conceptual model that may need improvement.

5.4.1.3 Experimental Results

Questionnaire

To validate the proposed conceptual model, we distributed a questionnaire to 10 participants. The questionnaire began by asking about their satisfaction with the conceptual model and its applicability. Specifically, we asked the participants: “To what extent are you satisfied with the conceptual model?" The results shown in Figure 5.3 indicate that the majority of the participants were satisfied, with 60% expressing satisfaction and 30% expressing complete satisfaction. 10% were neutral with regard to the model.

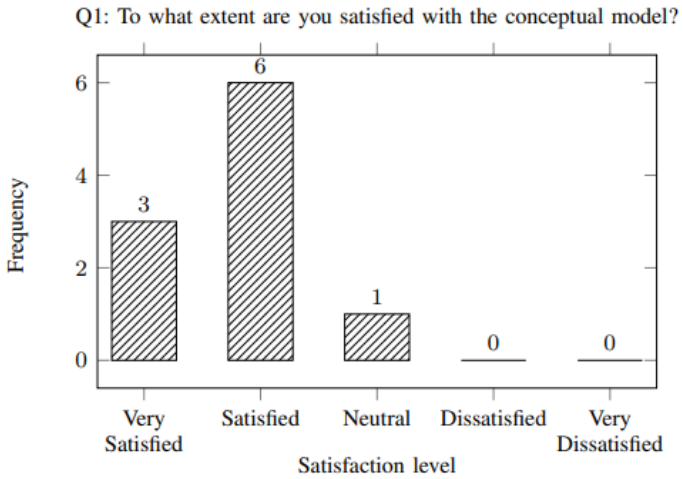


Fig. 5.3 Participant satisfaction with the conceptual model.

Next, the participants were asked about their thoughts on the applicability of the conceptual model. The results of this question are shown in Figure 5.4. It’s worth noting that there is complete agreement about the model, with 40% of participants indicating that they were completely satisfied and 60% indicating that they were satisfied. Based on these results, we can confirm that the proposed conceptual model is a good fit.

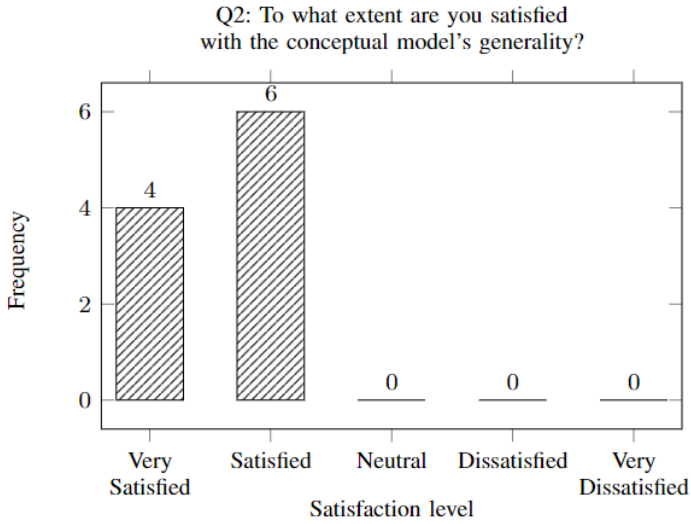


Fig. 5.4 Participant satisfaction with the conceptual model's generality.

Four questions were asked to evaluate the usability, configurability, maintainability, and available network topology of the IoTsim-Osmosis simulator, which is at the core of the proposed conceptual model. Specifically, participants were asked about their level of agreement with the ease of use of the simulator ("To what extent do you agree with its ease of use?"). The results, shown in Figure 5.5, indicate that 30% completely confirms the usability of the simulator, while 40% agrees with its ease of use. 20% were neutral about the simulator, and 10% disagreed with its ease of use.

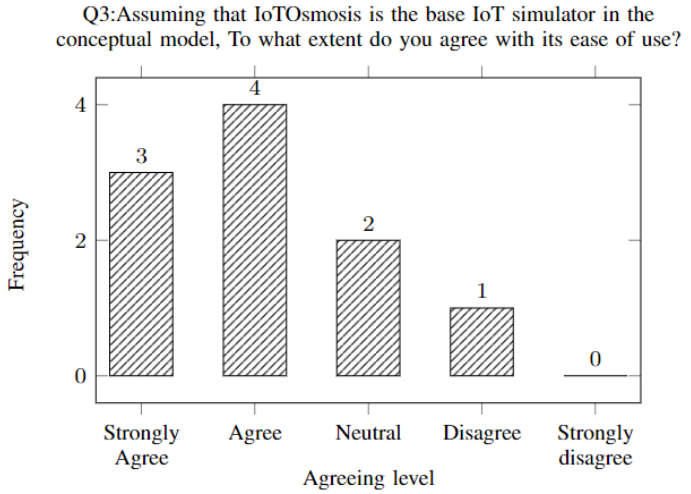


Fig. 5.5 Participant agreement with the ease of use of the IoTsim-Osmosis simulator.

Furthermore, participants were asked about the configurability of the IoTsim-Osmosis simulator (“To what extent do you agree with its configurability?"). The results shown in Figure 5.6 show that 60% of the participants (10% completely agree and 50% agree) are satisfied with the configurability of the simulator. 20% had neutral or disagreeing opinions. The participants were also asked about the maintainability of the simulator ("To what extent do you agree with its maintainability?"). The results shown in Figure 5.7 indicate a clear agreement with 70% of the participants (20% strongly agree and 50% agree), indicating satisfaction. 20% disagreed with the ease of maintainability, while 10% were neutral. A final question about the IoTsim-Osmosis simulator was, "To what extent do you agree with the network topology?" The results are shown in Figure 5.8.

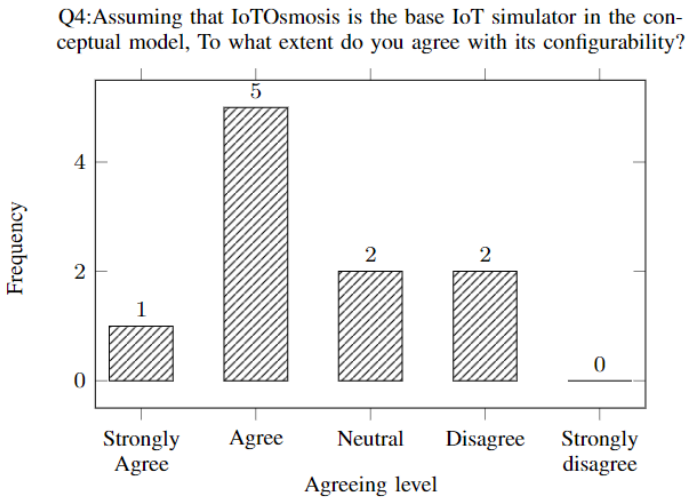


Fig. 5.6 Participant agreement with the configurability of the IoTsim-Osmosis simulator.

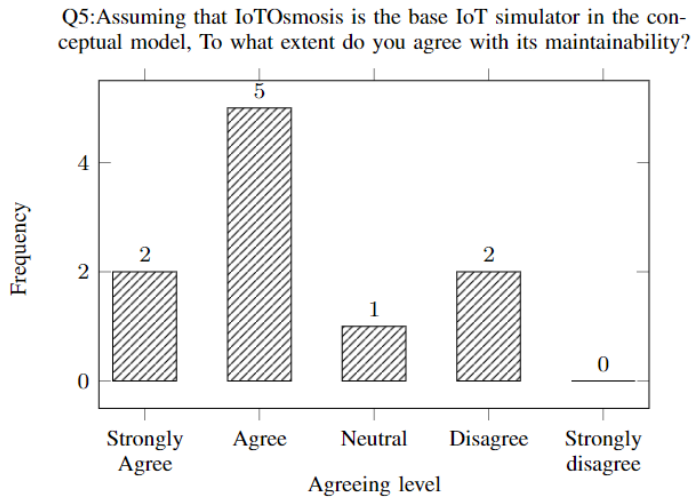


Fig. 5.7 Participant agreement with the maintainability of the IoTsim-Osmosis simulator.

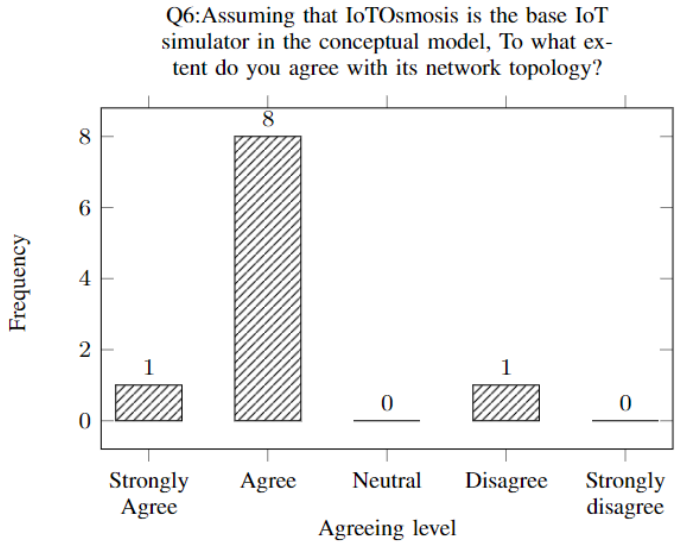


Fig. 5.8 Participant agreement with the effectiveness of the blockchain part in meeting their requirements.

Finally, the questionnaire concluded by asking participants their thoughts on the ability of blockchain to meet their requirements. The results are shown in Figure 5.9. A closer look at the figure reveals a high level of agreement (30% strongly agree and 50% agree) on the usefulness of blockchain. There were an equal number of neutral (10%) and disagreeing (10%) opinions.

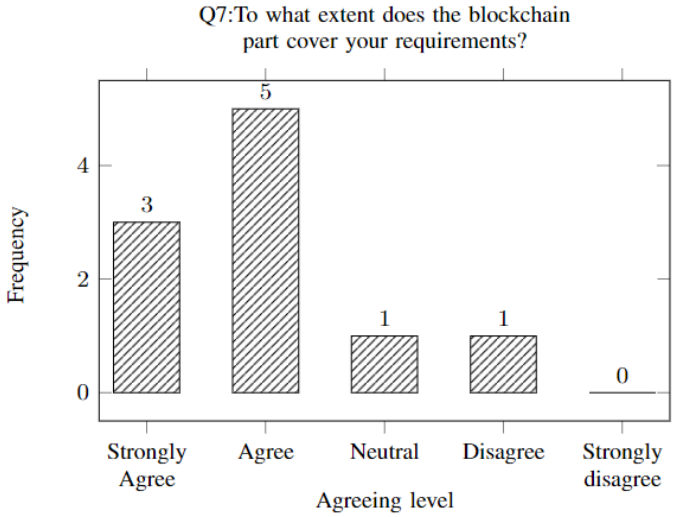


Fig. 5.9 Participant agreement with the effectiveness of the blockchain part in meeting their requirements.

Overall, a questionnaire was distributed to a total of 10 participants to validate the design of the proposed conceptual model. The questionnaire asked the participants about their satisfaction with the conceptual model and its applicability in an IoTSim-Osmosis simulator. It also enquired about the usability, configurability, maintainability, and network topology of the IoTSim-Osmosis simulator, which is integrated into the proposed model. Additionally, participants were asked about the effectiveness of blockchain in meeting their requirements. The results showed that 40% of the respondents were completely satisfied, while 60% were satisfied with the conceptual model and its applicability. Furthermore, most of the participants expressed satisfaction with the IoTSim-Osmosis simulator in terms of usability, configurability, maintainability, and network topology. Regarding the usefulness of the blockchain part for meeting participants' requirements, 30% strongly agreed, and 50% agreed.

Focus Group

P1 stated that *"In my opinion, the conceptual model is well designed and complete. It looks like it consists of a set of features and functionalities needed to enable effective*

simulation and evaluation of blockchain-based IoT systems. I think these capabilities will be important for understanding and optimising the performance of blockchain-based IoT systems and for identifying any potential issues or challenges that may arise during deployment".

P2 stated that *"Overall, the primary model is valuable and thoroughly considered. It covers every one of the key parts and capacities that we hope to find in a test system of this kind, and it gives off an impression of being generally evident when executing the plan that has been displayed. There are a couple of places where I figure the model could be extended. For example, it may be helpful to have more control over the different parameters and settings of the test system, such as the ability to specify the type of consensus algorithm or the block size".*

P3 stated that *"In my opinion, the conceptual model is a promising idea. The conceptual model includes a large variety of features and capabilities, which is quite amazing. Moreover, the simulation core is well-structured. It seems that the major components of the simulation core, like the transaction factory and workload feeder, consensus component and monitoring components, are designed carefully. However, for the developer who wants to customize the simulation, it would be better if there were more opportunities to make changes in parameters such as transaction types and the workload feeder".*

P4 stated that *"As a beginner in the fields of blockchain and IoT, the conceptual model of the simulator felt very easy to understand and well structured for me. It includes various components and their functions and gives a nice overview of how they work together. One thing that can be beneficial is to have an explanation about the specific consensus algorithms that are implemented in the implementation phase".*

P5 stated that *" The concept model of the simulator seems very promising because it encompasses a number of metrics that let users see how it will perform in the real world. On the other hand, it could be useful to make the model more flexible to allow it to be deployed to different layers, which would give users more flexibility to deploy the blockchain network where and how they want and make it much easier to fit different use cases and environments".*

P6 stated that *"The simulator's conceptual model is well-thought-out for providing a way for users to evaluate the performance of different blockchain and IoT solutions, as well as present users with a methodology to recognise hurdles and concerns in these solutions"*.

P7 stated that *"In my opinion, the simulator architecture is well-architected and comprehensive and includes most of the essential features to evaluate blockchain-based IoT systems. However, there may be potential to improve its capability in simulating enterprise blockchain environments and its integration with other simulators rather than IoTSim-Osmosis. Improving these features would make the simulator more versatile and highly valuable for diverse applications"*.

P8 stated that *"I think that the conceptual model appears to be good, as it includes multiple aspects. One aspect of the model that I particularly like is the generator component, as it allows the model to be used with different generating simulators"*.

P9 stated that *"I think the model is a solid foundation for further research and development. It includes a wide range of important features and abilities, and I feel that it could be valuable for many different uses. However, I think there are a few areas that could be improved on or expanded. For example, simulation abilities can be used to evaluate a variety of scenarios. Additionally, more options to generate and analyse results would be helpful. Being able to compare different simulation scenarios or run the simulations for a longer amount of time"*.

P10 stated that *"I think that the proposed conceptual model is good, and I really like the idea of adding the configurator component. The configurator allows users to customise the simulation parameters to meet their specific needs. However, I think it will help to have more options to configure the blockchain network. It would be nice if we could allow the user to choose different types of consensus algorithms or customise the block settings"*.

By analysing the responses of the participants as shown in Table5.3, the result of the evaluation of the conceptual model showed that it is generally well-regarded. The reason underpinning this attitude is the inclusion of a wide range of key features and capabilities that make it a suitable foundation for creating a simulation environment for blockchain-based IoT applications. However, there are also a few areas where the model could be

improved or expanded upon. Some participants have suggested adding more granular control over the various parameters and settings of the simulator. Others have suggested including more information about the specific consensus algorithms that will be implemented in the implementation phase and making the model more flexible by allowing it to be deployed in different layers.

Table 5.3 Summary of feedback on the conceptual model for the blockchain simulator from participants in the evaluation process.

Participant	Overall Thoughts	Areas for Improvement
P1	Well-designed and comprehensive	
P2	Solid and well-thought-out	More granular control over parameters and settings
P3	Promising concept	More options for customizing the simulation
P4	Easy to understand and well-organized	Consensus algorithms
P5	Very promising	Flexibility to deploy BC in different IoT layers
P6	Well-designed	
P7	Comprehensive and well-designed	Ability to simulate enterprise blockchain and support different IoT simulators
P8	Good	
P9	Strong foundation	Configuring the blockchain network
P10	Good	

5.4.2 Implementation Accuracy Evaluation

An important assessment objective is to evaluate and validate whether the performance report produced by the implemented simulation architecture can accurately represent the performance of a real blockchain platform. Evaluating this aspect is the key to determining the practical utility of the developed simulator. Transaction latency and throughput are representative performance metrics to conduct a comparative analysis between the implemented simulator and a real blockchain platform (namely, the Quorum blockchain platform). To achieve this, the assumed firefighting station scenario 5.1 is implemented using a real blockchain platform. Hyperledger Caliper [71], a well-recognised blockchain benchmark tool, is used to gauge the performance of real blockchain platforms and produce throughput and latency metrics. The same firefighting scenario is then modelled using the implemented simulator for replication purposes to reproduce the throughput and latency metrics.

The remainder of this section describes the data collection method used in this study and compares the results of the implemented simulator with the real-world blockchain.

5.4.2.1 Experimental Setup

There are four main steps for obtaining the latency and throughput performance metrics from the real-world blockchain platform and the implemented simulator.

1. **The Real Blockchain Platform**: Table 5.4 lists important facts about the launch of the real blockchain platform (Quorum blockchain platform). Refer to Appendix C.1.2 for technical details. The Blockchain network is deployed to a rented cloud-based infrastructure composed of 32 virtual CPUs (vCPUs) from an Intel(R) Xeon(R) Gold 6140 processor, clocked at 2.30GHz, along with 64GB of RAM. These resources are equally distributed to the blockchain network components.

2. **Hyperledger Caliper**: Hyperledger Caliper is used to benchmark the performance of the deployed real blockchain platform (See Appendix C.2). Three pivotal files are prepared for executing the performance benchmarking process: The business Logic file, the benchmark properties file, and the network interface file. On the one hand, the benchmark file is a YAML-based formatted manifesto set, which defines the workload for stressing the performance of the real blockchain platform. For the purpose of this thesis, it is customised to employ a single client (designated as one worker) tasked with delivering transactions at variable rates to the validating nodes within the Quorum network, as defined in Table 5.5. On the other hand, the network interface file is another YAML-based formatted manifesto that is instrumental in facilitating connectivity to and communication with the deployed blockchain network. It also defines the parameters needed to interact with the blockchain network, such as sender and smart contract addresses associated with a cryptographic wallet account. Finally, the business logic file is used for expressing and encoding the behaviour of the assumed IoT-based firefighting scenario as in 5.1. The code was written using the programming language adopted by Hyperledger Caliper, which is JavaScript.

3. **Python script**: A Python script is programmed for automating the execution of the performance benchmark process as defined in the configurations of Hyperledger Caliper (See Appendix C.2.4). It automates the benchmarking process across several iterations for each experimental run to accumulate a sufficient representative data set and ensure the calculation precision of the average throughput and latency metrics. The script orchestrates several critical operations. The script parses the results from each benchmark iteration into a DataFrame, with the aid of Pandas library, which standardises the data format to prepare for further analysis.

4. **Simulation**: The same IoT-based firefighting scenario as in 5.1 is also implemented in the simulator. Moreover, the configuration properties of the real blockchain platform are also applied to the simulator configuration whenever applicable. The simulator's IoT configuration aspects also consider the JavaScript implementation of the IoT scenario in Hyperledger Caliper.

Table 5.4 Configuration Parameters for Blockchain.

Parameter		Description	Specifications	
Blockchain Configuration	**Blockchain Platform**	The chosen blockchain technology for the study	Quorum Version 22.7.5	
	Consensus Algorithm	The algorithm employed for achieving consensus in the network	RAFT	
	Node Count	The total number of active nodes within the blockchain network	4	
	Block Time	The set block time for RAFT consensus in milliseconds	300	
Deployment Resources	**Resources Allocation**	Details of resources allocation	**OS**	Ubuntu Linux 20.04.2 (64-bit)
			CPUs	4
			RAM	64GB
			Bandwidth	100 Mbps
			Storage	200 GB
Dependencies	**Docker**	Use of Docker in the deployment and management of network nodes	20.10.21	
	NPM	Node Package Manager used	8.18.0	

Table 5.5 Configuration Parameters for Caliper.

Parameter	Specifications
Benchmark Tool	Hyperledger Caliper V0.5.0
Workers	1 worker
Total transactions per iteration	25 to 3200 transactions
Send rate type	Fixed
Send rate control	1 transaction per second (tx/sec)

5.4.2.2 Comparative Analysis and Validation

As mentioned above, the IoT-based firefighting station scenario is implemented in both the real blockchain platform and the proposed simulator. The performance metrics (namely,

throughput and latency) are obtained from both to conduct a comparative analysis and validate the accuracy of the implemented simulator.

Regarding the latency performance metric, as shown in Figure 5.10, the transaction latency from the real blockchain platform appears to be correlated with the transaction throughput, which increases progressively from approximately 1 second for processing a send rate of 25 transactions per second (TPS) to approximately 4.5 seconds for processing a send rate of 3200 TPS. Similarly, the simulation demonstrates a parallel ascendancy in latency, yet the latency values are consistently modestly lower, thereby suggesting the simulation's relative precision with a tendency towards optimistic projections. This comparative assessment affirms the credible capacity of the simulator to reasonably emulate the transaction latency of the corresponding real blockchain system.

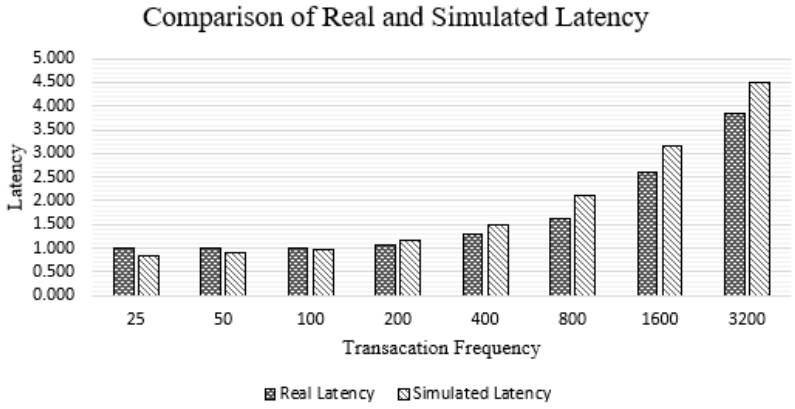


Fig. 5.10 Comparison of Real and Simulated Latency

Regarding the throughput performance metric, as shown in Figure 5.11. a consistent upward trajectory is observed in the transaction processing rate. That is, a trend is apparent in both the empirical environment and the simulation framework. The transaction throughput of the real blockchain platform demonstrates the ability to process nearly 13 TPS at a send rate of 25 TPS, ascending to approximately 165 TPS at a send rate of 3200 TPS. On the other hand, the simulator forecasts a marginally elevated throughput, from being able to process approximately 13 TPS up to nearly 176 TPS for identical send rates as the above. This finding also confirms the simulator's ability to accurately emulate the transaction throughput of a corresponding real-world blockchain platform.

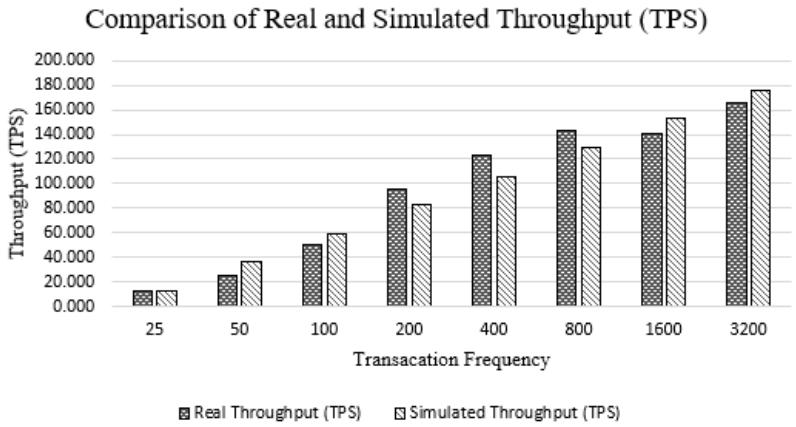


Fig. 5.11 Comparison of Real and Simulated Throughput (TPS)

To provide further statistical validation, consider a significance level (α) at 0.05, and the null hypothesis posited a difference between the simulator's performance and that observed in actual operation. The calculated p-values for latency and throughput were 0.269 and 0.428, respectively. Given that both values exceed the threshold of α, the null hypothesis is rejected. Statistically, this outcome suggests that there is no significant difference between the simulator's performance and the real-world scenario.

Moreover, the collected data from the real blockchain platform helped in deriving two mathematical models that can be useful in predicting Blockchain latency and throughput. These models can be utilised to estimate performance parameters prior to system deployment, as described below.

For latency:

$$y = -2 \times 10^{-7} x^2 + 0.0018x + 0.8003 \tag{5.1}$$

with a coefficient of determination, R^2, valued at 0.9589, indicating a high level of explanatory power.

For throughput:

$$y = 33.554 \ln(x) - 94.886 \tag{5.2}$$

with a coefficient of determination, R^2, valued at 0.9527, also indicating a strong linear relationship between the variables.

5.4.3 Code Quality Evaluation

This section outlines the evaluation strategy used to assess the effectiveness of the simulator's implementation, as viewed by experts in IoT and blockchain, through the Goal Question Metric (GQM) approach. The GQM method provides a systematic framework for reaching specified goals using measurable data [157] and has been applied in various research domains [158]. This approach is chosen instead of the requirements-gathering method used in Chapter 4 because, at this stage, the focus shifts from understanding needs and challenges to validating the practical implementation of the simulator. The GQM approach ensures evaluation of the simulator's performance and accuracy and confirms that it meets the defined objectives. This approach is structured into three principal levels, depicted in Figure 5.12, as follows:

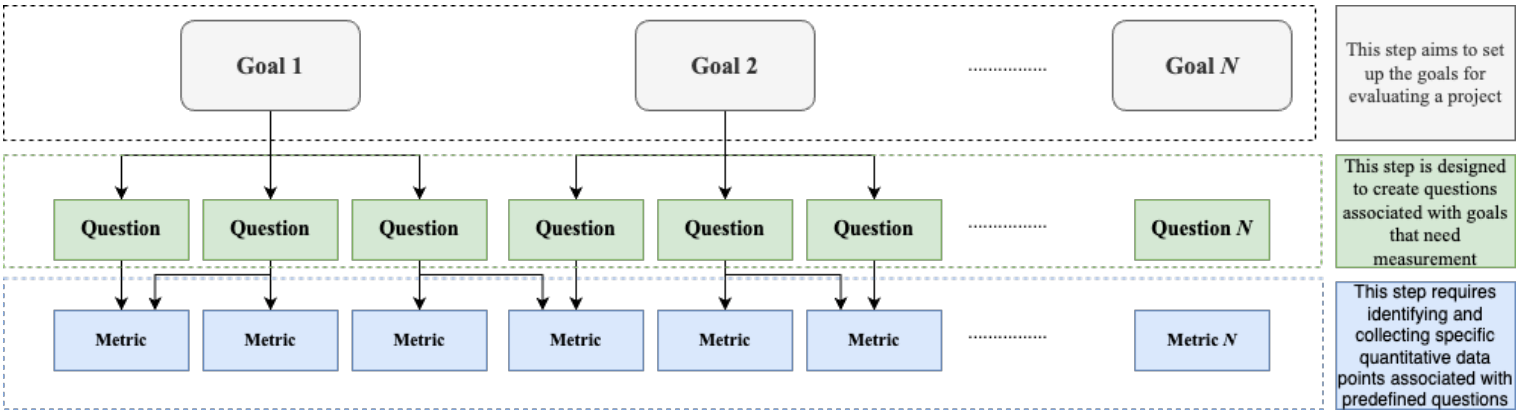


Fig. 5.12 Goal Question Metric (GQM) Hierarchy

1. **Goal Setting**: At the highest level, specific goals are defined for the process or product within a particular context from multiple perspectives for various objectives.

2. **Question development**: A set of questions is derived for each goal to characterise the degree to which the goal is achieved.

3. **Metric Identification**: At the lowest level, metrics (quantitative data) are identified for each question to provide a basis for answering them.

5.4.3.1 Defining the Goals

Our focus is on evaluating various facets of the Blockchain-based IoT simulator proposal, as viewed by experts in IoT/blockchain. These objectives are aligned with specific questions, as presented in Section 5.4.3.2 and metrics, as presented in Section 5.4.3.3 to quantitatively evaluate the proposed simulation. To define objectives, Solingen et al.[159] present a structured goal definition template that encapsulates essential elements, specific purposes, perspectives, and context characteristics. Consequently, we have established five objectives methodically, as outlined in Table 5.6.

Table 5.6 Goals Summary.

(a) User Experience Evaluation of the Configurator Component

Goal 1	Evaluate the User Experience of the Configurator Component
Purpose	Assess
Issue	Usability and coverage
Object	Of the Configurator component
Viewpoint	From the IoT/Blockchain perspective

(b) Accuracy Assessment of the Generator Component

Goal 2	Assess the Accuracy of the Generator Component
Purpose	Determine
Issue	Accuracy
Object	Of the configurations generated by the Generator component
Viewpoint	From the IoT/Blockchain perspective

(c) Efficiency Analysis of the Simulation Core

Goal 3	Analyze the Efficiency of the Simulation Core
Purpose	Assess
Issue	Performance and throughput
Object	Of the Simulation Transaction, Block Factory and Workload Feeder
Viewpoint	From the IoT/Blockchain perspective

(d) Examination of the Reporter Component Outputs

Goal 4	Examine the Reporter Component Outputs
Purpose	Examine
Issue	Comprehensibility and detail
Object	Of the reports generated by the Reporter component
Viewpoint	From the IoT/Blockchain perspective

(e) Overall Evaluation of the Blockchain-based IoT Simulator

Goal 5	Overall Evaluation of the Blockchain-based IoT Simulator
Purpose	Assess
Issue	The overall effectiveness and user satisfaction
Object	Of the Blockchain-based IoT simulator
Viewpoint	From the IoT/Blockchain perspective

5.4.3.2 Defining the Questions

For each predefined goal, we have prepared a series of questions aimed at examining different aspects of the proposed blockchain-based IoT simulator through the lens of IoT, blockchain and simulation experts. These questions are concerned with the following:

1. The usability and scope of the configurator component.

2. The precision of the Generator component.

3. The effectiveness of the Simulation Core.
4. The clarity and thoroughness of the Reporter component's outputs.
5. Overall satisfaction with the simulator experience.

Each group of questions are directly linked to a specific goal. The responses to these questions are quantified to measure the success of reaching each goal. The questions are as follows:

1. **Goal 1**
   (a) *How easy is it to set various parameters for the IoT infrastructure and the blockchain network using the Configurator component?*
   (b) *Were the provided options and settings in the Configurator sufficient for your needs?*
2. **Goal 2**
   (a) *Does the Generator component accurately reflect the configurations specified in the Configurator?*
   (b) *How would you rate the realism and functionality of the model generated by the Generator component?*
3. **Goal 3**
   (a) *How efficient do you believe the Transaction Factory and Workload Feeder components are in managing transaction flows?*
   (b) *How would you rate the sufficiency of the Blocks' data representation and structure?*
4. **Goal 4**
   (a) *Is the generated benchmark report comprehensive and clear?*
   (b) *How useful are the details provided in the "Overall result" section of the benchmark report?*

5. **Goal 5**

   (a) *In general, how would you rate the overall usability of the Blockchain-based IoT simulator?*

   (b) *How effective do you believe the Blockchain-based IoT simulator is in achieving its intended purpose?*

5.4.3.3 Defining the Metric

The participant responses are quantified and analysed to measure the overall satisfaction with the Simulator. To determine the satisfaction percentage for each predefined goal, the following steps are applied:

Step 1: *Assess Participant Feedback*: analyse participants' feedback regarding various aspects of the simulator's components for each of the 10 questions.

Step 2: *Calculate Individual Question Metrics*: For each question Q_i, where i ranges from 1 to 10, the metric $M_{i\%}$ is determined by computing the total score given by all participants P_1^n and converting it into a percentage of the maximum possible score, as illustrated in Equation 5.3:

$$M_{i\%} = \frac{\sum_{j=1}^{p} \text{Score}_{Q_i,j}}{p \times \text{Max Score}} \tag{5.3}$$

Here, $\text{Score}_{Q_i,j}$ represents the score of the participant i^{th} for the question j^{th} and Max Score is the highest possible score for the question. For the binary question, Max Score is typically 1 but may vary based on the scale used.

Step 3: *Aggregate Metric Score for Each Goal*: To compute the satisfaction percentage for each goal G_k, where k ranges from 1 to 5, the average metric score for each associated questions is calculated accordingly 5.4:

$$\bar{M}_{G_k} = \frac{1}{n_k} \sum_{i=1}^{n_k} M_i \tag{5.4}$$

where n_k is the number of questions related to a goal G_k.

Step 4:*Compile Overall Satisfaction and Dissatisfaction*: To determine the satisfaction percentage for each predefined goal, the following steps are applied 5.5 and 5.6:

$$\text{Satisfaction}_{G_k} = \bar{M}_{G_k} \times 100 \tag{5.5}$$

$$\text{Dissatisfaction}_{G_k} = (100 - \text{Satisfaction}_{G_k}) \times 100 \tag{5.6}$$

By systematically applying these equations, participant feedback is transformed into quantifiable metrics, allowing for a clear evaluation of the simulator against the predefined goals. This process ensures a more comprehensive and uniform assessment across all questions and goals.

5.4.3.4 Goal Question Metric (GQM) Conduct and Outcomes

Procedure

The following procedure was followed to execute the Goal Question Metric (GQM). First, a focus group session was held, during which the IoT-based firefighting scenario was presented to motivate the topic. Then, the participants were introduced to the Blockchain-based IoT simulation architecture and its implementation. Then, they were asked to attempt to simulate the IoT-based firefighting scenario using the simulator. Afterwards, a discussion took place to observe their views and impressions. Lastly, participants were asked to complete a survey related to the predefined goals outlined in Section 5.4.3.1.

Participants

The study involved 10 participants, including researchers, PhD students, and professionals in the domain of IoT, Blockchain and simulation topics.

5.4.3.5 Experiment Results

Table 5.7 presents the scores given by the 10 participants p_i for each question Q_i related to the five evaluation goals G_k. The headers of the columns indicate each question identifier

and the maximum possible score for it. For example, Q1/5 signifies that Question 1 has a maximum score of 5.

To illustrate the process for calculating the satisfaction and dissatisfaction rates as presented in Table 5.7, consider Goal 1 (G_1), which encompasses two questions pertaining to the Configurator Component's assessment. The score of each metric M_1 and M_2 for the first and second questions, respectively, are determined utilising Equation 5.3:

$$M_1 = \left(\frac{5+4+5+3+4+4+5+3+5+4}{10 \times 5} \right) = 0.84$$

$$M_2 = \left(\frac{1+1+1+1+0+1+1+1+1+1}{10 \times 1} \right) = 0.9$$

Upon calculating M_1 and M_2, the metric score for Goal 1 (G_1) is aggregated by applying equation 5.4:

$$M_{G_1} = \frac{M_1 + M_2}{2} = \frac{0.84 + 0.9}{2} = 0.87$$

The satisfaction and dissatisfaction percentages for G_1 are then computed using the satisfaction equation 5.5 and the dissatisfaction equation 5.6:

$$\text{Satisfaction}_{G_1} = \bar{0}.87 \times 100 = 87\%$$

$$\text{Dissatisfaction}_{G_1} = (100 - 87) \times 100 = 13\%$$

Table 5.7 Participant scores for each question related to the evaluation goals

p_i	G_1		G_2		G_3		G_4		G_5	
	$Q_1/5$	$Q_2/1$	$Q_3/5$	$Q_4/5$	$Q_5/5$	$Q_6/5$	$Q_7/5$	$Q_8/5$	$Q_9/5$	$Q_{10}/5$
1	5	1	5	4	5	5	4	3	5	4
2	4	1	4	4	4	4	5	4	4	5
3	5	1	5	3	5	5	4	5	5	4
4	3	1	3	4	4	3	4	3	4	4
5	4	0	4	5	3	2	4	4	3	4
6	4	1	4	4	4	4	3	4	5	3
7	5	1	5	3	5	5	4	4	4	5
8	3	1	3	3	4	3	5	4	3	4
9	5	1	5	4	4	5	4	5	4	4
10	4	1	4	5	4	5	4	3	5	4
M_i	**0.84**	**0.9**	**0.84**	**0.78**	**0.86**	**0.8**	**0.82**	**0.78**	**0.84**	**0.82**
Dissatisfaction$_{G_k\%}$	**13%**		**19%**		**17%**		**20%**		**17%**	
Satisfaction$_{G_k\%}$	**87%**		**81%**		**83%**		**80%**		**83%**	

Following these calculations for G_1, similar steps are taken for each of the remaining goals, as presented in Table 5.7. Table 5.7 presents the satisfaction percentages for the five evaluation goals outlined in Section 5.4.3.1. In particular, the Configurator component (G_1 5.6a) received the highest satisfaction rate at 87%, indicating its user-friendliness and coverage of set settings and available options to simulate Blockchain-based IoT. Similarly, the accuracy of the Generator component (G_2 5.6b) and the efficiency of the Simulation Core (G_3 5.6c) achieved satisfaction rates of 81% and 83%, respectively. This indicates the simulator's ability to sufficiently apply the defined configurations, execute transactions, and handle workloads as expected. However, the Reporter component (G_4 5.6d) had the lowest satisfaction percentage at 80%, suggesting that the generated reports could be improved compared to other aspects of the simulator. Despite this, the Blockchain-based IoT simulator (G_5 5.6e) achieved an overall satisfaction rate of 83%, demonstrating its potential to meet user requirements.

5.5 Conclusion

This chapter presents a novel simulation framework for modelling and evaluating the performance of blockchain-enabled IoT systems. The proposed architecture extends an IoT simulation tool, specifically IoTSim-Osmosis, to support blockchain capabilities. It enables the configuration of IoT environments and blockchain networks while achieving a seamless integration. The evaluation of the proposed simulator consists of three main parts. First, the validity of the conceptual design of the proposed simulator was revised and confirmed following experts' feedback using a focus group and a questionnaire. Second, a comparative analysis is conducted by comparing the performance metrics generated from a real blockchain platform (Quorum Blockchain platform) against their counterparts produced by the simulator. More specifically, the comparative study statistically compares the simulator and the real blockchain platform regarding two performance metrics: latency and throughput. The simulator demonstrates its ability to model and simulate realistic Blockchain-based IoT scenarios while producing reasonable performance evaluation reports. Third, IoT and blockchain experts evaluated the quality of implementation using the goal question metric approach, resulting in satisfaction rates of 80-87% across various aspects of the simulator. Overall, while the simulator demonstrates its potential, there can be areas of improvement to support extra features such as extra consensus mechanisms (e.g. Proof-of-Stake PoS) and more configuration parameters such as extended transaction payload, gossip protocols, and others. While challenging, the simulator can also be extended to support expressing and executing the IoT business logic as smart contracts to be deployed to simulated virtual machines for each blockchain node. Moreover, it is challenging for simulators to cope with the advancement pace of Blockchain technologies. For instance, Ethereum has transformed from dependency on PoW as a consensus protocol to PoS. Therefore, the next chapter proposes a middleware that facilitates the usage of the simulator for evaluating the performance of real blockchain platforms, where all missing features in the simulator can exist and be accessed, such as smart contracts.

Chapter 6

IoT Simulation for Evaluating Blockchain Performance: A Middleware Architecture

Summary

Recently, there has been an increasing interest in exploring the potential of Blockchain for serving several IoT-based domains. While it can be easy to access several open-source Blockchain platforms (like Hyperledger Fabric), this is not always the case with large-scale IoT infrastructure. To compensate for this shortcoming, IoT simulation can be a viable option in many cases. The previous chapter, Chapter 5, proposed a simulator approach for evaluating modelled blockchain-based IoT scenarios. As outlined by the conclusion of the previous chapter, simulation cannot keep rapid pace with the advancement of blockchain technology. In theory, utilising IoT simulators with real blockchain platforms makes it possible to remedy simulation shortcomings such as access to smart contracts. However, the question remains of how to address the disparity between the distinctive environments of IoT simulators and blockchain platforms.

This chapter introduces a novel middleware architecture that overcomes challenges associated with the interaction between blockchain platforms and IoT simulators. The middleware is tailored to meet the distinct operational facets of blockchain platforms and IoT simulators, enabling thorough evaluations of real blockchain-based solutions in simulated

IoT scenarios. We present a case study where the middleware connects IoTsim-Osmosis, an IoT simulation tool, with the Hyperledger Fabric blockchain platform. This chapter assumes a hypothetical scenario that requires evaluating the viability of employing blockchain for an IoT-based firefighting system at a large scale, which uses the middleware to enable interaction between real blockchain deployment and a simulated IoT model.

The remainder of this chapter is structured as follows: Section 6.1 provides a brief introduction to this chapter and highlights the challenges it seeks to address. Section 6.2 explains the Research Question (RQ), the contributions of this chapter, and its relevance to the published paper. Section 6.3 presents the proposed middleware architecture while detailing its key components. Section 6.4 evaluates the middleware by validating the correctness of the concurrent storage mechanism and demonstrating the integration with Hyperledger Fabric and IoTsim-Osmosis through a use case evaluation. Finally, Section 6.5 concludes the chapter and outlines future research directions

6.1 Introduction

Blockchain and the Internet of Things (IoT) are emerging technologies that each can promise to reshape today's business problems and available solutions in an unprecedented manner. One can envision all sorts of opportunities and benefits of merging IoT and Blockchain across various domains, including healthcare [153], supply chain [154], and logistics[155]. For instance, this chapter is based on a scenario depicted in Figure 6.1, where a fire station utilizes IoT technology to enhance response times to fires and reduce their severity. Smart homes are equipped with relevant sensors that can detect fire incidents and autonomously transmit alerts to fire stations over the Internet. Given the Blockchain features listed in section 2.1.2, the fire station authority opts for Blockchain as a trusted messaging infrastructure between the fire station and thousands of connected homes.

