DOKUZ EYLÜL UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

INTERNET OF THINGS SECURITY WITH BLOCKCHAIN

by

Hakan ALTAŞ

June, 2022

İZMİR

INTERNET OF THINGS SECURITY WITH BLOCKCHAIN

A Thesis Submitted to the Graduate School of Natural and Applied Sciences of Dokuz Eylül University In Partial Fulfillment of the Requirements for the Master of Science in Computer Engineering

by

Hakan ALTAŞ

June, 2022

İZMİR

M.Sc THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **"INTERNET OF THINGS SECURITY WITH BLOCKCHAIN"** completed by **HAKAN ALTAŞ** under supervision of **ASSOC. PROF. DR. GÖKHAN DALKILIÇ** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

……..

Assoc. Prof. Dr. Gökhan DALKILIÇ

Supervisor

……..	……..
Prof. Dr. Yalçın ÇEBİ	Asst. Prof. Dr. Ömer AYDIN
Jury Member	Jury Member

Prof. Dr. Okan FISTIKOĞLU

Director

Graduate School of Natural and Applied Sciences

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Assoc. Prof. Dr. Gökhan DALKILIÇ giving me valuable advice, guidance, and support always when needed.

Finally, I like to show my gratitude to thanks Mehmet Emin ALTAŞ and Ayşe ALTAŞ whose are my family. Also, I want to thank my friends and colleagues, Yiğit Sedat CEYHAN, Oğuzhan YAVUZ, Dhurata YMERI YÜKSEL, Arman Barış ÇALLI, Umut Can ÇABUK and Oğuzhan UYSAL for their constant support and encouragement to boost my positive energy and motivation throughout the process of writing my thesis.

Hakan ALTAŞ

INTERNET OF THINGS SECURITY WITH BLOCKCHAIN

ABSTRACT

The Internet of things (IoT) is one of the pioneer concepts that is shaping the "smart" world of today and the future. It is a core enabler of advancements in many fields, from agriculture to astronomy, industrial automation to autonomous vehicles, etc. However, most IoT deployments include cost-efficient lightweight devices with limited resources. Hence, the entire network must be built in the simplest way. This necessity forces engineers to include more sophisticated devices in the network, such as gateways and servers. So that, although the nodes are widely distributed in a geographical or topological sense, the network turns into a centralized structure that eventually creates bottlenecks and single-points-of-failure.

Among the issues caused by that phenomenon, we focused on solving the data manipulation and event management problems. In a nutshell, in most IoT scenarios, data flows from sensor devices to data storage and processing units. Contrarily, event-driven commands flow from these units to actuator devices, if any. Therefore, an attacker who gained access to the centralized units can leak, alter, or even remove critical data and may exploit event handling features. This is where blockchain, another revolutionary technology, may be extremely handy. Integration of a decentralized ledger as a data storage provides data integrity, immutability, and non-repudiation for an IoT deployment. Moreover, the adoption of a custom smart contract lets the IoT deployment benefit from decentralized and immutable event management features, too. Our thesis introduces a novel IoT architecture scheme that incorporates an Ethereum-based private blockchain that runs an ad-hoc smart contract in an MQTT-based IoT deployment containing sensor and actuator nodes to provide the functionalities. Apart from the architectural design, the thesis also presents a simulation-based proof-of concept built for validation purposes, as well as performance measurements. Per our benchmarks, the proposed scheme is shown to be valid, scalable, and efficient for various IoT scenarios.

Keywords: Blockchain, ethereum, immutability, IoT security, smart contract.

BLOK ZİNCİRİ İLE NESNELERİN İNTERNETİ GÜVENLİĞİ

ÖZET

Nesnelerin İnterneti (IoT), bugünün ve geleceğin "akıllı" dünyasını şekillendiren öncü kavramlardan biridir. Tarımdan astronomiye, endüstriyel otomasyondan otonom araçlara kadar birçok alanda gelişimin temel öncülerindendir. Bununla birlikte, çoğu IoT cihazı sınırlı kaynaklara sahiptir. Bunun ek olarak, bu cihazlar için tüm IoT ağının en basit şekilde oluşturulması gerekmektedir. Bu gereklilik, mühendisleri ağ geçitleri ve sunucular gibi daha karmaşık cihazları ağa dahil etmeye zorlar. Böylece, düğümler coğrafi veya topolojik anlamda geniş bir alana dağılmış olsa da, ağ, sonunda darboğazlar ve tek arıza noktasından etkilenebilir bir merkezi yapıya dönüşür.

Biz bu ilgili yapının neden olduğu sorunlardan veri manipülasyonu ve olay yönetimi sorunlarını çözmeye odaklandık. Özetle, çoğu IoT senaryosunda, veriler sensör cihazlarından veri depolama ve işleme birimlerine akar. Bunun dışında, gelen veriye göre bir aksiyon alınması gerekiyorsa, bu birimlerden aktüatör cihazlara komutlar aktarılır. Bu nedenle, merkezi birimlere erişim elde eden bir saldırgan kritik verileri sızdırabilir, değiştirebilir ve hatta kaldırabilir ve olay işleme özelliklerinden yararlanabilir. Bu sorun, bir başka devrim niteliğindeki teknoloji olan blockchain'in son derece yararlı olabileceği bir sorundur. Merkezi olmayan bir defterin, veri depolamak için entegrasyonu, bir IoT merkez birimi için veri bütünlüğü, değişmezlik ve reddedilemezlik sağlar. Ayrıca, özel bir akıllı sözleşmeninde eklenmesi, merkezi olmayan ve değiştirilemez olay yönetimi birimi sağlar. Tezimiz, işlevleri sağlamak için sensör ve aktüatör düğümleri içeren MQTT tabanlı bir IoT dağıtımında akıllı sözleşme çalıştıran Ethereum tabanlı bir özel blok zinciri içeren yeni bir IoT mimarisi şemasını tanıtıyor. Mimari tasarımın yanı sıra, performans ölçümlerinin yanı sıra doğrulama amacıyla oluşturulmuş simülasyon tabanlı bir örnek de sunmaktadır. IoT senaryoları için geçerli, ölçeklenebilir ve verimli olduğu gösterilmiştir.

Anahtar kelimeler: Blok zinciri, ethereum, değişmezlik, IoT güvenliği, akıllı sözleşmeler.

CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

Page

CHAPTER ONE
INTRODUCTION

In many Internets of things (IoT) ecosystems, the integration of intermediary subsystems, also called IoT platforms, has become prevalent. They are the intersection points, where data reporting (e.g., sensor) IoT devices send their messages and data collecting (e.g., actuator) devices get new messages, representing a sort of a specialized gateway. Such sensor data is becoming the main enabler of handling event management, advanced event processing, or big data analyses (Rogojanu, Ghita, Stanciu, Ciobanu, Marin, Pop & Dobre, 2018).

Analyzing big data collections, handling event management, and advanced event processing become integral functions of IoT deployments, while these activities are also supported by the IoT platforms. Within IoT platforms, dealing with various events and further processing of such are being done to drive or trigger other IoT devices with actuator capabilities by sending appropriate messages, and perform big data analysis for predicting the future behaviors of systems and nature, or obtaining meaningful patterns on the collected data set. Therefore, it can be said that sensor values are processed by an IoT platform to generate actuation messages that regulate the actuator devices' further operation.

Event management processes bring new challenges to the field, including severe security threats; thus, the entire system (as well as other interconnected systems) may be at risk of infiltration, eavesdropping, failures, or malfunctioning. Furthermore, modern IoT implementations with extended capabilities (driven by digital transformation efforts) may require larger network bandwidths. Data inaccuracy and hardware bottlenecks should also be mentioned as general problems in the IoT ecosystems. Among the aforementioned issues, infiltration attacks (of various kinds) stand at a different point from a data-oriented perspective. Because they, in theory, can cause manipulation or loss of the original data

that is being transferred through the network or stored in a database unit without even being detected for a long while, whereas malfunctioning is usually detectable, and eavesdropping does not cause loss of original data. Hence, such risks require specialized solutions that involve immutability and non-repudiation features.

Most large IoT deployments are all about scalability. Therefore, addressing security issues must be done via scalable and efficient solutions. Widespread adoption of fifth-generation cellular communication systems (5G) may help mitigate scalability and efficiency problems (Escolar, Alcaraz-Calero, Salva-Garcia, Bernabe & Wang, 2021). Nevertheless, the developers and the producers of IoT systems did not yet agree on universally applicable approaches and standards for security challenges. Hence, some costly countermeasures may be implemented on-demand, such as trusted third parties, private certificate authorities, and additional hardware layers.

1.1 Motivation

IoT and Blockchain are two of the major technologies for the world of today and the future. Also, both can work together to complete each other. IoT is responsible for gathering data. On the other hand, blockchain can be used to store these data. The mentioned concept is starting point of our motivation. In IoT ecosystem, security has been a challenge for all stakeholders since introduced of IoT. We know that store-side security can be solved by a Blockchain-based approach. Our motivation is based on idea of previous sentence. Our aim is to make proof of this concept in this thesis. Our other motivation is real-time event management with Blockchain for IoT. In IoT world, A lot of use case needs to take actuation based on predefined rules. Another completing point of Blockchain for IoT is smart contracts that are runnable short programming scripts. A lot of use cases are fulfilled by smart contracts. It shows another benefit of Blockchain on IoT. All these benefits make us motivated to work on this thesis.

1.2 Contribution

This thesis focuses on addressing portions of these security challenges by utilizing blockchain technology as a communication and data storage medium. That is to say, we explain and demonstrate how the risks of facing manipulation or loss of sensor data, as well as loss of control in the event management process caused by a potential infiltration attack, can be mitigated through the adoption of a blockchain base within a specially built IoT scheme. Additionally, this methodology provides means to mitigate data inaccuracy and inconsistency problems to a large extent. We demonstrated that implementing blockchain technologies can resolve or minimize these requirements at a reasonable cost. Integrating a blockchain-based communication and data storage mechanism to the existing IoT platforms provides secure two-way communication between an IoT platform and the corresponding IoT devices within the network. The blockchain implementation is experimentally realized by programming a unique smart contract to bring efficiency and scalability to the blockchain side. Our contributions can be listed as follows:

1. We introduce a unique IoT deployment, and a corresponding communication scheme empowered with an intermediary message queue that increases resilience against availability attacks, an IoT server that enables integration of a blockchain-based data storage, and a private blockchain that provides data integrity, immutability, and non-repudiation for the data to be stored.

2. To achieve the functionality, we have implemented a private Ethereum-based (Quorum) blockchain and developed a corresponding smart contract to store sensor data securely and handle event management altogether.

3. Through experiments, we have validated our claims regarding mentioned functionalities and conducted performance analyses, which show that the entire scheme is working at a reasonable speed and is, therefore, scalable.

The rest of this thesis is organized as follows: Chapter 2 evaluates the existing related studies, Chapter 3 provides preliminaries regarding the proposed solution, Chapter 4 explains the solution in detail, Chapter 5 introduces our tests and discusses the results, and finally, Chapter 6 concludes the thesis and discusses the future works.

During this study, we wanted to achieve a Blockchain-based IoT platform. Also, we aimed to examine performance metrics with smart-contract and without smart-contract. At end of this study, we generated a high-level architecture that is suitable to apply to many IoT applications. Message queue and micro-service architecture were applied to generate this flexible architecture. Another point, Message Queuing Telemetry Transport protocol is selected to address a common IoT protocol.

With this thesis, we show a new way for data immutability and event management in the IoT ecosystem. We defined this new approach to use in the IoT world. Also, security matters and event management can be handled by nature of blockchain. Our thesis aims to solve mentioned IoT challenges with this platform architecture for IoT systems that are suitable frequency.

1.3 Outline of the Thesis

The outline of this thesis is grouped in 6 different parts. First of them is "Introduction" which is the current chapter. It gives basic information about this thesis. Also, this part explains what we did and why we did it. This chapter introduces all this thesis. The second chapter which is "Related Works" shares general domain information. Another responsibility of this part reviews all similar studies and compares similar and different ways from others. Next one is "Preliminaries" which gives technical information about used technologies. The part is useful to learn technical base structure of the system. We hope that this part will be useful to other next studies in the future. Fourth part is "Proposed Scheme" which provides a high-level architecture and sequence diagram to explain main nodes in the system and two-way data flows. Fifth chapter includes test results on the

system. The part is useful to describe which domain is useful for this structure. Finally, we added our comments to last chapter to share our opinion about this thesis.

In the above side, we explained why did this thesis. Also, we gave information about our study motivation. The next section is a related literature review to find similar studies.

CHAPTER TWO
RELATED WORKS

Some studies are available in the literature, which introduce methods to let IoT systems benefit from blockchain technology in addressing some security issues. This section provides insight into these studies and briefly introduces their extents.

Ali, Vecchio, Pincheira, Dolui, Antonelli, & Rehmani (2019) find out some critical drawbacks and potential problems of centralized IoT platforms used within various IoT ecosystems, as follows:

- A successful denial of service (DoS) attack can make a centralized platform unusable and prevents its services for the clients. So that there is a single point of failure that affects the entire network.

- In an IoT ecosystem, a centralized platform stores users' (or devices') sensitive information whose leakage would be a significant risk. That is to say, the owners or producers of the information have very limited control over their data.

- When data is stored in a centralized cloud platform, the system faces potential accountability and traceability problems. Generally, such risks involve sensitive data to be removed or manipulated.

They also stated that these risks regarding the use of centralized systems are increasing as the IoT systems get more and more crowded, induced by the digital transformation trends. This growing number of IoT devices needs the scalability of connected platforms. Yet, building a centralized approach makes it complicated for scalable jobs. Another point, the study shows the following potential solution of decentralized platforms about the potential problems:

- A decentralized platform can remove the risk of a single-point-of-failure, different from a centralized approach. An additional advantage of the decentralized approach is that it does not need an extra third-party application for controlling to store user data.

- In the blockchain network, checking user identity is already integrated into the system. Also, the blockchain approach verifies the data that are sent by platform clients.
- For accountability and traceability, the blockchain can store event logs and data immutably. This feature provides high benefits to the IoT ecosystem against related potential problems.

- The smart contract is the most useful case for programmable logic to provide access control, confidentiality, and authentication to increase the security of IoT approaches with blockchain.

This study was a good guideline for our work on a blockchain-based security study. Our difference is that providing a specific use case implementation and showing its results. Our aim is to develop a more practical based study.

Another study, Kathayayani, Murthy, & Naik (2020, May), suggested a method to provide a tamper resistance system for IoT devices by using the Hyperledger Fabric blockchain, considering a smart power meter scenario. The authors have implemented a prototype platform where the IoT nodes produce some data and send them to a web server, whereas the web server posts the data to the connected blockchain. The peers of the Hyperledger Fabric network are made responsible for storing the smart contracts and ledgers. When a smart contract enforces a transaction, the corresponding transaction(s) are kept in the ledgers. During a block process transaction, the peers are responsible for running the predefined smart contracts and sending responses to the client applications, if any. Since all peers keep a local copy of the ledger, every peer communicates with each

other to check and sustain the blockchain. Likewise, Hang & Kim (2019, May) introduced a novel decentralized platform to handle identity and data security challenges caused by cyber-attacks and analyze the performance issues regarding their platform. The study focused on business logic applications that use smart contracts to conduct event management activities. Furthermore, it provided a solution for the constrained devices to efficiently join (and benefit from) the blockchain network. Both studies are near to our study. But used technologies and base aim are different from each other. We want to develop a platform for the specific use case. Our platform provides data immutability and real-time event management. Also, we used Ethereum and a Docker-based structure.

Moudoud, Cherkaoui & Khoukhi (2019) focused on integrating a blockchain structure to the supply chain management processes to increase the overall security of the systems. The paper suggested a new lightweight consensus algorithm approach to satisfy the speed requirements of IoT deployments. Their consensus algorithm, called \Lightweight Consensus for IoT" is important since it is a solid proof-of-concept regarding blockchain's applicability within IoT systems. Although we followed similar approaches to make blockchain adoption more efficient in an IoT deployment, we considered different methodologies for the development phases; so that we focus on securing the IoT platform via blockchain, and we aim to provide secure data storage on the platform-side to let constrained devices to work flawlessly over MQ telemetry transport (MQTT) protocol - the MQ part of the name is not an abbreviation per the latest official documentation). Another difference is that we preferred smart contracts to contain rules and conditions to handle event management (for controlling actuator devices), whereas the related study uses the contracts for providing transparency and efficiency and for managing the operations of the stakeholders. Ali, Ali, Musa & Zahrani (2018) aimed to transfer and store IoT (e.g., sensor) data on a blockchain using networked devices equipped with security-aware smart contracts. The study mentions IoT-specific challenges caused by delays and overheads within the blockchain. The authors preferred to use Hyperledger Fabric as the blockchain ledger with the motivation of establishing secure communication between the IoT nodes and within the decentralized ledger. Their motivation is parallel to ours, but

their field of application is exclusively different. Throughout the study, the authors elaborated on the smart home applications and potential use cases within that field. Lastly, they presented the results of their comparative analysis, including their proposal and the Quorum blockchain as an alternative.

