ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DOTTORATO DI RICERCA IN

INGEGNERIA ELETTRONICA, TELECOMUNICAZIONI E TECNOLOGIE DELL'INFORMAZIONE

Ciclo 37

Settore Concorsuale: 09/H1 - SISTEMI DI ELABORAZIONE DELLE INFORMAZIONI

Settore Scientifico Disciplinare: ING-INF/05 - SISTEMI DI ELABORAZIONE DELLE INFORMAZIONI

SMART CONTRACTS FOR CERTIFIED OPERATIONS IN INDUSTRIAL DISTRIBUTED APPLICATIONS

Presentata da: Nicola Elia

Coordinatore Dottorato

Davide Dardari

Supervisore

Andrea Acquaviva

Esame finale anno 2025

Abstract

The integration of smart contracts and blockchain technology is gaining momentum in industrial applications, especially in safety-critical systems such as predictive maintenance and real-time anomaly detection. These systems typically rely on sensor networks to collect data and stream it to cloud infrastructures for storage, analysis, and visualization. However, industrial applications pose unique challenges: they require sustainable storage solutions for long-term continuous monitoring, while ensuring data integrity and tamper-proof operations. At the same time, the performance of smart contracts in production-grade environments becomes critical, as industrial systems depend on them to handle high transaction volumes and maintain scalability under real-world conditions.

The first section of this thesis presents a blockchain-based framework enabling certified data removal for continuous monitoring systems. Smart contracts define and run data retention policies, enabling the secure deletion of non-compliant data. This solution is applied to a real-world Structural Health Monitoring (SHM) use case, where the blockchain guarantees the tamper-proof deletion of data in a railway bridge monitoring application.

In the second part, the focus shifts to the benchmarking of smart contracts in production-grade deployments. A step-by-step methodology is proposed for simulating real-world environments and evaluating smart contracts through key performance indicators such as Average Transaction Latency (ATL) and Average Transaction Throughput (ATT). The final aim is enabling organizations to make data-driven decisions regarding the introduction of smart contracts in industrial applications, evaluating various network constraints and blockchain configurations.

Finally, these two contributions merge into an experimental study examining a production-grade deployment of the proposed smart contract using the discussed methodology. A framework for benchmarking is implemented to assist users in evaluating smart contracts automatically. The results show how the methodology enables organizations to make informed decisions regarding smart contract deployment and scalability through quantification of network limitations and blockchain configuration effects.

Acknowledgements

The successful completion of this doctoral thesis was made possible through the support and encouragement of many individuals who have shaped my academic journey. Thereby, I would like to express my sincere gratitude to each of them.

First, I want to thank my academic supervisor, Professor *Andrea Acquaviva*, who first have me faith, then helped me find my research path and guided me in reaching my goals, up to writing my first papers and this thesis. Beyond his academic guidance, he showed remarkable kindness and was always ready to help, supporting me to grow both as a person and as a researcher.

My work has been made better by working with many researchers and PhD students within the ECS Laboratory. I especially want to thank *Francesco Barchi*, who helped me solve many technical problems during my PhD studies with great empathy, making him not just an excellent guide but also a wonderful person to work with.

I am very grateful to *Livio Pompianu* from the University of Cagliari, who played a crucial role particularly in the early stages of my research. His deep expertise in the field helped me understand the fundamental concepts that would shape my entire thesis work. Through our many discussions, he helped me identify promising research directions and understand their potential impact. I also want to thank my colleague *Alessia Pisu*, that involved me in interesting research projects that led to very interesting discussions and valuable research results.

I acknowledge support from *TIM S.p.A.* through the PhD scholarship. I want to express special thanks to *Danilo Gotta*, whose industry experience and insights greatly enriched my research. His guidance helped bridge the gap between academic research and real-world applications, leading to a successful patent application.

My deepest gratitude goes to my family, who supported me through the most challenging moments of this journey. To my sister *Maria Giovanna*, who has been a constant source of encouragement, and to my parents, *Piero* and *Luisa*, who supported my choices and helped me overcome every obstacle: your faith in me has made all the difference. A special thank you to my girlfriend *Debora*, whose patience, understanding, and support helped me stay focused and motivated even during the most difficult times. Your presence has made this challenging path much brighter.

The support of all these people has been fundamental for my growth as a person, researcher, and professional. Thanks to their guidance and teachings, my doctoral journey has been a truly transformative experience, and I could conclude this journey with valuable skills that will serve me well both in academia and industry.

Contents

1 Introduction

The integration of advanced technologies in industrial systems is transforming the way industries handle data-intensive and safety-critical applications. Among these technologies, **blockchain** has gained significant attention due to its potential to ensure data integrity, security, and certification in decentralized environments. By encapsulating business logic through **smart contracts**, blockchain offers a new paradigm for conducting certified operations in a tamper-proof and verifiable manner. This research explores the application of blockchain, and specifically smart contracts, to address key challenges in industrial distributed applications, with a focus on ensuring certified operations and evaluating their performance.

One of the primary motivations behind the adoption of blockchain in safety-critical monitoring systems is the growing need for trusted, decentralized systems capable of ensuring data integrity and compliance with regulatory standards. Safety-critical systems, such as **Structural Health Monitoring (SHM)**, industrial IoT networks, and energy distribution systems, rely on continuous data collection and analysis over extended periods. These applications demand high levels of data accuracy and transparency, as any tampering or data loss could have catastrophic consequences.

However, the long-term operation of such monitoring systems presents several challenges, particularly related to data storage and the sustainability of cloud databases. Over time, the accumulation of vast amounts of data from sensors leads to boundless growth in database size, necessitating certified data-removal policies that allow for efficient storage management without compromising the integrity of the data. Traditional methods of reducing data on the cloud pose significant risks, especially when dealing with safety-critical data, as malicious dele-

tions could go undetected, undermining the trust in the system.

To address these challenges, the Chapter 3 of this thesis proposes an innovative solution that leverages blockchain and smart contracts to implement **certified data-removal policies**, improving the sustainability of cloud-based monitoring applications while providing anti-tampering guarantees. By integrating smart contracts into the data management process, we ensure that data-removal algorithms are executed in a verifiable and tamper-proof manner, thereby preserving the trustworthiness of the monitoring infrastructure.

The initial experiments were conducted under **ideal conditions**, allowing us to establish that the overall system functions correctly and effectively addresses the use-case problem of sustainable and certified data removal for long-term data storage. However, to fully validate the system's applicability in real-world conditions, it is essential to assess the performance of the smart contracts in **production-grade deployments**. Indeed, while blockchain offers a promising solution for ensuring data integrity, its application to industrial systems requires careful evaluation of smart contract performance under realistic operational environments. The performance of smart contracts, particularly in terms of latency, throughput, and scalability, directly impacts the efficiency and feasibility of blockchain-based industrial systems.

Moreover, as the complexity and demand for blockchain applications grow in industries such as healthcare, smart grids, logistics, and IoT, understanding the performance limitations of smart contracts becomes essential. In this context, the Chapter 4 of this thesis introduces a **methodology** for assessing the performance of smart contracts deployed in industrial distributed applications under production-grade conditions. The proposed methodology focuses on Key Performance Indicators (KPIs) such as Average Transaction Latency (ATL) and Average Transaction Throughput (ATT), which provide a user-centric view of how effectively smart contracts

handle requests from other components within the industrial system. By treating the underlying blockchain as a black box, this evaluation framework allows for a more standardized and independent assessment of smart contract performance, regardless of the specific blockchain technology or use case scenario.

We comprehensively evaluate the proposed smart contract for certified data-removal by applying our methodology as discussed in Chapter 5. In particular, we evaluate the smart contract's performance in a production-grade environment using the methodology discussed in Chapter 4. This allows us to move beyond ideal conditions and assess the real-world applicability and scalability of our system in the real-world SHM use case.

Overall, the research work presented in this thesis contributes to the field by offering both a certified data-removal platform for safety-critical monitoring applications and a performance evaluation framework for smart contracts in industrial systems, aiming to foster the adoption of blockchain in industries that require verifiable, efficient, and sustainable distributed systems.

1.1 Motivations

The primary motivation behind this research stems from the increasing demand for certified, tamper-proof data management in industrial systems, particularly those involved in continuous monitoring and safety-critical systems. The need for data and operations integrity in long-term monitoring systems is critical for industries dealing with safety-critical infrastructures, where data tampering or loss could have severe consequences. Blockchain technology, with its decentralized and immutable nature, provides a viable solution for addressing these concerns.

