A BLOCKCHAIN-BASED FRAMEWORK FOR ENHANCING SECURITY AND PRIVACY IN IOT INTEGRATION

by

Ali EGHMAZI

THESIS PRESENTED TO ÉCOLE DE TECHNOLOGIE SUPÉRIEURE IN PARTIAL FULFILLMENT FOR A MASTER'S DEGREE
WITH THESIS IN ELECTRICAL ENGINEERING
M.A.Sc.

MONTREAL, MARCH 26, 2024

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

BOARD OF EXAMINERS

THIS THESIS HAS BEEN EVALUATED

BY THE FOLLOWING BOARD OF EXAMINERS

Mr. Rene Jr.Landry, Thesis Supervisor
Department of Electrical Engineering at École de technologie supérieure

Mrs. Naouel Moha, President of the Board of Examiners
Department of Software Engineering and IT at École de technologie supérieure

Mr. Kim Khoa Nguyen, Member of the jury
Department of Electrical Engineering at École de technologie supérieure

Mr. Guy Chevrette External Evaluator
IMETRIK Global Inc

THIS THESIS WAS PRESENTED AND DEFENDED

IN THE PRESENCE OF A BOARD OF EXAMINERS AND PUBLIC

MARCH 18, 2024

AT ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

ACKNOWLEDGMENT

I am profoundly grateful to Professor René Jr Landry for his invaluable guidance, support, and expertise throughout my master's research at ÉTS. His insightful perspectives and steadfast encouragement significantly shaped my journey, and his willingness to assist in any aspect of my work has been truly inspiring. His mentorship not only steered me through this research but also provided me with a unique opportunity to grow academically under his supervision, for which I am eternally thankful. My sincere gratitude extends to Mr. Guy Chevrette, whose profound knowledge and unwavering support have been pivotal in my research journey. His contributions have enriched my experience and understanding in ways that words cannot fully express.

I would also like to extend my thanks to Mr. Mohammad Hossein Attae, whose assistance in implementing case studies, articles, and other aspects of my research has been invaluable. His collaboration has greatly enhanced the quality and scope of my work. Working and collaborating with the members of the LASSENA lab has been an enriching experience. Each member's unique insights and contributions have not only facilitated a dynamic research environment but also fostered a space for learning and growth, for which I am deeply appreciative.

I must also express my deepest gratitude to my family - my parents and brother, whose love, guidance, and unwavering support have been my constant source of strength and motivation. Their belief in me has been a guiding light in all my pursuits. Lastly, I am immensely grateful to my friends for their endless support and encouragement, especially during the challenging moments of my master’s degree. Your solidarity and companionship have made this journey more memorable. Thank you all for being part of my academic voyage. Love and gratitude to each one of you!

Un cadre basé sur la blockchain pour renforcer la sécurité et la confidentialité dans l'intégration de l'ido

Ali Eghmazi

RÉSUMÉ

La propagation rapide de la technologie de l'Internet des Objets (IoT), qui devrait inclure des milliards d'appareils à l'avenir, nécessite le développement d'une plateforme sécurisée et évolutive pour l'administration et la préservation efficaces des informations et des données. Alors que les acteurs s'efforcent de réaliser le plein potentiel d'un tel déploiement à grande échelle, il devient évident que les solutions actuelles de sécurité et d'évolutivité pour l'IoT sont insuffisantes.

La blockchain a démontré sa capacité à valider, stocker et gérer correctement les données. Elle apparaît comme une solution potentielle pour une variété de difficultés liées aux données en raison de ses qualités intrinsèques de décentralisation, d'immuabilité et de transparence. Avec ses caractéristiques de sécurité naturelles, la technologie blockchain offre des options intrigantes pour surmonter ces défis.

Cette thèse tente de répondre aux difficultés actuelles en matière de sécurité des données IoT et de confidentialité des utilisateurs. À cette fin, nous proposons une infrastructure basée sur la blockchain conçue pour assurer un stockage de données sécurisé tout en améliorant la confidentialité des utilisateurs. Nous proposons une architecture à quatre couches dans cette thèse pour surmonter les problèmes liés à l'IoT massif. Notre stratégie utilise à la fois un stockage de données hors chaîne et en chaîne. Nous utilisons Hyperledger Fabric comme plateforme blockchain pour stocker les données de manière sécurisée, ce qui nous permet de vérifier l'intégrité des données. De plus, nous intégrons un stockage décentralisé pour améliorer la disponibilité des données.

Nous utilisons Apache Kafka pour le streaming de données en temps réel afin de garantir l'évolutivité. Nous donnons la priorité au chiffrement des données tout au long du processus pour assurer la confidentialité et la sécurité. Nous avons évalué la performance et l'utilisation des composants développés en utilisant une large gamme de plateformes, y compris Hyperledger Caliper et Explorer, parmi d'autres. Ces plateformes nous ont permis de tester et de mesurer minutieusement la résilience et les capacités de notre système sous de nombreux scénarios, offrant une évaluation complète de la performance et de la durabilité de la plateforme sous diverses conditions opérationnelles.

Cette thèse couvre la mise en œuvre et l'analyse de trois études de cas uniques. La plateforme a été testée dans des environnements pratiques pour démontrer sa capacité à répondre aux problèmes liés à l'utilisation croissante et à la complexité des appareils IoT. La thèse contribue aux efforts visant à rendre l'écosystème IoT plus sûr et plus fiable en utilisant les avantages de

la blockchain, conduisant finalement à une meilleure confiance des utilisateurs et à une acceptation plus large des technologies IoT.

Mots-clés: Blockchain, Hyperledger Fabric, Apache Kafka, Internet des objets, Sécurité, Confidentialité, Gestion des données

A Blockchain-based framework for enhancing security and privacy in IoT integration

Ali Eghmazi

ABSTRACT

The rapid spread of Internet of Things (IoT) technology, which is expected to include billions of devices in the future, need the development of a secure and scalable platform for the effective administration and preservation of information and data. As stakeholders work to realize the full potential of such large-scale deployment, it is becoming evident that present security and scalability solutions for IoT are insufficient.

Blockchain has shown its capacity to properly validate, store, and manage data. It appears as a possible solution for a variety of data-related difficulties because of its intrinsic qualities of decentralization, immutability, and transparency. With its natural security features, blockchain technology provides intriguing options to overcome these challenges.

This thesis attempts to address the current difficulties in MIoT data security and user privacy. To that aim, we provide a blockchain-based infrastructure designed to assure safe data storage while also improving user privacy. We propose a four-layered architecture in this thesis to overcome the issues related to Massive IoT. Our strategy makes use of both off-chain and on-chain data storage. We use Hyperledger Fabric as a blockchain platform to securely store data, allowing us to verify data integrity. In addition, we incorporate decentralized storage to improve data availability.

We utilize Apache Kafka for real-time data streaming to ensure scalability and low latency. We prioritize data encryption throughout the process to ensure privacy and security. We have evaluated the performance and utilization of the developed parts using a wide range of platforms, including Hyperledger Calliper and Explorer, among others. These platforms have allowed us to thoroughly test and measure our system's resilience and capabilities under numerous scenarios, offering a full assessment of the platform's performance and durability under varied operating conditions.

This thesis covers the implementation and analysis of three unique case studies. The platform has been proven in practical environments to demonstrate its capacity to address issues related to the rising usage and complexity of IoT devices. The thesis adds to attempts to make the IoT ecosystem safer and more dependable by using the benefits of blockchain, eventually leading to better user trust and wider acceptance of IoT technologies.

Keywords: Blockchain, Hyperledger Fabric, Apache Kafka, Internet of Things, Security, Privacy, Data Management

TABLE OF CONTENTS

INTRODUCTION 1

CHAPTER 1 LITERATURE REVIEW 5
1.1 Historical Evolution and Introduction of IoT 5
1.1.1 IoT Architecture, Key concepts, and protocols 7
1.1.2 IoT platforms 11
1.1.2.1 Types IoT Platform 12
1.1.3 Introduction on Real-time data streaming 13
1.1.3.1 Apache Kafka as a real time data streaming 14
1.1.3.2 Other Real-Time data streaming technologies 16
1.1.4 Issues and challenges in IoT 19
1.1.4.1 Challenges on security 19
1.1.4.2 Challenges on privacy 21
1.1.4.3 Other challenges and Issues 22
1.2 An overview of Blockchain 23
1.2.1 Blockchain's characteristics 25
1.2.2 Blockchain architecture 26
1.2.2.1 Hash function definition 28
1.2.2.2 Consensus algorithm in Blockchain 29
1.2.2.3 Smart contract 33
1.2.3 Confidentiality Tiers in Blockchain 35
1.2.4 Blockchain based IoT (BIoT) 36
1.2.4.1 Blockchain based IoT Challenges 38
1.2.5 An overview of Hyperledger project 40
1.2.6 Hyperledger Frameworks 41
1.2.6.1 Hyperledger-Fabric 41
1.2.6.2 Hyperledger project comparison 43
1.2.7 Recent Developments on Blockchain and IoT innovation 46
1.3 Chapter summary 48

CHAPTER 2 Methodology and Design of architecture 49
2.1 Proposed architecture 49
2.2 First Layer (physical layer) 52
2.2.1 Case studies 52
2.2.1.1 Pinnacle 100 and sensors pairing 53
2.2.1.2 Raspberry pie and sensor pairing 54
2.2.1.3 Automatic Dependent Surveillance-Broadcast (ADS-B) as a device 57

2.3 Second Layer (Network layer) 58
2.3.1 Case Study 59
2.3.1.1 Pico LTE as network layer 59
2.4 Third Layer (middle layer) 60
2.4.1 Third layer's component 62
2.4.1.1 Real-time data streaming component 63
2.4.1.2 Blockchain 65
2.4.1.3 Data base component 67
2.5 Forth layer (Application layer) 69
2.5.1 Authentication 70
2.5.2 User interface 71
2.6 Chapter summary 72

CHAPTER 3 Implementation of the platform 75
3.1 The Implementation Process 75
3.1.1 Real-time data streaming (KAFKA Apache) implementation 76
3.1.1.1 Kafka integrated with Network (Producer) 78
3.1.1.2 Apache Kafka Integrated with Data base (Consumer) 79
3.1.2 Data Base Implementation 81
3.1.3 Blockchain Implementations 83
3.1.3.1 Smart contracts 86
3.1.3.2 Blockchain Software Development Kit 87
3.1.3.3 Blockchain Authentication 88
3.2 User Interface (UI) implementation 91
3.2.1 Authentication in UI 92
3.2.2 Web interface 94
3.3 Real-time data visualization 96
3.4 Data Encryption/Decryption method 97
3.5 Chapter summary 98

CHAPTER 4 Results and Evaluation 101
4.1 Kafka performance 101
4.2 Blockchain performance 104
4.2.1 Monitoring with Hyperledger-Explorer 105
4.2.2 Evaluation with Hyperledger Caliper 107
4.2.2.1 Throughput results 109
4.2.2.2 Latency results 110
4.3 System resource utilization 113
4.4 Chapter summary 115

CONCLUSION…. 117

APPENDIX I…… 122

LIST OF BIBLIOGRAPHICAL REFERENCES 125

LIST OF TABLES

Page

Table 1.1 Complete History of The Evolution of IoT From 1969 to 2025 Taken from M R and Bhowmik (2023, p. 1) 6

Table 1.2 Standard Protocols Taken from M R and Bhowmik (2023, p. 3) 10

Table 1.3 High Level Overview Comparison 17

Table 1.4 Challenges in IoT and Blockchain Solutions Adapted from Sharma and Babu Battula (2022, p. 55) 37

Table 1.5 Comparison Between Hyperledger Projects 44

Table 2.1 Proposed Four-Layered Architecture in Detail 50

LIST OF FIGURES

Page

Figure 1.1 Detailed IoT Architecture Taken from Zhonghua and Goyal (2023, p. 73) 8

Figure 1.2 IoT Platform Architecture Taken from Astropekakis et al (2022, p. 304) 12

Figure 1.3 Apache Kafka High-Level Overview Taken from Vyas et al (2022, p. 466) 15

Figure 1.4 Comprehensive Blockchain's History Taken from Guo and Yu (2022, p. 2) 24

Figure 1.5 How Blockchain is Linked Taken from Bhushan et al (2021, p. 3) 27

Figure 1.6 Blockchain Structure Overview Taken from Sarmah (2018, p. 2) 28

Figure 1.7 Consensus Algorithms Classification Taken from K and S (2023, p. 3) 31

Figure 1.8 Compare POW and POS Taken from Leo and Hattingh (2021, p. 571) 33

Figure 1.9 Evolutionary Phases of a Smart Contract Taken from Wu et al (2022, p. 3) 34

Figure 1.10 Comprehensive BIoT Challenges Taken from Alzoubi et al (2022, p. 15) 40

Figure 1.11 Data flow in Hyperledger Fabric Taken from Xu et al (2021, p. 3) 42

Figure 1.12 Hyperledger Fabric Architecture Taken from Punathumkandi et al (2020, p. 90) 43

Figure 1.13 Blockchain Development Engagement Taken from Capocasale et al (2023, p. 6) 46

Figure 2.1 Four-Layered Architecture Components' 51

Figure 2.2 Left Image Shows Pinnacle 100 and right Image Shows BL6545 Sensor 54

Figure 2.3 Data flow in This Case Study 54

Figure 2.4 IbNav Sensor 6.1 56

Figure 2.5 Data Flow in This Case Study 56

Figure 2.6 Overview of the ADS-B Communication Taken from Sciancalepore, Alhazbi, and Di Pietro (2019, p. 3) 57

Figure 2.7 ADS-B Inside an Airplane Works as Sensor 58

Figure 2.8 Nutaq PicoLTE with 2 Antennas 60

Figure 2.9 Trinity Technologies in this Platform 63

Figure 2.10 Apache Kafka Architecture in this Platform 65

Figure 2.11 Hyperledger Fabric Structure in Platform 67

Figure 2.12 Data Structure in this Platform a with Chain Formatting 69

Figure 2.13 JWT Example 71

Figure 3.1 Steps of Integration in This Platform 76

Figure 3.2 Apache Kafka Containers 77

Figure 3.3 Apache Kafka User Interface 77

Figure 3.4 Apache Kafka Cluster Settings 78

Figure 3.5 Three Connected Sensors Data Format 79

Figure 3.6 Receiving Raw Data from Apache Kafka Consumer 79

Figure 3.7 Data after being Processed to String 80

Figure 3.8 Producer (left side) and Consumer (right side) Together 80

Figure 3.9 Three Topics Transferring in Apache Kafka Center 80

Figure 3.10 Detailed Topic with Parameters 81

Figure 3.11 IPFS Web User Interface 82

Figure 3.12 IPFS Status Web Page of Availability 83

Figure 3.13 Hyperledger Fabric Images 84

Figure 3.14 Containers Created in Hyperledger Fabric 84

Figure 3.15 Chaincode Deployment....85
Figure 3.16 Smart Contract Deployment on Each Peers....85
Figure 3.17 SDK in Hyperledger Fabric Taken from Su Wai et al (2020, p. 2)....88
Figure 3.18 LDAP User Interface....92
Figure 3.19 Schema Created in LDAP....93
Figure 3.20 User Profile in LDAP....93
Figure 3.21 Web User Interface Dashboard....94
Figure 3.22 Block of Data Requested by User to be Visualized....95
Figure 3.23 CID's Requested from Hyperledger Fabric....95
Figure 3.24 Visualized Data from a Sensor....96
Figure 3.25 Energy Consumption for Different Algorithms....97
Figure 4.1 Request Latency in Apache Kafka Producer....102
Figure 4.2 Request Latency in Apache Kafka Consumer....103
Figure 4.3 System Pool Usage in Apache Kafka....103
Figure 4.4 Hyperledger Explorer Main Dashboard....106
Figure 4.5 Running Network in Hyperledger Fabric....106
Figure 4.6 Transaction Details on Hyperledger Fabric....107
Figure 4.7 Details of Each Block Created by Hyperledger Fabric....107
Figure 4.8 Transaction Throughput (TPS) in Hyperledger Fabric....110
Figure 4.9 Inquiry performance (TPS) in Hyperledger Fabric....110
Figure 4.10 Maximum Latency for Transactions in Hyperledger Fabric....112
Figure 4.11 Minimum Latency for Transaction in Hyperledger Fabric....112
Figure 4.12 Average Latency for Transaction in Hyperledger Fabric....113
Figure 4.13 RAM Usage of Apache Kafka....114

Figure 4.14 CPU Usage of Apache Kafka 114

Figure 4.15 CPU Usage of Hyperledger Fabric 115

Figure 4.16 RAM Usage of Hyperledger Fabric 115

LIST OF ABREVIATIONS AND ACRONYMS

ADS-B	Automatic Dependent Surveillance-Broadcast
AMQP	Advanced Message Queuing Protocol
API	Application Programming Interface
BFT	Byzantine Fault Tolerance
CID	Content Identifier
CoAP	Constrained Application Protocol
DAG	Directed Acyclic Graph
DBFT	Delegated Byzantine Fault Tolerance
DDS	Data Distribution Service
DHT	Distributed Hash Table
DPoS	Delegated Proof of Stake
EPC	Evolved Packet Core
HLF	Hyperledger Fabric
IPFS	Interplanetary File System
IPSO	Internet Protocol for Smart Objects
IoT	Internet of Things
ITU	International Telecommunication Union
LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol
LTE eNodeB	Long-Term Evolution evolved NodeB
M2M	Machine-to-Machine
MQTT	Message Queue Telemetry Transport
PBFT	Practical Byzantine Fault Tolerance
PoB	Proof of Burn
PoC	Proof of Capacity
PoET	Proof of Elapsed Time
PoS	Proof of Stake
PoW	Proof of Work
REST	Representational State Transfer

RFID	Radio-Frequency Identification
SDK	Software Development Kit
Se-aaS	Sensors-as-a-Service
SOA	Service-Oriented Architecture
UEs	User Equipment’s
WAN	Wide Area Network
WI-FI	Wireless Fidelity

INTRODUCTION

Two game-changing breakthroughs have recently changed the digital landscape: the Internet of Things (IoT) and blockchain technology. While the Internet of Things has ushered in a new era of connectedness and smart interactions, it has also brought with it its own set of obstacles. Interestingly, the resilient and transparent nature of blockchain technology has the ability to overcome many of these difficulties. These technical wonders, when combined, have enormous potential to change businesses and reinvent our digital relationships. However, while the quest to smoothly integrate them is full of promise, it is also fraught with complications and hurdles.

The proliferation of IoT is evident in the growth of connected devices, which surged from 12.5 billion in 2010 to approximately 16.66 billion by 2019. Projections suggest that this number could skyrocket to an astounding 75.44 billion by 2025. The Internet of Things (IoT) represents a monumental shift in the way our world communicates with the digital universe. This convergence consolidates the entire world with a PC-based framework, extending a new dimension of communication for users. Rooted in foundational concepts from the 1990s, the IoT universe has grown exponentially, with the present era witnessing a marriage between real-life activities and their virtual counterparts.

Today, with the aid of technologies like Bluetooth, Wi-Fi, GPS, Zig-Bee, and RFID, our surroundings are more connected than ever, steering in what many term the ‘smart era’ (Faridul Islam Suny et al. 2021) However, this digital utopia is not devoid of challenges. The meteoric rise and adoption of IoT have brought forth issues of scalability, the imperative for devices to be self-organizing, and challenges related to the sheer volume of data being generated. Furthermore, the interpretation and meaningful analysis of this data remain significant hurdles (Verma and Prakash 2021).

In a parallel digital renaissance, blockchain technology, first conceptualized in 1991 by Stuart Haber and W. Scott Stornetta, emerged as a solution to the longstanding challenge of data

tampering, especially in economic transactions (Zambre, Panchal, and Chauhan 2023). It was in 2008, with Satoshi Nakamoto's introduction of Bitcoin, that the true potential of blockchain was globally acknowledged. This wasn't just the advent of a decentralized currency; it was a testament to blockchain's transformative capabilities. Beyond just facilitating financial transactions, blockchain promises a transparent, secure, and immutable ledger system, where trust is established not by central authorities, but by the very architecture of the technology. Each transaction, once recorded, is verifiable by every network participant, and the information, structured in blocks, forms an unalterable chain, representing a timeline of transactions (Singh and Chauhan 2021).

However, as individual technologies, while IoT and blockchain offer much promise, combining them leads us into new and unexplored areas. This convergence is seen by many as the next leap in digital innovation, promising an ecosystem where devices not only communicate smoothly but do so within a transparent, tamper-proof, and decentralized framework. Yet, this ambitious integration is not without its set of challenges. The inner complexities of each technology become magnified when combined, necessitating rigorous research and innovative solutions to realize their full potential in coupling (Alenizi and Al-Karawi 2023). The integration of IoT with blockchain faces significant challenges such as scalability, data volumes, interoperability, energy consumption, storage, and latency. Addressing these challenges is crucial for the effective merger of these transformative technologies.(Sadawi, Hassan, and Ndiaye 2021).

With the increasing need to connect IoT devices to the blockchain to ensure heightened security, it is imperative to understand and address the complexities and challenges inherent to this integration. Thus, the core research question driving this investigation is: How can we effectively connect IoT devices to the blockchain to ensure security, and what challenges might we encounter in this endeavor?

Research goals

The fundamental goal of this research is to investigate and comprehend the difficulties and constraints of combining the Internet of Things (IoT) with blockchain technology in order to achieve greater security. Given the revolutionary potential of both technologies, the purpose of this research is to give insights and approaches for overcoming integration obstacle and unlocking their combined potential.

The research will pursue the following aims to attain this operational goal:

- **Definition and Clarification:** To completely describe and clarify the problems and complexities of integrating IoT devices with blockchain to ensure increase security, particularly in light of both domains' fast progress and expansion.

- **Development of a Structured Framework:** To express the identified problems and complexities inside a structured framework that may guide future IoT and blockchain integration initiatives.

- **Operationalization and Application:** To operationalize the specified difficulties and complexities, allowing for effective and smooth application in real-world settings using the created framework.

Structure of thesis

This thesis initiates with an introduction that establishes the backdrop, emphasizing the contemporary advancements in the digital domain. Specifically, it highlights the emergence and significance of the Internet of Things (IoT) and blockchain technology. Chapter 1, the Literature Review, offers a comprehensive background on the Foundation of IoT, encompassing its platforms, real-time data streaming technologies, and inherent challenges. The chapter then expands into the realm of Blockchain, detailing it's ats architecture, diverse types, and the elaboration of the Hyperledger project.

Shifting gears, Chapter 2 describe the Architecture and Design of the Platform, chronicling its multi-layered structure and the components within. Chapter 3 ventures into the Implementation of the Platform, meticulously detailing the fusion of essential components such as Apache Kafka, Blockchain, and the Database, and elucidating the platfor''s data flow dynamics. Chapter 4 is pivotal, presenting Results and Evaluation and providing insights into various performance metrics. The thesis then culminates in Chapter 5, which offers reflections, highlights research limitations, and charts out directions for future explorations. This academic journey concludes with a synthesis of the primary findings and contributions, accompanied by a comprehensive Bibliography.

CHAPTER 1

LITERATURE REVIEW

In this chapter, we conduct a thorough literature analysis to examine the past and present status of research on two disruptive technologies: the Internet of Things (IoT) and Blockchain. This section delves into the technical architecture of IoT as well as the revolutionary ideas of Blockchain. This investigation provides us with insights into the difficulties and opportunities of both sectors. A precisely constructed four-layered architectural structure develops from this immense body of knowledge. This architecture, which incorporates the characteristics of both Blockchain and IoT, delivers creative answers to difficulties, bridging previously thought-to-be large and invincible gaps.

1.1 Historical Evolution and Introduction of IoT

Kevin Auston introduced the term "internet of thing" (IoT) in 1999, though its inception can be traced back to the creation of Arpanet in 1969. The journey of IoT experienced a notable development with the introduction of a toaster as the first IoT device in 1990. By 2005, the significance of IoT was underscored by the International Telecommunication Union's (ITU) study. The formation of the Internet Protocol for Smart Objects (IPSO) Alliance in 2008 further propelled the movement, and today, it boasts over 50 member companies, including major tech players. A landmark was achieved in 2010 when the number of IoT devices, totaling 12.5 billion, surpassed the global human population of 6.8 billion. For a comprehensive visual overview of the IoT's evolution, refer to Table 1.1 (M R and Bhowmik 2023).

Table 1.1 Complete History of The Evolution of
IoT From 1969 to 2025
Taken from M R and Bhowmik (2023, p. 1)

Year	Advancement in technology
1969	ARPANET concept
1974	TCP/IP protocols
1977	ARPANET implemented with TCP/IP
1990	Internet
1991	Tim Berners-Lee innvented www.
1997	Wi-Fi
1999	Term IoT was coined
1999	First device designed with RFID used in IoT
2003	Commercial deployment of RFID
2005	ITU's first report on IoT
2008	IoT gained recognition by European union
2011	IoT-Global standards Initiative was established
2013	Arduino, Rasberry pi and other platforms was developed and made more accessible
2014	3rd party devices like drones were incorporated
2016	IoT products available in market
2025	'prediction' 75 Billion Devices in IoT

The "Internet of Things" (IoT), sometimes termed the 'Internet of Objects', encompasses electrical devices of diverse size and functionalities linked to the Internet. This connection is predominantly established via wireless sensors, excluding devices mainly designed for person-to-person communication like the conventional Internet. A fundamental IoT ecosystem, as outlined by (Miraz and Ali 2018)consists of:

- **Sensors:** Responsible for gathering and relaying essential data.
- **Computing Node:** Housing the central processing unit (CPU), these nodes process the data from sensors.
- **Receiver:** A transceiver that facilitates the reception of data from either local and distant computing nodes or other gadgets.
- **Actuator:** An electromechanical system that, based on the Computing Node's decision and data from sensors or the Internet, triggers the relevant device for a specific action.
- **Device:** Executes the designated task when activated.

IoT is a paradigm that leverages network sensors, embedded technologies, artificial intelligence, and Radio-frequency identification (RFID) to interconnect various machines, facilitating data exchange over the Internet for diverse applications. The swift advancements in IoT research have unveiled several challenges, with cyber-attacks being a prominent vulnerability, as highlighted by (Nayancy, Dutta, and Chakraborty 2021).

1.1.1 IoT Architecture, Key concepts, and protocols

The Internet of Things (IoT) has emerged as a transformative force in the digital era. At its core, IoT is defined as a network of Internet-connected objects or devices equipped with embedded sensors. These sensors grant the devices the capability to collect, send, or exchange data. The essence of IoT lies in its ability to automatically, remotely, and continuously collect and share information without any interruptions. Today, the landscape of interconnected devices is vast, yet it lacks a standardized network or a clearly defined boundary. The IoT not only facilitates a numerous of applications but also addresses a broad spectrum of societal and industrial needs. It is anticipated to be a pivotal player in converting ordinary cities into smart cities, traditional homes into smart homes, and conventional electrical grids into smart grids. Furthermore, the IoT is seen as the backbone of the industrial sector's digitization, enhancing production processes and reducing costs (Obaidat et al. 2020; Sadawi et al. 2021).

When it comes to IoT architecture, multiple perspectives exist. The traditional view comprises three significant layers: Sensing, Network, and Application. The foundational layer, known as the perception or sensor layer, connects physical components to the IoT network, monitoring physical changes and transmitting data. The network or transmission layer aggregates this data and transfers it to decision-making units. The application layer, on the other hand, serves as the interface for end-users, providing essential services based on the data received. Another perspective views the IoT architecture as a service-oriented architecture (SOA) with four layers: perception, network, service, and application. This architecture emphasizes service discovery, quality, trustworthiness management, and service Application Programming Interfaces (API) (M R and Bhowmik 2023). A more detailed perspective presents a five-layer

architecture, encompassing the business layer, application layer, middle layer, network layer, and physical/sensor layer. Each layer has its distinct functions, ranging from data collection to business model formulation As Shown in Figure 1.1 (Zhonghua and Goyal 2023).

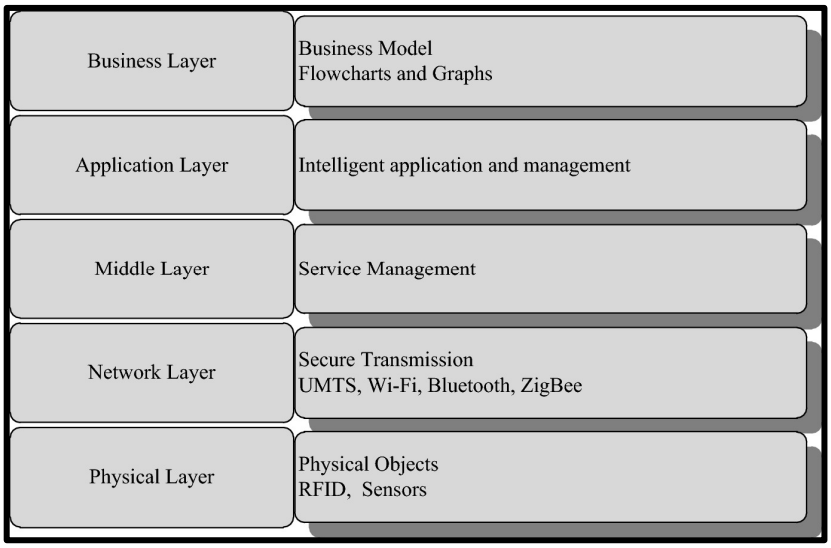


Figure 1.1 Detailed IoT Architecture
Taken from Zhonghua and Goyal (2023, p. 73)

As described in (Kenaza et al. 2022) each layer with its unique functions and responsibilities are:

- **Perception Layer:** Often called the Device Layer, it contains various sensors like RFID tags and Infrared. Its main function is to detect and collect data, such as temperature and motion, and send it to the next layer for secure transmission.
- **Network Layer:** Also termed the Transmission Layer, it ensures the secure transfer of data from the Perception layer using technologies like RFID, 5G, and Wi-Fi.
- **Middleware Layer:** This layer manages the communication between devices offering the same service type. It stores, processes, and makes decisions based on the data received from the Network Layer.
- **Application Layer:** Primarily responsible for global application management, it oversees all IoT system applications, from Smart Farming to Smart Healthcare.

- **Business Layer:** This layer oversees the entire IoT system, creating business models, charts, and reports from the data. It aids in making informed business decisions and strategizing.

Delving into the characteristics of IoT, several features stand out. Firstly, scalability and heterogeneity are foundational to IoT. Given that devices come from diverse hardware platforms, it is imperative for IoT devices to support this heterogeneity. Another crucial aspect is the need for unambiguous naming, addressing, or identification, especially with billions of interconnected devices in play. Interoperability, the ability of diverse devices to communicate and work cohesively, is essential for device integration within the IoT network. Moreover, IoT devices should be equipped to support mobile sensor networks where connectivity might be intermittent, ensuring reliable information delivery. Lastly, the dynamic nature of IoT necessitates support for the reprogramming of devices. These devices should be self-configurable, capable of self-discovery, and adept at processing vast amounts of data. IoT uses diverse communication protocols to ensure connectivity and application integration. These protocols, essential for transmitting sensor data, vary across LANs and WANs in an IoT system.(M R and Bhowmik 2023).

Lastly, in association with IoT technologies, advanced technologies are being incorporated into the IoT infrastructure. Cloud computing, for instance, offers on-demand access to a shared pool of computer resources, including servers, storage, and applications. Sensor-cloud, a heterogeneous computing environment, provides a platform for Sensors-as-a-Service (Se-aaS), allowing adaptive utilization of physical sensor resources. Another significant technology is fog computing, which combines networking, computation, and storage at the network's edge, delivering services closer to end-users. This technology is especially beneficial for applications that are latency-sensitive and produce vast amounts of data (Zhonghua and Goyal 2023).

Table 1.2 Standard Protocols
Taken from M R and Bhowmik (2023, p. 3)

Application Layer Protocols	User and node communication occurs through the application layer. Constrained application protocol (CoAP), extensible messaging and presence protocol (XMPP), advanced message queuing protocol (AMQP), and message queue telemetry transmission (MQTT) are a few application layer protocols.
Transport Layer Protocols	Transport layer protocols enable and safeguard the communication of the data as it travels between layers. Two primary transport layer protocols are transmission control protocol (TCP) and user datagram protocol (UDP).
Network Layer Protocols	The network layer protocols help individual devices to communicate with the router. Standard network layer protocols include Internet Protocol (IP) and low-power wireless personal area networks over IPV6 (6LoWPAN).
Data-Link Layer Protocols	Data link layer protocols transfer data within the system architecture, identifying and correcting errors found in the physical layer. Data link layer protocols include IEEE 802.15.4 and low power wide area network (LPWAN).
Physical Layer Protocols	The physical layer is the communication channel between devices within a specific environment. Physical layer protocols include Bluetooth Low Energy (BLE), Long-term evolution (LTE), Radio frequency identification (RFID), Near field communication (NFC), Wi-Fi/802.11, Z-Wave, and Zigbee.

Addressing the previously mentioned protocols, this section introduces various Internet of Things (IoT) application and transport layer protocols. These include Message Queue Telemetry Transport (MQTT), MQTT for Sensor Networks, Constrained Application Protocol (CoAP), Advanced Message Queuing Protocol (AMQP), Data Distribution Service (DDS), and Hypertext Transfer Protocol with Representational State Transfer (HTTP REST). These protocols are essential in middleware development since the IoT is heavily reliant on the internet. For example, the Constrained Application Protocol (CoAP) operates at the application layer, while Representational State Transfer (REST) protocol functions over HTTP for data communication between the client and server using CoAP. The Message Queue Telemetry Transport (MQTT) protocol connects networks and embedded devices to applications and middleware. The Advanced Message Queuing Protocol (AMQP) is designed specifically for message-oriented IoT scenarios, and the Data Distribution Service (DDS) is crafted for continuous Machine-to-Machine (M2M) communications. Table 1.2 provides a deeper understanding of the applicability and limitations of each of these IoT protocols (Deohate and Rojatkar 2021).

1.1.2 IoT platforms

IoT platforms, central to IoT solutions, serve as a bridge between the physical and digital realms. Navigating the vast landscape of platform vendors can be a challenging task, but with a meticulous assessment of specific factors, businesses can make informed decisions. It's vital to delve into platform specifics, such as its unique features, communication protocols, and its current operational status to determine its suitability (Ullah and Smolander 2019). Additionally, ensuring the secure and reliable connection of numerous devices is paramount, emphasizing the importance of user authentication, data integrity, and the secure transfer of data (Astropekakis et al. 2022).

Middleware, as described by (Deohate and Rojatkar 2021), is an indispensable software component in the IoT ecosystem, extending a spectrum of services ranging from integration to content management. IoT developers, when integrating middleware, must adhere to certain installation prerequisites. This installation process varies based on the project's existing structure and the anticipated platform needs post-installation. Moreover, understanding a middleware's business model, its associated communication protocols, and its current viability—whether operational or terminated by its developers—is crucial for its successful implementation.