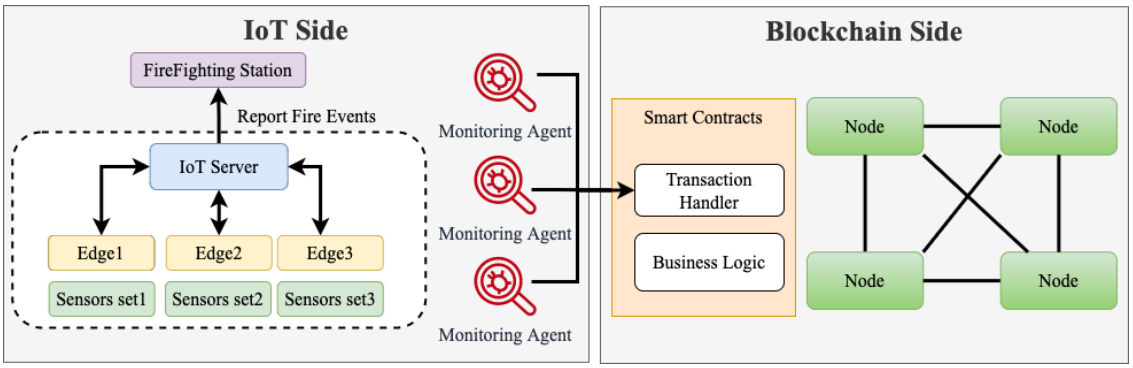


Fig. 6.1 Motivating scenario

Considering the criticality and complexity of the system, as well as the heterogeneity of the underlying technologies, it is vital to assess the viability and technology readiness level (TRL) before the production stage and actual operation [160]. From an evaluation perspective, experimenting with such a complex and heterogeneous system needs access to a blockchain platform and an IoT infrastructure. As with blockchain, the open nature of most blockchain platforms and simulators enables experimenting with virtually any blockchain-based scenario.

From the IoT side, however, implementing an IoT scenario (e.g. the firefighting scenario 6.1) can be difficult due to the restricted availability of IoT infrastructure. To appreciate the accessibility challenge, consider an evaluation goal where an experiment aims to assess the blockchain performance under severe conditions in which all the connected houses simultaneously emit fire alerts to the fire station. However, limited access to a sufficient IoT infrastructure will likely hinder the accomplishment of reliable and effective research conduct. To compensate for this shortcoming, IoT simulators, such as those listed in Table 2.2, can serve as a suitable alternative for modelling IoT systems, including physical devices, edge computing, networks, data centres, and cloud services [161]. Nevertheless, none of the existing IoT simulators is equipped with the capability to connect and interact with real blockchain platforms. While there can be great value in utilising simulated IoT environments with real blockchain platforms, we also recognise the challenges associated. Therefore, this chapter investigates these challenges and proposes a middleware architecture that closes the gap between the distinctive environment of IoT simulation and real blockchain runtime.

This chapter aims to enable IoT simulators to model IoT infrastructure and generate the workload necessary to benchmark essential blockchain performance metrics such as throughput, latency, and transaction success/failure rates. For demonstration purposes, this chapter selects IoTSim-Osmosis [16] for the IoT simulation side, whereas Hypereledger Fabric (HLF) is selected as a real-world blockchain platform. This selection is because of the support for Java programming language by both, which aids in demonstrating the proposed middleware approach. Formally put, this chapter proposes a middleware architecture to achieve seamless integration and enable communication and transactions between both ends. It demonstrates the middleware for using a modelled IoT infrastructure to benchmark the blockchain-based scenario performance presented in Figure 6.1. Given the distinctive execution environments between IoT simulators and real-world blockchain platforms, this chapter verifies the middleware's correct behaviour and its capability to handle related challenges, such as concurrency and race conditions.

6.2 Research Questions, Contributions, and Relevance to Published Work

Section 6.2.1 provides a detailed explanation of the Research Question (RQ) and highlights the contributions associated with this chapter. Section 6.2.2 clarifies the relevance of this chapter to the published paper, as outlined by the publications listed in Section 1.4.

6.2.1 Research Question and Contribution

Research Question 3 (RQ3): For performance evaluation purposes, not all blockchain features (i.e. smart contracts) can be perfectly simulated for every scenario. A set of real Blockchain platforms exists that are open source, accessible, and can be deployed to scalable cloud computing resources. However, large-scale IoT infrastructures are not easily accessible for research and development purposes. Accordingly, how feasible is it to utilise

IoT simulators as workload generators for benchmarking the performance of real blockchain platforms?

To answer this question, this thesis contributes a simulation framework for evaluating the performance of blockchain-based IoT ecosystems. This chapter contains a middleware architecture that enables an IoT simulator (IoTsim-Osmosis) to evaluate the performance of a real blockchain platform (namely, Hyperledger Fabric). The proposed middleware architecture attempts to resolve major challenges associated with the distinction between simulated execution environments (IoT simulators) and real execution environments (Blockchain platforms). Therefore, it enables connectivity and communication between both ends to enable utilising IoT simulators (i.e. IoTsim-Osmosis [16]) as workload generators to benchmark the performance of a real blockchain platform (i.e. Hyperledger Fabric [17]).

6.2.2 Relevance of the Chapter to the Published Paper

This chapter has been submitted to the *Blockchain: Research and Applications* journal and contains the main content presented in this chapter.

6.3 Middleware Architecture

6.3.1 Research Problem

Assume that IoT simulation is considered for evaluating and benchmarking the performance of a real blockchain platform. Therefore, as Figure 6.2 illustrates, this chapter suggests the utilisation of a middleware solution for integrating between two distinctive environments, which are a real blockchain platform (namely Hyperledger Fabric [17]) and an IoT simulator (namely IoTsim-Osmosis[16]). See section 2.4.3 for the rationale behind this selection.

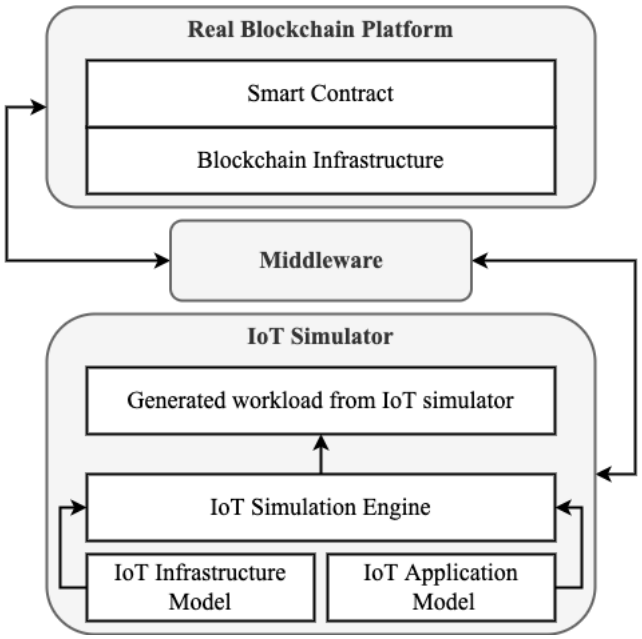


Fig. 6.2 Middleware to integrate an IoT simulation with a real blockchain platform

The aim is to utilise IoT simulators as workload generators for evaluating real-world blockchain platforms based on simulated IoT models. However, the difficulty of bridging distributed environments (like Hyperledger Fabric) with simulated environments in IoT simulators (such as IoTsim-Osmosis) remains a persistent challenge. The integration challenges are not limited to the IoT simulator's ability to communicate and interact with the blockchain platform but also extend to the distinctive nature of both sides. Several IoT simulators are concerned with a discrete event representation throughout a virtually short time. These simulators rely on predefined calculations and execution duration to generate their output.

Consequently, a naive integration between these two disparate and heterogeneous execution environments fails to align IoT simulators with the real blockchain platform. An example of a naive integration is the tight coupling of the IoT simulator with the blockchain platform, which will negatively influence the execution run-time of IoTsim-Osmosis. Consider that the IoT simulator must be stopped whenever it interacts with the blockchain. As a result, the IoT simulator, *being a discrete event simulator*, will demonstrate miscalculations, resulting in unrealistic outputs. Thus, the middleware must take into account the distinctive

nature between IoT simulation and real blockchain platforms to ensure accurate performance evaluation and benchmarking.

6.3.2 Proposed Architecture

The primary focus of this chapter lies in presenting a middleware architecture along with a reference implementation that aims to bridge IoT simulations with actual blockchain platforms. As illustrated in Figure 6.3, IoTsim-osmosis is selected as an IoT simulator, while Hyperledger Fabric is selected as a real blockchain platform. The main functionality of the middleware is to provide IoT simulators with the ability to do the following:

1. Acts as a workload generator to benchmark and evaluate the responsive blockchain platform.

2. Transact with the respective blockchain platform, which requires authentication, connectivity, communication, and transaction fail-safe mechanism.

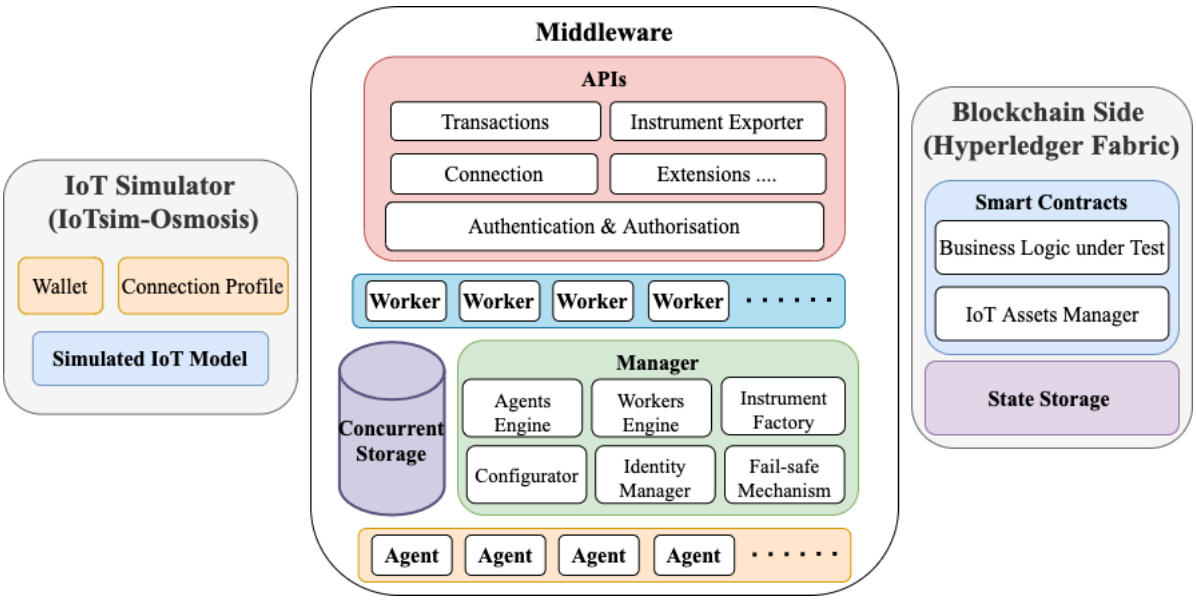


Fig. 6.3 A Middleware for Leveraging IoT Simulators to Evaluate Blockchain Performance.

The middleware architecture adopts the separation of concerns principle to address the distinction between the IoT simulator and the blockchain platform. The middleware

architecture demonstrates that by designating two separate but collaborative entities: agents and works. While each interfaces with a distinctive environment, they interact with each other through concurrent storage. On the one hand, *agents* are designed to consider the nature of the simulated environment. Hence, they interface with the simulator to gather data from the simulated IoT environment and store them in concurrent storage. The behaviour of these agents, such as how frequently and periodic data are gathered, is not specifically mandated and can be instructed as required by the middleware manager. *workers*, on the other hand, are designed to handle the nature of the blockchain platform. Hence, their primary role involves consolidating the information collected by agents from the simultaneous storage and sending it as transactions to the blockchain. Therefore, the middleware provides a set of APIs (application programming interfaces) that the workers can leverage when transacting with blockchain. For example, workers must be authenticated and connected to the blockchain platform to be able to submit transactions. The following sections provide a deeper look at key aspects of the proposed middleware architecture.

6.3.3 Smart Contracts and State Storage

From the blockchain side, middleware interfaces with various smart contracts that fall into two main categories: the IoT Asset Manager and the business logic under test. The former is dedicated to representing, defining and managing IoT assets and their properties. For example, within the context of firefighting using IoT technology, an asset could refer to a residence containing attributes like homeowner details, physical address, and geographical coordinates. The middleware utilises the IoT Asset Manager to execute Create, Read, Update, and Delete operations (CRUD) methods at the Blockchain state storage to manage IoT assets. The second category of smart contracts represents the business logic under test. For instance, the scenario presented in Figure 6.1 assumes fire alerts are submitted from IoT sensors through monitoring agents to the blockchain platform for processing and further actions. Therefore, the middleware interfaces with associated smart contracts and stresses them with the required workload from the IoT simulator in the form of submitted transactions.

6.3.4 Identity Management

Hyperledger Fabric operates as a blockchain platform with permissioned access. Thus, a cryptographic wallet is instrumental in connecting and facilitating communication between the middleware manager, workers, and the blockchain network. For instance, it allows workers to use their specific identities to compose and endorse transactions during smart contract executions. As depicted in Figure 6.4, the wallet creation requires registration and enrollment to the blockchain platform. The initial step involves registering a middleware's admin identity, which produces an ID and a secret key. Subsequently, the middleware submits these credentials to a Certificate Authority (CA) within the blockchain segment for self-enrollment. The CA assesses these credentials and assigns an admin identity to the middleware, which features administrative capabilities that facilitate the registration and enrollment of workers onto the blockchain network. The Identity within this context combines public/private key pairs encapsulated within a digitally signed X.509 certificate. The admin can then use their identity to generate another identity sufficient for workers' purposes.

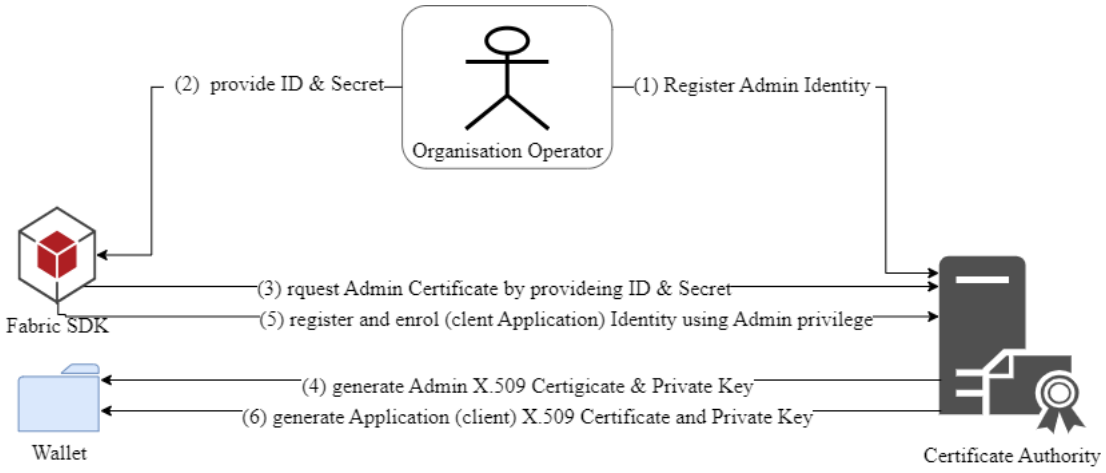


Fig. 6.4 Authenticating Workers to the Blockchain Platform

6.3.5 Transactions Fail-safe Mechanism

To ensure the integrity of the evaluation results, the middleware incorporates a fail-safe mechanism for handling possible transaction failures. As illustrated in Figure 6.5, the

middleware assigns a maximum number of attempts (e.g., $trials_{max} \leftarrow 5$). This is particularly useful in scenarios where, despite multiple attempts, a transaction fails to be accurately processed due to external factors such as network or blockchain environment. The middleware closely monitors transaction events on the blockchain for every smart contract call. It verifies whether a transaction is successfully executed and recorded on the ledger. If a transaction fails, the middleware attempts resubmission until the maximum number of attempts is reached. This aids in making informed decisions regarding the validity of the experiment and thus avoiding possible deviations from the anticipated behaviour.

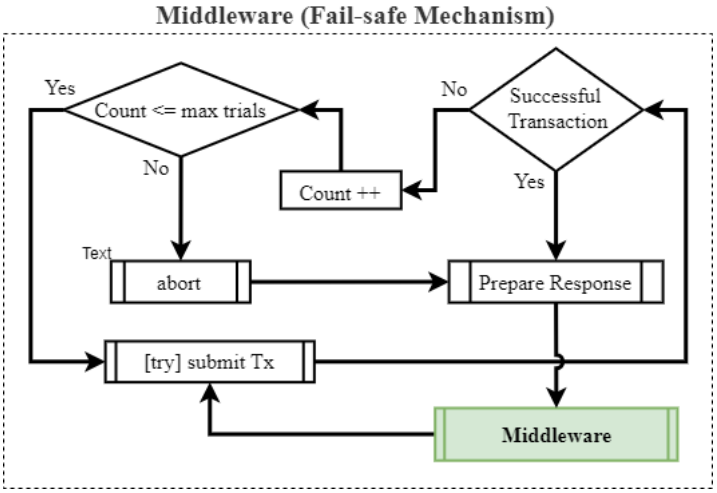


Fig. 6.5 A fail-safe mechanism for handling possible transaction failures

6.3.6 Thresholds Specifier

The middleware considers a specifier that enables defining a set of thresholds against metrics generated by the IoT simulator. The threshold specifier supports basic comparative operators against a specific value such as greater than, less than, equal to, not equal to, and not. For example, assume a threshold requirement υ_i to be $fireAlert \neq false$. Therefore, the *Thresholds Specifier* enables flirting the generated metrics accordingly. Such a threshold requirement can be used for several reasons. For example, agents can instruct them to capture metrics only when a fire event occurs. Additionally, agents can be instructed to classify generated metrics as positive or negative based on this threshold. As shown in Algorithm 1,

the middleware sets a specific threshold requirement υ_i. It also decides whether υ_i must be stored on both ends, the simulator and the blockchain, when the flag is set to true ($\rho = true$); otherwise, υ_i is only stored locally. Storing υ_i on the blockchain's state storage involves activating the IoT assets manager smart contract. If the transaction for creating it is successful, the middleware creates a copy of the threshold requirement υ_i' in the local storage; otherwise, it halts the entire process.

Algorithm 1: Threshold Specifier Creation.

Input : υ_i //Threshold level
ρ //whether υ_i is required in both sides
maxRetry //Max number to submit transactions

Output : whether υ_i and υ_i' are successfully created.

```
1  η := 0 //attempts count to submit transactions
2  if (ρ) then
       // Blockchain transaction is required
3      α := false //Initially, set the transaction status flag to false.
4      while (¬α) and (η ≤ maxRetry) do
5          Submit υ_i to the blockchain level.
6          η := η + 1
7          if υ_i then
               //successfully created
8              α := true
9          end
10     end
11 end
12 if α or ¬ρ then
       //Successful Blockchain transaction or not required
13     Create υ_i' at the simulator level.
14 end
```

6.3.7 Agents

As mentioned in section 6.3.2, the middleware provides agents that interface with the IoT simulator. Agents can be deployed wherever metrics $\{m_1, ..., m_n \in M\}$ are generated from the underlying simulator. For instance, a generated metric m_i can be the status of an alarm sensor, which can be positive (fire event) or negative (no fire event). Agents are aware of specified thresholds, which are the ones created by Algorithm 1. Each agent is assigned a set of generated metrics $\{m_i \in M\}$. Multiple agents can be deployed at the IoT simulator level to capture their assigned metrics once produced by the IoT simulator. Agents instantly evaluate captured metrics against a specified threshold $\{\upsilon_i \in \Upsilon\}$. Agents provide feedback on the evaluation by reporting the outcomes and determining whether a specific metric is labelled as a breach B or compliant (C). Algorithm 2 illustrates the process of examining generated metrics. For simplicity, the algorithm conveniently uses the increment notion ($C++$ or $B++$) to express the ability of agents to classify generated metrics and store them accordingly.

The middleware recognises that processing Blockchain transactions introduces an unnecessary delay to the simulator's execution runtime. Hence, it considers preventing the negative impact of the blockchain transaction life cycle on the performance and dependability of the IoT simulator underneath. While agents are loosely coupled to the IoT simulator, they are completely decoupled from the blockchain platform. Consequently, agents interface with the local concurrent storage instead of establishing direct communication with the associated blockchain platform. The concurrent storage maintains a set of specified threshold requirements $\{\upsilon_i \in \Upsilon\}$ along with their indicators sets, which are B set for breach instances to υ_i or C set for compliance instances with υ_i. See Figure 6.6 for a visual elaboration.

Algorithm 2: Agents' Evaluation Behaviour

Input : υ_i //Threshold requirement

m_i //Generated metric

Output : C or B //compliance or Breach

$l := \upsilon_i(level)$ //Threshold level (e.g. GraterThan, LessThan, Equals, Not)

$v := \upsilon_i(threshold)$ //threshold value

if $(l = GraterThan \text{ and } m_i < v)$ *or* $(l = LessThan \text{ and } m_i > v)$ *or* $(l = Equals \text{ and } m_i \neq v)$ *or* $(l = Not \text{ and } m_i = v)$ **then**

  $B := B + 1$

end

else

  $C := C + 1$

end

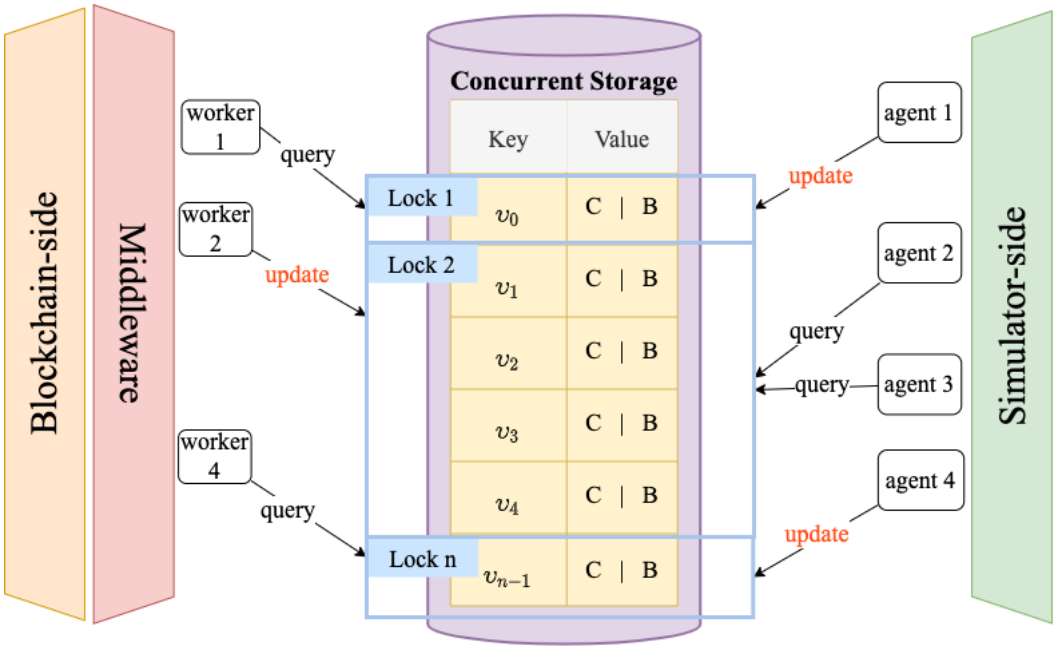


Fig. 6.6 The locking mechanism on the concurrent storage for agents or workers

6.3.8 Workers

The middleware schedules a pool of concurrent entities called workers for blockchain benchmark and evaluation purposes. These workers engage with and conduct transactions on the blockchain side. The primary responsibility of the main worker is to apply stress to the smart contract being tested. Recall that the local storage is concurrently shared between agents and workers (See Figure 6.6). Therefore, workers repeatedly examine the local storage to check whether there is an update by agents on the C and B sets associated with every threshold requirement $\{\upsilon_i \in \Upsilon\}$. If any employees try to submit the most recent update to the relevant smart contract, they will update the status of both the C and B sets in the state storage. This update reflects the difference between the workers' most recent values and their reported values.

Algorithm 3 offers a detailed insight into the operational dynamics of workers. Initially, the middleware sets up a worker with specific attributes: (i) the designation of a smart contract under test, (ii) a particular method (or function) within that smart contract designated for transaction execution, such as "report fire event," and (iii) a set of threshold criteria $\{\upsilon_i \in \Upsilon\}$. Workers refrain from transacting with the blockchain unless there is a status change concurrent storage concerning their assigned υ_i. This is important to mitigate the unnecessary load on the blockchain infrastructure that might cause undesired miscalculations. As stipulated in Algorithm 3 Line **4**, workers proceed with submitting B or C data to the blockchain is contingent upon the condition that $B > 0$ or $C > 0$. When the transaction is successful, the worker reflects that on the B and C values at the concurrent storage.

Algorithm 3: Workers Behaviour

Input : A set $\{contract, method, Q\}$

Output : Updated B and C sets for each $\upsilon_i \in \Upsilon$

for ***all*** υ_i *in* Υ **do**
  $B_{initial} := B$ //Initial Breach Set state
  $C_{initial} := C$ //Initial Compliance Set state
  if $B_{initial} > 0$ *OR* $C_{initial} > 0$ **then**
    Execute Transaction (contract, method, υ_i, $B_{initial}$, $C_{initial}$) //Dispatch metrics to the Blockchain
    if *transaction_success* = *true* **then**
      Transaction Executed Successfully
      $B_{final} := B$ //Updated Breach Set state
      $C_{final} := C$ //Updated Compliance Set state
      $\Delta B := B_{final} - B_{initial}$ //Compute Breach Set state change
      $\Delta C := C_{final} - C_{initial}$ //Compute Compliance Set state change
      if $\Delta B \geq 0$ **then**
        $B := \Delta B$ //Adjust and renew Breach Set
      end
      else
        $B := 0$ //Reset Breach Set
      end
      if $\Delta C \geq 0$ **then**
        $C := \Delta C$ //Adjust and renew Compliance Set
      end
      else
        $C := 0$ //Reset Compliance Set
      end
    end
  end
end

6.3.9 Storage Concurrent Locking Mechanism

Workers reflect successful transactions by updating B or C sets, which is not trivial due to the operations overlap between agents and workers on these shared sets. Agents continuously update these datasets with new metrics collected from the IoT simulator, while workers aim to denote submitted datasets as being successfully submitted to the blockchain. Specifically, a significant delay occurs when a worker evaluates B or C sets from t_1 to the point it concludes a transaction with the blockchain at t_2. During this interval, it's probable that agents or other workers have altered the state of B or C datasets. To mitigate potential race conditions and ensure data integrity, the middleware employs a concurrent locking mechanism designed to handle concurrent modifications effectively:

1. Only one entity, whether an agent or worker, is granted the authority to modify (update) any C or B dataset at any given time.

2. During the period $\Delta t = t_2 - t_1$, agents are allowed to update B or C datasets with new metrics.

3. A worker, after finalizing a transaction on the blockchain, is restricted from updating the datasets associated with $\{\upsilon_i \in \Upsilon\}$ until ensuring no other entity currently possesses the lock. Once the lock is available, the worker can secure it to incorporate the changes ΔB and ΔC, thus marking them as reported and avoiding redundant submissions.

4. To maintain accuracy and prevent errors, workers refrain from making changes ΔB and ΔC to the storage if the values are negative. This indicates that there are no actual differences to record.

The middleware organises concurrent storage as key-value pairs, the key representing a specific threshold requirement $\{\upsilon_i \in \Upsilon\}$ and the value encompassing the related metric datasets B and C. This storage system incorporates a locking mechanism that selectively authorises write operations (including insertions, deletions, and updates) to a single entity at a time while permitting unrestricted read access to any entity. To optimise performance, the storage is divided into segments to enable write locks to be implemented for each

segment individually rather than for the entire storage space. This segmentation facilitates simultaneous write operations by multiple entities on different parts of the storage to enhance throughput and overall system efficiency. Figure 6.6 demonstrates how storage segmentation and locking mechanisms work and present the interactions between agents and workers.

1. In Segment 1, agent 1 secures a lock to input data related to breaches (B) and compliances (C) for metrics compared against υ_0. Worker 1, on the other hand, can examine the most recent data for υ_0 without any obstacles since the lock doesn't limit reading operations.

2. Segment 2 is locked by worker 2 following a successful metric transaction concerning a group of threshold requirements $\{\upsilon_1, \upsilon_2, \upsilon_3, \upsilon_4\}$, intending to update ΔB and ΔC accordingly. Agents 2 and 3 can still access data within this segment for query purposes, avoiding operational conflict.

3. Should a worker or agent need to perform a write operation on a specific segment, it must wait for the existing lock to be released before acquiring the lock for its use.

6.3.10 Smart Contract Benchmarking Functionality

The middleware incorporates a built-in benchmarking tool for capturing critical data for assessing blockchain performance. It monitors fundamental performance indicators for each worker's transaction, including throughput, latency, success, and failure rates. The benchmarking mechanism follows the performance metrics assessment guidelines proposed by the Hyperledger Performance Working Group [64]. The subsequent sections delineate the processes involved in the indicators instrumentation, data collection, exportation, analytical evaluation, and graphical presentation.

6.3.10.1 Composing and Exporting Performance Instruments

Figure 6.7 depicts assembling and exporting performance metrics for each smart contract method invoked by workers. A dedicated module, termed *Instrument Exporter*, leverages

Micrometer[1]—an open-source, vendor-neutral tool that integrates with a broad spectrum of monitoring solutions. The *Instrument Exporter* creates a series of metrics that the middleware utilizes to evaluate the smart contract's performance during testing. These metrics encompass successful transactions (T_s), unsuccessful transactions (T_f), and transaction delay (T_d).

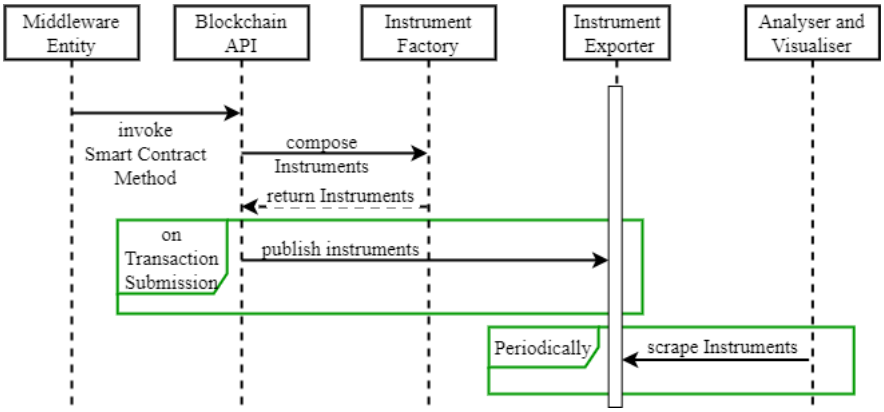


Fig. 6.7 Process of assembling and exporting metrics

As outlined in Algorithm 4, the middleware avoids repeated instrumentation by verifying whether the smart contract has been instrumented. If not, the *Instrument Exporter* initiates a fresh set of metrics, including T_s for successes, T_f for failures, and T_d for latency measurements. For each transaction a worker initiates, the middleware updates either T_s or T_f accordingly. The transaction latency T_d is determined by the elapsed time, calculated 6.1

$$T_d = timer_{end} - timer_{start} \tag{6.1}$$

where tx_{start} marks the transaction's submission to the blockchain and tx_{end} the moment the transaction completion. It's important to note that the calculation of T_d intentionally omits unsuccessful transactions due to the irrelevance to the latency indicator.

[1]https://micrometer.io

Algorithm 4: Instruments Composition

Input : $\{contract\}$

Output : Count of successful transactions $T_{success}$,

Count of failed transactions T_{fail},

Duration of transactions $T_{duration}$

Initialize $Methods := \{method_i | i \in \mathbb{N}\}$ //Enumerate all methods within the smart contract

for ***every*** *invocation of the smart contract* **do**

  if $method \notin Methods$ **then**

    Assemble monitoring tools $\{T_{success}, T_{fail}, T_{duration}\} \in method$

    Integrate $contract$ into $Methods$

  end

  Start $Timer$

  Initiate $Transaction$ to $method$

  if $Transaction$ *executes successfully* **then**

    Stop $Timer$

    $T_{duration} = Timer_{stop} - Timer_{start}$

    $T_{success} := T_{success} + 1$

  end

  else

    $T_{fail} := T_{fail} + 1$

  end

  Dispatch $\{T_{success}, T_{fail}, T_{duration}\}$ to HTTP server

end

6.3.11 Instruments Gathering, Calculation, and Visualisation

The middleware orchestrates the integration of three key components to collect and visualize the performance indicators $\{T_s, T_f, T_d\}$, which are: (i) An HTTP server tasked with the export

of metrics for each smart contract's method, (ii) Prometheus[2] for periodic collection of the indicators from the HTTP server, and (iii) Grafana[3] to compile the benchmarking data and facilitate their visualization. Accordingly, a series of benchmarking parameters can be established to evaluate the smart contract's performance as delineated in the following equations:

$$Avg_{tps} = \frac{\sum_i^n T_s}{\sigma} \tag{6.2}$$

$$Avg_{latency} = \frac{\sum_i^n T_d}{\sum_i^n T_s} \tag{6.3}$$

$$s_{rate} = \frac{T_s}{T_s + T_f} \times 100 \tag{6.4}$$

$$f_{rate} = \frac{T_f}{T_s + T_f} \times 100 \tag{6.5}$$

Avg_{tps} quantifies the transaction throughput, defined as the sum of successful transactions over σ. σ is defined as $lct - fst$, where fst is the timestamp of the first transaction that was successfully processed, and lct is the timestamp of the final transaction that was committed to the blockchain ledger. s_rate and f_rate indicate the proportions of successful and unsuccessful transactions, respectively, among the total number of transactions. The average latency $Avg_{latency}$ is determined by aggregating the total time to execute each transaction and dividing this by the number of successful transactions.

[2]https://prometheus.io
[3]https://grafana.com

6.4 Evaluation

6.4.1 Validating the Concurrent Storage

The correctness of the concurrent storage mechanism is crucial for the overall performance and reliability of the proposed middleware. Therefore, we designed a controlled experiment, as detailed in Algorithm 5, to validate its functionality and correctness. Table 6.1 illustrates the configuration settings for conducting the controlled experiment. Given the concurrent nature of the shared storage and the overlapped operations between agents and workers, the experiment focuses on the storage's ability to store generated metrics correctly. These metrics refer to the data points (e.g. values) that are measurable performance indicators generated by the IoT simulator during the experiment run. These generated metrics in our scenario categorize fire alerts as either compliant or in breach based on their relation to the threshold (υ) and store them in the concurrent storage.

To validate the correctness of the concurrent storage, we experiment with several iterations. For each iteration, agents (a_i) evaluate a set of metrics with increasing size from $|M| = 10^2$ to $|M| = 10^6$ (x ranges from 2 to 6). In each iteration, we increase the number of metrics produced to validate and evaluate the concurrent storage mechanism's scalability to ensure it maintains performance and reliability under varying load conditions. Therefore, agents update the concurrent storage accordingly as defined in Algorithm 2. Workers (w_i) process existing data in sets B and C and apply necessary updates. Note that there are delays introduced to control the experiment with regard to agents and workers. Agents initially wait 1 second before commencing their tasks. Then, they gather metrics from the simulator whenever generated. On the other hand, workers initially wait for 2 seconds and then periodically wait for 3 seconds to emulate the delay caused by the blockchain transaction processing. During the waiting period of a worker, agents are expected to gather as many generated metrics and store them in the concurrent storage.

Table 6.1 Test-bed Configuration for Concurrent Storage Validation

Parameter	**Quantity**	**Initial Delay (x=2 only)**	**Periodic Delay**
Agents	1	1 second	0 seconds
Workers	3	2 seconds	3 seconds
Generated Metrics	10^x ($x \in \{2,3,4,5,6\}$)	N/A	N/A
Compliance Rate	50%	50%	50%
Breach Rate	50%	50%	50%

Algorithm 5: Controlled Experiment on Core Functionalities

Input : A collection of metrics M, where each $m_j \in M$ is a rational number,
Agent a_i,
Worker w_i,
Threshold υ,
Initial total metrics assessed by agent a_i set to $tma = 0$,
Initial total metrics handled by worker w_i set to $tmw = 0$

Output : Guaranteed integrity of parallel storage and accurate functionalities

Establish threshold υ as $Latency \leq 1$
Initialize breach and compliance counts, $b = 0$ and $c = 0$ respectively
Apply threshold υ to worker w_i
Set counter $x = 100$
while $x < 1000000$ **do**
    Initialize metric counter $j = 0$
    while $j < x$ **do**
        if $x \leq 100$ **then**
            Pause for visual clarity
            Wait for 1 second
        end
        if $(j \bmod 2) = 0$ **then**
            Generate m_j randomly from $\mathbb{Q}$ with $m_j \leq Latency$ //indicating compliance.
        end
        else
            Generate m_j randomly from $\mathbb{Q}$ with $m_j > Latency$ //indicating breach.
        end
        Delegate metric m_j to agent a_i Increment j by 1
    end
    Calculate difference $\alpha := tma - tmw$
    Calculate sum $\beta := tma + tmw$ **if** $\alpha = 0$ *AND* $\beta = 2x$ **then**
        Condition met for the pass.
    end
    else
        Condition met for failure.
    end
    if $b \neq \frac{tma}{2}$ *OR* $c \neq \frac{tma}{2}$ **then**
        Misclassification results in failure
    end
    else
        Accurate classification results in success
    end
    Update x by multiplying by 10
end

The experiment's success relies on several criteria:

1. **Agent Evaluation**: Each agent (a_i) must evaluate all metrics ($tma = 10^x$) and correctly update sets B and C in the storage.

2. **Worker Processing**: Workers (w_i) must process all generated metrics ($tmw = 10^x$) without redundancy.

3. **Classification Accuracy:** The number of "compliance" C and "breach" B entries in the storage should reflect the actual metric distribution (50% each).

4. **Synchronization**: The difference between agent-processed metrics (tma) and worker-processed metrics (tmw) should be zero ($\alpha = 0$) upon completion of each iteration, indicating proper synchronization.

Figure 6.8 showcases the validation for the first iteration (where $x = 2$) with 100 generated metrics. The experiment successfully demonstrates the middleware's ability to handle the workload properly. While subsequent iterations (where $x > 2$) are not visualized here, the open-source code allows for replication and verification (link to GitHub repository [4]).

[4]https://github.com/aakzubaidi/BMBmid-Middleware

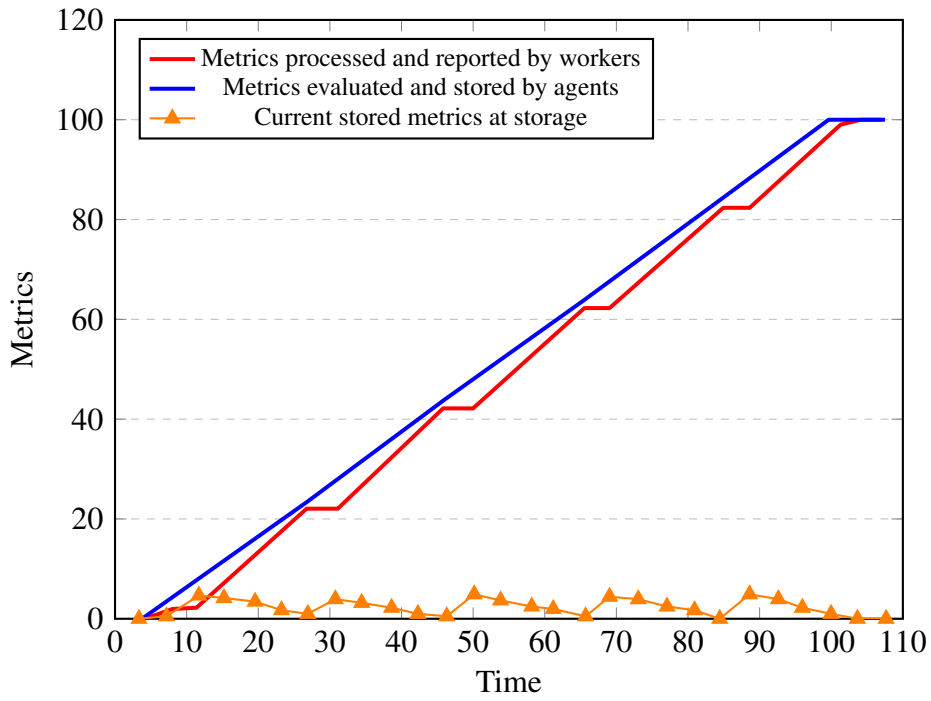


Fig. 6.8 Illustrating the proper execution of operations on concurrent storage through the utilization of 100 generated metrics

6.4.2 Use Case Evaluation

As visualised in Figure 6.1, assume a large number (i.e. 30,000) of smart homes are connected to a blockchain platform and report their fire status (fire or no fire) to the respective smart contract. The large scale of the IoT infrastructure is challenging to access for research and development purposes. IoT simulators such as IoTSim-Osmosis can be a viable option given that the middleware can connect it to a real blockchain platform and synchronise between a simulated model and a real blockchain platform. The following details the incorporation of the middleware with the IoTSim-Osmosis simulator to evaluate the performance of the real blockchain platforms.

6.4.2.1 Blockchain Platform

For this evaluation, we leveraged the Hyperledger Fabric v2.3.2 blockchain platform. The deployment utilized a cloud-based infrastructure provisioned with 32 vCPUs (Intel Xeon

Gold 6140 processors clocked at 2.30 GHz) and 64 GB of RAM. These resources are equally distributed to 4 validating nodes and 5 orderer nodes. The consensus mechanism employed is Raft, a crash-fault tolerant mechanism recommended by the Hyperledger Community. A copy of a smart contract under test is deployed to each validating node. The smart contract mainly persists v_i as outlined in Algorithm 1, and receives transactions about the fire status of each home from an authenticated and authorised client.

6.4.2.2 Simulated IoT Model

We employ the IoTSim-Osmosis simulator [16] to model the IoT-based firefighting architecture, as illustrated in Figure 6.1. The modelled IoT architecture is organised into two primary layers: (i) three geographically distinct edge data centres, each serving different areas, and (ii) a network consisting of 30,000 smart houses organised into groups of 10,000 houses, with each group being connected to a designated edge data centre. Table 6.2 highlights the identical specification of each IoT edge centre. Each house has sensors that generate random fire status (positive or negative).