Importantly, as blockchain adoption grows, the performance of smart contracts under real-world conditions

becomes a major concern for industries seeking to deploy these systems at scale. Assessing the performance of smart contracts in production-grade deployments is essential to ensure their feasibility and scalability in real-world industrial scenarios.

A detailed discussion about the motivations behind the presented research work is provided within each section of the following thesis.

1.2 Contributions

This thesis makes three primary contributions:

1. **Certified Data-Removal Policies for Safety-Critical Monitoring** - A novel architecture that implements certified data-removal policies using blockchain and smart contracts. This solution is specifically designed for long-term monitoring systems, such as SHM, where database sustainability and data integrity are critical. By leveraging blockchain, we ensure that data removal is carried out in a tamper-proof and certified manner, addressing the issue of storage space sustainability.
2. **Methodology for Performance Evaluation of Smart Contracts for Industrial Applications** - We propose a four-phase methodology that provides guidelines for measuring smart contract performance using standard literature metrics. The methodology helps to assess whether a production-grade deployment can meet the performance requirements of real-world industrial applications. It also enables experimental comparisons of different configurations (e.g., network characteristics, node count) to identify the most suitable environment for contract deployment.
3. **Reference Implementation on a Real-World Case-Study** - We illustrate the practical application of the presented methodology through the real-world SHM case study discussed as the first contribution. The reference implementation provides a concrete

example of how to assess smart contract performance in industrial contexts, particularly using Hyperledger Fabric. Through this implementation, we measure key performance metrics such as transaction throughput and latency, we estimate the number of concurrent industrial applications that can be served by a single smart contract, and we demonstrate how to fine-tune the system parameters to improve system performance.

1.3 Outline

The remainder of this thesis is organized as follows:

- Chapter 2 reviews the relevant background on blockchain technology, smart contracts, and industrial applications with focus on SHM.
- Chapter 3 presents the design and architecture of the proposed platform for certified operations in industrial monitoring systems.
- Chapter 4 introduces the performance evaluation methodology for smart contracts for real-world industrial scenarios.
- Chapter 5 applies the proposed methodology to the real-world SHM system presented in the first part of the thesis.
- Chapter 7 concludes the thesis by summarizing the key contributions and proposing avenues for future research.

Background 2

The application of blockchain technology to industrial systems, particularly for safety-critical and data-intensive applications, presents a promising opportunity to improve data integrity and certified operations. However, challenges related to storage sustainability, performance, and scalability need to be addressed to facilitate blockchain adoption in real-world industrial scenarios. This chapter provides an overview of the key concepts, technologies, and previous works that form the foundation of this research. The two main areas of focus are structural health monitoring (SHM) and the evaluation of smart contract performance in industrial applications.

2.1 Industrial Applications

Industrial applications refer to the practical uses of technologies, processes, systems, and innovations in various sectors to improve efficiency, productivity, and safety. These applications span a wide range of industries, including manufacturing, energy, construction, transportation, healthcare, and more. By integrating advanced technologies, industries can streamline operations, reduce costs, enhance product quality, and adapt to evolving market demands.

In various industrial sectors, the complexity and scale of operations have led to a reliance on advanced technologies and data-driven decision-making under the Industry 4.0 paradigm [1, 2].

In this context, the need for tamperproof data and operations is particularly critical in industries where safety, efficiency, and compliance are paramount. Many industries have indeed strict safety and maintenance standards that rely on accurate, untampered data for compliance.

Tampering could result in penalties, legal liabilities, or worse, loss of life.

A fitting example is Structural Health Monitoring (SHM), where structures such as public roads or bridges are monitored to prevent damages or accidents, thus involving both public authorities and private stakeholders. For large infrastructure projects such as bridges, tunnels, and high-rise buildings, SHM systems continuously gather data to assess structural integrity. In these contexts, tamper-proofing data is critical to ensure public safety, prevent catastrophic failures, and support maintenance decisions.

Importantly, all the involved stakeholders should always be able of transparently sharing information among themselves, with data access permissions managed transparently and securely. For example, consumers in a food supply chain should have the ability to access information such as food certificates (e.g., origin) whenever they choose.

2.2 Structural Health Monitoring (SHM)

Structural Health Monitoring (SHM) has become an increasingly critical field as aging infrastructures, such as bridges, buildings, and industrial plants, require frequent inspections to ensure their safety and integrity. With the costs of traditional manual inspections rising, SHM offers a more efficient solution through the continuous assessment of a structure's condition.

By leveraging SHM systems, companies and governments can continuously assess the health and integrity of infrastructures in real-time, enabling predictive maintenance and improving overall security and longevity [3–5].

Typical SHM systems have a layered architecture[6], consisting of:

- IoT sensing nodes that perform on-field continuous monitoring;
- A local network layer that orchestrates data collection;
- Cloud components for data storage, visualization, and analysis.

The primary purpose of these components is to detect structural changes or anomalies early, allowing domain experts to analyze data and identify maintenance needs before severe damage occurs.

One of the core technologies used in recent SHM systems are MEMS (Micro-Electro-Mechanical Systems) capacitive accelerometers, which are employed to measure the vibrations of the structure. In recent decades, these sensors have gained popularity due to their low cost, low power consumption, and scalability, making them ideal for long-term continuous monitoring applications. MEMS-based SHM systems can scale to hundreds of measurement points for a single building, significantly enhancing the monitoring coverage [7–9].

In such applications, high sensor sampling rates are often required to ensure accurate data for analysis, which leads to large data throughput and creates significant challenges for long-term data storage. Indeed, a major challenge often faced by SHM system is the management of the large volumes of data generated by their extensive sensor networks. To mitigate this, SHM systems can implement sparse sensor deployments, where fewer sensors are used to reduce the data stream and storage requirements [5]. Unfortunately, sparse monitoring can limit the system's ability to detect damage in complex and large structures—such as bridges—which require a more granular level of monitoring to effectively track potential issues [10].

2.3 Blockchain Technology and Smart Contracts

A blockchain is a decentralized, append-only Distributed Ledger Technology (DLT) maintained by a network of mutually distrusting nodes using a consensus protocol [11].

Originally introduced with Bitcoin in 2008 with the scope of facilitating the decentralized digital currency, the blockchain technology has evolved beyond cryptocurrency transactions to include a wide range of use cases, such as document notarization, supply chain management, and healthcare [12, 13].

The base idea of blockchain technology is to store an immutable ledger of transactions, that in the case of Bitcoin are currency transfers. All the transactions are stored in containers called blocks, which are chained together using their hash values. As a result, this chain of information, similar to a linked list, is immutable.

The mechanism that allows transactions to be endorsed relies on a distributed consensus algorithm. The whole consensus mechanism will ensure that the endorsed transaction only will be part of the blockchain, discarding all the blocks that have been tampered with.

The development of blockchain technologies like Ethereum introduced the concept of smart contracts, which are decentralized software programs that execute predefined actions when certain conditions are met [14, 15]. The same distributed entities that maintain the distributed network, also called blockchain nodes, are in charge of providing a distributed computing platform, often referred to as distributed virtual machine, that is capable of executing the functions coded in smart contracts as they were transactions on the blockchain. The execution of smart contracts inherits the properties of blockchain transactions, thus being tamper-proof.

Blockchains can be classified based on the permissions of the underlying ledger:

- Permissionless ledgers can be participated by everyone anonymously. Blockchains with permissionless ledgers are called public. Anyone can join the network, query the past transactions and request new transactions to be approved. Examples: Bitcoin, Ethereum.
- Permissioned ledgers can be participated by authorized members only. Blockchains with permissioned ledgers are called private or consortium. Only identified and pre-approved members can join these networks to query or make transactions. Examples: Hyperledger Fabric, Enterprise Ethereum.

These types of blockchain technologies may also be joined to create more complex systems in which consortium blockchains take profit of a public blockchain's distributed ledger to safely store transactions while keeping their content private.

In permissionless systems, any entity can join the network, necessitating consensus mechanisms like Proof of Work (PoW) that require significant computational resources [16]. In contrast, permissioned blockchains (e.g., Hyperledger Fabric [17]) limit network access to identified participants, allowing for the use of more efficient consensus protocols such as Byzantine Fault Tolerance (BFT) [18].

2.4 Hyperledger Fabric

Hyperledger Fabric is a permissioned blockchain platform designed for enterprise use, and it is one of the key projects under the Linux Foundation's Hyperledger[1] umbrella.

Unlike public blockchains, such as Bitcoin or Ethereum, Hyperledger Fabric is private, meaning that only authorized participants can join the network, making it an ideal choice for business environments where privacy, trust, and efficiency are paramount.