Huang, Bian, Li, Zhao & Shi (2019) worked on smart contract security as well. Their motivation was to uncover the potential security gaps within Hyperledger Fabric and Ethereum smart contracts. The study discusses smart contracts as specific software products that can be secured to some extent using predefined software life-cycle models. The paper compares the smart contract mechanisms of Ethereum and Hyperledger Fabric blockchains from a security perspective and considers three different classes of vulnerabilities, particularly for smart contracts, namely issues of specific development languages, features of specific blockchain platforms, and misunderstandings of common practices. They have noted several security problems, including the "self-destruct" bug of the Solidity language. This is caused by a specific bug within the Ethereum platform. More on the languages, they stated that non- determinism-related issues exist in the general-purpose programming language Go used in Hyperledger Fabric. However, this bug does not exist in Solidity and Ethereum thereof. An example of features of specific blockchain platform-based issues is the transaction order dependence (when using Gas-based accounting operations), as revealed. It is, however, not valid for Hyperledger Fabric. Likewise, Ethereum is not vulnerable to \read your write" attacks. Problems involving misunderstanding of common practices are threatening for both platforms. This study gives information about smart contract security. Based on this study, we don't want to face non-determinism-related issues. Therefore, Ethereum is more suitable for us. This study works on all possible security gaps in smart contract. But we introduce a new usage area for smart contracts in IoT. We can handle real-time event management with smart contracts in the IoT ecosystem.

Baliga, Subhod, Kamat & Chatterjee (2018) analyzed security features of Quorum, a unique open source blockchain platform. A Quorum node is divided into two sub-

modules: the transaction manager and the enclave. The transaction manager is responsible for managing the private transactions with cooperation. On the other hand, the enclave performs encryption and decryption operations related to blockchain transmissions. The Quorum platform supports two different consensus algorithms: Raft (Ongaro & Ousterhout, 2014), and Istanbul Byzantine Fault Tolerance (IBFT) (Moniz, 2020). The Raft consensus algorithm has been found to have small advantages over IBFT, such as better performance at higher transaction rates (i.e., higher than 1650 transactions per second). Therefore, we preferred to use Raft for our IoT setup, where Quorum is utilized. Xu, Zhang, Liu & Cao (2020) have elaborated the security of wireless networks that utilize a Raft-based consensus mechanism against jamming attacks. The study suggests a Raft-based wireless blockchain network divided into two different parts, namely, a client and a wireless blockchain consensus network. In Raft-based blockchain consensus networks, any node can be a leader or a follower. The selected leader sends downlink messages to its followers. Whenever a follower receives a related message from its leader, it makes a confirmation operation and sends feedback to the corresponding leader. After these steps, the leader node collects followers' feedback. This collection process aims to provide most of the followers' responses. The study also makes analyses on performance and success rates. All above side studies works on specific basic security concept for Blockchain. Our difference is to cover all potential problems in a specific IoT use case with a blockchain-based approach. Also, we share an end-to-end solution for this related IoT use case.

Mihelj, Zhang, Kos & Sedlar (2019, July) proposed a system that detects (and responds to) traffic incidents using some crowd- sourced data that are generated by users or preinstalled monitoring devices. They have utilized a "Turing Complete" blockchain platform as a means for data storage. This decentralized platform provides resilience to institutional data manipulation and supposedly provides reliability. In the study, researchers mentioned that the event detection platform was tested through well-defined simulations and using data that are generated from users. They implemented an event detection and resource reputation assessment system on a smart contract running over an

Ethereum instance. Their system architecture and motivation were inspiring for our study, while our scheme provides modularity and generalizability with a focus on IoT scenarios.

Khare, Ashraf, Yousuf & Rashid (2022, January) gave information about Blockchain Applications in IoT with their study. They mentioned that in the next 20 years, IoT applications will face security issues. They have the same idea about IoT security as us. These security requirements and reliability needs can be covered by Blockchain technologies. They also gave information on usage areas of Blockchain with IoT such as Home Automation, Public Well-Being Services, and Electricity and Capital Control. This study gives information about all possible IoT working areas in Blockchain. This study may be guided for us to implement a new Blockchain-based IoT use case. Our difference, we focus on more practical-based study. We provide a high-level architecture for Blockchain-based IoT.

Alabdali (2022) worked on a smart system which is based on IoT and Blockchain. The system is working for a special use-case to control milk stock in a fridge. Also, this study explains hardware and system structure details. Our main ideas are similar. Related study stores data in Blockchain as us. But this study includes hardware and machine learning points. Our differences are data mutability and event management on smart contracts. The study solves a real problem in the IoT world. But we give a more common structure to adopt new IoT use cases. Also, our smart contract usage idea is another main difference from this study.

Saputhanthri, Alwis & Liyanage (2021) have similar ideas about usage of blockchain in IoT domain. They made theoretical research about potential use cases and solutions. We have similar ideas, but our main difference is based on study context. They prefer just made research on this topic, we also realized our idea and made tests on this solution. In this study, the Internet of Vehicles and Content Trading are mentioned as working areas. We think that this comment may be useful for future working areas. Also, smart contract

idea of this study and our usage of smart contracts are matched. Another common point is that our and their studies suggest data immutability with blockchain.

Bouras, Lu, Dhelim & Ning (2021) worked on a lightweight approach to manage identity. They presented a decentralized system for this IoT requirement in the study. Also, they calculated latency and throughput for all transactions like us. Their structure presents trusted automatic registration, multi-factor authentication, and identity lookup mechanism. Our difference is based on addressing different problems. Also, they used a different blockchain platform which is Hyperledger Fabric.

Faisal & Reaz (2022) focused on remote patient monitoring concept with blockchain in IoT ecosystem. They want to take advantage of decentralized structure of blockchain to cover problems that are related to nature of centralized system. Aim of this study presents a structure to monitoring healthcare IoT sensors on patients to provide secure and continuously monitoring with Blockchain. Also, they mentioned advantages of smart contracts for remote patient monitoring. Main problem-solving approach of this study is near to us. But our study provides a more generic structure for IoT world.

Kaur & Ali (2022) gave information and fundamentals about IoT and Blockchain. They mentioned security details about blockchain-based IoT. Also, this study includes reviews of a lot of blockchain platforms. The study may be as a guideline to search for possible solution approaches. Despite they worked in same domain as us, we focused on more practical study.

In this chapter, we learned what previous studies made. Related information was useful to define our structure. This section provides an opportunity to compare our similar and different ways from others. We go on to study technical information in below chapter.

CHAPTER THERE
PRELIMINARIES

This section introduces our IoT model and the system component preferences. Our IoT model supports transmission control protocol (TCP)-based MQTT protocols. In addition, all our nodes in the system are packed by Docker as a container. In storage side and smart contract, we use Ethereum Blockchain network. On the other hand, we mentioned consensus algorithm selection in this chapter. Also, below side includes how a IoT message is operated by the system. You can find all technologies and approaches details under related topics in this section.

3.1 IoT Model

The following assumptions define our understanding of an IoT ecosystem:

- An IoT ecosystem may consist of (but not limited to) any combination of the following devices: clients (e.g., sensors, actuators, or hybrid devices), special-purpose routers (e.g., brokers), gateways, message queuing interfaces, servers, databases, monitoring devices, etc.

- All nodes in the IoT ecosystem implement the Internet protocol (IP) suite to some extent. The clients (e.g., sensors, actuators) can run the MQTT protocol, whereas an IoT server supports transmission control protocol and user datagram protocol (UDP), as well as transport layer security (TLS).

- A private blockchain implementation runs in real-time is integrated into the IoT network to provide decentralized data storage and event management via smart contracts.

3.2 System Components

This section provides preliminary information regarding the essential components of the proposed system. System Components has got technical details about system fundamentals. This chapter shares basic technologies step by step. First of them is 3.2.1 Virtual Environment section which is related to Docker. This sub-chapter explains base of micro-service structure in the system. Next section is 3.2.2 Blockchain gives information on why we chose Ethereum. In addition, this part explains basic structure of blockchain. The third section describes choosing and reason for 3.2.3 Consensus Algorithm. The section describes which node should write what information to the distributed ledger in the Blockchain system. 3.2.4 IoT Server explains our main node in the system and mentions its main responsibilities Next, 3.2.5 Message Queuing Interface is related to Kafka which provides synchronization between broker and IoT Server nodes. The chapter gives more detail about related structure. Last section is 3.2.6 Message Transmission Protocol mentions choosing a broker and background of choosing MQTT protocol.

3.2.1 Virtual Environment

Docker, a computer, and operating system virtualization solution is used to implement the IoT nodes and other network elements “virtually”. Docker allows building and packing applications with necessary dependencies and running multiple instances virtually independently on a host computer (by sharing some resources among the programs). It lets packed applications run in isolated environments called containers, which enables running the instances in parallel, thus providing scalability. This approach decreases the complexity of such studies (Docker, 2020). Developing a platform, however, is not only a programming challenge but also a question of building an architecture. Docker helps to implement the intended architecture easily. In our solution, all IoT nodes (as well as other system units) are created as a Docker container, therefore,"dockerized".

3.2.2 Blockchain

Some strict requirements of this project under development limited our choices regarding the adoption of a suitable blockchain ledger among tens of available options existing in the literature and the commercial domain. The first and most important requirement was the native support for smart contracts (to program the network). That limited our options to Ethereum, Hyperledger Fabric, Multichain, Lisk, Hyundai Digital Asset Currency (HDAC), and Quorum (although there may be others, we cared for popularity and community support to some extent) (Reyna, Martin, Chen, Soler, & Diaz, 2018). Another design decision was using Raft as the consensus protocol as it is IoT-compatible (justified in the next sub-section). Further, although not mandatory, blockchain technology would better have a specific cryptocurrency or token in circulation, which enables real-time live tests and guarantees continuous operation/development. Finally, transaction processing delays and throughputs are of crucial importance. Hence, Quorum appears as a viable option with necessary features and decent performance (Dabbagh, Choo, Beheshti, Tahir, & Safa, 2021).

Ethereum is defined as transaction state machine. The state machine starts with Genesis Block and every new transaction occur an operation on this machine (Wood, 2021). Also, Ethereum is a blockchain application that is generated from blocks. The block includes two main parts that are header and transactions. The header part is responsible for keeping information about previous block and block transactions. This concept is needed for immutability for block. It is base of security about blockchain structure. Figure 3.1 is an illustration of mentioned concept.

Above part gives information about Ethereum data storage. But Ethereum is not made from only data storage. Also, it includes smart contract part. Smart contract can be defined as a small program that is trigged with a transaction. The program scripts are useful to use business logic over transactions. Commonly, Solidity is used to programing language for smart contract. But it is not an obligation in Ethereum. Other programing languages can

be selected for smart contracts. Because Ethereum run smart contracts as a byte's codes. If any programming language transforms needed bytes code format, it can be run on Ethereum. Another point about running smart contract is Gas. Gas is a payment operation for Ethereum miner to run your smart contract on Ethereum. Second term is Gas Price that means payment price for running smart contract. The price value is depending on complexity of related smart contract operation. Last term is Gas limit means maximum willing price to pay for running related smart contract (Zastring, 2021).

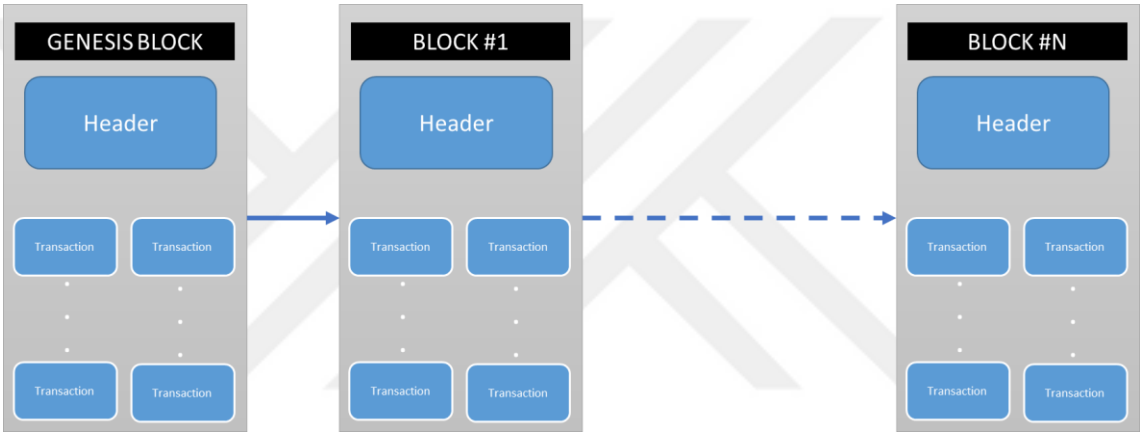

Figure 3.1 Diagram showing connection of blockchain.

Quorum is a distinct implementation of a private blockchain based on the well-known Ethereum platform (Consensys, 2020), which was developed by J. P. Morgan Chase and Ethereum Enterprise Alliance. Ethereum provides a public permissionless blockchain platform to implement decentralized applications in various domains. It also supports smart contracts, essentially decentralized applications running via blockchain, for different purposes in various domains. The idea behind our work is centered on a smart contract that can be used in event management to handle required IoT actuation(s) within the ecosystem. Quorum is preferred as the distributed ledger technology to benefit from an open-source Ethereum-based platform. It is a decent choice for storing IoT-related data within a decentralized structure and can easily be favored over a regular private Ethereum ledger for the following reasons:

- It enables management of network and peer permissions.

- It provides enhanced transaction and contract privacy.
- It supports a voting-based consensus mechanism.
- It offers better performance.

The development notes found in (51nodes, 2020) present a Raft-based Quorum blockchain network, implemented using Docker containers. In addition, the related containers run the Raft consensus algorithm without the transaction manager sub-module (e.g., Tessera), which means that the system cannot provide private (i.e., encrypted) transactions within the blockchain. On the other hand, in our proposed solution, the structure is more secure and robust since accessing the blockchain from outside of the original network is restricted by design. In addition, we considered that making non-private transactions in a blockchain increases the efficiency in terms of network transactions per second (TPS). This does not bring risks regarding privacy because the blockchain network cannot be reached by outsiders in our scenario.

3.2.3 Consensus Algorithm

A consensus algorithm is an essential element of blockchain-based systems. It regulates which node shall write what information to the distributed ledger. In our setup, Raft (Baliga, Subhod, Kamat & Chatterjee, 2018) (Ongaro & Ousterhout, 2014) is decided as the consensus algorithm for the integrated Quorum blockchain. There are actually not many options since Quorum only supports Raft and IBFT. Even if they were supported, many other algorithms (e.g., proof-of-work) would be infeasible for constrained IoT systems (Yao, Mai, Wang, Ji, Jiang & Qian, 2019). As reported in Baliga, Subhod, Kamat & Chatterjee (2018), Raft outperforms IBFT for higher transaction rates (which may be the case in live and crowded IoT ecosystems) in terms of latency and throughput.

Ongaro & Ousterhout (2014) mentions that a Raft based system includes many servers that are each them are candidate or leader or follower. At selection time, one of candidates is selected leader. All servers join this selection. If any candidate cannot be selected, this

selection repeated until select a new leader. Selected leader is responsible to handle request of system clients. Followers takes replications of transactions that are send by leader. If any client request reach to any follower without visiting leader, this request is redirected to leader. In Raft based systems, Remote Procedure Call (RPC) is used to communication. Request Vote and Append-Entries are these communications RPCs.