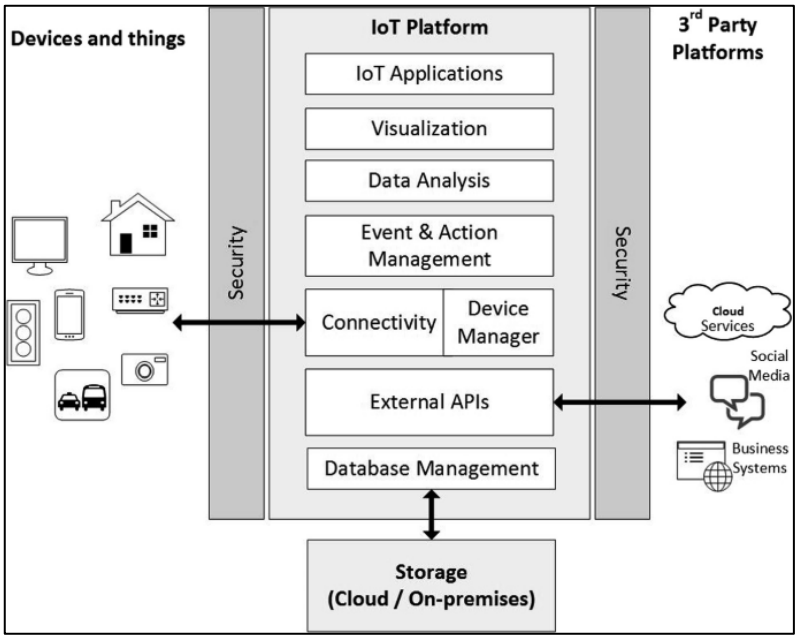


Figure 1.2 IoT Platform Architecture
Taken from Astropekakis et al (2022, p. 304)

1.1.2.1 Types IoT Platform

AWS IoT, developed by Amazon, centers around its AWS IoT Core, which ensures secure and controlled communication between devices and the AWS ecosystem. It offers a range of application protocols and prioritizes secure communication On the other hand, Google Cloud IoT focuses on securely connecting devices and boasts features that ensure safety against threats (Yu and Kim 2019). Furthermore, (Astropekakis et al. 2022) emphasize AWS IoT's data management, analytics capabilities, and its foundation on a secure cloud environment. Also, for Google cloud IoT highlight its analytics, visualization, and adaptability, emphasizing its user-friendly approach.

Samsung's ARTIK is an IoT platform integrating hardware, software, and cloud elements, focusing on advanced security measures and connectivity. Thing plug and GiGA IoTMakers are both open IoT platforms built on the oneM2M standard, with the former developed by SKT and the latter by the telecommunications company KT. Azure IoT, developed by Microsoft,

assists in building a plethora of IoT applications, integrating SaaS solutions, PaaS, and the intelligent Edge, with a unique security framework (Yu and Kim 2019). Alibaba IoT Cloud, Carriots, Cisco Kinetic, IBM Watson IoT, Oracle IoT Cloud, PTC Thingworx, SiteWhere, Thinger.io, ThingsBoard, ThingSpeak, Ubidots, and WSO2 are all versatile IoT platforms, each with its unique set of features, capabilities, and security measures, providing solutions ranging from device connectivity and data analytics to secure communication and AI integration (Astropekakis et al. 2022).

After establishing the significance and capacities of various IoT platforms, it is critical to acknowledge the role of data in this ecosystem. As IoT devices create massive volumes of data on a continual basis, the requirement for effective, fast, and dependable data processing becomes critical. Introduce yourself to the world of real-time data streaming technology. These solutions not only handle the flood of data from IoT devices, but also ensure that insights are generated instantly, allowing for quick decision-making and dynamic reactions in a variety of applications.

1.1.3 Introduction on Real-time data streaming

In the rapidly evolving digital landscape, the paradigm of real-time data streaming distinguishes itself from the episodic nature of traditional data processing. Modern IoT devices represent this shift, equipped to stream data in real-time, either directly to cloud platforms or through IoT gateways (Rahman and Das 2022). As technological advancements continue, a unique surge in data transmission across networks has emerged. This growing data ecosystem not only underscores the monumental significance of real-time data streaming but also introduces challenges. With the intensifying volume and velocity of data, there is an evident struggle in processing, storing, and filtering on singular servers, leading to potential delays and data loss (Torres and de Oliveira Silva 2023).

This data-centric era has ushered in a plethora of applications, redefining various sectors. From enhancing monitoring, management, and operational efficiency to reshaping decision-making

in businesses, the range of implications is endless. Personal health monitors, interconnected home appliances, medical diagnostic tools, and defect detection in industries are just some manifestations of this transformation. Furthermore, intricate applications such as participatory live street views for tourism, ultra-realistic sports broadcasts, real-time pedestrian flow management in urban landscapes, and monitoring systems for senior citizens highlight the depth of its impact. To navigate this intricate data landscape, robust technological frameworks are essential. Several platforms, including IBM Infosphere Stream, Amazon AWS IoT, Apache Storm and Spark, and Microsoft Stream Insight, have emerged, each specializing in high-speed real-time processing of substantial temporal data. Complementing these platforms, transmission protocols for IoT data streams like MQTT, CoAP, Web Socket, and IPv6LoWPAN emphasis a move towards edge-heavy computing (Banik, Cardenas, and Kim 2019).

1.1.3.1 Apache Kafka as a real time data streaming

Apache Kafka, originating from LinkedIn and open-sourced in 2016, is a powerful distributed streaming platform known for its scalability and fault-tolerance (Vyas et al. 2022). Built on publish-subscribe model, it consists of producers that publish data, brokers that manage it, and consumers that process it in real-time. Topics, which categorize data streams, can be divided into multiple partitions, enabling concurrent data writing by producers and parallel reading by consumers (Peddireddy 2023).

Kafka's architecture emphasizes topic partitioning, reflecting its degree of parallelism. Each topic's partitions are distributed among brokers, with provisions for replication across multiple replicas for fault tolerance. One replica acts as the leader, handling all data interactions, while the others synchronize data as followers. Within consumer groups, each consumer instance reads from a unique subset of partitions, maximizing parallel processing (Raptis and Passarella 2022).

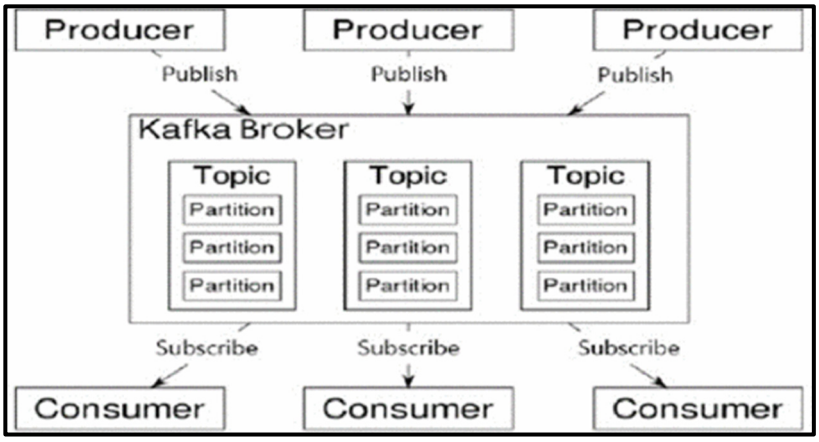


Figure 1.3 Apache Kafka High-Level Overview
Taken from Vyas et al (2022, p. 466)

However, Kafka faces challenges as data volumes grow, necessitating increased partitions to prevent bottlenecks. Balancing throughput and latency is crucial to prevent consumer lags or data loss. Despite its robustness, understanding Kafka's behavior, especially during drastic data volume changes, remains an area of active research (Vyas et al. 2022).

Performance

Apache Kafka, a distributed streaming platform, has been the subject of numerous studies examining its functionalities, applications, performance, latency, and reliability. These studies have delved into various aspects, from modeling the communication between producers and consumers to enhancing Kafka's fault tolerance and data recovery mechanisms. Additionally, research has been conducted on Kafka's applications in diverse fields, such as seismic waveform data processing and real-time analytics. Performance evaluations have also been undertaken to understand Kafka's behavior under different configurations and network conditions. Furthermore, the impact of various configuration parameters on Kafka's reliability and latency has been tested, with tools introduced to assess its reliability under varying network qualities (Raptis and Passarella 2022). (Peddireddy 2023) highlights Kafka's capabilities in data processing, real-time reporting, and alerting, emphasizing its scalability, real-time processing, fault tolerance, and flexibility.

Apache Kafka's performance is gauged using several key parameters. Throughput measures the system's capacity to process messages within a set time frame, indicating the efficiency of hardware and the data volume transmitted between producers and consumers. Latency represents the time duration for a message to travel from a Kafka producer to its consumer, encompassing various stages like production, publishing, committing, catching up, and extraction. Lastly, CPU and Memory Utilization assess Kafka's resource consumption, especially vital given its reliance on the file system for message storage and processing, particularly when handling extensive data (Vyas et al. 2022).

Apache Kafka boasts a distributed architecture that offers scalability, accommodating large data volumes. It supports real-time processing through Kafka Streams, ensuring up-to-date data analysis. Kafka's design prioritizes fault-tolerance, ensuring data availability amidst potential failures. Additionally, its flexibility is evident in its ability to integrate with various data sources via Kafka Connect. Despite its advantages, Kafka presents challenges. Its complexity requires a profound understanding of distributed systems for deployment and building. The financial cost associated with setting up Kafka, considering both hardware and software, can be significant. Furthermore, Kafka demands continuous maintenance and monitoring to maintain system health and optimal performance (Peddireddy 2023).

1.1.3.2 Other Real-Time data streaming technologies

The table below offers a high-level overview of several leading systems and frameworks in the realm of data processing, streaming, and analytics. Each entry succinctly captures the core essence and distinctive features of the respective system or framework. Ranging from the real-time data ingestion prowess of Apache Spark Streaming to the specialized spatiotemporal capabilities of JUST, this overview serves as a quick reference to the diverse functionalities and strengths of these cutting-edge platforms. Table 1.3 presents a high overview on real-time data streaming comparison between technologies.

Table 1.3 High Level Overview Comparison

System/Framework	Description	Key Features/Characteristics	Contribution
Apache Spark Streaming	Supports data ingestion from various sources	Supports streams to HDFS, AWS S3	(Navaz et al. 2019)
Apache Flink	Planning unbounded and restricted data sets	Coordinates with Hadoop YARN, Apache Mesos, Kubernetes	
Apache Storm	Real-time computing system	Can process unlimited data streams, supports any programming language	
System/Framework	Description	Key Features/Characteristics	Contribution
Apache Druid	OLAP software provider	Scales to a million RPM, highly available, multiple nodes	(Navaz et al. 2019)
Apache Samza	Stateful event-based applications	Runs on YARN, fault-tolerant, used by Uber, Netflix, etc.	
Amazon Kinesis Data Streams	Real-time data streaming service	Captures gigabytes of data, supports mobile applications	
ST-Hadoop	MapReduce framework for spatiotemporal data	Spatiotemporal data types, operations, and queries	(Sasaki 2022)

System/Framework	Description	Key Features/Characteristics	Contribution
STARK	Framework for spatiotemporal data on Spark	Integrates spatiotemporal data into Spark, supports DBSCAN	(Sasaki 2022)
GeoMesa	Geospatial querying and analytics	Spatiotemporal indexing, real-time stream processing	
TrajSpark	System for big trajectory data	IndexTRDD, global and local indexing	
DFT	Framework for similarity search queries over trajectory data	Segment-based partitioning, dual indices	
DITA	Distributed in-memory trajectory analytic system	Extends Spark SQL, supports trajectory similarity function	
UlTraMan	Unified storage and computing engine	Built on Spark, integrates Chronicle Map	
JUST	Spatiotemporal data engine	Uses NoSQL data store, GeoMesa indexing, Spark execution engine	

1.1.4 Issues and challenges in IoT

The Internet of Things (IoT) has emerged as a transformative force, bridging the gap between the digital and physical realms. With its promise of interconnected devices that can communicate and collaborate, the potential applications of IoT span various sectors, from healthcare to transportation (Beltran et al. 2022). However, these networks of devices, while offering unprecedented opportunities, also introduces a myriad of challenges. These challenges, ranging from technical to ethical, need to be addressed to ensure the broader adoption and trust in IoT technologies (Al-athwari and Hossain 2022).

The rapid proliferation of IoT devices has led to concerns about their security, privacy, and interoperability. As devices become more integrated into our daily lives, the stakes for ensuring their safe and reliable operation have never been higher. Addressing these challenges is not just about ensuring the smooth functioning of the devices but also about safeguarding the trust and confidence of the users (Rai, Kanday, and Thomas 2020)

1.1.4.1 Challenges on security

The interconnected nature of IoT devices exposes them to unsecured and unverified parts of the internet, posing significant risks to both private and governmental entities, and compromising the data being transmitted or shared. Major espionage events have stressed these vulnerabilities. Notably, many IoT devices are designed for broad dissemination and can autonomously establish connections, making their security landscape unique and complex (Beltran et al. 2022). Furthermore, most of these devices are "closed," preventing post-purchase security installations, and due to their hardware limitations, they often resort to lightweight algorithms, which can pose risks to the data stored on these devices (Sathi Reddy, Venkatesh, and Kumar 2022).

IoT's vast scale and diversity make it susceptible to a range of threats. Many devices, especially those with wireless sensors, are physically exposed, making them attractive targets for hackers.

This vulnerability is compounded by the fact that many IoT devices have limited computational resources. The challenges extend across the IoT architecture, from the physical level, where issues like jamming adversaries and insecure initializations arise (Patnaik, Padhy, and Srujan Raju 2021), to the logical level, which grapples with vulnerabilities in the application layer. The network layer, responsible for reliable data transmission, is particularly vulnerable to DoS attacks, while the transport and application layers face threats from insider attacks, misconfigurations, and unauthorized access, potentially compromising data storage and changeability (Sathi Reddy et al. 2022).

Addressing these security challenges requires a multi-faceted approach. The mobility of IoT, evident in applications like healthcare and transportation, necessitates adaptable security solutions (Rai et al. 2020). The lack of standardization in IoT security solutions, combined with the heterogeneity of data produced by different applications, further complicates matters. As the number of devices continues to grow, scalability becomes a pressing concern, and with the increasing range of attack sources, security solutions must be continuously updated and refined. Additionally, ensuring data confidentiality, integrity, and accessibility across all IoT systems is paramount, given the vulnerabilities associated with data storage, sharing, and manipulation (Al-athwari and Hossain 2022; Rai et al. 2020).

The summary of major security Issues will be:

- Interconnected nature leading to exposure on unsecured internet parts.
- "Closed" nature of many IoT devices preventing post-purchase security installations.
- Physical exposure of wireless sensors making them vulnerable to hackers.
- Challenges across IoT architecture, from physical to logical levels.
- Lack of standardization in IoT security solutions.
- Scalability concerns due to the rapid growth of IoT devices.
- Vulnerabilities associated with data storage, sharing, and manipulation.

1.1.4.2 Challenges on privacy

The Internet of Things (IoT) has revolutionized the way devices communicate and operate, but with this innovation comes significant concerns about privacy. The utility of IoT is dependent upon its ability to respect individual privacy choices, and any concerns in this domain could potentially hinder its deployment (Beltran et al. 2022). The omnipresence of IoT devices, capable of data selection and distribution almost anywhere, amplifies privacy concerns. This is especially true given the ease with which private data can be accessed globally without specific protective measures in place. Moreover, the protection and solitude of data in the IoT face threats from various digital attacks, risks, and exposures. Issues at the device level, such as lack of permission and confirmation, uncertain operating systems, firmware vulnerabilities, and weak transport layer encryption, further exacerbate these concerns (Srivastava and Pandey 2022).

IoT devices inherently collect, store, and communicate sensitive data. This necessitates a secure and confidential exchange of data across different networks. The vast connectivity of IoT creates numerous communication channels, which malicious attackers can exploit. While there are potential security measures for individuals, managing the data privacy of larger entities like businesses and organizations is more challenging. These entities must employ monitoring tools to safeguard against privacy threats and potential breaches. To bolster privacy, extensive research has been conducted, leading to the adoption of encryption algorithms. Digital signatures and Blockchain mechanisms have also been explored, though the latter may not be suitable for all IoT devices due to its resource-intensive nature (Al-athwari and Hossain 2022).

The practical implications of these privacy concerns are evident in scenarios like smart homes. Such homes utilize sensors to collect data on various activities, from morning routines to children's interactions with smart toys. While not all companies exploit this data, the potential for misuse is evident. The sheer volume of data collected can reveal intimate details about individuals and their behaviors. Ensuring privacy, therefore, is a shared responsibility between

consumers and service providers. Both parties must be transparent about their expectations and limitations. Surprisingly, many companies struggle to identify all IoT devices within their networks, underscoring the need for robust privacy solutions tailored to IoT applications (Rai et al. 2020).

The summary of major privacy concerns will be:

- Omnipresence of IoT devices leading to amplified privacy concerns.
- Device-level issues such as lack of permission and weak encryption.
- Potential misuse of data collected by IoT devices, especially in smart homes.
- The challenge of managing data privacy for larger entities like businesses.
- The need for robust encryption and security measures, with some solutions like blockchain not being universally suitable.

1.1.4.3 Other challenges and Issues

The Internet of Things (IoT) presents a multifaceted landscape of opportunities and challenges. One of the primary concerns is scalability, the system's ability to expand seamlessly. This is complicated by the diverse nature of IoT devices, each with varying storage capacities, data types, and bandwidths. Cloud-based IoT solutions exemplify scalable systems, offering the flexibility to expand the network as needed. However, the heterogeneity of devices, whether due to different communication protocols or data types, leads to issues like interoperability and privacy. Interoperability is crucial for the efficient exchange of information among IoT systems. Yet, the decentralized nature of these networks makes information exchange challenging. Additionally, many IoT devices, such as sensors and RFID tags, are resource-constrained, making them vulnerable to malicious attacks and limiting their computational capabilities (Garg et al. 2022; Srivastava and Pandey 2022).

(Alenizi and Al-Karawi 2023) further delineate challenges in the IoT domain. Smart objects in IoT should possess the capability to self-organize, adapting autonomously to their environments. The vast data volumes generated by IoT applications demand innovative

mechanisms for data storage, processing, and management. Accurate data interpretation is vital, requiring services that can draw meaningful conclusions from diverse sensor data. Interoperability remains a concern, necessitating common communication standards. Other challenges encompass automatic service discovery, software architecture complexity, security and privacy concerns, power supply limitations, and the evolution of efficient wireless communication standards.

The summary of major Issues and Concerns will be:

- Scalability concerns in accommodating diverse IoT devices.
- Heterogeneity and interoperability issues due to diverse communication protocols and data types.
- Resource constraints in IoT devices, making them susceptible to attacks.
- Need for self-organizing smart objects and efficient data interpretation.
- Software complexity and the necessity for enhanced wireless communication standards.
- Ensuring security and privacy in IoT transactions and interactions.

The increasing adoption of cloud computing in IoT systems offers enhanced analytical capabilities but also raises security and privacy concerns. These challenges make it harder to establish a trusted environment, especially when compared to IoT devices with potentially flawed security measures. Consequently, the reliability of IoT systems is threatened by these security and trust issues. To address these concerns, the importance of verifying unaltered data is emphasized. "Blockchain" is introduced as a potential solution to these challenges. To fully benefit from blockchain in IoT systems, it's crucial to understand and explore its value (Sadawi et al. 2021).

1.2 An overview of Blockchain

The concept of a blockchain-like protocol originated with Chaum in 1982, and the subsequent years witnessed pivotal advancements in the domain. Haber and Stornetta introduced cryptographic security in 1991, followed by the integration of Merkle trees in 1993. By 1998,

Szabo had conceptualized "bit gold," a pioneer to modern cryptocurrencies. However, the landscape truly transformed in 2008 when Nakamoto launched Bitcoin, a peer-to-peer electronic cash system, and concurrently introduced the term "blockchain" to describe its underlying distributed ledger. Ethereum, proposed by Buterin in 2013, marked another significant evolution, focusing on distributed data storage and smart contracts, with its enhanced version, Ethereum 2.0, launching between 2020 and 2022 to improve speed, scalability, and security (Guo and Yu 2022).

Nakamoto's foundational paper on Bitcoin, while not explicitly naming it a "blockchain," outlined the properties of modern blockchains as tamper-evident and tamper-resistant digital ledgers (Connors and Sarkar 2023). At its essence, blockchain is a decentralized, transparent, and verifiable ledger, with its most renowned application being in crypto-currencies like Bitcoin. However, its features of immutability and decentralization have found relevance in diverse industries, from healthcare to finance. As the technology acquire more attention, there's a growing emphasis on training students and professionals in its complexities, given the rising demand for blockchain expertise in the job market (Elliston et al. 2023).

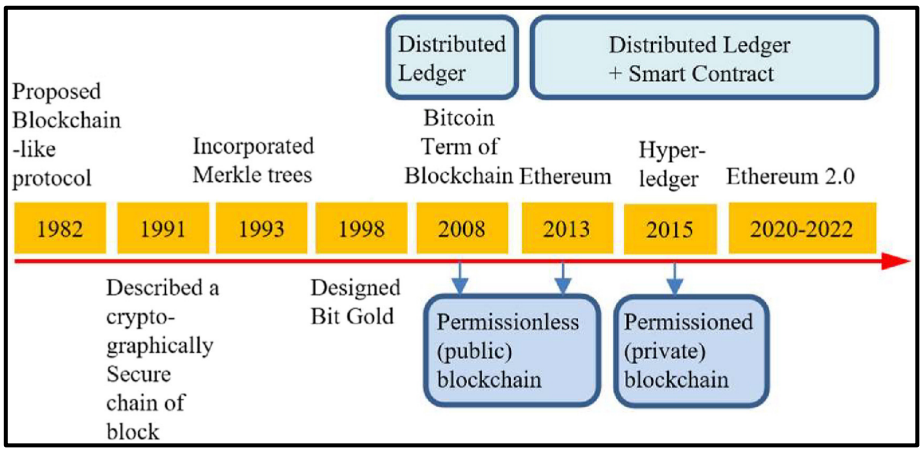


Figure 1.4 Comprehensive Blockchain's History
Taken from Guo and Yu (2022, p. 2)

1.2.1 Blockchain's characteristics

Based on the insights provided by (Rani and Saxena 2023; Sadawi et al. 2021; Zheng et al. n.d.), the following are the key characteristics of blockchain technology:

1. **Decentralization**: Blockchain does not rely on a central authority or administrator. Instead, it operates through a network of nodes, ensuring that no single entity has control over the entire blockchain. This structure contrasts with traditional centralized systems where a central agency, such as a bank, validates transactions. In blockchain, transactions can occur directly between peers without the need for central authentication, reducing costs and potential performance bottlenecks.
2. **Transparency**: All nodes in the blockchain network record and distribute data among each other. This ensures that data is available to every participant, promoting openness and traceability. Every transaction on the blockchain is validated, recorded with a timestamp, and can be easily verified and traced by accessing any node in the network.
3. **Autonomy**: Changes to the blockchain require the approval of the majority of nodes, ensuring that no single node can unilaterally alter the data.
4. **Immutability**: Once data is added to the blockchain, it cannot be altered unless someone controls more than 51% of the nodes. This feature, combined with the Blockchain's structure of linking blocks using hash algorithms, ensures that blocks cannot be modified or erased.
5. **Security**: Blockchain systems employ asymmetric cryptography, which includes a public key visible to everyone and a private key visible only to the owner. This ensures secure transactions and prevents tampering.
6. **Anonymity**: Blockchain supports privacy by authenticating transactions without revealing sensitive data of the involved parties. Users can interact with the network using generated addresses, and they can create multiple addresses to maintain privacy. However, perfect privacy preservation is not guaranteed due to certain constraints.
7. **Auditability**: Every transaction on the blockchain is time-stamped, allowing users to easily verify and trace previous records. In systems like the Bitcoin blockchain, each

transaction can be linked back to its predecessors, enhancing data traceability and transparency.

8. **Distribution**: Each node in the blockchain network holds a copy of the data records, which are continuously updated, ensuring data persistence and making tampering nearly impossible.
9. **Automation**: Blockchain can automatically trigger specific actions through smart contracts when predetermined conditions are met.
10. **Traceability**: Blockchain maintains a historical record of all data from its inception, allowing users to trace back to any original action.
11. **Privacy**: While blockchain operations are transparent, participant information remains anonymous using private/public keys.
12. **Reliability**: Due to its robust features and structure, blockchain has been successfully adopted by various organizations, showcasing its reliability.

In summary, blockchain technology offers a myriad of features that make it a revolutionary tool in various sectors. Its decentralized nature, combined with transparency, security, and traceability, among other features, provides a robust and reliable platform for various applications.

1.2.2 Blockchain architecture

Blockchain is a system structured in three main layers: applications, a decentralized ledger, and a peer-to-peer network. At its core, it is a sequence of blocks, each containing transaction records like a traditional public ledger. These blocks are cryptographically linked, with each block referencing the previous one through a hash value. This architecture ensures a consistent and tamper-proof global ledger where transactions are securely recorded and connected (Sarmah 2018; Zheng et al. n.d.).

The architecture of the blockchain is primarily divided into three layers: the top layer consists of applications, the middle layer is the decentralized ledger, and the lower layer is the Peer-to-

Peer Network. The application layer, for example, houses software like the Bitcoin wallet, which creates and stores keys, allowing users to oversee their transactions. The decentralized ledger in the middle layer validates a consistent and tamper-proof global ledger, where transactions are grouped into blocks that are cryptographically linked (Sarmah 2018). The working of a typical blockchain network involves users interacting with the blockchain using their keys, peers verifying the validity of transactions, transactions being mined into time-stamped blocks, and nodes verifying the proposed block's reference and its valid transactions All elements in the blockchain ensure the integrity and chain-like structure of the blockchain (Bhushan et al. 2021).

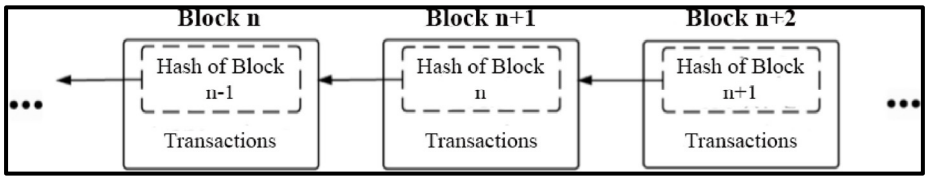


Figure 1.5 How Blockchain is Linked
Taken from Bhushan et al (2021, p. 3)

Blocks in a blockchain are data structures containing transaction records and headers. Key components of a block include the hash of the previous block, which links blocks together; a timestamp indicating when the block was created; a nonce used for block creation and validation; and the Merkle root, which simplifies transaction verification by containing hash values of all transactions (Rani and Saxena 2023).

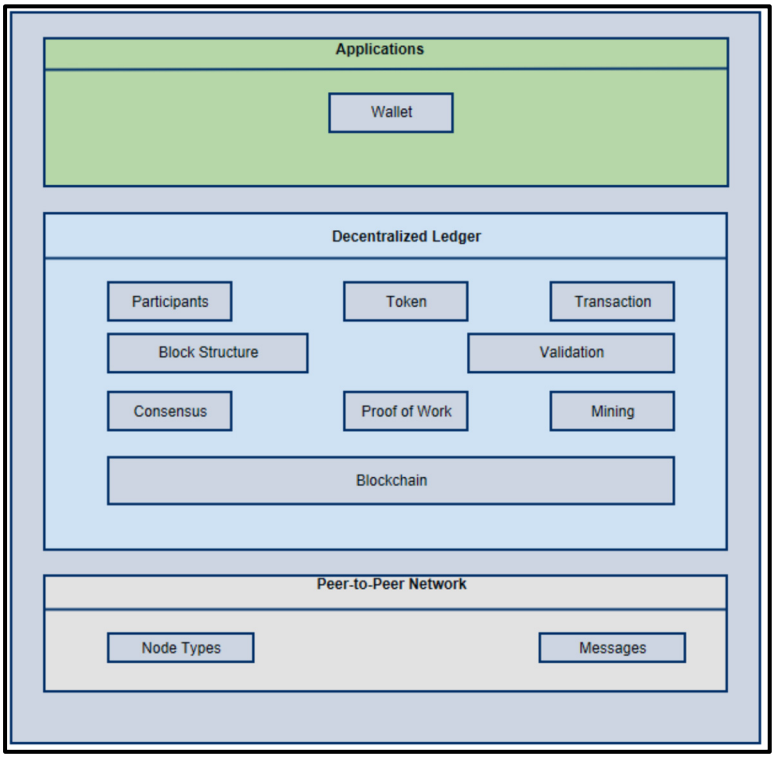


Figure 1.6 Blockchain Structure Overview
Taken from Sarmah (2018, p. 2)

1.2.2.1 Hash function definition

Hashing in computer science involves converting a character sequence, or key, into a typically shorter, fixed-length sequence called the hash value. This process uses a hash function, where the key is associated with specific data, aiding in data storage and retrieval. The primary advantage of hashing is efficiency; it allows for quicker searches by examining the hash value instead of the entire original string. Ideally, a hash function should produce a unique hash result for each key, ensuring consistent outputs for the same input and preventing overlap between different keys. While various hash functions exist for diverse purposes, each should adhere to this principle. However, a hash function suitable for database tasks might not be ideal for error-checking or cryptography (Sadeghi-Nasab and Rafe 2023).

Blockchain technology involves the creation of blocks that are added sequentially using cryptographic hash functions. Once a block is added, it cannot be altered without updating all subsequent blocks in the chain. This technology, exemplified by Bitcoin, records financial transactions in blocks, leading to its description as a distributed registry. Hash functions, traditionally, compress large data sets into smaller ones and have various applications, including in cryptography for ensuring data integrity and authentication (Belej, Staniec, and Więckowski 2020). Another crucial cryptographic method in blockchain is hashing, which encrypts data into a fixed-length string known as a hash. The SHA-256 algorithm, developed by the NSA in 2001, is a notable cryptographic hash function used widely in blockchain. It converts any data length into a fixed 256-bit (32-byte) string, producing a "hash" or "message digest" (Rani and Saxena 2023).

Applications of hash functions include (Sadeghi-Nasab and Rafe 2023):

- **File Verification**: Ensures message integrity and detects malicious changes using algorithms like MD5 and SHA-1.
- **Digital Signature**: Uses cryptographic hashes for signing messages, with SHA-256 being a popular choice.
- **Password Verification**: Stores only hash digests for security, using salts for added protection and algorithms like MD5 and SHA-2.
- **Proof of Work**: Deters network abuses and is key in Bitcoin mining, using algorithms like BLAKE2b and SHA-256.
- **File/Data Identifier**: Helps in identifying content in systems like Git and for quick data lookup in hash tables.

1.2.2.2 Consensus algorithm in Blockchain

In blockchain technology, achieving consensus among nodes that may not trust each other is akin to the Byzantine Generals Problem, where generals must agree on a strategy despite potential traitors sending conflicting messages. Given that blockchain networks are decentralized without a central authority to ensure consistency across nodes, protocols are

essential to maintain ledger uniformity (Zheng et al. n.d.). An indication of blockchain is user anonymity, which raises concerns about transaction honesty. To address this, consensus algorithms validate transactions by seeking agreement from all nodes. If consensus isn't achieved, the transaction is deemed invalid. It is generally believed that most nodes in a blockchain aim to maintain system integrity, making consensus algorithms vital for transaction validation and storage (Vyas and Deshmukh 2023).

Anonymity, while a sought-after feature, presents trust challenges. To ensure transaction legitimacy, every transaction is validated and then added to a block via consensus algorithms. These algorithms, central to blockchain transactions, set rules for participants in a decentralized system without universal trust. They operate on the principle that controlling more of a scarce resource grants greater influence over the blockchain (Guo and Yu 2022).

There are numerous consensus algorithms utilized in the blockchain realm. While many might be familiar with a variety of them, we will specifically delve into Proof of Work (PoW) and Proof of Stake (PoS) in detail. Apart from these two, some other noteworthy consensus algorithms include Delegated Proof of Stake (DPoS), Ripple's Consensus Algorithm, Tendermint, Proof of Burn (PoB), Proof of Capacity (PoC), Delegated Byzantine Fault Tolerance (DBFT), Directed Acyclic Graph (DAG), SIEVE , Proof of Elapsed Time (PoET), Raft, Byzantine Fault Tolerance (BFT), and Practical Byzantine Fault Tolerance (PBFT).

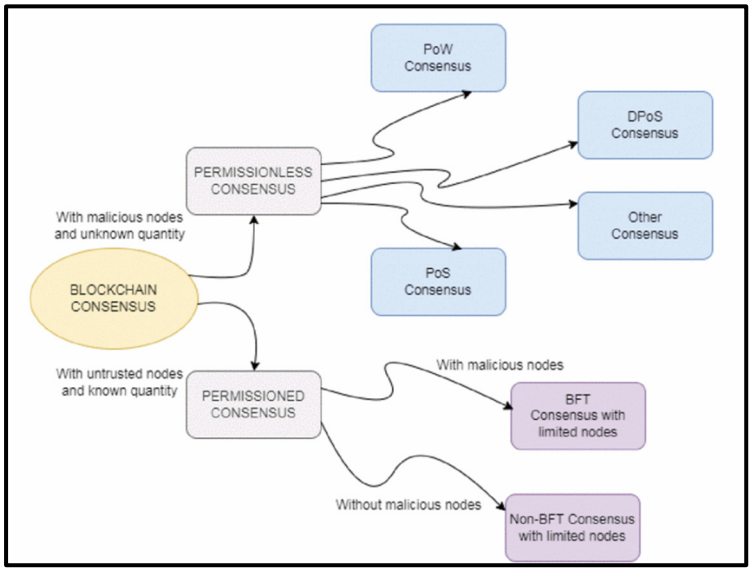


Figure 1.7 Consensus Algorithms Classification
Taken from K and S (2023, p. 3)

Proof of Work (POW) as consensus algorithm

Proof of Work (PoW) is a consensus mechanism integral to the Bitcoin network, necessitating complex computational tasks for authentication. In this system, nodes, termed "miners," continuously calculate hash values of a fluctuating block header, aiming for a value that meets a predefined target. Once achieved, other nodes validate its accuracy and the subsequent transactions in the new block. This process, known as "mining," is time-intensive, prompting incentives for miners, often in the form of Bitcoins (Zheng et al. n.d.).The network's open nature allows nodes to operate anonymously, yet in coordination, ensuring the Blockchain's growth even in the absence of some nodes (Vyas and Deshmukh 2023).

PoW's core involves problem-solving through guessing, specifically determining a nonce value that, combined with transaction data, satisfies a hash function's difficulty criteria. Nodes expend significant computational resources in this endeavor, with the successful node receiving a crypto-currency reward. Central to PoW is the hash function, with Bitcoin adopting the SHA-256 cryptographic variant. Both Bitcoin and Ethereum employ PoW as their consensus algorithm, but it's criticized for its high energy and time consumption (Guo and Yu 2022).