1: Hyperledger (or the Hyperledger Project) is an umbrella project of open source blockchains and related tools that the Linux Foundation started in December 2015.

Its permissioned nature is facilitated through a trusted *Membership Service Provider* (MSP), which controls the enrollment and identity management of participants. This contrasts with open permissionless systems, which require resource-intensive consensus mechanisms like Proof of Work (PoW) to validate transactions and maintain network security.

A notable feature is Fabric's ability to create *channels*, enabling subsets of network participants to maintain private ledgers for sensitive transactions, thus providing a higher level of privacy and confidentiality. For instance, in a business-to-business scenario, companies can perform transactions with specific participants without exposing sensitive information to the entire network.

The distributed ledger in Hyperledger Fabric consists of two primary components: the *world state* and the *transaction log*.
The *world state* represents the current state of the ledger at any given moment, functioning like a database where the values of the ledger are stored.
The *transaction log*, on the other hand, records all changes that led to the current world state, effectively serving as an immutable history of all transactions.
By default, the world state is stored using LevelDB, a key-value store database, but it can be configured to use other databases such as CouchDB.

Hyperledger Fabric supports smart contracts, referred to as *chaincode*. Chaincode can be written in several programming languages, including Go and Java. Typically, *chaincode* interacts primarily with the *world state*, querying and updating it based on the transactions initiated by participants. Unlike other blockchain platforms, *chaincode* in Fabric doesn't interact directly with the transaction log, focusing instead on the current state of the ledger.

Consensus in Hyperledger Fabric is modular, enabling the selection of different mechanisms to suit the needs of the network. In particular, Fabric allows network administrators to choose consensus protocols such as RAFT,

Kafka, or PBFT (Practical Byzantine Fault Tolerance). These protocols help to ensure that transactions are ordered and validated efficiently, while also addressing fault tolerance and ensuring that the network remains reliable even in the event of failures.

The architecture of Hyperledger Fabric is based on three key roles within the network: *clients*, *peers*, and *orderers* (grouped in *ordering services*). *Clients* are responsible for submitting transaction proposals to the network, which are then endorsed by *peers*. *Peers* simulate transactions, generating a read-write set, and return their endorsement to the *client*. Once the *client* has gathered enough endorsements, it sends the transaction to the *ordering service*, which sequences transactions and batches them into blocks. These blocks are then distributed across the network for validation and commitment to the ledger.

This transaction approval and ordering model is called *execute-order-validate*, and it is well explained in literature [17, 19]. In brief, it separates transaction execution from ordering and validation, which enhances performance, i.e. higher transactions throughput and lower transaction latency. A sequence diagram is provided in Figure 2.1 along with an explicative caption.

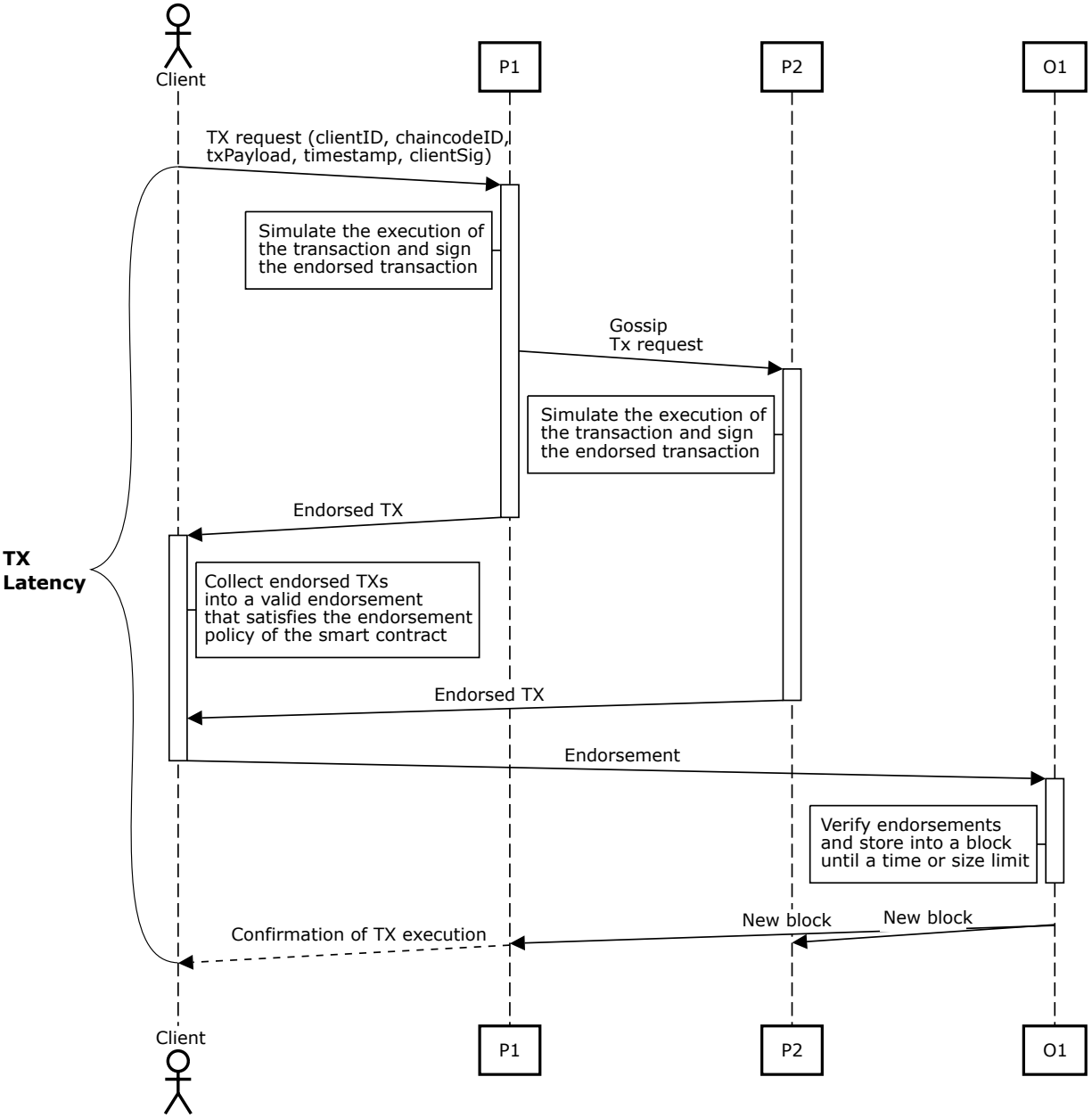

Figure 2.1: Sequence diagram of the Hyperledger Fabric transaction flow, considering a simplified deployment with two peers *P1, P2,* an ordering service made of a single orderer *O1* and a single *client*. In the *execution* phase, clients submit transaction proposals to endorsing peers, who simulate the transactions and return signed endorsements. In the *ordering* phase, the client sends the endorsed transaction to the ordering service, which orders transactions without executing them. Finally, in the *validation* phase, peers validate the transaction against the endorsement policy and update their copy of the ledger with the new block, committing the transactions that passed the validation process. The curly bracket measures the transaction latency intended as the time that occurs between the transaction request to its actual validation.

Smart Contracts for Certified and Sustainable Safety-Critical Monitoring

3

This chapter builds on the concepts and findings presented in my previous publication [20], where the integration of Databases and Smart Contracts for certified operations in industrial monitoring systems was explored and experimentally validated on a real-world SHM use-case.
The work has been expanded here to include additional context and further insights into the implementation and performance evaluation of the proposed framework.

The remainder of this chapter is organized as follows.
Section 3.1 provides an extension of the general background presented in Chapter 2, introducing the concept of sustainability in safety-critical monitoring applications.
Section 3.2 explains the motivations behind the research discussed in this chapter.
Section 3.3 reviews the related work on traditional and blockchain-assisted monitoring applications.
Section 3.4 describes the design of the proposed system, detailing its architecture, components and the smart contract implementation.
Section 3.5 presents the application of the framework to the SHM case study, along with the experimental results.
Section 3.6 concludes the chapter with a final discussion and a brief outline of the potential directions for future work.

3.1 Context

The deployment of sensor networks for monitoring safety-critical systems - i.e. those whose failure could result in loss of life, significant property damage, or severe environmental harm - has become increasingly

prevalent across various industrial applications, including Structural Health Monitoring (SHM) of buildings or bridges, industrial plants handling hazardous materials, smart buildings with critical life-safety systems, and energy distribution systems that maintain essential services.