Table 6.2 Configurations and Expectations

Description	**Configuration**	
Generated Fire Statuses	30,000, such that one fire status per smart home	
Edge data centres	**3 edge centres, each has**:	
	CPUs:	4
	Bandwith:	100 Mbps
	RAM:	4 GB
	MIPS/CPU:	250
	Storage:	200 GB
Agents Allocated per Edge Data Center	1	
Fire Status Reporting Frequency	1 per 100ms for each smart home	
Agent Execution Frequency	100ms	
Worker Execution Frequency (no initial delay)	1s	
Expected Throughput (TPS)	1	
Fire Statuses Updated per Transaction	≈ 30	
Expected Total Transaction Number	≈ 1000	

6.4.2.3 Middleware Configurations

The middleware is treated as a library imported into the simulator's framework to connect the IoT simulator to the blockchain platform. The configuration of the middleware involves key elements as follows:

1. The address (URI) of the certificate authority and its corresponding TLS certificate for secure communication.

2. Unique identification details (ID and secret) issued by the certificate authority for the simulator.

3. A specified location for storing generated identities within the simulator.

4. A connection profile for the blockchain network, allowing the simulator to discover and connect to relevant peers.

5. The name of the designated channel within the blockchain.

6. The name of the smart contract is used for the test.

7. A threshold requirement υ_i, which is set to $fireAlert \neq false$.

For preset expectations, Table 6.2 controls the experiment environment. Figure 6.9 visualise the workload flow from smart homes to the respective smart contract. An agent is deployed for each of the three data centres, which are concerned with intercepting the fire status of their associated smart homes. As in Algorithms 2, Agent are responsible for classifying the intercepted fire status based on the defined threshold υ_i, and storing the outcome into the local concurrent storage. For consistency and predictability, The frequency of creating a fire status for each smart home is limited to one transaction every 100 milliseconds. On average, each agent injects into the concurrent storage a fire status collection of 10 smart homes per 1 second. Since there are three agents working simultaneously, there should be 30 records per second stored on the concurrent storage. As Algorithm 3 dictates, workers periodically examine the concurrent storage to observe whether there are new entries to submit to the blockchain platform. In this controlled experiment, the worker is configured to periodically inspect the concurrent storage every second. In light of the above, the worker should be able to submit approximately 1000 transactions in total to the blockchain platform with a send rate of 1 transaction per second, such that each transaction should report a collection of fire status for 30 smart homes. Shall the inspection of the blockchain ledger match this expectation, we can generalise the outcomes to more transactions per second with different configurations.

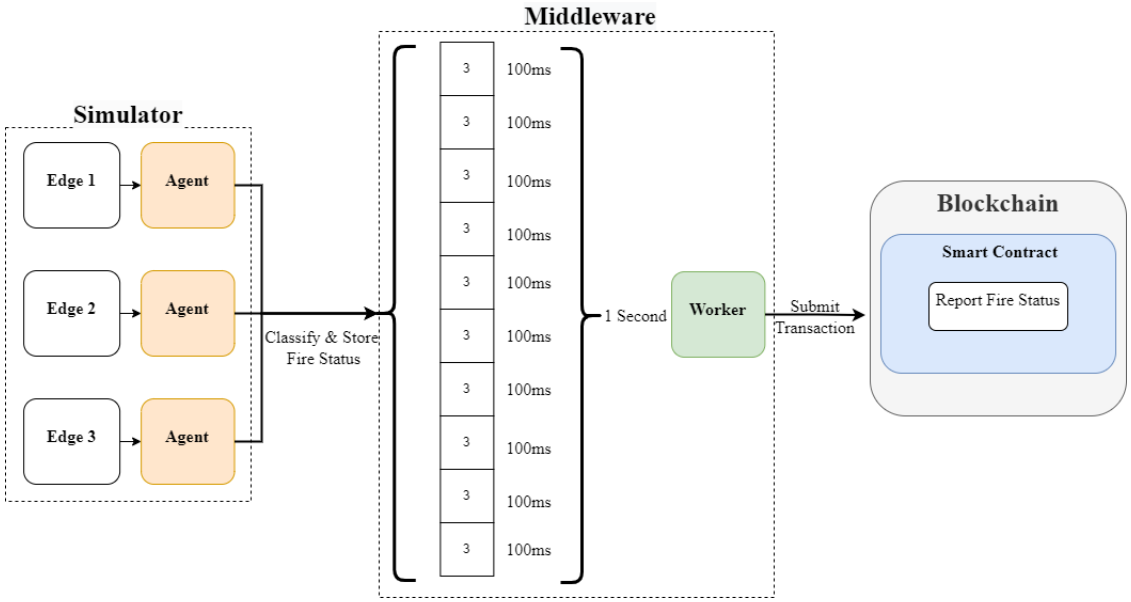


Fig. 6.9 Benchmarking Workload Flow from Simulated Edge Centres to The Smart Contract

6.4.2.4 Validation and Analysis

The experiment utilised Grafana to visualize the results of the key performance measurements defined by Equations 6.2, 6.3, 6.4, and 6.5. The experiment produced a total of 1,030 transactions by running the simulator, surpassing the expected value of 1,000 transactions as indicated in Table 6.2. The expected number of transactions is calculated as follows:

1. **Fire Status Generation**: Simulate 30,000 smart homes, each generating a fire status (either "fire" or "no fire") every 100 milliseconds. These homes are divided into three geographic regions, each connected to an edge data center.

2. **Status Aggregation**: One agent was deployed per edge data center, and each agent collected fire status from 10,000 homes. Agents gathered approximately 10 fire statuses per second.

3. **Transaction Bundling**: Each worker is responsible for examining the concurrent storage and bundling the collected fire statuses into a single transaction. Workers are configured to examine the concurrent storage every 1 second. Under normal conditions,

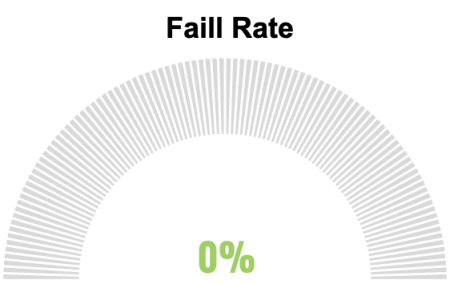
we expect the worker to find approximately 30 fire statuses (10 fire statuses per agent $\times$ 3 agents) every second and submit them as a single transaction to the blockchain.

4. **Expected Transactions Calculation**: The expected transactions is calculated as follows:

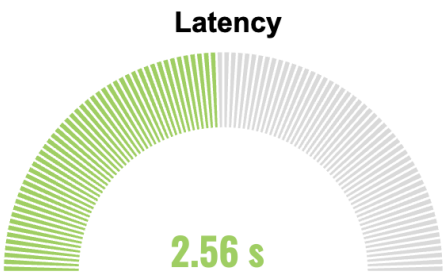
$$\begin{aligned}\text{• Total of expected transactions} &= \frac{\text{Generated Fire Statuses}}{\text{Fire Statuses per transaction}} \\ &= \frac{30{,}000}{30} \\ &= 1000 \text{ transactions}\end{aligned}$$

Figure 6.10 shows a visualised output by SimBlockLink about the essential performance measurement of the blockchain platform. Nevertheless, an investigation of the blockchain ledger revealed that each transaction reported approximately ± 30 metrics, resulting in a slight but acceptable deviation from the anticipated total number of transactions needed for reporting all 30,000 metrics. While the experiment anticipated precisely 1000 transactions, it actually required 1,030 transactions. We observed that a few transactions did not utilise their maximum capacity to contain metrics in their payloads. Thus, an additional 30 transactions are required to satisfy the total number of metrics (30,000). To help explain the deviation from the anticipated number of transactions, a future study will consider a further investigation to determine what led to this phenomenon by examining internal blockchain components and their interconnection, underlying infrastructure and resource allocation, connectivity between the middleware and the blockchain environment, or possibly other causes beyond immediate control.

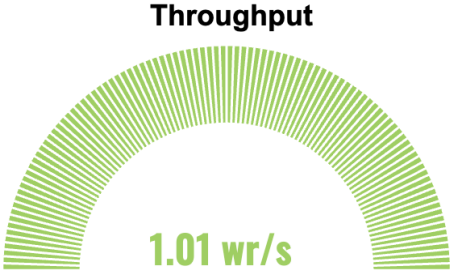
(a) Success Rate



(b) Faill Rate



(c) Latency



(d) Throughput

Fig. 6.10 Key Blockchain performance indicators produced by the middleware

Regardless of the number of total transactions, we verify all generated metrics are submitted to the blockchain platform. For that, we applied Equation 6.6 to the state storage, which checks whether all reported B and C sets are identical to the simulator-generated metrics $|M|$, which should be 30,000 in total. This calculation confirmed that all metrics were indeed reported. The observed marginal deviation in the total number of transactions likely represents a minor artefact, which we plan to investigate in a future study.

$$\left(\sum_{0}^{n-1} B + \sum_{0}^{n-1} C \right) \stackrel{?}{=} |M| \tag{6.6}$$

We hypothesize two potential causes for the observed discrepancy in total transactions. First, as discussed in Sukhwani et al. (2018) [162], Hyperledger Fabric's inherent transaction flow might influence the observed behaviour. The worker component might be forced to wait until the resolution of its current transaction before initiating the next round. Secondly, the concurrent storage locking mechanism may cause the worker to wait for the lock to be released by the agent, which leads to unexpected delays.

6.5 Conclusion

This chapter presented a middleware architecture that facilitates the usage of IoT simulators to evaluate the performance of real blockchain platforms. This allows for the assessment and comparison of blockchain performance in simulated IoT environments. The middleware incorporates agents for interfacing with the IoT simulator, workers for transacting with the blockchain, and a concurrent storage mechanism for data synchronisation. A case study about a hypothetical IoT-based firefighting scenario was implemented to demonstrate the utility of the middleware in integrating Hyperledger Fabric (a real blockchain platform) with IoTsim-Osmosis (an IoT simulator) and in evaluating the performance of the real blockchain. Subsequent efforts will concentrate on supporting additional blockchain platforms, like Ethereum and other non-Java simulators. So far, simulation proves viability for evaluating the performance of a Blockchain-based IoT ecosystem, whether in a full or partial simulation environment. Nonetheless, it can be difficult to calibrate all blockchain configuration parameters to achieve the best possible performance, whether in simulation or real settings. Therefore, the next chapter employs machine learning techniques to address this challenge.

Chapter 7

A Model-Based Machine Learning Approach for Assessing the Performance of Blockchain Applications

Summary

Blockchain technology offers significant potential for various applications, but evaluating the performance of blockchain solutions remains challenging due to the complex and decentralized nature of the underlying infrastructure. While previous research has often utilized simulation-based approaches, machine learning (ML) techniques are under-explored in this context. In previous chapters, this thesis has presented a simulation framework and a middleware architecture for evaluating blockchain performance in the IoT context. The experience so far with simulation proved challenging in fine-tuning and calibrating the plethora of blockchain configuration parameters to achieve the optimal performance possible. Accordingly, this chapter aims to address this gap by proposing two novel ML-based models. The First ML model is trained based on k nearest neighbour (*k*NN) and Support Vector Machine (SVM) algorithms to help estimate the blockchain performance based on predefined configuration parameters. The second model is based on an improved salp swarm optimization (ISO) algorithm that leverages rough set theory to identify optimal blockchain configurations for

achieving desired performance levels, even under uncertainty. To evaluate our models, we conduct experiments using a dataset generated and obtained from the proposed simulation framework in Chapter 5. The data sets contain several scenarios where different configuration parameter values are associated with certain performance levels. The results demonstrate that the *k*NN model outperforms SVM by 5% in classification accuracy. Furthermore, the ISO algorithm exhibits a 4% reduction in the accuracy deviation compared to standard salp optimization.

The remainder of this chapter is structured as follows: Section 7.1 provides a brief introduction to this chapter and highlights the challenges it seeks to address. Section 7.2 explains the Research Question (RQ), the contributions of this chapter, and its relevance to the published paper. Section 7.3 proposes our two models for predicting the overall blockchain performance and estimating the optimal configuration parameters, respectively. Section 7.4 conducts several experiments to validate the proposed models. Finally, Section 7.5 concludes the chapter.

7.1 Introduction

Fine-tuning optimal blockchain configurations can pose challenges as applications' requirements vary significantly. Factors influencing these requirements include the nature of stored data, the frequency and concurrency of transactions, the number of validating nodes, and infrastructure specifications such as CPU, memory, network bandwidth, and Input/Output speed. The constraints that these applications must account for can further complicate the development process, thus affecting the overall performance of the blockchain-based application in terms of throughput, latency, and the rate of successful/failed transactions [117].

This study draws inspiration from a hypothetical scenario where a healthcare organization considers integrating blockchain technology into its operations. Certain performance metrics, such as transaction volume, average transaction time, and transaction success rate, amongst

others, can be utilized to gauge the feasibility and potential success of the proposed project. These metrics can be leveraged for one of the specific purposes outlined subsequently:

1. Predicting how the blockchain-based application will perform under certain conditions and preset configuration parameters, given the limitation of the available resources.

2. Vice versa, given a target performance level, the task is to estimate the right configuration parameters. This is to answer questions like what configurations should be in place to enable an IoT-enabled hospital to achieve a blockchain throughput of at least 1000 transactions per second.

To approach the selection and implementation of blockchain-based applications systematically, it is necessary to conduct a comprehensive evaluation of the application's requirements. Numerous simulation frameworks for this purpose have been proposed [13]. However, it is challenging to provide a comprehensive and accurate representation of a specific blockchain application's performance due to blockchain systems' complexity. A blockchain system's interdependence and wide range of parameters make achieving an accurate performance evaluation a significant challenge.

Machine learning (ML), a subfield of artificial intelligence, uses historical data to develop algorithms and statistical models that aim for optimal performance [163]. This chapter mainly focuses on the supervised learning approach, specifically on classification and optimization algorithms. In the realm of machine learning, classification is concerned with understanding and sorting data into predetermined groups or "sub-populations". Classification algorithms use labelled training data to determine whether an object belongs to a predefined category. They identify recurring patterns and common features, thereby enabling "pattern recognition". The efficiency of these algorithms is evaluated based on their ability to classify objects correctly. In this study, we focus on two well-known classification algorithms: the k nearest neighbour (kNN) algorithm [164] [165], and the support vector machine (SVM) algorithm [166].

Swarm optimization, a machine learning technique, is gaining attention due to its ability to efficiently find near-optimal solutions for complex problems, even with limited resources,

such as processing power or time, and incomplete or imprecise knowledge about the problem [167] [168]. To overcome the limitations of traditional optimization methods, several ML algorithms have been proposed, such as Harris Hawk Optimization (HHO) [169], Grey Wolf Optimization (GWO) [170], Artificial Bee Colony (ABC) [171], Ant Colony Optimization (ACO) [172], Particle Swarm Optimization (PSO) [173], and Salp Optimization (SO) [174]. These algorithms provide robust optimization capabilities.

Another interesting machine learning algorithm is the Rough Set Theory (RST) [175], which provides a formal approach to approximate conventional or crisp sets using lower and upper approximations. If the lower and upper approximations are identical, RST provides a crisp set. If the approximations are different, variations of RST may result in rough sets.

7.2 Research Questions, Contributions, and Relevance to Published Work

Section 7.2.1 provides a detailed explanation of the Research Question (RQ) and highlights the contributions associated with this chapter. Section 7.2.2 clarifies the relevance of this chapter to the published paper, as outlined by the publications listed in Section 1.4.

7.2.1 Research Question and Contribution

Research Question 4 (RQ4): Blockchain infrastructure is heterogeneous and complex. Numerous factors influencing the overall blockchain performance must be considered before an organisation investigates blockchain viability before production. While simulation can help investigate the performance of blockchain-based IoT applications, there would be multiple trial and error processes until the right configuration parameters are determined for optimal blockchain performance. Therefore, the question is how machine learning techniques would assist in predicting blockchain performance metrics and identifying optimal configuration parameters.

To answer this question, this chapter utilises the simulation framework proposed and implemented by Chapter 5 to implement various scenarios to generate datasets of random configuration parameters and associated performance metrics. The generated datasets are used for training Machine Learning models for two purposes considered in the context of evaluating the performance of blockchain-based IoT as follows:

1. Employing the k-nearest neighbours (kNN) method and Support Vector Machine (SVM) algorithms for predicting the overall blockchain performance based on various configuration parameters such as the number of nodes, miners, and transactions.

2. Providing an enhanced version of the Salp Optimization (ISO) algorithm that incorporates rough set theory to handle performance uncertainty and identify optimal configuration parameters for achieving a target blockchain performance.

7.2.2 Relevance of the Chapter to the Published Paper

This chapter corresponds to the relevant publication [18] by introducing two machine learning models to predict and optimize blockchain performance. The chapter and the publication cover key topics, including using kNN, SVM, and ISO algorithms and presenting similar information and data. However, this chapter further distinguishes itself by comparing the kNN and SVM classifiers regarding precision, accuracy, and recall.

7.3 Proposed Models

7.3.1 Preliminaries

Assume a number of configuration parameters (input) and performance metrics (output) of a blockchain-based solution as follows:

1. **Configuration *Parameters*, P**: The set of l parameters $P = \{p_1, p_2, \ldots, p_l\}$ represents the *input configuration* of the blockchain network such as the quantity of participating

nodes, transactions frequency, payload size, selected consensus mechanism, and so forth.

2. **Performance *Metrics*, M**: The set of n metrics $M = \{m_1, m_2, \ldots, m_n\}$ represent the *conditional outputs* with respect to the given parameters P such as network throughput and latency.

Hypothetically, there is a strong correlation between configuration parameters and produced metrics. Therefore, we investigate the following:

1. Employing kNN algorithm as a regression tool for predicting the overall performance in terms of each metric $m \in M$ of a blockchain-based application based on a given set of configuration parameters P.

2. Employing the Salp Swarm Optimization (SO) algorithm to determine the optimal configuration parameters P based on a target level for each performance metric $m \in M$

7.3.2 kNN Regression Algorithms for Performance Predication

To identify commonalities, kNN algorithms compare a given set of parameters (P_0) with unknown values of performance metrics to their k neighbours. The commonalities are usually computed using a distance measure. The idea is that the set of parameters P_0 will be closer to the set of parameters P_i of similar characteristics. kNN trains vectors with class labels in a multidimensional feature space. Each training data row has its parameters setup and decision values. Here, measurements are decision-conditional features. Only the algorithm's training samples' feature vectors and class labels are stored. Averaging the metric values of k nearby objects should yield the anticipated value. Given a dataset D with l features (configuration parameters) and m performance metric, we refer to the parameter value j of object i as $v_{i,j}$. For example, $v_{2,5} = 6$ means that parameter number 2 of object number 2 has value 6. Moreover, the decision value m_k of object i is referred to as $v_{i,d}$.

kNN depends heavily on a distance measure. Euclidean distance is a typical distance metric for continuous values (parameters). The Euclidean distance $l(u_0, u_i)$ between two

different objects, u_0 and u_i is given by

$$l(u_0, u_i) = \sqrt{\left(\mathbf{V}'_{u_0} - \mathbf{V}'_{u_i}\right)^T \left(\mathbf{V}'_{u_0} - \mathbf{V}'_{u_i}\right)}, \tag{7.1}$$

where

$$\mathbf{V}'_{u_k} = < v_{u_i,a'_1}, v_{u_i,a_{r_2}}, \ldots, v_{u_i,a'_m} > ,$$

The proposed algorithm employing the previous steps is shown in Algorithm 6.

Algorithm 6: kNN regression algorithm for blockchain metircs prediction

Input : D //Training data
u_0 //Unknown query object
k //Number of nearest neighbour

Output : M //Set of predicted metrics

$L := \emptyset$
for *each* *object* $u_i \in D$ **do**
    Compute the Euclidean distance between u_0 and u_i as per Eq. (7.1) and add it to $L[i]$
end
Sort L in ascending order.
Find the first k objects in $L[i]$ with the least distance value
for *each* *metric* $m_i \in M$ **do**
    Compute the value of m_i for the unknown object u_0 by averaging the corresponding metric values of the k neighbouring objects.
    $m_j := \frac{1}{k}\sum_{j=1}^{k} v_{j,d}$, where $v_{j,d}$ is the decision value of object u_j in the first k objects in L
end

7.3.3 Improved Salp Optimization (ISO) Algorithm

Each salp has a number i, where $i = 1, 2, \ldots, \mathscr{P}$ and an identifier indicating whether it is a leader or not. The numerals are permanent, but the identifiers may vary between iterations. In the initial iteration, the P salps occupy arbitrary “positions" in the chain, i.e., they simply adhere to the chain. A “position" is a location vector that describes a set of parameter values. The algorithm finalizes the iteration by identifying the m salps with the greatest performance as leaders, moving them to the front of the chain, and allowing them to share their position data (location vectors) with the non-leaders. In other words, the algorithm accomplishes the parameter values identification assignment in two successive steps: the exploration step and the exploitation step, each of which is described in greater detail below.

ISO Exploration Step

Salp $i \geq 1$ has in iteration $k \geq 1$ a location vector $\mathbf{S}_{i_k} = [s_1, s_2, \ldots, s_n]$, the values of y_j has different ranges. Therefore, we feed the algorithm by the separate range of each s_j. In subsequent cycles, this s_j is constantly updated. The parameter vector defines the parameters' values. The s_j specifically reflects the value of parameter j.For example, $\mathbf{S}_{3_2} = [4, 2, 1, 0.064]$ means that salp 3 in iteration 2 is representing parameter configuration for four parameters with values 4,2,1,0.064, respectively.

At iteration 1, each salp is started by an *randomly* generated parameter vector of the l original parameters, acquiring an initial parameter vector. Remember that the random values are chosen with the s_j bounds in mind. This parameter vector is changed on each cycle. The dependence function evaluates the fitness of the parameter and vectors and also acts as an ambiguity-relaxing tool.

Specifically, the dependency value γ is computed by the end of each iteration $k \geq 1$, for the parameter vectors of all P salps in the chain is calculated. The salps with the greatest γ values are then designated as leaders. Those in charge are said to be closer to the ideal parameter setting than the others.

ISO Exploitation Steps

Let the set of p leaders in iteration $k \geq 1$ be $\mathbb{P}_k$. A non-leader salp i gets its updated parameter vector $\mathbf{S}_{i_{k+1}}$ in two stages in the subsequent iteration $k+1$ of the algorithm. Each leader salp modifies its parameter vector in the first phase in the manner described below.

$$\mathbf{S}_{i_k} = \mathbf{S}_{i_{k-1}} + r^2(ub - lb) + rlb \tag{7.2}$$

Similarly, each non-leader salp i, $i \notin \mathbb{P}_k$, will calculate mean difference of p vectors (one for each $j \in \mathbb{P}_k$) as follows.

$$\mathbf{D}_{i,j} = \frac{1}{m}\left[\sum r_1 \mathbf{S}_{i_k} - \mathbf{S}_{j_k}, \quad j \in \mathbb{P}_k\right], \tag{7.3}$$

where r is given by

$$r = 2e^{-\left(\frac{4m}{L}\right)}.$$

To this end, given a set of p salps $\mathbf{S}_{i_j}$ at iteration j, some of the salps parameter vectors may have ambiguous values. The ambiguous values are those that do not lead to promising solutions. Therefore, it will be a hard task to update such vectors. This problem may worsen by getting trapped in the local minima. Consequently, we introduce a goodness function γ depending on the well-known mathematical theory: rough set theory (RST), to solve such an issue. RST is known for its promising abilities in dealing with ambiguity through computing the approximation space, which is a set of approximations referred to as lower and upper. The former represents the set of objects with no ambiguity, while the latter represents the set of ambiguous objects. Assume we are optimizing the metric value m_k; i.e. m_k is the input value. First, we compute the fitness value $f_{\mathbf{S}_{i_j}}$ of each salp $\mathbf{S}_{i_j}$ by computing the regression value using kNN algorithm as per Section 7.3.2. Second, let τ be a user-defined value that serves as a threshold. The set of salps $\mathbb{S}_j^+$ at iteration j having $f_{\mathbf{S}_{i_j}} > \tau$ are considered good; otherwise; the set $\mathbb{S}_j^-$ are considered ambiguous.

Definition 1 (Goodness function, γ): Given finite set of n salps $\mathbf{S}_{i_j}$, we compute the equivalence relation E of each salp as follows.

$$E_{\mathbf{S}_{i_j}} = \{\mathbf{S}_{k_j} | l(\mathbf{S}_{i_j}, \mathbf{S}_{k_j}) < \frac{1}{2}(|\mathbf{D}_{i,j} - \mathbf{D}_{k,j}|)\} \tag{7.4}$$

The lower, upper and boundary approximations are given as follows.

$$\underline{Apr}(\mathbb{S}_j^+) = \{E_{\mathbf{S}_{i_j}} | : E_{\mathbf{S}_{i_j}} \subseteq \mathbb{S}_j^+\} \tag{7.5}$$

$$\overline{Apr}(\mathbb{S}_j^+) = \{E_{\mathbf{S}_{i_j}} | : E_{\mathbf{S}_{i_j}} | \bigcap \mathbb{S}_j^+ \neq \emptyset\} \tag{7.6}$$

$$BND(\mathbb{S}_j^+) = \overline{Apr}(\mathbb{S}_j^+) - \underline{Apr}(\mathbb{S}_j^+) \tag{7.7}$$

Finally, the goodness of the upper approximation is given by

$$\gamma = \frac{|\underline{Apr}(\mathbb{S}_j^+)|}{n}. \tag{7.8}$$

■

Having said this, to improve ISO algorithm convergence and to avoid getting trapped in the local optima, the set of salps in the boundary region $BND(\mathbb{S}_j^+)$ is completely deleted are regenerated concerning the salps having high goodness values.

The processes mentioned above are used by the ISO pseudocode displayed in Algorithm 7. Only the first iteration, where $k = 1$, uses the algorithm's initialization process. Then it executes a loop where a different method is used for every $k > 1$ iteration. The computational cost of ISO may be calculated by using Algorithm 6 and noting that P is the number of salps and R is the number of iterations. The *exploration* step involves ISO spanning P parameter vectors. With N salps in hands, computing the fitness function for each vector costs $O(N)$. The exploration phase thus costs $O(MN)$. Second, the ISO method changes each parameter vector R twice at most during the *exploitation* stage. As a result, this step's cost is $O(RMN2)$. The entire computing cost of the ISO method is $O(RMN2)$ since the exploitation step is the most important one.

Algorithm 7: Improved Salp Optimization Algorithm (ISO)

Input : m //Metric value to be optimized
P //Number of salps
R //Number of iterations ($R \geq 2$)

Output : P //Set of l parameters

//Initialization step:

$k = 1$

$\Gamma = [\,]$ //An empty list to save the dependency of all salps

for $i = 1$ ***to*** P **do**

  Construct parameter vector $\mathbf{S}_{i_k} = [s_1, s_2, ..., s_n]$, where y_j is set randomly according to the parameter constraints.

  Calculate the fitness $f_{\mathbf{S}_{i_k}}$ using kNN model as per Algorithm 6

end

Compute the goodness of the P salps as per Definition 1

Delete the salps appearing in $BND(\mathbb{S}_j^+)$ computed using Eq. (7.7)

Construct set $\mathbb{L}_k = \{i_1, i_2, , ..., i_m\}$, where the i_j are the indices of the highest p values in Γ. //Tag top performing salps as leaders.

Regenerate the deleted salps with respect to the leaders

//Iteration steps:

do

  $k = k + 1$

  $\Gamma = [\,]$

  for $i = 1$ ***to*** P **do**

    if $i \in \mathbb{L}_{k-1}$ **then**

      //If salp i is tagged as leader

      Calculate $f_{\mathbf{S}_{i_k}}$, using kNN and append it to Γ.

    else

      //If salp i is not tagged as leader

      Calculate parameter vector $\mathbf{S}_{i_k}$ from $\mathbf{S}_{i_{k-1}}$, as per (7.3). //Update parameter vector.

      Compute the goodness of the P salps as per Definition 1

      Delete the salps appearing in $BND(\mathbb{S}_j^+)$ computed using Eq. (7.7)

    end

  end

  Construct set $\mathbb{L}_k = \{i_1, i_2, ..., i_m\}$, where the i_j are the indices of the highest p values in Γ. //Tag top performing salps as leaders.

  Assign the highest $f_{\mathbf{S}_{i_k}}$ to $Hfit$.

  Set P to the best salp

while $(k < R)$;

7.4 Experimental Work

The proposed models were implemented using Python and executed on a system equipped with CentOS 7, a 2.4 GHz Intel Core i7 processor, and 16 GB of RAM. The code is available

on GitHub[1]. We conducted several experiments using the collected data with two main objectives in mind. First, we aimed to test the kNN model's ability to predict blockchain performance accurately. Second, we aimed to test the ISO algorithm's ability to identify the best parameter configurations required to achieve a user-defined value for a specific metric, such as throughput.

7.4.1 Data Collection

There is currently no readily accessible public dataset tailored to the tasks outlined in this work. Furthermore, considering our objective to validate the proposed concepts, we elected to utilize a dataset derived from a simulation environment. This approach allows us to control the parameters involved and generate a diverse array of performance data.

In the simulation scenario used for our study, we employed the Raft consensus algorithm. As per the operational constraints imposed by Raft, we were compelled to operate with a single miner node. This constraint is inherent to the design of the Raft consensus protocol and is not a limitation of our study.

It is important to note that the training of machine learning models necessitates a substantial volume of historical data. Therefore, we generated the requisite data using a blockchain simulator. The specifics of the parameters (P_i) that we manipulated to alter the blockchain's characteristics are described in Table 7.1. Our data generation approach provided us with the flexibility to adjust these parameters and collect a comprehensive dataset for our machine learning models.

[1]https://github.com/AlbshriAdel/BlockchainPerformanceML

Table 7.1 The description of the nine used parameters with their abbreviation, lower L and upper U bound of each.

Parameter	Abb.	Desc.	Format	L	U
Number of nodes	P_1	The number of nodes participating in the blockchain network	Integer	3	15
Number of miners	P_2	The number of miners participating in the blockchain network	Integer	1	1
Consensus algorithm	P_3	Consensus Algorithm “Raft"	String	-	-
#transactions/ second	P_4	The total number of transactions generated	Integer	9	1650
Max block size	P_5	The maximum amount of block size	Decimal	1	1
Max transaction size	P_6	The maximum transaction data size	Decimal	0.064	0.064
Min transaction size	P_7	The minimum transaction data size	Decimal	0.001	0.001
Block interval	P_8	Block processing time	Decimal	0.05	0.0099
Simulation time	P_9	The time taken for executing	Decimal	1	1

The simulated blockchain model is executed several times using different configuration values for the parameters described in Table 7.1. During these runs, we thoroughly examined the data. Having identified the set of conditional features, we now turn our attention to the decision features, which include the set of performance metrics M. These features are computed based on conditional features and can be used to evaluate the performance of the blockchain. Table 7.2 provides details about the metrics we have used.

Note that computing the decision characteristics presented in Table 7.2 in simulation mode requires computing the prior features, as shown in Table 7.1. To count these features, we need to have access to the details of each block, which can be a time-consuming process. Therefore, we can define the issue as follows: we will use the ML method (kNN) and conditional features to forecast metric choice feature values directly. In the following sections, we will train the ML model to predict decision feature values using conditional features.

Table 7.2 The description of the thirteen used metrics with their abbreviation.

Metric	Abb.	Desc.	Format
Total number of blocks	M_1	The number of blocks generated	Integer
Total number of blocks including transactions	M_2	The number of blocks that contains transactions	Integer
Total number of transactions	M_3	The number of transactions generated	Integer
Total number of pending transactions	M_4	The number of transactions not processed	Integer
Total number of blocks without transactions	M_5	The number of empty blocks	Integer
Average block size	M_6	The average blocks size	Decimal
Average number of transactions per block	M_7	Average transactions per block	Decimal
Average transaction inclusion time	M_8	Average transaction time	Decimal
Average transaction size	M_9	The average size of the transactions	Decimal
Average block propagation	M_{10}	Average block time	Decimal
Average transaction latency	M_{11}	The average time between transaction submission and confirmation	Decimal
Transactions execution	M_{12}	Average number of transactions per block	Decimal
Transaction Throughput	M_{13}	The rate of transactions throughput	Decimal

It is crucial to examine the statistical properties of the collected data to ensure the reliability of the subsequent ML results. Upon reviewing Table 7.1, we observe that there are six numerical features (P_5, P_6, $\ldots P_9$) present in the dataset.

Prompted by this observation, we sought to gain insights into the dispersion and distribution of these numerical features. We specifically calculated the mean and standard deviation for these features to evaluate the skewness, or asymmetry, of the distribution in the dataset. Additionally, we conducted an examination for any missing values that might affect the analysis.

The results of this comprehensive statistical analysis are detailed in Table 7.3. These preliminary findings will aid us in understanding the inherent characteristics of our dataset,

thereby assisting in the formulation of more accurate machine learning models and predictions.

In the context of a substantial dataset, it proves beneficial to ascertain its central tendency, often represented by a single value such as the mean, median, or mode. This central tendency provides an approximate average value, facilitating an understanding of the dataset's general characteristics. Referring to Table 7.3, it is evident that all numerical features exhibit a notably small standard deviation. This indicates that the data points for each feature are closely distributed around the mean, a sign of well-organized and reliable data. Additionally, to complement the numerical evaluation, we conducted a visual examination of the dataset. For instance, we inspected the distribution of one of the numerical features, namely the block interval feature (P_8). This analysis revealed a normal, or Gaussian, distribution, further validating the quality of the dataset. Furthermore, a meticulous inspection of the collected data did not identify any missing values. This absence of missing data implies that our dataset is complete and further contributes to the robustness of our subsequent ML analysis.

Table 7.3 Statistical analysis (mean, standard deviation, std, minimum and maximum values) for numerical features (5 parameters and 8 metrics).

Feature	Mean	Std	Min	Max
$\boldsymbol{P_5}$	1	0	1	1
$\boldsymbol{P_6}$	6.40E-02	1.40E-17	6.40E-02	6.40E-02
$\boldsymbol{P_7}$	1.00E-03	2.20E-19	1.00E-03	1.00E-03
$\boldsymbol{P_8}$	0.075	0.014	0.05	0.1
$\boldsymbol{P_9}$	1	0.0145	0.05	0.09
$\boldsymbol{M}_6$	0.585	0.287	0.0302	0.971
$\boldsymbol{M}_7$	18.044	8.887	1	30.846
$\boldsymbol{M}_8$	0.484	0.0275	0.421	0.585
$\boldsymbol{M}_9$	0.0325	0.0012	0.027	0.0373
$\boldsymbol{M}_{10}$	0.0381	0.009	0.0209	0.089
$\boldsymbol{M}_{11}$	0.0525	0.0521	0.016	0.266
$\boldsymbol{M}_{12}$	0.9303	0.0332	0.8047	0.999
$\boldsymbol{M}_{13}$	508.306	268.197	11.184	1248.655

The correlation matrix between parameter-conditional features is helpful for understanding the data and examining feature relationships. This information can be used to verify projected performance. Table 7.4 presents the results of this analysis. We have found that the total number of blocks without transactions (M_5) is unrelated to the other features and can therefore be overlooked. However, the average block size (M_6) and the average number of transactions per block (M_7) have a strong positive association, demonstrating the power of the decision features.

Table 7.4 Correlation matrix for the 22 features (9 parameters and 13 metrics) used in the experiments.

	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}	M_{11}	M_{12}	M_{13}
M_1	1	1	0.579	-0.099	0	0.25	0.25	-0.098	0.074	-0.93	-0.16	0.13	0.59
M_2	1	1	0.57	-0.09	0	0.25	0.25	-0.09	0.074	-0.93	-0.16	0.13	0.59
M_3	0.57	0.57	1	0.30	0	0.90	0.9	0.38	-0.10	-0.50	0.42	0.58	0.99
M_4	-0.09	-0.09	0.30	1	0	0.42	0.43	0.41	-0.07	0.12	0.89	0.36	0.28
M_5	0	0	0	0	0	0	0	0	0	0	0	0	0
M_6	0.25	0.25	0.90	0.42	0	1	0.99	0.49	-0.08	-0.24	0.61	0.66	0.9
M_7	0.25	0.25	0.90	0.43	0	0.99	1	0.50	-0.12	-0.24	0.61	0.65	0.9
M_8	-0.09	-0.09	0.38	0.41	0	0.49	0.50	1	-0.012	0.13	0.65	0.57	0.35
M_9	0.07	0.07	-0.10	-0.07	0	-0.08	-0.12	-0.01	1	-0.12	-0.08	0.03	-0.1
M_{10}	-0.93	-0.93	-0.50	0.12	0	-0.24	-0.24	0.13	-0.12	1	0.18	-0.10	-0.52
M_{11}	-0.16	-0.16	0.426	0.89	0	0.61	0.61	0.65	-0.08	0.18	1	0.55	0.39
M_{12}	0.13	0.13	0.58	0.36	0	0.66	0.65	0.57	0.03	-0.10	0.55	1	0.53
M_{13}	0.59	0.59	0.99	0.28	0	0.90	0.90	0.35	-0.10	-0.52	0.39	0.53	1

	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9
P_1	1	0	0	-0.03	0	0	0	-0.062	0
P_2	0	0	0	0	0	0	0	0	0
P_3	0	0	0	0	0	0	0	0	0
P_4	-0.027	0	0	1	0	0	0	-0.11	0
P_5	0	0	0	0	0	0	0	0	0
P_6	0	0	0	0	0	0	0	0	0
P_7	0	0	0	0	0	0	0	0	0
P_8	-0.062	0	0	-0.11	0	0	0	1	0
P_9	0	0	0	0	0	0	0	0	0

7.4.2 Prediction Results

To prevent the issue of feature dominance, all numerical features are normalized. A normalized feature value $\widehat{v}_{u_i,a_j}$ is obtained from its raw value v_{u_i,a_j} by the equation 7.9:

$$\widehat{v}_{u_i,a_j} = \frac{v_{u_i,a_j} - \min\limits_k \left(v_{u_k,a_j}\right)}{\max\limits_k \left(v_{u_k,a_j}\right) - \min\limits_k \left(v_{u_k,a_j}\right)}, \tag{7.9}$$

where $\min\limits_k \left(v_{u_k,a_j}\right)$ and $\max\limits_k \left(v_{u_k,a_j}\right)$ are the minimum and maximum values of feature a_j, considering all objects, respectively. This formula guarantees that $\widehat{v}_{u_i,a_j} \in [0,1]$ for all i and all j.

Our first test involves finding the best k value for the kNN algorithm. To do so, we ran the model multiple times while changing the k value and computing the root mean square error (RMSE). We then selected the k value with the best RMSE. The RMSE is calculated as the standard deviation of the residuals, which represent the prediction errors. Residuals indicate how far data points are from the regression line, while RMSE indicates how spread out these residuals are. In other words, it shows how closely the data is clustered around the line of best fit. Root mean square error is often used to evaluate the results of experiments in climatology, forecasting, and regression analysis. RMSE is given by the equation 7.10:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} ||y(i) - \hat{y}(i)||^2}{N}}, \tag{7.10}$$

where N represents the total count of data points, $y(i)$ denotes the i-th measurement in the dataset, and $\hat{y}(i)$ signifies the corresponding predictive estimation for the i-th observation. The result of this experiment is shown in Figure 7.1. It is evident that a choice of $k = 1$ leads to a very high RMSE. When k is set to 5, the RMSE reduces significantly, approximating a value of 67.06. Any further increment in the value of k results in a drastic drop in the RMSE. Consequently, it can be confidently inferred that $k = 5$ is the optimal choice for this particular scenario, yielding the most favourable results.

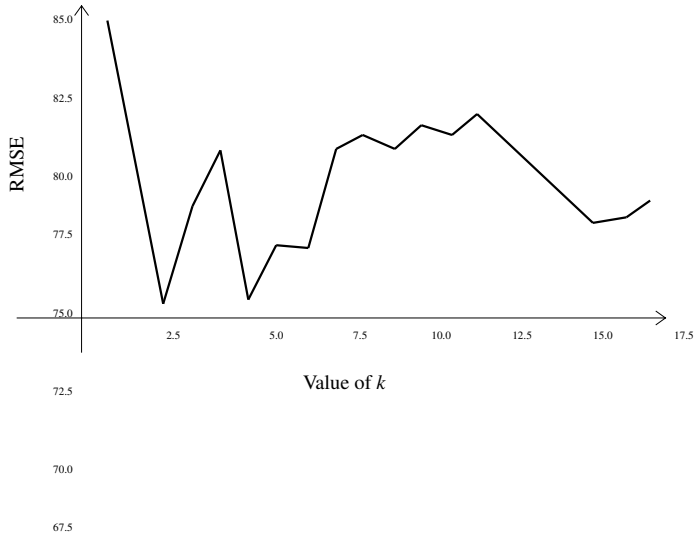


Fig. 7.1 RMSE as a function of the value of k. The figure shows that the optimal value of k is 5, where the lowest RMSE is achieved.

10-fold cross-validation ensures solid results. Nine sub-datasets are for training and one for testing. Each object appears once in a test set and nine times in training sets. Results are averaged after 10 separate tests. To further support our findings and evaluate the models, we considered additional metrics such as Accuracy (Equation 7.11), Precision (Equation 7.12), and Recall (Equation 7.13). The initial analysis suggests that the kNN model outperforms the SVM model, especially in terms of precision and recall, indicating its suitability for accurately identifying the minority class in our dataset. The comprehensive results of this extended analysis are provided in Table 7.5 and further supported by the comparison in Table 7.6.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \tag{7.11}$$

$$\text{Precision} = \frac{TP}{TP+FP} \tag{7.12}$$

$$\text{Recall} = \frac{TP}{TP+FN} \tag{7.13}$$

Table 7.5 Performance Metrics for *k*NN and SVM

Metric	***k*NN**	**SVM**
Accuracy	92%	89%
Precision	95%	90%
Recall	93%	88%

Now, to test the model for an instance level, we feed the model with specific conditional features and predict some of the decision features. The prediction is made concerning both *k*NN and SVM. The results of this experiment are shown in Table 7.6. The conditional feature values are copied from the first ten rows of the data. The interesting point is that the *k*NN prediction values are much closer to the target result than those of SVM. This ensures its correctness concerning blockchain performance prediction.

Table 7.6 The classification accuracy of *k*NN and SVM for three different metrics over 10 different parameter configurations.

Parameters (P)									**Metrics (M)**					
P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	M_{11}		M_{12}		M_{13}	
									***k*NN**	**SVM**	***k*NN**	**SVM**	***k*NN**	**SVM**
13	1	raft	519	1	0.064	0.001	0.083	1	0.033	0.024	**0.912**	0.99	**569.026**	559.99
6	1	raft	682	1	0.064	0.001	0.069	1	0.035	0.045	**0.92**	0.89	**737.72**	744.66
9	1	raft	66	1	0.064	0.001	0.070	1	0.022	0.055	**0.91**	0.84	**72.35**	72.44
9	1	raft	450	1	0.064	0.001	0.058	1	0.020	0.029	**0.93**	0.83	**480.88**	489.81
15	1	raft	893	1	0.064	0.001	0.072	1	0.12	0.19	**0.99**	0.74	**754.931**	759.899
9	1	raft	440	1	0.064	0.001	0.069	1	0.026	0.031	**0.940**	0.830	**467.88**	476.35
6	1	raft	965	1	0.064	0.001	0.065	1	0.1077	0.098	**0.98**	0.98	**982.21**	977.23
7	1	raft	17	1	0.064	0.001	0.095	1	0.033	0.055	**0.91**	0.97	**18.68**	19.67

We used non-parametric approaches such as the Friedman methodology since the distribution of these data was uncertain. For both its single and repeated testing options, this statistical investigation used MATLAB's Friedman's single and repeated test procedures. A Friedman function would construct a structure for the entire circumstance. This structure and a suitable post-hoc procedure will be used as input for a multi-comparison function. First, the Friedman test was performed with the null hypothesis of *k*NN and SVM. Friedman

values for *k*NN and SVM are 0.0015 and 0.0027, respectively. Because a significance level of alpha=0.05 was assumed, the null hypothesis is rejected in each case based on the p-values. As a result, we can now confirm that the accuracy of the two algorithms is different. In other words, SVM is statistically different from *k*NN.