Proof of Stake (POS) as consensus algorithm

Proof of Stake (PoS) is an energy-efficient consensus mechanism that contrasts with Proof of Work (PoW). Instead of relying on computational power, PoS requires users to demonstrate ownership of a certain amount of crypto-currency, with the belief that those holding more currency have a vested interest in the network's security. However, selection based solely on account balance can lead to centralization, prompting various solutions like Blackcoin's randomization method and Peercoin's coin age-based selection. Ethereum, initially using PoW, plans to transition to PoS. To merge the advantages of both PoW and PoS, Proof of Activity and other variations like Proof of Capacity have been introduced (Zheng et al. n.d.).

PoS addresses the energy and time consumption issues of PoW. In PoS, nodes stake coins to be eligible as block creators, receiving transaction fees for valid blocks and facing penalties for invalid ones. Ethereum 2.0 marked a notable shift from PoW to PoS (Guo and Yu 2022; Vyas and Deshmukh 2023). PoS provides decision-making authority based on stake percentage, offering faster transactions and reduced energy use. Delegated Proof of Stake (DPoS) allows stakeholders to vote for representative nodes, speeding up transactions but risking centralization (Kim and Kim 2023). PoS, introduced by Peercoin, selects validators based on coin collateral and specific conditions, such as Coin-Age Based Selection and Random Block Selection, ensuring network security and reducing the risk of attacks (K and S 2023).

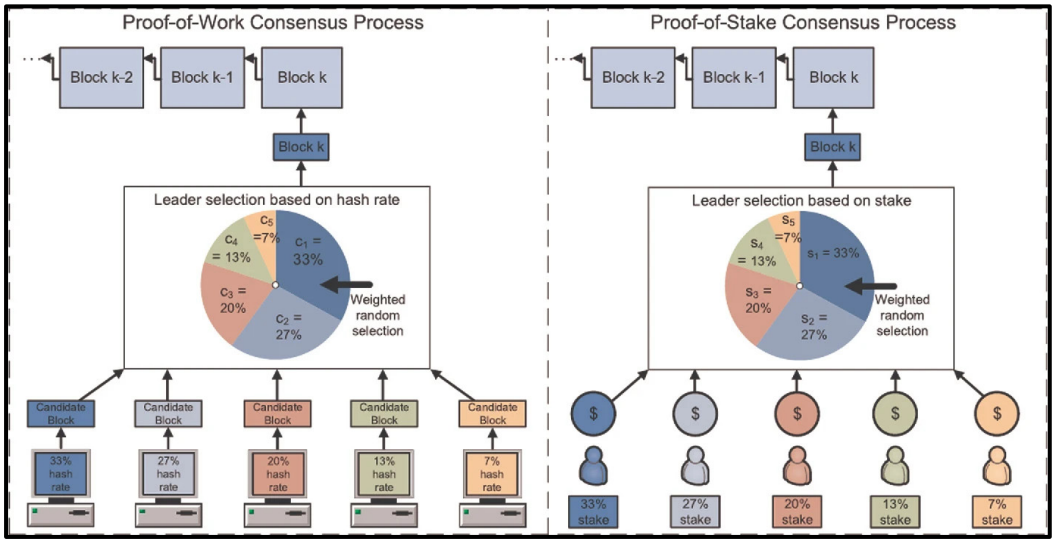


Figure 1.8 Compare POW and POS
Taken from Leo and Hattingh (2021, p. 571)

1.2.2.3 Smart contract

Smart contracts are Consider as contract, denoted as C, established by transaction participants u1, u2, ..., uk (where k is a positive integer). In the digital realm, this contract is represented as IC. When a trusted third-party entity, G, oversees this contract, the outcome of executing C is labeled as R. Thus, R can be expressed as R=C(U,G), and equivalently, R=IC(U) where U encompasses {u1, u2, ..., uk}. Smart contracts autonomously finalize transactions that, in the physical world, would necessitate oversight from a reliable third-party to guarantee their proper execution (Wu et al. 2022).

Smart contracts are automated digital agreements on the blockchain, with Ethereum being a notable platform using Solidity for their development. Introduced in the 1990s by Nick Szabo, they offer transparency by removing intermediaries and are immutable once activated. There are two categories: strong (tamper-proof) and weak. They're versatile, used in sectors like finance, real estate, and voting, and can operate on various blockchain networks. While they offer many advantages, they also face challenges in large-scale deployment. The choice of blockchain network for a smart contract depends on its intended application, considering

factors like security, transaction speed, and consensus protocol. Blockchain's decentralized and unchangeable nature enhances the security of smart contracts. Transaction speed is influenced by the contract's security and complexity, while consensus protocols like PoW and PoS determine transaction validity, with PoS offering better protection against threats (Alshahrani et al. 2023).

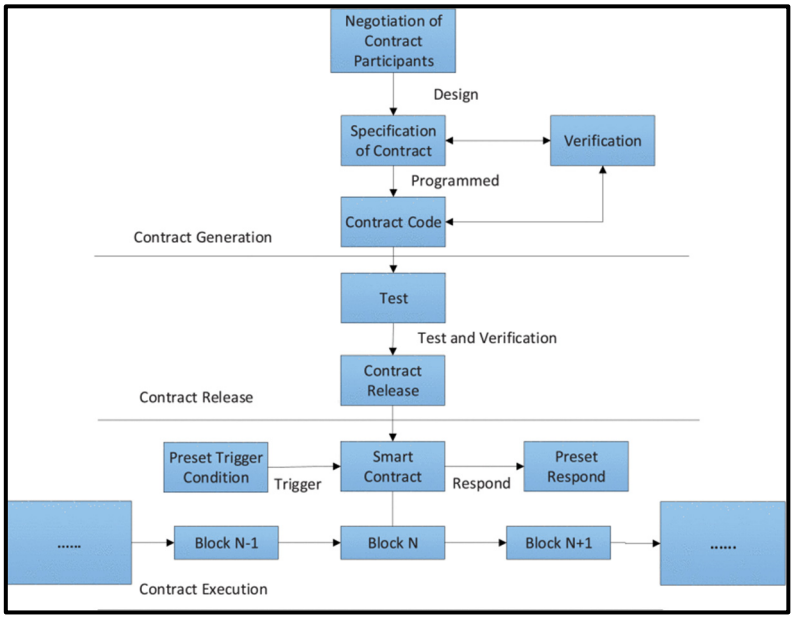


Figure 1.9 Evolutionary Phases of a Smart Contract
Taken from Wu et al (2022, p. 3)

When methods in a smart contract are called, the state updates. This leads to the creation of a block or transaction that reflects the state change. Subsequently, this change is recorded in the Blockchain, ensuring all miners are notified about the update resulting from the invoked method.

Smart contracts, especially on Ethereum, have three main components stored in the Blockchain (Pise and Patil 2022)

1. A balance, public key, and private key address. Users can transfer ether or money to this address.
2. Code specific to the smart contract, which miners initiate and store in the Blockchain.
3. A state component.

Here's an example of a smart contract using the SPESC language (Mao and Chen 2023):

```
Contract Purchase {
   Party Seller Address (ID) {
      Post()
      Collect()
   }
   Party Buyer {
      Pay()
      Receive()
   }
   // Definition of parties
   Asset Printer: Address (ID) {
      Name: "printer"
      Value: 1000$
   }
   // Definition of Asset
}
```

The seller initiates with post() after an order.

The buyer pays using pay().

After the buyer confirms receipt with receive(), the seller can access funds with collect.

1.2.3 Confidentiality Tiers in Blockchain

Depending on the required trust level in the network, the appropriate blockchain confidentiality is selected. This can be divided into two main categories:

1) **Public:** This allows any individual to join as a node in the shared blockchain and potentially influence decisions. However, participation might not always yield benefits for the users. The ledgers in this setup are not owned by any single entity and are open to all members. Blockchain applications employ a decentralized consensus mechanism and store

a copy of the ledger on local nodes (Sarmah 2018). This category further splits into (Dehez Clementi et al. 2019):

a) **Permissionless:** Any node with internet access and a computing device can contribute to the ledger and act as a validator, executing the consensus algorithm. All users have equal rights and permissions within the network.
b) **Permissioned:** While the network is open, only specific nodes have the capability to validate, with the rest serving as data repositories.

2) **Private:** These Blockchains are exclusive, accessible only to a chosen group of individuals or entities, with the database being shared solely among these participants(Sarmah 2018). This can be further divided into (Dehez Clementi et al. 2019) :
   a) **Permissionless:** Although the network requires authentication, once inside, all participants have equal read and write capabilities, akin to a company's internal network. Access is granted by a private organization, but any "verified" node can perform any function.
   b) **Permissioned:** Only a select few nodes can add content, and limited participants have access rights. This represents the most stringent blockchain type, where a third-party grants access and oversees permissions.

1.2.4 Blockchain based IoT (BIoT)

The Internet of Things (IoT) has significantly impacted various sectors, with its presence felt in areas like healthcare, agriculture, and asset tracking. Currently, over 20 billion IoT devices are in use, highlighting its widespread adoption. However, the technology faces challenges, especially concerning trust, security, and privacy. Blockchain, known for its decentralized nature and features like integrity and authentication, emerges as a potential solution to these challenges. It ensures data integrity, prevents potential security breaches, and its decentralized approach eliminates single points of failure in IoT systems. Furthermore, blockchain's success in the financial sectors underscores its potential in enhancing IoT's security and functionality.

Notably, initiatives like the Trusted IoT Alliance in 2016 have been formed to further the integration of blockchain into IoT (Aggarwal et al. 2021).

In the coming decades, both blockchain and IoT are poised to reshape production firms, enhancing productivity, fostering innovation, and improving efficiency. As the costs of sensors decrease, more manufacturing sectors are expected to adopt IoT. However, the vastness of IoT networks brings forth challenges in information security and privacy. While blockchain offers solutions, its resource-intensive nature might pose challenges for some IoT devices (Khare et al. 2022).

IoT's rapid growth has been impaired by issues like a lack of trust and an over-reliance on centralized authorities. This centralization poses risks, especially concerning data control and security. Blockchain, with its decentralized and trustless environment, addresses these concerns, providing enhanced security and efficiency. The two technologies, while having their unique features, can be integrated effectively for better results. The Table 1.4 provides a concise comparison of challenges faced by centralized IoT systems and the corresponding solutions offered by Blockchain technology (Sharma and Babu Battula 2022).

Table 1.4 Challenges in IoT and Blockchain Solutions
Adapted from Sharma and Babu Battula (2022, p. 55)

Issue in Centralized IoT	Blockchain (BC) Solution
Security	BC enhances security using public-key infrastructure, ensuring data integrity through immutable records and cryptographic communication.
Scalability	BC's distributed nature efficiently manages the increasing number of IoT devices.

Issue in Centralized IoT	Blockchain (BC) Solution
Point of Failure	BC's decentralized communication eliminates central server reliance, preventing single points of failure.
Address Space	BC offers a vast address space, surpassing even IPv6 by over 4 billion addresses.
Authentication & Access Control	BC provides robust identity management, decentralized authentication, and access control via smart contracts.
Data Integrity	BC ensures data remains unaltered unless verified by the majority of network participants.
Manipulation Vulnerability	BC's immutable environment prevents data manipulation, ensuring data integrity.
Ownership & Identity	BC offers reliable identity registration and ownership tracking, proven effective in goods monitoring.
Traceability	BC supports scalability across various open-source options and meets the demands of a scalable structure.
Cost & Storage Constraints	BC's peer-to-peer architecture negates the need for central authorities, reducing costs and hardware requirements.

1.2.4.1 Blockchain based IoT Challenges

Based on (Aggarwal et al. 2021; Garg et al. 2022) the integration of Blockchain with the Internet of Things (BIoT) faces several challenges:

- **Scalability:** Adding new blocks or transactions to a blockchain introduces delays due to validation requirements. As the number of nodes increases, real-time data

transmission, especially in applications like the food supply chain, becomes challenging.

- **Computational Resources:** Different consensus algorithms, such as PoW used by Bitcoin, demand varying computational strengths. Even less intensive algorithms than PoW still require significant computational resources.
- **Storage Size:** Blockchains have full nodes, which store the entire blockchain, and lightweight nodes with only block headers. Given the limited storage of many IoT devices, accommodating both types of nodes is problematic.
- **Energy Efficiency:** Blockchains consume significant power because of mining and continuous P2P communication.
- **Security:** Ensuring confidentiality, availability, and integrity is crucial for any information system, with data integrity being especially vital in the IoT context.
- **Privacy:** In IoT environments, maintaining user privacy during transactions is a primary concern, addressed by techniques like Zero-knowledge proof.

Figure 1.10 provides a comprehensive overview of the challenges associated with integrating Blockchain and IoT.

Figure 1.10 Comprehensive BIoT Challenges
Taken from Alzoubi et al (2022, p. 15)

1.2.5 An overview of Hyperledger project

In 2015, the Linux Foundation introduced the Hyperledger project, an open source blockchain software distinct from Bitcoin and Ethereum. It encompasses eight blockchain frameworks, five tools, and four libraries (Guo and Yu 2022). Hyperledger is a collaborative platform designed to support cross-industry blockchain developments, involving around 100 industry leaders. It's a permissioned blockchain that emphasizes access control, chain code-based smart contracts, and adaptable consensus methods, enhancing security and preventing attacks like Sybil attacks. Smart contracts in Hyperledger offer rapid execution times, making it suitable for IoT applications. Among its various constructions, Hyperledger Fabric is notably popular and open source. Studies comparing Ethereum and Hyperledger found that Hyperledger Fabric consistently outperforms Ethereum in latency, execution time, and throughput(Sharma and Babu Battula 2022).

1.2.6 Hyperledger Frameworks

The Hyperledger approach promotes the reuse of standard components, facilitates swift component development, and fosters interoperability across projects. Unlike public ledgers like Bitcoin and Ethereum, Hyperledger's business blockchain systems are designed for a consortium of organizations. Hyperledger nurtures and advances various business blockchain technologies, such as test applications, distributed ledger frameworks, smart contract engines, utility libraries, graphical interfaces, and client libraries (Punathumkandi, Meenakshi Sundaram, and Prabhavathy 2020).

1.2.6.1 Hyperledger-Fabric

Hyperledger Fabric, an open-source enterprise solution, is a permissioned distributed ledger technology (DLT) designed for business applications. It offers a unique, modular, and configurable architecture suitable for various industries, including banking, healthcare, cinema, IoT, and supply chain. Fabric's permissioned nature means participants are known to each other, fostering a governance model built on mutual trust. It introduces a novel transaction design, separating the transaction flow and creating channels for specific member coordination, ensuring privacy in competitive scenarios. The ledger in Fabric comprises the world state, representing the current state, and the transaction log, detailing the update history. Smart contracts in Fabric, termed chaincode, interact primarily with the world state and can be written in multiple languages, including Go and Node (Punathumkandi et al. 2020).

(Capocasale, Gotta, and Perboli 2023) highlight Fabric's role in addressing industrial needs like identity management, role definition, and data confidentiality. As part of the Hyperledger ecosystem, Fabric supports a modular and scalable structure, allowing for private transactions or the creation of independent chains. It employs various consensus algorithms, with a BFT consensus planned for future releases. Fabric differentiates between two node types: peers, responsible for transaction execution and ledger maintenance, and Orderers, tasked with block creation. The transaction process in Fabric involves three stages:

i) **Execute:** Transactions are associated with an endorsement policy, defining which peers must execute a given transaction. Clients send the transaction to endorsing peers, who process it without updating their ledger copy, then return a signed message to the client for Orderers. (Xu et al. 2021) refer to this stage as the "Simulation Phase."

ii) **Order:** Orderers create blocks by organizing the endorsed transactions received. Once a block is formed, it's broadcasted to all channel peers.

iii) **Validate:** Each peer verifies the correctness of each transaction within the received block and updates its ledger copy. Transactions conflicting with a previous transaction in the same block are deemed invalid.

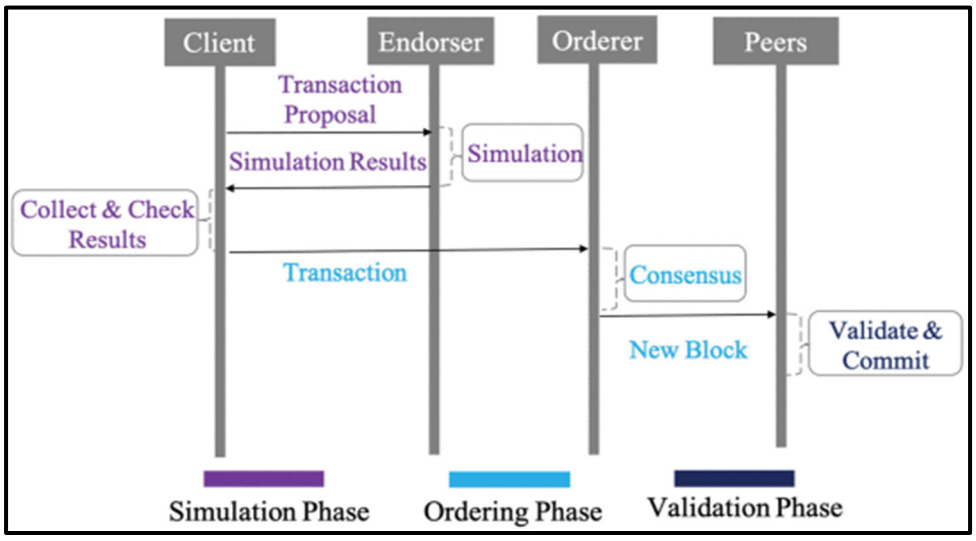


Figure 1.11 Data flow in Hyperledger Fabric
Taken from Xu et al (2021, p. 3)

(Palma, Pareschi, and Zappone 2021) recognize Hyperledger Fabric as a leading project within the Hyperledger suite, emphasizing its modularity and adaptability. It is designed as a foundation for building applications with a modular setup, allowing components to be interchangeable. Fabric's unique consensus approach ensures scalability while maintaining privacy.

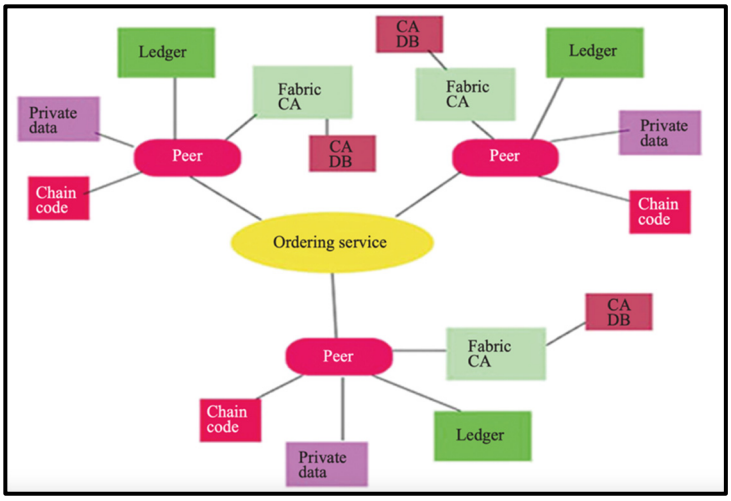


Figure 1.12 Hyperledger Fabric Architecture
Taken from Punathumkandi et al (2020, p. 90)

In summary, Hyperledger Fabric is a versatile and modular DLT solution tailored for enterprise applications, emphasizing privacy, scalability, and adaptability across various industries, with a distinct three-step transaction process.

1.2.6.2 Hyperledger project comparison

The Hyperledger project, hosted by the Linux Foundation, is a collaborative effort to advance cross-industry blockchain technologies. It encompasses a suite of frameworks, tools, and libraries designed to support the development of blockchain-based distributed ledgers for various business applications. Each framework within the Hyperledger umbrella offers distinct features, architectures, and capabilities tailored to meet specific industry needs and use cases. Based on (Capocasale et al. 2023; Palma et al. 2021; Punathumkandi et al. 2020) , the Table 1.5 provides a comparative overview of several prominent Hyperledger frameworks, highlighting their key technical attributes and functionalities.

Table 1.5 Comparison Between Hyperledger Projects

Hyperledger Project	Description	Key Features	Contribution
Hyperledger Fabric	General-purpose blockchain framework designed for modularity and configurability.	Modular architecture that separates the transaction flow into execute-order-validate.	(Palma et al., 2021)
Hyperledger Sawtooth	Open-source framework designed for flexibility and separation of concerns.	Abstracts the application layer from the security layer, dynamic & replaceable components.	(Capocasale et al., 2023)
Hyperledger Indy	Focuses on Decentralized Digital Identities (DIDs).	Provides tools, libraries, and reusable components for digital identities rooted on Blockchains or other distributed ledgers.	(Palma et al., 2021)
Hyperledger Iroha	Designed for simplicity and integration into infrastructure or IoT projects.	Core architecture inspired by Fabric. Emphasizes user-friendly interfaces and interoperability with other Hyperledger projects.	
Hyperledger Besu	Ethereum client designed for both public and private use cases.	Java-based and offers comprehensive permissioning schemes.	(Punathumkandi et al., 2020)

Hyperledger Project	Description	Key Features	Contribution
Hyperledger Burrow	Provides a permissioned smart contract interpreter built to the specification of the Ethereum virtual machine (EVM).	Extends work within the Hyperledger Project by providing a deterministic smart contract-focused blockchain structure.	(Punathumkandi et al., 2020)
Hyperledger Grid	Domain-specific platform for building supply-chain management solutions.	Provides components for developing smart contracts and client interfaces	(Palma et al., 2021)

In summary, while Hyperledger Fabric is a versatile and modular framework suitable for a wide range of industry use cases, other Hyperledger projects like Sawtooth, Indy, Iroha, Besu, Burrow, and Grid have specific focuses and features that differentiate them from Fabric. Each project has its unique strengths and is designed to address particular challenges or industry needs. Also, based on (Capocasale et al. 2023), Fabric's growing support, as evidenced by the developer activity on GitHub, indicates its increasing popularity and trustworthiness within the Blockchain community.

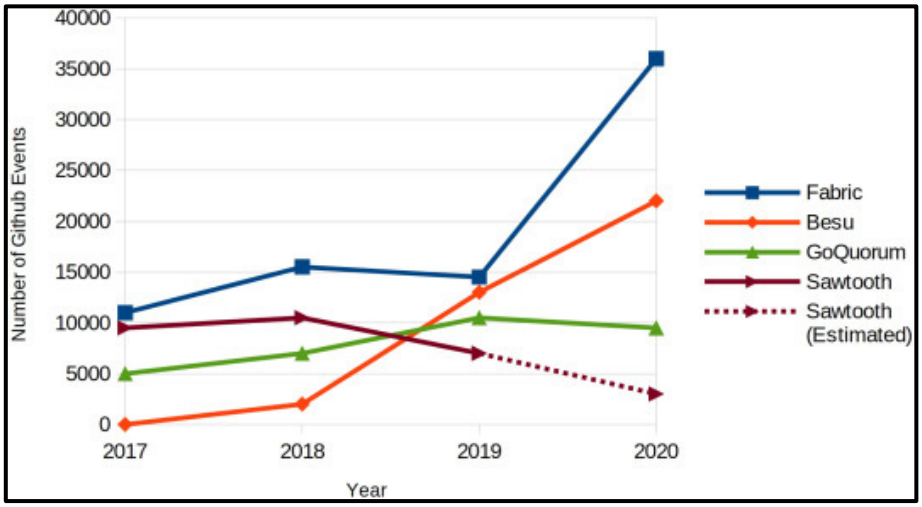


Figure 1.13 Blockchain Development Engagement
Taken from Capocasale et al (2023, p. 6)

1.2.7 Recent Developments on Blockchain and IoT innovation

In recent years, the convergence of blockchain technology and the Internet of Things (IoT) has emerged as a significant area of research and development. This integration aims to address some of the key challenges faced by IoT systems, such as data security, privacy, and scalability. Blockchain, with its decentralized nature and robust security mechanisms, offers a promising solution to these issues. The following section provides an overview of recent developments in the integration of blockchain technology within IoT systems, focusing on innovative approaches and architectures that have been proposed to enhance the performance and security of IoT networks.

In this research paper (Oikonomou et al. 2021), the authors introduce two novel concepts: Storing sensor data externally to the blockchain, while retaining only the sensor's identification and transaction details within a local blockchain. Implementing "RESET" transactions that transfer data from the local blockchain to liberate storage space, while storing the hash in the global blockchain (GB) to maintain data integrity. These innovative ideas link local blockchains to the Global Blockchain (GB), ensuring data integrity and implementing specific

chaincode and policies. Collectively, these strategies enhance scalability and efficiency in managing data storage and transactions.

Another study (Maeng, Heo, and Joe 2022), delves into a system configuration comprising a root server, Hyperledger Fabric, and various users. It delineates the user registration and verification process, designating users as agents and emphasizing the role of distinct certificate authorities. The system supports the establishment and administration of multiple groups, with regular monitoring and agent alterations to bolster stability and security.

The research article by (Al-Zoubi et al. 2022), presents a pioneering architecture that employs blockchain technology to facilitate interaction among IoT devices. This architecture is structured into four blocks: Sensors, webservice, ETH blockchain, users, and administrators. The system has been proven to function effectively under diverse conditions, demonstrating the secure and efficient application of blockchain technology. Nevertheless, it faces challenges such as dependency on a private ETH blockchain with limited transaction processing capacity, high costs associated with implementation on a public ETH blockchain, and the absence of encryption, privacy, and security measures on the device side.

In their paper (Su, Nguyen, and Sekiya 2022), the authors concentrate on the Combination of IoT systems with a private blockchain deployed on an ad-hoc IoT network. The preference for a private blockchain is justified by its benefits, such as a reduced number of nodes and lower power and resource consumption. The research highlights the establishment of an Ethereum-based private blockchain atop the network, with each IoT device employing the Geth client to establish a full blockchain connection with other nodes. The study aims to evaluate the performance of the integrated IoT-private blockchain system, particularly focusing on the connections between IoT devices and the underlying network.

As detailed in the article by (Dange and Nitnaware 2023), the authors explore the advantages of data storage and compare the associated gas prices in ETH. They emphasize the importance of pre-processing at the fog layer, which are governed by the administrator's policies. Users

can access blockchain, with smart contracts employed for validation purposes. However, some limitations are identified, such as the use of the ETH blockchain leading to low transactions per second (TPS) and the lack of a specific data structure for managing data.

The integration of blockchain technology into IoT systems has led to the development of various innovative solutions aimed at improving data integrity, security, and efficiency. These include off-chain data storage strategies, the use of private blockchains for reduced resource consumption, and the implementation of new architectures for secure interactivity among IoT devices. Despite the promising advancements, challenges such as scalability, transaction processing capacity, and the management of data privacy remain. Future research in this domain is expected to address these limitations and further refine the integration of blockchain technology with IoT systems, paving the way for more secure, efficient, and scalable IoT networks.

1.3 Chapter summary

In the initial section of this chapter, we delved into the architecture of the Internet of Things (IoT) and explored various IoT platforms. The advantages of real-time data streaming were underscored, and a comparative analysis was provided. Towards the end of this section, we examined the prevailing challenges inherent in the IoT infrastructure.

In the subsequent section, we turned our focus to the architecture of Blockchain. A comprehensive exploration of blockchain and its associated algorithms was undertaken, highlighting the unique attributes that make it a compelling choice. Furthermore, we discussed the convergence of blockchain and IoT, often referred to as BIoT, and addressed the potential challenges this integration might encounter. Additionally, we introduced one of the most renowned types of blockchain.

CHAPTER 2

Methodology and Design of architecture

As we delve further into the intricacies of integrating devices with Blockchain, this chapter will illuminate the architecture we have meticulously designed. Central to our exploration is understanding how each device can be smoothly linked to the blockchain, enabling data visualization, and facilitating user access to the platform.

In this chapter, we will:

1. Explore the multi-layered structure of the architecture, emphasizing the role and significance of each layer.
2. Delve into the types of devices incorporated into this framework and elucidate the design principles behind their interconnections.
3. Analyze the components integral to each layer, offering insight into their functionality and contribution to the overall system.

By the chapter's conclusion, readers will be equipped with a comprehensive understanding of our architecture's capabilities and the advantages it offers. A summary will encapsulate the key benefits of this platform, serving as a succinct reference for the insights gained.

2.1 Proposed architecture

Within complex systems, the architecture plays a vital role. It defines the interactions and operations of individual components, ultimately determining the system's overall performance. In the realm of the Blockchain-based Internet of Things (BIoT), comprehending the architecture is essential. Similar to a blueprint which provides clarity on a building's structure, understanding the architecture illuminates the engineering principles and potential of the BIoT system.

This chapter delves into the architecture we have developed. It emerges from extensive research and numerous design iterations, showcasing our approach to connecting devices to the blockchain with an emphasis on real-time data processing and efficient data visualization. We will systematically explore each aspect of our design, detailing the rationale behind every decision and elucidating how each component contributes to the robustness and innovation of our architecture.

Our proposed system is structured around a four-layer architecture, with each layer possessing distinct functionality and significance. The layers are as follows:

1. **Device Layer (Physical Layer):** This foundational layer encompasses the tangible components and devices.
2. **Network Layer:** Serving as the communication bridge, this layer ensures connectivity between layers.
3. **Middleware Layer:** Acting as an intermediary, this layer facilitates data processing and manages interactions between the system's components.
4. **Application Layer:** This topmost layer provides the user interface and ensures that end-users can efficiently interact with the system.

In subsequent sections, we will delve deeper into the details of each layer, elucidating their design choices, functionalities, and the roles they play in the overarching architecture.

Table 2.1 Proposed Four-Layered Architecture in Detail

Layer	Examples
Fourth Layer (Application/End-User)	Smart home systems, industrial automation tools, telehealth platforms, asset monitoring solutions, smart city applications

Layer	Examples
Third Layer (Middle layer)	Data warehousing, analytics processing, data transformation, security protocols, device management
Second Layer (Network)	Ethernet, Wi-Fi 6, Bluetooth 5, Zigbee 3.0, LoRaWAN, Cellular networks (4G LTE, 5G NR)
First Layer (IoT Device/Physical)	Infrared sensors, Proximity sensors, Pressure sensors, Gas leak sensors, Smart cameras, Intelligent lighting systems, connected vehicles, Drones, Airplanes

Figure 2.1 presents the details inside each component and how they will communicate within.

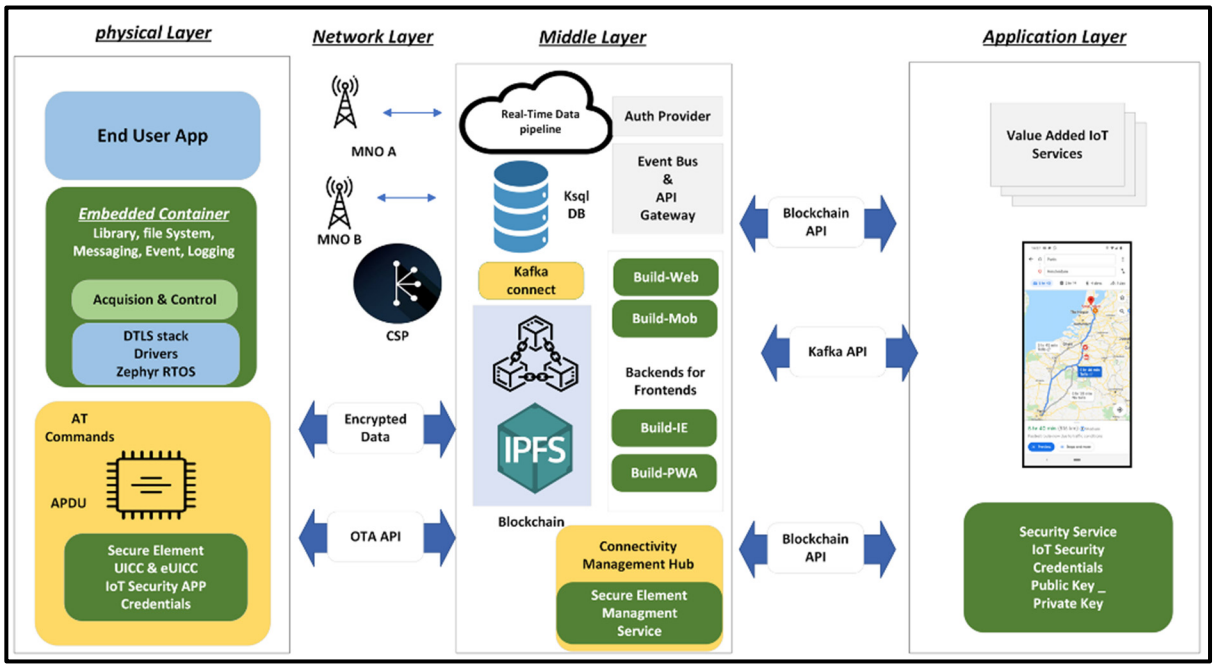


Figure 2.1 Four-Layered Architecture Components'

2.2 First Layer (physical layer)

The physical layer, also referred to as the first layer, serves as the primary interface for real-world data acquisition in our system. Within this layer, devices can be broadly classified into two principal categories:

1. Sensors (or Devices)
2. Gateways

Each of these categories possesses distinct functionalities and roles:

Sensors (or Devices): Sensors are specialized components designed for specific data acquisition tasks. Depending on their design specifications, they can measure a variety of environmental parameters such as humidity, temperature, water presence, door statuses (open or closed), light intensity, and physical impacts. The choice of data transmission protocol for these sensors is contingent upon their operational requirements and the nature of the data they capture. Commonly employed protocols include Bluetooth and Wi-Fi, Zigbee, and Z-Wave, among others.

Gateways: Gateways function as intermediaries in the system. Their primary role is to aggregate data from the sensors, potentially preprocess it, and then transmit it to the subsequent layers in the system's architecture. To ensure efficient and reliable data transmission, gateways utilize protocols tailored to the system's data transfer requirements, such as LwM2M and MQTT, CoAP, AMQP and cellular networks.

2.2.1 Case studies

In our previous discussions, we touched upon the types of devices and gateway variations available for use in IoT configurations. This vast landscape of choices has empowered developers and organizations to craft solutions tailored to specific needs. In this section, we will delve into some of the most common pairings of devices and gateways that have proven effective in real-world scenarios. These combinations have not only demonstrated their

reliability but also their adaptability across various applications. Below, we outline these exemplary pairings.

2.2.1.1 Pinnacle 100 and sensors pairing

Within this section, an in-depth exploration is presented concerning the integration methodologies of sensors and gateways in the context of an Internet of Things (IoT) configuration. The BL654 model was meticulously selected as the primary sensor due to its adeptness in monitoring three critical environmental parameters: humidity, pressure, and temperature. It is imperative to acknowledge that, notwithstanding the emphasis on these three sensing modalities, the inherent flexibility of IoT infrastructures permits the integration of an expansive array of sensors, each tailored to the nuanced requirements of distinct applications. Regarding the gateway component, the Pinnacle 100 was chosen predicated on its integral SIM card feature, ensuring an unbroken and consistent connectivity continuum. In the quest to refine data transmission at the physical stratum, the Bluetooth communication protocol was deemed optimal. Such a decision was instrumental in facilitating a streamlined data transmission from the BL654 sensors to the Pinnacle 100 gateway.