These systems typically follow a layered architecture that includes IoT sensing nodes for on-field continuous monitoring, a local network layer for data collection, and cloud-based components for storing, visualizing, and analyzing the collected data [6]. The integration of these technologies provides essential data not only for real-time monitoring but also for post-mortem analysis of critical infrastructure, helping to prevent catastrophic failures and ensure both public safety and operational reliability.

Despite these benefits, the widespread adoption of sensor networks presents several key challenges, particularly concerning sustainable data storage and data integrity management in safety-critical monitoring systems. In applications like SHM, where structural failures could have devastating consequences, high sensor sampling rates are often required to produce accurate and reliable data for analysis. This generates substantial data throughput, and as monitoring infrastructures are typically in operation for decades, the boundless growth of databases becomes a pressing concern. Ensuring sustainable management of these ever-expanding data sets while maintaining their integrity is critical, as tampered or missing data could lead to erroneous conclusions about system health and potentially catastrophic outcomes in environments where system failure could endanger human lives or critical infrastructure.

3.2 Motivations

As highlighted in Section 3.1, managing database size growth is a critical challenge in Safety-Critical Continuous Monitoring applications.

One of the primary difficulties lies in the limitations of embedded systems within IoT sensor nodes, which often lack the processing power to perform on-the-edge data filtering. As a result, most data processing and reduction are deferred to cloud infrastructures. However, performing on-cloud data reduction in safety-critical applications introduces the challenge of maintaining data integrity. In these contexts, it is essential to ensure that critical data is protected from malicious deletion, particularly for use in post-mortem analysis where accuracy and completeness are paramount.

If data reduction is not handled correctly, it can result in serious consequences, potentially compromising the entire monitoring system and data collection efforts. Additionally, the deletion of data may be susceptible to tampering by malicious actors.

This chapter presents a novel platform architecture that leverages blockchain and smart contracts to implement certified data-removal policies. These policies aim to improve the sustainability of cloud databases in monitoring applications while providing anti-tampering guarantees. By using smart contracts, we ensure that the data-removal algorithms and policies are applied in a verifiable and tamper-proof manner. The proposed solution is finally tested within a real-world SHM application for rail bridge monitoring, where accelerometer data from an IoT sensor network is continuously streamed to a remote database.

3.3 Related Works

This section extends Chapter 2, reviewing the existing literature on real-time IoT continuous monitoring systems, finally focusing on those that integrate blockchain technology.

3.3.1 Continuous Monitoring Systems

Continuous monitoring systems deployed in real-time IoT applications require tamper-proof logging mechanisms to ensure the integrity of the collected data. This is especially important for systems that perform safety-critical operations, where ensuring that data used for analysis is unmodified is essential for accurate and reliable decision-making. However, several prominent sensor-to-cloud monitoring solutions, such as ExaMon and MODRON, do not fully address these tamper-proof requirements.

ExaMon [21], for instance, is designed for real-time ingestion of heterogeneous data from multiple sensing nodes, utilizing an MQTT broker for efficient data transmission. Despite its robust data orchestration via KairosDB and Cassandra, ExaMon lacks blockchain integration, which means it does not offer the tamper-proof guarantees required for data integrity in critical applications.

MODRON [22] is another sensor-to-cloud architecture that employs InfluxDB, a time-series database optimized for large-scale data management. MODRON recently introduced a blockchain plugin that allows relevant measurements to be stored on a blockchain, providing some level of data integrity [23]. However, this system still lacks the ability to implement shared policies for executing certified operations, such as data removal, which is a key requirement for systems with extensive data retention needs.

Both the reported works provide essential features for continuous monitoring, but neither addresses the challenges of tamper-proof data management and certified data removal policies that are relevant in safety-critical IoT systems.

3.3.2 Blockchain in Continuous Monitoring Systems

The application of blockchain technology to Internet of Things (IoT) Continuous Monitoring systems has been extensively explored in recent years due to its potential to enhance security, data integrity, and decentralization [24]. Several works in the literature examine how blockchain can resolve common issues in various domains that need to ensure the authenticity and immutability of data, such as healthcare [25] and industrial systems.

Many works discuss the general benefits of blockchain for IoT [24, 26–28], highlighting its role in securing IoT infrastructures and ensuring trust among stakeholders. Other works, such as Griggs et al. [29], propose specific architectures for integrating blockchain and smart contracts in the healthcare sector. In their work, a private Ethereum-based blockchain is used to manage sensitive patient data and log critical medical events, ensuring privacy and verifiability.

In the industrial domain, He et al. [30] propose a blockchain-based system, BoSMoS, to enhance the security of industrial IoT systems. The system captures snapshots of device software and stores them on a blockchain, with a configurable consensus mechanism for added security. They also provide experimental results demonstrating the system's performance during intrusion attempts.

Another relevant work by Kovstal et al. [31] extends earlier research [32] by presenting an architecture for managing IoT systems via a private blockchain. Their solution focuses on tracking and managing device configuration history, providing an audit trail that ensures security and accountability in industrial IoT networks.

Although these works provide valuable insights into how blockchain can be integrated into IoT systems, they focus primarily on specific verticals such as security and privacy. Consequently, despite the growing body of research on the application of blockchain in IoT systems,

there remain significant gaps when it comes to addressing storage space management and the implementation of certified operations such as data removal.

3.3.3 Blockchain in Structural Health Monitoring (SHM)

Aguzzi et al. [22] and Sidorov et al. [33] make use of private or public blockchain in SHM contexts to perform direct data storage by means of blockchain, thus storing the actual collected data directly in the distributed ledger. These approaches face the scalability issue of blockchain that is well documented in literature [34, 35], that makes blockchain technology not suitable for direct data storage.

Importantly, none of these studies address the storage space sustainability or the need for certified data-removal policies, which are essential when dealing with large-scale, long-term IoT monitoring systems. By contrast, in this chapter we focus on the integration of smart contracts in SHM storage systems to manage obsolete data and perform certified operations. We propose and validate a system that executes policy-based, tamper-proof data deletion using blockchain technology. These policies are designed to enhance the sustainability of cloud databases in monitoring applications while providing robust anti-tampering assurances. Through the use of smart contracts, we ensure that data-removal algorithms are applied in a verifiable, secure, and tamper-resistant manner.

3.4 Smart Contract Design

In this section, we illustrate the main design choices.
Subsection 3.4.1 describes the general requirements of our framework.
Subsection 3.4.2 presents the cloud layer, which involves both database and blockchain.

Subsection 3.4.3 focuses on the design of the smart contract.

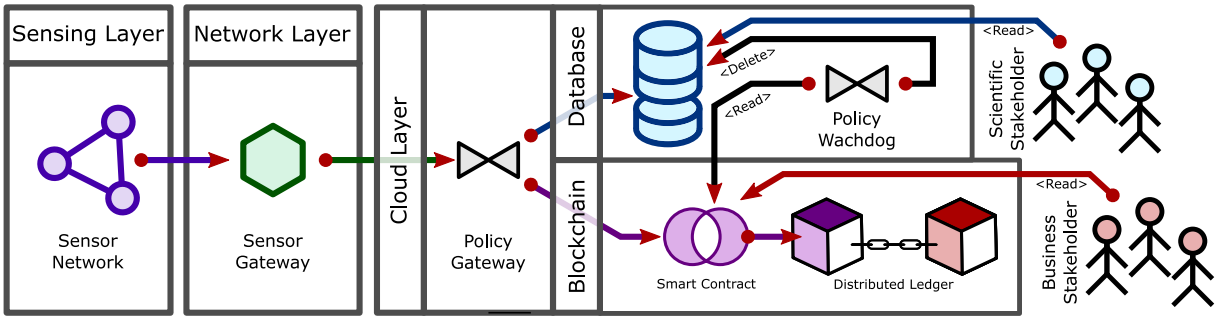

Figure 3.1: The proposed blockchain-enabled architecture. In the right side of the diagram, we can see the stakeholders enabled to interact with the main system components: the Database and the Smart Contract. While Scientific Stakeholders aim to a fast access, Business Stakeholders are interested in validating the operations made against the Database. The Policy Gateway acts as the intermediary component that orchestrates operations involving both the Smart Contract and the Database, and interfacing them with the IoT data ingestion system depicted on the left.

Figure 3.1 illustrates the proposed blockchain-enabled architecture, which orchestrates interactions between different stakeholders and system components. At the core of the system, a Policy Gateway mediates between the IoT data ingestion system, a traditional database for data storage, and a Smart Contract for transaction validation, effectively bridging the gap between high-throughput data collection and secure data management.