Additionally, Raft has proven compatibility with Docker containers as well. When Raft is in use, nodes in a Blockchain network delegate a leader among them, and all nodes are acknowledged regarding the leader node. Each new transaction is escalated by the leader and distributed to the nodes in a quasi-centralized manner. The escalated transaction is only written to the ledger if at least one more than half of the nodes confirm the transaction. The leader node broadcasts periodic heartbeat messages to sustain its leadership. When it leaves the network, another leader is delegated via election upon the detected timeout.

3.2.4 IoT Server

Within an IoT setup, an IoT server is a quasi-centralized computing and storage unit that is usually exempt from some constraints (e.g., battery shortage), limits the operability of smaller IoT nodes, and provides an expanded set of features to the rest of the IoT network it resides in. An IoT server is an optional component that may theoretically act as a data storage, a gateway, an intelligence unit, an automated controller (e.g., smart home systems), or a combination thereof. Our exemplary IoT server does not store data but mediates the communication between the Kafka message queue subsystem and the Ethereum-based ledger. The server is expected to collect necessary messages tagged with a label of the topics of interest from the Kafka message queue, and then it can convert these collected data to the format required for the Ethereum ledger. During this process, the IoT server utilizes the Actor model architecture. The concurrency in the system is provided by design without employing the complexity of threads. In the development, we used the Go-based proto-actor library (Asynkron, 2020). To translate the Kafka messages to blockchain-compatible format, the official Go implementation of the Ethereum protocol, shortly Go Ethereum (Ethereum, 2020) is used. Whenever a “send” transaction

is done, the server checks the smart contract addresses embedded in the message. If there exists a valid smart contract address in the message, the server sends the message to the smart contract to update the corresponding smart contract values. Later, it interacts with the smart contract to get the desired output.

Akka (2021) mention that problems for multi threads in traditional programming approach. Actor model is new programming model to handle the problems. First of them occurs when many threads try to call same method. Encapsulated object cannot give information about this case. In traditional programming approaches, Locks is potential solution for first problem. Also, Locks break multithreaded programming. Second problem occurs with shared memory on modern systems. Cores of CPU have got own cache to keep data for threads. A core cannot access to another core cache. It means if one thread is transferred to another core, related data on core cache should be transferred destination core cache. It is an extra cost for run related operation. Last problem is failure notification for main thread. In this case, caller thread cannot take failure notification when its delegated task is failed. Task is lost in the system. Above side is general problems for multitasking with traditional programming. Actor Model gives a new approach to solve these potential problems.

3.2.5 Message Queuing Interface

Apache Kafka is an open-source distributed event streaming platform that is also capable of serving as a message queuing system. It was implemented by Apache Software Foundation using Scala and Java languages. Like MQTT, it supports a publish/subscribe architecture, where publishing messages on designated topics and subscribing to these topics to get only relevant messages are possible for the attending nodes. In this platform, producers (e.g., sensors) can send their fresh messages upon production, and consumers (e.g., IoT servers) can listen to the messages published from these producers (Shaheen, 2017) in real-time. The main purpose of setting a Kafka message queue is to provide an intermediary layer between an IoT (i.e., MQTT) broker and the IoT server. Since Apache Kafka has built-in fault tolerance features and provides high performance, it increases the

overall resilience of the system against device (e.g., IoT server) failures and availability attacks (Kafka, 2020). For the development phase, we utilized the Confluent Kafka library (Confluent, 2020).

3.2.6 Message Transmission Protocol

MQ Telemetry Transport (MQTT) is a popular lightweight message transmission protocol that is very beneficial for IoT ecosystems (Collina, Corazza & Vanelli-Coralli, 2012). It also relies on a publish/subscribe architecture (i.e., brokers and clients) and is, therefore, natively compatible with Apache Kafka. All included MQTT clients can send and receive messages within a related channel, named a topic. The MQTT broker(s), on the other hand, is responsible for distributing messages coming from publisher clients to the subscriber clients (and to other devices, if any), depending on the topics they are interested in (Chien, Chen, Qiu, Liao, Hung, Lin, Kou, Chiang & Su ,2020). Mosca-JS, an MQTT brokerage system, is preferred for broker implementation due to its simplicity and ease of use (as well as potential performance benefits). It passes MQTT client messages to the Kafka messaging queue. Mosca-JS is a Node.js-based MQTT broker and allows easily scalable development (Mosca, 2020). For the client-side implementations, Mosquitto (for proof-of-concept) and Apache JMeter (for tests) are used.

In chapter three, we give technical information. Also, we explained why we use related technologies. The section provides a base for next section. Our high-level architecture was generated based on these technologies.

CHAPTER FOUR
PROPOSED SCHEME

The below part includes high-level design architecture as a scheme. The scheme's components, relations, and interactions are illustrated in Figure 4.1. As in the figure, all components are implemented in Docker containers. Docker-based nodes provide distributed and microservice for gathering IoT data. Nevertheless, other architectures may also be used since the tasks and responsibilities of each component are distinctly defined as they are stand-alone modules. With this approach, when any node is crashed, other nodes can go on to give their services. It is another advantage of our architecture. MQTT is a common standard IoT protocol in the IoT ecosystem. Therefore, we prefer MQTT protocol which is a publish and subscribe messaging queue. With this message queue, we can provide two-way communication between devices and the IoT platform.

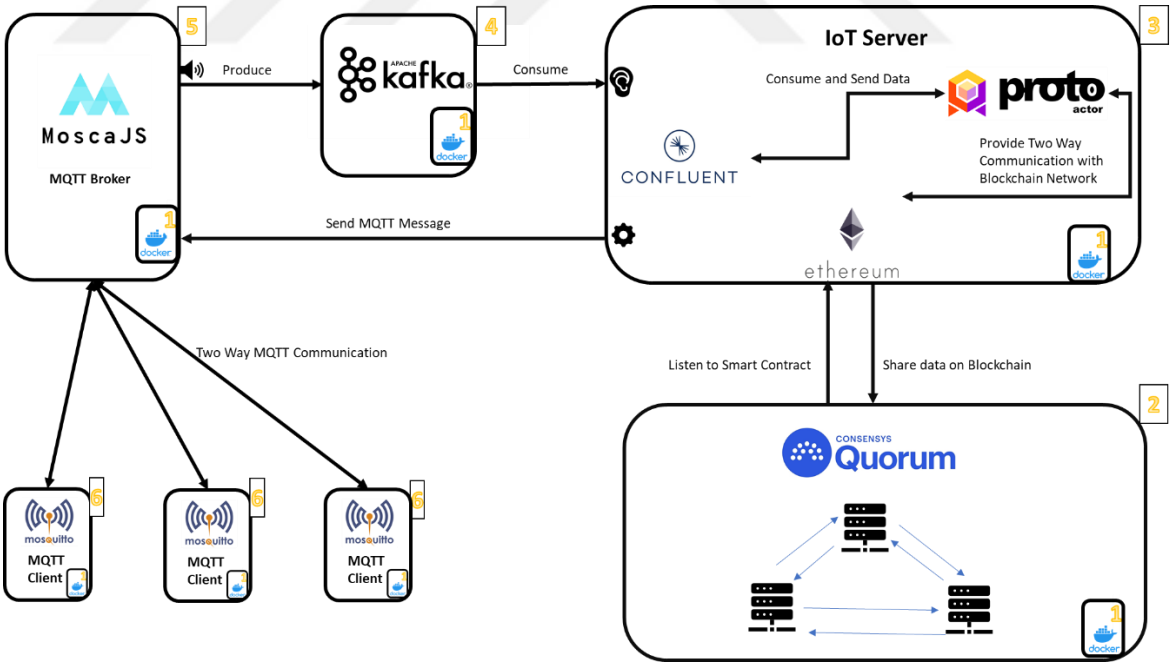

Figure 4.1 Diagram showing the system components, their interactions, data flows and the overall architecture.

4.1 MQTT Client

This actor in Figure 4.1 is out of the system nodes. All of the other nodes are part of the structure. Therefore, all of them are docker-based containers. Actually, our study does not focus on this actor in the figure. This is not part of our main structure. But it is needed to explain end-to-end data flow. For this reason, it is placed in Figure 4.1.

MQTT is a protocol for sharing messages on topics such as sensor values. An MQTT client (i.e., a sensor or actuator device that implements Mosquitto or JMeter) may transmit sensor or usage (i.e., analytics) data to the broker, and receive actuator duties or other commands, all over MQTT protocol. It means any IoT client publishes and subscribes to a topic on MQTT broker. It provides two-way communication between clients. As our above mentioned, it is so useful structure in IoT use cases instead of a classical server-client. In our structure sensor values are shared on a MQTT topic and commands of event management are taken over another topic.

4.2 MQTT Broker

MQ Telemetry Transport is a common and lightweight IoT messaging protocol in the IoT world. We mentioned previous parts. In this chapter, we focus on roles of our MQTT broker in the system. Another point, as we mentioned, it is a dockerized container to create a new one when it is needed.

An MQTT broker (that implements Mosca JS) is responsible for distributing the messages from the clients to other clients and the Kafka queue depending on the message topics and each devices' topic preferences. Topics define a specific channel between subscribers and publishers. When a publisher sends its message on a related topic, this message is received by subscribers which listen to this topic. Topics can have a multi-level structure. It also handles the communication in the opposite direction of the ow. It

means the platform can send a command message to devices for actuation based on predefined rules.

The above paragraph is described MQTT broker working principles. Our data flows are designed based on these principles. Another important note for this part, MQTT broker is our entry point of the system. It means system responsibilities start with this node.

4.3 Kafka

The message queue interface (that implements Kafka) transmits messages that it receives from the broker(s) to the IoT server (but not vice versa) over a produce/consume basis (like the publish/subscribe structure used in MQTT). Another responsibility of Kafka, Blockchain system, and MQTT broker have got different speeds on IoT devices. The difference needs to synchronization between both. Kafka handles this synchronization need. It is also responsible for handling, pre-processing, and temporarily keeping the data when necessary. Any IoT message is welcomed by MQTT broker. The broker is responsible from taking data from IoT devices. It means that MQTT broker is entry point of the system.

The scheme defines a microservices-based IoT solution which easily scalable with increasing nodes number. We care about this scalability to match basic IoT requirements. Also Using Kafka messaging queue and MQTT broker prevents messaging loss in the IoT Platform. The scheme can solve both potential problems.

Another illustration of the scheme is two-way communication which includes gathering IoT data and running accusation commands. The scheme shows following IoT data from device to blockchain system. Also, In the scheme, you can see how to send result of smart contract to devices.

4.4 IoT Server

An IoT server, apart from its other duties like locally storing data, post-processing data (e.g., for analytics purposes), or interacting with other systems, is responsible for establishing an interface between the rest of the IoT network and the blockchain network. The IoT server is implemented as two main functions using the Actor model (as in Figure 4.1). The first function (consuming part - Confluent - in the IoT server box) handles consumed messages from Kafka and pops them into the second function, namely blockchain client (Etherum part in the IoT server box), if necessary. The Ethereum part interacts with the blockchain ledger and produces messages to transmit to the broker accordingly. Such interaction includes (i) transmitting data (i.e., payload) to be stored in the ledger to Quorum and (ii) obtaining smart contract responses from Quorum. Lastly, the server transmits the received smart contract responses directly to the broker (Mosca JS) for client-side distribution. In this phase, the message queue (Kafka) is intentionally bypassed due to simplicity concerns. For the same reason, our implementation does not enforce other duties on the IoT server.

4.5 Quorum

Quorum, as an Ethereum-based decentralized ledger, has two important functions within the proposed scheme. Upon request, it (i) stores the IoT data (e.g., sensor data, usage statistics, etc.) consigned by the IoT server, and (ii) conducts event management (i.e., decision) activities via pre-programmed smart contracts that it runs. The way it stores data in the ledger (i.e., via decentralized hash chains) allows immutability and non-repudiation, as long as neither of the elements in the network is compromised during production or initial delivery of the data. Whenever some data is written into the ledger, it is no longer possible for a node or an intruder to alter or refuse it owing to the distribution of hashed copies among blockchain peers. By the way, a smart contract intervenes (only) the relevant incoming messages, and thus, makes a decision upon IoT client behaviors that are (i) triggering or starting a new process, (ii) ceasing an ongoing process, or (iii)

modifying an ongoing process. This decision is disseminated to the relevant IoT clients over the hierarchical data path 2 shown in Figure 4.1.

4.6 Data Flow

As a design decision concerning ease of use, implementation, and integrity, any message that will be sent from the IoT server to the blockchain should be formed as a JavaScript object notation (JSON) object containing relevant data fields (i.e., keys). There are three predefined types of such fields, namely Payload, Topic, and MessageContent, as shown below:

```
{
      "Payload": "<Data to store in the blockchain ledger>"
      "Topic": "<Topic to get a response from the smart contract>"
      "MessageContent": "<Smart contract values, address, and
topic>"
}
```

The field "Payload" includes the data (e.g., sensor data) to be stored in the decentralized ledger. "Topic" includes the topic label of the message. This topic is typically (but not mandatorily) the same as the MQTT topic. The "MessageContent" field contains the information relevant to updating the smart contracts and interacting with them to act or get any means of outcomes. Optionally, the keys can be set in different JSON messages instead of a single message. We defined mentioned structure for IoT messages. Because every IoT device generates its own device set. This is a huge amount of variation for the IoT platform. If we did not expect a defined message structure, the system cannot handle this variation. We preferred common message content (JSON) and protocol (MQTT).

Figure 4.2 illustrates the messaging procedures followed whenever the IoT server receives a message (as in the top-left box) from the Kafka queue. The figure defines the algorithmic workflow and the data paths via the arrows with sequence numbers. The first path stands for releasing an upcoming message originating from the Kafka queue on an arbitrary topic. The message is received (i.e., consumed) by the IoT server. The Kafka

consumer (i.e., IoT server) then sends the transaction records embedded in the new message to the Quorum to store them in the distributed ledger, as shown by the second arrow. After the transmission is completed, the IoT server checks the "MessageContent" field to find if there is a valid smart contract address. If there is such information in the message, the server updates smart contract values and interacts with the corresponding smart contract of Quorum as shown by the third path in Figure 4.2. As shown with the fourth path, if the "MessageContent" field has an additional "Publish Topic" key, the actor layer then interacts with the addressed smart contract in Quorum, gets a decisive order (for event management purposes) as in the fifth path, and sends the resulting order to the relevant IoT clients with "Publish Topic" eventually over MQTT protocol, as shown in the 1 sixth path. Finally, the MQTT clients listen to the topics that the server gives feedback on. The publish feature of the IoT server is developed using the Paho MQTT client library (Paho, 2020). For the clients, Mosquitto is preferred since it has a lightweight and suitable library, especially for listener roles (Mosquitto, 2020).

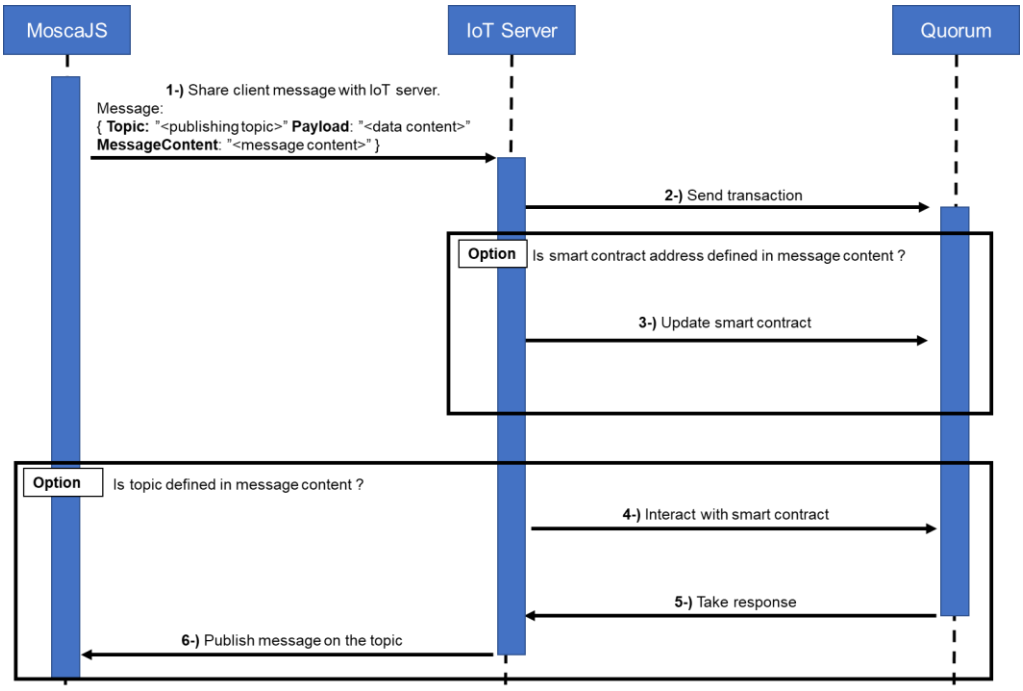

Figure 4.2 Message sequence chart showing the algorithmic workflow for handling new messages (e.g., sensor data).