Furthermore, the study incorporated the lightweight machine-to-machine (LwM2M) protocol to effectuate the data relay from the Pinnacle 100 to the ensuing network layer. The system, in its entirety, harnesses the capabilities of 4G or 5G networks to establish internet connectivity and subsequently relay the data to the designated network layer.

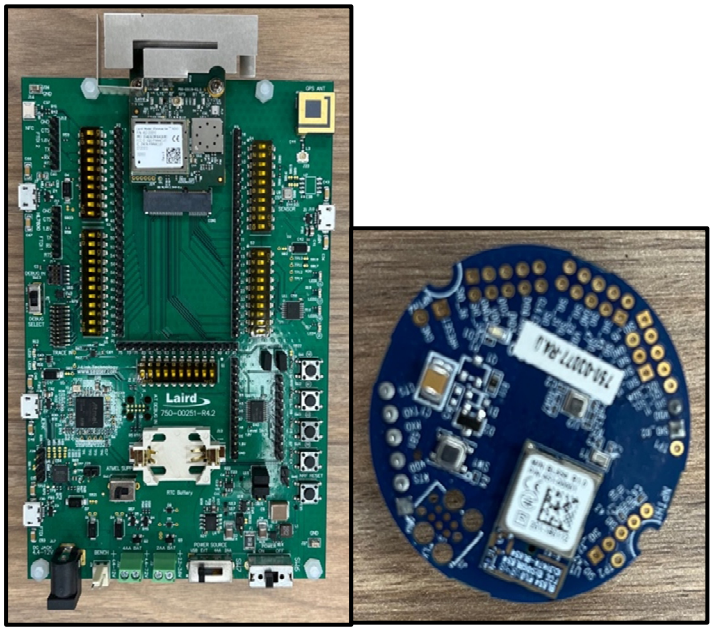


Figure 2.2 Left Image Shows Pinnacle 100
and
right Image Shows BL6545 Sensor

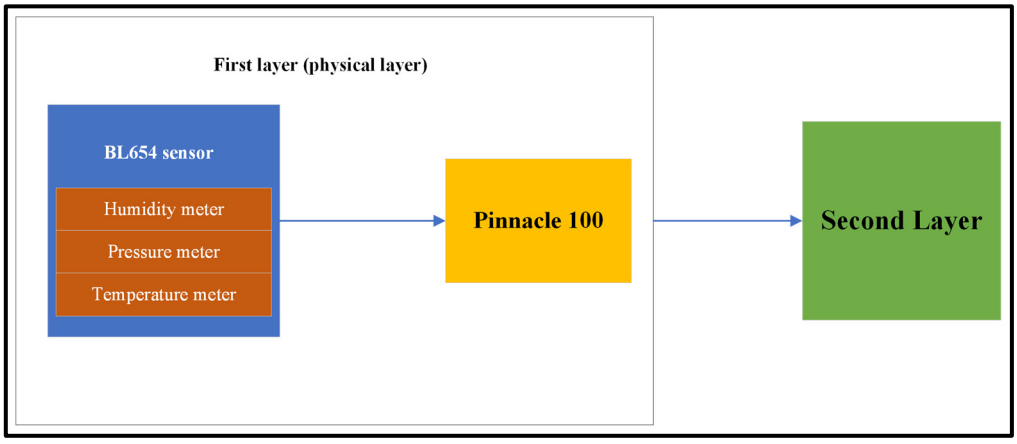


Figure 2.3 Data flow in This Case Study

2.2.1.2 Raspberry pie and sensor pairing

This section provides high-level overview of integrating the ib-nav sensor, a product of LASSENA's research endeavors, with the Raspberry Pi platform. The core focus is on the

sensor's unique indoor navigation capabilities predicated on the zero-velocity principle and the underlying mechanisms supporting data transfer and visualization.

Originating from the laboratories of LASSENA, the ibNav 6.1 sensor represents a significant advancement in indoor navigation technology. Distinct from conventional sensors, the Ibnav6.1 operates on the zero-velocity principle, enabling it to navigate indoor spaces devoid of external motion cues. This capability addresses the longstanding challenges posed by environments with weak or absent GPS signals.

A salient feature of the ibNav 6.1 sensor is its wireless data transmission functionality. Upon activation, the device channels its data via a WiFi conduit directly to a designated Raspberry Pi module. This mechanism guarantees a robust and consistent data flow, mitigating risks associated with data latency or integrity compromise.
Subsequent to the Raspberry Pi's data acquisition, the interfacing occurs with an advanced ground control station. Far from being a mere data receptacle, this station undertakes rigorous data processing and visualization tasks. The objective is to furnish users with an accurate and comprehensive depiction of the sensor's output, catering to both real-time surveillance and retrospective analyses.

To fortify the data communication pipeline, the MQTT (Message Queuing Telemetry Transport) protocol was incorporated. Renowned in academic circles for its lightweight nature, MQTT emerges as an optimal choice for scenarios with bandwidth constraints. The protocol not only ensures the efficient relay of data from the Raspberry Pi to subsequent layers but also underpins the system's reliability.

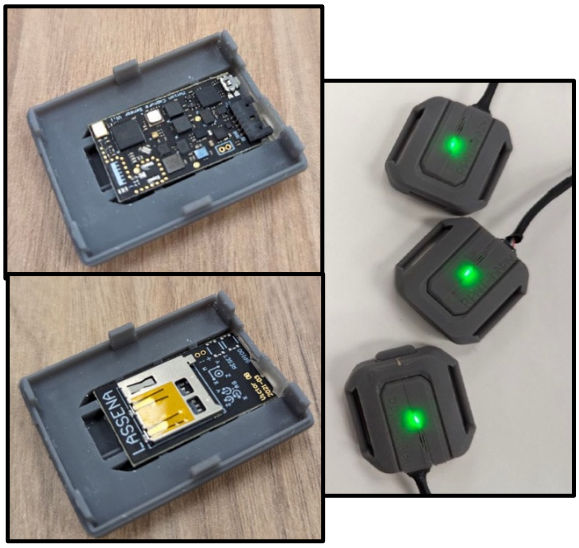


Figure 2.4 IbNav Sensor 6.1

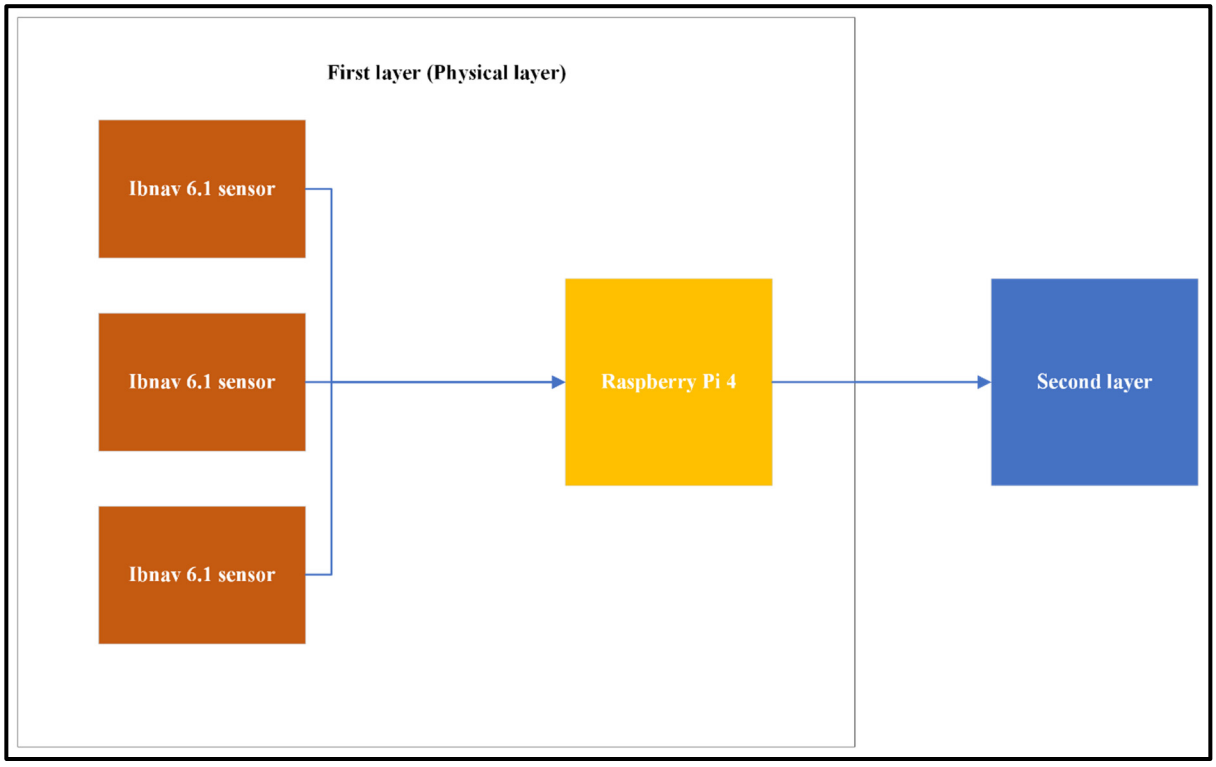


Figure 2.5 Data Flow in This Case Study

2.2.1.3 Automatic Dependent Surveillance-Broadcast (ADS-B) as a device

In this section, we present a case study that has been a focal point of our research. Initially, we will elucidate the concept of ADS-B and subsequently address its known vulnerabilities.

ADS-B is an air traffic management system that allows aircraft to broadcast their location, speed, and direction using an onboard GPS receiver. It has two main components: "ADS-B OUT" which sends data to ground stations, and "ADS-B IN" which receives data from other aircrafts. While it offers advantages like increased protection and airspace optimization (Costin and Francillon n.d.), ADS-B is vulnerable to various attacks . These attacks can be categorized by intent, such as passive information collection, financial disruption, terrorist threats, and state-sponsored cyber attacks. Common attack methods include message injection, modification, deletion, jamming, and eavesdropping (Wu, Shang, and Guo 2020). The system's vulnerabilities emphasize the need for improved security in air traffic systems.

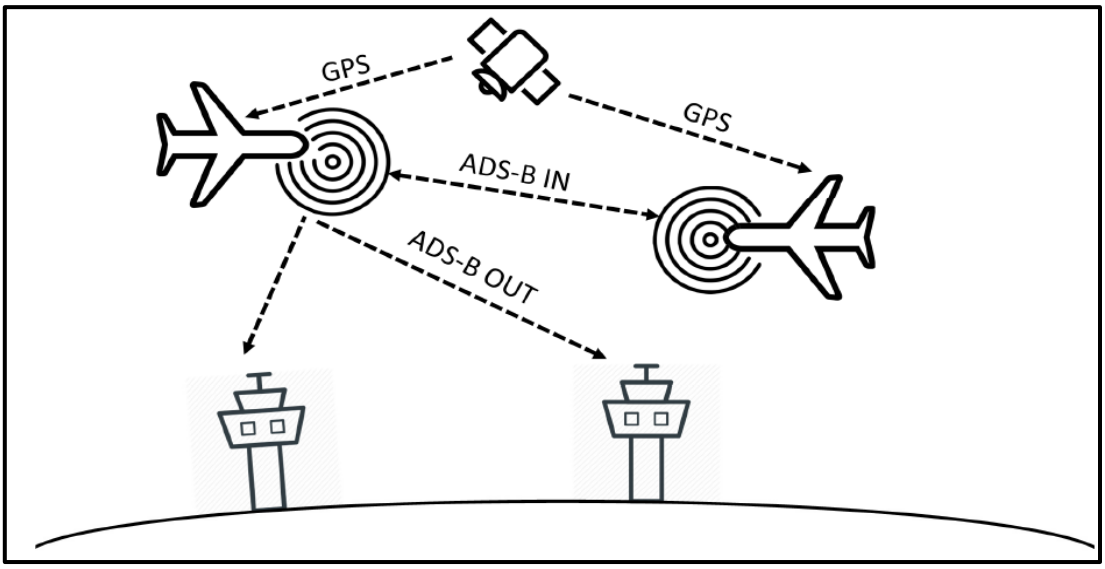


Figure 2.6 Overview of the ADS-B Communication
Taken from Sciancalepore, Alhazbi, and Di Pietro (2019, p. 3)

Building on this understanding, we postulate that beyond conventional devices such as standard sensors, mobile phones, and GPS systems, aircraft can also be viewed as devices in

their own right. A significant insight from our research is the effective utilization of ADS-B as a sensor. To facilitate data transfer from ADS-B, we employed the Message Queuing Telemetry Transport (MQTT) protocol, channeling the data to a subsequent analytical layer. The overarching objective of this case study is to augment the security measures of the existing ADS-B system by integrating our novel platform.

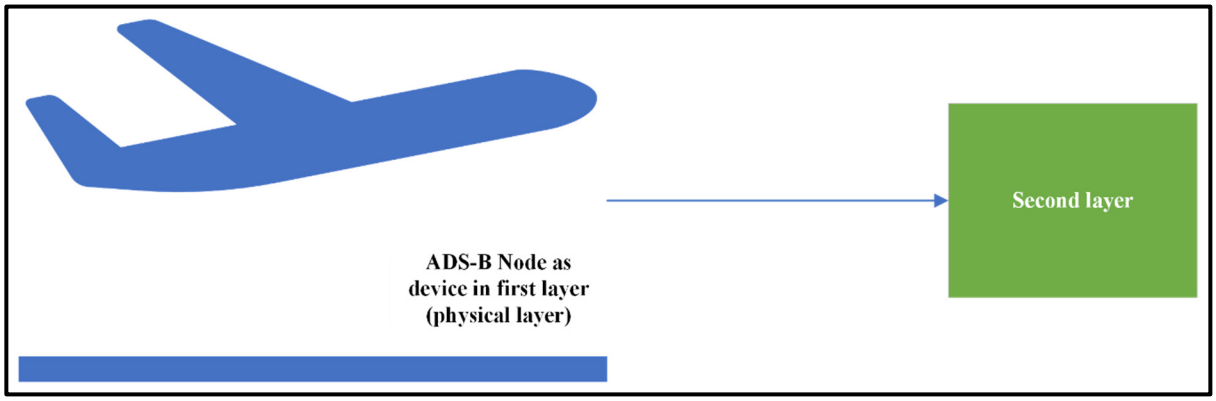


Figure 2.7 ADS-B Inside an Airplane Works as Sensor

2.3 Second Layer (Network layer)

Within the ensuing discourse, our primary objective is to elucidate the relationship between the physical layer and what is commonly referred to as the third layer. This particular layer can be conceptualized as a critical bridge, facilitating communication between the foundational physical layer and the more abstract middleware layer. In the realm of data communication, this layer shoulders a paramount responsibility. It is entrusted with the task of garnering data, be it from expansive gateways or from singular sensor units, and subsequently channeling this data upwards to the middle layer stratum. Once in the middle layer, the raw data undergoes rigorous analytical procedures and is then earmarked for storage, ensuring its availability for future requisition.

It is noteworthy to highlight that, as discussed in preceding sections, this layer is the principal domain wherein specific communication protocols, such as MQTT, find their operational ground. However, the versatility of this layer is further exemplified by its capability to harness

modern telecommunication infrastructures, particularly 4G and 5G networks, for data transmission in certain specialized scenarios.

2.3.1 Case Study

In this subsection, we will present a detailed examination of a case study associated with the network layer's implementation. Our research endeavors involved testing multiple protocols to determine the most effective means of transferring raw data from various sensors to the middle layer. Notably, one such case study centered on the utilization of PicoLTE as a potential solution. Beyond this, our team also invested efforts into evaluating other prominent protocols, specifically LwM2M and MQTT. Through a comprehensive assessment of these protocols, we sought to identify their respective strengths and limitations in the context of data transfer from sensors to the middleware infrastructure.

2.3.1.1 Pico LTE as network layer

Within the second layer, we utilize the Nutaq PicoLTE to facilitate data transfer from Pinnacle to the middle layer, capitalizing on the advantages offered by PicoLTE. In this specific use case, two PicoLTE devices are employed.

The Nutaq PicoLTE 2nd Generation emerges as a remarkable topic of interest within the field of software-defined radio and integrated network research. This system epitomizes the integration of real-time Long-Term Evolution evolved NodeB (LTE eNodeB) and Evolved Packet Core (EPC) functionalities within a singular framework, all while adhering to the 3GPP LTE PHY standards, specifically Rel. 13 and its successors. Notably, its design offers compatibility with a diverse range of commercial User Equipment’s (UEs) and is certain to accommodate emerging communication paradigms. From a research perspective, its cost-effectiveness is intriguing, suggesting a potential avenue for institutions operating under budgetary constraints. The device's technical attributes, spanning a comprehensive frequency range, integrated transceivers, and flexible bandwidths, warrant a detailed exploration. This is

further underscored by its provision for both onsite and online training, which can be instrumental in facilitating academic research and study (Nutaq Team, n.d.)

In summary, the Nutaq PicoLTE 2nd Generation offers a rich tapestry of features and capabilities, meriting its examination and study within the broader context of LTE network research.

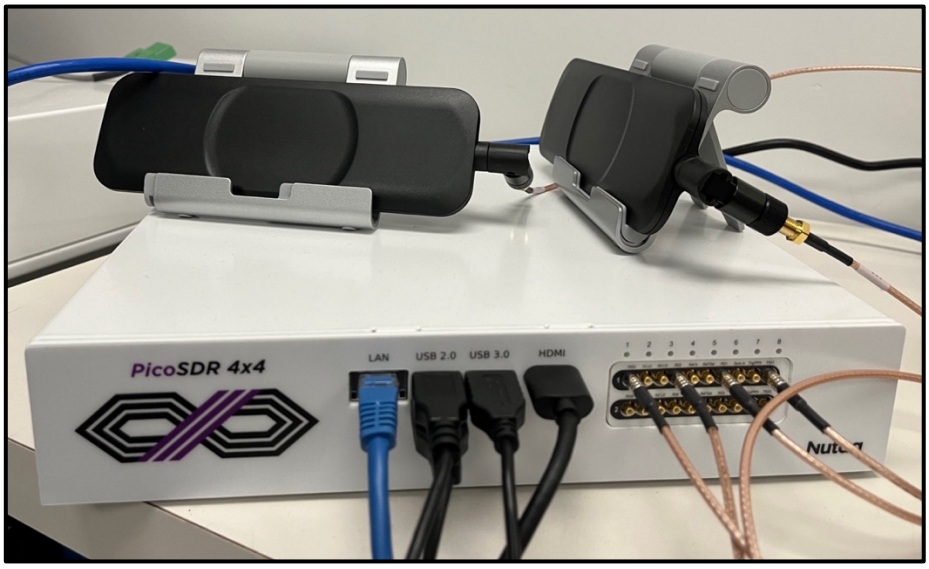


Figure 2.8 Nutaq PicoLTE with 2 Antennas

2.4 Third Layer (middle layer)

The middle layer, often referred to as the third layer of the IoT architecture, serves as a foundation in the IoT ecosystem. Its role is instrumental in shaping the development and deployment of IoT applications. Essentially, it acts as an intermediary interface facilitating communication between the Internet and the plethora of 'things' that comprise IoT.

Key features of this layer include:

- **Security:** Ensuring safe data transmission and storage.
- **Scalability:** Adapting to the growth in connected devices or data.
- **Privacy:** Ensuring user data confidentiality.
- **Transparency:** Offering clear insight into data processing.

- **Integrity:** Ensuring that the data remains unaltered and trustworthy.

APIs designed for interactions with this middle layer are grounded in standard application protocols. Moreover, API endpoints, crucial for data and service access, ought to be discoverable through an open catalog. These endpoints should also be accompanied by linked metadata detailing the resources.

In our literature review, we noted the existence of various platforms operating at the middle layer level. Nevertheless, we chose to develop our own platform. This decision was driven by several advantages:

- **Upgradability**: Swift adaptation to emerging technologies.
- **Maintainability**: Simplified system maintenance.
- **Cost-efficiency:** Reduced expenses.
- **Scalability and Flexibility:** The middle layer can easily scale and integrate with various components. Easier adaptability to changes. Also, has more compatibility with diverse technologies.
- **Cross-platform Compatibility**: device communication regardless of underlying software.
- **Ease of Maintenance**: Efficient monitoring and troubleshooting of IoT device performance.
- **Reusability:** Middle layer applications are crafted for reuse, aiding in system upgrades and cost reductions.
- **Data Security:** Enhanced protection against online threats.

The primary focus of this thesis revolves around this layer, emphasizing data security, user privacy, and data integrity.

The middle layer shoulders the responsibility of data reception from the network layer. It either stores this data, processes it (through filtering, aggregation, inference, etc.), or analyzes it. Additionally, this layer is responsible for event management, context detection, and enforcing

application policies such as security and privacy rules. It also monitors system operations and data-transmission failures while providing necessary access protocols.

Central components of this layer encompass:

- **Communication:** Facilitating real-time data streaming.
- **Security**: Safeguarding data and devices.
- **Data and Device Management**: Handling and organizing data effectively. Concurrently, Overseeing device operations and health.

2.4.1 Third layer's component

The Communication Middle layer manages interactions between IoT devices, providing data exchange protocols and facilitating data translation. The Data Management Middle layer deals with data produced by these devices, supplying tools for its collection, storage, processing, and allowing data integration from different sources. The Device Management is dedicated to handling IoT device functionalities, presenting tools for tasks such as registration, provisioning, and firmware updates, and also supporting remote device oversight. Lastly, the Security ensures the safety of IoT interactions by delivering authentication, authorization, and encryption tools, thus safeguarding communications between devices and applications.

These components can be addressed using three distinct but complementary technologies. Each of these technologies may serve multiple purposes, ensuring versatility and comprehensive coverage of the required functions. The trio of technologies that have been employed includes:

- **Blockchain:** Primarily utilized for ensuring the security and integrity of data, Blockchain acts as a tamper-proof ledger. Beyond its security functions, it also plays a role in data and device management, ensuring that every transaction or change is recorded in a transparent and immutable manner.
- **Real-time Data Streaming:** This technology is pillar for communication, especially when timely data transmission is essential. It enables the immediate and continuous

flow of data, ensuring that information is relayed as it is generated or received, minimizing delays and ensuring prompt responses.

- **Database:** Serving as the backbone for data management, databases store, organize, and retrieve vast amounts of information. They provide structured storage solutions that allow for efficient querying and data retrieval, ensuring that data remains accessible and organized for various applications.

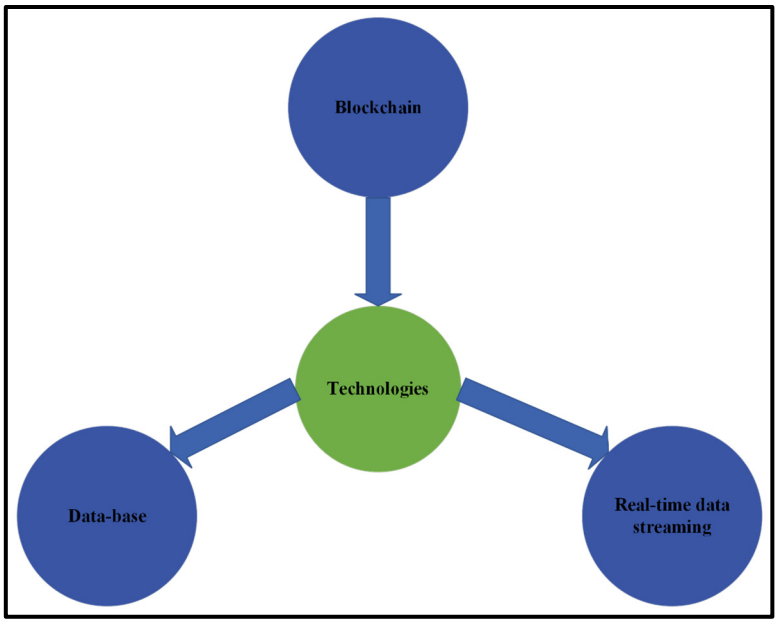


Figure 2.9 Trinity Technologies in this Platform

2.4.1.1 Real-time data streaming component

In this subsection, we delve into the implementation of real-time data streaming, a critical component of our system. This component is tasked with receiving data from the second layer. Based on our literature review, which highlighted the benefits of low latency, high scalability, and fault tolerance, we have chosen to employ Kafka Apache for our real-time data streaming needs. Essentially, Apache Kafka serves as a conduit to relay data from the second layer.

Kafka primarily consists of three core components:

1. Kafka Producer:
    - The Kafka Producer is a client that publishes or sends data/messages to the Kafka cluster. It retains records awaiting transmission and operates a background I/O thread.
    - This producer creates topics with data, where each topic corresponds to a specific device. These topics are then dispatched to the Kafka cluster. Multiple producers can exist simultaneously.
    - Topics in Kafka categorize messages and are segmented into Partitions. Producer applications deposit data into these topics. Each Partition, customizable in its quantity, ensures multiple users can access the data concurrently. Within a Kafka Cluster, individual servers oversee their respective Partitions. Messages are paired with keys that guide them to designated Partitions, ensuring messages with matching keys land in the same Partition for synchronized reading.

2. Kafka Cluster:
    - Kafka employs publish-subscribe methodology, necessitating a broker for efficient operation. The Kafka cluster, an assembly of these brokers, provides the scalability to the platform.
    - Kafka Clusters manage the persistence and replication of message data. If a primary cluster fails, backup Kafka Clusters can take over, ensuring uninterrupted service. Each server within the cluster controls its Partitions. Messages are assigned keys which determine their Partition placement. Messages with identical keys are routed to the same Partition, facilitating simultaneous access.
3. Kafka Consumer:
    - The Kafka Consumer reads or ingests messages from the Kafka cluster. Brokers allocate data/messages to the appropriate consumers based on their needs. Consequently,
    - Kafka Consumer retrieves topics directly from the Kafka cluster.

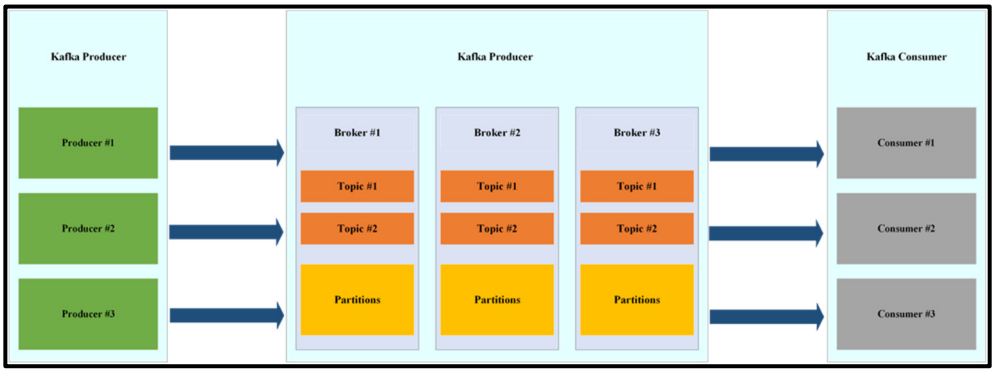


Figure 2.10 Apache Kafka Architecture in this Platform

2.4.1.2 Blockchain

The technology under discussion stands as a cornerstone within this section. It is not just another component; it serves as a foundational element that underpins the entire system. When we delved into the literature review, we noted several distinct advantages of Blockchain technology. Among these are security, privacy, integrity, immutability, and transparency. Each of these attributes is not merely a 'nice-to-have' but essential features that characterize the very essence of blockchain technology. Within this context, emphasizing the security of the middle layer becomes paramount. This layer does not just play a supportive role; it is the beating heart of the platform, ensuring smooth interaction and communication between different parts.

For this critical component of our system, we've chosen to work with Hyperledger Fabric. This is not an arbitrary choice. Hyperledger Fabric stands out as a private, permissioned blockchain, which introduces a layer of control and governance often missing in public Blockchains. This means that every participant in our network is vetted and approved, creating a trusted environment. Unauthorized users are unable to join the network, ensuring that all participants are recognized and authenticated by the system's administrator.

Our innovative approach incorporates both on-chain and off-chain strategies. The rationale behind this is twofold. Firstly, by using a database to store the bulk of the data, we can ensure

speed and efficiency. This database then provides a unique identifier for each data piece, which is subsequently stored on the blockchain. This method ensures that while the data is securely stored off-chain, its integrity and authenticity can be verified on-chain. Secondly, our system leverages the blockchain not just as a static ledger but as a dynamic tool for device management. This approach maximizes the potential of blockchain technology beyond mere data storage.

To further enhance the system's capabilities, we have introduced two specialized smart contracts. The division of responsibilities between these contracts has been meticulously planned. The first contract is focused on data storage, ensuring that each piece of data is securely and accurately logged. In contrast, the second contract takes on a more managerial role, overseeing both our devices and user interactions. This dual-contract system ensures that tasks are specialized and efficient.

Within the confines of Hyperledger Fabric, we've also established multiple organizations. These are not just abstract entities; they represent real-world stakeholders and entities eager to monitor and analyze data from their respective clients. In this architecture, a channel — a dedicated private communication pathway — links two or more organizations within the Hyperledger Fabric network. Providing each organization with the capability to manage its users introduces a level of autonomy and flexibility. As a result, users within these organizations can assign devices to themselves, be it large-scale gateways or individualized sensors. The nature of the device assignment is inherently flexible, adapting to the unique structure and needs of each organization.

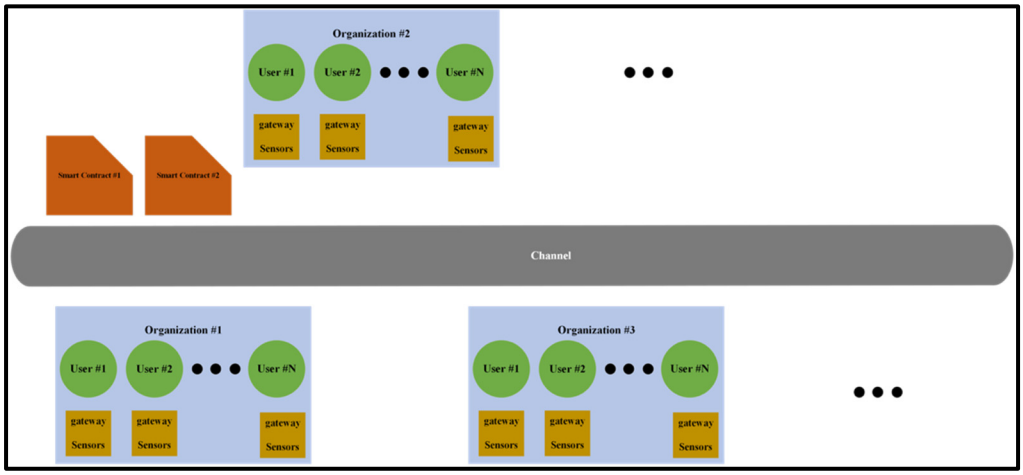


Figure 2.11 Hyperledger Fabric Structure in Platform

2.4.1.3 Data base component

In this section, we discuss the components of the database layer, which has the responsibility of data storage. There are generally two types of databases:
Centralized databases predominantly rely on one main server for controlling services and data, making them susceptible to significant disruptions, as exemplified by the 2017 Amazon AWS incident. On the other hand, decentralized databases operate on a principle where all nodes function in a peer-to-peer capacity, ensuring no single entity dominates, providing higher reliability and resilience against potential system-wide failures (Fong, Selvarajah, and Nabi 2022).

For our purposes, we have chosen a decentralized approach, utilizing the InterPlanetary File System (IPFS) as our database. The Interplanetary File System (IPFS) is a ground-breaking protocol that aims to revolutionize the web by facilitating peer-to-peer file storage and sharing, conflicting the need for centralized servers. When combined with blockchain technology, IPFS promises enhanced data security, rapid file retrieval, and consistent data availability across its network. This combination offers a robust solution that ensures data remains secure, accessible, and readily available, even when certain nodes become compromised (Anthal, Choudhary, and Shettiyar 2023).

In IPFS, data is initially divided into smaller blocks, with each block being assigned a unique Content Identifier (CID) based on its content. When the user requests a piece of data, IPFS employs mechanisms like the Kademlia Distributed Hash Table (DHT) and Bitswap to identify which peers in the network possess the desired CID. Following the location determination, the data transfer is facilitated primarily by Bitswap. There are also alternative transfer avenues available, such as HTTP Gateways for applications not native to IPFS and Sneakernet for offline scenarios. Once the data is received, IPFS ensures its authenticity and integrity by re-computing and verifying the CID. Furthermore, the stored data can be accessed either directly by IPFS nodes or indirectly via bridges like IPFS-to-HTTP gateways, ensuring versatility in data retrieval (IPFS Docs n.d.).

In our architectural design, we begin by generating a file derived from the data we have collected. This data is then segmented into discrete units, often referred to as "chunks" or "packets." Each of these packets is bundled into a comprehensive block. This block is more than just a collection of data packets; it is a detailed record that contains timestamps indicating when the block was created. Additionally, it incorporates the preceding Content Identifier (CID) from IPFS, establishing a clear lineage of data blocks. Once these blocks are fully assembled with all the necessary components, they undergo a processing phase to transform them into structured files. These files are then primed for submission to IPFS. it is worth noting that for the very first block in this sequence, the CID is initialized to a value of 0, serving as a starting reference point for subsequent blocks.

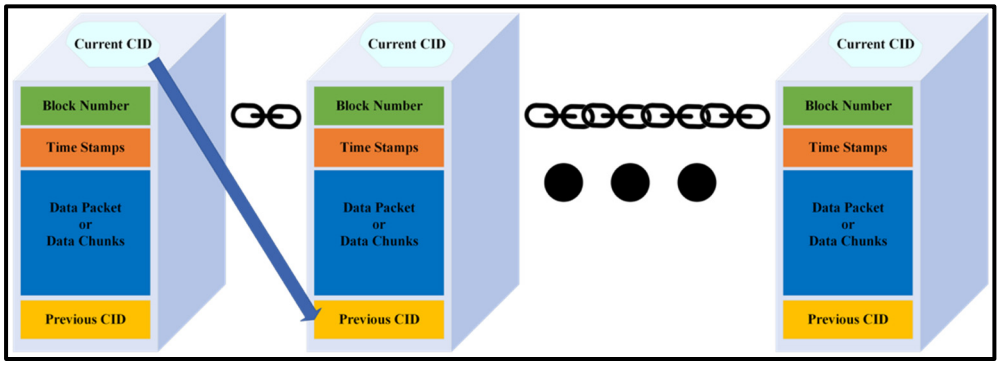


Figure 2.12 Data Structure in this Platform a with Chain Formatting

2.5 Forth layer (Application layer)

In the forthcoming section, we will explore an examination of the fourth and final layer of our architectural framework. This layer is responsible for interacting with the third layer. Comprised of applications and services constructed atop the middle layer, this layer is the most prominent to the end-users. It encompasses smart home applications, connected vehicular systems, digital healthcare solutions, and intelligent urban infrastructure. This layer affords users access to data aggregated and processed in the preceding layers. Moreover, it facilitates the visualization, analysis, and actionable insights derived from the network data. The responsibility of application layer is to ensure secure and intuitive user interactions, prerequisites for the efficacious deployment of an IoT architecture.