3.4.1 General requirements

Different stakeholders may need to interact with the Cloud Layer of Continuous Monitoring systems to access stored data, depending on the nature of the monitored infrastructure.
In Structural Health Monitoring (SHM) scenarios, such as the one used in this work for system validation, at least four distinct actors can typically be identified:

1. The *Infrastructure Manager*, responsible for overseeing the building and its maintenance.
2. A *Data Management Company*, tasked with handling data collection and analysis from the measurement systems.

3. The *Installation Company*, responsible for installing and setting up the monitoring equipment.
4. The *Government Ministry for Transport*, which oversees infrastructure compliance and safety at the national level.

In the event of disputes — often arising from accidents — these stakeholders may not fully trust one another, yet all may need access to the stored data. As a result, the cloud layer is responsible for storing data acquired by the sensors while guaranteeing the following properties:

- *Fast access* to time-series data to enable timely data analysis.
- *Data integrity*, ensuring the data cannot be altered or tampered with.

A typical cloud layer is built on top of a data ingestion platform, such as ExaMon [21], which utilizes a time-series database to provide efficient access to stored time-series data, thus addressing the need for rapid data retrieval.

However, the presence of non-informative data in the collected measurements — such as readings dominated by sensor noise or data collected during periods of known inactivity (e.g., nighttime in buildings) — negatively impacts system performance in two ways. Firstly, processing these uninformative measurements during data post-processing leads to unnecessary computational overhead without providing meaningful insights about the monitored system. Secondly, storing such data consumes valuable storage space that could be better utilized for measurements containing actual information about the system's behavior. For instance, in a building monitoring system, sensor readings collected during off-hours might primarily capture background noise rather than structural responses of interest, making them redundant for analytical purposes.

To address these issues, we propose a system architecture that assigns an expiry date to stored data based on a specific policy. This policy is designed to perform

computations on data received from the sensor network, applying an algorithm to determine the appropriate expiry time-lapse, i.e. the amount of time after which the data can be safely deleted to free up storage space without compromising the monitoring system's effectiveness.

3.4.2 Cloud Layer

The Cloud Layer architecture proposed in this work is designed to fulfill several tasks:

- Ingest and route data from the sensor network;
- Store the data in a fast-access database for use by scientific stakeholders;
- Implement a mechanism to delete non-meaningful data according to a policy agreed upon by the business stakeholders, thereby improving the long-term sustainability of the database.

Unlike the traditional database-centric approach to data storage, we propose supplementing the database with a blockchain system that executes transactions exposed by the Policy Smart Contract. To facilitate interactions between the blockchain and the database, two custom services have been developed:

- **Policy Gateway (PGW)** - This service is responsible for routing data from the network layer to both the database and the blockchain. It handles the execution of transactions on the blockchain while simultaneously writing data into the database.
- **Policy Watchdog (PWD)** - This service monitors the blockchain and manages the deletion of data from the database based on policy-driven conditions.

To ensure consistency between the data stored in the database and in the blockchain, the PGW assigns a unique string identifier to each block. This identifier serves as a key for recognizing and managing blocks across different storage systems. The PWD uses these

block identifiers to initiate delete requests in the database, ensuring that the blockchain and database remain synchronized.

Both the PGW and PWD are developed as Go[1] applications, leveraging the Hyperledger Fabric SDK[2] to communicate with the blockchain. This allows them to invoke the transactions defined in the Policy Smart Contract, ensuring seamless integration between the blockchain and database for executing certified data operations.

1: go.dev

2: Go Packages - fabricsdk package

3.4.3 Policy Smart Contract: requirements and motivations

Generally, as introduced in Section 2.3, a smart contract contains the definition of a set of suitable methods useful to interact with the distributed ledger of a blockchain. The methods which are intended to be used by the blockchain participants are exposed as transactions. In the Hyperledger Fabric framework, as detailed in Section 2.4, these transactions can be used to update the *world state* database, while the history of endorsed transactions is recorded within the blockchain's chain of blocks. The source code for the smart contract in Hyperledger Fabric is referred to as *chaincode*.

The *world state* can be populated with *assets*, which are collections of key-value pairs. The definition of these assets is included within the *chaincode*, and only the transactions defined in the chaincode are permitted to manage those assets. As a result, each *chaincode* operates on its specific *assets* within the distributed ledger.

In a permissioned blockchain, smart contracts can only be installed on peers once all relevant stakeholders have provided their approval. In the discussed system, the smart contract is designed to store and enforce the expiry time-lapse policy. Since all transactions executed are permanently stored on the blockchain's distributed ledger, the application of the policy is transparently tracked over time in an immutable and auditable ledger.

As represented in Figure 3.2, each business stakeholder is required to endorse the deployment of the Policy Smart Contract. Once approved, the Policy Smart Contract exposes various transactions that can be triggered both by the PGW and PWD services and by the business stakeholders themselves.

3.4.4 Policy Smart Contract: technical details

The smart contract presented in this thesis is designed to support multiple heterogeneous policies, allowing assets to be processed according to different client requirements. Indeed, a *policy* is defined as a class with a unique string identifier and a function that takes a data chunk as input, applies the specific policy's logic, and outputs an expiry period. This enables the deployment of multiple policy instances within the smart contract, each implementing its own logic, thereby supporting heterogeneous data processing through a multi-policy approach.

An implementation of Policy Smart Contract based on the Expiry Date Policy has been developed within the Hyperledger Fabric blockchain framework. The smart contract has been developed as a Go chaincode, utilizing the Hyperledger Fabric SDK to build the smart contract. It defines the assets to be stored by the blockchain, and the set of available transactions.

The **asset** is defined as a data structure containing several key elements:

- A *unique policy string identifier*, which allows keeping track of which policy have been applied to the asset;
- A *unique asset string identifier*, that is provided by the Policy Gateway (PGW) along with the corresponding data chunk;
- The *md5 hash of the data chunk* time series, computed at data chunk arrival;
- The asset's *expiry time-lapse*, determined by the applied policy and expressed as a time interval;

- The asset's *expiry date*, computed applying the *expiry time-lapse* to the reference creation time. To provide a shared, immutable timeline, the reference creation time is set to be equal to the blockchain's block creation date, which is a property assigned by the blockchain itself.

It's important to note that the actual sensor data is never stored on the blockchain, preventing unnecessary blockchain growth. The data chunk can always be retrieved from the system database using the unique chunk identifier.

A set of atomic private transactions has been implemented to interact with the *world state* and blockchain blocks. These transactions include CRUD operations (Create, Read, Update, Delete) on the *world state* database, which allow for the manipulation of stored assets. Additionally, they provide read operations on blockchain blocks to retrieve data from past transactions. These private transactions can be invoked by public transactions, which are available to smart contract users.

The fundamental **public transactions** include:

- **AddChunkWithPolicy**: computes the hash of a data chunk, triggers the specified policy to determine its expiry time, and creates the corresponding asset, which is then committed to the world state;
- **UpdateChunkExpiryDate**: updates the asset with a deterministic expiry date based on the asset's creation time, ensuring that the creation date cannot be falsified, thus preventing fraudulent policy application;
- **GetExpiredChunks**: returns all assets in the world state that have expired, provided that the expiry date is not in the future.

Detailed documentation on the implementation of the Policy Smart Contract can be found in the official repository[1].

1: Public repository available at: gitlab.com/ecs-lab/hyper-watchdog .

The designed smart contract supports various workflows, enhancing policy flexibility according to system

requirements. Policies can be applied when the asset is created or assigned at a later point. Furthermore, the policy applied to an asset can be changed before the asset is deleted.