The proposed Scheme is our high-level architecture. With this chapter, our study is completed as software. In the next section, we validate our structure.

CHAPTER FIVE
VALIDATION

Given the use case scenario introduced in the following subsection, the proposed scheme is tested thoroughly to check if it provides the intended functionalities and how fast it works.

5.1 Case Study

The proposed system architecture may not work flawlessly for particular IoT deployments, such as real-time industrial IoT systems. Because blockchain-based data storage may not allow transacting or processing data at very high speeds (due to the nature of blockchain used). Yet, under the assumptions given earlier, it is possible to generalize the proposed architecture for many IoT scenarios, especially where data push frequency is relatively lower, but the data is critically important (see the achieved data speeds given in this section). Accordingly, smart grid applications (e.g., electricity, water, etc.) and supply chain management are mostly suitable domains for the proposed architecture since the speed of data flow is relatively lower and the need for immutable secure storage is obvious. Edge and fog computing approaches may increase computing capabilities as well as speed (Elazhary, 2019). But we focus on unmanipulated data and secure smart contracts. Therefore, the computing approaches are out of the scope of this study.

We have considered a unique scenario with security-critical storage requirements (but requiring moderate speeds), that is, a system detecting the current risk of forest fires in deployment. This use case can easily be implemented with a wireless sensor network system, as stated in Son, Her & Jung-Gyu (2006). Its potential benefits were surveyed in Datta, Das, Kumar, Khushboo & Sinha (2020) It can also be integrated into an IoT ecosystem by extending the network to the Internet. It is by no means necessary to have a steady stream of sensor data with a known frequency. By employing our proposal, the risk estimation and decision-making activities can be carried out by a smart contract via the collected real-time sensor data. Son et al. Son, Her & Jung-Gyu (2006) have mentioned a

risk detection measure, called the danger index (DI), originally published by the Korea Forest Service. It is calculated by formula 1.

$$DI = 6.87 + (0.64 \times \_P) + (0.15 \times \_EH) + (1774.94 \div \_CS) \quad (1)$$

Where CS is the current solar radiation, P is the precipitation ratio, and EH is the effective humidity that is calculated as formula 2.

$$EH = (1 - r) \times (H_0 + (r \times H_1) + (r^2 \times H_2) + (r^3 \times H_3) + (r^4 \times H_4)) \quad (2)$$

Where H_{0-4} are five different humidity measurements and r is a coefficient of fixed value (0.7). All constants are predefined by the original publisher and are of no significance to the system under development. Yet, the calculated danger index (DI) is then classified using the following scale: $0 \leq DI < 60$ indicates a Low-Risk state, $60 \leq DI < 80$ indicates a High-Risk state, and $80 \leq DI < 100$ indicates an Extreme Risk state. The terms and abbreviations used in the validation section are collected in Table 5.1.

Table 5.1 Terms & Definitions

Abbreviation	**Definition**	**Remarks**
CS	Current solar radiation	-
DI	Danger index	-
EH	Effective humidity	-
H_{0-4}	Humidity values	Assumed 5 distinct values
P	Precipitation ratio	-
r	Coefficient	Set to 0.7 in example
AM	Action message	IoT message type
RM	Registry message	IoT message type

5.2 Implementation

In the simulation environment, we have implemented the features of the case study on our components. The clients provide *CS*, *P*, and H_{0-4}values to the broker, and eventually, to the IoT server. The values are preset for simulation purposes since their actual values

do not matter. Calculations of EH in Equation 2 and DI in Equation 1 are done by the IoT server using the smart contract. The smart contract is implemented on the ledger, using Solidity language. The code of our implementation is given in below. We used the OpenZeppelin math library (OpenZeppelin, 2020) for Solidity to conduct arithmetic operations.

```
pragma solidity ^0.7.5;
import "openzeppelin-contracts/contracts/math/SafeMath.sol";

contract Inbox {
      uint256 Y;
      uint256 r= uint256(70);
      uint256 P;
      uint256 CS;
      uint256 H0HR;
      uint256 EF;

      function Set(uint256 _CS, uint256 _P, uint256 _H0, uint256 _H1,
      uint256 _H2, uint256 _H3, uint256 _H4) view public returns(uint256)
      {

            P=       SafeMath.div(SafeMath.mul(uint256(64),       _P),
      uint256(10**2));

            CS= SafeMath.div(uint256(17749400), _CS);

            H0HR=SafeMath.add(SafeMath.add(SafeMath.add(SafeMath.add(_H
            0,SafeMath.div(SafeMath.mul(r,_H1),uint256(10**2))),SafeMat
            h.div(SafeMath.mul(r**2,_H2),uint256(10**4))),SafeMath.div(
            SafeMath.mul(r**3,_H3),uint256(10**6))),SafeMath.div(SafeMa
            th.mul(r**4, _H4), uint256(10**8)));

            EF=SafeMath.div(SafeMath.mul(uint256(15),SafeMath.mul(uint2
            56(30), H0HR)), uint256(10**4));

            Y=SafeMath.div(SafeMath.add(SafeMath.add(SafeMath.add(uint2
            56(687),P),EF),CS), uint256(10**2));

            return Y;
      }

      function Get() view public returns(uint256) {
            return Y;
      }
}
```

As a remark, Solidity 0.7.5 version causes some implementation challenges since decimal multiplications are problematic. Our approach was multiplying the decimal numbers with powers of 10 to eliminate fractions and then dividing the results by the same factor to normalize per to the range of the danger index. The results are also summarized in Table 5.2. All the graphs reflect the confidence interval that was preset at 95% and the coefficient of determination (r-squared), as hinted in Seneviratne, Wijesekara & Leung, 2020.

Table 5.2 Summary of test results

Test	**Speed (TPS)**	**Time for 1000 Threads (s)**	**Avg. Latency (ms)**
RM	5	200	22.727
RM	10	100	17.866
RM	12.5	80	19.597
AM	4*	250	53.089
* 4 in clients, 8 in the blockchain.			

5.3 Testbed & Configuration

For testing purposes, the MQTT clients are simulated via Apache JMeter (2020). It allows the making of load tests and is used as a simulation tool (Paz & Bernardino, 2017). More, we used the MQTT plugin to send MQTT messages from Apache JMeter instances. The host hardware has the following properties: Windows 10 (64-bit) operating system, 32 GB of RAM, and an Intel i7 processor.

The time measurements are repeated with two different message types, namely a registry message (RM) and an action message (AM). A registry message merely contains some information (e.g., sensor data) to be recorded on the ledger, whereas an action message includes an additional smart contract configuration (e.g., address, parameters, etc.) as well as some information to be recorded. The action message enforces the smart contract (pointed in the address) to run, make some decisions, and escalate an action. This triggering mechanism comes at an expected cost of larger messages and higher latency. In

either case, the messages are sent over MQTT version 3.1 with quality of service (QoS) level 2 and with merely one topic. The process does not require retaining the messages. During the test runs, we calculated the IoT server's performance to assess the efficiency, usability, and scalability of the proposed scheme. Therefore, each MQTT message has a timestamp of its creation time. Additionally, when the blockchain operations are done, a new timestamp has been generated. The difference between those two indicates the completion time of the required actions implied by each MQTT message. This time measurement applies to all messages sent with different rates of transactions per second. Different TPS rates are achieved by scheduling 1000 concurrent threads accordingly.

5.4 Test of Registry Messages

The test is repeated three times, each at different rates, consecutively at 5 TPS, 10 TPS, and 12.5 TPS. Each case (i.e., speed) has been run with 1000 concurrent threads (representing the clients). Total run times of each request are added in Appendices chapter. You can access test details under Table of 4 TPS, Table of 5 TPS, Table of 10 TPS, and Table of 12.5 TPS.

Below side, Figure 5.1 and 5.2 illustrates 5 TPS result for 1000 request. When we check related tables, we can see ordered results in both figures. We cannot mention special restriction comments for these figures. In Figure 5.1, you can see that most of requests are in prediction intervals. Also, you can see duration details of each request in Table A.1.

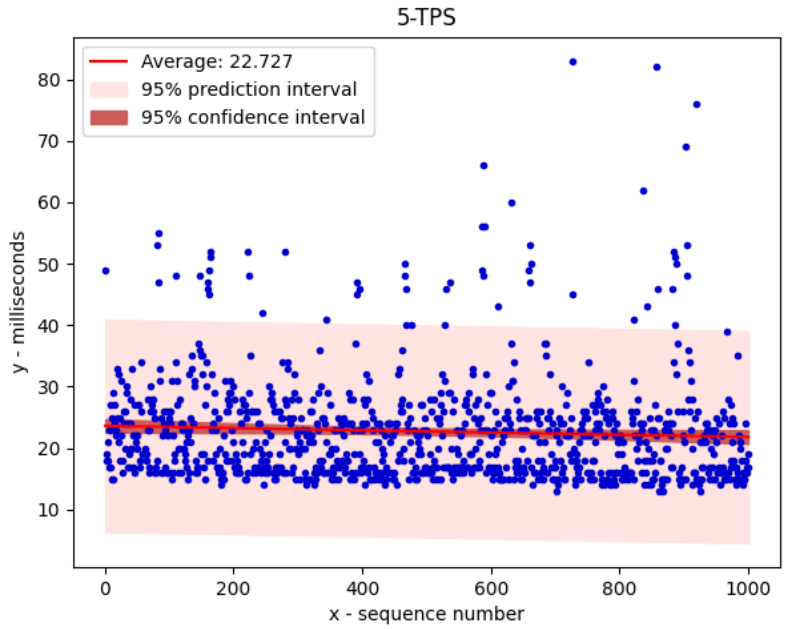

Figure 5.1 Registry messages (RM) @ 5 TPS.

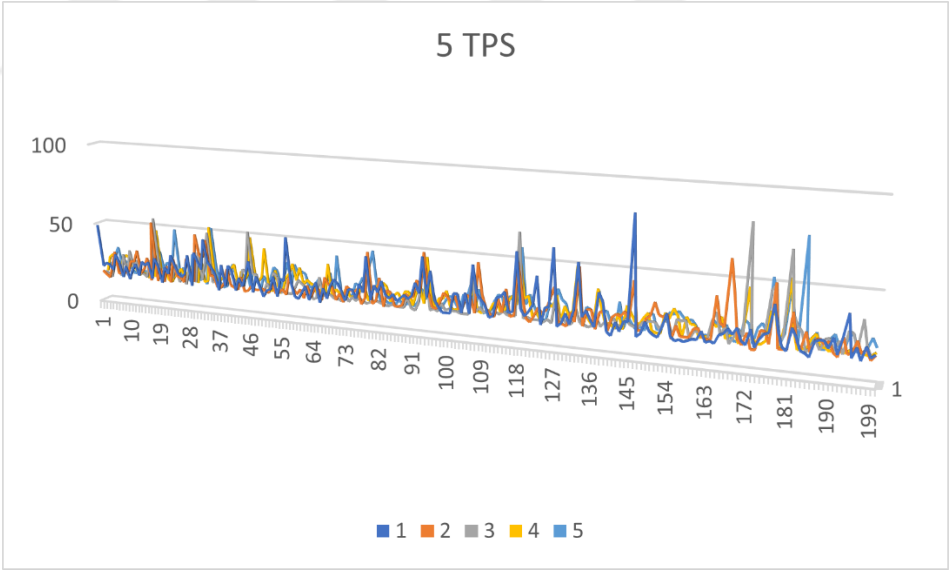

Figure 5.2 Registry messages (RM) 3D Line Chart @ 5 TPS.

Next figures, Figure 5.3 and Figure 5.4 represent 10 TPS 1000 requests which be given details in Table A.2. We can see that requests are more orderly than 5 TPS. Like Figure 5.1, most requests are placed in prediction intervals. But we cannot mention these

comparison comments again for these tests. Because all of prediction intervals are the same.

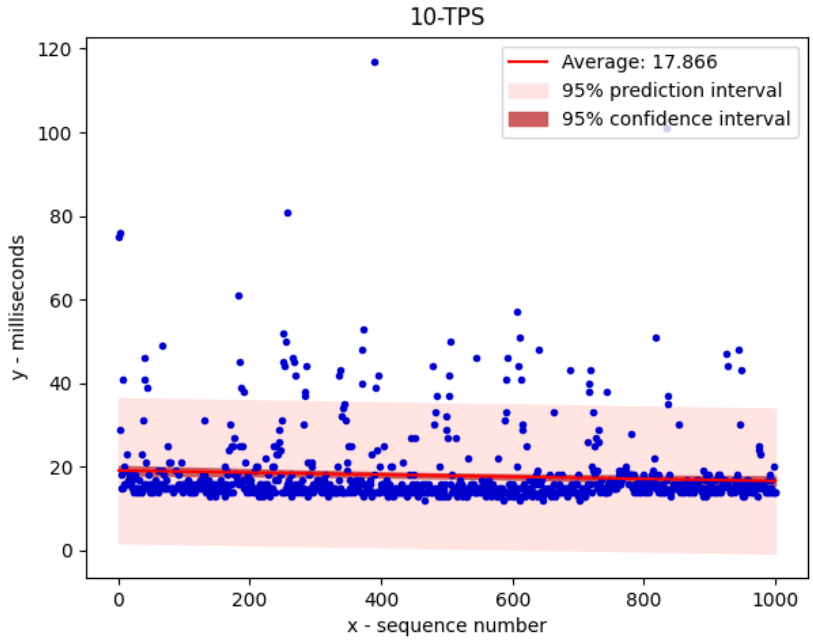

Figure 5.3 Registry messages (RM) @ 10 TPS.

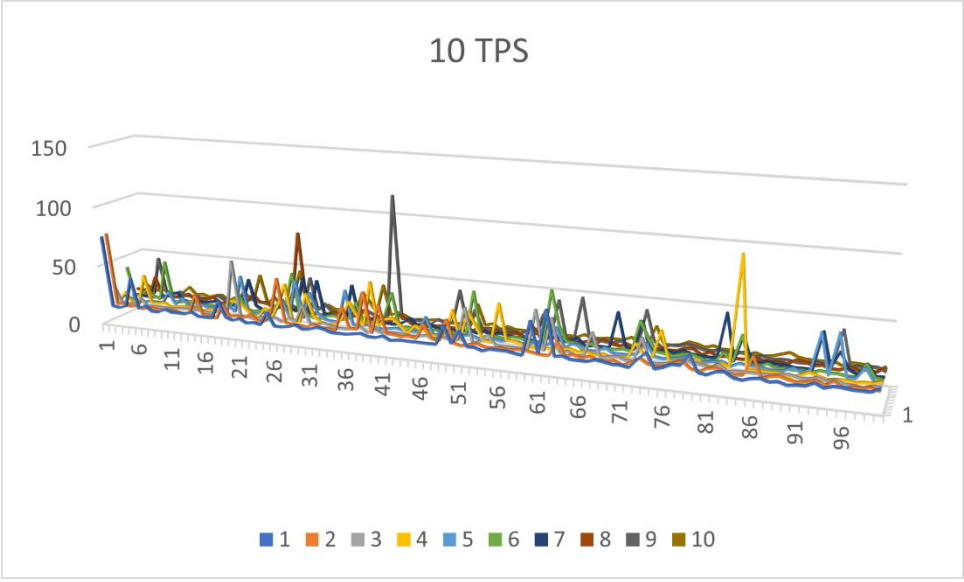

Figure 5.4 Registry messages (RM) 3D Line Chart @ 10 TPS.

Figures 5.5 and 5.6 represent 12.5 TPS for 1000 requests. In addition, Table A.3 represents a running time for each as a table. Figures are same as previous results of test figures. Another way, prediction intervals that are the most important point are the same as previous ones.

All regarding each run, the individual measurements and the mean averages are depicted in Figures 5.1, 5.2, 5.3, 5.4, 5.5, and 5.6 in the given order. All test result for prediction intervals is same in registry messages. We cannot compare prediction intervals for these test results. But average response times are different for each different TPS. Despite they had no big differences, we can say that 10 TPS gives the best average duration with 17.866 milliseconds for registry message tests. But the system can handle more TPS without failure. Also, we can see both from our test results.