We proceeded to the second step, which was dubbed “post hoc," once we realized that the accuracy of the two algorithms was not the same. During this phase, we conducted four Friedman tests for each of the four different situations. Because repeated testing leads to an increased risk of making a Type I error—that is, incorrectly concluding that a null hypothesis should be rejected when, in fact, it should be accepted—we were forced to use one of the post hoc methods at our disposal in order to find a solution to this issue. We used two: Fisher’s least significant difference approach and Tukey’s honest significant difference criteria. Table 7.7 provides a view of the adjusted p-values, reflecting significant statistical testing following the Friedman tests. Specifically, the p-values under the columns for Fisher’s and Tukey’s corrections indicate whether the differences observed between the *k*NN and SVM algorithms under various parameter settings are statistically significant. Notably, p-values less than 0.05 demonstrate a statistically significant difference, affirming that *k*NN typically outperforms SVM under these settings. This detailed validation is crucial for substantiating the choice of one algorithm over the other in practical applications.

Table 7.7 Post hoc *p*-values resulting from Friedman tests of *K*NN for four parameters.

Correction		P_5	P_6	P_7	P_8	P_9
Features	Fisher	0.0001	0.0007	0.0575	0.1372	0.0166
	Turkey	0.0028	0.0126	0.4808	0.7532	0.2007

7.4.3 ISO Validation Results

This section looks at how well the ISO algorithm works to find parameters in a blockchain. To make the investigation meaningful, we compare ISO’s performance to that of five other competitor algorithms: PSO, HHO, GWO, ABC, ACO, and SO. The five algorithms we compare are very recent.

Each algorithm searches for the ideal configuration based on a metric input value. To verify, the kNN regressor receives the parameter vector. The method is more reliable the closer the original value is to the anticipated one. We used 20 salps for 50 iterations in this experiment with three leaders. Table 7.8 shows that ISO (last row) won this experiment. In the last row, $M13 = 1100$, ISO produced parameter vector has 83% accuracy, while the best competitor, the classic salp technique, has 81 accuracy.

Table 7.8 The fitness value achieved by ISO and six competitors. Clearly, ISO (the last row) comes out as a clear winner.

Algo.	**Metrics (M)**												
	M_1 = 33	M_2 = 29	M_3 = 1102	M_4 = 50	M_5 = 0.5	M_6 = 0.7	M_7 = 25	M_8 = 0.2	M_9 = 0.02	M_10 = 0.07	M_11 = 0.25	M_12 = 0.8	M_13 = 1100
PSO	21	25	559	22	0.4	0.5	21	0.2	0.018	0.06	0.15	0.4	752
HHO	25	21	687	29	0.5	0.6	21	0.18	0.011	0.05	0.19	0.5	897
GWO	26	20	714	25	0.4	0.7	22	0.15	0.019	0.06	0.18	0.5	774
ABC	26	27	752	26	0.3	0.5	24	0.2	0.025	0.06	0.19	0.8	687
ACO	26	28	777	29	0.4	0.5	20	0.19	0.23	0.05	0.19	0.7	744
SO	29	26	798	29	0.4	0.7	22	0.2	0.03	0.05	0.22	0.8	899
ISO	**31**	**29**	**912**	**43**	**0.5**	**0.7**	**24**	**0.2**	**0.021**	**0.07**	**0.23**	**0.8**	**915**

The development of the fitness value across iterations serves as another comparison test. Figure 7.2 shows this trend. The ISO curve is generally superior to all other curves. This suggests that ISO consistently outperforms other standards, regardless of the statistics. As a result, the evolution paints a clear picture of the algorithm's conduct from the beginning to the finish of the assignment.

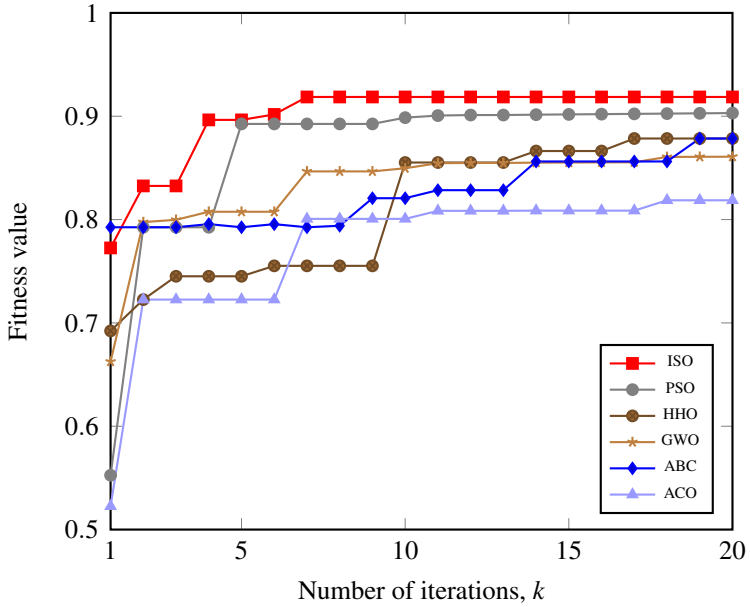


Fig. 7.2 Fitness value compared to the number of iterations: A higher fitness value indicates quicker convergence of the algorithm.

One final experiment is to look at the predicted parameter vector by ISO and its five competitors. In this experiment, we are attempting to identify the parameter vector that will result in $M_{13} = 1100$. The results of this experiment are shown in Table 7.9. Noting that the last column (achieved value) corresponds to the value of M_{13} in the resulting parameter vector, we can notice that ISO has the best-achieved value, which is much closer to 1100 than any other competitor.

Table 7.9 The predicted parameters form the 6 algorithm for $m_{13} = 1100$. Clearly, ISO reached a parameter vector, achieving the closest value.

Algo.	**Parameters (P)**									**Achieved value**
	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	
PSO	6	1	1	654.670	1	0.368	0.557	0.0796	1	681.390
HHO	6	1	1	1087.262	1	0.264	0.509	0.0752	1	806.143
GWO	6	1	1	733.183	1	0.428	0.441	0.0787	1	777.512
ABC	8	1	1	796.841	1	0.393	0.501	0.068	1	739.642
SO	6	1	1	750.958	1	0.347	0.401	0.059	1	766.865
ISO	6	1	1	919.76	1	0.297	0.567	0.064	1	**823.940**

7.5 Conclusion

The advent of blockchain technology has initiated a paradigm shift across numerous sectors due to its inherent potential and advanced capabilities. Nevertheless, the intricate and decentralized characteristics of blockchain's underlying infrastructure introduce challenges in assessing the performance of blockchain-based applications. Thus, the necessity for a dependable modelling methodology becomes paramount to aid the creation and performance evaluation of such applications. Historically, research has predominantly focused on simulation-based solutions to evaluate blockchain application performance, while the exploration of machine learning (ML) model-based techniques remains comparatively scant in this context. This study sought to bridge this gap by proposing two innovative ML-based techniques.

The first approach integrated a k nearest neighbour (kNN) and a support vector machine (SVM) to predict blockchain performance by leveraging predefined configuration parameters. The second method utilized salp swarm optimization (SO), an ML model, to identify the most advantageous blockchain configurations for achieving the desired performance benchmarks. To further refine the efficacy of SO, we incorporated rough set theory, thus formulating an Improved Swarm Optimization (ISO) model. The ISO model displayed superior capabilities in generating accurate recommendations for optimal parameter configurations amidst uncertainties. Upon comparative statistical evaluation, our proposed models exhibited a competitive advantage. Specifically, the kNN model outperformed the SVM by a margin of 5%, while the ISO model demonstrated a 4% reduction in accuracy deviation relative to the standard SO model. These results are considered encouraging because they not only validate the effectiveness of our models in accurately predicting blockchain performance under diverse and uncertain conditions but also demonstrate tangible improvements over traditional methods. The ability of the ISO model to refine parameter selection processes and reduce inaccuracies in predictions significantly contributes to the field of blockchain performance evaluation, highlighting the practical implications of our research in real-world applications.

While the proposed ML models in this chapter were trained using data generated from the simulation framework, their results can also be applicable to real blockchain platforms (namely, the Quorum Blockchain platform). The applicability is justified because the comparative analysis in section 5.4.2 demonstrates a high degree of similarity between the simulated scenario and the real-world implementation of the scenario. However, even in the case of non-applicability, the models proposed in this chapter can be retained to use newly generated datasets from new versions of the Quorum blockchain platform or other blockchain platforms. Future studies will consider employing and experimenting with generative artificial intelligence (AI) to compose the best possible Blockchain configuration parameters.

Chapter 8

Conclusion and Future Work

Summary

This chapter begins with highlighting the main contributions of this research 8.1. It then provides a discussion of the research questions and how the contributions of this thesis address each research question 8.2. Additionally, it discusses the limitations of the current work and explores potential directions for future research 8.3. Finally, Section 8.4 explores the advantages and limitations of using simulations for blockchain-based IoT and concludes whether they are valuable for researchers and developers.

8.1 Thesis Contribution

This thesis has made several significant contributions to the field of Blockchain and IoT as follows:

1. The first contribution is a systematic mapping study of current blockchain simulation research, including an analysis of each existing simulator's features, capabilities, and limitations, as discussed in Chapter 3.

2. The second contribution proposes a novel framework for evaluating blockchain-based IoT performance, developed through requirements gathering, analysis, design, implementation and validation, as discussed in Chapters 4 and 5.

3. The third contribution introduces a middleware architecture that bridges IoT simulators and real blockchain platforms to evaluate the performance of Blockchain-based IoT applications, as discussed in Chapter 6.

4. The fourth contribution designs and implements Machine learning models to predict blockchain performance using k-nearest neighbours (kNN) and optimize configurations with an improved Salp Optimization (ISO) algorithm incorporating rough set theory, as discussed in Chapter 7.

8.2 Thesis Summary

This thesis has made several significant contributions to the field of blockchain and IoT. It includes a systematic mapping study of blockchain simulators, developing and implementing a blockchain-based IoT simulation framework, creating a novel middleware architecture, and applying machine learning models to predict blockchain performance. These contributions are critically tied to the research questions outlined at the beginning of this thesis 1.2, and the following sections will reflect on how each contribution addresses these questions.

RQ1: What techniques and configurations are used in current blockchain simulators?

Research Question 1 (RQ1) aimed to achieve an understanding of blockchain simulation by systematically categorizing existing blockchain simulators based on their modeling approaches, the blockchain platforms they simulate, and the specific features they support. Additionally, it aimed to evaluate the effectiveness of these simulators in performance assessment and to identify critical research gaps that need to be addressed to advance the field further.

To address the research question, we adopted a systematic mapping methodology involving structured steps: defining the research questions, conducting a comprehensive literature search, screening relevant studies, keywording abstracts, and performing detailed data extraction and mapping. Initially, we searched for all papers using the keyword in different scientific databases, yielding 259 papers, of which 20 were deemed relevant after thorough screening. These papers provided the foundation for this study and offered an overview of various blockchain simulators, comparing them based on their available open-source implementations, supported configuration parameters (inputs), and generated metrics (outputs). Additionally, the source code quality of the simulators was assessed using tools such as SonarQube and CLOC.

The findings revealed that no current simulator fully encompasses the wide range of features and capabilities inherent to existing blockchain technologies. While there are several promising efforts in blockchain simulation, offering interesting and useful features, these simulators still lack the ability to integrate with emerging technologies such as cloud computing or IoT. This limitation restricts their applicability within broader technological ecosystems.

The first contribution answers **Research Question 1 (RQ1)** by systematically comparing and evaluating blockchain simulators, thereby providing in-depth insights into their capabilities and limitations. Through this process, it identified significant disparities in the suitability of these simulators for various use cases and, in particular, highlighted gaps in their applicability to emerging technologies such as cloud computing and the Internet of Things (IoT). Consequently, these findings underscore the need to develop more versatile simulators that can integrate with these technologies, thereby expanding blockchain's applicability within broader technological ecosystems. Moreover, the research lays a solid foundation for understanding and comparing blockchain simulators while also suggesting avenues for further exploration, such as empirical studies under varying conditions, to gain deeper insights into their strengths and limitations. Ultimately, this contribution advances the understanding of blockchain simulation and aligns the research outcomes with the initial goals set out by RQ1.

RQ2: Given the lack of existing simulation frameworks for evaluating the performance of Blockchain-based IoT ecosystems, what is required to bridge the gap?

Research Question 2 (RQ2) aimed to identify key requirements by systematically gathering and analyzing the essential factors necessary for integrating blockchain with IoT platforms. Furthermore, it aimed to design a simulation framework that accurately mimics the behavior of a real blockchain-based IoT system, validate the framework through expert feedback, refine it to meet design goals, and address identified weaknesses. Finally, it sought to implement the framework, test it using a real-world use case, and benchmark its performance against real blockchain platforms to ensure its effectiveness.

To address the research question, we adopted a collaborative methodology involving structured steps. We began with a mixed-method approach to gather and analyze simulation tool requirements based on feedback from subject-matter experts. This two-part study included a questionnaire to validate the benefits of a simulator for evaluating IoT integrated with blockchain and interviews to identify key challenges, necessary features, and how blockchain can enhance IoT performance. The insights informed the design of a conceptual framework comprising four components: Configurator, Generator, Simulation Core, and Reporter. The design was reviewed through a focus group, and the framework was implemented using Java and the IoTsim-Osmosis simulation framework.

The initial phase of gathering requirements identified significant challenges and opportunities in integrating blockchain technology with IoT systems, emphasizing the need for advanced simulation tools. In response, we developed a conceptual model featuring extensive, tailored functionalities to meet these needs. This conceptual model was rigorously assessed and received high praise for its comprehensive capabilities, establishing it as a solid foundation for simulating blockchain-based IoT applications. Following its development, the simulator underwent thorough testing, with results indicating that transaction latency and overall performance closely mirrored those of real-world blockchain systems, confirming the

model's practical effectiveness. Furthermore, the quality of implementation results revealed high satisfaction, with an 80-87% approval rate across various aspects of the simulator.

The second contribution answers **Research Question 2 (RQ2)** by systematically developing and validating a novel simulation framework tailored for blockchain-based IoT ecosystems. First, the research identified key requirements through a mixed-method approach, incorporating expert feedback to address significant challenges and opportunities in integrating blockchain with IoT systems. Consequently, this led to the design of a conceptual framework with comprehensive functionalities to accurately mimic real blockchain-based IoT behaviors. Furthermore, the framework underwent rigorous evaluation, including thorough testing and validation against real-world blockchain platforms, thereby demonstrating its effectiveness in replicating transaction latency and overall performance in alignment with real-world systems. However, while the study provides a robust simulation, there remains potential for future enhancements, as discussed in Section . Ultimately, this contribution advances knowledge in the evaluation of blockchain-based IoT performance approaches and aligns the research outcomes with the objectives set out by RQ2

RQ3: How feasible is it to utilize IoT simulators as workload generators to benchmark the performance of real blockchain platforms?

Research Question 3 (RQ3) aimed to design middleware that bridges the gap between IoT simulators and real blockchain platforms, ensuring seamless interoperability and data synchronization for performance benchmarking purposes. It sought to implement the middleware and validate it by benchmarking the performance of an IoT simulator with a real blockchain platform.

To address the research question, we designed and implemented the middleware using the Java programming language, integrating it with the IoTSim-Osmosis simulator on the IoT simulation side while selecting Hyperledger Fabric (HLF) as the real-world blockchain platform. The architecture included agents for gathering and evaluating metrics, workers for submitting transactions, concurrent storage for data synchronization, and a transaction fail-

safe mechanism for reliability. It also incorporated smart contracts for IoT asset management, identity management for secure transactions, and a benchmarking tool for performance metrics.

The findings revealed that the concurrent storage mechanism effectively managed data synchronization between the IoT simulator and the blockchain platform, ensuring correctness and reliability under varying workloads. This directly addressed the critical need for reliable data handling in mixed environments. Additionally, the results confirmed the middleware's capability to evaluate key blockchain performance metrics, including throughput, latency, and transaction success/failure rates. This demonstrates the middleware's ability to handle the generation and processing of IoT workloads for accurate performance benchmarking.

The third contribution answers **Research Question 3 (RQ3)** by systematically designing and implementing a middleware solution that bridges the gap between IoT simulators and real blockchain platforms. First, the research aimed to ensure seamless interoperability and data synchronization by integrating the middleware with the IoTSim-Osmosis simulator and Hyperledger Fabric (HLF). Consequently, the middleware architecture was developed with key components such as metrics gathering agents, transaction submission workers, concurrent data synchronization storage, and a transaction fail-safe mechanism to enhance reliability. Furthermore, the middleware was evaluated through testing and validation, demonstrating its capability to manage data synchronization and evaluate key blockchain performance metrics like throughput, latency, and transaction success/failure rates. However, while the middleware effectively bridges the gap between IoT simulators and real blockchain platforms for performance evaluation, there remains potential for further development, as discussed in Section . Ultimately, this contribution provides a valuable tool for overcoming challenges between two distinctive natures of IoT simulation and real blockchain for performance optimization, aligning the research outcomes with the objectives set out by RQ3.

RQ4: How do machine learning techniques assist in predicting blockchain performance metrics and identifying optimal configuration parameters?

Research Question 4 (RQ4) aimed to develop and deploy machine learning (ML) models to predict key blockchain performance metrics, such as throughput and latency. Additionally, it sought to develop ML models that could recommend optimal configuration parameters to achieve desired blockchain performance outcomes, ensuring efficiency and effectiveness in blockchain operations.

To address this research question, and given the lack of a readily accessible public dataset specifically tailored to the tasks outlined in this work, we used a dataset derived from a simulation environment to validate the proposed concepts. In the first model, we employed k-nearest neighbor (kNN) and Support Vector Machine (SVM) algorithms to predict blockchain performance based on predefined configuration parameters. In the second model, we enhanced the Salp Swarm Optimization (SSO) algorithm by incorporating rough set theory to identify optimal blockchain configurations that align with desired performance levels.

The findings revealed that the kNN model outperformed the SVM model by 5% in predicting blockchain performance. This indicated that kNN was a more effective algorithm for this specific application, providing a reliable performance prediction method. Additionally, the improved SSO model exhibited a 4% decrease in accuracy deviation compared to the standard SSO model, demonstrating its enhanced stability and reliability in identifying optimal blockchain configurations.

The fourth contribution answers **Research Question 4 (RQ4)** by developing ML models for predicting and optimizing blockchain performance. Initially, the research focused on developing robust ML models to predict blockchain performance metrics, such as throughput and latency, using algorithms like kNN and SVM. Subsequently, the kNN model demonstrated superior performance, providing a reliable method for predicting blockchain outcomes and enabling more informed decision-making regarding configurations. Additionally, the second model, which enhanced the SSO algorithm with rough set theory, was evaluated

for its ability to identify optimal blockchain configurations. The enhanced SSO model exhibited improved stability and reliability, offering precise recommendations for configuring blockchain systems to achieve optimal performance. Despite these advancements, there is potential for further refinement and development, as discussed in Section . Ultimately, this contribution enriches the field by developing ML models for optimizing blockchain-based IoT performance, aligning with the objectives outlined in RQ4.

8.3 Limitations and Future Work

This section discusses the limitations of our current studies and proposes several potential avenues for future research to enhance and expand upon the contributions made in this thesis.

Blockchain-based IoT Simulation Framework (*Chapter 5*)

Although the simulator demonstrates its potential, there are limitations and potential future work to enhance the research, as follows:

- The current implementation of the simulator focuses on integration with IoTsim-Osmosis, which is a representative IoT simulator; there are other IoT simulators available, such as iFogSim [85] and CloudSim [86]. Future work will extend the simulator's compatibility with these and other popular IoT simulators.
- The current simulator implementation focuses on modelling the Raft and Proof-of-Work (PoW) consensus algorithms. Future work will extend the simulator's capabilities to model and evaluate the performance of blockchain-IoT ecosystems using other consensus algorithms, such as Proof-of-Stake (PoS) and more configuration parameters, such as extended transaction payload and gossip protocols.
- There is room for improvement in terms of user-friendliness by developing a user-friendly interface such as a graphical user interface (GUI).

IoT Simulation for Evaluating Blockchain Performance: A Middleware Architecture (*Chapter 6*)

While the proposed middleware has shown viability for evaluating the performance of a Blockchain-based IoT ecosystem, there are limitations and potential future work to enhance the work, as follows:

- The current implementation of the middleware focuses on the integration of IoTsim-Osmosis with Hyperledger Fabric. Future work will explore the compatibility of the middleware with other IoT simulators (e.g. Cloudsim [86], iFogSim [85] etc.) and other blockchain platforms (e.g. Quorum [22]).

- There is another significant aspect to consider, which is the user experience and usability of the middleware. The current implementation provided a functional integration between IoTsim-Osmosis and Hyperledger Fabric. However, there is room for improvement in terms of user-friendliness and ease of use. In future work, it would be interesting to develop an interactive user interface that would allow developers and researchers to easily configure and monitor the middleware's operations. Additionally, providing detailed documentation and code examples that facilitate the adoption and utilisation of middleware by the IoT and blockchain community.

A Model-Based Machine Learning Approach for Assessing the Performance of Blockchain Applications (*Chapter 7*)

Despite the advancements presented in this chapter through the development of machine learning models for evaluating blockchain-based IoT performance, there are limitations and potential future work to enhance the research, as follows:

- The current study relied on a dataset generated using a blockchain-based IoT simulator, which allowed controlled parameter manipulation and the generation of various performance data. Although beneficial for initial model training and testing, the simulated data may not fully capture the complexities and unpredictable dynamics of real-world

blockchain-based IoT applications. Future work focuses on collecting data from real-world blockchain platforms to assess the applicability and performance of the proposed models in practical scenarios.

- The study focused mainly on employing the k-nearest neighbour (*k*NN) and SVM algorithms for performance prediction and the salp swarm optimisation algorithm for parameter optimization. While the *k*NN model demonstrated perfect performance compared to the Support Vector Machine (SVM) in the tests, its performance may vary with different blockchain configurations or under different operating conditions not covered by simulation. Similarly, despite its accuracy in parameter optimisation, the improved Salp Optimisation (ISO) algorithm still faces challenges related to scalability and computational efficiency, particularly with larger datasets. Future work could investigate the applicability and performance of other state-of-the-art machine learning algorithms, such as deep learning models, in the context of blockchain-based IoT performance prediction and optimisation.

8.4 Why Use Simulation for Blockchain and IoT

This section explores the advantages and limitations of using simulations to evaluate blockchain and Internet of Things (IoT) systems, highlighting their value to researchers and developers.

1. **Advantages**: Simulation offers key advantages in blockchain and IoT for design, testing, and optimization. These advantages include:

   (a) **Creating a Controlled Testing Environment**: One of the most notable benefits of simulation is its ability to create a controlled environment for testing Blockchain and IoT scenarios. This controlled setting allows users to explore various conditions without the risks associated with real-world implementation. By employing simulations, developers can identify and address potential issues early in the development process, thereby enhancing the efficiency and effectiveness of system design.

(b) **Facilitating Detailed Performance Analysis**: Simulations also offer the opportunity for in-depth performance analysis under diverse conditions. This capability is crucial for blockchain and IoT systems. It enables researchers to fine-tune parameters and optimize system configurations in ways that theoretical analysis alone might not reveal. This detailed examination helps researchers gain insights essential for refining and improving their systems.

(c) **Promoting Cost-Effective**: Another significant advantage is the cost-effectiveness of simulations. For Blockchain-based IoT systems, simulations provide a method to experiment with new ideas and technologies without the need for expensive physical prototypes or testbeds. This approach allows developers to explore various design options and configurations, significantly reducing financial barriers to innovation.

(d) **Enabling Extensive Testing Under Extreme Conditions**: Simulations also facilitate extensive testing under extreme or rare conditions, which is essential for ensuring the robustness and resilience of Blockchain-based IoT systems. By exposing systems to a broad spectrum of potential challenges, researchers can identify and address vulnerabilities, thereby contributing to the development of more reliable and resilient technologies.

2. **Limitations**: While simulations offer numerous advantages in blockchain and IoT domains, their effectiveness hinges on the quality of underlying assumptions and data models. Real-world complexities can be challenging to replicate accurately, potentially limiting the applicability of simulation results. To mitigate these limitations, simulations should be complemented with real-world testing. This combined approach grounds insights in practical reality, enhancing the reliability and relevance of findings for blockchain and IoT systems.

8.4.1 Evaluating Simulation's Contribution to Blockchain and IoT Advancement

Considering the numerous advantages and acknowledging the limitations, it is evident that using simulations for Blockchain and IoT is worth investing time and resources. The ability to conduct risk-free experiments, perform detailed analyses, innovate cost-effectively, and test under extreme conditions provides immense value to researchers and developers in this field. While simulations should not be viewed as a complete substitute for real-world testing, they serve as an invaluable tool in developing and optimising Blockchain and IoT systems.

The insights gained from simulations can significantly accelerate development cycles, reduce costs, and enhance the overall quality of the final systems. By allowing for rapid iteration and experimentation, simulations enable developers to explore a wider range of possibilities and innovative solutions that might be impractical or too risky to attempt in real-world scenarios. This capability is particularly crucial in the fast-evolving landscapes of Blockchain and IoT, where staying ahead of technological curves can be a significant competitive advantage. Moreover, simulations foster a more comprehensive understanding of complex system behaviours, interactions, and potential vulnerabilities. This deeper insight is instrumental in developing more robust, efficient, and secure Blockchain and IoT solutions. As these technologies continue to integrate into various sectors of the economy and society, the role of simulations in ensuring their reliability, scalability, and security becomes increasingly vital.

Using simulations in the context of Blockchain and IoT is beneficial and often essential for advancing the field and creating robust, efficient, and innovative solutions. As these technologies continue to evolve and expand their applications, the importance of simulation as a tool for development, testing, and optimization is likely to grow, further cementing its role as a cornerstone in the advancement of Blockchain and IoT technologies.

References

[1] N. Adhikari and M. Ramkumar, "Iot and blockchain integration: Applications, opportunities, and challenges," *Network*, vol. 3, no. 1, pp. 115–141, 2023. [Online]. Available: https://doi.org/10.3390/network3010006

[2] P. Asghari, A. M. Rahmani, and H. H. S. Javadi, "Internet of things applications: A systematic review," *Comput. Networks*, vol. 148, pp. 241–261, 2019. [Online]. Available: https://doi.org/10.1016/j.comnet.2018.12.008

[3] P. M. Chanal and M. S. Kakkasageri, "Security and privacy in iot: A survey," *Wirel. Pers. Commun.*, vol. 115, no. 2, pp. 1667–1693, 2020. [Online]. Available: https://doi.org/10.1007/s11277-020-07649-9

[4] R. Roman, J. Zhou, and J. López, "On the features and challenges of security and privacy in distributed internet of things," *Comput. Networks*, vol. 57, no. 10, pp. 2266–2279, 2013. [Online]. Available: https://doi.org/10.1016/j.comnet.2012.12.018

[5] N. K. Tran, M. A. Babar, and J. Boan, "Integrating blockchain and internet of things systems: A systematic review on objectives and designs," *J. Netw. Comput. Appl.*, vol. 173, p. 102844, 2021. [Online]. Available: https://doi.org/10.1016/j.jnca.2020.102844

[6] A. Alzubaidi, E. Solaiman, P. Patel, and K. Mitra, "Blockchain-based SLA management in the context of iot," *IT Prof.*, vol. 21, no. 4, pp. 33–40, 2019. [Online]. Available: https://doi.org/10.1109/MITP.2019.2909216

[7] K. Wüst and A. Gervais, "Do you need a blockchain?" in *Crypto Valley Conference on Blockchain Technology, CVCBT 2018, Zug, Switzerland, June 20-22, 2018*. IEEE, 2018, pp. 45–54. [Online]. Available: https://doi.org/10.1109/CVCBT.2018.00011

[8] D. Pavithran, K. Shaalan, J. N. Al-Karaki, and A. Gawanmeh, "Towards building a blockchain framework for iot," *Clust. Comput.*, vol. 23, no. 3, pp. 2089–2103, 2020. [Online]. Available: https://doi.org/10.1007/s10586-020-03059-5

[9] M. Kaur and S. Gupta, "Optimization of a consensus protocol in blockchain-iot convergence," in *2023 International Conference on Emerging Smart Computing and Informatics (ESCI)*, 2023, pp. 1–5. [Online]. Available: https://doi.org/10.1109/ESCI56872.2023.10100031

[10] X. Wang, X. Zha, W. Ni, R. P. Liu, Y. J. Guo, X. Niu, and K. Zheng, "Survey on blockchain for internet of things," *Comput. Commun.*, vol. 136, pp. 10–29, 2019. [Online]. Available: https://doi.org/10.1016/j.comcom.2019.01.006

[11] J. Zheng, C. Dike, S. Pancari, Y. Wang, G. C. Giakos, W. Elmannai, and B. Wei, "An in-depth review on blockchain simulators for iot environments," *Future Internet*, vol. 14, no. 6, p. 182, 2022. [Online]. Available: https://doi.org/10.3390/fi14060182

[12] J. Banks, J. S. C. II, B. L. Nelson, and D. M. Nicol, *Discrete-Event System Simulation, 5th New Internatinal Edition*. Pearson Education, 2010.

[13] A. Albshri, A. Alzubaidi, B. Awaji, and E. Solaiman, "Blockchain simulators: A systematic mapping study," in *IEEE International Conference on Services Computing, SCC 2022, Barcelona, Spain, July 10-16, 2022*. IEEE, 2022, pp. 284–294. [Online]. Available: https://doi.org/10.1109/SCC55611.2022.00049

[14] A. Albshri, B. Awaji, and E. Solaiman, "Investigating the requirement of building blockchain simulator for iot applications," in *IEEE International Conference on Smart Internet of Things, SmartIoT 2022, Suzhou, China, August 19-21, 2022*. IEEE, 2022, pp. 232–240. [Online]. Available: https://doi.org/10.1109/SmartIoT55134.2022.00044

[15] A. Albshri, A. Alzubaidi, M. Alharby, B. Awaji, K. Mitra, and E. Solaiman, "A conceptual architecture for simulating blockchain-based iot ecosystems," *J. Cloud Comput.*, vol. 12, no. 1, p. 103, 2023. [Online]. Available: https://doi.org/10.1186/s13677-023-00481-z

[16] K. Alwasel, D. N. Jha, F. Habeeb, U. Demirbaga, O. F. Rana, T. Baker, S. Dustdar, M. Villari, P. James, E. Solaiman, and R. Ranjan, "Iotsim-osmosis: A framework for modeling and simulating iot applications over an edge-cloud continuum," *J. Syst. Archit.*, vol. 116, p. 101956, 2021. [Online]. Available: https://doi.org/10.1016/j.sysarc.2020.101956

[17] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. D. Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolic, S. W. Cocco, and J. Yellick, "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *Proceedings of the Thirteenth EuroSys Conference, EuroSys 2018, Porto, Portugal, April 23-26, 2018*, R. Oliveira, P. Felber, and Y. C. Hu, Eds. ACM, 2018, pp. 30:1–30:15. [Online]. Available: https://doi.org/10.1145/3190508.3190538

[18] A. Albshri, A. Alzubaidi, and E. Solaiman, "A model-based machine learning approach for assessing the performance of blockchain applications," in *IEEE International Conference on Smart Internet of Things, SmartIoT 2023, Xining, China, August 25-27, 2023*. IEEE, 2023, pp. 46–55. [Online]. Available: https://doi.org/10.1109/SmartIoT58732.2023.00015

[19] A. Kumar, B. K. Sah, T. Mehrotra, and G. K. Rajput, "A review on double spending problem in blockchain," in *2023 International Conference on Computational Intelligence and Sustainable Engineering Solutions (CISES)*. IEEE, 2023, pp. 881–889. [Online]. Available: https://doi.org/10.1109/CISES58720.2023.10183579

[20] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008. [Online]. Available: www.bitcoin.org

[21] V. Buterin, "A next-generation smart contract and decentralized application platform," 2014. [Online]. Available: http://www.the-blockchain.com/docs/Ethereum_white_paper-a_next_generation_smart_contract_and_decentralized_application_platform-vitalik-buterin.pdf

[22] J. P. Morgan Chase., "A permissioned implementation of Ethereum supporting data privacy," Jan. 2024, original-date: 2016-11-14T05:42:57Z. [Online]. Available: https://github.com/Consensys/quorum

[23] 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*, pp. 1–12, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8643084/

[24] I. Makhdoom, M. Abolhasan, H. Abbas, and W. Ni, "Blockchain's adoption in iot: The challenges, and a way forward," *J. Netw. Comput. Appl.*, vol. 125, pp. 251–279, 2019. [Online]. Available: https://doi.org/10.1016/j.jnca.2018.10.019

[25] 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 2018, Changshu, Suzhou, China, June 26-30, 2018*. IEEE, 2018, pp. 280–285. [Online]. Available: https://doi.org/10.1109/IVS.2018.8500557

[26] A. I. Sanka and R. C. C. Cheung, "A systematic review of blockchain scalability: Issues, solutions, analysis and future research," *J. Netw. Comput. Appl.*, vol. 195, p. 103232, 2021. [Online]. Available: https://doi.org/10.1016/j.jnca.2021.103232

[27] C. Komalavalli, D. Saxena, and C. Laroiya, *Overview of Blockchain Technology Concepts*. Elsevier, 2020, p. 349–371. [Online]. Available: http://dx.doi.org/10.1016/b978-0-12-819816-2.00014-9

[28] D. Romano and G. Schmid, "Beyond bitcoin: A critical look at blockchain-based systems," *Cryptogr.*, vol. 1, no. 2, p. 15, 2017. [Online]. Available: https://doi.org/10.3390/cryptography1020015

[29] N. Szabo, "Smart Contracts," 1994. [Online]. Available: https://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html

[30] L. Atzori, A. Iera, and G. Morabito, "Understanding the internet of things: definition, potentials, and societal role of a fast evolving paradigm," *Ad Hoc Networks*, vol. 56, pp. 122–140, 2017. [Online]. Available: https://doi.org/10.1016/j.adhoc.2016.12.004

[31] H. Allioui and Y. Mourdi, "Exploring the full potentials of iot for better financial growth and stability: A comprehensive survey," *Sensors*, vol. 23, no. 19, p. 8015, 2023. [Online]. Available: https://doi.org/10.3390/s23198015

[32] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (iot): A vision, architectural elements, and future directions," *Future Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, 2013. [Online]. Available: https://doi.org/10.1016/j.future.2013.01.010

[33] W. Yan, Z. Wang, H. Wang, W. Wang, J. Li, and X. Gui, "Survey on recent smart gateways for smart home: Systems, technologies, and challenges," *Trans. Emerg. Telecommun. Technol.*, vol. 33, no. 6, 2022. [Online]. Available: https://doi.org/10.1002/ett.4067

[34] J. J. Peralta Abadía, C. Walther, A. Osman, and K. Smarsly, "A systematic survey of internet of things frameworks for smart city applications," *Sustainable Cities and Society*, vol. 83, p. 103949, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2210670722002700

[35] A. Raghuvanshi, U. K. Singh, and C. Joshi, "A review of various security and privacy innovations for iot applications in healthcare," *Advanced Healthcare Systems: Empowering Physicians with IoT-Enabled Technologies*, pp. 43–58, 2022. [Online]. Available: https://doi.org/10.1002/9781119769293.ch4

[36] Z. Alavikia and M. Shabro, "A comprehensive layered approach for implementing internet of things-enabled smart grid: A survey," *Digit. Commun. Networks*, vol. 8, no. 3, pp. 388–410, 2022. [Online]. Available: https://doi.org/10.1016/j.dcan.2022.01.002

[37] A. I. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Commun. Surv. Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015. [Online]. Available: https://doi.org/10.1109/COMST.2015.2444095

[38] M. R. Bouakouk, A. Abdelli, and L. Mokdad, "Survey on the cloud-iot paradigms: Taxonomy and architectures," in *IEEE Symposium on Computers and Communications, ISCC 2020, Rennes, France, July 7-10, 2020*. IEEE, 2020, pp. 1–6. [Online]. Available: https://doi.org/10.1109/ISCC50000.2020.9219638

[39] L. D. Xu, W. He, and S. Li, "Internet of things in industries: A survey," *IEEE Trans. Ind. Informatics*, vol. 10, no. 4, pp. 2233–2243, 2014. [Online]. Available: https://doi.org/10.1109/TII.2014.2300753

[40] R. A. Radouan Ait Mouha, "Internet of things (iot)," *Journal of Data Analysis and Information Processing*, vol. 09, no. 02, p. 77–101, 2021. [Online]. Available: http://dx.doi.org/10.4236/jdaip.2021.92006

[41] T. Ramathulasi and M. Rajasekhara Babu, *Comprehensive Survey of IoT Communication Technologies*. Springer Singapore, 2020, p. 303–311. [Online]. Available: http://dx.doi.org/10.1007/978-981-15-0135-7_29

[42] I. D. Zyrianoff, A. Heideker, D. Silva, J. H. Kleinschmidt, J. Soininen, T. S. Cinotti, and C. Kamienski, "Architecting and deploying iot smart applications: A performance-oriented approach," *Sensors*, vol. 20, no. 1, p. 84, 2020. [Online]. Available: https://doi.org/10.3390/s20010084

[43] A. Khanna and S. Kaur, "Internet of things (iot), applications and challenges: A comprehensive review," *Wirel. Pers. Commun.*, vol. 114, no. 2, pp. 1687–1762, 2020. [Online]. Available: https://doi.org/10.1007/s11277-020-07446-4

[44] A. A. Sadawi, M. S. Hassan, and M. M. Ndiaye, "A survey on the integration of blockchain with iot to enhance performance and eliminate challenges," *IEEE Access*, vol. 9, pp. 54 478–54 497, 2021. [Online]. Available: https://doi.org/10.1109/ACCESS.2021.3070555

[45] K. Seemakhupt, S. Liu, Y. Senevirathne, M. Shahbaz, and S. M. Khan, "Pmnet: In-network data persistence," in *48th ACM/IEEE Annual International Symposium on Computer Architecture, ISCA 2021, Virtual Event / Valencia, Spain, June 14-18, 2021*. IEEE, 2021, pp. 804–817. [Online]. Available: https://doi.org/10.1109/ISCA52012.2021.00068

[46] P. K. Sharma, M. Chen, and J. H. Park, "A software defined fog node based distributed blockchain cloud architecture for iot," *IEEE Access*, vol. 6, pp. 115–124, 2018. [Online]. Available: https://doi.org/10.1109/ACCESS.2017.2757955

[47] D. Anandayuvaraj and J. C. Davis, "Reflecting on recurring failures in iot development," in *37th IEEE/ACM International Conference on Automated Software Engineering, ASE 2022, Rochester, MI, USA, October 10-14, 2022*. ACM, 2022, pp. 185:1–185:5. [Online]. Available: https://doi.org/10.1145/3551349.3559545

[48] M. A. Uddin, A. Stranieri, I. Gondal, and V. Balasubramanian, "A survey on the adoption of blockchain in iot: challenges and solutions," *Blockchain: Research and Applications*, vol. 2, no. 2, p. 100006, 2021. [Online]. Available: http://dx.doi.org/10.1016/j.bcra.2021.100006

[49] K. C. Chan, X. Zhou, R. Gururajan, X. Zhou, M. Ally, and M. Gardiner, "Integration of blockchains with management information systems," in *2019 International Conference on Mechatronics, Robotics and Systems Engineering (MoRSE)*. IEEE, 2019. [Online]. Available: http://dx.doi.org/10.1109/morse48060.2019.8998694

[50] S. Kar, B. Chakravorty, S. Sinha, and M. P. Gupta, *Analysis of Stakeholders Within IoT Ecosystem*. Springer International Publishing, 2018, p. 251–276. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-78378-9_15

[51] H. R. Hasan, E. Alhadhrami, A. AlDhaheri, K. Salah, and R. Jayaraman, "Smart contract-based approach for efficient shipment management," *Comput. Ind. Eng.*, vol. 136, pp. 149–159, 2019. [Online]. Available: https://doi.org/10.1016/j.cie.2019.07.022

[52] C. Kolias, G. Kambourakis, A. Stavrou, and J. M. Voas, "Ddos in the iot: Mirai and other botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017. [Online]. Available: https://doi.org/10.1109/MC.2017.201

[53] S. Kikuchi and T. Aoki, "Evaluation of operational vulnerability in cloud service management using model checking," in *Seventh IEEE International Symposium on Service-Oriented System Engineering, SOSE 2013, San Francisco, CA, USA, March 25-28, 2013*. IEEE Computer Society, 2013, pp. 37–48. [Online]. Available: https://doi.org/10.1109/SOSE.2013.31

[54] A. Kuznetsov, I. Oleshko, V. Tymchenko, K. Lisitsky, M. Rodinko, and A. Kolhatin, "Performance analysis of cryptographic hash functions suitable for use in blockchain," *International Journal of Computer Network and Information Security*, vol. 13, no. 2, p. 1–15, 2021. [Online]. Available: http://dx.doi.org/10.5815/ijcnis.2021.02.01

[55] L. Alevizos, V. Ta, and M. H. Eiza, "Augmenting zero trust architecture to endpoints using blockchain: A state-of-the-art review," *Secur. Priv.*, vol. 5, no. 1, 2022. [Online]. Available: https://doi.org/10.1002/spy2.191

[56] P. Wang, "Fedchain: An efficient and secure consensus protocol based on proof of useful federated learning for blockchain," *CoRR*, vol. abs/2308.15095, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2308.15095

[57] S. M. Alrubei, E. A. Ball, J. M. Rigelsford, and C. A. Willis, "Latency and performance analyses of real-world wireless iot-blockchain application," *IEEE Sensors Journal*, vol. 20, no. 13, p. 7372–7383, 2020. [Online]. Available: http://dx.doi.org/10.1109/jsen.2020.2979031

[58] M. Alaslani, F. Nawab, and B. Shihada, "Blockchain in iot systems: End-to-end delay evaluation," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8332–8344, 2019. [Online]. Available: https://doi.org/10.1109/JIOT.2019.2917226

[59] V. Aslot, M. J. Domeika, R. Eigenmann, G. Gaertner, W. B. Jones, and B. Parady, "Specomp: A new benchmark suite for measuring parallel computer performance," in *OpenMP Shared Memory Parallel Programming, International Workshop on OpenMP Applications and Tools, WOMPAT 2001, West Lafayette, IN, USA, July 30-31, 2001 Proceedings*, ser. Lecture Notes in Computer Science, R. Eigenmann and M. Voss, Eds., vol. 2104. Springer, 2001, pp. 1–10. [Online]. Available: https://doi.org/10.1007/3-540-44587-0_1

[60] W. F. Tichy, "Should computer scientists experiment more?" *Computer*, vol. 31, no. 5, pp. 32–40, 1998. [Online]. Available: https://doi.org/10.1109/2.675631

[61] Y. Endo, Z. Wang, J. B. Chen, and M. I. Seltzer, "Using latency to evaluate interactive system performance," in *Proceedings of the Second USENIX Symposium on Operating Systems Design and Implementation (OSDI), Seattle, Washington, USA, October 28-31, 1996*, K. Petersen and W. Zwaenepoel, Eds. ACM, 1996, pp. 185–199. [Online]. Available: https://doi.org/10.1145/238721.238775

[62] J. C. Mogul, "Brittle metrics in operating systems research," in *Proceedings of The Seventh Workshop on Hot Topics in Operating Systems, HotOS-VII, Rio Rico, Arizona, USA, March 28-30, 1999*, P. Druschel, Ed. IEEE Computer Society, 1999, pp. 90–95. [Online]. Available: https://doi.org/10.1109/HOTOS.1999.798383

[63] D. Beyer, S. Löwe, and P. Wendler, "Reliable benchmarking: requirements and solutions," *Int. J. Softw. Tools Technol. Transf.*, vol. 21, no. 1, pp. 1–29, 2019. [Online]. Available: https://doi.org/10.1007/s10009-017-0469-y

[64] Hyperledger, "Hyperledger Blockchain Performance Metrics," Hyperledger Performance and Scale Working Group, Tech. Rep., 2018. [Online]. Available: https://www.hyperledger.org/learn/publications/blockchain-performance-metrics

[65] C. Luo, J. Zhan, Z. Jia, L. Wang, G. Lu, L. Zhang, C. Xu, and N. Sun, "Cloudrank-d: benchmarking and ranking cloud computing systems for data processing applications," *Frontiers Comput. Sci.*, vol. 6, no. 4, pp. 347–362, 2012. [Online]. Available: https://doi.org/10.1007/s11704-012-2118-7

[66] A. Das, S. Patterson, and M. P. Wittie, "Edgebench: Benchmarking edge computing platforms," in *2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion, UCC Companion 2018, Zurich, Switzerland, December 17-20, 2018*, A. Sill and J. Spillner, Eds. IEEE, 2018, pp. 175–180. [Online]. Available: https://doi.org/10.1109/UCC-Companion.2018.00053

[67] B. Varghese, N. Wang, D. Bermbach, C. Hong, E. de Lara, W. Shi, and C. Stewart, "A survey on edge performance benchmarking," *ACM Comput. Surv.*, vol. 54, no. 3, pp. 66:1–66:33, 2022. [Online]. Available: https://doi.org/10.1145/3444692

[68] "IEEE 2418.1-2023 - IEEE standard for the framework of Blockchain use in Internet of things (iot)," IEEE Standards Association, 2023, accessed: 2024-03-30. [Online]. Available: https://standards.ieee.org/ieee/2418.1/10460/

[69] M. I. Seltzer, D. Krinsky, K. A. Smith, and X. Zhang, "The case for application-specific benchmarking," in *Proceedings of The Seventh Workshop on Hot Topics in Operating Systems, HotOS-VII, Rio Rico, Arizona, USA, March 28-30, 1999*, P. Druschel, Ed. IEEE Computer Society, 1999, pp. 102–109. [Online]. Available: https://doi.org/10.1109/HOTOS.1999.798385

[70] T. T. A. Dinh, J. Wang, G. Chen, R. Liu, B. C. Ooi, and K. Tan, "BLOCKBENCH: A framework for analyzing private blockchains," in *Proceedings of the 2017 ACM International Conference on Management of Data, SIGMOD Conference 2017, Chicago, IL, USA, May 14-19, 2017*, S. Salihoglu, W. Zhou, R. Chirkova, J. Yang, and D. Suciu, Eds. ACM, 2017, pp. 1085–1100. [Online]. Available: https://doi.org/10.1145/3035918.3064033

[71] C. Hyperledger, "Hyperledger Caliper." [Online]. Available: https://hyperledger.github.io/caliper/

[72] S. Rouhani and R. Deters, "Performance analysis of ethereum transactions in private blockchain," in *2017 8th IEEE international conference on software engineering and service science (ICSESS)*. IEEE, 2017, pp. 70–74. [Online]. Available: https://doi.org/10.1109/ICSESS.2017.8342866

[73] A. Baliga, I. Subhod, P. Kamat, and S. Chatterjee, "Performance evaluation of the quorum blockchain platform," *CoRR*, vol. abs/1809.03421, 2018. [Online]. Available: http://arxiv.org/abs/1809.03421

[74] 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. [Online]. Available: https://doi.org/10.1016/j.bcra.2021.100026

[75] X. Chen, K. Nguyen, and H. Sekiya, "An experimental study on performance of private blockchain in iot applications," *Peer-to-Peer Netw. Appl.*, vol. 14, no. 5, pp. 3075–3091, 2021. [Online]. Available: https://doi.org/10.1007/s12083-021-01148-9

[76] J. Ferreira, M. Antunes, M. Zhygulskyy, and L. Frazão, "Performance of hash functions in blockchain applied to iot devices," in *2019 14th Iberian Conference on Information Systems and Technologies (CISTI)*. IEEE, 2019, pp. 1–7. [Online]. Available: https://doi.org/10.23919/CISTI.2019.8760885

[77] A. Maria, "Introduction to modeling and simulation," in *Proceedings of the 29th conference on Winter simulation, WSC 1997, Atlanta, GA, USA, December 7-10, 1997*, S. Andradóttir, K. J. Healy, D. H. Withers, and B. L. Nelson, Eds. ACM, 1997, pp. 7–13. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/WSC.1997.640371

[78] D. R. Anderson, D. J. Sweeney, T. A. Williams, J. D. Camm, and J. J. Cochran, *An introduction to management science: quantitative approach*. Cengage learning, 2018.