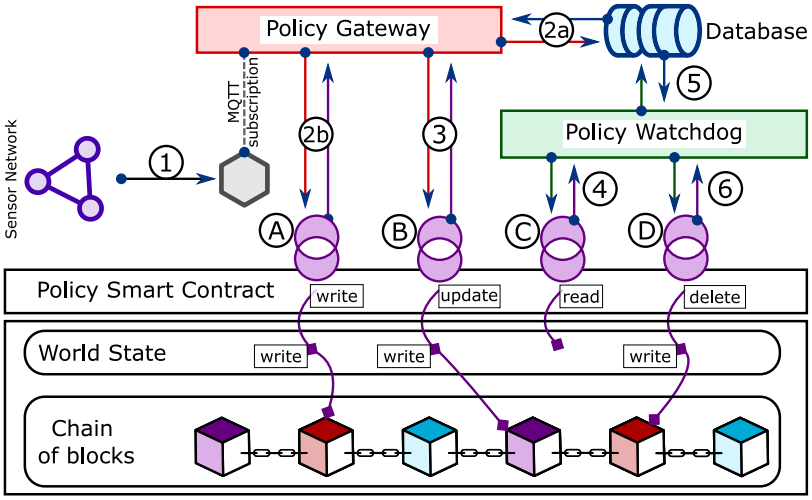

Figure 3.2: Proposed **system workflow** and **asset lifecycle**: Policy Gateway receives the data chunk from sensor network (1) through an MQTT broker; it requests the *AddChunkWithPolicy* transaction (A) to store chunk information into the database (2a) and the blockchain (2b). Then, *UpdateChunkExpiryDate* (B) is triggered (3) for making the blockchain compute the chunk expiry date. When the chunk has expired, its unique id will be included in the list returned by *GetExpiredChunks* (C) transaction, therefore the Policy Watchdog will be aware that the block can be deleted (4) and will issue a delete request against the database (5). If data removal is successful, the Policy Watchdog notifies it to the blockchain (6) by triggering the *DeleteChunkIfExpired* (D).

The workflow implemented in this work is illustrated in Figure 3.2. At the end of its lifecycle, the asset will no longer exist in the *world state* or the database. However, the full transaction history will remain permanently stored within the blockchain's blocks. As a result, it is always possible to verify that a missing data chunk was processed by the Policy Smart Contract.

3.5 Feasibility Analysis

In this section, we discuss the application of the presented framework on a real-world use-case belonging to the domain of Structural Health Monitoring.

It is important to note that while the data used in this analysis comes from a real-world application, the evaluation is conducted under ideal conditions. This means that

the experiments are designed to demonstrate the conceptual viability and potential benefits of the framework without accounting for the complexities and constraints introduced by a production-grade blockchain deployment. As such, the current evaluation confirms that the overall system functions as intended and effectively addresses the use-case problem.

A thorough validation of the smart contract's applicability, considering the features and challenges of a production-grade blockchain environment, requires further investigation. In the subsequent chapters, we will address this by introducing a methodology for comprehensive benchmarking and performance evaluation of smart contracts in industrial settings. We will leverage the smart contract presented in this chapter as the subject of our evaluation, providing a deeper understanding of its performance and scalability under real-world conditions.

The remainder of the section is organized as follows:
Subsection 3.5.1 introduces the use-case scenario.
Subsection 3.5.2 presents the developed expiry time-lapse policy.
Subsection 3.5.3 illustrates the environment that has been set up to run the evaluation.
Subsection 3.5.4 finally presents the validation results.

3.5.1 The Structural Health Monitoring Use-case

As introduced in Section 3.1, the proposed framework is designed to support continuous monitoring systems, with a particular focus on improving the sustainability of safety-critical industrial applications. During my PhD, I had the opportunity to leverage a real-world use case thanks to the ECS Lab at the Department of Electrical, Electronic, and Information Technology of the University of Bologna, where I worked under the supervision of my academic tutor, Professor Acquaviva Andrea. The

laboratory is involved in a project related to the continuous monitoring of a railway bridge, providing the perfect scenario to validate the proposed framework, leveraging data coming from a real-world use-case.

The validation use case is therefore based on this real-world railway bridge continuous monitoring application, and the architecture consists of various elements and services, as shown in Figure 3.1.

- The *Sensing Layer* represents the data acquisition system, comprising sensors that capture data at a frequency of $f_s = 833Hz$.
- The *Network Layer* takes care of ingesting and pre-processing data. It re-synchronizes the time series from the sensors and composes data chunks of a specified length, N_{chunk}. Each chunk contains the time series collected over a time span equal to $f_s N_{chunk}$. These chunks are then made available to the rest of the system via an MQTT broker.
- The *Cloud Layer* handles the storage of data chunks received from the upper layer, making them accessible to scientific stakeholders for further data analysis.

3.5.2 Expiry time-lapse policy

To determine the expiry time-lapse for stored sensor data, we designed a simple algorithm that is based on the mean signal energy of the sensor data arrays. In detail, let's first acknowledge that each 3-axis accelerometer produces three separate data arrays, corresponding to measurements along each axis. Then, the policy calculates the mean energy of each data array according to Equation 3.1.

$$\overline{E} = \frac{1}{N_{chunk}} \sum_{0}^{N_{chunk}} |x_i|^2 \tag{3.1}$$

High-energy data is typically recorded when the bridge is under stress, such as during the passage of a train.

In contrast, data collected when the bridge is not under stress (i.e., low-energy data) is likely to be non-meaningful and can be discarded earlier. The physical interpretation of the formula is that low-energy data is not as significant and, therefore, can be scheduled for earlier deletion.

The algorithm we propose compares the computed energy values against three different thresholds, one per each axis, established by a domain expert; based on these comparisons, the policy establishes an appropriate expiry time-lapse for the data.

3.5.3 Test bench setup

The system has been tested and validated by deploying the Policy Smart Contract as a Go chaincode running on an instance of the Hyperledger Fabric Test Network[2]. The deployed blockchain network includes two peer organizations, each with a single peer, and a Raft ordering service consisting in a single node organization. All experiments were conducted using a host machine equipped with an Intel^{R} Core^{TM} i7-10750H CPU @ 2.60GHz running 64-bit Ubuntu 20.04.4 LTS with 16 GB memory. The Fabric network was deployed using Docker Compose, with all nodes running as containers under the same Compose network on the same host machine. Consequently, network latency and other real-world limitations were not factored into the experiments.

2: For more information, see the official documentation at: hyperledger-fabric.readthedocs.io

For testing purposes, the inclusion of a real database did not impact system performance, as requests to the database could be processed in parallel with those to the blockchain. Thus, the experiments were conducted solely with the blockchain network and the associated services: the Policy Gateway (PGW) and the Policy Watchdog (PWD). These services were simulated using a Go application, while data chunks were injected into the PGW via an MQTT broker using a custom Python simulator. This simulator retrieved data from a historical time-series

database associated with the continuous monitoring application described in the previous section.

All the tests have been conducted implementing the policy proposed in Subsection 3.5.2, employing a blockchain-in-the-loop approach. The policies were fine-tuned so that the system could issue deletions at least a dozen times over the course of the experiment.

3.5.4 Quantitative Evaluation

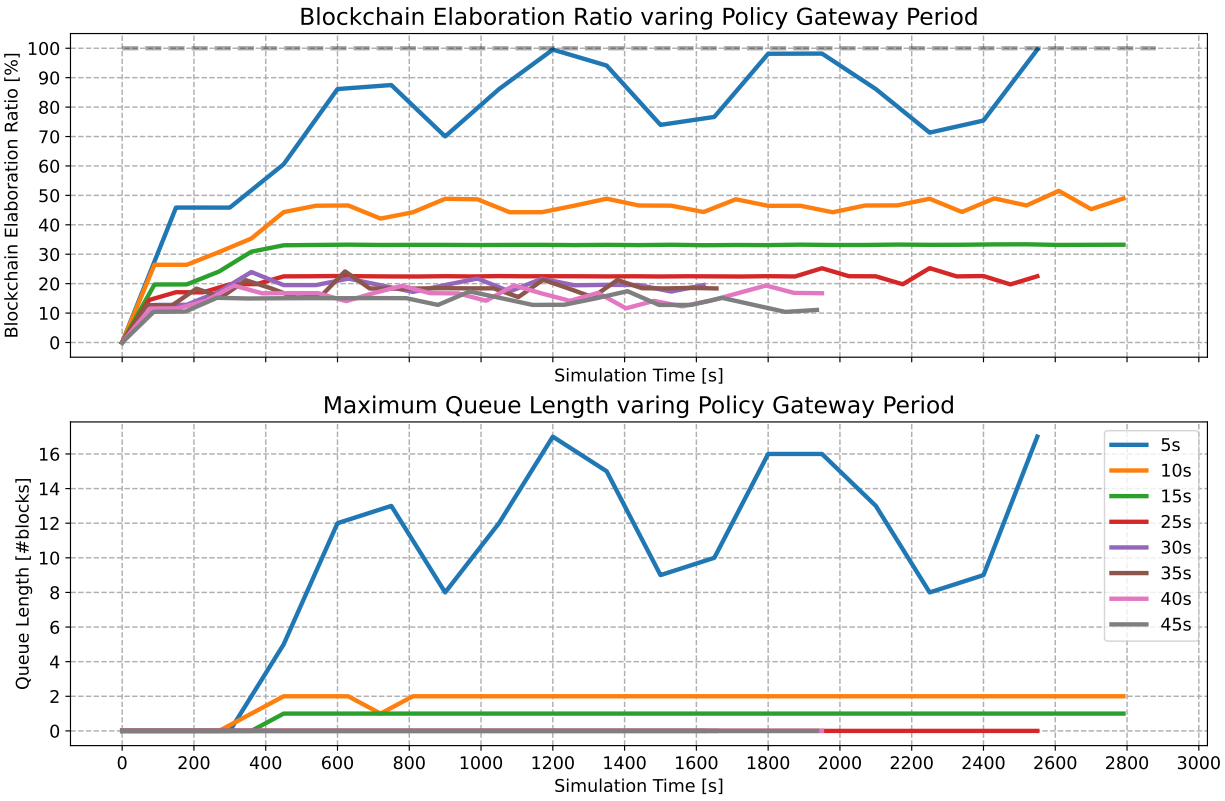

Figure 3.3: Blockchain Elaboration Ratio (BER) and queue length versus simulation time (s). Overflow condition BER >= 100% is represented as dashed grey line.