The overall result, all TPS can be run without any problem. All registry message tests are so similar. Therefore, we can say that our recognized system is suitable for all these concepts. But we cannot ignore that if size of message payload is increased, these test results are not meaningful. It means that these test results are valid with this setup and this payload size.

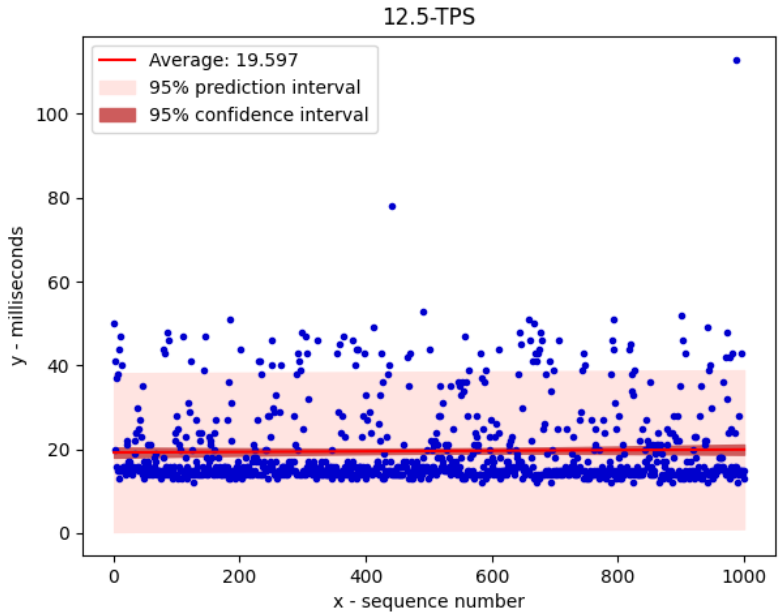

Figure 5.5 Registry messages (RM) @ 12.5 TPS.

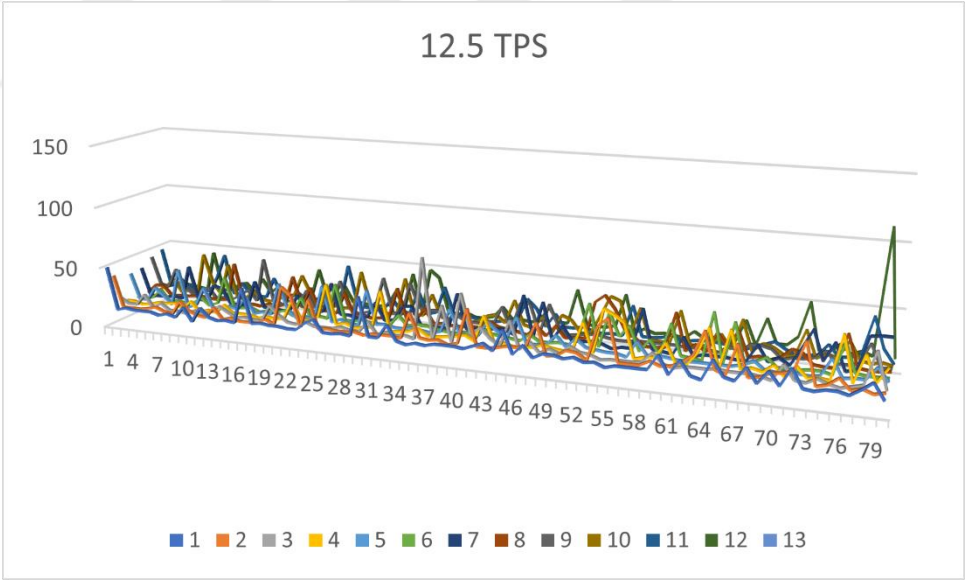

Figure 5.6 Registry messages (RM) 3D Line Chart @ 12.5 TPS.

5.5 Test of Action Messages

The test is only conducted once at a fixed speed of 4 TPS. Nevertheless, this is the speed set at the client-side. However, in our scheme, triggering a smart contract enforces the smart contract to respond to the clients. Hence, the speed of the blockchain is 8 TPS

due to this request-and-response style of communication. During the test, the smart contract is triggered only once. On the other hand, the size of the messages sent within the two tests are different; in RM, the size of a packet is 108 bytes, whereas it becomes 159 bytes in AM. As in RM, 1000 MQTT threads were run concurrently, and an average is found out. The individual results are depicted in Figures 5.7 and 5.8. These 1000 requests are made in a table in Table A.4.

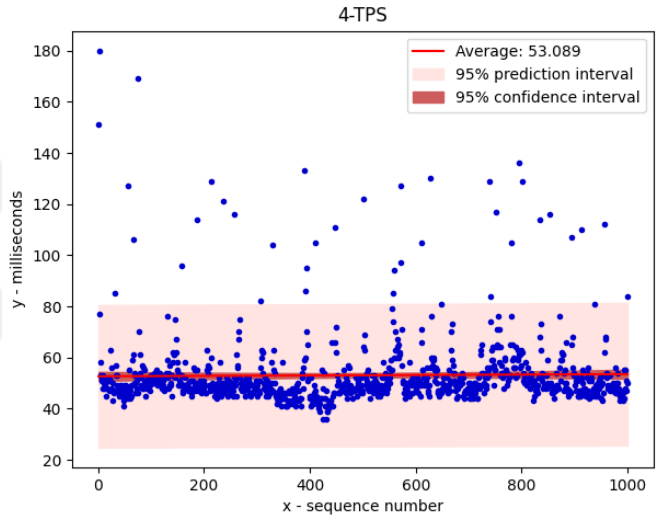

Figure 5.7 Action messages (AM) @ 4 TPS.

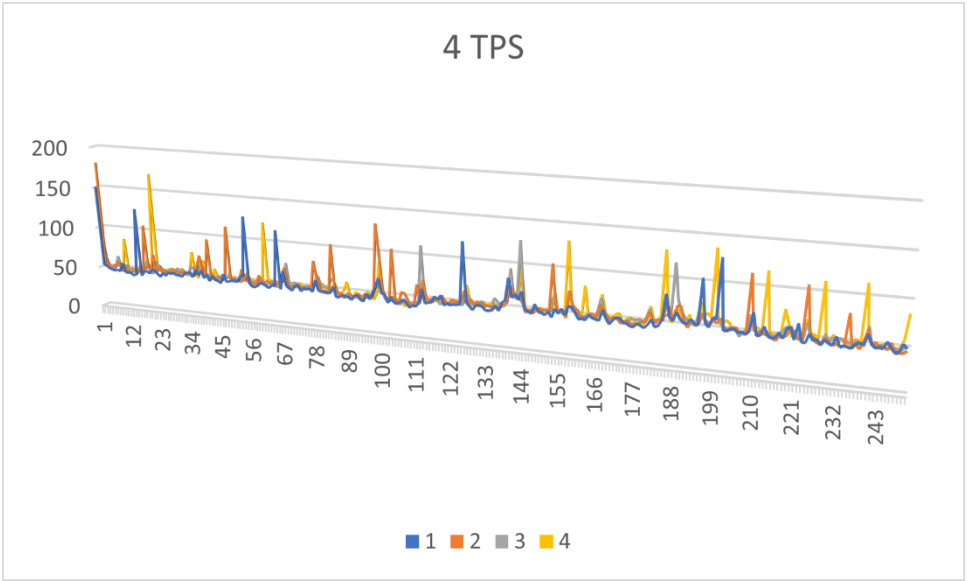

Figure 5.8 Action messages (AM) 3D Line Chart @ 4 TPS.

A proof-of-concept of the proposed scheme 1 is, thus, developed following a specific use case. It was made to run flawlessly and validated thereafter. As inferred from the results, the registry messages can be processed significantly (i.e., nearly three times) faster than the action messages (which also means a proportionally lower average latency) because the action messages include two subsequent operations on the blockchain (i.e., Quorum). Furthermore, the size of action messages is (nearly 1.5 times) bigger than that of registry messages. Therefore, increasing the size of the transmitted messages and the complexity of the smart contract decreases the system performance dramatically. As a side note, although the running times are purely acceptable for an IoT server, some performance improvements may be necessary for the blockchain implementation to make the scheme more scalable. Ultimately, the results indicate that if the collected data is sensitive and the data collection frequency is low, then the data storage and event management activities can easily be carried out via the proposed scheme.

5.6 QoS Assessment

Quality of service (QoS) is a design and operation paradigm in information and communication technologies that involves defining and tracking a set of (most likely quantitative) measures and characteristics of a serving system. These measures help quantify how well the system-of-interest works and what actions should be taken to make it serve better, if any.

The proposed system in this thesis is a Docker-powered microservices application that can easily be deployed online, preferably at a cloud server, to be offered as a public or private service. The client IoT devices shall use the cloud based MQTT broker for sending their data to the blockchain application. Therefore, the entire system is a software as a service (SaaS), where common QoS metrics used for SaaS are also applicable. Ezenwoke, Daramola & Adigun (2018) described some quantitative and qualitative attributes about QoS concerning cloud services. These attributes include response time, cost, availability,

usability, security management, and flexibility. Considering our research, the following statements can be reached regarding the abovesaid QoS attributes of our proposal:

- Response time: The delay between a request (e.g., a data push) and the actual time it completes is the response time. Although it is application-dependent, increasing the capacity of resources (e.g., processing power, bandwidth, 1 memory, etc.) of the cloud platform will decrease the response time (Schaffer, Angelo & Salzer, 2019). The average response time of the Ethereum network, which is around 12 to 14 seconds, is a justifiable reference point for QoS determination (Ethereum-Blocks, 2020). We can guarantee even lower response times for the proposed setup, thanks to the consensus protocol chosen, namely Raft, which works faster. Further, as discussed in Section 3.2 that Raft works even faster without the transaction manager sub- module. These performance improvements are carefully considered for the proposed setup.

- Cost: The cost in a cloud-based service is a function of the allocated resources. There is a tradeoff between the cost and the resources. Requirements and feasibility change per application. More discussion can be found later in Section 5.7.

- Availability: The system's uptime in which it is active and responsive defines the availability. IoT devices generate data on a continuous basis. Therefore, the telco-grade availability rate of %99.99 (Benefield, 2009) can be considered to represent the high availability requirements. Throughout our tests, no outage and no packet loss have been recorded.

The proposed system may be implemented to provide different QoS standards, including the strictest industry considerations.

5.7 Discussion of Complexity

A brief complexity analysis concerning our study's exemplary use case (implemented within the smart contract) is given. We utilized the Big-O notation to find an execution time for the related algorithm irrespective of processor or programming language selection. The smart contract implementation algorithm executes two functions, namely, the effective humidity (EH) and the danger index (DI).

Within *EH*, the atomic base operation is the multiplication (as the addition is clearly dominated by the multiplication) and the input is the number of data sources (e.g., sensor nodes). As the number of such sources increases, multiplications increase quadratically since they exist among *H* values and the power terms of r . Accordingly, the correlation between the node count (n) and the multiplication count (f(n)) can be stated as f(n) = $0.5n^2$- $0.5n$ +1. So that, *O*(f(n)) leads to O(n^2) . On the other hand, the computational complexity of *DI* is trivial and is determined 1 by the *EH* function. Nevertheless, this analysis is only valid for our specific use case. A broader analysis of the proposed architecture can be obtained by evaluating the message complexity.

Again, the message complexity is strictly dependent on the number of nodes involved in the IoT deployment (n). Moreover, it also depends on the number of processing nodes within the blockchain network (m). During the log replication process, the Raft consensus algorithm creates a single two-way path between the leader node and each of the clients, which results in linearity in terms of message complexity. The (occasional) leader election phases, however, requires quadratic messaging between all the node pairs. But the dominance of this phase depends on the predefined leader election periods and can be negligible for longer periods. When sufficiently longer periods are considered, the overall message complexity of the system becomes *O*(nm) ; otherwise, *O*(nm^2) is obtained.

The above chapter gave information on our validation test concepts and their results. In the next section, we mentioned our study conclusions.

CHAPTER SIX
CONCLUSION

This thesis first discusses data integrity, immutability, and non-repudiation issues, as well as their adverse effects on event management processes within IoT ecosystems organized in a centralized way. Briefly, these issues arise with a successful infiltration attempt to the servers and database(s) of the system. An attacker is then able to alter (or even wipe out) the collected data and may interfere with event-response mechanisms resulting in malfunctioning of the entire system. Integrating a blockchain mechanism (i.e., a decentralized ledger) as a data storage and processing unit is found to be helpful in addressing these challenges.

Earlier works involving blockchain integration to IoT systems in the literature, to the best of our knowledge, either lack implementational details, or omit performance analyses, or are not generalizable (i.e., special purpose), or simply do not perform well (i.e., not scalable, not efficient, etc.). On the other hand, we proposed a well-defined, implemented, and tested IoT architecture that combines MQTT-based IoT networks and an Ethereum-based blockchain in a modular way to mitigate the mentioned immutability and event management issues in a scalable and efficient manner with the help of a custom smart contract. The proposed scheme involves IoT clients (i.e., sensor and actuator nodes) that communicate through MQTT via Mosquitto, broker nodes that implement Mosja-JS, a message queue node that runs Apache Kafka, an IoT server that drives the Proto Actor model, and an Ethereum-based blockchain node that inherits Quorum ledger with smart contract support as demonstrated in Figures 4.1 and 4.2. So, messages containing sensor data are stored in the blockchain network, and the smart contract runs to assess event management decisions (e.g., actuator triggers) when necessary.

6.1 Thesis Results

A successful implementation was made via Docker containers on a PC. Without loss of generality, a forest fire monitoring system was considered as the use case for validation purposes. Mission-critical IoT data can reliably be stored and processed using the proposed scheme, while for trivial data, there is still no need to undertake the operating costs of a blockchain ledger. In addition to the importance of the collected data, the frequency of it is also important. The scheme may be tackled by bottlenecks when data transmission frequency is very high (e.g., several hundreds of TPS, although this is mostly application-dependent). Quorum with Raft consensus protocol works well under the test case conditions so that in a very large network consisting of 1000 clients, sensor data can be recorded to the ledger by a rate of 12.5 TPS, and event trigger messages can be disseminated to the actuators by a rate of 4 TPS (including requests and 8 TPS considering one-way rate). A smaller network would perform even better as the scheme has polynomial time and message complexity.

6.2 Future Work

There are hundreds of alternative open source blockchains and counting. Thus, further research may reveal better-performing ledgers and/or consensus algorithms for this purpose. Different IoT protocols such as constrained application protocol 1 (CoAP) or lightweight machine-to-machine (LWM2M) may be better for various scenarios. Future works shall also include the integration of sidechains (as explained in Musungate, Candan, Çabuk, & Dalkılıç (2019)) to extend the network with additional layers of secure data storage. Throughout this study, we have discovered two possible barriers to widespread adoption of the proposed method: The first is the considerably low number of allowed transactions per second within most blockchain systems. The issue is still an open problem for crowded or highly active networks. It occurs when the IoT system transmits data much faster than its blockchain counterpart can record to the ledger. But this may not be the case as we already see that our TPS requirements can be met by an Ethereum network. The

second is the lack of unique smart contract libraries for the IoT. We will be working on new IoT-oriented smart contract libraries to increase the applicability of blockchain into the IoT domain. Finally, our validation use case is another new working area for IoT. New studies may focus on forest fire detection sensor mash network structure and their numbers to find best practices.

REFERENCES

Akka (2021, February 12). *Why Modern Systems Need a New Programming Model*. https://doc.akka.io/docs/akka/current/typed/guide/actors-motivation.html

Alabdali, M. A. (2022). A Novel Framework of an IOT-Blockchain-Based Intelligent System. *Hindawi, Wireless Communications and Mobile Computing, 2022*.

Ali, J., Ali, T., Musa, S. & Zahrani, A. (2018). Towards secure iot communication with smart contracts in a blockchain infrastructure. *International Journal of Advanced Computer Science and Applications, 9*(10), 578-585.

Ali, M. S., Vecchio, M., Pincheira, M., Dolui, K., Antonelli, F., & Rehmani, M. H. (2019). Applications of blockchains in the internet of things: A comprehensive survey. *IEEE Communications Surveys Tutorials, 21*(2), 1676-1717.