[79] R. R. Hill, "Discrete-event simulation: A first course," *J. Simulation*, vol. 1, no. 2, pp. 147–148, 2007. [Online]. Available: https://doi.org/10.1057/palgrave.jos.4250012

[80] B. R. Haverkort, *Performance of computer communication systems - a model-based approach*. Wiley, 1998.

[81] A. Márkus and A. Kertész, "A survey and taxonomy of simulation environments modelling fog computing," *Simul. Model. Pract. Theory*, vol. 101, p. 102042, 2020. [Online]. Available: https://doi.org/10.1016/j.simpat.2019.102042

[82] S. Sotiriadis, N. Bessis, E. Asimakopoulou, and N. Mustafee, "Towards simulating the internet of things," in *28th International Conference on Advanced Information Networking and Applications Workshops, AINA 2014 Workshops, Victoria, BC, Canada, May 13-16, 2014*, L. Barolli, K. F. Li, T. Enokido, F. Xhafa, and M. Takizawa, Eds. IEEE Computer Society, 2014, pp. 444–448. [Online]. Available: https://doi.org/10.1109/WAINA.2014.74

[83] N. Mohan and J. Kangasharju, "Edge-fog cloud: A distributed cloud for internet of things computations," in *2016 Cloudification of the Internet of Things, CIoT 2016, Paris, France, November 23-25, 2016*. IEEE, 2016, pp. 1–6. [Online]. Available: https://doi.org/10.1109/CIOT.2016.7872914

[84] D. N. Jha, K. Alwasel, A. Alshoshan, X. Huang, R. K. Naha, S. K. Battula, S. Garg, D. Puthal, P. James, A. Y. Zomaya, S. Dustdar, and R. Ranjan, "Iotsim-edge: A simulation framework for modeling the behavior of internet of things and edge computing environments," *Softw. Pract. Exp.*, vol. 50, no. 6, pp. 844–867, 2020. [Online]. Available: https://doi.org/10.1002/spe.2787

[85] H. Gupta, A. V. Dastjerdi, S. K. Ghosh, and R. Buyya, "ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments," *Softw. Pract. Exp.*, vol. 47, no. 9, pp. 1275–1296, 2017. [Online]. Available: https://doi.org/10.1002/spe.2509

[86] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. D. Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Softw. Pract. Exp.*, vol. 41, no. 1, pp. 23–50, 2011. [Online]. Available: https://doi.org/10.1002/spe.995

[87] C. Sonmez, A. Ozgovde, and C. Ersoy, "Edgecloudsim: An environment for performance evaluation of edge computing systems," *Trans. Emerg. Telecommun. Technol.*, vol. 29, no. 11, 2018. [Online]. Available: https://doi.org/10.1002/ett.3493

[88] M. Seufert, B. K. Kwam, F. Wamser, and P. Tran-Gia, "Edgenetworkcloudsim: Placement of service chains in edge clouds using networkcloudsim," in *2017 IEEE Conference on Network Softwarization, NetSoft 2017, Bologna, Italy, July 3-7, 2017*. IEEE, 2017, pp. 1–6. [Online]. Available: https://doi.org/10.1109/NETSOFT.2017.8004247

[89] M. M. Lopes, W. A. Higashino, M. A. Capretz, and L. F. Bittencourt, "Myifogsim: A simulator for virtual machine migration in fog computing," in *Companion Proceedings of The10th International Conference on Utility and Cloud Computing*, ser. UCC '17 Companion. New York, NY, USA: Association for Computing Machinery, 2017, p. 47–52. [Online]. Available: https://doi.org/10.1145/3147234.3148101

[90] T. Qayyum, A. W. Malik, M. A. K. Khattak, O. Khalid, and S. U. Khan, "Fognetsim++: A toolkit for modeling and simulation of distributed fog environment," *IEEE Access*, vol. 6, pp. 63 570–63 583, 2018. [Online]. Available: https://doi.org/10.1109/ACCESS.2018.2877696

[91] N. Mohan and J. Kangasharju, "Edge-fog cloud: A distributed cloud for internet of things computations," in *2016 Cloudification of the Internet of Things, CIoT 2016, Paris, France, November 23-25, 2016*. IEEE, 2016, pp. 1–6. [Online]. Available: https://doi.org/10.1109/CIOT.2016.7872914

[92] S. Sotiriadis, N. Bessis, N. Antonopoulos, and A. Anjum, "Simic: Designing a new inter-cloud simulation platform for integrating large-scale resource management," in *27th IEEE International Conference on Advanced Information Networking and Applications, AINA 2013, Barcelona, Spain, March 25-28, 2013*, L. Barolli, F. Xhafa, M. Takizawa, T. Enokido, and H. Hsu, Eds. IEEE Computer Society, 2013, pp. 90–97. [Online]. Available: https://doi.org/10.1109/AINA.2013.123

[93] R. E. Schantz and D. C. Schmidt, "Middleware for distributed systems," in *Wiley Encyclopedia of Computer Science and Engineering*, B. W. Wah, Ed. John Wiley & Sons, Inc., 2008. [Online]. Available: https://doi.org/10.1002/9780470050118.ecse241

[94] IBM, "Middleware," https://www.ibm.com/topics/middleware, 2024, accessed: insert access date here.

[95] N. Tapas, G. Merlino, and F. Longo, "Blockchain-based iot-cloud authorization and delegation," in *2018 IEEE International Conference on Smart Computing, SMARTCOMP 2018, Taormina, Sicily, Italy, June 18-20, 2018*. IEEE Computer Society, 2018, pp. 411–416. [Online]. Available: https://doi.org/10.1109/SMARTCOMP.2018.00038

[96] M. Samaniego and R. Deters, "Zero-trust hierarchical management in iot," in *2018 IEEE International Congress on Internet of Things, ICIOT 2018, San Francisco, CA, USA, July 2-7, 2018*. IEEE Computer Society, 2018, pp. 88–95. [Online]. Available: https://doi.org/10.1109/ICIOT.2018.00019

[97] S. M. Danish, "A blockchain-based adaptive middleware for large scale internet of things data storage selection," in *Proceedings of the 20th International Middleware Conference Doctoral Symposium, Middleware 2019, Davis, CA, USA, December*

09-13, 2019, F. Nawab and E. Rivière, Eds. ACM, 2019, pp. 17–19. [Online]. Available: https://doi.org/10.1145/3366624.3368159

[98] N. Wang, B. Wang, T. Liu, W. Li, and S. Yang, "A middleware approach to synchronize transaction data to blockchain," in *29th International Conference on Computer Communications and Networks, ICCCN 2020, Honolulu, HI, USA, August 3-6, 2020*. IEEE, 2020, pp. 1–8. [Online]. Available: https://doi.org/10.1109/ICCCN49398.2020.9209722

[99] E. Zhou, H. Sun, B. Pi, J. Sun, K. Yamashita, and Y. Nomura, "Ledgerdata refiner: A powerful ledger data query platform for hyperledger fabric," in *Sixth International Conference on Internet of Things: Systems, Management and Security, IOTSMS 2019, Granada, Spain, October 22-25, 2019*, M. A. Alsmirat and Y. Jararweh, Eds. IEEE, 2019, pp. 433–440. [Online]. Available: https://doi.org/10.1109/IOTSMS48152.2019.8939212

[100] W. Jin, "Research on machine learning and its algorithms and development," *Journal of Physics: Conference Series*, vol. 1544, no. 1, p. 012003, 2020. [Online]. Available: http://dx.doi.org/10.1088/1742-6596/1544/1/012003

[101] H. Zheng, Y. Feng, Y. Gao, and J. Tan, "A robust predicted performance analysis approach for data-driven product development in the industrial internet of things," *Sensors*, vol. 18, no. 9, p. 2871, 2018. [Online]. Available: http://dx.doi.org/10.3390/s18092871

[102] M. Bez, G. Fornari, and T. Vardanega, "The scalability challenge of ethereum: An initial quantitative analysis," in *2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)*. IEEE, 2019, pp. 167–176.

[103] A. Kumari, R. Gupta, S. Tanwar, S. Tyagi, and N. Kumar, "When blockchain meets smart grid: Secure energy trading in demand response management," *IEEE Netw.*, vol. 34, no. 5, pp. 299–305, 2020. [Online]. Available: https://doi.org/10.1109/MNET.001.1900660

[104] I. Abu-elezz, A. Hassan, A. Nazeemudeen, M. S. Househ, and A. A. Abd-alrazaq, "The benefits and threats of blockchain technology in healthcare: A scoping review," *Int. J. Medical Informatics*, vol. 142, p. 104246, 2020. [Online]. Available: https://doi.org/10.1016/j.ijmedinf.2020.104246

[105] M. Umar, C.-W. Su, S. K. A. Rizvi, and X.-F. Shao, "Bitcoin: A safe haven asset and a winner amid political and economic uncertainties in the us?" *Technological Forecasting and Social Change*, vol. 167, p. 120680, 2021. [Online]. Available: http://dx.doi.org/10.1016/j.techfore.2021.120680

[106] X. Chen, K. Nguyen, and H. Sekiya, "An experimental study on performance of private blockchain in iot applications," *Peer-to-Peer Netw. Appl.*, vol. 14, no. 5, pp. 3075–3091, 2021. [Online]. Available: https://doi.org/10.1007/s12083-021-01148-9

[107] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[108] I. A. Omar, H. R. Hasan, R. Jayaraman, K. Salah, and M. Omar, "Implementing decentralized auctions using blockchain smart contracts," *Technological Forecasting and Social Change*, vol. 168, p. 120786, 2021. [Online]. Available: http://dx.doi.org/10.1016/j.techfore.2021.120786

[109] N. C. K. Yiu, "Toward blockchain-enabled supply chain anti-counterfeiting and traceability," *Future Internet*, vol. 13, no. 4, p. 86, 2021. [Online]. Available: https://doi.org/10.3390/fi13040086

[110] S. Xu, B. Liao, C. Yang, S. Guo, B. Hu, J. Zhao, and L. Jin, "Deep reinforcement learning assisted edge-terminal collaborative offloading algorithm of blockchain computing tasks for energy internet," *International Journal of Electrical Power & Energy Systems*, vol. 131, p. 107022, 2021. [Online]. Available: http://dx.doi.org/10.1016/j.ijepes.2021.107022

[111] J. J. Hunhevicz and D. M. Hall, "Do you need a blockchain in construction? use case categories and decision framework for DLT design options," *Adv. Eng. Informatics*, vol. 45, p. 101094, 2020. [Online]. Available: https://doi.org/10.1016/j.aei.2020.101094

[112] S. Smetanin, A. Ometov, M. M. Komarov, P. Masek, and Y. Koucheryavy, "Blockchain evaluation approaches: State-of-the-art and future perspective," *Sensors*, vol. 20, no. 12, p. 3358, 2020. [Online]. Available: https://doi.org/10.3390/s20123358

[113] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic mapping studies in software engineering," in *12th International Conference on Evaluation and Assessment in Software Engineering, EASE 2008, University of Bari, Italy, 26-27 June 2008*, ser. Workshops in Computing, G. Visaggio, M. T. Baldassarre, S. G. Linkman, and M. Turner, Eds. BCS, 2008. [Online]. Available: http://ewic.bcs.org/content/ConWebDoc/19543

[114] V. Anilkumar, J. A. Joji, A. Afzal, and R. Sheik, "Blockchain simulation and development platforms: Survey, issues and challenges," *2019 International Conference on Intelligent Computing and Control Systems, ICCS 2019*, pp. 935–939, 5 2019. [Online]. Available: http://dx.doi.org/10.1109/iccs45141.2019.9065421

[115] H. Wan, K. Li, and Y. Huang, "Blockchain: A review from the perspective of operations researchers," in *Winter Simulation Conference, WSC 2022, Singapore, December 11-14, 2022*. IEEE, 2022, pp. 283–297. [Online]. Available: https://doi.org/10.1109/WSC57314.2022.10015500

[116] S. Smetanin, A. Ometov, N. Kannengießer, B. Sturm, M. M. Komarov, and A. Sunyaev, "Modeling of distributed ledgers: Challenges and future perspectives," in *22nd IEEE Conference on Business Informatics, CBI 2020, Antwerp, Belgium, June 22-24, 2020. Volume 1*. IEEE, 2020, pp. 162–171. [Online]. Available: https://doi.org/10.1109/CBI49978.2020.00025

[117] C. Fan, S. Ghaemi, H. Khazaei, and P. Musílek, "Performance evaluation of blockchain systems: A systematic survey," *IEEE Access*, vol. 8, pp. 126 927–126 950, 2020. [Online]. Available: https://doi.org/10.1109/ACCESS.2020.3006078

[118] R. Paulavicius, S. Grigaitis, and E. Filatovas, "A systematic review and empirical analysis of blockchain simulators," *IEEE Access*, vol. 9, pp. 38 010–38 028, 2021. [Online]. Available: https://doi.org/10.1109/ACCESS.2021.3063324

[119] M. Alharby, A. Aldweesh, and A. van Moorsel, "Blockchain-based smart contracts: A systematic mapping study of academic research (2018)," in *2018 International Conference on Cloud Computing, Big Data and Blockchain (ICCBB)*. IEEE, 2018, pp. 1–6. [Online]. Available: http://dx.doi.org/10.1109/iccbb.2018.8756390

[120] A. Miller and R. Jansen, "Shadow-bitcoin: Scalable simulation via direct execution of multi-threaded applications," *IACR Cryptol. ePrint Arch.*, p. 469, 2015. [Online]. Available: https://dl.acm.org/doi/10.5555/2831120.2831127

[121] K. Wang and H. S. Kim, "Fastchain: Scaling blockchain system with informed neighbor selection," in *IEEE International Conference on Blockchain, Blockchain 2019, Atlanta, GA, USA, July 14-17, 2019*. IEEE, 2019, pp. 376–383. [Online]. Available: https://doi.org/10.1109/Blockchain.2019.00058

[122] L. Stoykov, K. Zhang, and H.-A. Jacobsen, "Vibes: fast blockchain simulations for large-scale peer-to-peer networks: demo," in *Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference: Posters and Demos*, ser. Middleware '17. ACM, 2017. [Online]. Available: http://dx.doi.org/10.1145/3155016.3155020

[123] A. Deshpande, P. Nasirifard, and H. Jacobsen, "evibes: Configurable and interactive ethereum blockchain simulation framework," in *Proceedings of the 19th International Middleware Conference (Posters), Middleware 2018, Rennes, France, December 10-14, 2018*. ACM, 2018, pp. 11–12. [Online]. Available: https://doi.org/10.1145/3284014.3284020

[124] P. Piriou and J. Dumas, "Simulation of stochastic blockchain models," in *14th European Dependable Computing Conference, EDCC 2018, Iaşi, Romania, September 10-14, 2018*. IEEE Computer Society, 2018, pp. 150–157. [Online]. Available: https://doi.org/10.1109/EDCC.2018.00035

[125] B. Wang, S. Chen, L. Yao, B. Liu, X. Xu, and L. Zhu, "A simulation approach for studying behavior and quality of blockchain networks," in *Blockchain - ICBC 2018 - First International Conference, Held as Part of the Services Conference Federation, SCF 2018, Seattle, WA, USA, June 25-30, 2018, Proceedings*, ser. Lecture Notes in Computer Science, S. Chen, H. Wang, and L. Zhang, Eds., vol. 10974. Springer, 2018, pp. 18–31. [Online]. Available: https://doi.org/10.1007/978-3-319-94478-4_2

[126] R. Memon, J. Li, and J. Ahmed, "Simulation model for blockchain systems using queuing theory," *Electronics*, vol. 8, no. 2, p. 234, 2019. [Online]. Available: http://dx.doi.org/10.3390/electronics8020234

[127] M. Alharby and A. van Moorsel, "Blocksim: A simulation framework for blockchain systems," *SIGMETRICS Perform. Evaluation Rev.*, vol. 46, no. 3, pp. 135–138, 2018. [Online]. Available: https://doi.org/10.1145/3308897.3308956

[128] ——, "Blocksim: An extensible simulation tool for blockchain systems," *Frontiers Blockchain*, vol. 3, p. 28, 2020. [Online]. Available: https://doi.org/10.3389/fbloc.2020.00028

[129] J. Polge, S. Ghatpande, S. Kubler, J. Robert, and Y. L. Traon, "Blockperf: A hybrid blockchain emulator/simulator framework," *IEEE Access*, vol. 9, pp. 107 858–107 872, 2021. [Online]. Available: https://doi.org/10.1109/ACCESS.2021.3101044

[130] P. Ramachandran, N. Agrawal, O. Biçer, and A. Küpçü, "Blocksim-net: a network-based blockchain simulator," *Turkish J. Electr. Eng. Comput. Sci.*, vol. 30, no. 2, pp. 392–405, 2022. [Online]. Available: https://doi.org/10.3906/elk-2105-184

[131] C. Faria and M. Correia, "Blocksim: Blockchain simulator," in *IEEE International Conference on Blockchain, Blockchain 2019, Atlanta, GA, USA, July 14-17, 2019*. IEEE, 2019, pp. 439–446. [Online]. Available: https://doi.org/10.1109/Blockchain.2019.00067

[132] S. M. Fattahi, A. Makanju, and A. M. Fard, "SIMBA: an efficient simulator for blockchain applications," in *50th Annual IEEE-IFIP International Conference on Dependable Systems and Networks, DSN 2020, Valencia, Spain, June 29 - July 2, 2020 - Supplemental Volume*. IEEE, 2020, pp. 51–52. [Online]. Available: https://doi.org/10.1109/DSN-S50200.2020.00028

[133] S. Pandey, G. Ojha, B. Shrestha, and R. Kumar, "Blocksim: A practical simulation tool for optimal network design, stability and planning," in *IEEE International Conference on Blockchain and Cryptocurrency, ICBC 2019, Seoul, Korea (South), May 14-17, 2019*. IEEE, 2019, pp. 133–137. [Online]. Available: https://doi.org/10.1109/BLOC.2019.8751320

[134] L. Alsahan, N. Lasla, and M. Abdallah, "Local bitcoin network simulator for performance evaluation using lightweight virtualization," in *IEEE International Conference on Informatics, IoT, and Enabling Technologies, ICIoT 2020, Doha, Qatar, February 2-5, 2020*. IEEE, 2020, pp. 355–360. [Online]. Available: https://doi.org/10.1109/ICIoT48696.2020.9089630

[135] B. Wang, S. Chen, L. Yao, and Q. Wang, *ChainSim: A P2P Blockchain Simulation Framework*. Springer Singapore, 2021, p. 1–16. [Online]. Available: http://dx.doi.org/10.1007/978-981-33-6478-3_1

[136] D. K. Gouda, S. Jolly, and K. Kapoor, "Design and validation of blockeval, A blockchain simulator," in *13th International Conference on COMmunication Systems & NETworkS, COMSNETS 2021, Bangalore, India, January 5-9, 2021*. IEEE, 2021, pp. 281–289. [Online]. Available: https://doi.org/10.1109/COMSNETS51098.2021.9352838

[137] Y. Aoki, K. Otsuki, T. Kaneko, R. Banno, and K. Shudo, "Simblock: A blockchain network simulator," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops, INFOCOM Workshops 2019, Paris, France, April 29 - May 2, 2019*. IEEE, 2019, pp. 325–329. [Online]. Available: https://doi.org/10.1109/INFCOMW.2019.8845253

[138] M. Basile, G. Nardini, P. Perazzo, and G. Dini, "On improving simblock blockchain simulator," in *IEEE Symposium on Computers and Communications, ISCC 2021, Athens, Greece, September 5-8, 2021*. IEEE, 2021, pp. 1–6. [Online]. Available: https://doi.org/10.1109/ISCC53001.2021.9631470

[139] R. Banno and K. Shudo, "Simulating a blockchain network with simblock," in *IEEE International Conference on Blockchain and Cryptocurrency, ICBC 2019, Seoul, Korea (South), May 14-17, 2019*. IEEE, 2019, pp. 3–4. [Online]. Available: https://doi.org/10.1109/BLOC.2019.8751431

[140] M. Zander, T. Waite, and D. Harz, "Dagsim: Simulation of dag-based distributed ledger protocols," *SIGMETRICS Perform. Evaluation Rev.*, vol. 46, no. 3, pp. 118–121, 2018. [Online]. Available: https://doi.org/10.1145/3308897.3308951

[141] IOTA. (2022) Research papers | iota. [Online]. Available: https://www.iota.org/foundation/research-papers

[142] M. Bertoli, G. Casale, and G. Serazzi, "JMT: performance engineering tools for system modeling," *SIGMETRICS Perform. Evaluation Rev.*, vol. 36, no. 4, pp. 10–15, 2009. [Online]. Available: https://doi.org/10.1145/1530873.1530877

[143] SonarQube. (2022) Code quality and code security | sonarqube. [Online]. Available: https://www.sonarqube.org/

[144] A. Danial, "cloc," 2021. [Online]. Available: https://github.com/AlDanial/cloc

[145] B. R. Haverkort, *Performance of Computer Communication Systems: A Model-Based Approach*. Wiley, 2001. [Online]. Available: http://dx.doi.org/10.1002/0470841923

[146] S. Ferretti and G. D'Angelo, "On the ethereum blockchain structure: A complex networks theory perspective," *Concurr. Comput. Pract. Exp.*, vol. 32, no. 12, 2020. [Online]. Available: https://doi.org/10.1002/cpe.5493

[147] M. Harbers, M. S. Bargh, R. Pool, J. V. Berkel, S. W. van den Braak, and S. Choenni, "A conceptual framework for addressing iot threats: Challenges in meeting challenges," in *51st Hawaii International Conference on System Sciences, HICSS 2018, Hilton Waikoloa Village, Hawaii, USA, January 3-6, 2018*, T. Bui, Ed. ScholarSpace / AIS Electronic Library (AISeL), 2018, pp. 1–10. [Online]. Available: https://hdl.handle.net/10125/50166

[148] Q. Zhu, S. W. Loke, R. Trujillo-Rasua, F. Jiang, and Y. Xiang, "Applications of distributed ledger technologies to the internet of things: A survey," *ACM Comput. Surv.*, vol. 52, no. 6, pp. 120:1–120:34, 2020. [Online]. Available: https://doi.org/10.1145/3359982

[149] S. G. H. Soumyalatha, "Study of iot: understanding iot architecture, applications, issues and challenges," in *1st International Conference on Innovations in Computing & Net-working (ICICN16), CSE, RRCE. International Journal of Advanced Networking & Applications*, vol. 478, 2016.

[150] J. W. Creswell and V. L. P. Clark, *Designing and conducting mixed methods research*. Sage publications, 2017.

[151] B. J. Oates, M. Griffiths, and R. McLean, *Researching information systems and computing*. Sage, 2022.

[152] V. Braun and V. Clarke, "Using thematic analysis in psychology," *Qualitative Research in Psychology*, vol. 3, no. 2, pp. 77–101, 2006. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1191/1478088706qp063oa

[153] A. Sharma, S. Kaur, and M. Singh, "A comprehensive review on blockchain and internet of things in healthcare," *Trans. Emerg. Telecommun. Technol.*, vol. 32, no. 10, 2021. [Online]. Available: https://doi.org/10.1002/ett.4333

[154] H. Zhang and K. Sakurai, "Blockchain for iot-based digital supply chain: A survey," in *Advances in Internet, Data and Web Technologies, The 8th International Conference on Emerging Internet, Data and Web Technologies, EIDWT 2020, Kitakyushu, Japan. 24-26 February 2020*, ser. Lecture Notes on Data Engineering and Communications Technologies, L. Barolli, Y. Okada, and F. Amato, Eds., vol. 47. Springer, 2020, pp. 564–573. [Online]. Available: https://doi.org/10.1007/978-3-030-39746-3_57

[155] S. Huckle, R. Bhattacharya, M. White, and N. Beloff, "Internet of things, blockchain and shared economy applications," in *The 7th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN 2016)/The 6th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH-2016)/Affiliated Workshops, September 19-22, 2016, London, United Kingdom*, ser. Procedia Computer Science, E. M. Shakshuki, Ed., vol. 98. Elsevier, 2016, pp. 461–466. [Online]. Available: https://doi.org/10.1016/j.procs.2016.09.074

[156] A. Alzubaidi, E. Solaiman, P. Patel, and K. Mitra, "Blockchain-based SLA management in the context of iot," *IT Prof.*, vol. 21, no. 4, pp. 33–40, 2019. [Online]. Available: https://doi.org/10.1109/MITP.2019.2909216

[157] V. R. B.-G. Caldiera and H. D. Rombach, "Goal question metric paradigm," *Encyclopedia of software engineering*, vol. 1, no. 528-532, p. 6, 1994.

[158] A. Alqahtani, E. Solaiman, P. Patel, S. Dustdar, and R. Ranjan, "Service level agreement specification for end-to-end iot application ecosystems," *Softw. Pract. Exp.*, vol. 49, no. 12, pp. 1689–1711, 2019. [Online]. Available: https://doi.org/10.1002/spe.2747

[159] R. Van Solingen and E. W. Berghout, *The Goal/Question/Metric Method: a practical guide for quality improvement of software development*. McGraw-Hill, 1999.

[160] L. Trautmann and R. Lasch, *Blockchain-based Smart Contracts in Procurement: A Technology Readiness Level Analysis*. Wiesbaden: Springer Fachmedien Wiesbaden, 2021, pp. 133–170. [Online]. Available: https://doi.org/10.1007/978-3-658-32895-5_6

[161] M. Chernyshev, Z. A. Baig, O. Bello, and S. Zeadally, "Internet of things (iot): Research, simulators, and testbeds," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 1637–1647, 2018. [Online]. Available: https://doi.org/10.1109/JIOT.2017.2786639

[162] H. Sukhwani, N. Wang, K. S. Trivedi, and A. J. Rindos, "Performance modeling of hyperledger fabric (permissioned blockchain network)," in *17th IEEE International Symposium on Network Computing and Applications, NCA 2018, Cambridge, MA, USA, November 1-3, 2018*. IEEE, 2018, pp. 1–8. [Online]. Available: https://doi.org/10.1109/NCA.2018.8548070

[163] X. Guo and P. Hao, "Using a random forest model to predict the location of potential damage on asphalt pavement," *Applied Sciences*, vol. 11, no. 21, p. 10396, 2021. [Online]. Available: https://doi.org/10.3390/app112110396

[164] A. Hamed, M. Tahoun, and H. Nassar, "Knnhi: Resilient knn algorithm for heterogeneous incomplete data classification and k identification using rough set theory," *Journal of Information Science*, vol. 49, no. 6, p. 1631–1655, 2022. [Online]. Available: http://dx.doi.org/10.1177/01655515211069539

[165] A. Hamed, A. Sobhy, and H. Nassar, "Accurate classification of covid-19 based on incomplete heterogeneous data using a knn variant algorithm," *Arabian Journal for Science and Engineering*, vol. 46, no. 9, p. 8261–8272, 2021. [Online]. Available: http://dx.doi.org/10.1007/s13369-020-05212-z

[166] R. Panigrahi, S. K. Kuanar, and L. Kumar, *Method Level Refactoring Prediction by Weighted-SVM Machine Learning Classifier*. Springer Nature Singapore, 2023, p. 93–104. [Online]. Available: http://dx.doi.org/10.1007/978-981-19-6893-8_8

[167] C. Rodríguez-Gallego, F. D. Muñoz, M. L. M. Ruiz, A. Gabaldón, M. Dolón-Poza, and I. Pau, "A collaborative semantic framework based on activities for the development of applications in smart home living labs," *Future Gener. Comput. Syst.*, vol. 140, pp. 450–465, 2023. [Online]. Available: https://doi.org/10.1016/j.future.2022.10.027

[168] O. A. M. Salem, F. Liu, Y. P. Chen, A. Hamed, and X. Chen, "Effective fuzzy joint mutual information feature selection based on uncertainty region for classification problem," *Knowl. Based Syst.*, vol. 257, p. 109885, 2022. [Online]. Available: https://doi.org/10.1016/j.knosys.2022.109885

[169] A. Hamed and M. F. Mohamed, "A feature selection framework for anxiety disorder analysis using a novel multiview harris hawk optimization algorithm," *Artif. Intell. Medicine*, vol. 143, p. 102605, 2023. [Online]. Available: https://doi.org/10.1016/j.artmed.2023.102605

[170] A. H. Attia and H. M. Nassar, "Efficient feature selection for inconsistent heterogeneous information systems based on a grey wolf optimizer and rough set theory," *Soft Comput.*, vol. 25, no. 24, pp. 15 115–15 130, 2021. [Online]. Available: https://doi.org/10.1007/s00500-021-06375-z

[171] V. S. Handur and S. L. Deshpande, *Artificial Bee Colony Optimization-Based Load Balancing in Distributed Computing Systems—A Survey*. Springer Nature Singapore, 2022, p. 733–740. [Online]. Available: http://dx.doi.org/10.1007/978-981-16-9967-2_69

[172] F. Karimi, M. B. Dowlatshahi, and A. Hashemi, "Semiaco: A semi-supervised feature selection based on ant colony optimization," *Expert Syst. Appl.*, vol. 214, p. 119130, 2023. [Online]. Available: https://doi.org/10.1016/j.eswa.2022.119130

[173] D. R. Ganesh and M. Chithambarathanu, *A Survey on Hybrid PSO and SVM Algorithm for Information Retrieval*. Springer Nature Singapore, 2022, p. 121–130. [Online]. Available: http://dx.doi.org/10.1007/978-981-19-6004-8_11

[174] G. H. Kumar and P. T. Rao, "An energy efficiency perceptive on MIMO-OFDM systems using hybrid fruit fly-based salp swarm optimization technique," *Concurr. Comput. Pract. Exp.*, vol. 35, no. 1, 2023. [Online]. Available: https://doi.org/10.1002/cpe.7416

[175] A. H. Attia, A. S. Sherif, and H. Nassar, "Distributed approach for computing rough set approximations of big incomplete information systems," *Inf. Sci.*, vol. 547, pp. 427–449, 2021. [Online]. Available: https://doi.org/10.1016/j.ins.2020.08.049

Appendix A

Reference Implementation of the Blockchain-based IoT Simulation Framework

This appendix demonstrates the implementation of the proposed blockchain-based IoT simulation architecture, presented in Chapter 5. It is noteworthy that the simulator is designed to provide the ability to model and evaluate the performance of scenarios where blockchain is integrated with IoT. To prevent reinventing the wheel, encouraged by feedback received from relevant experts, the implementation extends a recognised IoT simulator, namely: IoTsim-Osmosis [16], which happens to also extend on a well-established simulation framework called CloudSim [86]. Therefore, the implementation uses the same programming language as used in the underlying simulators to support blockchain aspects and performance evaluation features. Eclipse is the main IDE (Integrated Development Environment) used for developing, testing, and deploying the implementation of the simulator architecture. The implementation is publicly available on a GitHub repository to provide a reference implementation for the conceptualised simulation design proposed by this thesis and to encourage the research community to introduce further enhancement and contribution based on it. The following highlights the implementation in further detail.

A.1 Simulation Project Hierarchy

Figure A.1 provides a structured overview of blockchain-based IoT within the IDE that illustrates the packages and classes developed to simulate integrated IoT and blockchain environments, as presented in Sections A.2.1, A.2.4, and A.2.5.

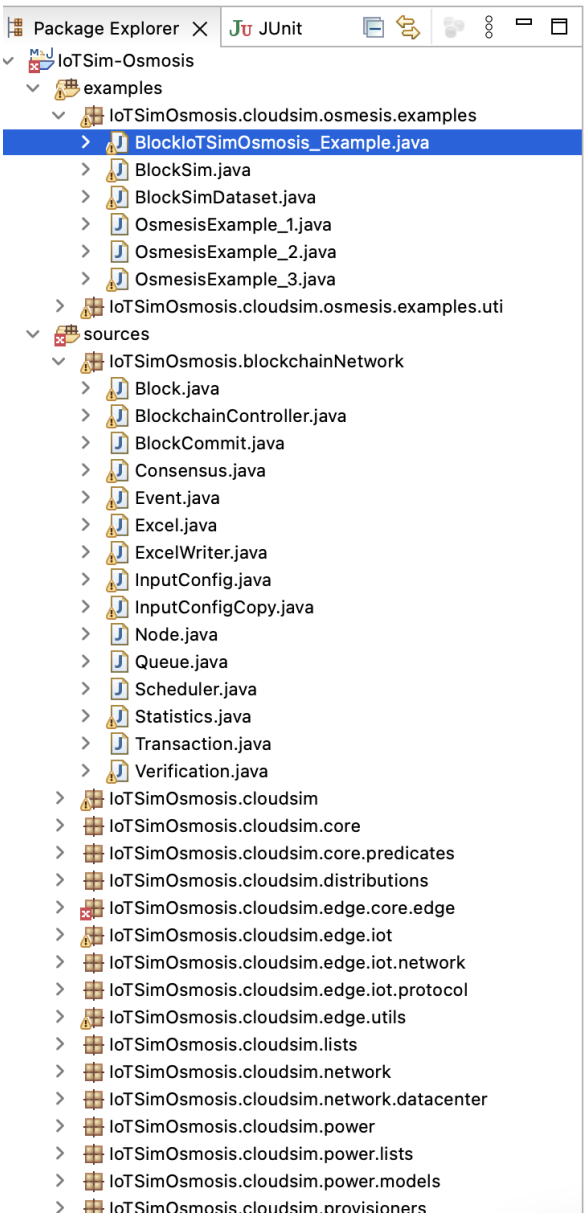


Fig. A.1 Simulation Project Hierarchy

A.2 Implementation Code

This section lists major Java classes of the implementations which play vital roles in operating the simulator. For clarity and self-explanation, the listed code of each class follows the best practice of naming convention to indicate the purposes of its elements such as constructors, functions, arrays, variables, and others (e.g. Nodes, Blocks, transactions, consensus, performance, events, etc.). Moreover, the majority of the listed codes are annotated with relevant comments that are intended to help the reader comprehend their purposes. The following distributes relevant implementation Java class to each of the primary components of the proposed simulator conceptual design, which are Configurator, Simulation Core, Generator, and Report. The underlying IoTsim-Osmosis simulator role is also highlighted wherever applicable.

A.2.1 Configurator

This section is divided into two main subsections, namely Blockchain Configurations and IoT Configurations, as follows.