The dataset of the use-case we chose contains data coming from a session of 24 hours of continuous monitoring. Accordingly, we designed a test to simulate the system in an accelerated way that allows us to explore different parameters in a narrower time range.

More specifically, our main goal is to evaluate the system with a series of iterations using different chunk sizes, N. This parameter indeed heavily impacts the timings of

transactions, as will be shown by the following considerations. Since the sensors' acquisition frequency f_s is given, each chunk of data will store measurements for a time span equal to $t_c = N/f_s$ (s). As a consequence, the frequency at which the PGW is run in real-time context can be computed using Equation 3.2.

$$f_{\text{PGW}} = 1/t_c \quad \text{(Hz)} \tag{3.2}$$

Since in our blockchain-in-the-loop test environment the chunks are available a-priori, the Policy Gateway (PGW) can be fed-up at a higher frequency than it would in a real-time context. This allows us to perform accelerated simulations with a speed-up coefficient that can be evaluated using Equation 3.3:

$$\chi = t_c/t_{\text{PGW}} \tag{3.3}$$

In a blockchain-in-the-loop simulation, the blockchain is plugged into the simulation as-is; therefore, its performance is not affected by the speed-up coefficient of the simulation. Concerning the time duration δ_t of a transaction, i.e. the amount of time for a transaction to be fully endorsed after being requested, we can therefore state that it depends only on the blockchain's performance.

To take into account the slowness of the real-time blockchain within an accelerated simulation, we have to make an important consideration: between two consecutive interventions of the Policy Gateway (PGW), the transactions will be slower to be executed with respect to the accelerated environment. Therefore, transaction requests will eventually accumulate over time.

To address this issue, it is necessary to set a constraint to force the Policy Watchdog (PWD) transactions to be executed after an appropriate number of PGW transactions. This interval can be calculated as t_{PWD}/t_c , where t_{PWD} is the time that occurs between PWD runs.

If the total transaction time exceeds t_{PWD}, transaction requests will start to accumulate, which could lead to

incorrect identification of expired data chunks by the PWD. We define this condition as *overflow condition*, which can be expressed as in Equation 3.4:

$$\sum_{t}^{t+t_{\text{PWD}}} \delta_t > t_{\text{PWD}} \tag{3.4}$$

Consequently, we can define the Blockchain Elaboration Ratio (BER), which quantifies the system's ability to process transactions without generating queues, as shown in Equation 3.5:

$$\text{BER} = 100 \cdot \frac{\sum_{t}^{t+t_{\text{PWD}}} \delta_t}{t_{\text{PWD}}} \tag{3.5}$$

The overflow condition can then be expressed as a function of the BER as in Equation 3.6:

$$\text{BER} >= 100\% \tag{3.6}$$

By plotting the evolution of pending transaction requests over time (i.e., the queue length), we can observe that shorter chunk times t_c result in longer queues.
As demonstrated by the tests shown in Figure 3.3, smaller t_c values lead to an increased number of transactions, resulting in higher BER values.

3.6 Final remarks

In many different application areas, spanning from structural monitoring to logistics, either private or public actors have the need of managing data flows under the constraint of being able to store only a portion of the collected data. By combining two key technologies, namely the continuous monitoring data storage and a blockchain, we propose a framework that effectively limits the storage requirements of continuous monitoring systems leveraging the safe deletion of non-essential data, with the benefit of safeguarding critical information.

The system architecture, illustrated in Figure 3.1, includes a database that stores all incoming data from the sensor network. The Policy Smart Contract assesses the retention period for each data portion, recording this information on the blockchain, while the Policy Watchdog (PWD) filters out data that no longer meets the policy's relevance criteria.

This solution provides scientific stakeholders with the tools to perform high-performance data analysis, while business stakeholders can verify data integrity without the need for mutual trust. In this way, the system offers a robust tampering detection mechanism.

We analyzed the feasibility of the solution leveraging accelerometer data from a real-world railway bridge monitoring system, abstracting our experiments from the data storage technology and implementing the Policy Smart Contract on top of the Hyperledger Fabric blockchain framework. The validation demonstrates that the policy can accurately distinguish high-stress periods, such as train passages, from less significant data. At the same time, the system effectively controls storage space, preventing it from growing indefinitely.

More generally, the test results show that with well-tuned policies for a given use case, the system can free up a significant amount of database space while also improving overall security compared to traditional data storage systems.

Regardless of the nature of the continuous monitoring scenario, if the policy generates finite expiry intervals, it is possible to define an upper threshold for storage space, which will depend on the amount of deleted data relative to the ingested data. If a portion of the data is assigned an infinite expiry time, the storage space will eventually diverge. In the former case, our system can set a fixed upper limit on storage use, while in the latter, it helps slow the rate of storage growth.

The presented Policy Smart Contract system can be applied to any continuous monitoring application to

enhance long-term database sustainability without compromising performance. The blockchain technology at the system's core does not affect data ingestion speed. However, if too many policies are deployed or configured with high transaction rates, the data deletion system may slow down, potentially leading to inconsistent results.

To determine whether our system is suitable for a specific use case, simulation tests should be conducted, as outlined in the previous sections. The Blockchain Elaboration Ratio (BER) must be evaluated, and the overflow condition should be avoided. Additionally, simulations are necessary to estimate blockchain storage requirements and assess the impact of different blockchain deployment configurations, depending on the blockchain framework chosen.

For instance, when using Hyperledger Fabric, considering an approximate transaction size S_{tr} of $\sim 5\,\text{KiB}$, we can predict the blockchain size using Equation 3.7:

$$S = S_{tr} t \sum f_i \tag{3.7}$$

where t is the elapsed time and f_i is the frequency of requested transactions of type i.

For our use case, there are two transactions for each policy evaluation triggered by the PGW for chunk creation and expiry computation, and a variable number of transactions, γ, for policy checks and data deletion. The variable γ depends on the frequencies of PGW and PWD, defined as $\gamma = \lceil \frac{f_{PGW}}{f_{PWD}} \rceil$. Thus, the blockchain size can also be expressed as in Equation 3.8:

$$S = S_{tr} t \left(2 f_{PGW} + (\gamma + 1) f_{PWD}\right) \tag{3.8}$$

For example, with a PGW period of 1 minute and a PWD period of 3 minutes, assuming the worst case where all chunks expire immediately (i.e. $\gamma = 3$), the blockchain size would grow at a rate of $S = 0.0446t$ GiB/day.

Over one year, without data deletion, the database would reach 1.7 TiB. With data deletion, implementing the policy described earlier, the blockchain would only store around 8 GiB of transaction data, while shrinking the database by up to 75%, providing a storage efficiency gain of ~163 times the size of the blockchain.

To fully validate the smart contract's applicability in a real-world industrial setting, especially when factoring in the complexities and challenges of deploying it within a production-grade blockchain environment, we acknowledge that additional investigation is necessary. In the following chapters, we will address this by introducing a methodology for comprehensive benchmarking and performance evaluation of smart contracts in industrial settings. By applying this methodology to the smart contract presented in this chapter, we aim to gain deeper insights into its performance and scalability under real-world conditions.

Estimating the Performance of Smart Contracts for Industrial Applications

4

During the development of smart contracts, developers commonly utilize test networks to verify contract functionality. Test networks offer several advantages:

- They eliminate the need for real cryptocurrencies to execute transactions;
- They can be operated locally, simplifying the infrastructure setup;
- They often emulate the blockchain consensus mechanism, enabling faster approval of transaction requests to speed up the testing process.