Apache JMeter (2020, November 28). *Apache JMeter*. https://jmeter.apache.org

Asynkron (2020, November 9). *Proto Actor*. https://proto.actor

Baliga, A., Subhod, I., Kamat, P. & Chatterjee, S. (2018). Performance evaluation of the quorum blockchain platform. *arXiv preprint arXiv:1809.03421*.

Benefield R. (2009). Agile Deployment: Lean Service Management and Deployment Strategies for the SAAS Enterprise. *42nd Hawaii International Conference on System Sciences; 1,* 5.

Bouras, M. A., Lu, Q., Dhelim, S. & Ning, H. (2021). A Lightweight Blockchain-Based IoT Identity Management Approach. *Future Internet 2021*, *13*(2).

Chien, H. Y., Chen, Y. J., Qiu, G. H., Liao, J. F., Hung, R. W., Lin, P. C., Kou, X. A., Chiang, M. L., & Su, C. (2020). *A MQTT-API-compatible IoT security-enhanced platform. International Journal of Sensor Networks, 32*(1), 54.

Collina, M., Corazza, G. E., & Vanelli-Coralli, A. (2012). Introducing the Qest Broker: Scaling the IoT by Bridging Mqtt and Rest. *In 2012 IEEE 23rd International Symposium on Personal, Indoor and Mobile Radio Communications - (PIMRC*), 36-41.

Confluent (2020, November 17). *Confluent*. https://confluent.io

Consensys (2020, October 28). *Build on Quorum, The Complete Open Source Blockchain Platform for Business*. https://consensys.net/quorum

Dabbagh, M., Choo, K. R., Beheshti, A., Tahir, M. & Safa, N. S. (2021). A Survey of Empirical Performance Evaluation of Permissioned Blockchain Platforms: Challenges and Opportunities. *Computers Security, 100, 102078.*

Datta, S., Das A.K., Kumar, A., Khushboo & Sinha D. (2020). Authentication and Privacy Preservation in IoT Based Forest-Fire Detection by Using Blockchain. *4th International Conference on Internet of Things and Connected Technologies; Jaipur, India, 2020,* 133-143.

Docker (2020, October 24). *Docker Overview*. https://docs.docker.com/get-started/overview

Elazhary, H. (2019). Internet of Things (IoT), Mobile Cloud, Cloudlet, Mobile IoT, IoT Cloud, Fog, Mobile Edge, and Edge Emerging Computing Paradigms: Disambiguation and Research Directions. *Journal of Network and Computer Applications,* 105-140.

Escolar, A. M., Alcaraz-Calero, J. M., Salva-Garcia, P., Bernabe, J. B., & Wang, Q. (2021). Adaptive Network Slicing in 24 Multi-Tenant 5G IoT Networks. *IEEE Access, 9, 14048-14069*.

Ethereum (2020, November 15). *Go Ethereum*. https://github.com/ethereum/go-ethereum

Ethereum-Blocks (2020, November 28). *Ethereum Blocks*. https://ethereum.org/en/developers/docs/blocks

Faisal, B. A. & Reaz, R. (2022). IoT-Blockchain in Remote Patient Monitoring. *ICFNDS 2021: The 5th International Conference on Future Networks & Distributed Systems*, 186–194.

Hang, L. & Kim, D. (2019, May). Design and Implementation of An Integrated IoT Blockchain Platform for Sensing Data Integrity. *Sensors, 19*(10*),* 2228.

Huang, Y., Bian, Y., Li, R., Zhao, J. L. & Shi, P. (2019). Smart Contract Security: A Software Lifecycle Perspective. *IEEE Access, 7, 150184-150202*.

Kafka (2020, November 17). *Apache Kafka*. https://kafka.apache.org

Kathayayani, N., Murthy, J. K., & Naik, V. N. (2020, May). Hyperledger Fabric Blockchain for Data Security in IoT Devices. *International Journal of Recent Technology and Engineering*, *9*(1), 2571-2577.

Kaur, R. & Ali, A. (2022). Implementation of Blockchain in IoT. *Emergent Converging Technologies and Biomedical Systems, Select Proceedings of ETBS, 2021*, 149-161.

Khare, S., Ashraf, A., Yousuf, M. & Rashid, M. (2022, January). Blockchain: Structure, Uses, and Applications in IoT. *Blockchain Security in Cloud Computing*, 131-144.

Mihelj, J., Zhang, Y., Kos, A. & Sedlar, U. (2019, July). Crowdsourced Traffic Event Detection and Source Reputation Assessment Using Smart Contracts. *Sensors (Basel, Switzerland), 19*(15), 3267.

Moniz, H. (2020). The Istanbul BFT Consensus Algorithm. *arXiv preprint arXiv:2002.03613.*

Mosca (2020, November 22). *Mosca*. https://github.com/moscajs/mosca

Mosquitto (2020, November 25). *Eclipse Mosquitto*. https://mosquitto.org

Moudoud, H., Cherkaoui, S. & Khoukhi, L. (2019). An IoT Blockchain Architecture Using Oracles and Smart Contracts: the Use-Case of A Food Supply Chain. *In 2019 IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC),* 1-6.

Musungate, B., Candan, B., Çabuk, U. & Dalkılıç, G. (2019). Sidechains: Highlights and Challenges. *Innovations in Intelligent Systems and Applications Conference (ASYU); 1,* 5.

Ongaro, D. & Ousterhout, J. (2014). In Search of An Understandable Consensus Algorithm. *In Proceedings of the 2014 USENIX Conference on USENIX Annual Technical Conference, USENIX ATC'14*, 305-320.

OpenZeppelin (2020, November 27). *OpenZeppelin*. https://github.com/OpenZeppelin/openzeppelin-contracts

Paho (2020, November 20). *Eclipse Paho*. https://eclipse.org/paho/index.php?page=clients/golang/index.php

Paz, S. & Bernardino, J. (2017). Web Platform Assessment Tools: An Experimental Evaluation. *In: Web Information Systems 24 and Technologies, 45,* 63.

Reyna, A., Martin, C., Chen, J., Soler, E. & Diaz, M. (2018). On Blockchain and Its Integration with IoT Challenges and Opportunities. *Future Generation Computer Systems, 88,* 173-190.

Rogojanu, T., Ghita, M., Stanciu, V., Ciobanu, R., Marin, R., Pop, F., & Dobre, C. (2018). Netiot: A Versatile IoT Platform Integrating Sensors and Applications. *In 2018 Global Internet of Things Summit (GIoTS).*

Saputhanthri, M., Alwis, C. & Liyanage, M. (2021). Emergence of Blockchain based IoT Marketplaces. *2021 Joint European Conference on Networks and Communications (EuCNC) & 6G SummitAt: Porto, Portugal.*

Schaffer, M.,Angelo, M. D. & Salzer G. (2019). Performance and Scalability of Private Ethereum, Blockchains. *Business Process Management: Blockchain and Central and Eastern Europe Forum,* 103-118.

Seneviratne, C., Wijesekara, P. & Leung, H. (2020). Performance Analysis of Distributed Estimation for Data Fusion Using a Statistical Approach in Smart Grid Noisy Wireless Sensor Networks. *Sensors 2020; 20 (2): 567*. https://doi.org/10.3390/s20020567.

Shaheen, J. A. (2017). Apache Kafka: Real Time Implementation with Kafka Srchitecture Review. *International 42 Journal of Advanced Science and Technology, 109,* 35-42.

Son, B., Her, Y. & Jung-Gyu K. (2006). A Design and Implementation of Forest-Fires Surveillance System Based on Wireless Sensor Networks for South Korea Mountains. *International Journal of Computer Science and Network Security, 6*(9), 124-130.

Wood, G. (2021, December 18). *Ethereum: A Secure Decentralized Generalised Transaction Ledger*. https://ethereum.github.io/yellowpaper/paper.pdf.

Xu, H., Zhang, L., Liu, Y., & Cao, B. (2020). Raft Based Wireless Blockchain Networks in The Presence of Malicious Jamming. *IEEE Wireless Communications Letters, 9*(6), 817-821.

Yao, H., Mai, T., Wang, J., Ji, Z., Jiang, C. & Qian, Y. (2019). Resource Trading in Blockchain-Based Industrial Internet of Things. *IEEE Transactions on Industrial Informatics, 15*(6), 3602-3609.

Zastring (2021, December 18*). Gas, Gas Price and Gas Limit*. https://www.zastrin.com/courses/ethereum-primer/lessons/2-6.

51nodes (2020, November 5). *Quorum Local Raft Network. 2020*. https://github.com/51nodes/quorum-local-raft-network

APPENDICES

Table A.1: Table of 4 TPS

Table of 4 TPS									
# of Transaction	**Duration (ms)**	**# of Transaction**	**Duration (ms)**	**# of Transaction**	**Duration (ms)**	**# of Transaction**	**Duration (ms)**	**# of Transaction**	**Duration (ms)**
1	151	2	180	3	77	4	58	5	53
6	51	7	52	8	48	9	52	10	53
11	50	12	48	13	49	14	50	15	45
16	48	17	48	18	52	19	51	20	51
21	47	22	52	23	63	24	51	25	48
26	56	27	50	28	43	29	47	30	46
31	47	32	85	33	57	34	53	35	53
36	49	37	46	38	48	39	48	40	45
41	46	42	46	43	49	44	49	45	46
46	47	47	45	48	41	49	43	50	49
51	48	52	49	53	45	54	46	55	46
56	44	57	127	58	55	59	48	60	47
61	44	62	45	63	49	64	59	65	49
66	106	67	61	68	54	69	50	70	53
71	46	72	50	73	48	74	51	75	46
76	169	77	52	78	70	79	61	80	54
81	52	82	49	83	57	84	48	85	48
86	56	87	55	88	51	89	45	90	50
91	48	92	50	93	50	94	52	95	50
96	50	97	47	98	49	99	53	100	53
101	51	102	51	103	51	104	52	105	50
106	54	107	50	108	50	109	51	110	52
111	55	112	51	113	49	114	51	115	53
116	54	117	49	118	51	119	53	120	51
121	48	122	51	123	49	124	47	125	52
126	52	127	48	128	54	129	53	130	52
131	59	132	76	133	54	134	48	135	57
136	49	137	49	138	50	139	53	140	45
141	58	142	62	143	61	144	61	145	52
146	75	147	62	148	67	149	58	150	55
151	54	152	49	153	49	154	51	155	50
156	49	157	53	158	96	159	49	160	47
161	46	162	49	163	49	164	58	165	52
166	49	167	49	168	49	169	48	170	47
171	50	172	47	173	47	174	51	175	48
176	46	177	45	178	47	179	51	180	48

Table A.1 continues

181	56	182	55	183	49	184	50	185	47
186	114	187	55	188	48	189	48	190	47
191	47	192	50	193	49	194	51	195	50
196	46	197	49	198	47	199	48	200	51
201	49	202	49	203	46	204	46	205	55
206	63	207	57	208	54	209	56	210	49
211	48	212	47	213	129	214	60	215	50
216	49	217	49	218	48	219	47	220	48
221	45	222	47	223	49	224	47	225	44
226	46	227	50	228	52	229	46	230	58
231	48	232	45	233	50	234	50	235	48
236	121	237	49	238	50	239	46	240	50
241	47	242	52	243	48	244	51	245	45
246	49	247	51	248	47	249	49	250	49
251	46	252	50	253	48	254	49	255	45
256	47	257	116	258	52	259	49	260	51
261	45	262	61	263	56	264	57	265	67
266	70	267	75	268	52	269	50	270	48
271	49	272	47	273	47	274	48	275	51
276	47	277	45	278	49	279	52	280	50
281	51	282	48	283	49	284	51	285	52
286	50	287	46	288	46	289	45	290	46
291	49	292	46	293	49	294	47	295	49
296	49	297	48	298	51	299	48	300	45
301	49	302	47	303	47	304	51	305	46
306	82	307	56	308	63	309	60	310	62
311	62	312	54	313	48	314	50	315	49
316	52	317	47	318	45	319	51	320	46
321	48	322	45	323	50	324	47	325	46
326	48	327	58	328	50	329	47	330	104
331	63	332	55	333	54	334	50	335	46
336	45	337	42	338	44	339	44	340	43
341	43	342	47	343	43	344	48	345	45
346	41	347	44	348	58	349	44	350	44
351	46	352	44	353	44	354	46	355	41
356	41	357	41	358	43	359	42	360	47
361	46	362	46	363	42	364	43	365	47
366	48	367	43	368	42	369	42	370	42
371	45	372	46	373	41	374	48	375	43
376	49	377	50	378	47	379	47	380	43
381	44	382	44	383	44	384	41	385	52

Table A.1 continues

386	54	387	56	388	48	389	59	390	133
391	60	392	86	393	70	394	95	395	65
396	51	397	52	398	52	399	45	400	44
401	52	402	45	403	41	404	44	405	42
406	43	407	42	408	46	409	49	410	105
411	47	412	45	413	43	414	42	415	42
416	40	417	42	418	42	419	39	420	44
421	44	422	55	423	44	424	36	425	48
426	54	427	44	428	41	429	44	430	47
431	41	432	36	433	39	434	39	435	43
436	40	437	42	438	42	439	43	440	43
441	41	442	66	443	45	444	41	445	46
446	46	447	111	448	66	449	62	450	72
451	53	452	49	453	46	454	51	455	46
456	46	457	47	458	49	459	49	460	49
461	48	462	47	463	49	464	47	465	55
466	50	467	49	468	46	469	52	470	51
471	44	472	47	473	57	474	56	475	48
476	49	477	46	478	48	479	47	480	44
481	48	482	47	483	47	484	48	485	47
486	52	487	48	488	45	489	48	490	51
491	50	492	49	493	48	494	48	495	47
496	46	497	49	498	52	499	46	500	47
501	122	502	64	503	63	504	69	505	52
506	49	507	54	508	51	509	48	510	55
511	47	512	49	513	44	514	49	515	48
516	48	517	50	518	53	519	47	520	46
521	50	522	51	523	53	524	51	525	49
526	50	527	47	528	47	529	45	530	49
531	48	532	48	533	45	534	49	535	47
536	52	537	46	538	47	539	58	540	48
541	52	542	49	543	54	544	50	545	46
546	48	547	49	548	51	549	56	550	52
551	50	552	53	553	59	554	55	555	55
556	79	557	85	558	74	559	94	560	60
561	63	562	70	563	60	564	63	565	67
566	64	567	65	568	60	569	61	570	62
571	127	572	97	573	71	574	57	575	50
576	51	577	50	578	46	579	48	580	44
581	49	582	50	583	48	584	46	585	47
586	46	587	50	588	45	589	58	590	51

Table A.1 continues

591	49	592	50	593	50	594	52	595	46
596	50	597	49	598	50	599	50	600	48
601	47	602	52	603	49	604	46	605	51
606	51	607	49	608	49	609	45	610	105
611	66	612	71	613	55	614	55	615	52
616	49	617	50	618	52	619	51	620	49
621	56	622	52	623	47	624	46	625	49
626	45	627	56	628	130	629	60	630	76
631	57	632	59	633	59	634	60	635	51
636	55	637	53	638	56	639	55	640	50
641	46	642	46	643	47	644	45	645	47
646	47	647	47	648	81	649	46	650	48
651	50	652	48	653	52	654	52	655	48
656	48	657	49	658	52	659	47	660	46
661	48	662	49	663	55	664	51	665	47
666	63	667	70	668	73	669	59	670	57
671	54	672	54	673	50	674	51	675	49
676	47	677	44	678	48	679	45	680	48
681	49	682	51	683	49	684	52	685	52
686	50	687	50	688	54	689	52	690	45
691	46	692	44	693	47	694	47	695	46
696	47	697	47	698	48	699	48	700	50
701	44	702	46	703	50	704	49	705	44
706	46	707	49	708	44	709	48	710	46
711	50	712	50	713	46	714	49	715	48
716	44	717	44	718	49	719	52	720	49
721	48	722	61	723	63	724	65	725	54
726	54	727	49	728	50	729	46	730	49
731	52	732	49	733	48	734	54	735	52
736	56	737	55	738	53	739	56	740	129
741	84	742	74	743	59	744	63	745	59
746	64	747	57	748	49	749	55	750	55
751	117	752	64	753	65	754	71	755	76
756	64	757	58	758	71	759	64	760	53
761	54	762	54	763	57	764	52	765	53
766	48	767	54	768	60	769	59	770	55
771	55	772	49	773	54	774	51	775	53
776	52	777	53	778	53	779	65	780	76
781	105	782	65	783	59	784	60	785	56
786	56	787	56	788	64	789	52	790	53
791	53	792	60	793	63	794	59	795	61