A.2.1.1 Blockchain Configurations

The InputConfig class is a configuration class designed to set up and customise the parameters of a blockchain network simulator. It focusses primarily on initialising various simulation parameters, such as the number of nodes and miners in the network, transaction characteristics (including the number and size of transactions), block parameters (e.g., block size), the consensus algorithm used, and overall simulation settings (such as simulation time).

```
1 package IoTSimOsmosis.blockchainNetwork;
2
3 import java.lang.reflect.Array;
4 import java.util.ArrayList;
5 import java.util.Random;
6 import java.util.concurrent.ThreadLocalRandom;
```

```
7
8  public class InputConfig {
9
10   /**
11    * @param numberOfNodes
12    * @param numberOfMiner
13    *
14    *        Miner is part of total number of nodes e.g. No.nodes
          5 and Miner 2
15    * (total number of nodes and miner is 5)
16    */
17   /**************** To Create Node Parameters ****************/
18   private static int numberOfNodes = 4;
19   private static int numberOfMiner = 1;
20
21   /**
22    *
23    * @param transactionNumber : Maximum number of transactions
         created during
24    *     running simulator.
25    * @param maxTXsize : Maximum Transaction size in MB.
26    * @param minTXsize : Minimum Transaction size in MB.
27    * @param TransactionGasLimit : maximum amount of gas units
         the transaction
28    *     can used.
29    *
30    */
31   /************* To Create Transaction Parameters ************/
32   private static final double transactionDelay
33       = 0.05; // average transaction propagation delay in
34               // seconds
```

```
35  private static final double maxTXsize
36      = 0.064; // Maximum Transaction size in MB (64KB quorum)
37  private static final double minTXsize
38      = 0.001; // Minimum Transaction size in MB (1 KB)
39
40  /**
41   *
42   * @param maxBlockSize
43   * @param blockGasLimit
44   * @param blockInterval
45   *
46   */
47  /************* To Create Block Parameters ************/
48  private static final long maxBlockSize = 1; //
       4194304;//4194304;
49  private static final double blockGasLimit = 1000000; //
       30000000;//30000000;
50  private static double Binterval = 12; // 12.41 raft(50ms)
51
52  /**
53   * @param consensusAlgorithm : PoW and raft
54   */
55  /************* To configure consensus Algorithm ************/
56  private static final String consensusAlgorithm = "raft"; //
       raft or PoW
57
58  /**
59   *
60   * @param simTime
61   * @param simulatorRun
62   */
```

```
63  /************** To configure simulator ************/
64  static int simTime = 500;
65  private static final int simulatorRun = 1;
66
67  public static int getSimulatorRun() { return simulatorRun; }
68
69  public static double getBlockInterval() { return Binterval; }
70
71  /**
72   * Return the Number of nodes that created
73   *
74   * @return numberOfNodes
75   */
76  public static int getNumberOfNodes() { return numberOfNodes; }
77
78  /**
79   * Return the number of Miner is created
80   *
81   * @return
82   */
83  public static int getNumberOfMiner() { return numberOfMiner; }
84
85  /**
86   * Return the maximum block gas limit
87   *
88   * @return
89   */
90  public static double getBlockGasLimit() { return blockGasLimit
       ; }
91
92  /**
```

```
93      * Return the maximum of block size
94      *
95      * @return maxBlockSize
96      */
97     public static long getMaxBlockSize() { return maxBlockSize; }
98
99     /**
100     * Return the minimum of transaction size
101     *
102     * @return minTXsize
103     */
104    public static double getMinTransactionSize() { return
           minTXsize; }
105
106    /**
107     * Return the maximum of transaction size
108     *
109     * @return maxTXsize
110     */
111    public static double getMaxTransactionSize() { return
           maxTXsize; }
112
113    /**
114     * Return the simulation time that is configured
115     *
116     * @return simTime
117     */
118    public static int getSimulationTime() { return simTime; }
119
120    /**
121     * Return the consensus algorithm that used in the simulator
```

```
122     *
123     * @return consensusAlgorithm
124     */
125    public static String getConsensusalgorithm() { return
           consensusAlgorithm; }
126
127    public static double getTransactiondelay() { return
           transactionDelay; }
128
129    public static void setSimulationTime(int simTime)
130    {
131      InputConfig.simTime = simTime;
132    }
133
134    public static void setNumberOfNodes(int numberOfNodes)
135    {
136      InputConfig.numberOfNodes = numberOfNodes;
137    }
138
139    public static void setNumberOfMiner(int numberOfMiner)
140    {
141      InputConfig.numberOfMiner = numberOfMiner;
142    }
143
144    public static void setBinterval(double binterval) { Binterval
           = binterval; }
145 }
```

A.2.1.2 IoT Configurations

The IoT Configurations is designed to set up parameters for an IoT environment. It includes the identification and functions of IoT devices, data transmission rates, duration of data generation activities, and data processing capacities of different layers.

OsmesisApp	ID	NumOfLayer (IoT, edges, clouds)	DataRate_Sec	StopDataGeneration_Sec	IoTDevice	IoTLayer Name	OsmesisIoTLet_MI	IoTDeviceOutputData_Mb	MELName	datacenter	Osmesis Edgelet_MI	MELOutputData_Mb	VmName	datacenter	Osmesis Cloudlet_MI
FireFighting	1	3	1.2	250	Flame_sensor	IoT_1	50	90	MEL_1	Edge_1	250	70	VM_1	Cloud_1	200

Fig. A.2 IoT-based Firefighting Configurations

A.2.2 IoTsim-Osmosis

The generateIoTData method is designed for the generation of data from an IoT environment and its subsequent processing and recording on a blockchain, achieved by invoking the BlockchainController A.2.3.

```
1  static int count = 0;
2  private void generateIoTData(SimEvent ev)
3  {
4    OsmesisAppDescription app = (OsmesisAppDescription)ev.getData
       ();
5    count++;
6    if (CloudSim.clock() <= app.getStopDataGenerationTime()
7        && !app.getIsIoTDeviceDied()) {
8      sendNow(app.getIoTDeviceId(), OsmosisTags.SENSING, app);
9      double dealy = app.getDataRate();
10     send(this.getId(), dealy, OsmosisTags.GENERATE_OSMESIS, app)
         ;
11     BlockchainController.createOsmosisTransaction(ev.eventTime()
         );
12
13   } else {
14   }
```

```
15 }
```

A.2.3 Generator

The BlockchainController class is designed for the IoTSim-Osmosis framework to orchestrate the dynamic aspects of the blockchain simulation, including the generation of nodes and transactions and managing the state of the blockchain network during simulations.

```
1  package IoTSimOsmosis.blockchainNetwork;
2
3  import IoTSimOsmosis.cloudsim.core.SimEvent;
4  import IoTSimOsmosis.osmosis.core.Infrastructure.EdgeDatacenters
       ;
5  import IoTSimOsmosis.osmosis.core.OsmosisBuilder;
6  import java.text.DateFormat;
7  import java.text.SimpleDateFormat;
8  import java.util.Calendar;
9  import java.util.Random;
10 import java.util.concurrent.ThreadLocalRandom;
11
12 public class BlockchainController {
13   static Random rand = new Random();
14   static Calendar now = Calendar.getInstance();
15   static DateFormat df = new SimpleDateFormat("dd-MM-yyy-HH:mm")
         ;
16
17   /**
18    * This method is responsible for generate light node
19    */
20   public static void generateNodes()
21   {
22
```

```
23      if (InputConfig.getConsensusalgorithm().equals("raft")) {
24        if (InputConfig.getNumberOfNodes() >= 3
25            && InputConfig.getNumberOfNodes() > InputConfig.
                  getNumberOfMiner()) {
26
27          for (int i = 0; i < InputConfig.getNumberOfNodes(); i++)
                {
28            String time = df.format(now.getTime());
29            Node.getNodes().add(new Node(i, "follower", time));
30          }
31        }
32      } else if (InputConfig.getConsensusalgorithm().equals("PoW")
           ) {
33        for (int i = 0; i < InputConfig.getNumberOfNodes(); i++) {
34          String time = df.format(now.getTime());
35          Node.getNodes().add(new Node(i, "node", time));
36        }
37      }
38      Consensus.consensus(InputConfig.getConsensusalgorithm());
39    }
40
41    /**
42     * This method is responsible for generate light node
43     */
44    public static void generateOsmosisNodes(OsmosisBuilder
         topologyBuilder)
45    {
46
47      int dgeNodes = topologyBuilder.getEdgeDatacentres().size();
48      Calendar now = Calendar.getInstance();
49      DateFormat df = new SimpleDateFormat("dd-MM-yyy-HH:mm");
```

```
50
51      for (int i = 0; i < dgeNodes; i++) {
52        String time = df.format(now.getTime());
53        Node.getNodes().add(new Node(i, "follower", time));
54      }
55      Consensus.consensus(InputConfig.getConsensusalgorithm());
56    }
57
58    /**
59     * This method is responsible for generate transactions
          without an integrated
60     */
61
62    public static void createOsmosisTransaction(double
         osmosisTransactionTime)
63    {
64
65      for (int i = 0; i < InputConfig.getTransactionNumber(); i++)
            {
66        double creationTime = osmosisTransactionTime;
67        double transactionSize
68            = ThreadLocalRandom.current().nextDouble(100, 1000);
69        Transaction tx = new Transaction(osmosisTransactionTime);
70        Node.getNodes().get(0).getTransactionsPool().add(tx);
71
72        for (Node n : Node.getNodes()) {
73          if (n.getNodeType().equals("leader")
74              || n.getNodeType().equals("miner")) {
75            n.getTransactionsPool().add(tx);
76          }
77        }
```

```
78        Statistics.noTransactionsConfig += 1;
79      }
80
81      public static void creatTransactions()
82      {
83        int countTransaction = 0; // to count number of
              transaction per second
84        double maxTxTime = 0.89;
85        double minTxTime = 0;
86        int Psize = InputConfig.getSimulationTime()
87            * InputConfig.getTransactionNumber();
88        int i = 0;
89        while (i < Psize) {
90          if (countTransaction < InputConfig.getTransactionNumber
                ()) {
91            double transactionCreatingTime
92                = ThreadLocalRandom.current().nextDouble(0,
                      maxTxTime);
93            Transaction tx = new Transaction(
                  transactionCreatingTime);
94            for (Node n : Node.getNodes()) {
95              if (n.getNodeType().equals("leader")
96                  || n.getNodeType().equals("miner")) {
97                n.getTransactionsPool().add(tx);
98              }
99            }
100           countTransaction += 1;
101           i += 1;
102         } else {
103           countTransaction = 0;
104           maxTxTime += 1;
```

```
105             minTxTime += 1;
106           }
107         }
108       }
109
110       /**
111        * This method is responsible for generate transactions
              without an
112        * integrated
113        */
114
115       public static void creatTransactionsWithoutIntegrated()
116       {
117
118         for (int i = 0; i < InputConfig.getTransactionNumber(); i
              ++) {
119           double creationTime = ThreadLocalRandom.current().
                nextDouble(
120               0, InputConfig.getTransactionNumber() - 1);
121           Transaction tx = new Transaction(creationTime);
122
123           for (Node n : Node.getNodes()) {
124             if (n.getNodeType().equals("leader")
125                 || n.getNodeType().equals("miner")) {
126               n.getTransactionsPool().add(tx);
127             }
128           }
129         }
130       }
131
132       /**
```

```
133      * This method is responsible for generate transactions with
             IoT simulator
134      */
135
136     public static void creatTransactionsWithIntegrated(
137         double transactionTime, int fromAddress, int toAddress)
138     {
139
140       double creationTime = transactionTime;
141       double transactionSize = ThreadLocalRandom.current().
             nextDouble(
142           InputConfig.getMinTransactionSize(),
143           InputConfig.getMaxTransactionSize());
144       Transaction tx = new Transaction(creationTime);
145     }
146
147     /**
148      *
149      */
150     public static void restState() { Node.getNodes().clear(); }
151   }
```

A.2.4 Simulation Core

The Simulation Core includes a collection of classes designed to simulate blockchain-based IoT environments. The following subsections present these classes and their operations.

A.2.4.1 Block Class

Block class defines the structure and behaviour of a blockchain-based IoT block by a set of attributes such as block ID, previous block ID, miner node, transaction list and block size.

```
1  package IoTSimOsmosis.blockchainNetwork;
2
3  import java.util.ArrayList;
4  import java.util.List;
5
6  public class Block {
7    // Block ID for each block created
8    private long blockID;
9    // The previous block ID
10   private long previousBlockID;
11   // the index of block in the local BCL
12   private int blockDepth;
13   // the time when the block is created
14   private double blockTimestamp;
15   // Miner node who mined (created) the block
16   private Node miner;
17   // list of transaction that included in the block
18   private ArrayList<Transaction> transactions;
19   // The block size
20   private double blockSize;
21   // The block gas limit
22   private double blockGas;
23   // The block used gas limit
24   // private double blockUsedGas;
25   // is block include Tx?
26   private boolean hasTx;
27
28   /**
29    * A constructor method for block class
30    */
```

```
31    public Block()
32    {
33      this.blockID = 0;
34      this.previousBlockID = -1;
35      this.blockDepth = 0;
36      this.blockTimestamp = 0.0;
37      this.miner = null; // need to change just for test
38      this.transactions = new ArrayList<>();
39    }
40
41    /**
42     * Return block ID
43     *
44     * @return depth
45     */
46    public long getBlockID() { return blockID; }
47
48    /**
49     * Return Previous BlocK ID
50     *
51     * @return previousBlocKID
52     */
53    public long getPreviousBlockID() { return previousBlockID; }
54
55    /**
56     * Return block depth which pointer the index of block in the
          local BCL
57     *
58     * @return depth
59     */
60    public int getBlockDepth() { return blockDepth; }
```

```
61
62    /**
63     * Return block Timestamp
64     *
65     * @return blockTimestamp
66     */
67    public double getBlockTimestamp() { return blockTimestamp; }
68
69    /**
70     * Return miner who created the block
71     *
72     * @return miner
73     */
74    public Node getMiner() { return miner; }
75
76    /**
77     * Return transactions that included in the block
78     *
79     * @return transactions
80     */
81    public ArrayList<Transaction> getTransactions() { return
          transactions; }
82
83    /**
84     * Return block size
85     *
86     * @return blockSize
87     */
88    public double getBlockSize() { return blockSize; }
89
90    /**
```

```
91      * Return block gas
92      *
93      * @return blockGas
94      */
95     public double getBlockGas() { return blockGas; }
96
97     /**
98      * is the block include Tx
99      *
100     * @return boolean hasTx
101     */
102    public boolean getHasTx() { return hasTx; }
103
104    /**
105     * Set block ID
106     *
107     * @param id
108     */
109    public void setBlockID(long id) { this.blockID = id; }
110
111    /**
112     * Set Previous Block ID
113     *
114     * @param previous
115     */
116    public void setPreviousBlockID(long previousID)
117    {
118      this.previousBlockID = previousID;
119    }
120
121    /**
```

```
122     * Set block depth
123     *
124     * @param depth
125     */
126    public void setBlockDepth(int depth) { this.blockDepth = depth
           ; }
127
128    /**
129     * set the miner who created the block
130     *
131     * @param miner
132     */
133    public void setMiner(Node miner) { this.miner = miner; }
134
135    /**
136     * set the transactions that included to the block
137     *
138     * @param transactions
139     */
140    public void setTransactions(ArrayList<Transaction>
           transactions)
141    {
142      this.transactions = transactions;
143    }
144
145    /**
146     * set block size
147     *
148     * @param size
149     */
```

```
150   public void setBlockSize(double size) { this.blockSize = size;
          }
151
152   /**
153    * set block gas
154    *
155    * @param blockgas
156    */
157   public void setBlockGas(double blockGas) { this.blockGas =
          blockGas; }
158
159   /**
160    * set block Timestamp
161    *
162    * @param timestamp
163    */
164   public void setBlockTimestamp(double timestamp)
165   {
166     this.blockTimestamp = timestamp;
167   }
168
169   /**
170    * set the funcation if block includes Tx (true)
171    *
172    * @param hasTx
173    */
174   public void setHasTx(boolean hasTx) { this.hasTx = hasTx; }
175 }
```

A.2.4.2 Block Commit Class

The Block Commit class is used to handle the generation of new blocks, committing them to the blockchain, and propagating them across the network. Therefore, it includes methods for processing blocks based on the current consensus algorithm, updating node ledgers, and managing transaction pools.

```
1  package IoTSimOsmosis.blockchainNetwork;
2
3  /**
4   *
5   * @author adelalbshri
6   *
7   */
8
9  public class BlockCommit {
10
11   /**
12    *
13    * @param event
14    */
15   static void generateNewBlock(Event event)
16   {
17     Node miner = event.getBlock().getMiner();
18     double eventTime = event.getTime();
19     long blockPrevious = event.getBlock().getPreviousBlockID();
20
21     if (blockPrevious == miner.getLastBlock().getBlockID()) {
22       if (InputConfig.getConsensusalgorithm() == "PoW") {
23         event.getBlock().setTransactions(Transaction.
              executeTranscationsPoW(
24             miner, event.getBlock(), eventTime));
```

```
25          event.getBlock().setBlockGas(Transaction.blockGaslimit);
26        } else if (InputConfig.getConsensusalgorithm() == "raft")
             {
27          event.getBlock().setTransactions(Transaction.
               executeTranscationsRaft(
28              miner, event.getBlock(), eventTime));
29          event.getBlock().setBlockSize(Transaction.
               getBlockSizeLimit());
30        }
31
32        if (event.getBlock().getTransactions().size() > 0) {
33          event.getBlock().setHasTx(true);
34
35          miner.getBlockchainLedger().add(event.getBlock());
36          propagateBlock(event.getBlock());
37        }
38      }
39      generateNextBlock(miner, eventTime);
40    }
41
42    /**
43     *
44     * @param newBlock
45     */
46    private static void propagateBlock(Block newBlock)
47    {
48      for (Node miner : Node.getNodes()) {
49        if (newBlock.getMiner().getNodeId() != miner.getNodeId())
             {
50
51          Scheduler.receiveBlockEvent(miner, newBlock);
```

```
52        }
53      }
54    }
55
56    /**
57     *
58     * @param event
59     */
60    private static void receiveBlock(Event event)
61    {
62      Node miner = event.getBlock().getMiner();
63      double eventTime = event.getTime();
64      long blockPrevious = event.getBlock().getPreviousBlockID();
65      Node node = event.getMiner();
66      long lastBlockID = node.getLastBlock().getBlockID();
67      if (blockPrevious == lastBlockID) {
68        node.getBlockchainLedger().add(event.getBlock());
69        updateTransactionsPool(miner, event.getBlock());
70        generateNextBlock(node, eventTime);
71      } else {
72        int depth = event.getBlock().getBlockDepth() + 1;
73        if (depth > node.getBlockchainLedger().size()) {
74          updateLocalBlockchainLedger(node, miner, depth);
75          generateNextBlock(node, eventTime);
76        }
77      }
78    }
79
80    /**
81     *
82     * @param node
```

```
83      * @param miner
84      * @param depth
85      */
86     private static void updateLocalBlockchainLedger(
87         Node node, Node miner, int depth)
88     {
89
90       for (int i = 0; i < depth; i++) {
91
92         if (i < node.getBlockchainLedger().size()) {
93           if (node.getBlockchainLedger().get(i).getBlockID()
94               != miner.getBlockchainLedger().get(i).getBlockID())
                     {
95             Block newBlock = miner.getBlockchainLedger().get(i);
96             node.getBlockchainLedger().add(newBlock);
97             updateTransactionsPool(miner, newBlock); // ********
98           }
99         } else {
100          Block block = miner.getBlockchainLedger().get(i);
101          node.getBlockchainLedger().add(block);
102          updateTransactionsPool(miner, block); // **********
103        }
104      }
105    }
106
107    /**
108     *
109     * @param node
110     * @param block
111     */
```

```
112   private static void updateTransactionsPool(Node node, Block
          block)
113   {
114     int i = 0;
115     while (i < (block.getTransactions().size())) {
116       for (int count = 0; count < node.getTransactionsPool().
              size(); count++) {
117         if (block.getTransactions().get(i).getTransactionID()
118             == node.getTransactionsPool().get(count).
                   getTransactionID()) {
119           node.getTransactionsPool().remove(count);
120         }
121       }
122       i += 1;
123     }
124   }
125
126   /**
127    *
128    *
129    */
130   public static void generateInitialEvents()
131   {
132     int currentTime = 0;
133
134     for (Node miner : Node.getNodes()) {
135       if (miner.getNodeType().equals("leader")
136           || miner.getNodeType().equals("miner")) {
137         generateNextBlock(miner, currentTime);
138       }
139     }
```

```
140   }
141
142   /**
143    *
144    *
145    */
146   private static void generateNextBlock(Node miner, double
          currentTime)
147   {
148     if (miner.getNodeType().equals("leader")) {
149       double blockTime = currentTime
150           + Consensus.protocal();
151       Scheduler.createBlockEvent(miner, blockTime);
152     } else if (miner.getNodeType().equals(
153                     "miner") /* && miner.getHashPower()>0 */) {
154       double blockTime
155           = currentTime + Consensus.protocalPoW(miner);
156       Scheduler.createBlockEvent(miner, blockTime);
157     }
158   }
159
160   /**
161    *
162    *
163    */
164   public static void handleEvent(Event event)
165   {
166     if (event.getType() == "create_block") {
167       generateNewBlock(event);
168
169     } else if (event.getType() == "receive_block") {
```

```
170         receiveBlock(event);
171       }
172     }
173 }
```

A.2.4.3 Node Class

Nodes class is used to simulate participants' nodes for blockchain-based IoT by a set of properties and behaviours of nodes such as their ID, type (e.g., leader, miner, follower), and their interaction with the blockchain (e.g., maintaining a local copy of the blockchain ledger, managing a pool of transactions).

```
1  package IoTSimOsmosis.blockchainNetwork;
2
3  import java.util.ArrayList;
4
5  public class Node {
6
7    // node ID
8    private final int nodeID;
9    // Node type
10   private String nodeType;
11
12   private String joinTime;
13
14   private double hashPower;
15
16   // Blockchain Ledger
17   private final ArrayList<Block> blockchainLedger;
18   // transactions Pool
19   private final ArrayList<Transaction> transactionsPool;
20   // Node list
```

```
21  private final static ArrayList<Node> NodesList = new ArrayList
        <>();
22
23  /**
24   *
25   * @param nodeID
26   * @param nodeType
27   * @param joinTime
28   */
29  public Node(int nodeID, String nodeType, String joinTime)
30  {
31
32    this.nodeID = nodeID;
33    this.nodeType = nodeType;
34    this.joinTime = joinTime;
35    this.blockchainLedger = new ArrayList<>();
36    this.transactionsPool = new ArrayList<>();
37  }
38
39  /**
40   * a method to generate genesis Block for all miner in the
        network
41   */
42  public static void generateGenesisBlock()
43  {
44    for (Node node : Node.getNodes()) {
45      node.getBlockchainLedger().add(new Block());
46    }
47  }
48
49  /**
```

```
50      * Return the node ID
51      *
52      * @return nodeID
53      */
54     public int getNodeId() { return nodeID; }
55
56     /**
57      * Return the node type (e.g. light node and miner)
58      *
59      * @return nodeType
60      */
61     public String getNodeType() { return nodeType; }
62
63     /**
64      * Return the the time that node join to blockchain network
65      *
66      * @return joinTime
67      */
68     public String getJoinTime() { return joinTime; }
69
70     public double getHashPower() { return hashPower; }
71
72     public void setHashPower(double hashPower) { this.hashPower =
           hashPower; }
73
74     /**
75      * Return the local blockchain ledger
76      *
77      * @return ArrayList<Block>
78      */
```

```
79  public ArrayList<Block> getBlockchainLedger() { return
        blockchainLedger; }
80
81  /**
82   * Return array of transactions pool
83   *
84   * @return ArrayList<Transaction>
85   */
86  public ArrayList<Transaction> getTransactionsPool()
87  {
88    return transactionsPool;
89  }
90
91  /**
92   * set node type
93   *
94   * @param nodeType
95   */
96  public void setNodeType(String nodeType) { this.nodeType =
        nodeType; }
97
98  /**
99   * Return the last block at the nodes local blockchain
100  *
101  * @return block
102  */
103 public Block getLastBlock()
104 {
105   return this.getBlockchainLedger().get(blockchainLedger.size
          () - 1);
106 }
```

```
107
108   /**
109    * Return an arrayList of node
110    *
111    * @return
112    */
113   public static ArrayList<Node> getNodes() { return NodesList; }
114 }
```

A.2.4.4 Consensus Class

Consensus Class is to implement the different consensus algorithms within the simulation to facilitate the simulation of how consensus is achieved in the network for block validation and addition.

```
1  package IoTSimOsmosis.blockchainNetwork;
2
3  import java.util.ArrayList;
4  import java.util.Random;
5  import java.util.concurrent.ThreadLocalRandom;
6
7  /**
8   *
9   * @author adelalbshri
10  *
11  */
12 public class Consensus {
13
14   private static ArrayList<Block> globalBlockchain = new
        ArrayList<>();
15   private static ArrayList<Object[]> nodesLog = new ArrayList
        <>();
```

```
16  static Random rand = new Random();
17
18  /**
19   *
20   * @param ConcensusAlgorithm
21   */
22  public static void consensus(String ConcensusAlgorithm)
23  {
24
25    if (ConcensusAlgorithm.equals("raft")) {
26      statusNodeLog();
27    } else {
28      AssignPoWMiner();
29    }
30  }
31
32  /**
33   *
34   * @return
35   */
36  public static ArrayList<Block> getGlobalBlockchain()
37  {
38    return globalBlockchain;
39  }
40
41  /**
42   *
43   * @return
44   */
45  public static ArrayList<Object[]> getNodesLog() { return
      nodesLog; }
```

```
46
47    /**
48     * Choice random node to become candidate
49     *
50     * @return int node index
51     */
52    private static void becomeCandidateNode()
53    {
54      int countCandidate = 0;
55      int i = 0;
56      while (countCandidate < InputConfig.getNumberOfMiner()) {
57        int NodeID = rand.nextInt(Node.getNodes().size());
58        if (!Node.getNodes().get(NodeID).getNodeType().equals("
              candidate")
59            && !Node.getNodes().get(NodeID).getNodeType().equals("
                leader")) {
60          Node.getNodes().get(NodeID).setNodeType("candidate");
61          nodesLog.add(new Object[] { "become Candidate",
62              Node.getNodes().get(NodeID).getNodeId(),
63              Node.getNodes().get(NodeID).getNodeType(),
64              Node.getNodes().get(NodeID).getJoinTime() });
65          votingFor(Node.getNodes().get(NodeID));
66          countCandidate += 1;
67        }
68      }
69    }
70
71    /**
72     * this to method to vote to candidate node
73     *
74     * @param nodeID
```

```
75      *
76      */
77     public static void votingFor(Node candidate)
78     {
79       boolean nodeVote;
80       int countNodeLeader = 0;
81       int countVotingCandidate = 0;
82       int round = 0;
83       for (round = 0; round < Node.getNodes().size(); round++) {
84         for (int i = 0; i < Node.getNodes().size(); i++) {
85           // int select = rand.nextInt(randomVoting.length);
86           if (candidate.getNodeId() != Node.getNodes().get(i).
                 getNodeId()) {
87             nodeVote = rand.nextBoolean();
88             if (nodeVote == true) {
89               countVotingCandidate += 1;
90             }
91           }
92         }
93         if (countVotingCandidate >= Node.getNodes().size() / 2) {
94           candidate.setNodeType("leader");
95           round = Node.getNodes().size();
96         }
97         countVotingCandidate = 0;
98       }
99     }
100
101    /**
102     * this to method to return miner node
103     *
104     * @return miner Node
```

```
105     *
106     */
107    public static Node getAassignLeader()
108    {
109      Node miner = null;
110
111      if (InputConfig.getConsensusalgorithm().equals("raft")) {
112        for (Node node : Node.getNodes()) {
113          if (node.getNodeType().equals("leader")) {
114            miner = node;
115          }
116        }
117      } else {
118        for (Node node : Node.getNodes()) {
119          if (node.getNodeType().equals("miner")) {
120            miner = node;
121          }
122        }
123      }
124      return miner;
125    }
126
127    /**
128     * this to method to keep tracking node status logs
129     *
130     *
131     */
132    public static void statusNodeLog()
133    {
134
135      for (Node node : Node.getNodes()) {
```

```
136       nodesLog.add(new Object[] { "Initial Nodes", node.
              getNodeId(),
137           node.getNodeType(), node.getJoinTime() });
138     }
139
140     becomeCandidateNode();
141
142     for (Node node : Node.getNodes()) {
143       nodesLog.add(new Object[] { "become leader", node.
              getNodeId(),
144           node.getNodeType(), node.getJoinTime() });
145     }
146   }
147
148   public static void AssignPoWMiner()
149   {
150     int countMiner = 0;
151     int i = 0;
152     for (i = 0; i < Node.getNodes().size(); i++) {
153       int NodeID = rand.nextInt(Node.getNodes().size());
154       if (!Node.getNodes().get(NodeID).getNodeType().equals("
              miner")
155           && countMiner < InputConfig.getNumberOfMiner()) {
156         Node.getNodes().get(NodeID).setHashPower(
157             70 /* rand.nextInt((100-10) + 10) */);
158         Node.getNodes().get(NodeID).setNodeType("miner");
159         countMiner += 1;
160       }
161       if (countMiner < InputConfig.getNumberOfMiner()) {
162         i = 0;
163       }
```

```
164     }
165   }
166
167   public static double protocalPoW(Node miner)
168   {
169     double totalHash = 0;
170
171     for (Node node : Node.getNodes()) {
172       if (node.getNodeType().equals("miner")) {
173         totalHash += node.getHashPower();
174       }
175     }
176
177     double hash = miner.getHashPower() / totalHash;
178
179     return ThreadLocalRandom.current().nextDouble(
180         hash, InputConfig.getBlockInterval());
181   }
182
183   public static void fork()
184   {
185
186     ArrayList<Integer> a = new ArrayList<>();
187     ArrayList<Integer> b = new ArrayList<>();
188     ArrayList<Node> c = new ArrayList<>();
189     int Z = 0;
190     int Max = 0;
191
192     for (Node node : Node.getNodes()) {
193       if (node.getNodeType().equals("miner")) {
194         a.add(node.getBlockchainLedger().size());
```

```
195       }
196     }
197
198     for (int i = 1; i < a.size(); i++) {
199       if (a.get(i) > Max) {
200         Max = a.get(i);
201       }
202     }
203
204     for (Node node : Node.getNodes()) {
205       if (node.getNodeType().equals("miner")) {
206         if (node.getBlockchainLedger().size() == Max) {
207           b.add(node.getNodeId());
208           Z = node.getNodeId();
209         }
210       }
211     }
212     for (Node node : Node.getNodes()) {
213       if (node.getNodeType().equals("miner")) {
214         if (node.getBlockchainLedger().size() == Max
215             && node.getLastBlock().getMiner().getNodeId() == Z)
                {
216           for (Block block : node.getBlockchainLedger()) {
217             globalBlockchain.add(block);
218           }
219         }
220       }
221     }
222   }
223
224   /**
```

```
225     * The time it takes the miner to generate next block
226     *
227     * @return double
228     */
229
230   public static double protocal()
231   {
232     return ThreadLocalRandom.current().nextDouble(
233         0, InputConfig.getBlockInterval());
234   }
235 }
```

A.2.4.5 Transaction Class

The transaction class models the characteristics and behaviours of transactions within the blockchain network in the IoTSim-Osmosis framework.

```
1  package IoTSimOsmosis.blockchainNetwork;
2
3  import java.util.ArrayList;
4  import java.util.Random;
5  import java.util.concurrent.ThreadLocalRandom;
6
7  /**
8   *
9   * @author adelalbshri
10  *
11  */
12 public class Transaction {
13   static Random rand = new Random();
14   // transaction id
15   private long transactionID = 0;
```

```
16    // timestamp of transaction send from the IoT side
17    private double creationTime;
18    // timestamp of transaction inclusion in a confirmed block
19    private double confirmationTime;
20    // transaction sender address
21    private int fromAddress;
22    // transaction receiver address
23    private int toAddress;
24    // transaction size
25    private double transactionSize;
26    // the amount of gas used by the transaction
27    private double usedGas;
28    // the maximum amount of gas units the transaction can use
29    private double transactionGasLimit;
30    // A variable to calculate (count) the remaining limit of
          Block gas used
31    static double blockGaslimit = 0;
32    // A variable to calculate (count) the remaining limit of
          Block size used
33    static double blockSizelimit = 0;
34
35    /**
36     * A constructor method for transaction class
37     *
38     * @param creationTime
39     * @param txSize
40     * @param gasLimit
41     * @param usedGas
42     * @param fromAddress
43     * @param toAddress
44     */
```

```
45    public Transaction(double creationTime)
46    {
47      this.transactionID = ThreadLocalRandom.current().nextLong
          (10000000000L);
48      this.creationTime = creationTime;
49      this.transactionGasLimit
50          = InputConfig.getTransactionGaslimit(); // 100;//50;
              //8000000;
51      this.usedGas = getRandomNumber(0, (int)transactionGasLimit);
52      this.transactionSize = InputConfig.getMinTransactionSize()
53          + rand.nextDouble()
54              * (InputConfig.getMaxTransactionSize()
55                  - InputConfig.getMinTransactionSize());
56      ;
57    }
58
59    /**
60     * Return the transaction ID
61     *
62     * @return transactionID
63     */
64    public long getTransactionID() { return transactionID; }
65
66    /**
67     * Return creation time for each transaction
68     *
69     * @return creationTime
70     */
71    public double getCreationTime() { return creationTime; }
72
73    // check if we need it
```

```
74  public static int getRandomNumber(int min, int max)
75  {
76    return (int)((Math.random() * (max - min)) + min);
77  }
78
79  /**
80   * Return transaction size for each transaction
81   *
82   * @return transactionSize
83   */
84  public double getTransactionSize() { return transactionSize; }
85
86  /**
87   * Return transaction confirmation Time when adding to a block
88   *
89   * @return transactionSize
90   */
91  public double getConfirmationTime() { return confirmationTime;
        }
92
93  /**
94   * Return the amount of gas used by the transaction after its
        execution on the
95   * EVM
96   *
97   * @return usedGas
98   */
99  public double getUsedGas() { return usedGas; }
100
101 /**
```

```
102     * Return the maximum amount of gas units the transaction can
            use.
103     *
104     * @return gasLimit
105     */
106    public double getTransactionGasLimit() { return
           transactionGasLimit; }
107
108    public int getFromAddress() { return fromAddress; }
109
110    public int getToAddress() { return toAddress; }
111
112    /**
113     * To set creation time for each transaction
114     *
115     * @param creationTime
116     */
117    public void setCreationTime(double creationTime)
118    {
119      this.creationTime = creationTime;
120    }
121
122    /**
123     * To set the amount of gas units that can use.
124     *
125     * @param usedGas
126     */
127    public void setUsedGas(double usedGas) { this.usedGas =
           usedGas; }
128
129    /**
```

```
130     * To set confirmation Time of transaction.
131     *
132     * @param confirmationTime
133     */
134    public void setConfirmationTime(double confirmationTime)
135    {
136      this.confirmationTime = confirmationTime;
137    }
138
139    /*
140     * Remaining limit of Block gas used
141     */
142
143    public static double getLimit() { return blockGaslimit; }
144
145    /*
146     * Remaining limit of Block gas used
147     */
148
149    public static double getBlockSizeLimit() { return
         blockSizelimit; }
150
151    /**
152     * 1- blockGaslimit subtract transaction used gas. 2- Block
          size subtract
153     * transaction size
154     *
155     * @param miner
156     * @param eventTime
157     * @return
158     */
```

```
159  public static ArrayList<Transaction> executeTranscationsPoW(
160      Node miner, Block block, double eventTime)
161  {
162    ArrayList<Transaction> transactions = new ArrayList<>();
163
164    double blockGas = InputConfig.getBlockGasLimit();
165    blockGaslimit = 0;
166    int count = 0;
167
168    miner.getTransactionsPool().sort(
169        (t1, t2) -> Double.compare(t2.getUsedGas(), t1.
           getUsedGas()));
170
171    while (count < miner.getTransactionsPool().size()) {
172
173      if (blockGas >= miner.getTransactionsPool().get(count).
          getUsedGas()
174          && miner.getTransactionsPool().get(count).
             getCreationTime()
175              <= eventTime) {
176        blockGas -= miner.getTransactionsPool().get(count).
           getUsedGas();
177        transactions.add(miner.getTransactionsPool().get(count))
           ;
178        miner.getTransactionsPool().get(count).
           setConfirmationTime(eventTime);
179        blockGaslimit += miner.getTransactionsPool().get(count).
           getUsedGas();
180      }
181      count += 1;
182    }
```

```
183
184     return transactions;
185   }
186
187   public static ArrayList<Transaction> executeTranscationsRaft(
188       Node miner, Block block, double eventTime)
189   {
190     ArrayList<Transaction> transactions = new ArrayList<>();
191
192     double blockSize = InputConfig.getMaxBlockSize();
193     blockSizelimit = 0;
194     int count = 0;
195
196     miner.getTransactionsPool().sort(
197         (t1, t2) -> Double.compare(t1.getCreationTime(), t2.
            getCreationTime()));
198
199     while (count < miner.getTransactionsPool().size()) {
200       if (blockSize
201               >= miner.getTransactionsPool().get(count).
                     getTransactionSize()
202           && miner.getTransactionsPool().get(count).
                 getCreationTime()
203               <= eventTime) {
204         blockSize
205             -= miner.getTransactionsPool().get(count).
                   getTransactionSize();
206         transactions.add(miner.getTransactionsPool().get(count))
              ;
207         miner.getTransactionsPool().get(count).
              setConfirmationTime(eventTime);
```

```
208             blockSizelimit
209                 += miner.getTransactionsPool().get(count).
                        getTransactionSize();
210           }
211           count += 1;
212         }
213
214         return transactions;
215       }
216     }
```

A.2.4.6 Event Class

The Event class encapsulates events within the blockchain simulation in the IoTSim-Osmosis framework to represent various actions and changes within the simulation.

```
1  package IoTSimOsmosis.blockchainNetwork;
2
3  import java.util.ArrayList;
4
5  /**
6   *
7   * @author adelalbshri
8   *
9   */
10
11 public class Event {
12
13   private String type;
14   private Node minerNode;
15   private double time;
16   private Block block;
```

```
17
18    public Event(String type, Node minerNode, double time, Block
          block)
19    {
20      super();
21      this.type = type;
22      this.minerNode = minerNode;
23      this.time = time;
24      this.block = block;
25    }
26
27    /**
28     * @return the block
29     */
30    public Block getBlock() { return block; }
31
32    /**
33     * @return the type
34     */
35    public String getType() { return type; }
36
37    /**
38     * @return the node
39     */
40    public Node getMiner() { return minerNode; }
41
42    /**
43     * @return the time
44     */
45    public double getTime() { return time; }
46  }
```

A.2.4.7 Queue Class

The Queue class is to manage the scheduling and processing of events within the blockchain-based IoT simulation by acting as an organised buffer for events awaiting execution.

```
1  package IoTSimOsmosis.blockchainNetwork;
2
3  import java.util.ArrayList;
4
5  public class Queue {
6
7    private static ArrayList<Event> eventList = new ArrayList<>();
8
9    public static void addEvent(Event e) { getEventList().add(e);
       }
10
11   public static void removeEvent(Event e)
12   {
13     getEventList().remove(getEventList().indexOf(e));
14   }
15
16   public static Event getNextEvent()
17   {
18     getEventList().sort((t1, t2) -> Double.compare(t1.getTime(),
         t2.getTime()));
19     return getEventList().get(0);
20   }
21
22   public static int size() { return getEventList().size(); }
23
24   public static boolean isEmpty() { return getEventList().size()
       == 0; }
```

```
25
26    public static ArrayList<Event> getEventList() { return
          eventList; }
27
28    public static void setEventList(ArrayList<Event> eventList)
29    {
30      Queue.eventList = eventList;
31    }
32  }
```

A.2.4.8 Scheduler Class

The Scheduler class orchestrates the timing and execution of events within the blockchain simulation by acting as a central controller for initiating actions such as block creation, transaction processing, and block propagation across the network.

```
1  package IoTSimOsmosis.blockchainNetwork;
2
3  import java.util.ArrayList;
4  import java.util.concurrent.ThreadLocalRandom;
5
6  /**
7   *
8   * @author adelalbshri
9   *
10  */
11 public class Scheduler {
12
13   static ArrayList<Block> countGenerateBlockByMiner = new
         ArrayList<>();
14
15   /**
```

```
16      *
17      * @param miner
18      * @param eventTime
19      */
20     public static void createBlockEvent(Node miner, double
           eventTime)
21     {
22       String eventType = "create_block";
23       if (eventTime <= InputConfig.getSimulationTime()) {
24         Block block = new Block();
25         block.setBlockID(ThreadLocalRandom.current().nextLong
               (100000000000L));
26         block.setBlockDepth(miner.getBlockchainLedger().size());
27         block.setBlockTimestamp(eventTime);
28         block.setMiner(miner);
29         block.setPreviousBlockID(miner.getLastBlock().getBlockID()
               );
30         Event event = new Event(
31             eventType, block.getMiner(), eventTime, block); //
                   change here
32         Queue.addEvent(event);
33       }
34     }
35
36     /**
37      *
38      * @param node
39      * @param newBlock
40      * @param blockDelay
41      */
```

```
42    public static void receiveBlockEvent(Node node, Block newBlock
          )
43    {
44      String eventType = "receive_block";
45      double receiveBlockTime = newBlock.getBlockTimestamp();
46      if (receiveBlockTime <= InputConfig.getSimulationTime()) {
47        updateTx(newBlock);
48        Event event = new Event(eventType, node, receiveBlockTime,
              newBlock);
49        Queue.addEvent(event);
50      }
51    }
52
53    public static void updateTx(Block block)
54    {
55
56      int count = 0;
57      for (count = 0; count < block.getTransactions().size();
           count++) {
58        for (Node node : Node.getNodes()) {
59          if ((node.getNodeType().equals("leader")
60                  || node.getNodeType().equals("miner"))
61              && node != block.getMiner()) {
62            for (int i = 0; i < node.getTransactionsPool().size();
                  i++) {
63              if (block.getTransactions().get(count)
64                  == node.getTransactionsPool().get(i)) {
65                node.getTransactionsPool().remove(i);
66              }
67            }
68          }
```

```
69          }
70      }
71    }
72  }
```

A.2.5 Reporter

The Simulation Core encompasses a collection of classes designed for analysis and reporting of simulation experiments. The following subsections present these classes and their operations.