This layer is divided into two distinct sub-layers:

1. **Authentication layer:** This sub-layer is tasked with verifying the credentials of users, ensuring that only authorized individuals gain access.
2. **User Web Interface layer**: This serves as the primary interface through which users can interact seamlessly with the platform.

Within this layer, users or administrators possess the capability to authenticate themselves, effectuate modifications to their devices, or monitor the operational status of said devices.

The design and functionalities inherent to this layer are intrinsically tailored to the specific requirements of the client or user. In our endeavor to enhance user experience, we have pioneered a web-based user interface allowing users to securely store their cryptographic private and public keys. To facilitate connectivity with the blockchain, this layer employs a RESTful API. HTTP/S stands as a quintessential example of an application layer protocol, witnessing widespread adoption across the digital domain.

2.5.1 Authentication

In this section, our primary focus is on the process of user authentication, which stands as an integral and major aspect of the Application layer. This aspect is instrumental in safeguarding the digital identities and access rights of both users and administrative personnel. User authentication not only acts as the first line of defense against unauthorized access but also maintains the integrity and confidentiality of sensitive information. By precisely review and validating user credentials and access requests, it becomes possible to aid the overarching security framework, ensuring that only legitimate users can interact with the system. In our pursuit of offering robust and secure authentication mechanisms, we have incorporated two distinct methodologies tailored to meet user-specific requirements and to address potential security challenges.

1. **Token-based Authentication:** Upon administrative approval, a pair of public and private keys is generated for the user, accompanied by a unique sequence of 13 to 17 random words. Utilizing these words, we create a JSON Web Token (JWT). The primary interaction of this JWT is with the Middleware layer, especially the blockchain.
    a. JWTs are encrypted tokens facilitating secure data transfer between clients and servers, composed of three elements: the Header (detailing encryption), the Payload (holding user data and metadata), and the Signature (ensuring authenticity). The website JWT.IO provides a tool for decoding and analyzing

the structure of JWTs, excluding their private signature (Akanksha and Chaturvedi 2022).

Figure 2.13 JWT Example

2. **Lightweight Directory Access Protocol (LDAP**) Authentication: We have adopted LDAP for authentication, and beyond its primary function, it serves to store user profiles, which will be leveraged in future implementations. Our user base is divided into administrators and general users. Administrators possess the privilege to enroll new users, while general users primarily engage with the platform in a monitoring capacity.
    a. LDAP is a protocol used for querying directory services for years, serving as a lightweight version of the X.500 protocol's Directory Assistance Service. Many applications, including Microsoft's Active Directory Server, leverage LDAP for managing directory services due to its lightweight nature. These services store attribute-value pairs for users, applications, and devices. Enterprise applications utilize LDAP for authentication across various platforms, including email clients, SSH, servers, and workstations (Srinivasa, Pedersen, and Vasilomanolakis 2022).

2.5.2 User interface

In this part, we explore further into the Application layer's second sub-layer, specifically the User Interface (UI) element, a critical component that supports user interaction with the

platform. This interface links users and the system's core functions, delivering a smooth and natural experience.

Our approach to designing this sub-layer resulted in creating unique web interfaces. We used ReactJS, a sophisticated JavaScript package, for the front end. Because of its speed, versatility, and modular architecture, ReactJS stands out in front-end development, making it a popular choice for creating dynamic user interfaces. Its position as one of the premier front-end application libraries attests to its powerful capabilities. In the digital age, raw data typically has little value without meaningful interpretation. Recognizing this, we have combined many libraries to create a data visualization module. This module converts complicated datasets into understandable visual representations, allowing users to gain insights and make educated decisions based on the visualized data.

2.6 Chapter summary

Chapter 2 explains the architecture and design of the platform, segregating it into four distinct layers to facilitate comprehension. Initially, the chapter introduces the overall architecture, setting the stage for a deeper dive into each layer. The First Layer, termed the physical layer, is explored with various case studies. It looks into the pairing of Pinnacle 100 and sensors, the integration between Raspberry Pi and sensors, and presents the Automatic Dependent Surveillance-Broadcast (ADS-B) as a unique device. Following the physical layer, the chapter transitions into the Second Layer, which is the network layer. A case study focusing on the use of PicoLTE as a network layer is presented, providing insights into its functionalities and relevance.

As the narrative progresses, the Third Layer, commonly referred to as the middle layer, is elaborated upon. This section dives deep into the essential components of the third layer, discussing the role and significance of real-time data streaming through Apache Kafka, the incorporation of Blockchain technology, and the structure and utilization of the database. The chapter culminates with the Fourth Layer, which is the application layer. This layer is integral

for end-user interaction, and the chapter details its authentication mechanisms and the design of the user interface. By the end of Chapter 2, readers gain a holistic understanding of the platform's layered architecture, its components, and their interrelationships.

CHAPTER 3

Implementation of the platform

In this chapter, we explore the details of our implementation. The primary emphasis of this thesis lies on the middle layer and application layers. We will explore our implementation strategies for Apache Kafka, Blockchain, and our database, explaining how they were integrated. A thorough examination of our authentication procedures will be provided, including the chosen schema for authentication.

Furthermore, we will elucidate our approach to achieving real-time data streaming and provide insights into the configuration specifics of the blockchain and authentication processes. Our exploration will encompass the Blockchain's Software Development Kit (SDK) and detail the coding methodologies employed for the server-side backend. In addition, we will highlight the APIs leveraged during the development phase.

A comprehensive list of libraries used in our project will be presented, followed by an explanation of our blockchain monitoring strategy. The platforms chosen for this monitoring task will also be introduced.

3.1 The Implementation Process

In the course of our platform's implementation, we employed Docker images for containerization. Docker, established in 2013, has emerged as a pivotal platform in the domain of DevOps, securing the foremost position in the 2022 StackOverflow survey. The adoption of containerization is projected to witness a significant upsurge, with forecasts suggesting that 85% of organizations will integrate it by 2025, a substantial increase from the 30% recorded in 2020. Docker facilitates the deployment of software applications within lightweight virtual environments termed as containers. This deployment is orchestrated via a Dockerfile, which delineates the execution environment requisite for an application. DockerHub, an accessible

repository, provides a platform for developers to disseminate and build upon existing Docker images (Rosa, Scalabrino, and Oliveto 2022).

Upon utilizing Docker images, it becomes imperative to designate a specific port, either on a local machine or a server, to facilitate access and interaction with the said container. We have architected a Node.js server tasked with interlinking these containers to ensure cohesive operation. The implementation process was initiated by integrating our real-time data streaming mechanism, Apache Kafka. This was subsequently tethered to our primary database, IPFS. The data processed via IPFS yields a CID, which is subsequently archived in the Blockchain. This intricate implementation is segmented into three distinct stages, each of which will be expounded upon, inclusive of specific configurations, in the ensuing sections.

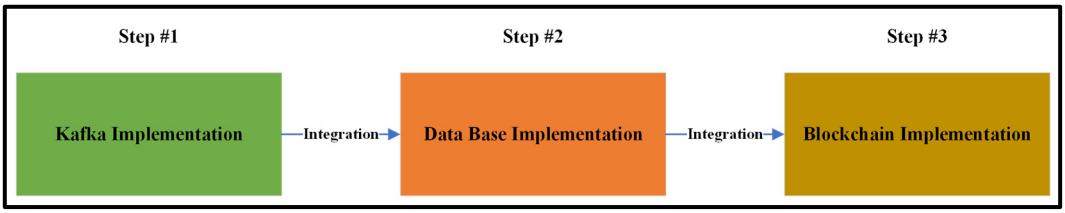


Figure 3.1 Steps of Integration in This Platform

3.1.1 Real-time data streaming (KAFKA Apache) implementation

This section focuses on Apache Kafka's role in real-time data streaming, emphasizing its containerized implementation for optimal execution across diverse platforms.
Utilizing a docker-compose file -- a tool that facilitates the orchestration of multi-container Docker applications-- ensures systematic orchestration of Docker applications. Such approach ensures uniformity in Kafka's execution across heterogeneous platforms. This file specifies the implementation of three containers integral to the Kafka ecosystem:

1. **ZooKeeper:** Serving as a coordination interface, ZooKeeper is essential for ensuring systematic communication between Kafka brokers and consumers. It plays a crucial role in managing cluster metadata and maintaining broker leader statuses.

2. **Broker:** Central to Kafka's publish/subscribe (pub/sub) paradigm, the broker is responsible for data storage and client request servicing. It is essentially linked with ZooKeeper for coordination activities. Proper broker configuration is imperative, with the Kafka Apache documentation providing authoritative guidance.
3. **Control Center:** This web interface offers an analytical overview of Kafka operations. In this study, it is accessible via localhost:9021.

Figure 3.2 Apache Kafka Containers

To consider the scalability while the current configuration comprises a single broker within one cluster, Kafka's design innately supports scalability. Its architecture allows for the addition of brokers and clusters in response to increasing data demands.

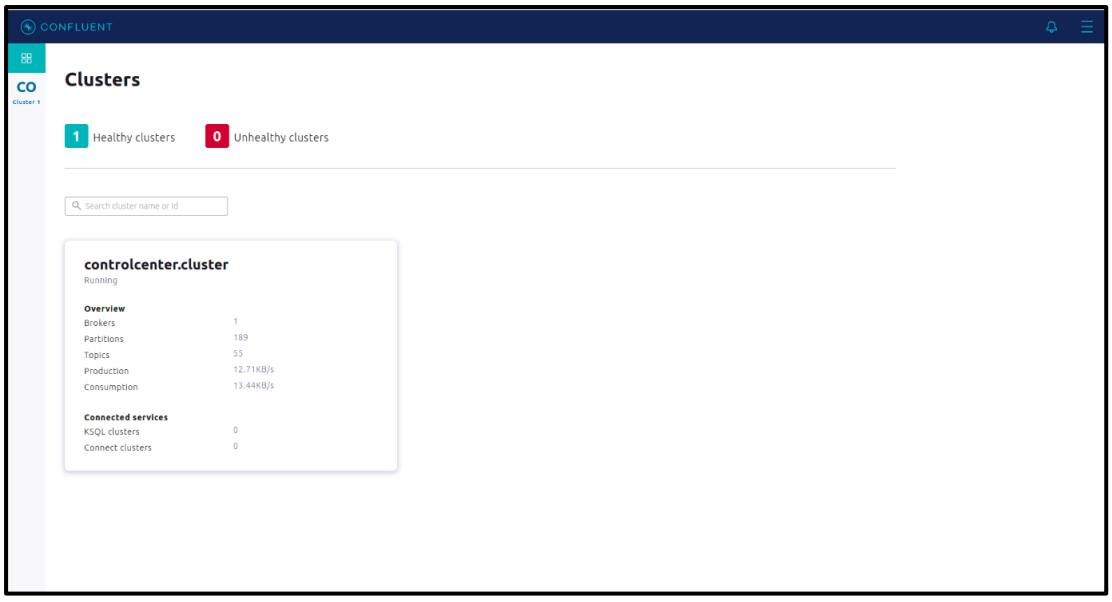


Figure 3.3 Apache Kafka User Interface

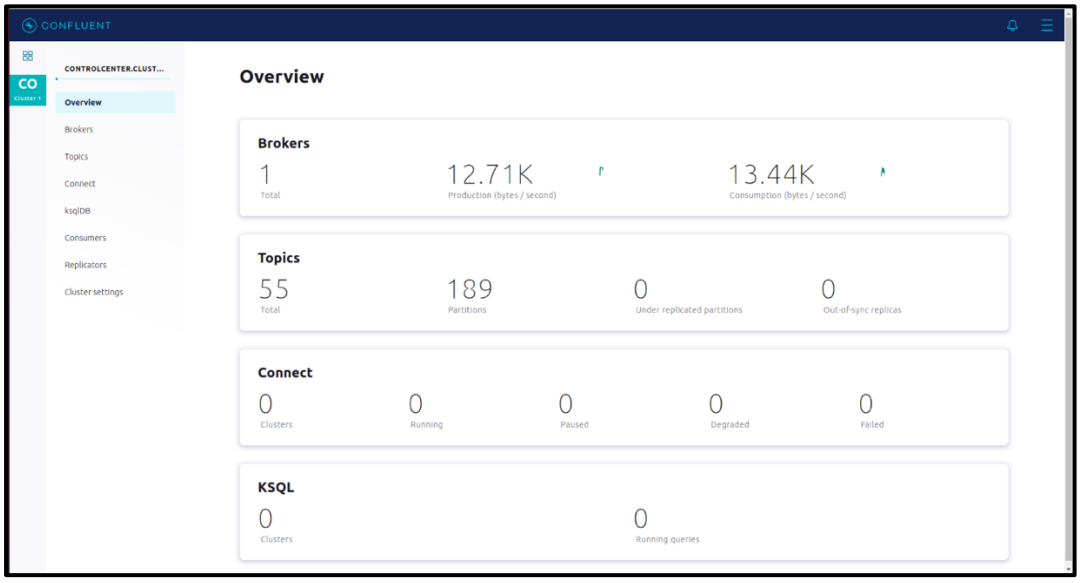


Figure 3.4 Apache Kafka Cluster Settings

3.1.1.1 Kafka integrated with Network (Producer)

This section provides a scholarly examination of Apache Kafka's "producer" component. The Kafka producer interfaces with the MQTT protocol to receive data from a designated second layer. In instances where data is transmitted using other protocols, such as LwM2M with PicoLTE over 5G or 4G, an intermediary agent is employed. This agent's function is to transcode the incoming data protocol to MQTT, ensuring compatibility with the Kafka producer. Upon initialization, the producer subscribes to an MQTT topic, priming itself for data reception. The ingested data undergoes serialization, being encoded to the "utf-8" format. Subsequently, the KafkaProducer command dispatches the serialized data to the Kafka cluster, associating it with a specified topic.

In Figure 3.5, three sensors are depicted as connected to a pinnacle. Data from these sensors is transferred via the MQTT protocol through an intermediary agent. This data is then received by the Kafka producer and subsequently relayed to the Kafka clusters.

```
Sending the data related to the sensor1 is:  {'humidity': 14, 'temperature': 6, 'pressure': 941, 'time': 'Fri Aug 25 16:35:47 2023'}
Sending the data related to the sensor2 is:  {'humidity': 91, 'temperature': 40, 'pressure': 812, 'time': 'Fri Aug 25 16:35:47 2023'}
Sending the data related to the sensor3 is:  {'humidity': 92, 'temperature': 6, 'pressure': 839, 'time': 'Fri Aug 25 16:35:47 2023'}
Sending the data related to the sensor1 is:  {'humidity': 76, 'temperature': 27, 'pressure': 1143, 'time': 'Fri Aug 25 16:35:50 2023'}
Sending the data related to the sensor2 is:  {'humidity': 62, 'temperature': 7, 'pressure': 1198, 'time': 'Fri Aug 25 16:35:50 2023'}
Sending the data related to the sensor3 is:  {'humidity': 93, 'temperature': 28, 'pressure': 1058, 'time': 'Fri Aug 25 16:35:50 2023'}
```

Figure 3.5 Three Connected Sensors Data Format

3.1.1.2 Apache Kafka Integrated with Data base (Consumer)

In this section, we delve into the implementation of the consumer component within the Apache Kafka framework. Once data or information is dispatched to the Kafka cluster by the Kafka producer under a specific topic, it becomes the obligation of the consumer to retrieve this data. The primary role of the consumer is to receive this data and subsequently forward it to the requisite storage system. For the purposes of this study, we have selected the InterPlanetary File System (IPFS) as our decentralized database.

To facilitate this process, we employed the `KafkaClient` function to subscribe to the Kafka cluster, thereby enabling the retrieval of the pertinent data. Furthermore, we have instantiated the 'Consumer' from the Kafka library, ensuring that the appropriate topics are selected for data integration.

It is imperative to note that upon retrieval of the raw data from Kafka, there exists a necessity to transform this data from its initial buffer format into a more accessible string format. This step is crucial to ensure compatibility and integration with subsequent processing stages.

```
{
  value: <Buffer 7b 22 68 75 6d 69 64 69 74 79 22 3a 20 31 32 2c 20 22 74 65 6d 70 65 72 61 74 75 72 6
5 22 3a 20 32 39 2c 20 22 70 72 65 73 73 75 72 65 22 3a 20 38 34 ... 38 more bytes>,
  size: 88,
  key: null,
  topic: 'sensor1',
  offset: 583,
  partition: 0,
  timestamp: 1692996122481
}
```

Figure 3.6 Receiving Raw Data from Apache Kafka Consumer

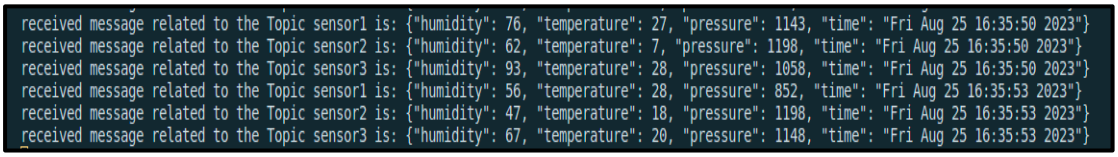


Figure 3.7 Data after being Processed to String

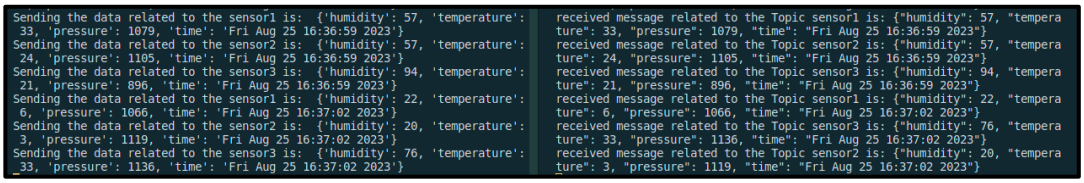


Figure 3.8 Producer (left side) and Consumer (right side) Together

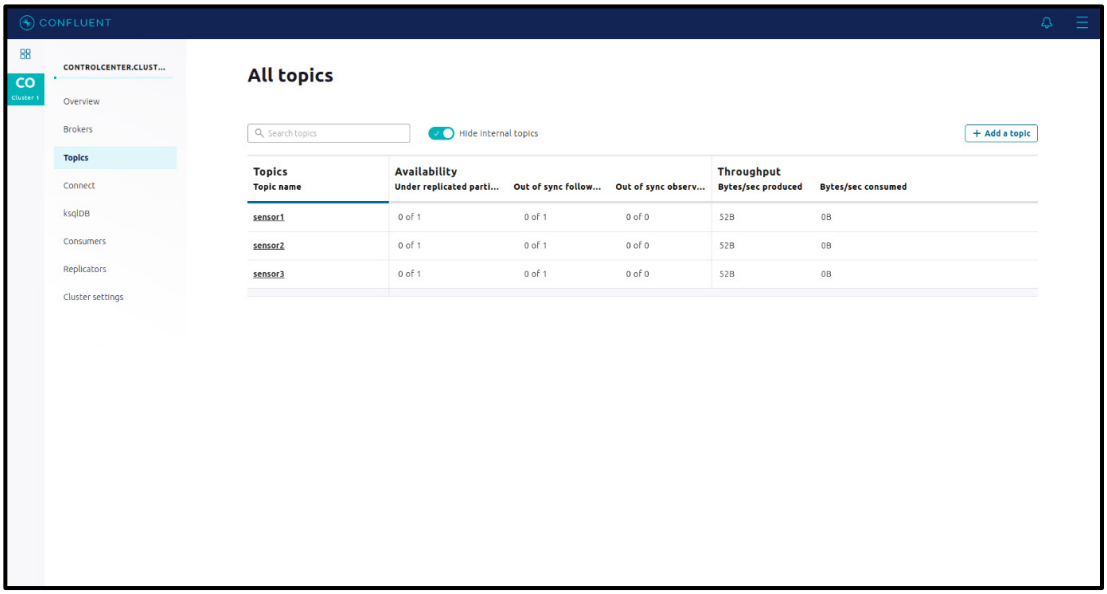


Figure 3.9 Three Topics Transferring in Apache Kafka Center

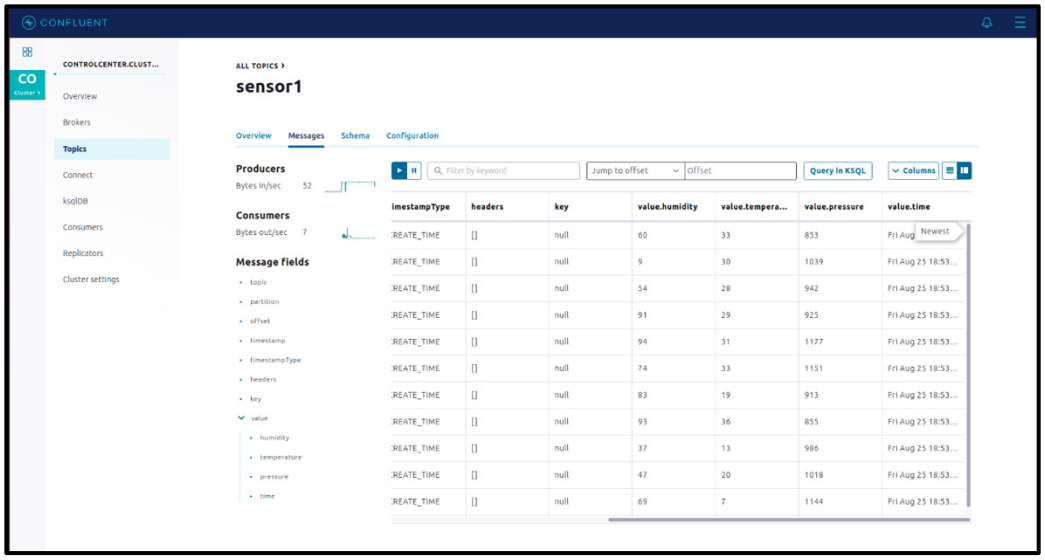


Figure 3.10 Detailed Topic with Parameters

3.1.2 Data Base Implementation

In the forthcoming section, we delineate the complication of our database implementation, anchoring our discussion in the choices we've made regarding data storage strategies.

We have strategically opted for a combined off-chain and on-chain approach for data storage. This necessitates the implementation of a database. In our pursuit of preserving the decentralization of the platform, it is imperative to employ a decentralized database.

Upon rigorous evaluation of available options, we have zeroed in on the InterPlanetary File System (IPFS) as our decentralized database of choice. IPFS is renowned for its decentralized nature, ensuring the autonomy and resilience of our data storage mechanism.

To effectively integrate with the IPFS ecosystem, it is essential to establish a node. This allows our platform to interact with the broader IPFS community. The establishment of this node entails the installation of the IPFS node software. In our endeavor to connect this node with our application, we diligently perused the official IPFS documentation, ensuring best practices are adhered to. Initiating the node is achieved by executing the command “daemon ipfs”. Upon successful initialization of IPFS, one can verify its operational status. For more intuitive

interaction and real-time monitoring of IPFS, there's a dedicated web interface. This interface can be accessed at http://127.0.0.1:5001/webui, providing users with a comprehensive overview of the IPFS node's activities and status.

Data ingested from the Kafka consumer undergoes a transformation process. Initially, this data is segmented into discrete chunks. Leveraging these chunks, we fabricate specific block types, as elucidated in prior sections. Post block creation, this data is relayed to IPFS. A consequential outcome of this process is the generation of a Content Identifier (CID). This CID plays an important role in subsequent platform operations, particularly within our blockchain implementation. To provide further clarity, raw data is transmuted into data chunks, as expounded in our methodology section. This chunked data is subsequently relayed to IPFS for persistent storage. Additionally, metadata, such as the timestamp marking the block's creation, is appended to each data block.

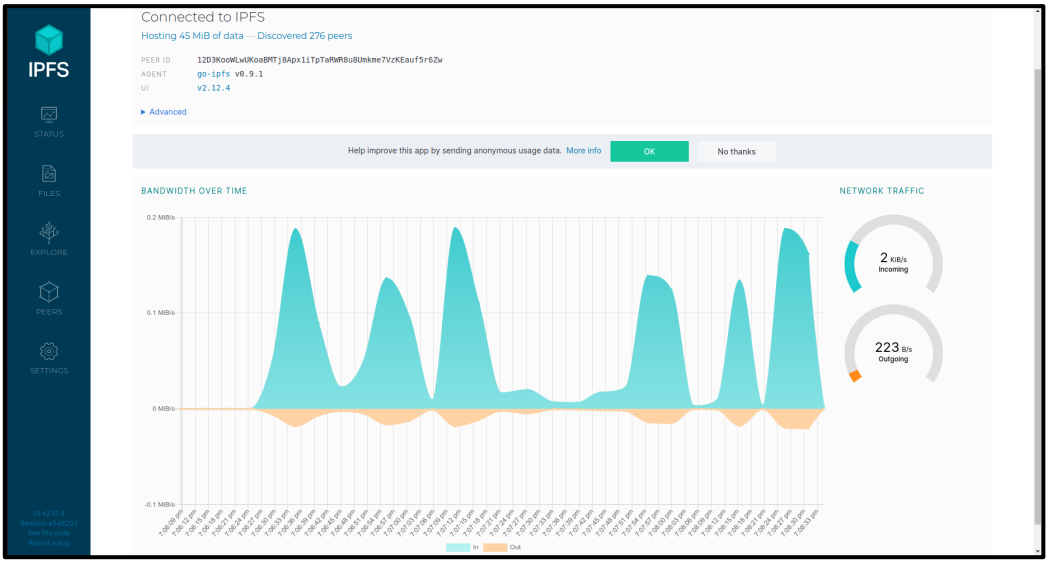


Figure 3.11 IPFS Web User Interface

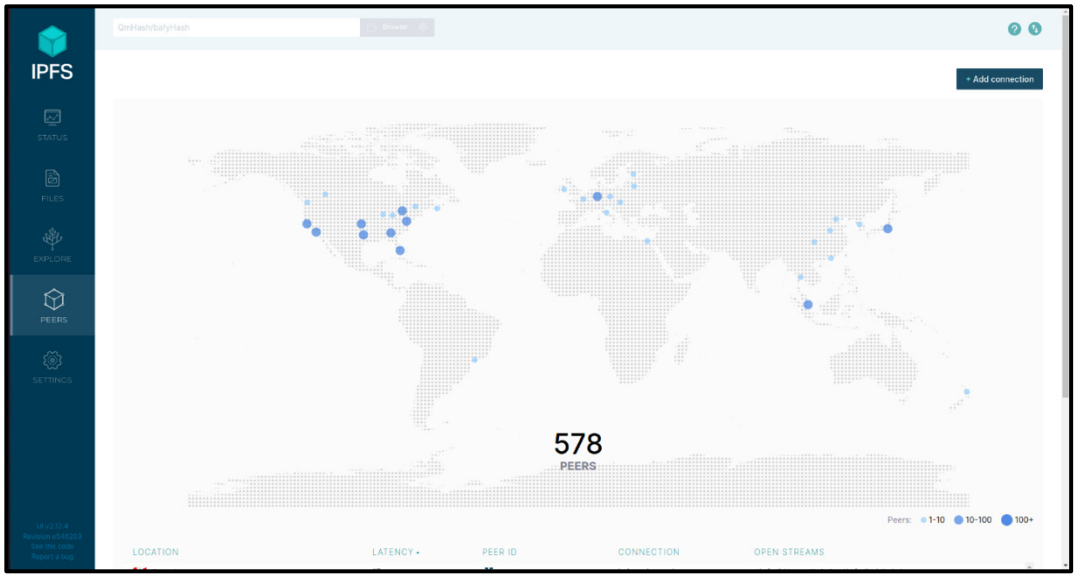
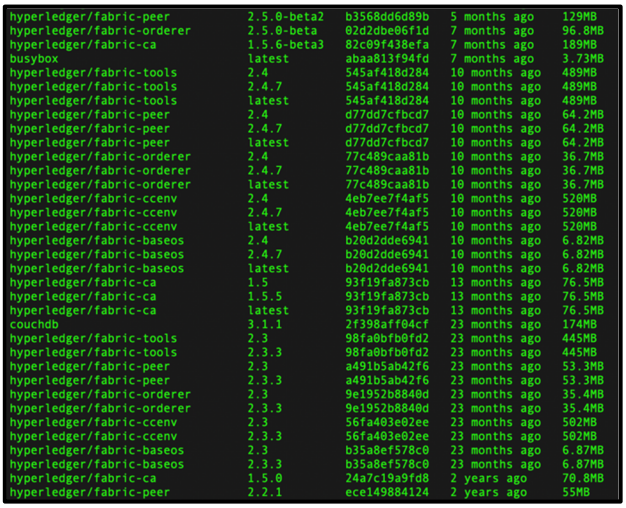


Figure 3.12 IPFS Status Web Page of Availability

3.1.3 Blockchain Implementations

In this section, we represent the procedure for deploying our Blockchain. As previously discussed in the methodologies section, our choice of blockchain technology is Hyperledger Fabric (HLF). This decision was informed by several considerations: foremost, the promising performance metrics of HLF; its extensive support within the developer community; and its inherent flexibility which enables us to accommodate a broad spectrum of operations.

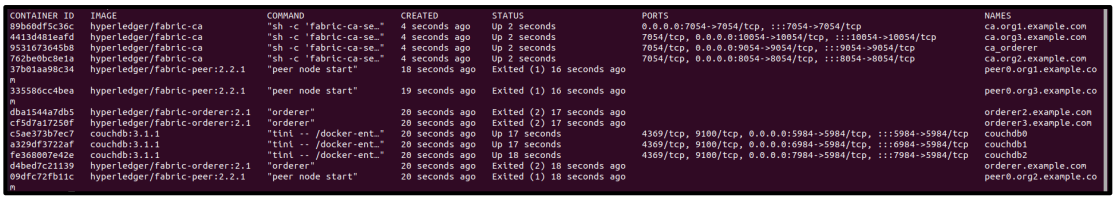
To operationalize HLF, there are specific prerequisites that need to be addressed during the implementation phase. Firstly, certain software components, including Golang, Docker, Docker-compose engine, and Node.js, must be installed. Subsequent to these installations, the next course of action is to download the binary file for HLF, specifically version 2.x. Although version 2.x is recommended, it would be prudent to acquire the latest version to ensure optimal functionality. Alongside the binary file, it is essential to download all pertinent Docker images associated with HLF. To verify the successful download of these images, one can execute the command “docker images”; this provides a comprehensive list of all the Docker images that have been secured.



hyperledger/fabric-peer	2.5.0-beta2	b3568dd6d89b	5 months ago	129MB
hyperledger/fabric-orderer	2.5.0-beta	02d2dbe06f1d	7 months ago	96.8MB
hyperledger/fabric-ca	1.5.6-beta3	82c09f438efa	7 months ago	189MB
busybox	latest	abaa813f94fd	7 months ago	3.73MB
hyperledger/fabric-tools	2.4	545af418d284	10 months ago	489MB
hyperledger/fabric-tools	2.4.7	545af418d284	10 months ago	489MB
hyperledger/fabric-tools	latest	545af418d284	10 months ago	489MB
hyperledger/fabric-peer	2.4	d77dd7cfbcd7	10 months ago	64.2MB
hyperledger/fabric-peer	2.4.7	d77dd7cfbcd7	10 months ago	64.2MB
hyperledger/fabric-peer	latest	d77dd7cfbcd7	10 months ago	64.2MB
hyperledger/fabric-orderer	2.4	77c489caa81b	10 months ago	36.7MB
hyperledger/fabric-orderer	2.4.7	77c489caa81b	10 months ago	36.7MB
hyperledger/fabric-orderer	latest	77c489caa81b	10 months ago	36.7MB
hyperledger/fabric-ccenv	2.4	4eb7ee7f4af5	10 months ago	520MB
hyperledger/fabric-ccenv	2.4.7	4eb7ee7f4af5	10 months ago	520MB
hyperledger/fabric-ccenv	latest	4eb7ee7f4af5	10 months ago	520MB
hyperledger/fabric-baseos	2.4	b20d2dde6941	10 months ago	6.82MB
hyperledger/fabric-baseos	2.4.7	b20d2dde6941	10 months ago	6.82MB
hyperledger/fabric-baseos	latest	b20d2dde6941	10 months ago	6.82MB
hyperledger/fabric-ca	1.5	93f19fa873cb	13 months ago	76.5MB
hyperledger/fabric-ca	1.5.5	93f19fa873cb	13 months ago	76.5MB
hyperledger/fabric-ca	latest	93f19fa873cb	13 months ago	76.5MB
couchdb	3.1.1	2f398aff04cf	23 months ago	174MB
hyperledger/fabric-tools	2.3	98fa0bfb0fd2	23 months ago	445MB
hyperledger/fabric-tools	2.3.3	98fa0bfb0fd2	23 months ago	445MB
hyperledger/fabric-peer	2.3	a491b5ab42f6	23 months ago	53.3MB
hyperledger/fabric-peer	2.3.3	a491b5ab42f6	23 months ago	53.3MB
hyperledger/fabric-orderer	2.3	9e1952b8840d	23 months ago	35.4MB
hyperledger/fabric-orderer	2.3.3	9e1952b8840d	23 months ago	35.4MB
hyperledger/fabric-ccenv	2.3	56fa403e02ee	23 months ago	502MB
hyperledger/fabric-ccenv	2.3.3	56fa403e02ee	23 months ago	502MB
hyperledger/fabric-baseos	2.3	b35a8ef578c0	23 months ago	6.87MB
hyperledger/fabric-baseos	2.3.3	b35a8ef578c0	23 months ago	6.87MB
hyperledger/fabric-ca	1.5.0	24a7c19a9fd8	2 years ago	70.8MB
hyperledger/fabric-peer	2.2.1	ece149884124	2 years ago	55MB

Figure 3.13 Hyperledger Fabric Images

Once these preliminary steps are accomplished, the subsequent task is the generation of certificate authorities for all the necessary organizations and orderers. This phase culminates in the creation of a 'crypto-config' file, which consolidates all the indispensable keys for the holistic functioning of HLF. With this file in place, our next responsibility is the configuration of our artifacts. This entails defining the plethora of parameters associated with the Blockchain, such as the specific consensus algorithm employed and the criteria for the formation of a new block. It is worth noting that a genesis block—often referred to as the inaugural block or block number 0—will be generated during this phase.



CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
89b60df5c36c	hyperledger/fabric-ca	"sh -c 'fabric-ca-se…"	4 seconds ago	Up 2 seconds	0.0.0.0:7054->7054/tcp, :::7054->7054/tcp	ca.org1.example.com
4413d481eafd	hyperledger/fabric-ca	"sh -c 'fabric-ca-se…"	4 seconds ago	Up 2 seconds	7054/tcp, 0.0.0.0:10054->10054/tcp, :::10054->10054/tcp	ca.org3.example.com
953167364 5b8	hyperledger/fabric-ca	"sh -c 'fabric-ca-se…"	4 seconds ago	Up 2 seconds	7054/tcp, 0.0.0.0:9054->9054/tcp, :::9054->9054/tcp	ca_orderer
762be0bc8e1a	hyperledger/fabric-ca	"sh -c 'fabric-ca-se…"	4 seconds ago	Up 2 seconds	7054/tcp, 0.0.0.0:8054->8054/tcp, :::8054->8054/tcp	ca.org2.example.com
37b01aa98c34	hyperledger/fabric-peer:2.2.1	"peer node start"	18 seconds ago	Exited (1) 16 seconds ago		peer0.org1.example.com
335586cc4bea	hyperledger/fabric-peer:2.2.1	"peer node start"	19 seconds ago	Exited (1) 16 seconds ago		peer0.org3.example.com
dba1544a7db5	hyperledger/fabric-orderer:2.1	"orderer"	20 seconds ago	Exited (2) 17 seconds ago		orderer2.example.com
cf5d7a17250f	hyperledger/fabric-orderer:2.1	"orderer"	20 seconds ago	Exited (2) 17 seconds ago		orderer3.example.com
c5ae373b7ec7	couchdb:3.1.1	"tini -- /docker-ent…"	20 seconds ago	Up 17 seconds	4369/tcp, 9100/tcp, 0.0.0.0:5984->5984/tcp, :::5984->5984/tcp	couchdb0
a329df3722af	couchdb:3.1.1	"tini -- /docker-ent…"	20 seconds ago	Up 17 seconds	4369/tcp, 9100/tcp, 0.0.0.0:6984->5984/tcp, :::6984->5984/tcp	couchdb1
fe360807e42e	couchdb:3.1.1	"tini -- /docker-ent…"	20 seconds ago	Up 18 seconds	4369/tcp, 9100/tcp, 0.0.0.0:7984->5984/tcp, :::7984->5984/tcp	couchdb2
d4bed7c21139	hyperledger/fabric-orderer:2.1	"orderer"	20 seconds ago	Exited (2) 18 seconds ago		orderer.example.com
09dfc72fb11c	hyperledger/fabric-peer:2.2.1	"peer node start"	20 seconds ago	Exited (1) 18 seconds ago		peer0.org2.example.com

Figure 3.14 Containers Created in Hyperledger Fabric

Following the successful conclusion of these preparations, our attention then pivots to the configuration of our docker-compose file. This is instrumental in initiating all containers and facilitating their seamless interconnection. For the current deployment, we utilize CouchDB as

our blockchain database. As a case in point, this deployment encompasses three organizations, each equipped with a single peer, and an aggregate of three Orderers. The ensuing phase is focused on channel configuration. For the purpose of this demonstration, all organizations have been integrated into the channel, and its deployment is now in progress.

Figure 3.15 Chaincode Deployment

Concluding this process, the deployment of our smart contracts takes precedence. As an illustrative example, two smart contracts have been conceptualized and will be sequentially deployed.

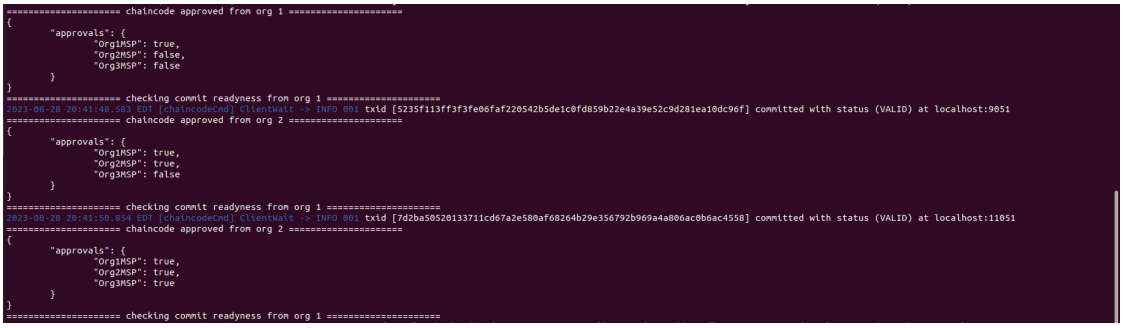


Figure 3.16 Smart Contract Deployment on Each Peers

3.1.3.1 Smart contracts

In the subsequent discourse, an examination of the implemented smart contract structures will be presented, elucidating their operational mechanisms and the integral roles they occupy within the proposed system. Fundamentally, a smart contract is a digitalized agreement that parallels conventional contracts. However, it possesses distinct attributes: it is autonomously executable and verifiable. Such contracts facilitate the automatic enforcement of obligations and associated actions upon the fulfillment of predefined criteria. This eliminate the requirement for intermediate agents, thus cementing trustworthiness amongst engaged entities.

Within the framework of Hyperledger Fabric, a notable Blockchain infrastructure, the decision was made to architect our smart contracts utilizing GoLang. This choice is predicated upon GoLang's performance metrics and the substantial backing from the Hyperledger community. Each contract encompasses various functions, each precisely tailored for an explicit task, thus ensuring the contract's efficient performance.

To explain the architectural framework:

1. **User Management Smart Contract:** This represents the foundational for orchestrating user engagements within the Hyperledger Fabric infrastructure. It not only facilitates administrators in the registration of users with efficacy but also grants upon them the capacity to designate distinct roles. Such design ensures that users can exclusively interface with data and functionalities compatible to their designated role. In addition, this contract functions as a repository, preserving a thorough record of users, their corresponding entities, and attendant devices. It essentially functions as a core for users operations.
2. **Data Management and Verification Smart Contract**: In the contemporary digital era, data stands as an invaluable asset. Acknowledging its paramount importance, a distinct contract has been devised to supervise, authenticate, and determine the consistency of the data incorporated into our platform. This contract employs a stringent methodology for data validation. Each data fragment is tagged with a CID

(Content Identifier), which undergoes a cross-referencing process to validate its authenticity and structural integrity. Notably, the CID is architecturally configured to establish interconnectedness, culminating in a chain configuration. This interconnected structure offers a lucid and graphical delineation of data origin, ensuring transparency in tracking the data's provenance and subsequent alterations.

In conclusion, the combinations of these two smart contracts establishes a fortified groundwork for the proposed platform, ensuring the streamlined and inviolable administration of users and data.

3.1.3.2 Blockchain Software Development Kit

In the subsequent section, we shall delve into a detailed analysis of the Hyperledger Fabric Software Development Kit (SDK).

In the context of Hyperledger Fabric, an SDK, or Software Development Kit, is a set of tools that allows users to interact with the blockchain system. Users utilize the SDK in pair with the Membership Service Provider to operate within the Hyperledger Fabric system. The endorsing peer, after simulating a chaincode for a requested user on a specific set of peers, sends back permission to an application SDK. The SDK plays a significant role in the process, as the ordering service gathers endorsed transactions from it. This ordering service then forms blocks based on the sequence of transactions and subsequently distributes them to every peer node in the channel (Su Wai, Htoon, and Myint Thein 2020).

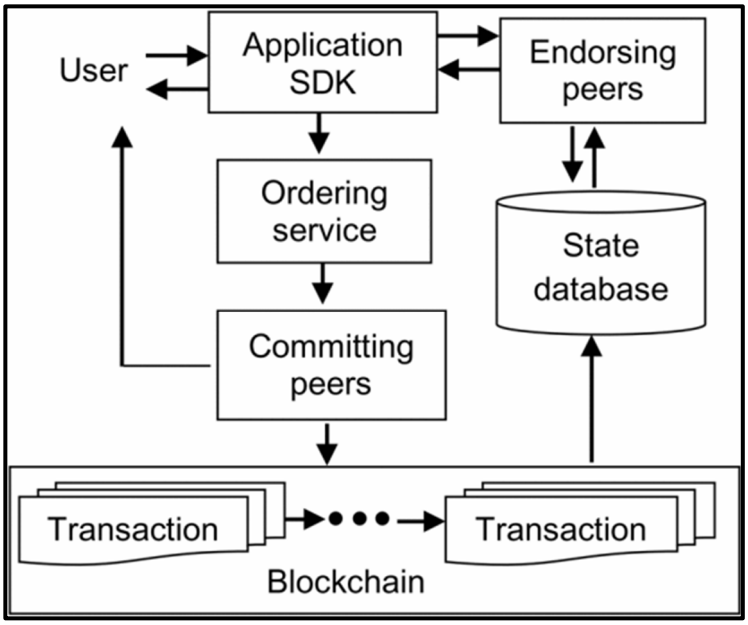


Figure 3.17 SDK in Hyperledger Fabric
Taken from Su Wai et al (2020, p. 2)

Utilizing this SDK facilitates interaction with the Hyperledger Fabric (HLF). It provides a sophisticated, high-level Application Programming Interface (API) that enables transactions, inquiries, and other related operations. For the purposes of user registration, transaction invocation, and historical queries, we have employed a JavaScript file.

3.1.3.3 Blockchain Authentication

In this section, we delve into the mechanisms by which the blockchain facilitates connectivity of users and devices to the platform. Within this architectural layer, we have incorporated the use of the JSON Web Token (JWT) to authenticate and verify every user and device. This token is uniquely designated for each user as well as the devices associated with them.

To ensure secure access, users are required to generate both a public and a private key, based on the RSA 1024-bit encryption standard, through a passphrase ranging from 13 to 17 words. These keys subsequently aid in the creation of the JWT, which possesses an expiration duration. For enhanced security, this token has a relatively short lifespan and needs to be

regenerated every few months (depend on the admin's permission). A concise expiration period inherently boosts the security profile of the JWT. Importantly, with this secret passphrase, users can regenerate their public and private keys as needed. When required, the token can be regenerated using the aforementioned public and private keys.

Here are some examples of secret passphrase and public and private key related to them.

Secret key1: affect direction triangle produce shelter him wonderful acres zipper huge score slept made article search lay bit

Public key:

-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDs0GLED4HKFo5s/ID3bT2hJ/9g
x3ibjOx0YgOGagx5s2NwrIC0sfRzB4+6Rk5IAGqA3GM2GNG8N4UFg808E834V3E9
kxQyvNN04eNheECe19ex56yoDOzWfp1upunVIa8A6nxfW8v4v9FBgnrDbMf8AIAI
JtFUVom/jdU1KbQOuwIDAQAB
-----END PUBLIC KEY-----

Private Key:

-----BEGIN RSA PRIVATE KEY-----
MIICXQIBAAKBgQDs0GLED4HKFo5s/ID3bT2hJ/9gx3ibjOx0YgOGagx5s2NwrIC0
sfRzB4+6Rk5IAGqA3GM2GNG8N4UFg808E834V3E9kxQyvNN04eNheECe19ex56yo
DOzWfp1upunVIa8A6nxfW8v4v9FBgnrDbMf8AIAIJtFUVom/jdU1KbQOuwIDAQAB
AoGBALQXGTT+kfZMRw2szKrdQYP/9d70cszgU6WCMCoVFd2rRVEXbl18A2lC89N1
yexJnLTYZP5ry3w2QIvcGsS4TuY5jeRDI/Q4uxkMVOYGjCW86NtZjoJ8geaTSnmC
cWhii9Nr2wzNYTYfmq0238tmHYtubIpEtzqzwky5OxYV0kbZAkEA+3qTd84FTeiQ
XQ8c80sqVwGePEhgdvwAiWL/P0oUc9dJWcJS+Yn8YPSeZM9K79VSl/XwB+coV4xp
biyKf1vFlwJBAPESUL75e2ByPyqTRz/tjwshHs77yBoSNRsv4W+26giZXWEuNoWT
l7NRYFFIHkkoDaNH8ouwTSxLOsTUlRuajH0CQAogWnXVjvMfLUkCBclqOm88enG0
/GVuKltd6CdVRVOQ1LxPjeXMf6Qr1YD7s+nKbkP+PEclMMOtvMUZ+A2+1UsCQDKh

KhwxwVusIuAKNniSp+wqdJH8BzaShFzFXY9c1yIfM6FpV0IOkVmzyYrInrO2mcal
Iad8y3h2BE26Z+Z4OvECQQDXmOmuc2+NbXY28l+hHcMNBfi/VtBzfDZ2c+qilb+r
qRbrhRRPeCSQPlpqmJ9CgIQ6y7MQST8taDmM2Inbj+Nd
-----END RSA PRIVATE KEY-----

Secret key2:

measure mud gentle combination situation damage somewhere speak author further off solid upper therefore whale away design search

Public key:

-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCyM/P9jd8ewpixr0U3MzuscbbN
YpzyvunX9W7oB1VboJp9/LogURtzJR+s3fwvmA21FHmDwdYoW5PRFnWlB/TWr74v
nrzeLmqQW8pasBQVzWMBpC/dNPQNWXTysUWglfhm9XFwXNCok6wORGED9HKn//8n
S4RvKdjk6SL/TQnmRQIDAQAB
-----END PUBLIC KEY-----

Private key:

-----BEGIN RSA PRIVATE KEY-----
MIICXQIBAAKBgQCyM/P9jd8ewpixr0U3MzuscbbNYpzyvunX9W7oB1VboJp9/Log
URtzJR+s3fwvmA21FHmDwdYoW5PRFnWlB/TWr74vnrzeLmqQW8pasBQVzWMBpC/d
NPQNWXTysUWglfhm9XFwXNCok6wORGED9HKn//8nS4RvKdjk6SL/TQnmRQIDAQ
AB
AoGBAJpwFgtvUafZ4/VRvb2qJBQ99Lwos3ZY6FZl+TkTafFfzaRUS4ZIZG61BK+P
LsickXyWgv0iFxSg0QlK2qgsrg2QoO9QzvLUAtt2iCXCjW46A3tFVWcpx68CC7cL
MfDU9k42dEwCHxDp7WknyBX7CEWFbJQb8gU0MJK20lToonVBAkEA7ei4bnQCKXS9
a/C9+3IGFksQ7HPOyNBIBVA9RV/wK3vjlgcpDUmzVxEbbwtytTs0f4Gjp8+5o+2f
Kur+QpTH0QJBAL/A9Sw6E8P+ZQnlFhslUaL+C3E0xchCnXaH2tv+THshktDUXUSr

NuBUrNkR15CCPzt7vrWjyJesmWDb+JueCDUCQD0zX7ZyO1gkwtGwpX64j15OwzTA
edJo2g4b3RcqneLhxOMERog3jF36dZ80R7bdWxzt4Ya6xhuodgiZWP0RvvECQQCl
21K1NG7QQgRG8L2UMU1RfAeNnaXNN8FXOt8VFfo1Lq78rhMWSDmpA9SV1RbtSZt
D
6h7koYvplUL9Qobgo2pZAkBz3WoCz0TiwbiYn/xBLtqe2HWxSlMYpfWUR3qaMQyA
FDMZJyeclR+VPKgOk/a1hYdSg/7sawBPNPyPxYENWvRN
-----END RSA PRIVATE KEY-----

Token:

eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjEwMTY5MzQwOTIwNywidXNlc
m5hbWUiOiJqaGFpanNjIiwib3JnTmFtZSI6Ik9yZzEiLCJpYXQiOjE2OTM0MDkyMDd9.g
W-_9KXNzEdrlV0q2XNLUp9TuSL3LeqCxE3Uf9MRfhXlRC_c06ol-
7YtIYUDzbnvIitOLULBN4Hq4Pd7EHR5ZpIqYFPDqrs91FPGAOmPxU23-
PqyGwzKQjX4fKxnkWPBGGxMmSpk7PjMvPijVxVJvbKxuJjdezfz7_J2w0RD9Ow

3.2 User Interface (UI) implementation

In the subsequent discourse, attention will be devoted to the fourth layer, conventionally referred to as the application layer. This particular layer assimilates data transferred from its predecessor, the third or middle layer. Functionally, the application layer provides a quintessential interface, presenting data in a manner favorable to academic and professional examination. Users are granted the capability to access their designated accounts, therein obtaining the current status of their respective devices, perusing pertinent information, and initiating direct engagements with said devices.

The application layer is divided into two distinct sub-layers:

1. The primary sub-layer is predominantly concerned with the authentication of users, necessitating rigorous verification of their credentials.
2. The secondary sub-layer, in contrast, is primarily oriented towards facilitating interaction with the third layer, serving as a conduit for device communication.

3.2.1 Authentication in UI

In the designated sub-layer, we have introduced an authentication mechanism to enhance the security of the fourth layer. Specifically, we have employed an authentication process for web users, facilitating their ability to securely log into their accounts.

We have adopted the Lightweight Directory Authentication Protocol (LDAP) as our primary authentication mechanism. Beyond mere authentication, LDAP also supports the storage of customer profiles. Integral to the operation of OpenLDAP is the necessity for a schema tailored for authentication. Accordingly, we have designed a schema to manage both users and administrative roles, ensuring streamlined management and enhanced security.

The LDAP system we have implemented is divaricate into two main components. The first component involves the deployment process, which utilizes a docker-compose file containing the OpenLDAP image alongside a web interface, phpLDAPadmin.

The user interface is presented in a graphical format, allowing administrators to manage user profiles with ease. Additionally, an Application Programming Interface (API) has been developed to bridge the user interface and the LDAP container. This API serves a plethora of functions: querying user profiles, verifying passwords, modifying passwords, adding email addresses, and more. For backend interactions with OpenLDAP, we have integrated the ldapjs library.

Figure 3.18 LDAP User Interface

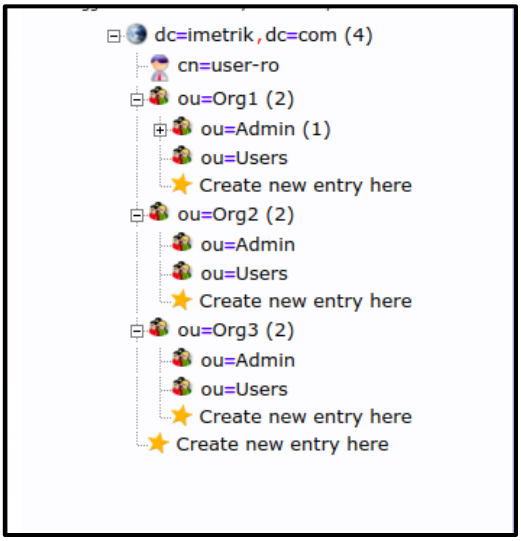


Figure 3.19 Schema Created in LDAP

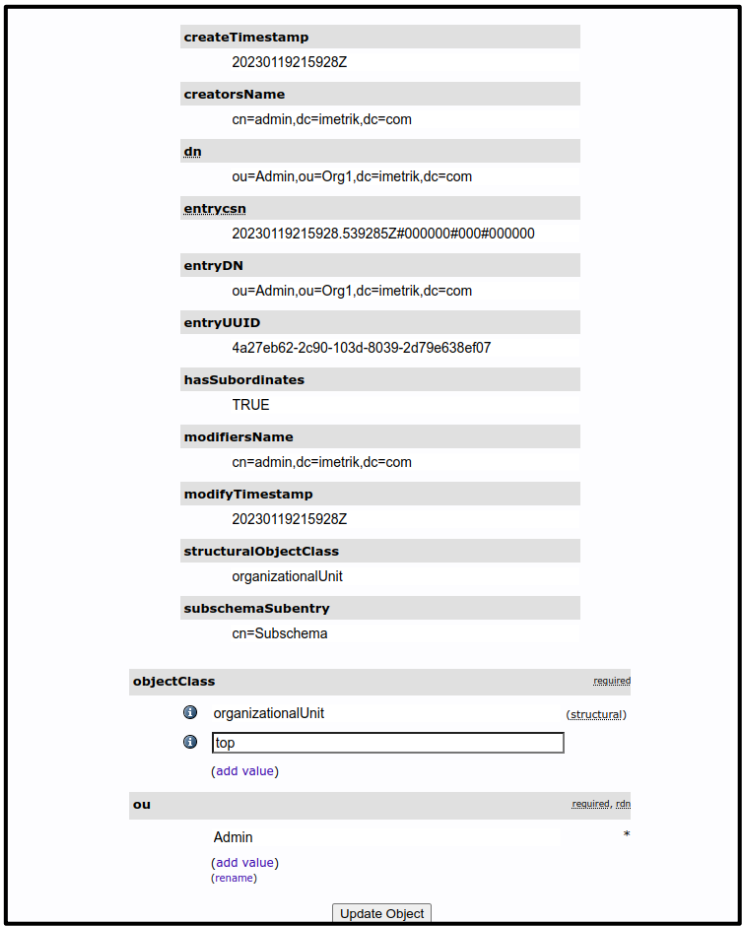


Figure 3.20 User Profile in LDAP

3.2.2 Web interface

The second sub-layer within the fourth layer is designated as our web interface. Once users authenticate and gain access to their respective accounts, they are endowed with the capability to engage with the platform. It is incumbent upon the administrator to incorporate both users and devices. For the development of the interface, we employed ReactJS, a renowned JavaScript library that emphasizes component-based architecture.

We have designed a dashboard, enabling users to connect and interact within their designated space. The registration of these devices is orchestrated through a smart contract, tailored for the efficient management of devices and users. Upon successful registration, a unique UUID is generated for each device. Within this section, we also emphasize the importance of data integrity. We meticulously examine each block to ensure that its hashed data is properly linked and chained, safeguarding the consistency and reliability of the entire dataset. Visualization techniques have been implemented using an array of libraries, including react-plotly.js.

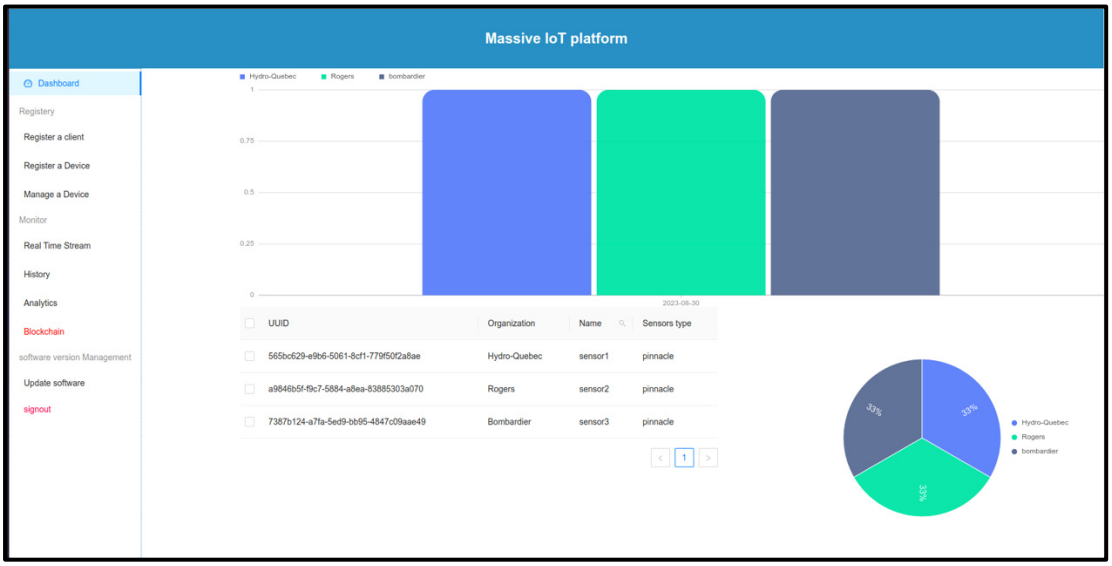


Figure 3.21 Web User Interface Dashboard

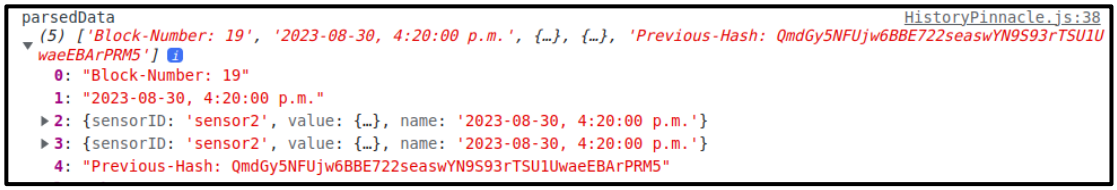


Figure 3.22 Block of Data Requested by User to be Visualized

Figure 3.23 CID's Requested from Hyperledger Fabric

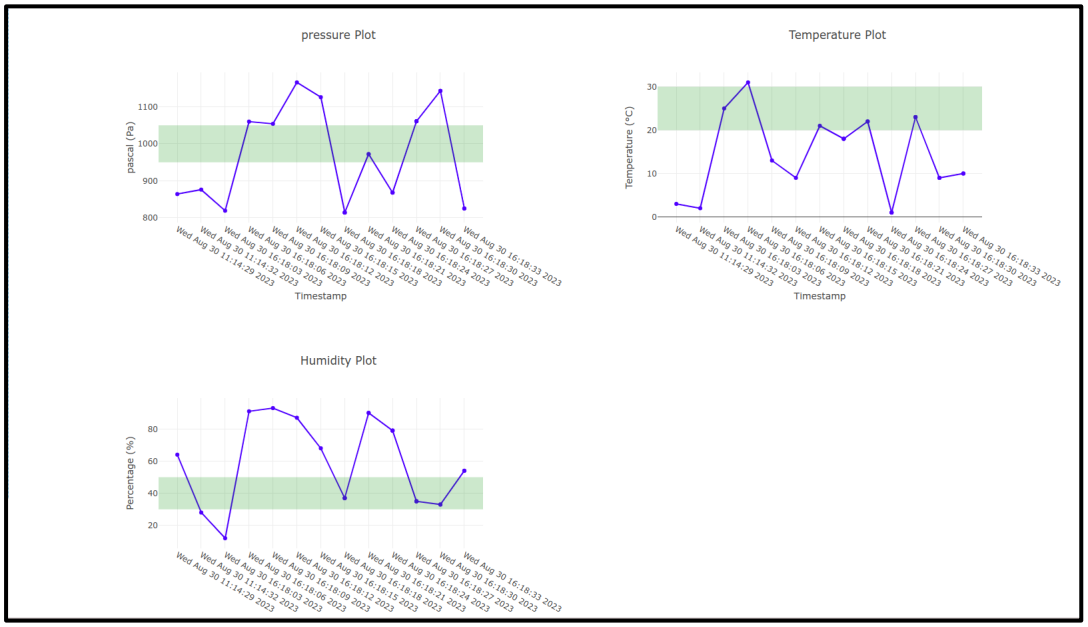


Figure 3.24 Visualized Data from a Sensor

3.3 Real-time data visualization

In this section, we clarify the methodology employed for the implementation of real-time data visualization. Utilizing the Hyperledger Fabric (HLF), one can essentially retrieve historical data. A notable limitation of Blockchain technology is its non-real-time nature, subject to delays contingent on network performance. To mitigate this challenge, we have instituted a mechanism that enables the acquisition of real-time data via a websocket. Subsequent to this, the data is presented through a developed web-based user interface.

Web Socket, a protocol grounded on remote server-client interactions, was formulated to minimize communication overhead. Its security framework mirrors browser-based models. Communication initiation hinges on a handshake process between the client and server (Kumar N.V. and Kumar P. 2020). During the implementation phase, as data is procured from the Kafka consumer, a websocket is established, suspended for a handshake with the end user. Upon server validation of the JSON Web Token (JWT) - previously outlined - to authenticate the user, the handshake reaches completion. Consequently, users can view real-time data within the user interface.

3.4 Data Encryption/Decryption method

In the following section, we shall discuss encryption methodologies. Encryption is vital for ensuring the security and privacy of data, IoT devices, with their limited computational capabilities, encounter challenges in implementing robust cryptographic standards like the RSA algorithm. However, the selection of an appropriate encryption method for these devices underscores the need to safeguard data effectively. This is evidenced by the emphasis on optimizing computational load, memory requirements, and energy consumption—all factors critical for effective encryption in constrained environments. The ongoing discussions about RSA key lengths further highlight the significance of strong encryption in protecting information from potential vulnerabilities or unauthorized breaches (Nartey et al. 2021).

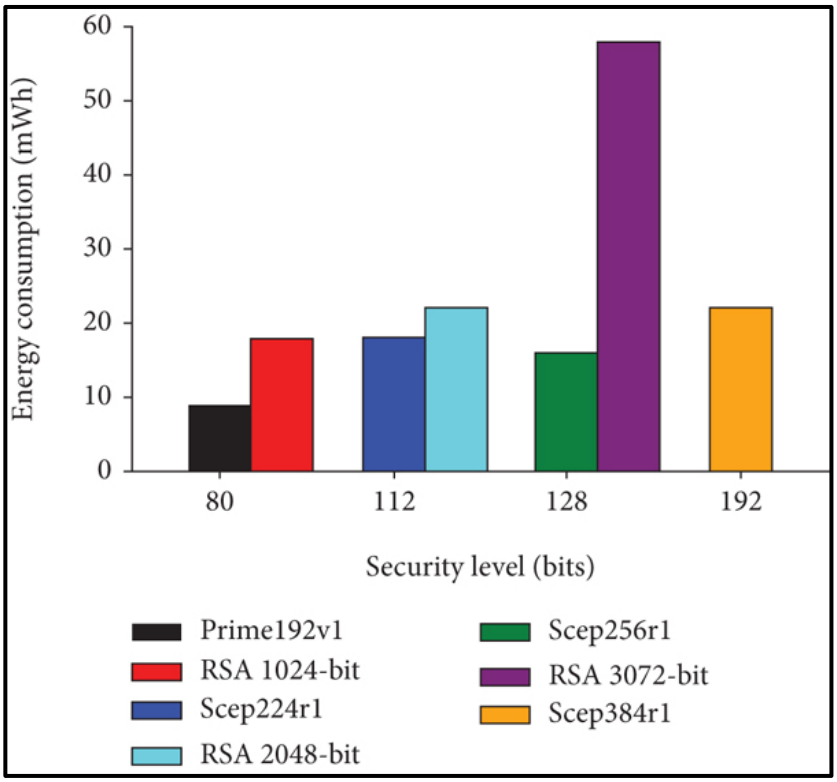


Figure 3.25 Energy Consumption for Different Algorithms
Taken from Nartey et al (2021, p. 14)

Owing to its comparatively lower energy consumption relative to other RSA encryption variants, we have selected RSA with a 1024-bit key size. We have implemented end-to-end

data encryption to ensure data integrity and confidentiality throughout the transmission process. Data can then be decrypted by the end-users utilizing their unique private keys. The encryption is executed using the public key, whereas decryption is facilitated using the private key.

Example of encrypted data:
"YS1u2eYzHTO0Moida/mBDmRX/ib5pIFhx1TpUvCn9OVTBoaDTnY4hiFiG0V066LZww
tagRZuAS1tG+bmeJOOMrffx1zK/LwvS77LstCV4+WpAgtctrEnBn8JI/MB9G7PEG2vEyiH
liUt1NPVvLvvo+lv5n/3u5A/sBsE0fCdVkxnKsPiUpxNZr6OPdBsiYKoSe4WzkHBPXHJ2Ja
LCqtIOCh2teDuf/ATQ6GyV//n1rAUlZ7HreWR0Kvxrrm1dvSCi+Hsnzq+zlnCwwud37oNx
Xpcjj9e1yCywWzMxPB0kTOaPyngkNa42AvSQYAvn5pIGruvL3STosyefZ0bvsvO+y9QA
QcA686tLKAhdHSN0hlscuKZDf4acJmGLgDG2ZNOS/57TERw7N/2Fw07l0axE42ZU9z9y
VAyq24MjzKOFu3EV8eCovqSUoTH/7CFj8KNnFBHzs4GfY99i9b4Y95S4Br+BW8vLN0t
R8VSTjC/cb4N2IvpI1huS+nTACM+xDptfpnAwbYg29/ZKlt5sW3fnYycYxISuLvARnnW
mP7y4DrOOu+oatI0ph+PDTJEL5/V5YOldB8OJ+Ef2KZUHhTccMUSkyFaeTLbV6ZScjwg
n8ieqBHDJSxt1f1jmPBFaT+5YOYeGgzPvlIKMUimKB3wy9cj1mYvSWjCTSw9VJSyutpC
8TJLa13EUkGnse+nRtxVCpH5bne/4tI2g70egtyUW+yi1cEbA9vsETgmtM5Wc9aJoMhBai
mF/szFs4pBO/E8YZw7V018PHl2KSc94oXAaAwQZiR3rOqPUd6mUzQkeNSZWPM//y0b
ZJVpYKgkaCY3rBZn4H8qPy0n7jokWHbEpA=="

3.5 Chapter summary

In this chapter, we discussed the implementation of the third and fourth layers. A comprehensive exploration of each step was undertaken to ensure the effective integration of essential technologies required for the deployment of the platform. We examined the incorporation of real-time data streaming technologies, specifically Apache Kafka, and how it has been synergized with the decentralized database, IPFS. Additionally, the method by which the database is integrated with the blockchain system, known as Hyperledger Fabric (HLF), was elaborated upon. Our discussion further expanded on the deployment of Kafka and IPFS.

For the deployment of our HLF, each step was described, highlighting procedures such as joining channels, understanding the roles and responsibilities of smart contracts, and the use of the SDK. we addressed the SDK utilized and elucidated the procedures to authenticate each user. The paramount importance of security was acknowledged, and as such, we traversed the topic of end-to-end data encryption. This is critical in fortifying the entire platform. Additionally, we outlined methods to verify the integrity of data received from various devices. Lastly, an in-depth analysis of the fourth layer's implementation was provided, detailing all sub-layers pertinent to authentication and visualization processes.

CHAPTER 4

Results and Evaluation

In this chapter, we shall present a comprehensive report of the research findings obtained from evaluations and analyses of the platform. The analyses will be divided into three segments:

1. In the first section,
    a. Assess the performance of Apache Kafka.
    b. Pay attention to its latency and system resource utilisation.
2. In the next section:
    a. Examine the performance of Blockchains.
    b. Emphasise transaction rates as well as accompanying latencies.
3. Analyse system resource utilisation in the third segment.
    a. Examine the use of the central processor unit (CPU) and random-access memory (RAM) in particular.

For assessing the performance of the Hyperledger Fabric (HLF), we employed two primary Hyperledger tools: Hyperledger Caliper and Hyperledger Explorer.

4.1 Kafka performance

In the domain of distributed systems research, an analytical exploration was conducted to assess the performance metrics of Apache Kafka, a prominent open-source stream-processing software platform. Metrics pertaining to performance and latency were ascertained using the Kafka Control Center, an instrumental platform facilitating real-time monitoring capabilities of each constituent broker. The computational environment for the assessment comprised a MacBook Pro with an M2 Pro chip and 16MB RAM, wherein Kafka was operationalized with a solitary broker. Data processing was conducted at a sample rate of 0.01, with the scope of topics oscillating between 50 and 1,000, encompassing both creation and subsequent processing.

Upon analysis under multifarious scenarios, latency was discerned in two principal modalities. The initial modality pertained to the producers' request latency, which unveiled an obvious

correlation between the escalating number of topics and concomitant latency. Pertinently, for the 99.9% percentile of the data set, latency exhibited an increment from an initial 5 ms, culminating at 40 ms.