Table A.1 continues

796	136	797	61	798	56	799	63	800	57
801	129	802	62	803	54	804	51	805	50
806	52	807	50	808	58	809	50	810	49
811	50	812	56	813	52	814	52	815	52
816	48	817	53	818	54	819	51	820	51
821	49	822	49	823	43	824	45	825	51
826	49	827	47	828	47	829	48	830	49
831	46	832	52	833	52	834	114	835	68
836	63	837	73	838	54	839	51	840	53
841	48	842	49	843	49	844	53	845	49
846	53	847	48	848	50	849	59	850	49
851	49	852	116	853	49	854	49	855	50
856	49	857	47	858	46	859	46	860	48
861	48	862	44	863	47	864	48	865	54
866	49	867	49	868	48	869	48	870	48
871	56	872	76	873	59	874	51	875	59
876	62	877	61	878	61	879	54	880	47
881	48	882	50	883	55	884	50	885	66
886	48	887	47	888	45	889	46	890	47
891	46	892	46	893	46	894	107	895	65
896	68	897	56	898	54	899	53	900	53
901	49	902	51	903	47	904	52	905	51
906	45	907	45	908	47	909	45	910	47
911	45	912	110	913	51	914	52	915	55
916	50	917	48	918	47	919	46	920	48
921	46	922	45	923	52	924	54	925	56
926	47	927	51	928	44	929	45	930	45
931	52	932	51	933	47	934	45	935	48
936	46	937	46	938	81	939	49	940	45
941	45	942	46	943	45	944	48	945	50
946	48	947	51	948	52	949	52	950	44
951	49	952	43	953	50	954	53	955	46
956	112	957	62	958	67	959	68	960	54
961	52	962	49	963	48	964	48	965	48
966	53	967	47	968	49	969	49	970	47
971	49	972	47	973	48	974	50	975	47
976	49	977	56	978	46	979	46	980	50
981	50	982	54	983	48	984	47	985	45
986	43	987	47	988	46	989	46	990	44
991	43	992	46	993	55	994	44	995	48
996	55	997	51	998	47	999	50	1000	84

Table A.2: Table of 5 TPS

Table of 5 TPS									
# of Transaction	**Duration (ms)**	**# of Transaction**	**Duration (ms)**	**# of Transaction**	**Duration (ms)**	**# of Transaction**	**Duration (ms)**	**# of Transaction**	**Duration (ms)**
1	49	2	19	3	18	4	18	5	21
6	23	7	17	8	17	9	27	10	23
11	25	12	15	13	15	14	29	15	27
16	24	17	22	18	23	19	25	20	33
21	21	22	32	23	25	24	18	25	22
26	31	27	18	28	16	29	26	30	24
31	26	32	20	33	30	34	29	35	24
36	17	37	19	38	17	39	28	40	20
41	24	42	20	43	33	44	22	45	17
46	16	47	27	48	22	49	21	50	20
51	22	52	27	53	27	54	17	55	18
56	26	57	34	58	19	59	19	60	19
61	19	62	20	63	19	64	18	65	19
66	28	67	23	68	17	69	16	70	23
71	26	72	30	73	28	74	27	75	26
76	28	77	16	78	25	79	17	80	16
81	18	82	53	83	55	84	47	85	33
86	31	87	16	88	25	89	16	90	22
91	26	92	24	93	17	94	23	95	26
96	16	97	16	98	22	99	16	100	25
101	26	102	26	103	24	104	15	105	31
106	22	107	16	108	18	109	19	110	48
111	34	112	31	113	27	114	20	115	16
116	28	117	20	118	16	119	24	120	24
121	23	122	20	123	20	124	19	125	18
126	21	127	17	128	15	129	22	130	16
131	23	132	32	133	25	134	21	135	33
136	35	137	24	138	24	139	30	140	31
141	16	142	18	143	17	144	16	145	17
146	37	147	48	148	36	149	35	150	19
151	32	152	35	153	19	154	17	155	15
156	27	157	34	158	30	159	18	160	47
161	46	162	45	163	49	164	52	165	51
166	32	167	23	168	18	169	24	170	26
171	29	172	23	173	19	174	17	175	17
176	25	177	16	178	17	179	20	180	14
181	15	182	16	183	16	184	22	185	15

Table A.2 continues

186	31	187	19	188	18	189	26	190	26
191	18	192	28	193	29	194	27	195	29
196	31	197	18	198	23	199	30	200	21
201	28	202	17	203	19	204	16	205	18
206	16	207	16	208	17	209	26	210	16
211	24	212	24	213	28	214	21	215	27
216	19	217	15	218	27	219	25	220	15
221	26	222	16	223	52	224	48	225	29
226	35	227	19	228	26	229	26	230	20
231	16	232	17	233	26	234	21	235	16
236	26	237	17	238	17	239	24	240	24
241	20	242	18	243	16	244	42	245	19
246	14	247	17	248	21	249	16	250	23
251	18	252	16	253	17	254	16	255	30
256	19	257	22	258	28	259	29	260	28
261	27	262	16	263	26	264	15	265	17
266	15	267	17	268	16	269	17	270	26
271	26	272	19	273	17	274	25	275	34
276	25	277	16	278	16	279	28	280	21
281	52	282	27	283	16	284	34	285	33
286	25	287	20	288	20	289	24	290	28
291	16	292	17	293	25	294	32	295	29
296	15	297	20	298	23	299	22	300	28
301	19	302	15	303	21	304	25	305	20
306	23	307	23	308	20	309	23	310	20
311	15	312	16	313	14	314	22	315	23
316	22	317	21	318	15	319	15	320	26
321	26	322	22	323	28	324	21	325	16
326	14	327	15	328	17	329	16	330	24
331	23	332	29	333	18	334	36	335	30
336	17	337	25	338	18	339	16	340	16
341	25	342	24	343	16	344	21	345	41
346	29	347	16	348	16	349	19	350	19
351	27	352	20	353	15	354	17	355	16
356	23	357	15	358	20	359	23	360	18
361	17	362	25	363	15	364	17	365	17
366	24	367	23	368	18	369	18	370	25
371	24	372	22	373	16	374	17	375	24
376	19	377	24	378	18	379	24	380	29
381	28	382	15	383	16	384	24	385	28
386	23	387	19	388	15	389	16	390	37

Table A.2 continues

391	45	392	47	393	16	394	23	395	46
396	19	397	16	398	17	399	24	400	23
401	28	402	18	403	17	404	27	405	14
406	24	407	32	408	16	409	28	410	17
411	31	412	15	413	15	414	26	415	15
416	23	417	25	418	15	419	17	420	16
421	21	422	24	423	16	424	21	425	15
426	24	427	16	428	15	429	23	430	18
431	20	432	16	433	15	434	19	435	14
436	22	437	17	438	15	439	24	440	16
441	24	442	24	443	21	444	24	445	16
446	22	447	22	448	15	449	16	450	15
451	24	452	23	453	14	454	24	455	18
456	32	457	33	458	19	459	27	460	26
461	26	462	28	463	36	464	18	465	20
466	48	467	50	468	40	469	46	470	23
471	24	472	27	473	15	474	21	475	25
476	40	477	18	478	16	479	16	480	16
481	23	482	17	483	15	484	22	485	15
486	19	487	21	488	20	489	22	490	19
491	16	492	27	493	16	494	27	495	23
496	16	497	25	498	17	499	17	500	23
501	16	502	22	503	16	504	16	505	19
506	28	507	26	508	16	509	17	510	15
511	28	512	18	513	15	514	17	515	25
516	17	517	20	518	15	519	25	520	24
521	29	522	27	523	15	524	22	525	32
526	17	527	26	528	40	529	31	530	17
531	46	532	17	533	17	534	17	535	29
536	29	537	47	538	20	539	16	540	15
541	26	542	28	543	17	544	17	545	16
546	23	547	20	548	19	549	17	550	17
551	16	552	17	553	16	554	22	555	23
556	17	557	16	558	18	559	24	560	27
561	28	562	27	563	17	564	17	565	26
566	29	567	19	568	19	569	19	570	16
571	32	572	33	573	17	574	25	575	27
576	20	577	17	578	17	579	28	580	24
581	21	582	18	583	17	584	23	585	25
586	56	587	49	588	66	589	48	590	56
591	23	592	32	593	16	594	27	595	23

Table A.2 continues

596	19	597	17	598	17	599	29	600	29
601	22	602	15	603	16	604	17	605	23
606	26	607	19	608	16	609	20	610	17
611	43	612	24	613	16	614	15	615	15
616	15	617	21	618	15	619	22	620	15
621	25	622	23	623	17	624	25	625	25
626	26	627	30	628	15	629	24	630	23
631	60	632	37	633	31	634	31	635	34
636	15	637	17	638	18	639	16	640	28
641	17	642	16	643	18	644	25	645	26
646	23	647	15	648	15	649	20	650	17
651	17	652	23	653	16	654	23	655	25
656	25	657	18	658	25	659	49	660	47
661	53	662	50	663	17	664	22	665	14
666	22	667	16	668	16	669	22	670	25
671	26	672	26	673	15	674	18	675	24
676	21	677	26	678	16	679	15	680	18
681	17	682	15	683	21	684	37	685	35
686	37	687	25	688	16	689	25	690	23
691	32	692	26	693	19	694	17	695	24
696	15	697	15	698	17	699	18	700	17
701	13	702	23	703	22	704	15	705	14
706	21	707	25	708	18	709	24	710	30
711	16	712	24	713	17	714	15	715	15
716	22	717	27	718	22	719	29	720	22
721	22	722	23	723	19	724	25	725	18
726	83	727	45	728	18	729	20	730	15
731	14	732	17	733	18	734	19	735	17
736	16	737	17	738	21	739	23	740	15
741	23	742	24	743	22	744	26	745	25
746	18	747	26	748	22	749	16	750	25
751	16	752	34	753	22	754	15	755	16
756	15	757	27	758	18	759	15	760	18
761	21	762	26	763	23	764	27	765	24
766	29	767	28	768	23	769	24	770	27
771	16	772	14	773	15	774	30	775	26
776	14	777	22	778	25	779	26	780	28
781	15	782	16	783	24	784	14	785	25
786	14	787	16	788	25	789	26	790	23
791	15	792	16	793	14	794	22	795	24
796	16	797	21	798	19	799	22	800	21

Table A.2 continues

801	16	802	22	803	19	804	15	805	15
806	21	807	17	808	16	809	16	810	16
811	15	812	16	813	15	814	16	815	16
816	16	817	16	818	26	819	24	820	25
821	15	822	41	823	31	824	22	825	22
826	20	827	26	828	18	829	28	830	22
831	21	832	24	833	24	834	16	835	23
836	24	837	62	838	14	839	18	840	16
841	21	842	43	843	18	844	20	845	14
846	25	847	18	848	23	849	14	850	15
851	18	852	15	853	19	854	23	855	27
856	16	857	22	858	82	859	46	860	25
861	25	862	13	863	18	864	18	865	14
866	15	867	13	868	15	869	15	870	23
871	20	872	24	873	20	874	15	875	26
876	23	877	24	878	24	879	17	880	22
881	24	882	17	883	46	884	34	885	52
886	40	887	51	888	50	889	32	890	37
891	26	892	15	893	16	894	14	895	15
896	17	897	15	898	22	899	26	900	21
901	15	902	15	903	69	904	53	905	48
906	28	907	36	908	34	909	26	910	31
911	23	912	20	913	28	914	14	915	16
916	16	917	16	918	25	919	15	920	76
921	15	922	26	923	14	924	15	925	21
926	13	927	15	928	17	929	21	930	17
931	22	932	22	933	21	934	24	935	24
936	23	937	23	938	16	939	17	940	15
941	23	942	20	943	22	944	17	945	15
946	19	947	22	948	23	949	15	950	21
951	25	952	16	953	16	954	17	955	24
956	16	957	14	958	22	959	16	960	17
961	26	962	25	963	15	964	15	965	15
966	39	967	16	968	28	969	16	970	24
971	15	972	17	973	27	974	18	975	25
976	21	977	24	978	18	979	17	980	15
981	14	982	17	983	35	984	16	985	15
986	22	987	20	988	21	989	15	990	18
991	16	992	14	993	15	994	15	995	24
996	18	997	17	998	16	999	17	1000	19

Table A.3: Table of 10 TPS

Table of 10 TPS									
# of Transaction	**Duration (ms)**	**# of Transaction**	**Duration (ms)**	**# of Transaction**	**Duration (ms)**	**# of Transaction**	**Duration (ms)**	**# of Transaction**	**Duration (ms)**
1	75	2	76	3	29	4	15	5	18
6	41	7	15	8	19	9	16	10	20
11	16	12	16	13	16	14	23	15	16
16	17	17	16	18	18	19	17	20	16
21	15	22	17	23	15	24	14	25	15
26	18	27	15	28	14	29	14	30	15
31	17	32	15	33	15	34	14	35	14
36	23	37	15	38	31	39	46	40	16
41	41	42	16	43	21	44	39	45	15
46	18	47	15	48	14	49	16	50	14
51	16	52	14	53	15	54	16	55	15
56	16	57	17	58	15	59	15	60	14
61	19	62	16	63	15	64	16	65	15
66	49	67	19	68	15	69	18	70	16
71	15	72	15	73	15	74	17	75	25
76	15	77	21	78	15	79	15	80	21
81	15	82	15	83	15	84	14	85	15
86	15	87	15	88	19	89	15	90	15
91	18	92	16	93	15	94	15	95	21
96	16	97	14	98	15	99	15	100	16
101	16	102	15	103	16	104	16	105	15
106	17	107	15	108	15	109	14	110	14
111	16	112	14	113	16	114	15	115	14
116	14	117	15	118	16	119	15	120	16
121	16	122	15	123	16	124	15	125	17
126	15	127	16	128	18	129	18	130	17
131	18	132	31	133	16	134	14	135	15
136	15	137	17	138	14	139	15	140	13
141	15	142	15	143	15	144	16	145	16
146	14	147	13	148	15	149	14	150	16
151	15	152	15	153	15	154	15	155	15
156	17	157	16	158	15	159	14	160	15
161	15	162	13	163	20	164	16	165	19
166	18	167	15	168	14	169	24	170	14
171	30	172	25	173	16	174	25	175	14
176	27	177	18	178	18	179	18	180	18
181	17	182	16	183	61	184	25	185	45
186	16	187	39	188	25	189	17	190	38

Table A.3 continues

191	15	192	17	193	15	194	21	195	15
196	14	197	17	198	18	199	16	200	14
201	17	202	16	203	15	204	15	205	14
206	15	207	14	208	19	209	15	210	20
211	14	212	20	213	14	214	13	215	16
216	16	217	14	218	14	219	13	220	14
221	15	222	15	223	16	224	22	225	15
226	16	227	16	228	18	229	14	230	13
231	14	232	15	233	14	234	15	235	20
236	25	237	20	238	14	239	17	240	23
241	25	242	23	243	17	244	26	245	29
246	24	247	18	248	31	249	14	250	45
251	14	252	52	253	14	254	44	255	15
256	50	257	16	258	81	259	17	260	14
261	14	262	16	263	18	264	13	265	46
266	15	267	45	268	14	269	42	270	16
271	15	272	15	273	16	274	14	275	15
276	16	277	15	278	14	279	15	280	14
281	17	282	15	283	30	284	38	285	37
286	14	287	44	288	13	289	21	290	18
291	14	292	14	293	15	294	20	295	21
296	14	297	16	298	15	299	15	300	15
301	15	302	17	303	15	304	15	305	16
306	17	307	16	308	16	309	14	310	15
311	17	312	15	313	17	314	14	315	15
316	18	317	15	318	16	319	15	320	13
321	16	322	15	323	15	324	14	325	15
326	15	327	14	328	14	329	15	330	14
331	14	332	14	333	14	334	20	335	42
336	17	337	43	338	15	339	14	340	32
341	14	342	34	343	14	344	35	345	14
346	31	347	14	348	21	349	19	350	25
351	14	352	17	353	15	354	25	355	14
356	15	357	16	358	14	359	16	360	16
361	15	362	17	363	17	364	16	365	15
366	16	367	16	368	17	369	14	370	40
371	16	372	48	373	13	374	53	375	14
376	16	377	15	378	16	379	16	380	15
381	14	382	15	383	16	384	17	385	13
386	23	387	14	388	14	389	117	390	18
391	14	392	39	393	14	394	24	395	14