A.2.5.1 Excel Class

The Excel class is designed to handle basic Excel operations, including creating workbooks, writing data to sheets, and formatting cells.

```
1  package IoTSimOsmosis.blockchainNetwork;
2
3  import com.google.common.collect.Table.Cell;
4  import java.awt.Desktop;
5  import java.io.File;
6  import java.io.FileNotFoundException;
7  import java.io.FileOutputStream;
8  import java.io.IOException;
9  import java.time.LocalDateTime;
10 import java.time.format.DateTimeFormatter;
11 import java.util.ArrayList;
12 import org.apache.poi.ss.usermodel.FillPatternType;
13 import org.apache.poi.ss.usermodel.IndexedColors;
14 import org.apache.poi.ss.usermodel.Row;
15 import org.apache.poi.xssf.usermodel.XSSFCell;
16 import org.apache.poi.xssf.usermodel.XSSFCellStyle;
```

```
17 import org.apache.poi.xssf.usermodel.XSSFFont;
18 import org.apache.poi.xssf.usermodel.XSSFRow;
19 import org.apache.poi.xssf.usermodel.XSSFSheet;
20 import org.apache.poi.xssf.usermodel.XSSFWorkbook;
21
22 public class Excel {
23   static ArrayList<Object[]> df3 = new ArrayList<>();
24
25   public static ArrayList<Object[]> getDf3() { return df3; }
26
27   public static void printToExcel(int simulationRunNumber)
28   {
29     XSSFWorkbook workbook = new XSSFWorkbook();
30
31     ArrayList<Object[]> df4 = new ArrayList<>();
32
33     df4.add(new Object[] {
34         "Run simulator",
35         "Node ID",
36         "Node Type",
37     });
38
39     for (Object[] chain : getDf3()) {
40       df4.add(chain);
41     }
42     writeData(df4, workbook, "tes");
43
44     String fname = "Statistics.xlsx";
45
46     try (FileOutputStream outputStream
47         = new FileOutputStream("output/Statistics.xls")) {
```

```
48        workbook.write(outputStream);
49      } catch (FileNotFoundException e) {
50        e.printStackTrace();
51      } catch (IOException e) {
52        e.printStackTrace();
53      }
54    }
55
56    private static void writeData(
57        ArrayList<Object[]> DataFrame, XSSFWorkbook workbook,
              String sheetName)
58    {
59
60      XSSFSheet sheet = workbook.createSheet(sheetName);
61
62      int rowCount = 0;
63
64      for (Object[] rowData : DataFrame) {
65        XSSFRow row = sheet.createRow(++rowCount);
66
67        int columnCount = 0;
68        for (Object field : rowData) {
69          XSSFCell cell = row.createCell(++columnCount);
70          if (field instanceof String) {
71            cell.setCellValue((String)field);
72          } else if (field instanceof Integer) {
73            cell.setCellValue((Integer)field);
74          } else if (field instanceof Double) {
75            cell.setCellValue((Double)field);
76          } else if (field instanceof Long) {
77            cell.setCellValue((Long)field);
```

```
78          }
79        }
80      }
81
82      formatExcelSheet(DataFrame.get(0).length, sheet, workbook);
83    }
84
85    private static void formatExcelSheet(
86        int columns, XSSFSheet sheet, XSSFWorkbook workbook)
87    {
88
89      XSSFFont font = workbook.createFont();
90      font.setFontHeightInPoints((short)11);
91      font.setBold(true);
92      font.setColor(IndexedColors.WHITE.getIndex());
93      XSSFCellStyle style = workbook.createCellStyle();
94      style.setFont(font);
95      style.setFillBackgroundColor(IndexedColors.BLACK.getIndex())
          ;
96      style.setFillPattern(FillPatternType.SOLID_FOREGROUND);
97      XSSFRow header = sheet.getRow((short)1);
98      for (int i = 1; i < header.getLastCellNum(); i++) {
99        header.getCell(i).setCellStyle(style);
100     }
101
102     for (int i = 1; i < columns + 1; i++) {
103       sheet.autoSizeColumn(i);
104     }
105   }
106 }
```

A.2.5.2 ExcelWriter Class

The ExcelWriter class expands on the functionality of the Excel class A.2.5.1 by offering more specialized and methods for reporting simulation results.

```
1  package IoTSimOsmosis.blockchainNetwork;
2
3  import java.awt.Desktop;
4  import java.io.File;
5  import java.io.FileNotFoundException;
6  import java.io.FileOutputStream;
7  import java.io.IOException;
8  import java.time.LocalDateTime;
9  import java.time.format.DateTimeFormatter;
10 import java.util.ArrayList;
11 import org.apache.poi.ss.usermodel.FillPatternType;
12 import org.apache.poi.ss.usermodel.IndexedColors;
13 import org.apache.poi.xssf.usermodel.XSSFCell;
14 import org.apache.poi.xssf.usermodel.XSSFCellStyle;
15 import org.apache.poi.xssf.usermodel.XSSFFont;
16 import org.apache.poi.xssf.usermodel.XSSFRow;
17 import org.apache.poi.xssf.usermodel.XSSFSheet;
18 import org.apache.poi.xssf.usermodel.XSSFWorkbook;
19
20 public class ExcelWriter {
21   static XSSFWorkbook workbook = new XSSFWorkbook();
22   public static int runNumber;
23
24   public static void printToExcel(int simulationRunNumber)
25   {
26     runNumber = simulationRunNumber;
27     if (InputConfig.getConsensusalgorithm() == "PoW") {
```

```
28          configPoW();
29          resultPoW();
30          blockchainLedgerPoW();
31          blockchainTranscations();
32          transcationPool();
33          transcationLatency();
34          statisticResult();
35      } else if (InputConfig.getConsensusalgorithm() == "raft") {
36          configRaft();
37          resultRaft();
38          blockchainLedgerRaft();
39          blockchainTranscations();
40          transcationLatency();
41          transcationPool();
42          statisticResult();
43          nodeLog();
44      }
45
46      String fname = "Blockchain-" + (simulationRunNumber) + ".
            xlsx";
47
48      try (FileOutputStream outputStream
49              = new FileOutputStream("output/" + fname)) {
50          workbook.write(outputStream);
51          outputStream.close();
52
53          Desktop.getDesktop().open(new File("output/" + fname));
54
55      } catch (FileNotFoundException e) {
56          e.printStackTrace();
57      } catch (IOException e) {
```

```
58        e.printStackTrace();
59      }
60    }
61
62    /**
63     * to print simulator configuration for PoW
64     */
65
66    public static void configPoW()
67    {
68      ArrayList<Object[]> df1 = new ArrayList<>();
69
70      df1.add(new Object[] { "Simulator No. Run", "No. of Node", "
          No. of Miner",
71          "consensus Algorithm", "No. of Transactions", "Block Gas
              Limit",
72          "Transaction Gas Limit", "Block Interval", "Simulation
             Time" });
73
74      df1.add(new Object[] {
75          runNumber,
76          Node.getNodes().size(),
77          InputConfig.getNumberOfMiner(),
78          InputConfig.getConsensusalgorithm(),
79          InputConfig.getTransactionNumber(),
80          InputConfig.getBlockGasLimit(),
81          InputConfig.getTransactionGaslimit(),
82          InputConfig.getBlockInterval(),
83          InputConfig.getSimulationTime(),
84      });
85
```

```
86      // writing data frames to workbook
87      writeData(df1, workbook, "config");
88    }
89
90    /**
91     * to print simulator configuration for Raft
92     */
93
94    public static void configRaft()
95    {
96      ArrayList<Object[]> configRaft = new ArrayList<>();
97
98      configRaft.add(new Object[] {
99          "Simulator No. Run",
100         "No. of Node",
101         "No. of Miner",
102         "consensus Algorithm",
103         "Total No of Transactions Per Sec",
104         "Max Block Size",
105         "Max Tx Size",
106         "Min Tx Size",
107         "Block Interval",
108         "Simulation Time",
109     });
110
111     configRaft.add(new Object[] {
112         runNumber,
113         Node.getNodes().size(),
114         InputConfig.getNumberOfMiner(),
115         InputConfig.getConsensusalgorithm(),
116         InputConfig.getTransactionNumber(),
```

```
117             InputConfig.getMaxBlockSize(),
118             InputConfig.getMaxTransactionSize(),
119             InputConfig.getMinTransactionSize(),
120             InputConfig.getBlockInterval(),
121             InputConfig.getSimulationTime(),
122         });
123 
124         // writing data frames to workbook
125         writeData(configRaft, workbook, "config");
126     }
127 
128     /**
129      * to print simulator result PoW
130      */
131 
132     public static void resultPoW()
133     {
134         ArrayList<Object[]> df2 = new ArrayList<>();
135 
136         df2.add(new Object[] {
137                 "Simulator No. Run",
138                 "Total No. of Blocks",
139                 "Total No. of Blocks include Tx",
140                 "Total No. of  Blocks without Tx",
141                 "Total No of Transactions Per Sec",
142                 "Avg. No. of Tx per block",
143                 "Avg. of Tx Inclusion Time (secs)",
144                 "Avg. Tx Used Gas",
145                 "Total No. of Pending Tx",
146                 "Avg. Block Propagation (secs)",
147                 "Avg. Transaction Latency (secs)",
```

```
148             "Transactions execution (secs)",
149             "Transaction Throughput (Tx/secs)",
150         });
151 
152         df2.add(new Object[] {
153             runNumber,
154             Statistics.totalNumberOfBlock,
155             Statistics.blockIncludeTx,
156             Statistics.blockWithoutTx,
157             Statistics.totalNumberOfTx,
158             Statistics.TxPerBlock,
159             Statistics.TxInclusionTime,
160             Statistics.TxUsedGas,
161             Statistics.pendingTx,
162             Statistics.blockPropagationTime,
163             Statistics.averageLatency,
164             Statistics.totalTransactionsTime,
165             Statistics.transactionsThroughput,
166         });
167 
168         // writing data frames to workbook
169         writeData(df2, workbook, "Results");
170     }
171 
172     /**
173      * to print simulator result for Raft
174      */
175 
176     public static void resultRaft()
177     {
178         ArrayList<Object[]> df2 = new ArrayList<>();
```

```
179
180     df2.add(new Object[] {
181         "Simulator No. Run",
182         "Total No. of Blocks",
183         "Total No. of Blocks include Tx",
184         "Total No. of  Blocks without Tx",
185         "Avg. Block Size (MB)",
186         "Total No of Transactions",
187         "Avg. No. of Tx per block",
188         "Avg. of Tx Inclusion Time (secs)",
189         "Avg. Tx Size (MB)",
190         "Total No. of Pending Tx",
191         "Avg. Block Propagation (secs)",
192         "Avg. Transaction Latency (secs)",
193         "Transactions execution (secs)",
194         "Transaction Throughput (Tx/secs)",
195     });
196
197     df2.add(new Object[] {
198         runNumber,
199         Statistics.totalNumberOfBlock,
200         Statistics.blockIncludeTx,
201         Statistics.blockWithoutTx,
202         Statistics.blockSize,
203         Statistics.totalNumberOfTx,
204         Statistics.TxPerBlock,
205         Statistics.TxInclusionTime,
206         Statistics.TxSize,
207         Statistics.pendingTx,
208         Statistics.blockPropagationTime,
209         Statistics.averageLatency,
```

```
210         Statistics.totalTransactionsTime,
211         Statistics.transactionsThroughput,
212       });
213
214       // writing data frames to workbook
215       writeData(df2, workbook, "Results");
216     }
217
218     /**
219      * to print blockchain blocks PoW
220      */
221
222     public static void blockchainLedgerPoW()
223     {
224
225       ArrayList<Object[]> df3 = new ArrayList<>();
226
227       df3.add(new Object[] { "Simulator No. Run", "Block ID", "
          Previous Block ID",
228           "Block Depth", "Block Timestamp", "Block Used Gas",
229           "No. of Transactions", "Mined by", "hash power" });
230
231       for (Object[] chain : Statistics.getChains()) {
232         df3.add(chain);
233       }
234
235       writeData(df3, workbook, "block");
236     }
237
238     /**
239      * to print blockchain blocks Raft
```

```
240     */
241
242   public static void blockchainLedgerRaft()
243   {
244
245     ArrayList<Object[]> df3 = new ArrayList<>();
246
247     df3.add(new Object[] { "Simulator No. Run", "Block ID", "
          Previous Block ID",
248         "Block Depth", "Block Timestamp", "Block Size", "No. of
            Transactions",
249         "Mined by" });
250
251     for (Object[] chain : Statistics.getChains()) {
252       df3.add(chain);
253     }
254
255     writeData(df3, workbook, "block");
256   }
257
258   /**
259    * to print global blockchain
260    */
261
262   public static void globalBlockchain()
263   {
264
265     ArrayList<Object[]> df4 = new ArrayList<>();
266
267     df4.add(new Object[] { "Simulator No. Run", "Block ID", "
          Previous Block ID",
```

```
268         "Block Depth", "Block Timestamp", "Block Received Time",
                "Block Size",
269         "No. of Transactions" });
270
271     for (Object[] globalBlockchain : Statistics.
            getGlobalBlockchain()) {
272       df4.add(globalBlockchain);
273     }
274
275     writeData(df4, workbook, "globalBlockchain");
276   }
277
278   /**
279    * to print transcations
280    */
281
282   public static void blockchainTranscations()
283   {
284
285     ArrayList<Object[]> df5 = new ArrayList<>();
286     if (InputConfig.getConsensusalgorithm() == "PoW") {
287       df5.add(new Object[] { "Simulator No. Run", "Transaction
              ID",
288           "Creation time ", "Confirmation time", "Transaction
                  size",
289           "Transaction Used Gas", "Block ID" });
290       for (Object[] transaction : Statistics.getTransactions())
              {
291         df5.add(transaction);
292       }
293     } else if (InputConfig.getConsensusalgorithm() == "raft") {
```

```
294         df5.add(new Object[] { "Simulator No. Run", "Transaction
                ID",
295             "Creation time ", "Confirmation time", "Transaction
                    size",
296             "Block ID" });
297         for (Object[] transaction : Statistics.getTransactions())
                {
298           df5.add(transaction);
299         }
300       }
301
302       writeData(df5, workbook, "Transcations");
303     }
304
305     /**
306      * to print transcations latency
307      */
308
309     public static void transcationLatency()
310     {
311
312       ArrayList<Object[]> df6 = new ArrayList<>();
313       df6.add(new Object[] { "Simulator No. Run", "Transaction ID"
              ,
314           "Creation time ", "Confirmation time", "Transaction
                  Latency" });
315
316       for (Object[] transactionLatency : Statistics.
              getTransactionLatencies()) {
317         df6.add(transactionLatency);
318       }
```

```
319
320     writeData(df6, workbook, "TransactionLatency");
321   }
322
323   /**
324    * to print transcations pool
325    */
326
327   public static void transcationPool()
328   {
329     ArrayList<Object[]> df7 = new ArrayList<>();
330
331     if (InputConfig.getConsensusalgorithm() == "PoW") {
332       df7.add(new Object[] { "Simulator No. Run", "Transaction
          ID",
333           "Creation time ", "Tx Used Gas ", "Status" });
334       for (Object[] transactionLatency : Statistics.
          getTransactionsPool()) {
335         df7.add(transactionLatency);
336       }
337     } else if (InputConfig.getConsensusalgorithm() == "raft") {
338       df7.add(new Object[] { "Simulator No. Run", "Transaction
          ID",
339           "Creation time ", "Transaction Size", "Status" });
340       for (Object[] transactionLatency : Statistics.
          getTransactionsPool()) {
341         df7.add(transactionLatency);
342       }
343     }
344
345     writeData(df7, workbook, "TransactionPool");
```

```
346   }
347
348   /**
349    * to print nodes log for raft
350    */
351   public static void nodeLog()
352   {
353     ArrayList<Object[]> df8 = new ArrayList<>();
354
355     df8.add(new Object[] { "Stage", "Node ID", "Node Type", "
          Joining Time" });
356
357     for (Object[] NodesLog : Consensus.getNodesLog()) {
358       df8.add(NodesLog);
359     }
360
361     writeData(df8, workbook, "NodesLog");
362   }
363
364   /**
365    * to print statistic Result
366    */
367
368   public static void statisticResult()
369   {
370     ArrayList<Object[]> df9 = new ArrayList<>();
371
372     df9.add(new Object[] {
373         "Item", "Minimum", "Maximum", "Mean", "Standard
            Deviation" });
374
```

```
375     df9.add(new Object[] { "Block Time", Statistics.minBlockTime
            ,
376         Statistics.maxBlockTime, Statistics.meanBlockTime,
377         Statistics.SDBlockTime });
378
379     writeData(df9, workbook, "statistic");
380   }
381
382   /**
383    * Writes each Array within Data frame as a Row in the excel
           sheet.
384    *
385    * @param DataFrame
386    * @param workbook
387    * @param sheetName
388    */
389   private static void writeData(
390       ArrayList<Object[]> DataFrame, XSSFWorkbook workbook,
              String sheetName)
391   {
392
393     XSSFSheet sheet = workbook.getSheet(sheetName);
394     if (sheet == null) {
395       sheet = workbook.createSheet(sheetName);
396     }
397
398     int rowCount = 0;
399
400     for (Object[] rowData : DataFrame) {
401       XSSFRow row = sheet.createRow(++rowCount);
402
```

```
403       int columnCount = 0;
404       for (Object field : rowData) {
405         XSSFCell cell = row.createCell(++columnCount);
406         if (field instanceof String) {
407           cell.setCellValue((String)field);
408         } else if (field instanceof Integer) {
409           cell.setCellValue((Integer)field);
410         } else if (field instanceof Double) {
411           cell.setCellValue((Double)field);
412         } else if (field instanceof Long) {
413           cell.setCellValue((Long)field);
414         }
415       }
416     }
417
418     // Basic aesthetic formating of Excel Sheet
419     formatExcelSheet(DataFrame.get(0).length, sheet, workbook);
420   }
421
422   /**
423    * Aesthetic formating of excel sheet
424    *
425    * @param columns
426    * @param sheet
427    * @param workbook
428    */
429   private static void formatExcelSheet(
430       int columns, XSSFSheet sheet, XSSFWorkbook workbook)
431   {
432
433     // Creating header font.
```

```
434         XSSFFont font = workbook.createFont();
435         font.setFontHeightInPoints((short)11);
436         font.setBold(true);
437         font.setColor(IndexedColors.WHITE.getIndex());
438 
439         // Setting header font and header filling.
440         XSSFCellStyle style = workbook.createCellStyle();
441         style.setFont(font);
442         style.setFillBackgroundColor(IndexedColors.BLACK.getIndex())
                ;
443         style.setFillPattern(FillPatternType.SOLID_FOREGROUND);
444 
445         XSSFRow header = sheet.getRow((short)1);
446         // Setting style for each cell in the header row.
447         for (int i = 1; i < header.getLastCellNum(); i++) {
448           header.getCell(i).setCellStyle(style);
449         }
450 
451         // Resize column widths
452         for (int i = 1; i < columns + 1; i++) {
453           sheet.autoSizeColumn(i);
454         }
455       }
456 }
```

A.2.5.3 Statistics Class

The Statistics class is designed for collecting, calculating, and reporting statistical data from blockchain-based IoT simulation.

```
1 package IoTSimOsmosis.blockchainNetwork;
2 
```

```
3  import java.util.ArrayList;
4  import java.util.HashMap;
5  import java.util.Iterator;
6
7  public class Statistics {
8
9    private static ArrayList<Object[]> chains = new ArrayList<>();
10   private static ArrayList<Object[]> globalBlockchain = new
         ArrayList<>();
11   private static ArrayList<Object[]> transactions = new
         ArrayList<>();
12   private static ArrayList<Object[]> transactionLatencies = new
         ArrayList<>();
13   private static ArrayList<Object[]> transactionsPool = new
         ArrayList<>();
14   private static ArrayList<Object[]> Result = new ArrayList<>();
15   //
16   public static int totalNumberOfBlock = 0;
17   public static int totalNumberOfTx = 0;
18   public static double blockPropagationTime = 0;
19   public static double averageLatency = 0;
20   public static double transactionsThroughput = 0;
21   public static double firstCreationTime = 0;
22   public static double lastConfirmiationTime = 0;
23   public static double totalTransactionsTime = 0;
24   public static double blockIncludeTx = 0;
25   public static double blockWithoutTx = 0;
26   public static double TxPerBlock = 0;
27   public static double TxInclusionTime = 0;
28   public static double TxUsedGas = 0;
29   public static double TxSize = 0;
```

```
30    public static double pendingTx = 0;
31    public static double blockSize = 0;
32    public static int runNumber = 0;
33    public static int blockTim = 0;
34    public static double SDBlockTime = 0;
35    public static double minBlockTime = 0;
36    public static double maxBlockTime = 0;
37    public static double meanBlockTime = 0;
38    public static double medBlockTime = 0;
39    public static double numberRun;
40
41    public static void calculate(int simulationRunNumber)
42    {
43      numberRun = simulationRunNumber;
44      blockchainLedger();
45      globalBlockchain();
46      transaction();
47      transactionLatency();
48      calculateLatency();
49
50      transactionsPool();
51      statisticResultBlockTime();
52      overallResults();
53      ExcelWriter.printToExcel(simulationRunNumber);
54      rest();
55    }
56
57    public static void calculateLatency()
58    {
59
60      Node miner = null;
```

```
61      for (Node m : Node.getNodes()) {
62        if (m.getNodeType().equals("leader") || m.getNodeType().
             equals("miner")) {
63          miner = m;
64        }
65      }
66      int blockchainSize = miner.getBlockchainLedger().size();
67      int transactionListSize = miner.getBlockchainLedger()
68                                     .get(blockchainSize - 1)
69                                     .getTransactions()
70                                     .size();
71      if (blockchainSize > 0) {
72        if (miner.getBlockchainLedger().get(0).getTransactions().
             size() > 0) {
73          firstCreationTime = miner.getBlockchainLedger()
74                                   .get(1)
75                                   .getTransactions()
76                                   .get(0)
77                                   .getCreationTime();
78        }
79        if (miner.getBlockchainLedger()
80                .get(blockchainSize - 1)
81                .getTransactions()
82                .size()
83             > 0) {
84          lastConfirmiationTime = miner.getBlockchainLedger()
85                                       .get(blockchainSize - 1)
86                                       .getTransactions()
87                                       .get(transactionListSize -
                                          1)
88                                       .getConfirmationTime();
```

```
89        }
90        totalTransactionsTime = lastConfirmiationTime -
              firstCreationTime;
91      }
92
93      if (totalTransactionsTime > 0) {
94        transactionsThroughput = totalNumberOfTx /
              totalTransactionsTime;
95      }
96    }
97
98    private static void blockchainLedger()
99    {
100
101     Node miner = null;
102     for (Node node : Node.getNodes()) {
103       if (node.getNodeType().equals("leader")) {
104         miner = node;
105       }
106     }
107
108     Iterator<Block> iterator = miner.getBlockchainLedger().
          iterator();
109     while (iterator.hasNext()) {
110
111       Block b = iterator.next();
112       if (InputConfig.getConsensusalgorithm() == "PoW") {
113         if (b.getBlockID() == 0) {
114           totalNumberOfBlock += 1;
115           Object[] info = { numberRun, b.getBlockID(), b.
                getPreviousBlockID(),
```

```
116             b.getBlockDepth(), b.getBlockTimestamp(), 0, 0, "
                    Null", 0 };
117           getChains().add(info);
118         } else {
119           totalNumberOfBlock += 1;
120
121           Object[] info = { numberRun, b.getBlockID(), b.
                  getPreviousBlockID(),
122             b.getBlockDepth(), b.getBlockTimestamp(), b.
                    getBlockGas(),
123             b.getTransactions().size(), b.getMiner().getNodeId()
                    ,
124             b.getMiner().getHashPower() };
125           getChains().add(info);
126         }
127       } else if (InputConfig.getConsensusalgorithm() == "raft")
             {
128         if (b.getBlockID() == 0) {
129           totalNumberOfBlock += 1;
130           Object[] info = { numberRun, b.getBlockID(), b.
                  getPreviousBlockID(),
131             b.getBlockDepth(), b.getBlockTimestamp(), 0, 0, "
                    Null" };
132           getChains().add(info);
133         } else {
134
135           totalNumberOfBlock += 1;
136           Object[] info = { numberRun, b.getBlockID(), b.
                  getPreviousBlockID(),
137             b.getBlockDepth(), b.getBlockTimestamp(), b.
                    getBlockSize(),
```

```
138             b.getTransactions().size(), b.getMiner().getNodeId()
                    };
139           getChains().add(info);
140         }
141       }
142     }
143     blockPropagationTime = miner.getBlockchainLedger()
144                                 .get(totalNumberOfBlock - 1)
145                                 .getBlockTimestamp()
146         / totalNumberOfBlock;
147   }
148
149   private static void globalBlockchain()
150   {
151
152     Node miner = Consensus.getAassignLeader();
153     Iterator<Block> iterator = miner.getBlockchainLedger().
          iterator();
154     while (iterator.hasNext()) {
155
156       Block b = iterator.next();
157
158       if (b.getBlockID() == 0) {
159         Object[] info = { numberRun, b.getBlockID(), b.
              getPreviousBlockID(),
160           b.getBlockDepth(), b.getBlockTimestamp(), b.
                getBlockSize(),
161           b.getTransactions().size(), 0, 0 };
162         getGlobalBlockchain().add(info);
163       } else {
```

```
164         Object[] info = { numberRun, b.getBlockID(), b.
                getPreviousBlockID(),
165           b.getBlockDepth(), b.getBlockTimestamp(), b.
                getBlockSize(),
166           b.getTransactions().size(), b.getMiner().getNodeId(),
167           b.getMiner().getHashPower() };
168         getGlobalBlockchain().add(info);
169       }
170     }
171   }
172
173   private static void transaction()
174   {
175
176     Node miner = Consensus.getAassignLeader();
177
178     for (Block b : miner.getBlockchainLedger()) {
179       for (Transaction transaction : b.getTransactions()) {
180         if (InputConfig.getConsensusalgorithm() == "PoW") {
181           Object[] info = { numberRun, transaction.
                getTransactionID(),
182             transaction.getCreationTime(), transaction.
                  getConfirmationTime(),
183             transaction.getTransactionSize(), transaction.
                  getUsedGas(),
184             b.getBlockID() };
185           getTransactions().add(info);
186         } else if (InputConfig.getConsensusalgorithm() == "raft"
              ) {
187           Object[] info = { numberRun, transaction.
                getTransactionID(),
```

```
188             transaction.getCreationTime(), transaction.
                    getConfirmationTime(),
189             transaction.getTransactionSize(), b.getBlockID() };
190           getTransactions().add(info);
191         }
192       }
193     }
194
195     totalNumberOfTx = getTransactions().size();
196   }
197
198   private static void transactionLatency()
199   {
200     double TransactionLatency;
201     double totalTxLatency = 0;
202
203     Node Miner = Consensus.getAassignLeader();
204
205     for (Block b : Miner.getBlockchainLedger()) {
206       for (Transaction transaction : b.getTransactions()) {
207         Object[] info = { numberRun, transaction.
                getTransactionID(),
208           transaction.getCreationTime(), transaction.
                  getConfirmationTime(),
209           TransactionLatency
210           = transaction.getConfirmationTime() - transaction.
                  getCreationTime() };
211         getTransactionLatencies().add(info);
212         totalTxLatency += TransactionLatency;
213       }
214     }
```

```
215
216       averageLatency = totalTxLatency / totalNumberOfTx;
217     }
218
219     private static void overallResults()
220     {
221       Node Miner = Consensus.getAassignLeader();
222       for (Block b : Miner.getBlockchainLedger()) {
223         blockSize += b.getBlockSize();
224
225         if (b.getHasTx() == false) {
226           blockWithoutTx += 1;
227
228         } else if (b.getHasTx() == true) {
229           blockIncludeTx += 1;
230           TxPerBlock += b.getTransactions().size();
231         }
232
233         for (Transaction t : b.getTransactions()) {
234           TxInclusionTime += t.getConfirmationTime();
235           TxUsedGas += t.getUsedGas();
236           TxSize += t.getTransactionSize();
237         }
238       }
239
240       blockSize = blockSize / totalNumberOfBlock;
241       TxPerBlock = TxPerBlock / totalNumberOfBlock;
242       TxInclusionTime = TxInclusionTime / totalNumberOfTx;
243       TxUsedGas = TxUsedGas / totalNumberOfTx;
244       TxSize = TxSize / totalNumberOfTx;
245     }
```

```
246
247   private static void transactionsPool()
248   {
249     Node node = Consensus.getAassignLeader();
250     if (InputConfig.getConsensusalgorithm() == "PoW") {
251       node.getTransactionsPool().sort(
252           (t1, t2) -> Double.compare(t2.getUsedGas(), t1.
                getUsedGas()));
253       for (Transaction transaction : node.getTransactionsPool())
              {
254         pendingTx += 1;
255         Object[] info = { numberRun, transaction.
              getTransactionID(),
256           transaction.getCreationTime(), transaction.getUsedGas
                (), "Pending" };
257         getTransactionsPool().add(info);
258       }
259     } else if (InputConfig.getConsensusalgorithm() == "raft") {
260       node.getTransactionsPool().sort(
261           (t1, t2)
262               -> Double.compare(t1.getCreationTime(), t2.
                    getCreationTime()));
263       for (Transaction transaction : node.getTransactionsPool())
              {
264         pendingTx += 1;
265         Object[] info = { numberRun, transaction.
              getTransactionID(),
266           transaction.getCreationTime(), transaction.
                getTransactionSize(),
267           "Pending" };
268         getTransactionsPool().add(info);
```

```
269       }
270     }
271   }
272
273   public static void statisticResultBlockTime()
274   {
275     double sumBlockTime = 0;
276
277     Node miner = Consensus.getAassignLeader();
278     for (int i = 0; i < miner.getBlockchainLedger().size(); i++)
          {
279       sumBlockTime += miner.getBlockchainLedger().get(i).
            getBlockTimestamp();
280       if (miner.getBlockchainLedger().get(i).getBlockTimestamp()
281           < minBlockTime) {
282         minBlockTime = miner.getBlockchainLedger().get(i).
              getBlockTimestamp();
283       }
284       if (miner.getBlockchainLedger().get(i).getBlockTimestamp()
285           > maxBlockTime) {
286         maxBlockTime = miner.getBlockchainLedger().get(i).
              getBlockTimestamp();
287       }
288     }
289     meanBlockTime = sumBlockTime / (totalNumberOfBlock - 1);
290
291     for (Block block : miner.getBlockchainLedger()) {
292       SDBlockTime += Math.pow(block.getBlockTimestamp() -
            meanBlockTime, 2);
293     }
```

```
294     SDBlockTime = Math.sqrt(SDBlockTime / (totalNumberOfBlock -
            1));
295   }
296
297   public static void rest()
298   {
299
300     totalNumberOfBlock = 0;
301     totalNumberOfTx = 0;
302     blockPropagationTime = 0;
303     averageLatency = 0;
304     transactionsThroughput = 0;
305     firstCreationTime = 0;
306     lastConfirmiationTime = 0;
307     totalTransactionsTime = 0;
308     blockIncludeTx = 0;
309     blockWithoutTx = 0;
310     TxPerBlock = 0;
311     TxInclusionTime = 0;
312     TxUsedGas = 0;
313     TxSize = 0;
314     pendingTx = 0;
315     blockSize = 0;
316     runNumber = 0;
317     blockTim = 0;
318     SDBlockTime = 0;
319     minBlockTime = 0;
320     maxBlockTime = 0;
321     meanBlockTime = 0;
322     medBlockTime = 0;
323     numberRun = 0;
```

```
324
325       getChains().clear();
326       getTransactions().clear();
327       getTransactionLatencies().clear();
328       getTransactionsPool().clear();
329       getResult().clear();
330       getGlobalBlockchain().clear();
331     }
332
333     public static ArrayList<Object[]> getChains() { return chains;
          }
334
335     public static ArrayList<Object[]> getTransactions() { return
        transactions; }
336
337     public static ArrayList<Object[]> getTransactionLatencies()
338     {
339       return transactionLatencies;
340     }
341
342     public static ArrayList<Object[]> getTransactionsPool()
343     {
344       return transactionsPool;
345     }
346
347     public static ArrayList<Object[]> getResult() { return Result;
          }
348
349     public static ArrayList<Object[]> getGlobalBlockchain()
350     {
351       return globalBlockchain;
```

```
352     }
353
354     public static int getRunNumber() { return runNumber; }
355 }
```

Appendix B

Simulator User Manual

B.1 Getting Started

B.1.1 Lifecycle of Blockchain-based IoT simulation

The overall architecture of Blockchain-based IoT simulation is divided into four main layers: Configurator, Generator, Simulation Core, and Reporter, as discussed in the appendix A. The simulation requires to configure both blockchain and IoTSim-Osmosis, as shown in Section B.2.

B.1.2 System and Software Requirements

Table B.1 System and Software Requirements

Operating System	Windows, Linux or Mac OS
CPU	1-GHz processor or equivalent (Minimum)
RAM	2GB (Minimum))
Java	JDK version 11+
IDE	Any IDE for Java programming language (e.g. Eclipse or NetBeans)

B.1.3 Download Blockchain-based IoT simulation

Blockchain-based IoT simulation can be downloaded directly from the GitHub repository [1], as shown in Figure B.1.

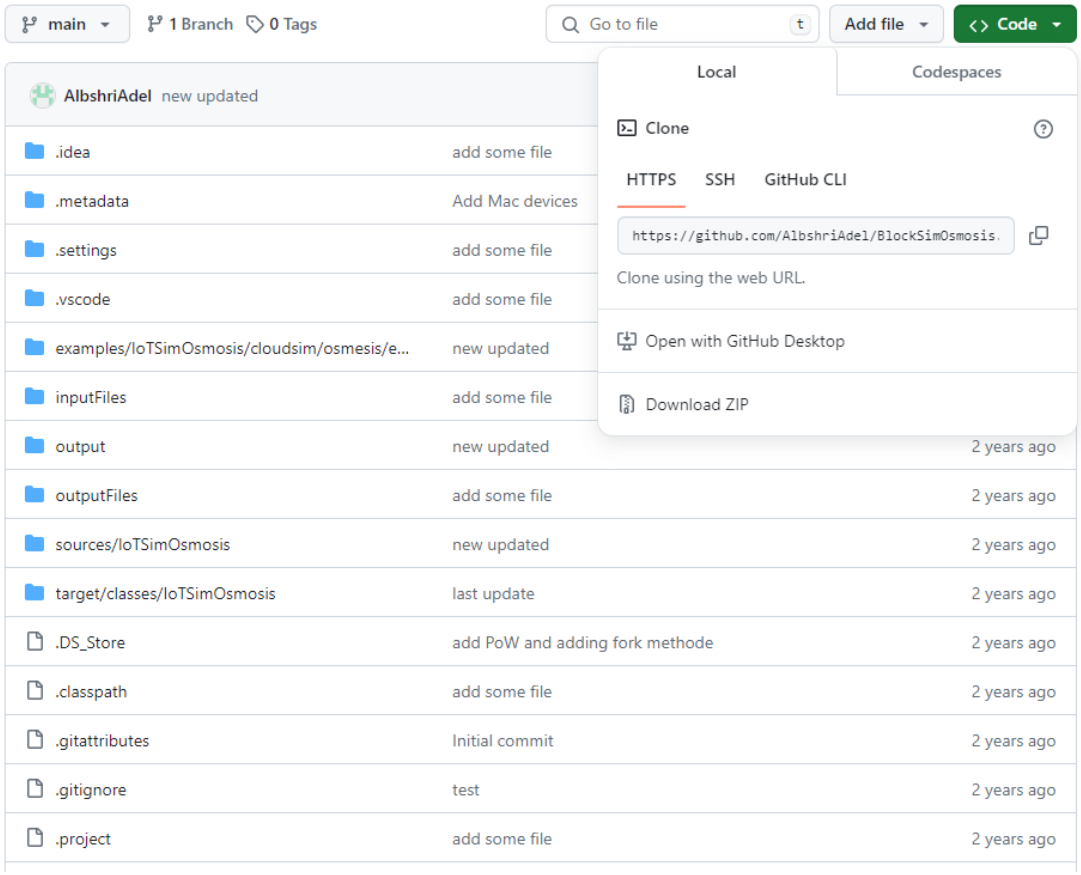


Fig. B.1 Download form GitHub

B.1.4 Directory Structure of Blockchain-based IoT simulation

The figure B.2 shows the structure of the Blockchain-based IoT simulation framework, which is defined as follows.

[1]https://github.com/AlbshriAdel/BlockSimOsmosis

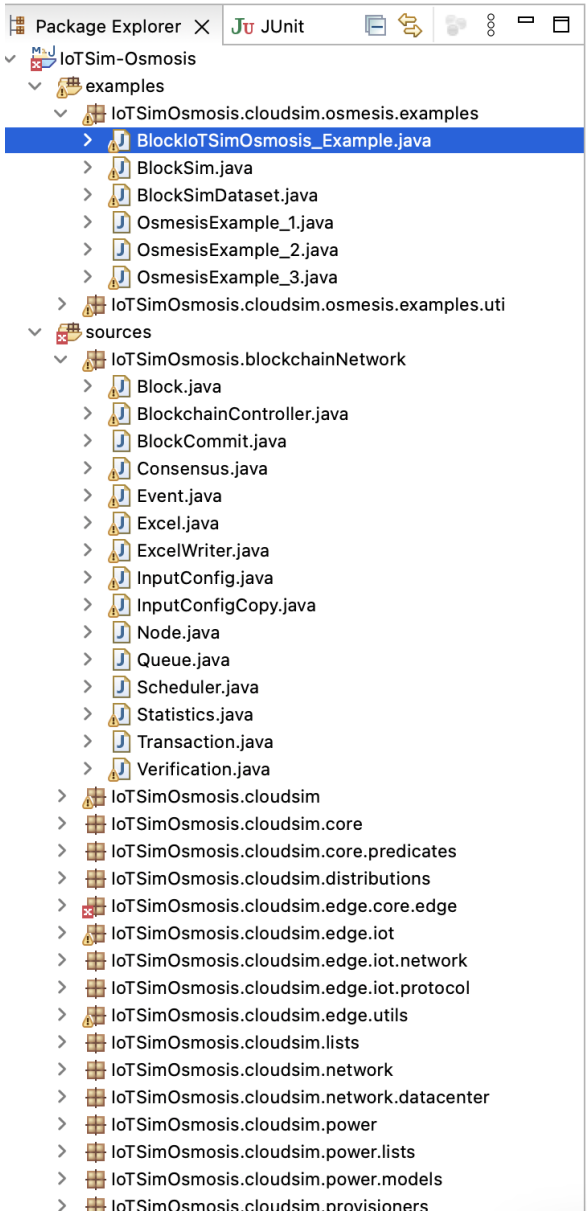


Fig. B.2 Directory Structure of Blockchain-based IoT simulation

- **BlockSimOsmosis**: The root directory of the Blockchain-based IoT simulation framework.
  - **examples**:This directory contains examples of Blockchain-based IoT simulations, as shown in Figure B.3.

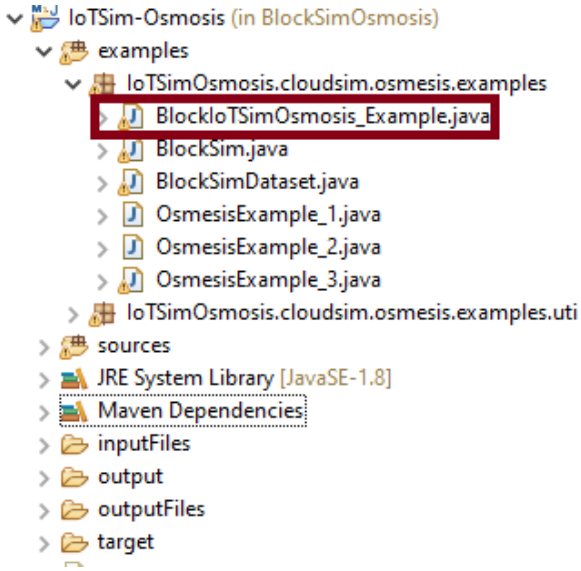


Fig. B.3 Examples Directory

– **sources**:contains the source code of Blockchain-based IoT simulation along with any input files that are necessary to run the simulation, as shown in Figure B.4.

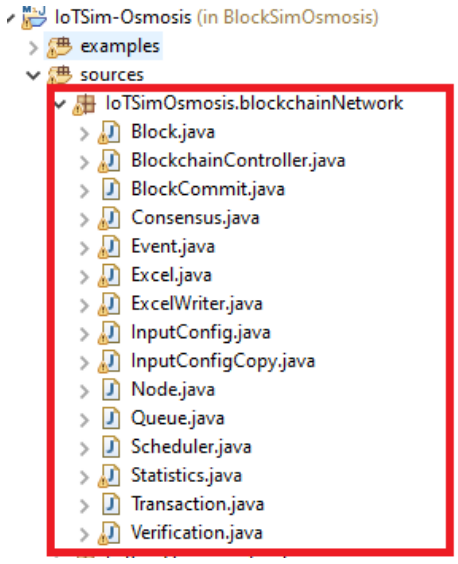


Fig. B.4 Blockchain-based IoT Sources

– **inputFiles**:Contains the required files for IoTSim-Osmosis. This directory is important for simulations that integrate IoT simulations with the blockchain, as shown in Figure B.5.

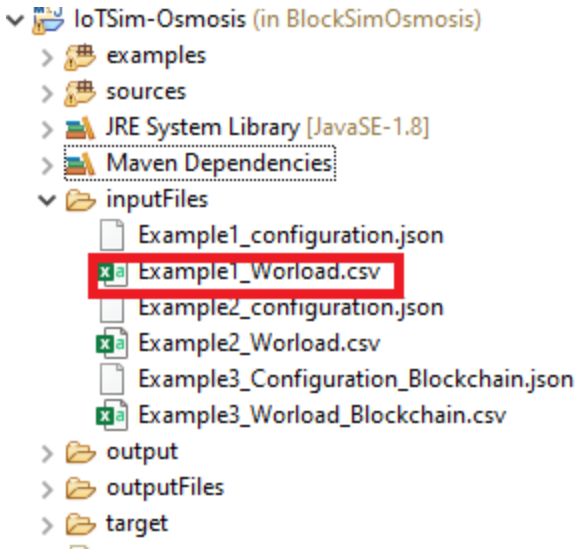


Fig. B.5 inputFiles for IoTSim-Osmosis

– **output**: Contains all output results in Excel file format, as shown in Figure B.5.

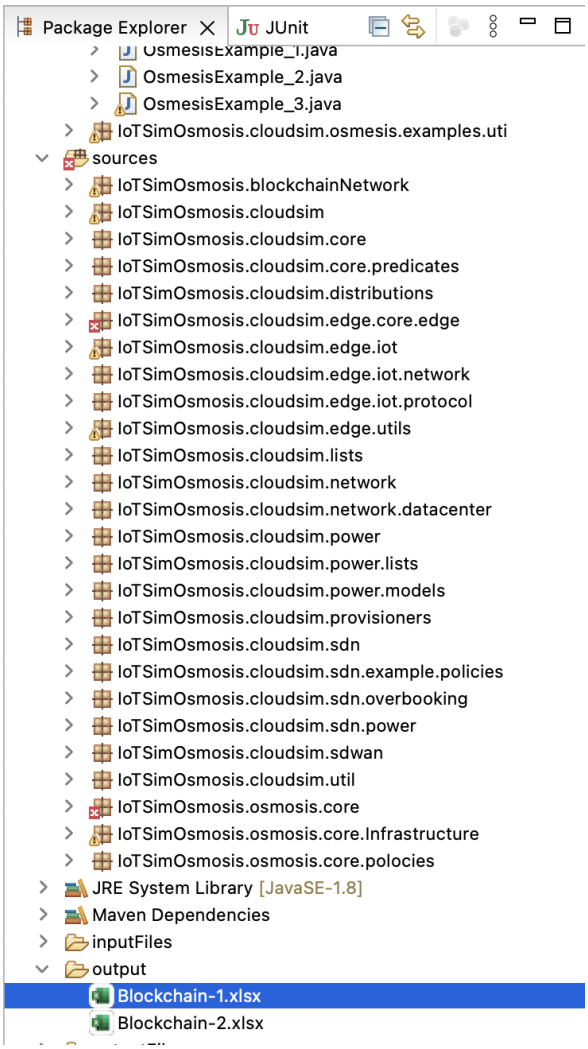


Fig. B.6 Output File for Blockchain-based IoT Simulation

B.1.5 Setup Blockchain-based IoT simulation

Before utilising Blockchain-based IoT Simulation, it is essential to correctly import and configure the project within your development environment. Therefore, this user manual provides step-by-step instructions for importing the blockchain-based IoT Simulation project using Eclipse. This simulation is based on Maven for dependency management. The primary steps include the following.