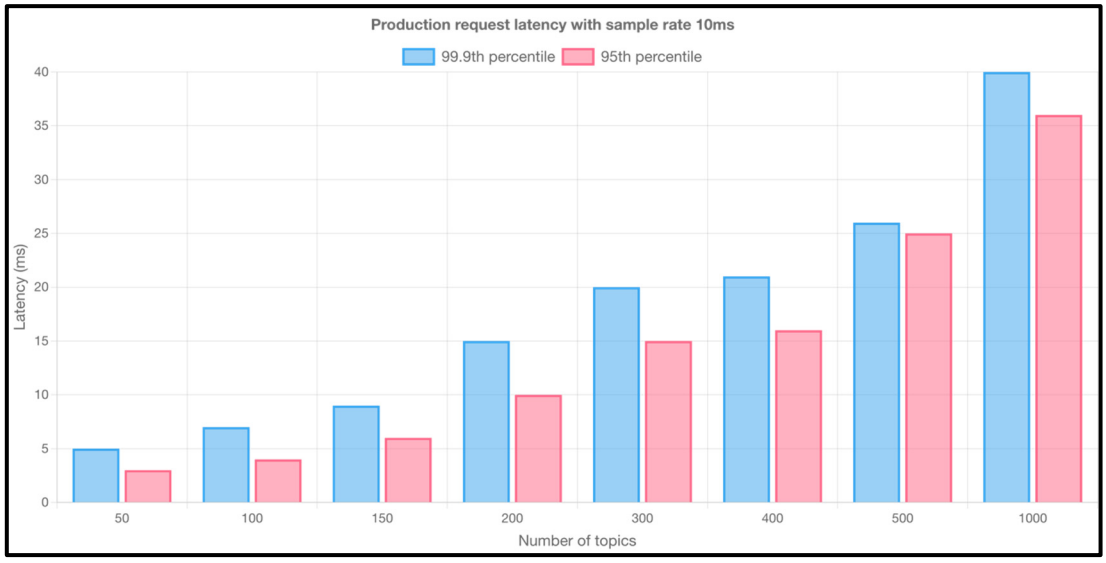


Figure 4.1 Request Latency in Apache Kafka Producer

Subsequently, consumer-derived latency was scrutinized, revealing values commencing at 500 ms and terminating proximate to 700 ms.

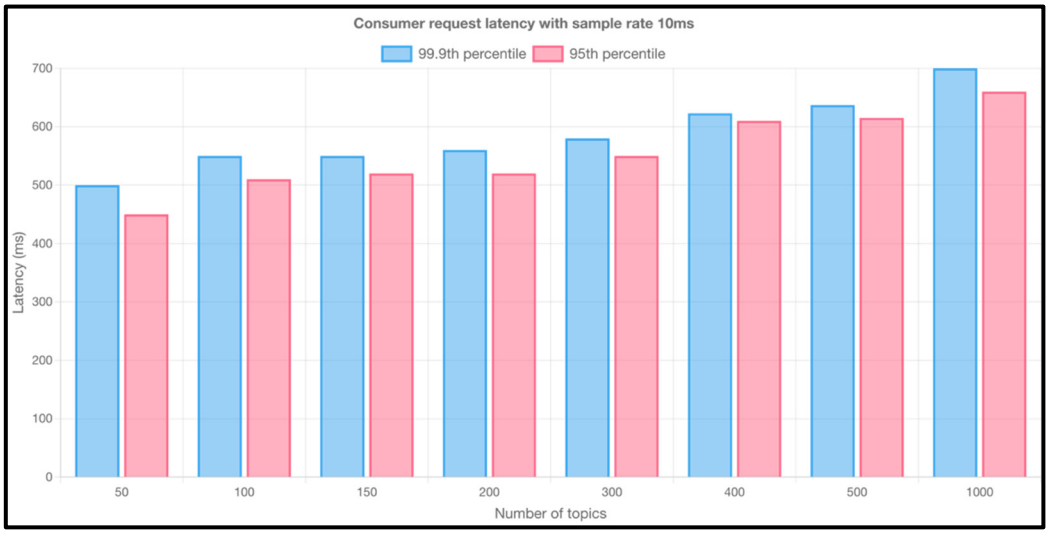


Figure 4.2 Request Latency in Apache Kafka Consumer

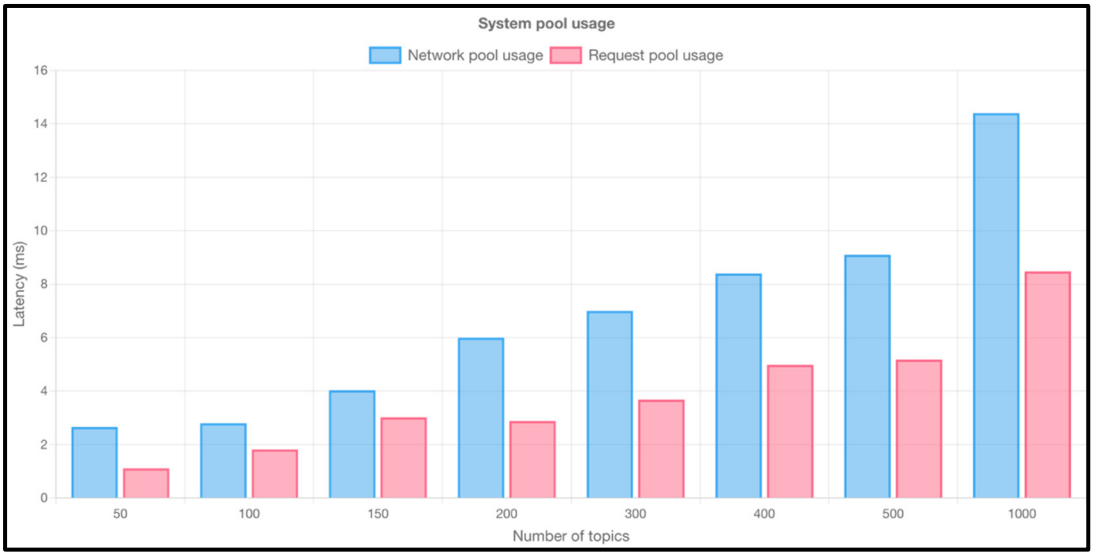


Figure 4.3 System Pool Usage in Apache Kafka

Expanding the analytical purview, system pool usage was subjected to rigorous examination. This analysis was bifurcated into (Confluent Documentation n.d.) :

1. Network Pool Usage: A metric signifying the mean network pool capacity utilization across the entirety of brokers, effectively quantifying the temporal percentage wherein network processor threads were operative.

2. Request Pool Usage: Serving as an indicator of the median request handler capacity employed across all brokers, quantifying the temporal duration when the request handler threads were in an operative state.

An emergent trend was the augmentation in system pool usage concomitant with an expansion in topic numbers. The findings elucidated an anticipated amplification in the non-idle system usage percentage. This scholarly investigation of Apache Kafka highlight its inherent scalability and efficacy. However, it also foregrounds the consequential augmentations in latency associated with intensified system demands. Forthcoming research endeavors are thus recommended to investigate potential optimizations, aiming to attenuate latency repercussions while accentuating resource utilization efficiency.

In our comprehensive evaluation, we observed a failure rate of zero percent. This indicates that there were no issues related to data queuing and processing. Further examination of this result suggests an optimal efficiency in the system's handling and management of data. This absence of data backlog and processing delays is suggestive of a robust and well-optimized system infrastructure, which warrants further investigation to understand the underlying factors contributing to its commendable performance. To enhance the system's availability, we can also consider augmenting the number of brokers.

4.2 Blockchain performance

In this section, we assessed the performance of the blockchain using our specified settings and configurations. The evaluation comprises two distinct segments:

1. Monitoring conducted via Hyperledger-Explorer.
2. Analysis performed using Hyperledger Caliper.

Hyperledger provides a suite of tools designed to optimize and visualize blockchain operations. Caliper is one such tool, dedicated to benchmarking Blockchain implementations. Through its predefined use cases, it evaluates the performance of several blockchain solutions, including Ethereum and multiple Hyperledger systems, producing detailed performance reports. Another

significant tool from Hyperledger is the Explorer. Explorer offers a user-centric platform to view and interact with blockchain data. Users can view, invoke, and query various Blockchain details, and the tool also boasts the flexibility to integrate with diverse authentication platforms, enhancing its usability and adaptability (Punathumkandi et al. 2020).

For these tests, our experiments were conducted on a MacBook Pro M2 Pro, detailed with the following specifications:

- CPU: Apple M2 Pro chip (10-core)
- RAM: 16GB unified memory

4.2.1 Monitoring with Hyperledger-Explorer

This study aims to investigate the Hyperledger Explorer, focusing specifically on its implementation perspective. Hyperledger Explorer is an essential tool that facilitates the monitoring and interaction with blockchain infrastructures. A brief overview of the file structure pertinent to this tool is presented:

1. Test-network.json: This file encompasses details concerning the specific network underutilization.
2. Config.json: Within this document, one defines the network organization and associated peers.
3. Docker-compose: Herein lies the definitions for the Hyperledger Explorer containers and the corresponding environment configurations.

For successful integration, it is paramount to incorporate the credentials from Hyperledger Fabric (HLF) into our respective files, notably the Crypto-config. This integration ensures that Hyperledger Explorer establishes an uninterrupted connection with the Blockchain.
Hyperledger Explorer is a versatile platform, offering users the capability to interact with and scrutinize various blockchain operations. It provides access to a plethora of information, including but not limited to, details on smart contracts, individual blocks, transaction specifics,

and transaction counts. Subsequently, we shall delve into the salient features of Hyperledger Explorer.

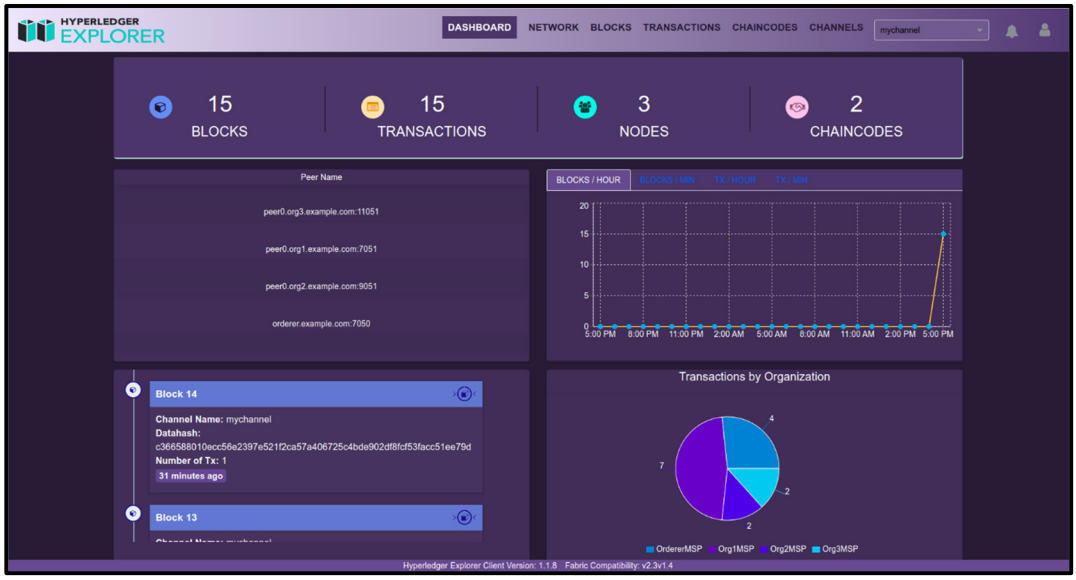


Figure 4.4 Hyperledger Explorer Main Dashboard

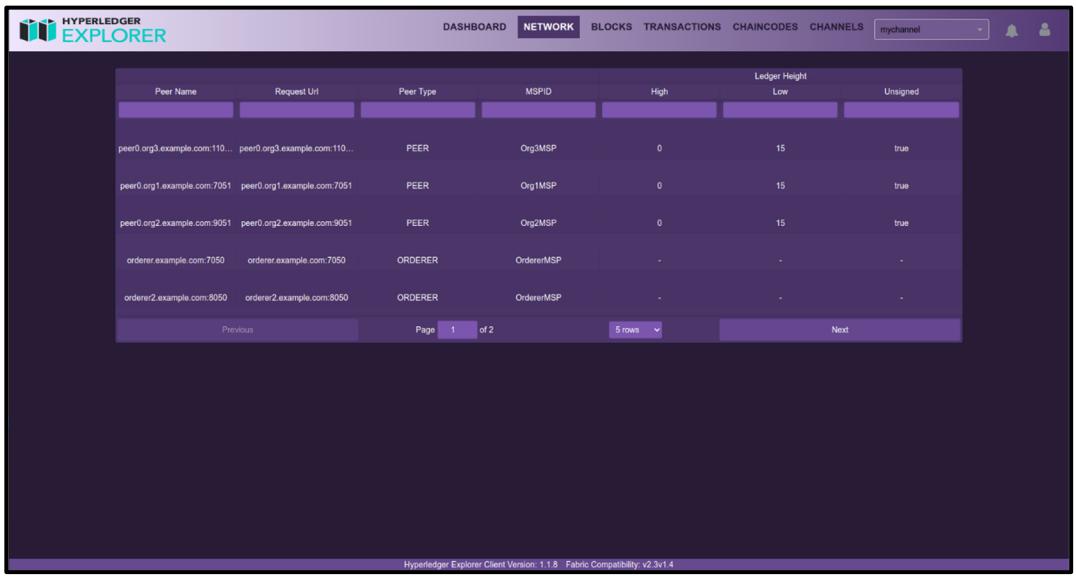


Figure 4.5 Running Network in Hyperledger Fabric

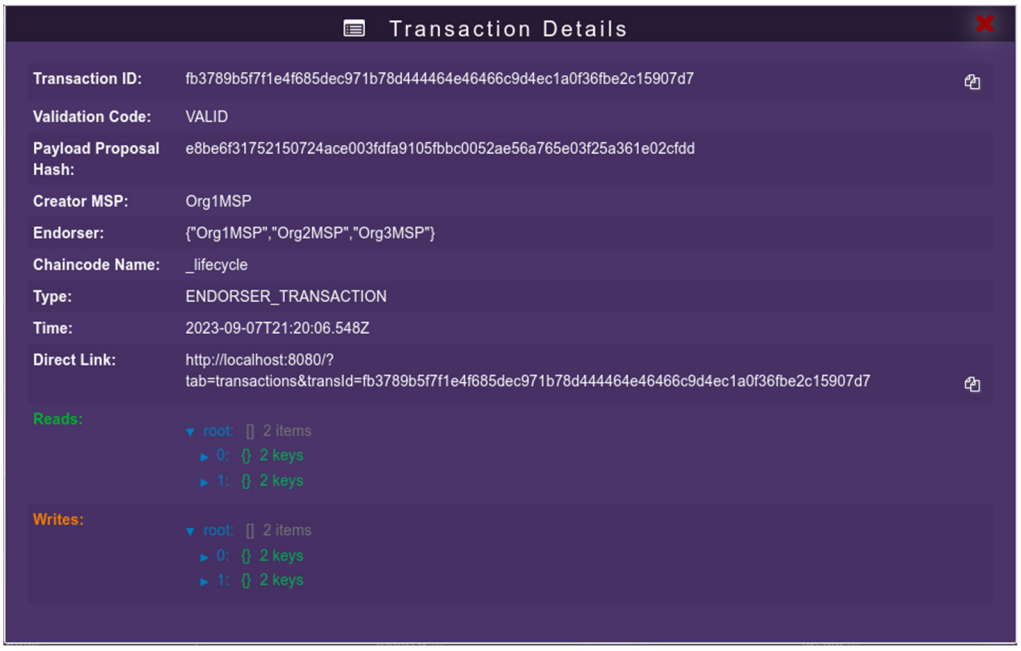


Figure 4.6 Transaction Details on Hyperledger Fabric

Figure 4.7 Details of Each Block Created by Hyperledger Fabric

4.2.2 Evaluation with Hyperledger Caliper

In the following subsection, we employed Hyperledger Caliper to evaluate the performance and structure of our Hyperledger Fabric (HLF) implementation.

The architecture of Hyperledger Caliper can be delineated into the subsequent categories:

Benchmark Configuration: This governs the specific test scenarios that are run. Within this configuration file, we determine the number of transactions (Tx), rate controls, and the quantity of workers assigned to execute transactions. Additionally, the transaction duration and rate control for inquiries from the blockchain are stipulated herein. Concurrently, the monitoring of peers and orderers is conducted.

Network Configuration: This section delineates the specific nature of our blockchain. It includes the naming conventions for channels, coupled with detailed information on the credentials of the participating organizations.

Workload Configuration: This part pertains to the JavaScript files, detailing the procedures for executing transactions and making inquiries to the blockchain.
In Hyperledger Caliper, the worker plays a pivotal role in benchmarking (Anon n.d.):

Activation: It starts upon receiving a message from the manager to initiate the next round.
Workload Generation Loop: Primarily, the worker operates within this loop, which involves:

1. Rate Controller's Delay: The worker waits for the rate controller, which determines when to process the next transaction, based on set rates.
2. Interaction with Workload Module: After approval from the rate controller, the worker lets the workload module set up and send the transaction details to the System Under Test (SUT).
3. Progress Reporting: Throughout, the worker updates the manager about its progress.

In essence, the worker in Caliper manages transaction benchmarks based on rate controller cues and updates the manager (Choi and Hong 2021).
Sample Reports: These are generated post-experimentation and provide insights into the performance metrics. Here is how the latency and throughput will be calculated in Hyperledger caliper (Choi and Hong 2021).
Latency: The time difference between when a transaction is sent and when it's committed.

a. Latency = Confirmation time – Submit time

Throughput: Indicates the number of transactions carried out per second.

b. Throughput = Committed txs / Time ∈ seconds

4.2.2.1 Throughput results

In our recent series of experiments, we aimed to understand the performance scalability of our system. We varied the number of workers involved to see how this might affect the throughput, specifically using teams of 2, 5, 10, and 20 workers. Alongside this, we also manipulated the transaction counts to study its effect on performance. The tested transaction counts were 1,000; 2,000; 3,000; 4,000; 5,000; 10,000; and 20,000.

The results from these tests were insightful. When observing the Transaction Per Second (TPS) rate, the range started from a minimum of 366.4 TPS. Interestingly, as we increased the transaction number to its maximum, the peak throughput also soared to as high as 798.3 TPS. This gave us a clear picture of how the system performs under varying loads.

However, it is worth noting that while our system was able to handle higher loads, there was a trade-off in terms of reliability. The failure rate, which started at a commendable 0%, increased to 10% when the number of workers are increased to 20 as we reached the higher ends of our transaction count tests 20000 transactions. This indicates potential areas of improvement in our system's resilience and reliability under heavy load conditions.

Additionally, we also conducted a separate "Read an Asset" test. In this test, the throughput results showed a similar trend. The TPS ranged from 655.2 at its lowest to 970.5 at its peak. It demonstrates the system's consistent performance across different types of operations.

In conclusion, these experiments provided valuable data on how our system performs under different conditions. While we are pleased with the scalability in terms of throughput, the next steps would involve addressing the reliability concerns highlighted by the increasing failure rate at higher transaction counts.

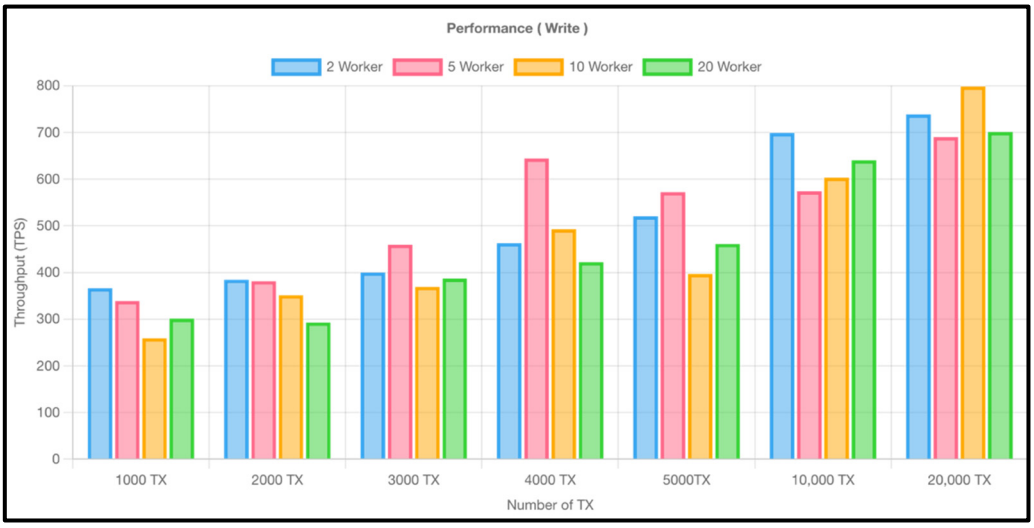


Figure 4.8 Transaction Throughput (TPS) in Hyperledger Fabric

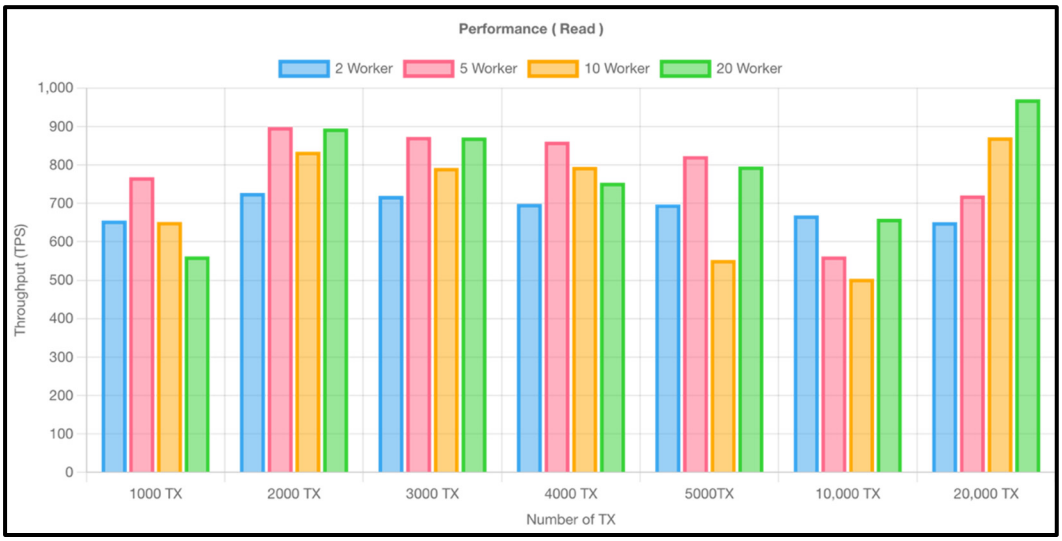


Figure 4.9 Inquiry performance (TPS) in Hyperledger Fabric

4.2.2.2 Latency results

The conducted study aimed to understand the influence of varying numbers of workers and transaction counts on the latency of a Hyperledger Caliper benchmark. Four sets of workers -

2, 5, 10, and 20 - were paired with transaction counts ranging from 1,000 to 20,000. The results indicate a clear trend.

Minimum Latency: As we increased the number of workers, there was a notable rise in minimum latency. For 2,000 transactions, for example, the latency increased from 1.415 ms (2 workers) to 1.99 ms (20 workers). Yet, intriguingly, for 10,000 transactions, latency dipped to 1.705 ms with 2 workers but escalated to 7.935 ms for 20 workers.

Average Latency: A consistent trend was observed here: as both the number of workers and transaction counts increased, the average latency consistently went up. With 2,000 transactions, the latency went from 3.485 ms (2 workers) to 4.715 ms (20 workers). For the highest transaction count of 20,000, the latency ranged from 11.625 ms (2 workers) to 15.555 ms (20 workers).

Maximum Latency: This too followed a consistent growth pattern. At 2,000 transactions, we saw an escalation from 4.75 ms (2 workers) to 6.445 ms (20 workers). By 20,000 transactions, the latency increased from 16.04 ms (2 workers) to a significant 20.93 ms (20 workers).

In summary, while an increase in workers leads to higher latencies, a rise in transaction counts amplifies this effect, particularly evident in the average and maximum latency metrics.

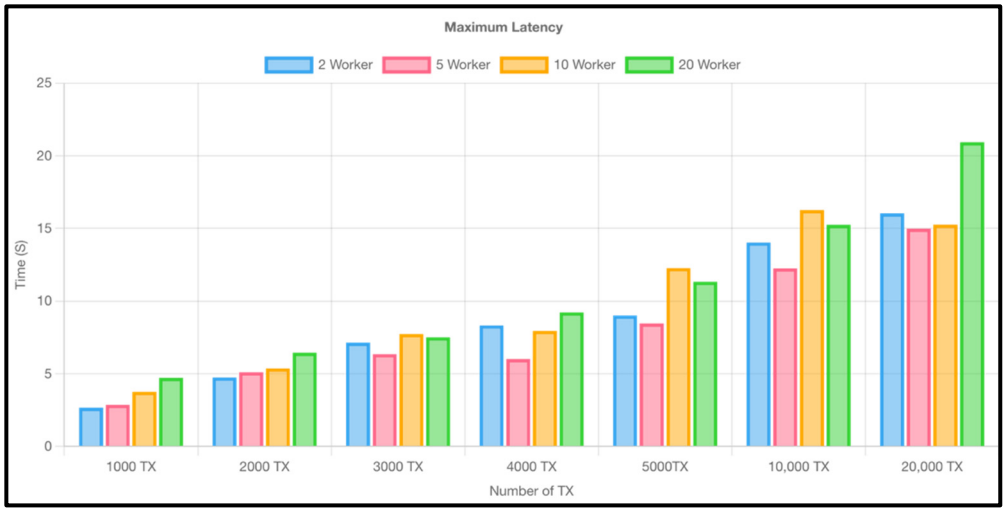


Figure 4.10 Maximum Latency for Transactions in Hyperledger Fabric

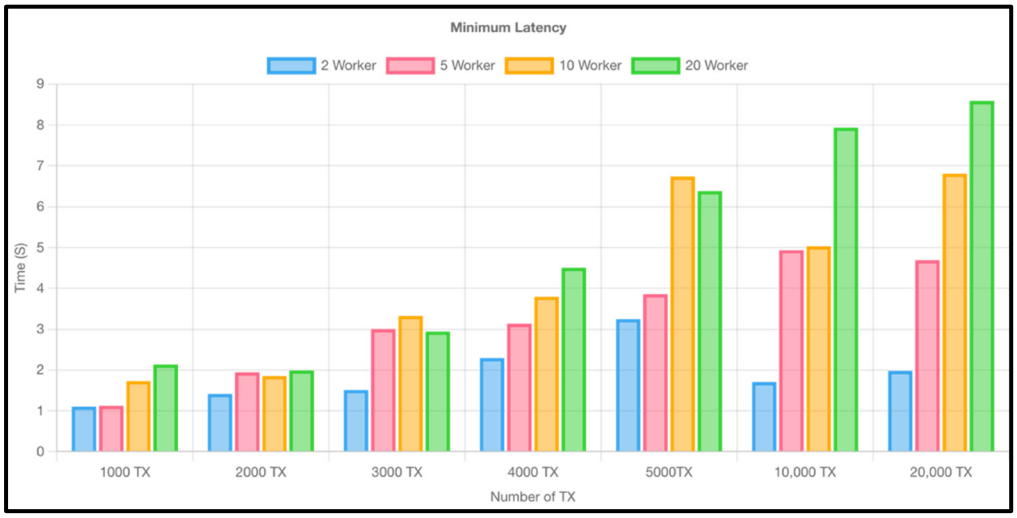


Figure 4.11 Minimum Latency for Transaction in Hyperledger Fabric

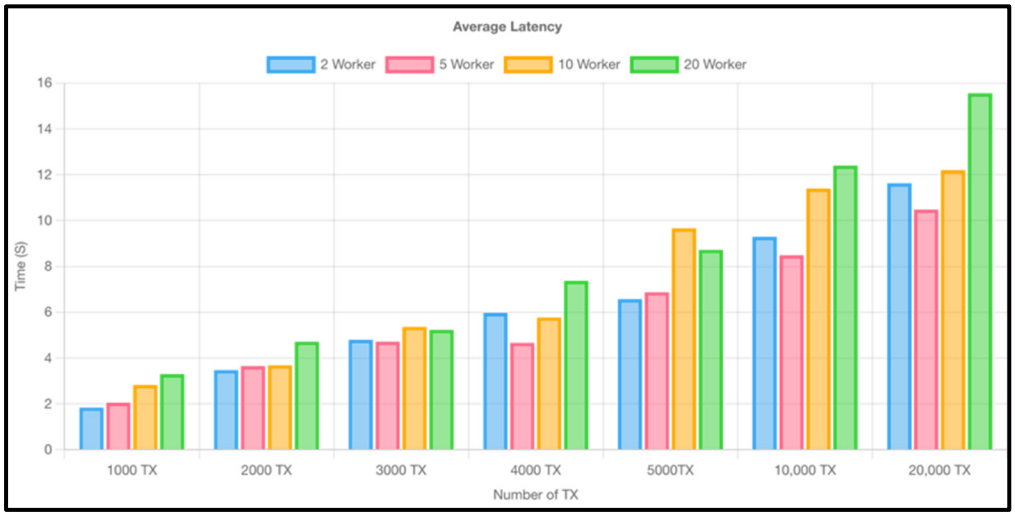


Figure 4.12 Average Latency for Transaction in Hyperledger Fabric

4.3 System resource utilization

In the following section, we shall provide an in-depth exploration of the procedures and methodologies employed to monitor CPU usage and memory of the platform. To optimize and oversee resource consumption, three distinct containers have been utilized. The initial container, known as the 'node exporter,' facilitates the measurement of diverse machine resources, encompassing memory, disk, and CPU utilization. After this, 'Prometheus' serves as our primary data repository. Lastly, 'Grafana', an open-source analytics and monitoring tool compatible with various databases, has been incorporated. It should be noted that Prometheus is contingent upon the functionality of the node exporter, while Grafana, in turn, relies on Prometheus.

The execution phase will encompass the initiation of all containers. Subsequent to this, we shall access Grafana to integrate the data source and establish a dashboard, thereby visualizing network consumption in a comprehensive manner.

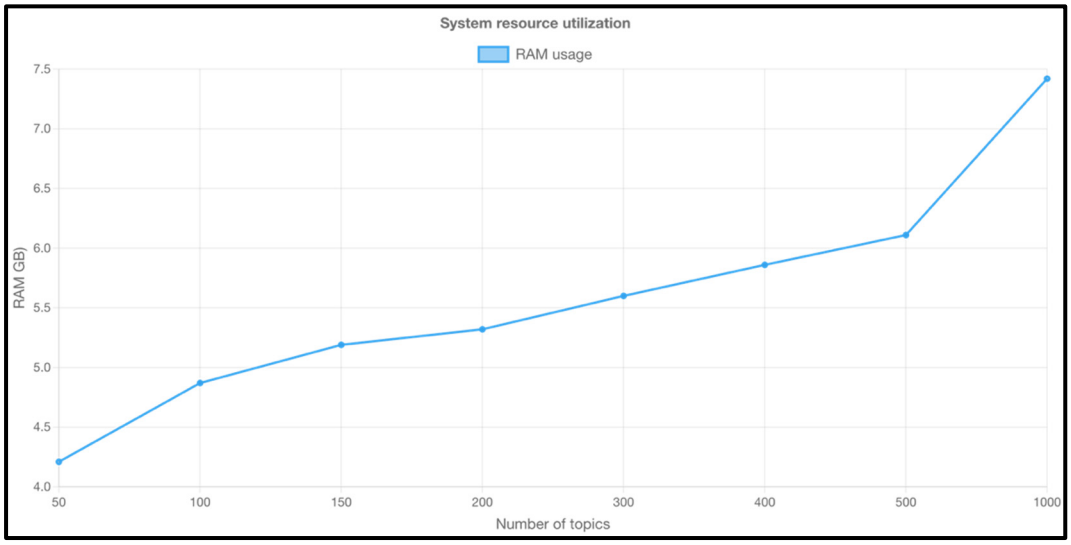


Figure 4.13 RAM Usage of Apache Kafka

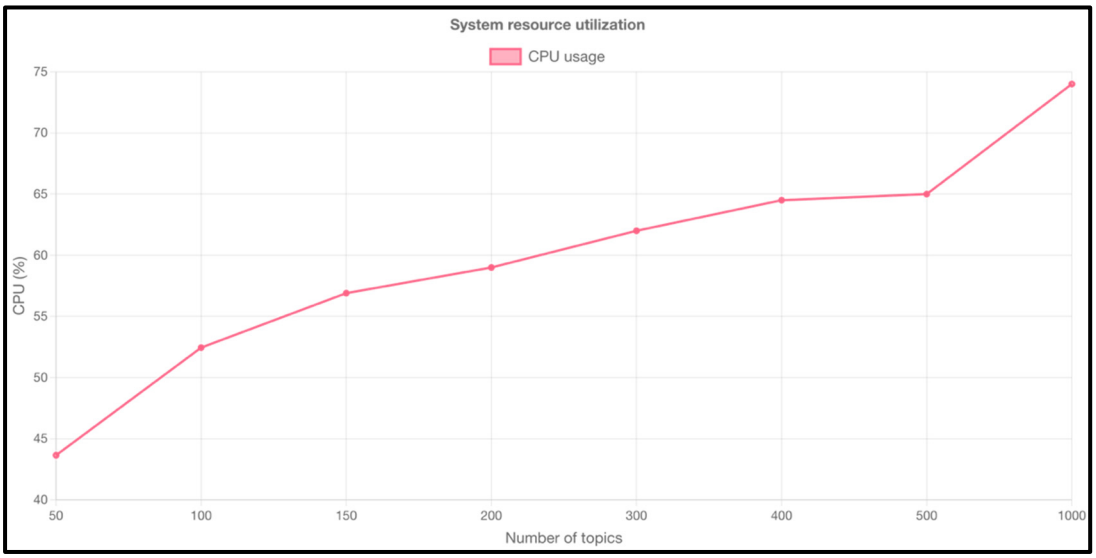


Figure 4.14 CPU Usage of Apache Kafka

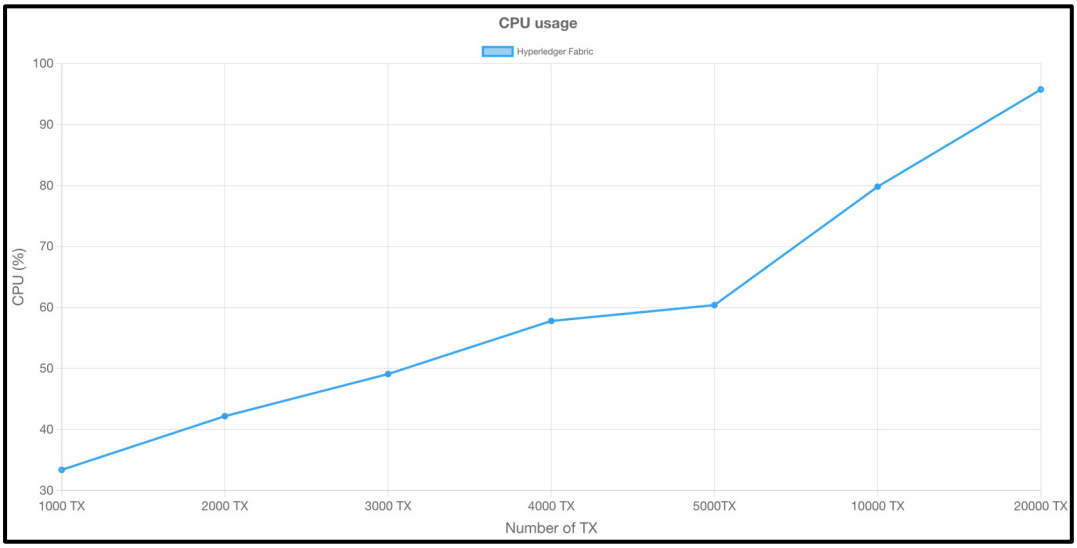


Figure 4.15 CPU Usage of Hyperledger Fabric

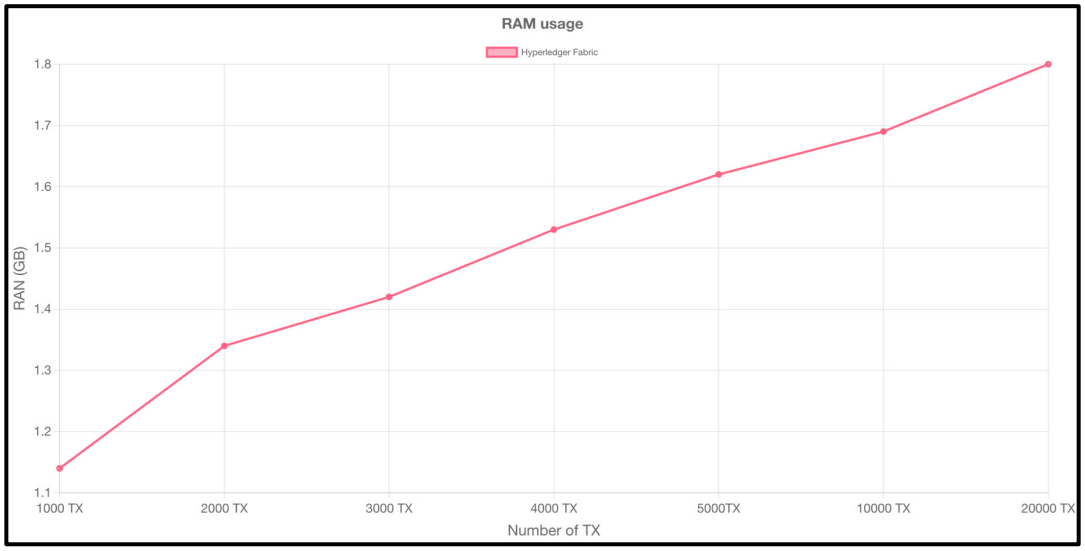


Figure 4.16 RAM Usage of Hyperledger Fabric

4.4 Chapter summary

In the present chapter, we undertook a rigorous analysis of Hyperledger Fabric in conjunction with real-time data streaming. Utilizing the Hyperledger Explorer, we successfully monitored the Hyperledger Fabric (HLF) and ascertained the throughput of both Kafka and HLF.