Table A.3 continues

396	42	397	14	398	20	399	15	400	16
401	16	402	16	403	15	404	25	405	15
406	14	407	16	408	17	409	16	410	13
411	13	412	16	413	15	414	20	415	14
416	15	417	20	418	13	419	15	420	14
421	14	422	15	423	15	424	14	425	14
426	14	427	15	428	14	429	15	430	14
431	14	432	16	433	14	434	21	435	14
436	14	437	14	438	15	439	13	440	15
441	14	442	15	443	15	444	18	445	27
446	14	447	16	448	14	449	18	450	15
451	14	452	27	453	16	454	15	455	13
456	15	457	14	458	15	459	15	460	14
461	14	462	16	463	14	464	14	465	14
466	12	467	16	468	14	469	16	470	14
471	14	472	15	473	16	474	15	475	14
476	15	477	14	478	14	479	44	480	14
481	30	482	33	483	15	484	37	485	14
486	13	487	14	488	20	489	14	490	13
491	16	492	14	493	15	494	13	495	15
496	14	497	16	498	13	499	29	500	32
501	27	502	13	503	42	504	37	505	16
506	50	507	14	508	18	509	14	510	15
511	15	512	15	513	14	514	27	515	14
516	13	517	16	518	14	519	15	520	17
521	15	522	16	523	16	524	15	525	13
526	15	527	14	528	15	529	14	530	13
531	13	532	14	533	22	534	14	535	15
536	15	537	16	538	15	539	16	540	16
541	15	542	14	543	14	544	46	545	14
546	14	547	15	548	15	549	15	550	14
551	15	552	15	553	14	554	17	555	15
556	16	557	14	558	15	559	13	560	14
561	15	562	14	563	14	564	17	565	15
566	16	567	14	568	13	569	16	570	14
571	14	572	15	573	14	574	15	575	13
576	14	577	13	578	22	579	14	580	15
581	13	582	15	583	14	584	14	585	15
586	14	587	12	588	14	589	31	590	33
591	41	592	14	593	46	594	13	595	16
596	15	597	16	598	14	599	14	600	13

Table A.3 continues

601	16	602	13	603	14	604	14	605	15
606	57	607	20	608	15	609	44	610	12
611	51	612	13	613	41	614	15	615	30
616	29	617	15	618	16	619	13	620	14
621	15	622	25	623	14	624	15	625	14
626	17	627	15	628	14	629	14	630	13
631	16	632	14	633	16	634	16	635	18
636	17	637	19	638	13	639	48	640	16
641	14	642	16	643	13	644	14	645	17
646	12	647	13	648	15	649	14	650	14
651	14	652	14	653	14	654	15	655	14
656	18	657	14	658	13	659	15	660	14
661	14	662	16	663	33	664	13	665	14
666	14	667	14	668	18	669	14	670	15
671	16	672	14	673	14	674	15	675	18
676	15	677	14	678	15	679	14	680	15
681	15	682	14	683	15	684	13	685	14
686	15	687	43	688	14	689	16	690	15
691	14	692	15	693	16	694	14	695	14
696	16	697	16	698	16	699	13	700	18
701	14	702	12	703	14	704	13	705	14
706	15	707	14	708	14	709	13	710	13
711	13	712	15	713	14	714	26	715	16
716	40	717	38	718	16	719	43	720	14
721	20	722	19	723	33	724	18	725	25
726	27	727	16	728	15	729	19	730	29
731	26	732	15	733	14	734	16	735	15
736	17	737	15	738	15	739	14	740	15
741	14	742	14	743	14	744	38	745	14
746	16	747	17	748	15	749	17	750	15
751	16	752	15	753	15	754	15	755	15
756	15	757	15	758	15	759	17	760	16
761	19	762	18	763	17	764	16	765	16
766	17	767	18	768	18	769	19	770	19
771	19	772	20	773	19	774	18	775	18
776	18	777	16	778	18	779	18	780	18
781	28	782	15	783	18	784	18	785	14
786	16	787	15	788	14	789	16	790	16
791	15	792	14	793	17	794	15	795	15
796	14	797	15	798	17	799	16	800	18
801	14	802	17	803	15	804	14	805	16

Table A.3 continues

806	15	807	15	808	14	809	17	810	16
811	17	812	18	813	19	814	18	815	17
816	22	817	51	818	15	819	16	820	16
821	18	822	15	823	14	824	15	825	14
826	15	827	14	828	14	829	16	830	15
831	14	832	14	833	15	834	101	835	17
836	37	837	35	838	14	839	18	840	14
841	13	842	15	843	15	844	15	845	16
846	15	847	15	848	16	849	15	850	16
851	15	852	30	853	15	854	15	855	16
856	14	857	15	858	15	859	16	860	14
861	16	862	14	863	15	864	15	865	15
866	17	867	16	868	14	869	14	870	15
871	14	872	16	873	15	874	15	875	14
876	14	877	15	878	15	879	15	880	17
881	15	882	17	883	16	884	15	885	15
886	15	887	15	888	14	889	15	890	19
891	13	892	15	893	17	894	13	895	18
896	14	897	14	898	18	899	14	900	16
901	14	902	14	903	15	904	15	905	15
906	15	907	14	908	17	909	14	910	15
911	14	912	15	913	15	914	18	915	18
916	14	917	14	918	15	919	16	920	15
921	17	922	17	923	17	924	16	925	47
926	16	927	44	928	16	929	15	930	15
931	14	932	14	933	15	934	14	935	15
936	15	937	17	938	14	939	15	940	16
941	16	942	18	943	17	944	15	945	48
946	15	947	30	948	14	949	43	950	13
951	16	952	13	953	14	954	15	955	15
956	14	957	17	958	17	959	15	960	16
961	15	962	15	963	14	964	14	965	15
966	14	967	14	968	15	969	16	970	16
971	15	972	15	973	13	974	15	975	24
976	25	977	23	978	16	979	16	980	15
981	15	982	16	983	15	984	15	985	14
986	15	987	14	988	15	989	17	990	14
991	18	992	14	993	15	994	17	995	14
996	14	997	14	998	20	999	14	1000	14

Table A.4: Table of 12.5 TPS

Table of 12.5 TPS									
# of Transaction	**Duration (ms)**	**# of Transaction**	**Duration (ms)**	**# of Transaction**	**Duration (ms)**	**# of Transaction**	**Duration (ms)**	**# of Transaction**	**Duration (ms)**
1	50	2	41	3	20	4	16	5	37
6	15	7	38	8	15	9	44	10	13
11	47	12	15	13	40	14	15	15	15
16	16	17	16	18	16	19	14	20	15
21	22	22	21	23	17	24	15	25	16
26	17	27	15	28	16	29	14	30	16
31	16	32	15	33	22	34	19	35	24
36	24	37	16	38	30	39	16	40	15
41	25	42	15	43	27	44	23	45	15
46	14	47	35	48	13	49	17	50	15
51	16	52	16	53	15	54	17	55	21
56	14	57	16	58	15	59	21	60	15
61	14	62	14	63	15	64	16	65	18
66	19	67	17	68	14	69	15	70	15
71	15	72	15	73	14	74	15	75	13
76	14	77	15	78	14	79	18	80	44
81	14	82	43	83	16	84	16	85	48
86	14	87	46	88	15	89	16	90	14
91	14	92	14	93	13	94	15	95	16
96	16	97	22	98	24	99	28	100	14
101	14	102	25	103	13	104	21	105	16
106	20	107	15	108	15	109	13	110	15
111	47	112	14	113	14	114	23	115	18
116	16	117	13	118	31	119	14	120	29
121	15	122	15	123	41	124	14	125	20
126	12	127	15	128	14	129	15	130	15
131	17	132	27	133	16	134	14	135	14
136	24	137	22	138	14	139	24	140	14
141	16	142	14	143	15	144	39	145	15
146	47	147	18	148	14	149	16	150	14
151	17	152	23	153	22	154	21	155	18
156	15	157	15	158	14	159	24	160	20
161	14	162	27	163	16	164	15	165	14
166	18	167	15	168	16	169	15	170	14
171	14	172	15	173	14	174	14	175	16
176	16	177	16	178	14	179	15	180	27
181	16	182	36	183	13	184	51	185	13
186	31	187	22	188	14	189	15	190	14

Table A.4 continues

191	15	192	15	193	14	194	13	195	15
196	16	197	17	198	16	199	15	200	15
201	44	202	14	203	14	204	13	205	16
206	15	207	15	208	14	209	14	210	18
211	16	212	16	213	15	214	16	215	14
216	14	217	15	218	15	219	14	220	14
221	17	222	18	223	21	224	20	225	15
226	17	227	18	228	27	229	14	230	41
231	17	232	14	233	41	234	14	235	38
236	14	237	16	238	17	239	16	240	15
241	20	242	15	243	20	244	13	245	28
246	16	247	28	248	23	249	17	250	40
251	16	252	46	253	15	254	30	255	14
256	29	257	17	258	33	259	14	260	13
261	15	262	14	263	29	264	15	265	40
266	14	267	16	268	16	269	14	270	16
271	15	272	16	273	14	274	15	275	14
276	15	277	15	278	17	279	16	280	15
281	15	282	17	283	15	284	15	285	15
286	17	287	28	288	15	289	21	290	38
291	14	292	20	293	43	294	14	295	41
296	14	297	39	298	14	299	48	300	15
301	25	302	15	303	14	304	47	305	14
306	43	307	16	308	32	309	17	310	15
311	16	312	15	313	15	314	15	315	16
316	14	317	15	318	17	319	14	320	16
321	14	322	13	323	46	324	14	325	14
326	15	327	15	328	16	329	17	330	14
331	14	332	14	333	15	334	15	335	17
336	14	337	14	338	14	339	16	340	15
341	14	342	14	343	14	344	16	345	16
346	13	347	20	348	16	349	14	350	15
351	15	352	16	353	14	354	15	355	43
356	14	357	29	358	14	359	45	360	13
361	24	362	28	363	15	364	47	365	14
366	23	367	16	368	16	369	14	370	15
371	15	372	17	373	14	374	14	375	15
376	16	377	16	378	13	379	46	380	17
381	14	382	16	383	40	384	16	385	44
386	13	387	44	388	15	389	16	390	16
391	16	392	14	393	15	394	23	395	28

Table A.4 continues

396	16	397	43	398	15	399	33	400	13
401	27	402	16	403	14	404	15	405	15
406	15	407	29	408	15	409	24	410	15
411	14	412	49	413	15	414	15	415	15
416	15	417	16	418	17	419	17	420	14
421	26	422	22	423	33	424	16	425	43
426	13	427	36	428	13	429	19	430	15
431	17	432	14	433	14	434	23	435	15
436	38	437	15	438	40	439	15	440	17
441	78	442	14	443	14	444	15	445	14
446	15	447	15	448	14	449	16	450	14
451	14	452	16	453	14	454	14	455	16
456	14	457	14	458	14	459	21	460	17
461	16	462	16	463	16	464	16	465	17
466	42	467	15	468	35	469	13	470	43
471	14	472	17	473	13	474	15	475	13
476	16	477	14	478	14	479	16	480	17
481	15	482	15	483	15	484	16	485	15
486	14	487	13	488	14	489	16	490	16
491	53	492	18	493	15	494	15	495	16
496	15	497	16	498	20	499	13	500	15
501	15	502	44	503	22	504	14	505	16
506	20	507	16	508	21	509	15	510	22
511	14	512	24	513	21	514	18	515	15
516	28	517	35	518	28	519	16	520	18
521	17	522	31	523	16	524	16	525	15
526	21	527	15	528	20	529	15	530	21
531	15	532	18	533	16	534	14	535	35
536	17	537	15	538	25	539	16	540	16
541	15	542	15	543	17	544	14	545	15
546	15	547	36	548	16	549	35	550	14
551	33	552	18	553	36	554	13	555	34
556	13	557	47	558	13	559	36	560	14
561	28	562	15	563	39	564	15	565	18
566	14	567	14	568	14	569	19	570	15
571	21	572	13	573	24	574	14	575	28
576	23	577	20	578	21	579	14	580	14
581	14	582	43	583	14	584	37	585	12
586	25	587	16	588	36	589	14	590	39
591	13	592	17	593	15	594	14	595	14
596	23	597	20	598	27	599	15	600	16

Table A.4 continues

601	18	602	14	603	17	604	15	605	23
606	15	607	26	608	14	609	14	610	24
611	15	612	44	613	13	614	18	615	15
616	14	617	14	618	15	619	14	620	15
621	15	622	15	623	14	624	17	625	17
626	16	627	20	628	21	629	16	630	22
631	23	632	15	633	15	634	16	635	15
636	14	637	16	638	20	639	18	640	19
641	17	642	38	643	16	644	48	645	16
646	45	647	17	648	30	649	13	650	16
651	15	652	14	653	16	654	15	655	15
656	15	657	15	658	51	659	15	660	46
661	14	662	18	663	22	664	16	665	41
666	13	667	50	668	14	669	43	670	12
671	41	672	14	673	43	674	13	675	44
676	13	677	48	678	14	679	46	680	14
681	25	682	14	683	26	684	14	685	15
686	38	687	14	688	19	689	15	690	15
691	14	692	41	693	12	694	34	695	14
696	20	697	15	698	39	699	14	700	18
701	15	702	15	703	14	704	14	705	25
706	15	707	46	708	14	709	15	710	15
711	14	712	15	713	14	714	15	715	15
716	15	717	16	718	15	719	25	720	15
721	14	722	15	723	14	724	15	725	16
726	15	727	19	728	23	729	17	730	16
731	14	732	14	733	15	734	15	735	14
736	18	737	15	738	15	739	27	740	16
741	15	742	29	743	15	744	39	745	13
746	43	747	15	748	40	749	13	750	14
751	13	752	17	753	13	754	14	755	14
756	15	757	27	758	14	759	14	760	14
761	15	762	20	763	14	764	25	765	21
766	15	767	14	768	14	769	14	770	13
771	20	772	15	773	14	774	16	775	15
776	14	777	27	778	15	779	24	780	14
781	13	782	15	783	15	784	14	785	15
786	16	787	17	788	38	789	12	790	46
791	15	792	44	793	14	794	51	795	15
796	31	797	13	798	25	799	15	800	14
801	28	802	18	803	16	804	13	805	14

Table A.4 continues

806	13	807	15	808	21	809	19	810	14
811	15	812	15	813	15	814	16	815	15
816	15	817	44	818	14	819	45	820	14
821	34	822	15	823	38	824	12	825	33
826	14	827	39	828	15	829	15	830	16
831	14	832	14	833	14	834	22	835	14
836	15	837	14	838	14	839	25	840	15
841	15	842	17	843	20	844	17	845	27
846	16	847	22	848	21	849	18	850	36
851	14	852	16	853	14	854	15	855	14
856	21	857	15	858	14	859	20	860	19
861	21	862	17	863	21	864	24	865	20
866	12	867	21	868	13	869	14	870	19
871	17	872	13	873	14	874	16	875	17
876	14	877	21	878	25	879	13	880	15
881	14	882	14	883	24	884	15	885	14
886	12	887	24	888	18	889	28	890	23
891	14	892	22	893	14	894	14	895	28
896	22	897	15	898	35	899	13	900	52
901	14	902	46	903	13	904	14	905	13
906	15	907	43	908	14	909	17	910	14
911	14	912	15	913	14	914	13	915	14
916	17	917	15	918	14	919	15	920	20
921	13	922	23	923	18	924	23	925	15
926	15	927	16	928	14	929	14	930	28
931	13	932	35	933	13	934	14	935	16
936	14	937	22	938	21	939	15	940	22
941	12	942	49	943	14	944	39	945	14
946	40	947	14	948	30	949	15	950	15
951	13	952	14	953	14	954	14	955	15
956	16	957	16	958	15	959	16	960	16
961	18	962	18	963	18	964	18	965	16
966	16	967	36	968	15	969	15	970	42
971	14	972	32	973	14	974	48	975	15
976	24	977	13	978	42	979	16	980	25
981	15	982	43	983	14	984	14	985	16
986	24	987	113	988	16	989	12	990	15
991	14	992	28	993	15	994	15	995	43
996	15	997	15	998	14	999	13	1000	15