Step 1:

- Install the Eclipse IDE using the following link[2].
- Then, install Maven on Eclipse by following the steps explained in the following link[3].

Step 2:

To import a blockchain-based IoT simulation into Eclipse as a Maven project, open Eclipse, select the **File** menu and choose **Import**, as illustrated in Figure B.7.

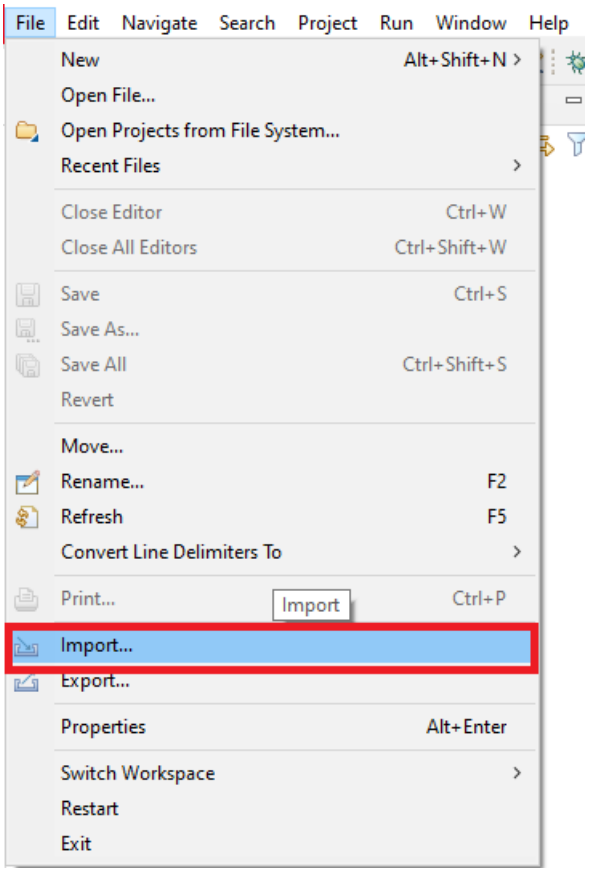


Fig. B.7 Import the Blockchain-based IoT simulation

Step 3:

Choose **Maven** and then **Existing Maven Projects**, as illustrated in Figure B.8.

[2]https://www.eclipse.org/downloads/
[3]https://www.eclipse.org/m2e/

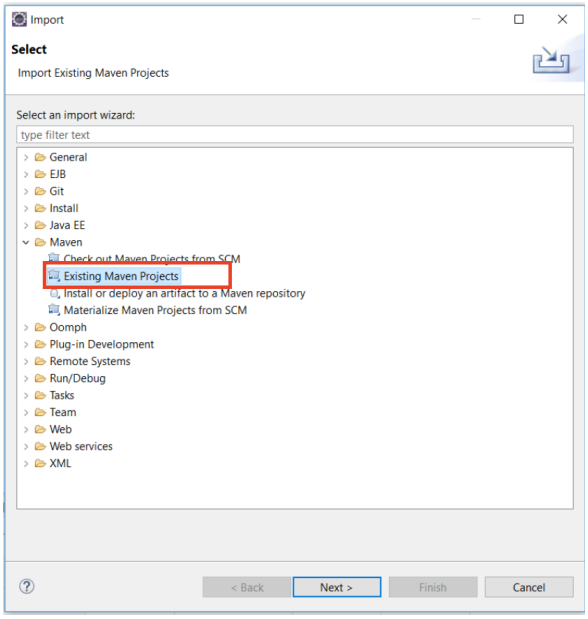


Fig. B.8 Existing Maven for Block-based IoT Project

Step 4:

Navigate to and select the Blockchain-based IoT project folder. Then, click **Finish**, as illustrated in Figure B.9 .

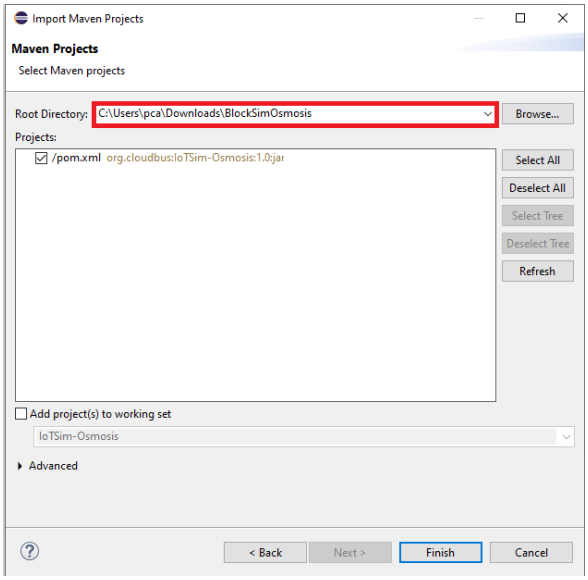


Fig. B.9 Navigate to and select the Blockchain-based IoT project

Step 5:

Right-click on the Blockchain-based IoT project, choose **Maven**, and then select **Update Project**, as illustrated in Figure B.10.

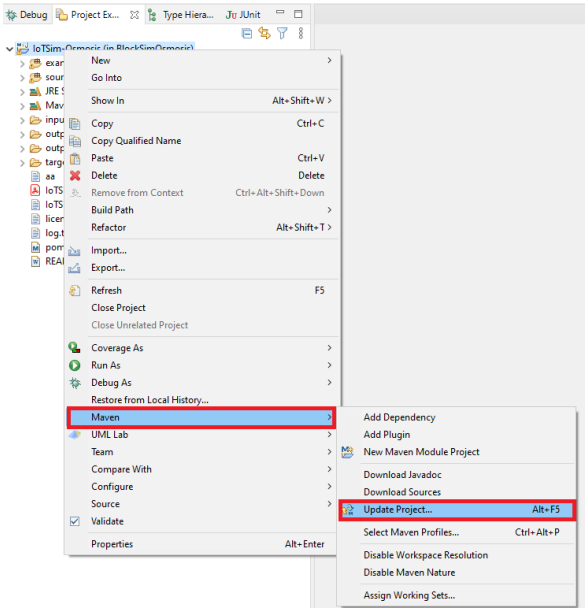


Fig. B.10 Maven Updating

Upon a successful Maven build of the blockchain-based IoT project in Eclipse, you will observe a **BUILD SUCCESS** message, as depicted in Figure B.11.

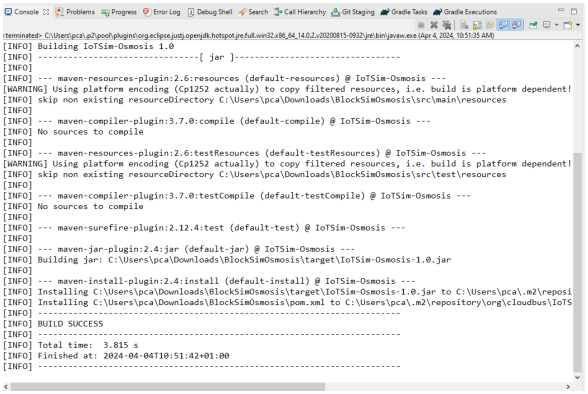


Fig. B.11 A message of build success for importing the Blockchain-based IoT Project

Step 6:

The Blockchain simulation is based on IoTSim-Osmosis which utilises the Lombok library

for entity configuration. Navigate to the Maven Dependencies directory, right-click on the **Lombok*.jar** file, and select **Run As** followed by **Java Application**, as illustrated in Figure B.12.

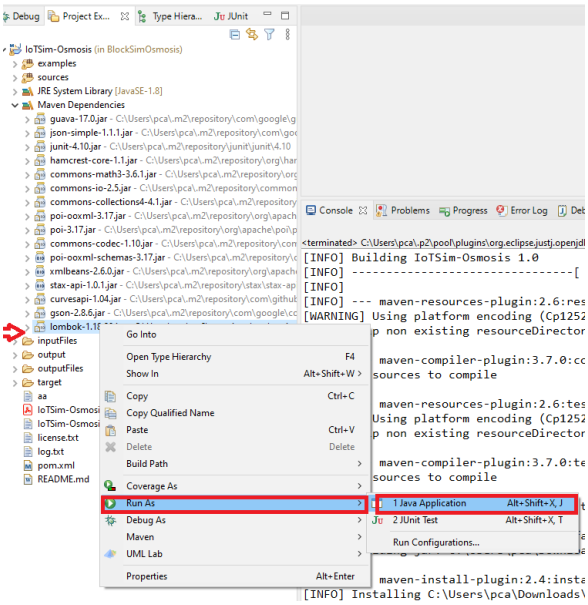


Fig. B.12 Lombok library

B.2 Simulation configuration

Before initiating the actual simulation, you must configure both the blockchain and the IoT components, as illustrated in Figures B.13 and B.14.

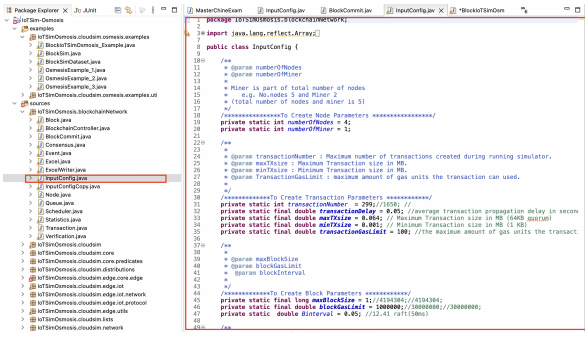


Fig. B.13 Blockchain configuration

OsmesisApp	ID	NumOfLayer (IoT, edges, clouds)	DataRate_Sec	StopDataGeneration_Sec	IoTDevice	IoTLayerName	OsmesisIoTLet_MI	IoTDeviceOutputData_Mb	MELName	datacenter	OsmesisEdgelet_MI	MELOutputData_Mb	VmName	datacenter	OsmesisCloudlet_MI
BC-IoT	1	3	1.2	250	flame sensor	IoT_1	50	90	MEL_1	Edge_1	250	70	VM_1	Cloud_1	200

Fig. B.14 IoT configuration

B.3 Simulation example

There is an example of running blockchain-based IoT simulation, as shown in Figures B.15 and B.16.

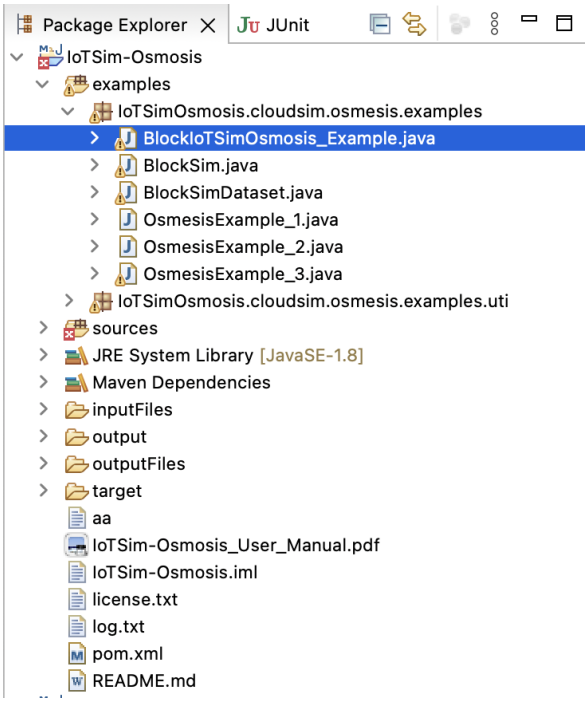


Fig. B.15 Example

```java
package IoTSimOsmosis.cloudsim.osmesis.examples;

import java.io.FileNotFoundException;

public class BlockIoTSimOsmosis_Example {

    public static final String configurationFile = "inputFiles/Example3_Configuration_Blockchain.json";
    public static final String osmesisAppFile = "inputFiles/Example3_Worload_Blockchain.csv";
    static OsmosisBuilder topologyBuilder;
    OsmesisBroker osmesisBroker;
    static EdgeDatacenters edge;
    List<OsmesisDatacenter> datacenters;
    List<MEL> melList;
    EdgeSDNController edgeSDNController;
    List<Vm> vmList;

    public static void main(String[] args) throws Exception {
        for (int runCount = 1; runCount <= InputConfig.getSimulatorRun(); runCount++) {

            BlockIoTSimOsmosis_Example osmesis = new BlockIoTSimOsmosis_Example();
            osmesis.initialOsmosisTopology();

            BlockchainController.generateOsmosisNodes(topologyBuilder);
            Node.generateGenesisBlock();
            osmesis.start();
            BlockCommit.generateInitialEvents();

            double clock = 0;
            while (!Queue.isEmpty() && (clock <= InputConfig.getSimulationTime())) {
                Event nextEvent = Queue.getNextEvent();

                clock = nextEvent.getTime();

                BlockCommit.handleEvent(nextEvent);
                Queue.removeEvent(nextEvent);

            }

            Statistics.calculate(runCount);
            System.out.println("run complete " + runCount);
            System.out.println("");
            BlockchainController.restState();

        }
    }

```

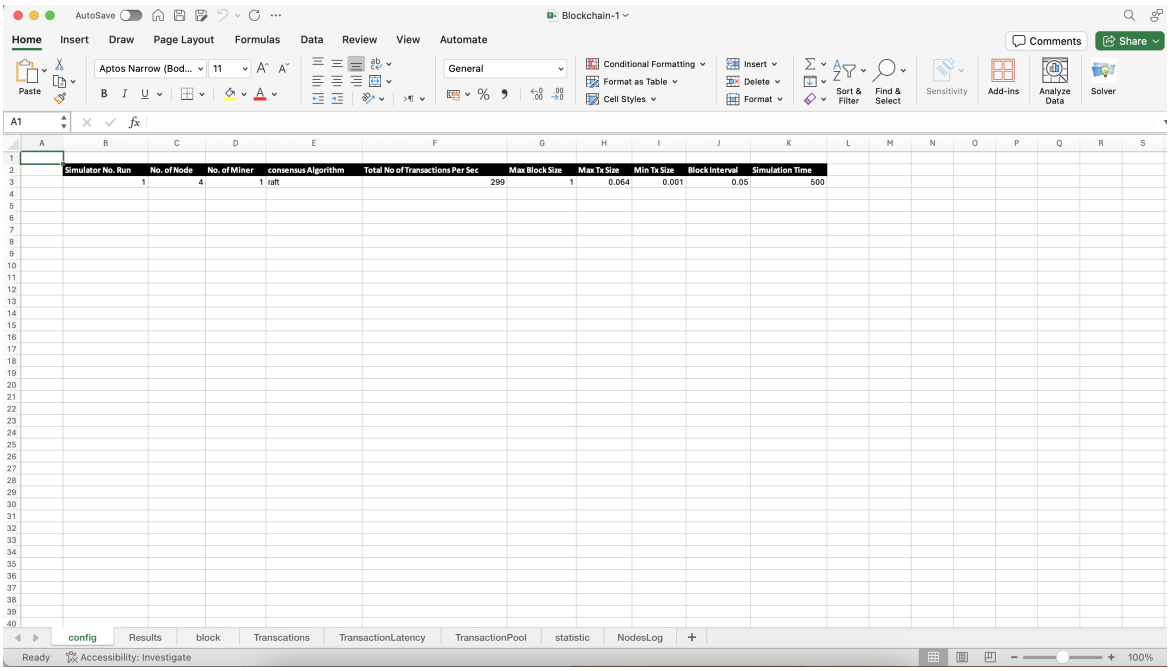
Fig. B.16 The main example to run the Blockchain-based IoT simulation

B.4 Output results

Once the blockchain-based IoT simulation finishes running, it would produce results in Excel format. The results are structured as follows:

Configuration:

This provides information on the parameters used to carry out the experiment, as shown in Figure B.17.



Simulator No. Run	No. of Node	No. of Miner	consensus Algorithm	Total No of Transactions Per Sec	Max Block Size	Max Tx Size	Min Tx Size	Block Interval	Simulation Time
1	4	1	raft	299	1	0.064	0.001	0.05	500

Fig. B.17 Configuration

Overall result:

A benchmark report provides a summary of the overall performance of a blockchain-based IoT, as shown in Figure B.18.

No. of Blocks	Total No. of Blocks include Tx	Total No. of Blocks without Tx	Avg. Block Size (MB)	Total No of Transactions	Avg. No. of Tx per block	Avg. of Tx Inclusion Time (secs)	Avg. Tx Size (MB)	Total No. of Pending Tx	Avg. Block Propagation (secs)	Avg. Transaction Latency (secs)	Transactions execution (secs)	Transaction Throughput (Tx/secs)
16354	16353	1	0.29678425	149499	9.141433288	125.3152266	0.032465833	0	0.030518105	0.283722451	499.0930897	299.5413142

Fig. B.18 Overall result

Blocks overview:

A benchmark report provides details about the individual blocks that were added to the blockchain during the simulation, as shown in Figure B.19.

Simulator No. Run	Block ID	Previous Block ID	Block Depth	Block Timestamp	Block Size	No. of Transactions	Mined by
1	0	-1	0	0	0	0	Null
1	95331905147	0	1	0.036610215	0.999976357	32	0
1	43649050289	95331905147	2	0.046620706	0.998286875	29	0
1	89125324301	43649050289	3	0.084185056	0.99953011	34	0
1	4622255798	89125324301	4	0.097169123	0.999738202	33	0
1	64413791986	4622255798	5	0.126720616	0.999776302	29	0
1	67573982436	64413791986	6	0.162834127	0.999734602	35	0
1	24015996743	67573982436	7	0.196484958	0.999817735	28	0
1	96505169462	24015996743	8	0.216037037	0.999565197	29	0
1	52309259556	96505169462	9	0.236434679	0.999462432	29	0
1	41776480733	52309259556	10	0.284671881	0.998068898	33	0
1	49895341845	41776480733	11	0.321999599	0.999563856	33	0
1	91151734106	49895341845	12	0.355270142	0.999407422	31	0
1	99528545301	91151734106	13	0.369592545	0.99907302	31	0
1	23003298553	99528545301	14	0.387379355	0.999442003	30	0
1	33423091316	23003298553	15	0.432429164	0.999898949	39	0
1	90810279838	33423091316	16	0.47574051	0.999962951	33	0
1	36044944498	90810279838	17	0.480801991	0.999574891	29	0
1	43017960533	36044944498	18	0.49753344	0.999271616	33	0
1	70517404478	43017960533	19	0.505698616	0.999018409	27	0
1	28295726424	70517404478	20	0.51814313	0.999949197	34	0
1	79561345451	28295726424	21	0.522170905	0.999244202	31	0
1	47458270359	79561345451	22	0.540142703	0.999925968	31	0
1	36098746585	47458270359	23	0.558851138	0.999703328	37	0
1	54898279451	36098746585	24	0.58378775	0.999766488	30	0
1	6845847707	54898279451	25	0.628328278	0.999286529	35	0
1	43386606885	6845847707	26	0.644782122	0.998811315	30	0
1	64549601907	43386606885	27	0.682916195	0.999242463	27	0
1	50845544641	64549601907	28	0.727499893	0.999543333	34	0
1	70517490532	50845544641	29	0.772670042	0.999782398	27	0
1	28142577783	70517490532	30	0.803411817	0.999983835	26	0
1	51793397875	28142577783	31	0.832673838	0.999899292	30	0
1	58130037529	51793397875	32	0.842200383	0.999214235	31	0
1	85716638079	58130037529	33	0.864984058	0.999795417	34	0
1	5214101607	85716638079	34	0.889997046	0.999649227	34	0
1	93221179566	5214101607	35	0.890348008	0.998982943	30	0
1	54172040881	93221179566	36	0.891883038	0.99888901	30	0
1	11010092878	54172040881	37	0.924060873	0.999757411	31	0

Fig. B.19 Blocks overview

Transactions latency overview:

A benchmark report provides details on the latency for each transaction in a blockchain-based IoT, as shown in Figure B.20.

Simulator No. Run	Transaction ID	Creation time	Confirmation time	Transaction Latency
1	4489308117	9.40026E-05	0.036610215	0.036516212
1	9894469430	0.000393402	0.036610215	0.036216813
1	8369963418	0.001509106	0.036610215	0.035101109
1	5259357711	0.003047675	0.036610215	0.03356254
1	2172664072	0.003228376	0.036610215	0.033381839
1	9639045370	0.003622316	0.036610215	0.032987899
1	8134921392	0.003858889	0.036610215	0.032751326
1	2525974781	0.003962162	0.036610215	0.032648053
1	6325104922	0.004957841	0.036610215	0.031652374
1	3024814198	0.005187609	0.036610215	0.031422606
1	4817535841	0.005204698	0.036610215	0.031405517
1	9932429784	0.006235673	0.036610215	0.030374542
1	4258848733	0.006512934	0.036610215	0.030097281
1	9580340874	0.007460464	0.036610215	0.029149751
1	6870309267	0.008019692	0.036610215	0.028590523
1	4577416916	0.008519084	0.036610215	0.028091131
1	7948084247	0.008557827	0.036610215	0.028052388
1	2402701855	0.008918928	0.036610215	0.027691287
1	4194953418	0.010143289	0.036610215	0.026466926
1	3693618146	0.010373245	0.036610215	0.02623697
1	3086946458	0.010401262	0.036610215	0.026208953
1	9826292571	0.01101138	0.036610215	0.025598835
1	3160891819	0.011987466	0.036610215	0.024622749
1	9350929897	0.014810656	0.036610215	0.021799559
1	7365686791	0.014855058	0.036610215	0.021755157
1	2129405733	0.014857719	0.036610215	0.021752496
1	4750566423	0.015204414	0.036610215	0.021405801
1	9525887510	0.015711944	0.036610215	0.020898271
1	1851900717	0.015714877	0.036610215	0.020895338
1	2819044160	0.015726956	0.036610215	0.020883259
1	1356972801	0.01594348	0.036610215	0.020666735
1	1918174571	0.017077733	0.036610215	0.019532482
1	4647123191	0.015940304	0.046620706	0.030680402
1	7343413096	0.01652377	0.046620706	0.030096936
1	6603656192	0.016651018	0.046620706	0.029969688
1	3202402654	0.016689287	0.046620706	0.029931419
1	8622696301	0.01729096	0.046620706	0.029329746
1	5539627640	0.017434342	0.046620706	0.029186364

Fig. B.20 Transactions latency overview

Appendix C

Real Blockchain Implementation and Performance Benchmarks

This Appendix demonstrates the practical application of a real blockchain (e.g., quorum Blockchain) setup and deployment into a cloud infrastructure. It also illustrates the process of utilising Hyperledger Caliper (a well-established blockchain performance framework) to evaluate the performance of the real blockchain application and obtain its performance metrics.

C.1 Environment Setup

This section describes the infrastructure and software requirements for deploying a real-world blockchain system over a cloud infrastructure.

C.1.1 Cloud Infrastructure Deployment

In our experiment, we utilised a cloud-based infrastructure leased for the experiment purposes. It consists of 32 virtual CPUs (vCPUs) derived from an Intel(R) Xeon(R) Gold 6140 processor, operating at a frequency of 2.30GHz, complemented by 64GB of RAM, as discussed in Section 5.4.2.

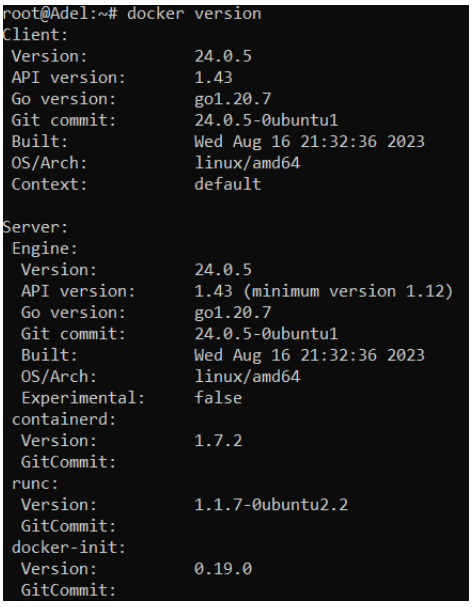
C.1.1.1 Dependency Resources

1. The operating system used for the experiments Linux Ubuntu (server version).

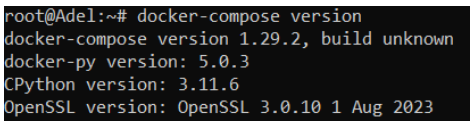
2. Docker and Docker Compose are mainly used for installing and building the blockchain network. The following commands ensure their installation:

```
# Update the package lists.
sudo apt-get update
# Install Docker.
sudo apt-get install docker.io
# Install Docker Compose.
sudo apt-get install docker-compose
```

Figures C.1a and C.1b delineate the versions of Docker and Docker Compose that we utilize in our experiment.



(a) Docker Version



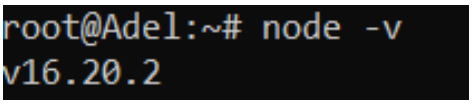
(b) Docker-compose Version

Fig. C.1 Docker

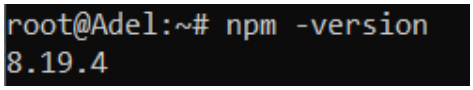
3. The following command is to install Node.js and npm, which are critical for running JavaScript server-side and managing project dependencies, respectively. They are

especially important for managing the deployment of smart contracts and providing an interface that enables IoT clients and other software to interact with the deployed smart contract. Figure C.2a and C.2b show the Nodejs and npm installed versions.

```
# Install Node.js.
apt install nodejs
# Install npm.
apt install npm
```



(a) Nodejs Version



(b) NPM Version

Fig. C.2 Nodejs and NPM

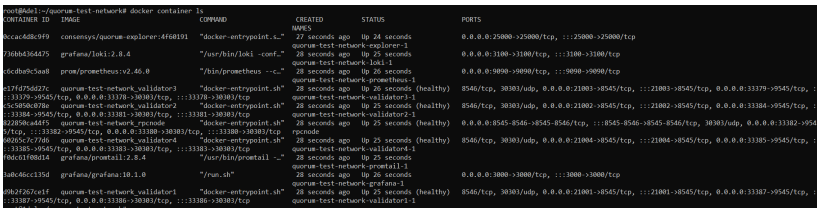
C.1.2 Deployment of the Real Blockchain Platform

Quorum blockchain is the real blockchain platform used for the purposes of evaluating the accuracy of the proposed simulation in chapter 5. The deployment of the blockchain took place on the rented cloud infrastructure, as previously mentioned in Section C.1.1, by leveraging the development quick start guide provided by Quorum, as presented below.

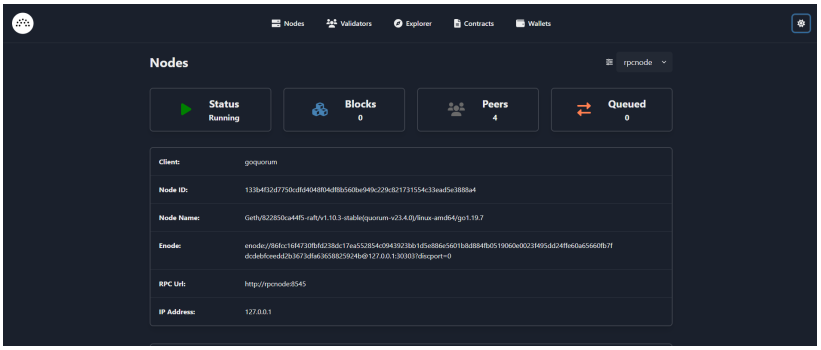
```
npx quorum-dev-quickstart
```

Figure C.3a delineates the procedural steps undertaken for deploying the Quorum blockchain network. Subsequently, Figure C.3b illustrates the operational state of the blockchain nodes. The functionality and status of these nodes were examined through the inspection of Docker containers, with each node operating within its respective container. Additionally, Figure C.3c offers an overview of operational blockchain nodes within the network.

(a) The deployment of the Quorum blockchain



(b) Active blockchain nodes



(c) An overview of active nodes

Fig. C.3 Quorum blockchain

C.2 Hyperledger Caliper

Hyperledger Caliper is a well-established blockchain performance evaluation framework that can adapt to various blockchain platforms such as Quorum, Hyperledger Fabric and Ethereum. This thesis utilises it to benchmark the performance of the deployed blockchain network to obtain real performance metrics. These metrics are intended to be compared with ones produced by the proposed simulator in chapter 5. Following are the configuration details, the benchmark configurations, the network interface specifications, and the IoT behaviour logic for each benchmark worker.

C.2.1 Benchmark Configuration of IoT-based Firefighting Scenario

Listing C.2.1 illustrates an example of configuring Hyperledger Caliper to benchmark the real blockchain platform. It defines the IoT device and its functionality. It also sets the total number of transactions (i.e. 3200), The number of transactions to be sent per second (i.e. 3200 indicating all at once), and the logic the benchmark worker should follow. For instance, this thesis encoded an IoT-based logic to express the process of sensing a fire alert based on a positive alert emitted from a flame sensor. The benchmark listing here shows that this logic is defined in a JavaScript file named (sendFireAlerts.js). As shown in Figure C.6, the benchmark undertook several rounds where the send rate changes for each round from 25 transactions per second to 3200 transactions per second. Therefore, this file changes with this respect for every benchmark round to reflect the intended number of sent transactions per second.

```
1 simpleArgs: &simple-args
2   IoTDevice: 'flameSensor'
3   fireAlert: true
4 test:
5   name: simple
6   description: >-
7     This is an example benchmark for Caliper, to test the
        backend DLT's
```

```
8     performance with IoT-based firefighting system and fire
          alert transactions.
9   workers:
10    type: local
11    number: 1
12  rounds:
13    - label: Fire Alerts
14      description: >-
15        Test description for the sending of fire alert to smart
              contract.
16      txNumber: 3200
17      rateControl:
18        type: fixed-rate
19        opts:
20          tps: 3200
21      workload:
22        module: /root/benchmark/sendFireAlerts.js
23        arguments: *simple-args
```

C.2.2 Network Configuration of IoT-based Firefighting Scenario and Smart Contract Deployment

The Hyperledger Caliper has to interface with the blockchain system under test to conduct the performance evaluation. Listing C.2.2 illustrates a network interface file for connecting and communicating with the corresponding blockchain platform. First, a smart contract is encoded to represent the logic of responding to fire alerts emitted from false sensors through benchmark workers. The smart contract is then compiled and deployed to each validating blockchain node in the network to execute received transactions (fire alerts). A special property called contracts indicates how to access the smart contract (through a file named IoTEmergencySystem.json). Also, this file defines the types of the blockchain network under

test. Here, it indicates Ethereum because Qoroum is essentially a consortium blockchain version extended from Ethereum. The network interface file also defines which URL address to route transactions to and what account to pay the transaction fees for transactions and contract deployment. The experiment uses dummy coins for conducting the test to be consumed by the benchmark workers.

```
{
  "caliper": {
      "blockchain": "ethereum",
      "command" : {
        "start": "echo start benchmark",
        "end": "echo finish benchmark"
        }
  },
  "ethereum": {
      "url": "ws://127.0.0.1:8546",
      "fromAddress": "0xc9c913c8c3c1cd416d80a0abf475db2062f
         161f6",
      "contractDeployerAddress": "0xc9c913c8c3c1cd416d80a0
         abf475db2062f161f6",
      "transactionConfirmationBlocks": 12,
      "contracts": {
          "IoTEmergencySystem": {
              "path": "/root/benchmark/IoT/
                 IoTEmergencySystem.json",
              "gas": {
                  "sendFireAlerts": 45000
              }
          }
      }
```

```
22    }
23 }
```

C.2.3 Worker Behaviour: Fire Alerts

Each worker participates in benchmarking the blockchain performance by sending transactions as defined in the benchmark configurations and the network interface files. Listing C.2.3 express the IoT logic to be followed by each worker when sending transactions (fire alerts) to the respective smart contract over the blockchain network. It also captures the duration of when a transaction begins and ends for measuring the throughput and latency. All capture values are recorded into an Excel file for later evaluation.

```
1  'use strict';
2
3  const fs = require('fs');
4  const OperationBase = require('/root/benchmark/utils/operation-
       base');
5  const SimpleState = require('/root/benchmark/utils/simple-state'
       );
6
7  /**
8   * Workload module for initializing the SUT with various
        accounts.
9   */
10 class IoTLogic extends OperationBase {
11     /**
12      * Initializes the parameters of the workload.
13      */
14     constructor() {
15         super();
```

```
16          this.csvFile = '/root/benchmark/transaction_details.csv';
                // Path to the CSV file
17          this.counter = 0; // Counter for transactions
18      }
19
20      /**
21       * Create an empty state representation.
22       * @return {SimpleState} The state instance.
23       */
24      createSimpleState() {
25          return new SimpleState(this.workerIndex, this.IoTDevice,
                this.fireAlert);
26      }
27
28      /**
29       * Assemble TXs for IoTLogicing new accounts.
30       */
31      async submitTransaction() {
32          let createArgs = this.simpleState.
                getIoTLogicAccountArguments();
33          const startTime = Date.now();
34          try {
35              await this.sutAdapter.sendRequests(this.
                    createConnectorRequest('IoTLogic', createArgs));
36          } catch (error) {
37              console.error('Error submitting transaction:', error
                    );
38              return;
39          }
40          const endTime = Date.now();
41          const latency = endTime - startTime;
```

```
42          const csvData = `${this.counter},${startTime},${endTime
               },${latency}\n`;
43          try {
44              fs.appendFileSync(this.csvFile, csvData);
45          } catch (error) {
46              console.error('Error saving transaction details:',
                   error);
47          }
48          this.counter++;
49      }
50  }
51
52  /**
53   * Create a new instance of the workload module.
54   * @return {WorkloadModuleInterface}
55   */
56  function createWorkloadModule() {
57      return new IoTLogic();
58  }
59  module.exports.createWorkloadModule = createWorkloadModule;
```

C.2.4 Python Script to Automate Deployment and Performance benchmark

Given the multiple rounds required for obtaining sufficient performance data, all commands related to executing the above-mentioned configurations and specifications are automated by encoding into Python script, as per Listing C.2.4. The generated performance reports by Hyperledger Caliper are captured by this script and recorded into an Excel file for later statistical analysis and pattern observation (such as changes in average throughput and latency and their correlation). Figure C.4 shows the outcome of the Script.

```python
import subprocess
import pandas as pd
from bs4 import BeautifulSoup
import re
import time

# Define the number of times to run the benchmark
num_runs = 1

# Generate the shell command for running the benchmark.
shell_command = f"npx caliper launch manager \
        --caliper-networkconfig quorum-test-network/
           networkconfig.json \
        --caliper-benchconfig benchmark/benchconfig.yaml"

# Define the CSV file name
csv_file = "benchmark_results.csv"

# Run the benchmark multiple times
for i in range(num_runs):
    print(f"Running benchmark iteration {i + 1}")
    subprocess.run(shell_command, shell=True)

    # Read data from the report.html file
    report_file = f"report.html"

    # Extract the table content using BeautifulSoup
    with open(report_file, "r") as f:
        html_content = f.read()

    soup = BeautifulSoup(html_content, "html.parser")
```

```
    tables = soup.find_all("table")

    if tables:
        # Parse the first table HTML into a DataFrame
        report_data = pd.read_html(str(tables[0]))[0]

        # Rename the columns to match the desired format
        column_names = ["Name", "Succ", "Fail", "Send Rate (TPS)", "Max Latency (s)", "Min Latency (s)", "Avg Latency (s)", "Throughput (TPS)"]
        report_data.columns = column_names

        # Modify the "Name" column values
        report_data["Name"] = report_data["Name"].str.replace("_R", f"_{i+1}")

        # Append data to the CSV file
        report_data.to_csv(csv_file, mode="a", index=False, header=not i)
    else:
        print(f"No performance metrics table found in {report_file}.")

    # Add a 1-second delay before the next iteration
    time.sleep(120)
```

	Name	Succ	Fail	Send Rate (TPS)	Max Latency (s)	Min Latency (s)	Avg Latency (s)	Throughput (TPS)
2	open_1	25	0	26.1	1.03	1.01	1.01	12.7
3	open_2	25	0	26.1	1.03	1.01	1.01	12.7
4	open_3	25	0	26.2	1.04	1.01	1.01	12.7
5	open_4	25	0	26.2	1.03	1.01	1.01	12.7
6	open_5	25	0	26.2	1.05	1.01	1.02	12.7
7	open_6	25	0	26.2	1.03	1.01	1.01	12.7
8	open_7	25	0	26.2	1.03	1.01	1.01	12.7
9	open_8	25	0	26.2	1.02	0.03	0.97	12.7
10	open_9	25	0	26.2	1.02	0.04	0.97	12.7
11	open_10	25	0	26.2	1.03	1.01	1.02	12.7
12	open_11	25	0	26.2	1.04	1.01	1.02	12.7
13	open_12	25	0	26.2	1.04	0.05	0.98	12.7
14	open_13	25	0	26.2	1.04	1.01	1.01	12.7
15	open_14	25	0	26.1	1.03	1.01	1.01	12.7
16	open_15	25	0	26.2	1.04	1.01	1.01	12.7
17	open_16	25	0	26.2	1.03	1.01	1.01	12.7
18	open_17	25	0	26.1	1.02	0.03	0.97	12.7
19	open_18	25	0	26.2	1.02	0.04	0.97	12.7
20	open_19	25	0	26.2	1.03	1.01	1.01	12.7
21	open_20	25	0	26.2	1.03	1.01	1.01	12.7
22	open_21	25	0	26.2	1.04	1.01	1.02	12.7
23	open_22	25	0	26.1	1.03	1.01	1.01	12.7
24	open_23	25	0	26.2	1.03	1.01	1.01	12.7
25	open_24	25	0	26.3	1.04	1.01	1.01	12.7
26	open_25	25	0	26.2	1.03	1.01	1.01	12.7
27	open_26	25	0	26.1	1.04	1.01	1.02	12.7
28	open_27	25	0	26.2	1.03	1.01	1.01	12.7
29	open_28	25	0	26.2	1.03	1.01	1.01	12.7
30	open_29	25	0	26.2	1.04	1.01	1.01	12.7
31	open_30	25	0	26.2	1.03	0.03	0.97	12.7
32	open_31	25	0	26.1	1.03	1.01	1.01	12.7
33	open_32	25	0	26.2	1.03	1.01	1.01	12.7
34	open_33	25	0	26.1	1.03	1.01	1.01	12.7
35	open_34	25	0	26.2	1.03	1.01	1.01	12.7
36	open_35	25	0	26.1	1.03	1.01	1.01	12.7
37	open_36	25	0	26.2	1.03	1.01	1.01	12.7
38	open_37	25	0	26.2	1.03	1.01	1.01	12.7

Fig. C.4 Reformatting and Organising the Collected Data from HyperLedger Caliper into Excel Format

C.2.5 Blockchain Performance Report Generated by Hyperledger Caliper

By executing each benchmark round, a performance report is generated and preserved as in Figure C.5.

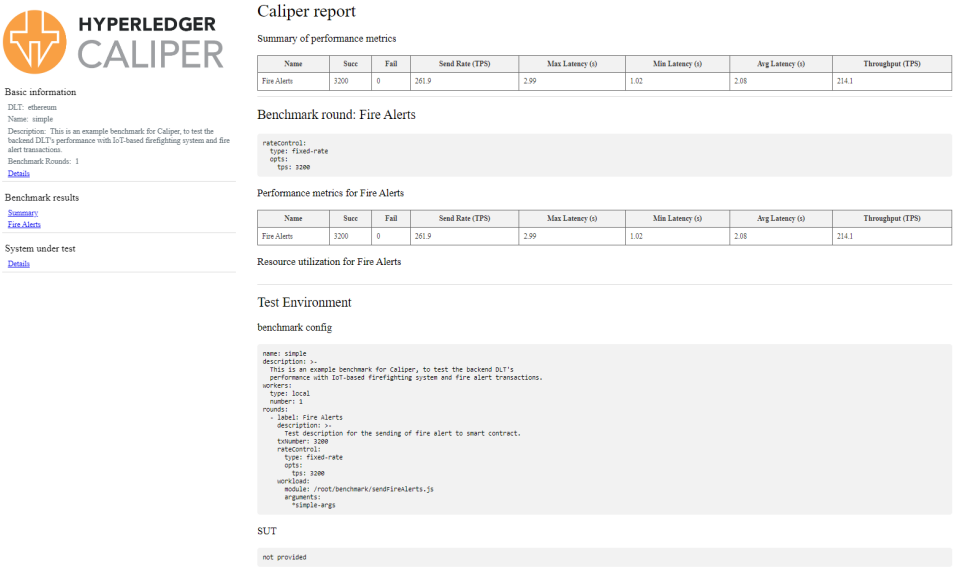


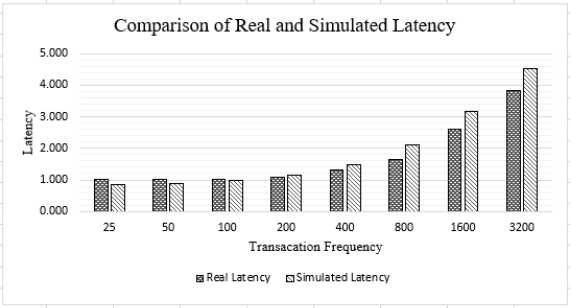
Fig. C.5 Example Blockchain Performance Report for One of the Benchmark Rounds

C.2.6 Comparative Analysis

Figure C.6 depicts the process of accumulating data about the performance evaluation from the simulator and the real-world blockchain platform, as explained above. Two performance metrics are considered: throughput and latency, which are used for the comparative analysis between the simulator and the real blockchain platform.

Transcation	Real Latency	Simulated Latency
25	1.004	0.845
50	1.007	0.890
100	1.011	0.978
200	1.084	1.152
400	1.300	1.488
800	1.631	2.112
1600	2.590	3.168
3200	3.832	4.512

Frequency	Real Throughput (TPS)	Simulated Throughput (TPS)
25	12.698	13.121
50	25.144	36.378
100	50.078	59.636
200	95.536	82.894
400	123.312	106.152
800	143.473	129.410
1600	140.653	152.667
3200	165.292	175.925



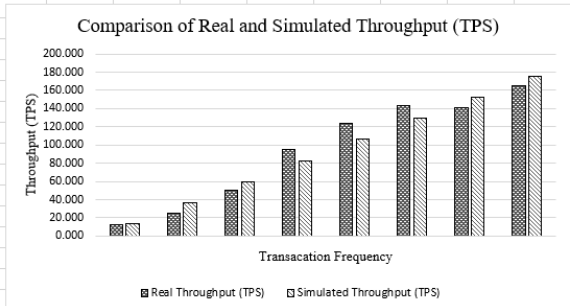


Fig. C.6 Comparative Analysis between the Real Blockchain Platform and the Simulator in Terms of Latency and Throughput