Further enhancing our methodology, resource monitoring was implemented utilizing tools such as node-exporter, Prometheus, and Grafana, enabling a detailed observation of CPU and RAM consumption associated with the aforementioned technologies. Our findings indicate that as the data volume escalates, there is a corresponding increase in both latency and resource utilization.

CONCLUSION

In the following chapter, a thorough examination of the platform in question will be undertaken. Emphasis will be placed on illustrating the inherent advantages and delineating the features that confine its distinctiveness. In the spirit of scholarly rigor, it is important to acknowledge the limitations concomitant with the present research and the challenges encountered during its empirical implementation. Such recognition not only ensures an objective appraisal but also paves the way for future scholarly endeavors. Concluding this discourse, a prospective analysis will be offered, contemplating potential advancements and innovations. By articulating this prospective trajectory, the intention is to provide a robust framework for future explorations and a clear direction for subsequent academic inquiries.

Discussion

In this section, we elucidate the myriad benefits and advantages conferred by the adoption of this architecture. Through this implementation, organizations stand poised to actualize a plethora of invaluable outcomes and augment their operational efficiency.

Security: Through multi-faceted security protocols, robust protection is assured. Initial user authentication is managed at the primary layer, while an amalgamation of public and private keys engenders the creation of a JWT (JSON Web Token), reinforcing the Blockchain's security. The Blockchain's intrinsic security mechanisms are especially salient. Furthermore, Hyperledger Fabric, a private permissioned blockchain, engenders trust among its user community.

Integrity: The Blockchain functions as an indelible ledger, thus guaranteeing data integrity. Once an entry is etched into the Blockchain, its immutable nature ensures resistance against tampering or alteration.

Privacy: Data encryption forms the bedrock of user privacy. Data decryption and access are restricted solely to users in possession of the requisite private keys, guaranteeing the sanctity and confidentiality of the information. Consequently, sensitive user data remains impervious to unauthorized access or exposure.

Access Control: Administrative personnel are endowed with the discretion to harness and discern the insights derived from the data. This circumscribed access ensures administrators can judiciously utilize analyzed data for decision-making, while concurrently upholding user privacy.

Scalability: Hyperledger Fabric, the framework of our choice, is distinguished for its impressive scalability, demonstrated by its commendable transaction per second (TPS) rate. Moreover, our avant-garde data architecture facilitates adept management of the platform's efficiency. There exists a capability to modulate the data block size responsively, aligning with fluctuating demands, thus assuring unparalleled scalability.

Data Preservation: Harnessing the innate database functionality offered by Kafka, the platform retains data indefatigably. In scenarios where Blockchain latency might surge temporarily, data is securely ensconced within Kafka's database until it is transmitted to the Blockchain. This guarantees unerring data preservation and reliability.

In summary, in the proposed architecture, organizations are positioned to benefit from a suite of enhanced features that amplify operational efficiency. Central to this architecture's prowess is the unparalleled security facilitated by multi-layered authentication protocols and the inherent protective mechanisms of the Hyperledger Fabric. This framework guarantees data integrity through its immutable Blockchain ledger and ensures stringent user privacy via encryption, rendering unauthorized access virtually impossible. Furthermore, administrators are empowered with selective access, fostering judicious decision-making without compromising user privacy. Scalability is a hallmark, with the capacity for dynamic

adjustments to meet changing demands, and data preservation is ensured, courtesy of Kafka's robust database functionality, ensuring no data loss even in high-latency scenarios.

Research limitation

To ensure a comprehensive understanding, it's pivotal to acknowledge that every technological platform, regardless of its sophistication, inherently possesses certain limitations. While efforts have been made to attenuate these constraints, the following elucidates some of the intrinsic challenges associated with the platforms under discussion:

Performance Implications in Hyperledger Fabric: The efficiency of Hyperledger Fabric is influenced by a myriad of factors. Among them are the size of the network, its configuration, and the intricacy of the chaincode. These elements can potentially constrain its capacity to process elevated volumes of transactions.

Cost Implications: Implementing a system that incorporates Hyperledger Fabric as a blockchain and Kafka as a real-time data streaming network necessitates substantial infrastructure and resources. Such requirements can escalate the overall financial outlay associated with the deployment and continuous operation of blockchain applications.

Deployment Complexity: The integration of a comprehensive system involving Hyperledger Fabric (HLF), Kafka, and the InterPlanetary File System (IPFS) is not without its intricacies. The interoperability mandated by these platforms demands a significant degree of technical proficiency, potentially elongating the time and resources needed in the deployment phase.

Maintenance Considerations: Maintaining the harmonious operation of three distinct technological platforms (HLF, Kafka, and IPFS) introduces an elevated maintenance overhead. This necessitates the allocation of specialized teams to constantly oversee, diagnose, and update each individual component.

Limitations Concerning Multimedia Data: Kafka's design does not inherently cater to image or video processing in the manner that specialized multimedia processing utilities do. While Kafka is capable of transmitting and archiving diverse data forms, including binary data

formats like images and videos, it is contingent on encapsulating them into messages that adhere to Kafka's maximum message size constraints. To exploit real-time processing capabilities after retrieving data from a Kafka topic, one might contemplate integrating Kafka with advanced processing tools such as Apache Flink, Apache Spark.

Future work

In the subsequent section, we shall provide a comprehensive exploration into potential areas of further research, as well as pragmatic enhancements that could be pivotal in optimizing the platform under discussion. At the forefront of potential improvements is the integration of a multi-layered blockchain framework. Such a structure is not merely an augmentation of existing architectures but a transformative approach. The multi-layered framework envisages multiple tiers of Hyperledger Fabric (HLF). When interconnected, these tiers serve a dual purpose: they are anticipated to significantly enhance the throughput, which refers to the capacity to process transactions over time, and concurrently, boost the overall performance metrics of the blockchain system.

Moving on, the modified data architecture offers more than just structural changes; it introduces a strategic mechanism to enhance system performance. This strategy is rooted in the continuous analysis of data volume. As the volume undergoes fluctuations, there arises an imperative to adaptively adjust the payload size in the data packets. Simply put, as network congestion intensifies, there is a proportional requirement to expand the data packet size, ensuring that data transfer remains efficient even during peak loads.

Concluding our array of proposed improvements is the nuanced approach to micro-processing management. At its core, this strategy champions a distributed processing paradigm, which seamlessly spans from the third layer down to the inaugural first layer. What this entails in practical terms is a framework wherein individual IoT devices take on a more proactive role. Instead of merely transmitting raw data, these devices would engage in preliminary data processing. Once this preprocessing phase is concluded, they would then dispatch the condensed and refined data to the third layer. Such an approach holds the promise of

significantly reducing network traffic volume, thereby optimizing data flow and overall system efficienc

APPENDIX I

Consensus algorithms

In this section based on (Guo and Yu 2022; K and S 2023; Leo and Hattingh 2021; Vyas and Deshmukh 2023; Zheng et al. n.d.) we will explain other examples of consensus algorithm which has been developed

- Byzantine Fault Tolerance (BFT)
    - Blockchain is decentralized, posing challenges with inconsistent node information.
    - The "Byzantine generals problem" was addressed, leading to the BFT algorithm.
    - An updated proof of stake or computational power of proof of work can tackle this problem
- Practical Byzantine Fault Tolerance (PBFT)
    - PBFT addresses the Byzantine generals problem in decentralized systems
    - Hyperledger Fabric uses PBFT, handling up to 1/3 malicious replicas
    - The algorithm involves phases like pre-prepared, prepared, and commit, requiring 2/3 node votes
    - Developed by Liskov and Castro, PBFT emphasizes safety in asynchronous networks and uses cryptographic methods for message authenticity
    - PBFT ensures consensus if malicious nodes are less than a third of the total
    - PBFT allows for some malicious nodes, ensuring continuous block additions and requires 2/3 network nodes to act correctly
- Delegated Proof of Stake (DPoS)
    - Stakeholders elect delegates for block validation.
    - Operates on representative democracy.

    - Faster transactions due to fewer validators.
    - Dishonest delegates can be replaced.
    - Underpins Bitshares and was conceived by Dan Larimer for platforms like Bitshares, Steem, and EOS
- Ripple's Consensus Algorithm
    - Nodes divided into servers (for consensus) and clients (for fund transfers).
    - Servers use a Unique Node List (UNL) for querying.
    - Ensures ledger accuracy if faulty nodes in UNL are below 20%
- Tendermint
    - Byzantine consensus algorithm with a proposer broadcasting blocks.
    - Steps include prevote, precommit, and commit.
    - Validators lock coins and face penalties for dishonesty
- Proof of Burn (PoB)
    - Miners show proof of burning coins by sending to an unspendable address.
    - Relies on PoW-mined cryptocurrencies
- Proof of Capacity (PoC)
    - Miners allocate hard drive space for rewards.
    - Burstcoin employs PoC, requiring significant storage
- Delegated Byzantine Fault Tolerance (DBFT)
    - Introduced by NEO; token holders vote for bookkeepers.
    - Bookkeepers use BFT for consensus.
    - Ensures "Absolute finality" in transactions
- Directed Acyclic Graph (DAG)
    - Unique consensus data structure.
    - Transactions are vertices added without mining.
    - Uses PoW for spam prevention
- SIEVE (Hyperledger)
    - Detects/removes non-deterministic requests.
    - Executes operations and compares outputs to remove disparities
- Proof of Elapsed Time (PoET)

    - Developed by Intel.
    - Block mining winners determined by random waiting times on trusted platforms
- Raft
    - Alternative to Paxos protocol.
    - Clusters servers into follower, candidate, or leader states.
    - Leaders elected based on term numbers

LIST OF BIBLIOGRAPHICAL REFERENCES

Aggarwal, Vikash Kumar, Nikhil Sharma, Ila Kaushik, Bharat Bhushan, and Himanshu. 2021. "Integration of Blockchain and IoT (B-IoT): Architecture, Solutions, & Future Research Direction." *IOP Conference Series: Materials Science and Engineering* 1022(1):012103. doi: 10.1088/1757-899X/1022/1/012103.

Akanksha, and Akshay Chaturvedi. 2022. "Comparison of Different Authentication Techniques and Steps to Implement Robust JWT Authentication." Pp. 772–79 in *2022 7th International Conference on Communication and Electronics Systems (ICCES)*.

Al-athwari, Baseem, and Md. Azam Hossain. 2022. "IoT Architecture: Challenges and Open Research Issues." Pp. 408–19 in *Proceedings of 2nd International Conference on Smart Computing and Cyber Security*, *Lecture Notes in Networks and Systems*, edited by P. K. Pattnaik, M. Sain, and A. A. Al-Absi. Singapore: Springer Nature.

Alenizi, Abdulrahman S., and Khamis A. Al-Karawi. 2023. "Internet of Things (IoT) Adoption: Challenges and Barriers." Pp. 217–29 in *Proceedings of Seventh International Congress on Information and Communication Technology*, *Lecture Notes in Networks and Systems*, edited by X.-S. Yang, S. Sherratt, N. Dey, and A. Joshi. Singapore: Springer Nature.

Alshahrani, Norah, M. L, B. Zaidan, A. Alamoodi, and Abdu Saif. 2023. "A Review of Smart Contract Blockchain Based on Multi-Criteria Analysis: Challenges and Motivations." *Computers, Materials & Continua* 75(2):2833–58. doi: 10.32604/cmc.2023.036138.

Al-Zoubi, Abdallah, Tariq Saadeddin, Mamoun Dmour, and Luma Adi. 2022. "An Interactive IoT-Blockchain System for Big Data Management." Pp. 71–76 in *2022 4th IEEE Middle East and North Africa COMMunications Conference (MENACOMM)*. Amman, Jordan: IEEE.

Anon. n.d. "Architecture." *Hyperledger Caliper*. Retrieved September 12, 2023 (https://hyperledger.github.io/caliper/v0.4.2/architecture/).

Anthal, Jyotsna, Shakir Choudhary, and Ravikumar Shettiyar. 2023. "Decentralizing File Sharing: The Potential of Blockchain and IPFS." Pp. 773–77 in *2023 International Conference on Advancement in Computation & Computer Technologies (InCACCT)*.

Astropekakis, Konstantinos, Emmanouil Drakakis, Konstantinos Grammatikakis, and Christos Goumopoulos. 2022. "A Survey of IoT Software Platforms." Pp. 299–326 in *Advances in Computing, Informatics, Networking and Cybersecurity: A Book Honoring Professor Mohammad S. Obaidat's Significant Scientific Contributions*, *Lecture Notes in Networks and Systems*, edited by P. Nicopolitidis, S. Misra, L. T. Yang, B. Zeigler, and Z. Ning. Cham: Springer International Publishing.

Banik, Sejuti, Irvin Steve Cardenas, and Jong Hoon Kim. 2019. “IoT Platforms for 5G Network and Practical Considerations: A Survey.”

Belej, Olexander, Kamil Staniec, and Tadeusz Więckowski. 2020. “The Need to Use a Hash Function to Build a Crypto Algorithm for Blockchain.” Pp. 51–60 in *Theory and Applications of Dependable Computer Systems*, *Advances in Intelligent Systems and Computing*, edited by W. Zamojski, J. Mazurkiewicz, J. Sugier, T. Walkowiak, and J. Kacprzyk. Cham: Springer International Publishing.

Beltran, Joel Alanya, Pankaj Mudholkar, Megha Mudholkar, Vikas Tripathi, Carlos Valderrama-Zapata, and Melanie Lourense. 2022. “Security Issues and Challenges in Internet of Things (IoT) System.” Pp. 57–60 in *2022 5th International Conference on Contemporary Computing and Informatics (IC3I)*.

Bhushan, Bharat, Preeti Sinha, K. Martin Sagayam, and Andrew J. 2021. “Untangling Blockchain Technology: A Survey on State of the Art, Security Threats, Privacy Services, Applications and Future Research Directions.” *Computers & Electrical Engineering* 90:106897. doi: 10.1016/j.compeleceng.2020.106897.

Capocasale, Vittorio, Danilo Gotta, and Guido Perboli. 2023. “Comparative Analysis of Permissioned Blockchain Frameworks for Industrial Applications.” *Blockchain: Research and Applications* 4(1):100113. doi: 10.1016/j.bcra.2022.100113.

Choi, Wonseok, and James Won-Ki Hong. 2021. “Performance Evaluation of Ethereum Private and Testnet Networks Using Hyperledger Caliper.” Pp. 325–29 in *2021 22nd Asia-Pacific Network Operations and Management Symposium (APNOMS)*.

Confluent Documentation, Apache Kafka. n.d. “Brokers in Control Center | Confluent Documentation.” Retrieved September 7, 2023 (https://docs.confluent.io/platform/current/control-center/brokers.html).

Connors, Collin, and Dilip Sarkar. 2023. “Survey of Prominent Blockchain Development Platforms.” *Journal of Network and Computer Applications* 216:103650. doi: 10.1016/j.jnca.2023.103650.

Costin, Andrei, and Aurelien Francillon. n.d. “Ghost in the Air(Traffic): On Insecurity of ADS-B Protocol and Practical Attacks on ADS-B Devices.” 9.

Dange, Smita, and Prashant Nitnaware. 2023. “Secure Share: Optimal Blokchain Integration in IoT Systems.” *Journal of Computer Information Systems* 1–13. doi: 10.1080/08874417.2023.2193943.

Dehez Clementi, Marina, Nicolas Larrieu, Emmanuel Lochin, Mohamed Ali Kaafar, and Hassan Asghar. 2019. “When Air Traffic Management Meets Blockchain Technology: A Blockchain-Based Concept for Securing the Sharing of Flight Data.” Pp. 1–10 in

2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC). San Diego, CA, USA: IEEE.

Deohate, Ankita, and Dinesh Rojatkar. 2021. "Middleware Challenges and Platform for IoT-A Survey." Pp. 463–67 in *2021 5th International Conference on Trends in Electronics and Informatics (ICOEI)*.

Elliston, Janei, Hongmei Chi, Shonda Bernadin, and Maryam Taeb. 2023. "Integrating Blockchain Technology into Cybersecurity Education." Pp. 1–15 in *Proceedings of the Future Technologies Conference (FTC) 2022, Volume 2*, *Lecture Notes in Networks and Systems*, edited by K. Arai. Cham: Springer International Publishing.

Faridul Islam Suny, Md., Md. Monjourur Roshed Fahim, Mushfiqur Rahman, Nishat Tasnim Newaz, and Tajim Md. Niamat Ullah Akhund. 2021. "IoT Past, Present, and Future a Literary Survey." Pp. 393–402 in *Information and Communication Technology for Competitive Strategies (ICTCS 2020)*, *Lecture Notes in Networks and Systems*, edited by M. S. Kaiser, J. Xie, and V. S. Rathore. Singapore: Springer Nature.

Fong, Desmond Kong Ze, Vinesha Selvarajah, and M. S. Nabi. 2022. "Secure Server Storage Based IPFS through Multi-Authentication." Pp. 1–7 in *2022 International Conference on Advancements in Smart, Secure and Intelligent Computing (ASSIC)*.

Garg, Chirag, Deepak Kumar Mishra, Dikshant Raj, and Pawan Singh Mehra. 2022. "A Survey on Integration of Blockchain and IoT (BIoT): Open Issues, Challenges & Solution." Pp. 880–86 in *2022 Third International Conference on Intelligent Computing Instrumentation and Control Technologies (ICICICT)*.

Guo, Huaqun, and Xingjie Yu. 2022. "A Survey on Blockchain Technology and Its Security." *Blockchain: Research and Applications* 3(2):100067. doi: 10.1016/j.bcra.2022.100067.

IPFS Docs, How IPFS works. n.d. "How IPFS Works | IPFS Docs." Retrieved August 23, 2023 (https://docs.ipfs.tech/concepts/how-ipfs-works/#subsystems-overview).

K, Harshini Poojaa, and Ganesh Kumar S. 2023. "Evolution of Consensus Algorithms in Blockchain Technology." Pp. 493–98 in *2023 International Conference on Intelligent Systems for Communication, IoT and Security (ICISCoIS)*.

Kenaza, Rabah, Ameur Khemane, Hakim Bendjenna, Abdallah Meraoumia, and Lakhdar Laimeche. 2022. "Internet of Things (IoT): Architecture, Applications, and Security Challenges." Pp. 1–5 in *2022 4th International Conference on Pattern Analysis and Intelligent Systems (PAIS)*.

Khare, Shanu, Azher Ashraf, Mir Mohammad Yousuf, and Mamoon Rashid. 2022. "Blockchain: Structure, Uses, and Applications in IoT." Pp. 131–44 in *Blockchain Security in Cloud Computing*, *EAI/Springer Innovations in Communication and*

Computing, edited by K. M. Baalamurugan, S. R. Kumar, A. Kumar, V. Kumar, and S. Padmanaban. Cham: Springer International Publishing.

Kim, Heesang, and Dohoon Kim. 2023. "A Taxonomic Hierarchy of Blockchain Consensus Algorithms: An Evolutionary Phylogeny Approach." *Sensors* 23(5):2739. doi: 10.3390/s23052739.

Kumar N.V., Rajeesh, and Mohan Kumar P. 2020. "Survey on State of Art IoT Protocols and Applications." Pp. 1–3 in *2020 International Conference on Computational Intelligence for Smart Power System and Sustainable Energy (CISPSSE)*.

Leo, J., and M. J. Hattingh. 2021. "Key Characteristics to Create Optimized Blockchain Consensus Algorithms." Pp. 567–79 in *Responsible AI and Analytics for an Ethical and Inclusive Digitized Society*, *Lecture Notes in Computer Science*, edited by D. Dennehy, A. Griva, N. Pouloudi, Y. K. Dwivedi, I. Pappas, and M. Mäntymäki. Cham: Springer International Publishing.

M R, Prathyusha, and Biswajit Bhowmik. 2023. "IoT Evolution and Recent Advancements." Pp. 1725–30 in *2023 9th International Conference on Advanced Computing and Communication Systems (ICACCS)*. Vol. 1.

Maeng, Juhyun, Yoonnyoung Heo, and Inwhee Joe. 2022. "Hyperledger Fabric-Based Lightweight Group Management (H-LGM) for IoT Devices." *IEEE Access* 10:56401–9. doi: 10.1109/ACCESS.2022.3177270.

Mao, Tian, and Junhua Chen. 2023. "Smart Contract in Blockchain." Pp. 868–75 in *Proceedings of the 2022 International Conference on Bigdata Blockchain and Economy Management (ICBBEM 2022)*. Vol. 5, *Atlantis Highlights in Intelligent Systems*, edited by D. Qiu, Y. Jiao, and W. Yeoh. Dordrecht: Atlantis Press International BV.

Miraz, Mahdi H., and Maaruf Ali. 2018. "Blockchain Enabled Enhanced IoT Ecosystem Security." Pp. 38–46 in *Emerging Technologies in Computing*, edited by M. H. Miraz, P. Excell, A. Ware, S. Soomro, and M. Ali. Cham: Springer International Publishing.

Nartey, Clement, Eric Tutu Tchao, James Dzisi Gadze, Eliel Keelson, Griffith Selorm Klogo, Benjamin Kommey, and Kwasi Diawuo. 2021. "On Blockchain and IoT Integration Platforms: Current Implementation Challenges and Future Perspectives." *Wireless Communications and Mobile Computing* 2021:e6672482. doi: 10.1155/2021/6672482.

Nayancy, Sandip Dutta, and Soubhik Chakraborty. 2021. "IoT-Based Secure Communication to Enhance Blockchain Model." Pp. 255–64 in *Proceedings of the Fourth International Conference on Microelectronics, Computing and Communication Systems*, *Lecture Notes in Electrical Engineering*, edited by V. Nath and J. K. Mandal. Singapore: Springer.

Obaidat, Muath A., Suhaib Obeidat, Jennifer Holst, Abdullah Al Hayajneh, and Joseph Brown. 2020. “A Comprehensive and Systematic Survey on the Internet of Things: Security and Privacy Challenges, Security Frameworks, Enabling Technologies, Threats, Vulnerabilities and Countermeasures.” *Computers* 9(2):44. doi: 10.3390/computers9020044.

Oikonomou, Filippos Pelekoudas, Jose Ribeiro, Georgios Mantas, Joaquim Manuel C. S. Bastos, and Jonathan Rodriguez. 2021. “A Hyperledger Fabric-Based Blockchain Architecture to Secure IoT-Based Health Monitoring Systems.” Pp. 186–90 in *2021 IEEE International Mediterranean Conference on Communications and Networking (MeditCom)*. Athens, Greece: IEEE.

Palma, Stefano Dalla, Remo Pareschi, and Federico Zappone. 2021. “What Is Your Distributed (Hyper)Ledger?” Pp. 27–33 in *2021 IEEE/ACM 4th International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB)*.

Patnaik, Ranjit, Neelamadhab Padhy, and K. Srujan Raju. 2021. “A Systematic Survey on IoT Security Issues, Vulnerability and Open Challenges.” Pp. 723–30 in *Intelligent System Design*, *Advances in Intelligent Systems and Computing*, edited by S. C. Satapathy, V. Bhateja, B. Janakiramaiah, and Y.-W. Chen. Singapore: Springer.

Peddireddy, Kiran. 2023. “Streamlining Enterprise Data Processing, Reporting and Realtime Alerting Using Apache Kafka.” Pp. 1–4 in *2023 11th International Symposium on Digital Forensics and Security (ISDFS)*.

Pise, Rohini, and Sonali Patil. 2022. “A Deep Dive into Blockchain-Based Smart Contract-Specific Security Vulnerabilities.” Pp. 1–6 in *2022 IEEE International Conference on Blockchain and Distributed Systems Security (ICBDS)*.

Punathumkandi, Swathi, Venkatesan Meenakshi Sundaram, and P. Prabhavathy. 2020. “A Deep Dive into Hyperledger.” Pp. 85–107 in.

Rahman, Muhammad Saifur, and Rohit Kumar Das. 2022. “RTID: On-Demand Real-Time Data Processing for IoT Network.” *Materials Today: Proceedings* 62:4721–25. doi: 10.1016/j.matpr.2022.03.168.

Rai, Mritunjay Kumar, Rajeev Kanday, and Reji Thomas. 2020. “Current Issues and Challenges of Security in IoT Based Applications.” Pp. 578–89 in *Advances in Intelligent Systems and Interactive Applications*, *Advances in Intelligent Systems and Computing*, edited by F. Xhafa, S. Patnaik, and M. Tavana. Cham: Springer International Publishing.

Rani, Anjana, and Monika Saxena. 2023. “A Review Survey of the Algorithms Used for the Blockchain Technology.” Pp. 655–68 in *Soft Computing for Problem Solving*, *Lecture Notes in Networks and Systems*, edited by M. Thakur, S. Agnihotri, B. S. Rajpurohit, M. Pant, K. Deep, and A. K. Nagar. Singapore: Springer Nature.

Raptis, Theofanis P., and Andrea Passarella. 2022. “On Efficiently Partitioning a Topic in Apache Kafka.” Pp. 1–8 in *2022 International Conference on Computer, Information and Telecommunication Systems (CITS)*.

Rosa, Giovanni, Simone Scalabrino, and Rocco Oliveto. 2022. “Assessing and Improving the Quality of Docker Artifacts.” Pp. 592–96 in *2022 IEEE International Conference on Software Maintenance and Evolution (ICSME)*.

Sadawi, Alia Al, Mohamed S. Hassan, and Malick Ndiaye. 2021. “A Survey on the Integration of Blockchain With IoT to Enhance Performance and Eliminate Challenges.” *IEEE Access* 9:54478–97. doi: 10.1109/ACCESS.2021.3070555.

Sadeghi-Nasab, Alireza, and Vahid Rafe. 2023. “A Comprehensive Review of the Security Flaws of Hashing Algorithms.” *Journal of Computer Virology and Hacking Techniques* 19(2):287–302. doi: 10.1007/s11416-022-00447-w.

Sarmah, Simanta Shekhar. 2018. “Understanding Blockchain Technology.” *Computer Science and Engineering 2018, 8(2): 23-29*. doi: 10.5923/j.computer.20180802.02.

Sathi Reddy, Padala, Nukella Venkatesh, and S. Kumar. 2022. “Anatomization: IoT Security Issues and Its Future Challenges.” in *Advances in Transdisciplinary Engineering*, edited by R. M. Singari and P. K. Kankar. IOS Press.

Sharma, Arunima, and Ramesh Babu Battula. 2022. “Challenges and Issues in Blockchain-Based IoT Services.” Pp. 47–80 in *Blockchain based Internet of Things*, *Lecture Notes on Data Engineering and Communications Technologies*, edited by D. De, S. Bhattacharyya, and J. J. P. C. Rodrigues. Singapore: Springer.

Singh, Chaitanya, and Deepika Chauhan. 2021. “IoT–Blockchain Integration-Based Applications Challenges and Opportunities.” Pp. 87–116 in *Mobile Radio Communications and 5G Networks*, *Lecture Notes in Networks and Systems*, edited by N. Marriwala, C. C. Tripathi, D. Kumar, and S. Jain. Singapore: Springer.

Srinivasa, Shreyas, Jens Myrup Pedersen, and Emmanouil Vasilomanolakis. 2022. “Deceptive Directories and ‘Vulnerable’ Logs: A Honeypot Study of the LDAP and Log4j Attack Landscape.” Pp. 442–47 in *2022 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*.

Srivastava, Nipun, and Pallavi Pandey. 2022. “Internet of Things (IoT): Applications, Trends, Issues and Challenges.” *Materials Today: Proceedings* 69:587–91. doi: 10.1016/j.matpr.2022.09.490.

Su Wai, Khin Su, Ei Chaw Htoon, and Nwe Nwe Myint Thein. 2020. “Performance Evaluation of M/D/1 Queuing Model on Hyperledger Fabric.” Pp. 36–41 in *2020 International Conference on Advanced Information Technologies (ICAIT)*.

Su, Yue, Kien Nguyen, and Hiroo Sekiya. 2022. “Latency Evaluation in Ad-Hoc IoT-Blockchain Network.” Pp. 124–28 in *2022 5th World Symposium on Communication Engineering (WSCE)*. Nagoya, Japan: IEEE.

Torres, Agmar A., and Flávio de Oliveira Silva. 2023. “SANKMO: An Approach for Ingestion, Processing, Storing, and Sharing IoT Data in Near Real-Time.” Pp. 279–91 in *Advanced Information Networking and Applications*, *Lecture Notes in Networks and Systems*, edited by L. Barolli. Cham: Springer International Publishing.

Ullah, M., and K. Smolander. 2019. “Highlighting the Key Factors of an IoT Platform.” Pp. 901–6 in *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*.

Verma, Garima, and Shiva Prakash. 2021. “Emerging Security Threats, Countermeasures, Issues, and Future Aspects on the Internet of Things (IoT): A Systematic Literature Review.” Pp. 59–66 in *Advances in Interdisciplinary Engineering*, *Lecture Notes in Mechanical Engineering*, edited by N. Kumar, S. Tibor, R. Sindhwani, J. Lee, and P. Srivastava. Singapore: Springer.

Vyas, Komal, and Ashwini Deshmukh. 2023. “A Survey Paper on Blockchain Technology and Consensus Algorithms.” Pp. 1–6 in *2023 11th International Conference on Emerging Trends in Engineering & Technology - Signal and Information Processing (ICETET - SIP)*.

Vyas, Shubham, Rajesh Kumar Tyagi, Charu Jain, and Shashank Sahu. 2022. “Performance Evaluation of Apache Kafka – A Modern Platform for Real Time Data Streaming.” Pp. 465–70 in *2022 2nd International Conference on Innovative Practices in Technology and Management (ICIPTM)*. Vol. 2.

Wu, Canghai, Jie Xiong, Huanliang Xiong, Yingding Zhao, and Wenlong Yi. 2022. “A Review on Recent Progress of Smart Contract in Blockchain.” *IEEE Access* 10:50839–63. doi: 10.1109/ACCESS.2022.3174052.

Wu, Zhijun, Tong Shang, and Anxin Guo. 2020. “Security Issues in Automatic Dependent Surveillance - Broadcast (ADS-B): A Survey.” *IEEE Access* 8:122147–67. doi: 10.1109/ACCESS.2020.3007182.

Xu, Xiaoqiong, Xiaonan Wang, Zonghang Li, Hongfang Yu, Gang Sun, Sabita Maharjan, and Yan Zhang. 2021. “Mitigating Conflicting Transactions in Hyperledger Fabric-Permissioned Blockchain for Delay-Sensitive IoT Applications.” *IEEE Internet of Things Journal* 8(13):10596–607. doi: 10.1109/JIOT.2021.3050244.

Yu, Jin-Yong, and Young-Gab Kim. 2019. “Analysis of IoT Platform Security: A Survey.” Pp. 1–5 in *2019 International Conference on Platform Technology and Service (PlatCon)*.

Zambre, Pranav, Mitali Panchal, and Ankit Chauhan. 2023. “A Future to the Blockchain Technology and Its Concepts.” Pp. 111–22 in *ICT with Intelligent Applications*, *Smart Innovation, Systems and Technologies*, edited by J. Choudrie, P. Mahalle, T. Perumal, and A. Joshi. Singapore: Springer Nature.

Zheng, Zibin, Shaoan Xie, Hong-Ning Dai, Xiangping Chen, and Huaimin Wang. n.d. “Blockchain Challenges and Opportunities: A Survey.”

Zhonghua, Chen, and S. B. Goyal. 2023. “Blockchain-Based Framework to Handle Security and Privacy for IoT System.” Pp. 71–82 in *Proceedings of Third Doctoral Symposium on Computational Intelligence*, *Lecture Notes in Networks and Systems*, edited by A. Khanna, D. Gupta, V. Kansal, G. Fortino, and A. E. Hassanien. Singapore: Springer Nature.