While test networks are effective for validating smart contract logic, they fall short when it comes to assessing the performance of smart contracts under real-world conditions. Performance metrics observed in these environments can differ significantly from those in a production-grade deployment. For instance, blockchain networks like the Hyperledger Fabric Test Network[1] run on a single machine, often with Transport Layer Security (TLS) deactivated, and without accounting for real-world network dynamics such as latency and jitter.

1: For more information, see the official documentation at: hyperledger-fabric.readthedocs.io

In contrast, real-world deployments operating across the Internet face numerous performance constraints like network latency, jitter, and the physical distribution of nodes, that may drastically affect the performance of smart contracts. These limitations can impede the smooth operation of smart contracts, leading to notable differences between testing environments and production-grade conditions.

The work discussed in this chapter extends the preliminary studies presented at the DLT24 workshop [36], where the theme of estimating the performance of smart contracts was explored. More in detail, we focus on the challenges of estimating the performance of smart contracts when deployed in industrial applications. We

introduce a comprehensive methodology for assessing the end-to-end performance of smart contracts in industrial scenarios. The methodology is designed to be independent of the underlying blockchain technology, making it broadly applicable to various industrial contexts. To validate this approach, we apply it to the SHM use case presented in the previous chapter, which monitors a railway bridge using IoT devices while the blockchain framework manages the secure removal of data while ensuring tamper-proof operations.

The remainder of this chapter is organized as follows.
Section 4.1 provides an extension of the general background presented in Chapter 2, giving a broader idea of the industrial applications actually leveraging blockchain technology.
Section 4.2 explains the motivations behind the research discussed in this chapter.
Section 4.3 reviews the related works.
Section 4.4 discusses the choice of the metrics for evaluating the performance of smart contracts.
Section 4.5 presents the step-by-step methodology, discussing each single phase in detail. Section 4.6 contains some final remarks about the research discussed in this chapter.

4.1 Context

Blockchain technology has gained significant traction across various industries due to its ability to foster trust among stakeholders. A key feature of blockchain-based systems is the use of smart contracts, which encapsulate business logic and are executed on blockchain-specific distributed virtual machines [37]. With the growing adoption of blockchain in diverse sectors, it has become critical to thoroughly understand the performance of smart contract execution, particularly under the demanding conditions found in real-world industrial applications.

The importance of real-world scenarios becomes evident when considering the increasing prevalence of blockchain infrastructures in data-intensive and critical industrial applications. Examples include Healthcare Systems [38, 39], Smart Urban Transportation [40], Industrial Internet-of-Things [41], Smart Grids [42], Smart Logistics [43] and Structural Health Monitoring (SHM) [20, 44]. In these sectors, it is essential to assess the performance of smart contracts in real-world conditions, where factors such as scalability, efficiency, and responsiveness are crucial to the success of the whole system.

In the context of our SHM use case presented in Chapter 3, the blockchain framework manages real-time data from IoT devices monitoring critical infrastructure. The smart contract is responsible for executing operations such as the secure removal of IoT data from cloud databases. As with other industrial applications, these performance metrics—timely execution, scalability, and data integrity—are crucial for maintaining operational efficiency and meeting the stringent requirements of safety-critical environments.

Further investigation about the need for assessing smart contract performances in industrial scenarios is provided in Section 4.3.

4.2 Motivations

Several decision frameworks [45–47] have been developed to help organizations navigate the complexities of integrating blockchain into industrial applications. These frameworks guide stakeholders in evaluating blockchain's suitability by considering the need for decentralization, the balance between transparency and privacy, and the potential for trust enhancements.

Moving from strategic decisions to technical evaluation, smart contract execution performance becomes a critical factor in assessing the overall effectiveness of blockchain technology. It directly influences the feasibility,

efficiency, and scalability of blockchain-based solutions. The ongoing research into blockchain scalability [48] highlights the importance of this area of study.

Diverse interrelated factors influence the performance of smart contracts:

- *Blockchain internal components*, such as the consensus protocol.
- The *optimization of blockchain client implementations*, which impacts transaction execution speed.
- The *computational capabilities* and *resource allocation* of the infrastructure hosting blockchain nodes, which directly affect transaction processing efficiency.
- The *network infrastructure* connecting blockchain nodes, with its latency, throughput, and reliability, plays a crucial role in the system's overall responsiveness.

These interconnected factors must be considered when estimating smart contract performance, particularly in safety-critical industrial applications like SHM, where precise and timely operations are essential for detecting infrastructure anomalies and ensuring data integrity.

4.3 Related works

In this section, we review the existing works about the performance benchmarking of smart contracts in industrial applications. A thorough understanding of the current literature and available tools is essential to position the proposed methodology within the broader research landscape and address specific industry needs.

The following subsections discuss the existing literature that highlights the necessity of assessing performance metrics for smart contracts deployed in industrial settings. Then, we focus on the works presenting benchmarks and software frameworks for benchmarking. This exploration includes an analysis of their capabilities and

limitations, particularly in light of the methodological guidelines introduced by this paper.

4.3.1 Smart contracts for industrial applications

The integration of blockchain technology in industrial applications presents potential and unique challenges, particularly in relation to the performance of smart contracts. Various studies have highlighted the necessity of evaluating key performance metrics, such as throughput and latency, which are critical to ensuring the efficiency and responsiveness of blockchain-based solutions in industrial environments.

The survey presented by Bovenzi et al. [49] identifies the key drivers behind the adoption of blockchain technology within the Industry 4.0 sector. One of the primary challenges noted in their study is the poor transaction throughput of certain blockchain technologies, which becomes especially problematic in industrial scenarios that demand high performance.

Similarly, Shah et al. [50] discuss the role of blockchain technology in improving interconnectivity and automation across industrial sectors, such as manufacturing, IoT, and supply chain management. Their work underlines the importance of high throughput and minimal delay for real-time applications and efficient data exchange, emphasizing the need for blockchain systems to scale with the growing number of users, machines, and interactions.

Another relevant study by Fernandez et al. [51] explores the application of blockchain in enhancing cybersecurity and decentralization in Industry 4.0. Specifically, it discusses the benefits and challenges of integrating blockchain and smart contracts in industrial settings to improve security, transparency, and efficiency in managing complex automated systems and data exchange networks. The authors emphasize the need to evaluate the throughput of blockchain solutions in Industrial

Internet-of-Things (IIoT) scenarios, which often require the management of large volumes of transactions in real or quasi-real time. They also highlight the throughput and latency constraints present in Cyber-Physical Production Systems (CPPS), where real or quasi-real time reaction to data events is essential.

These studies clearly demonstrate the need for robust benchmarking frameworks that can assess the latency and throughput of smart contracts in industrial settings. These performance metrics are crucial for ensuring that the operations within these industrial applications are efficient and meet real-time or quasi-real-time processing needs, particularly in time-critical environments where timing constraints are stringent and can impact the overall system reliability and functionality.

The research presented in this chapter builds on this foundation by developing a methodology tailored specifically for such applications, focusing on the performance characteristics most critical to time-sensitive and safety-critical industrial operations.

4.3.2 Smart Contract benchmarking

Numerous studies in the literature focus on the performance evaluation of smart contracts, particularly in industrial settings. These works have been summarized in Table 4.1 for reference.

Table 4.1: Related works concerning blockchain benchmarks.

Work	Runs on a single machine	Networking	Target Blockchain Technologies	Use-case	Throughput evaluation	Latency evaluation
[52]	Yes	Single host networking (docker)	Hyperledger Fabric	Generic application	Yes	Yes
[53]	No	WAN	Hyperledger Fabric	Generic real-world industrial application	Yes	Yes
[54]	Yes	Single host networking (docker)	Hyperledger Fabric, Hyperledger Sawtooth, ConsenSys Quorum	Generic real-world industrial application	Yes	No
[55]	No	LAN (1Gbit Ethernet links)	Hyperledger Fabric	Generic application	Yes	Yes

A significant portion of the existing literature focuses on idealized conditions for performance evaluation [52, 54, 55], where the blockchain system operates under optimal conditions without taking into account the constraints and limitations associated with production-grade deployments. For instance, many studies fail to consider factors such as network latency, distributed node configurations, and the complexities of real-world network conditions, which can significantly affect the performance of smart contracts. Consequently, the results obtained may not accurately reflect realistic use-case scenarios.

An exception to this trend is the work by Guggenberger et al. [53], which specifically benchmarks Hyperledger Fabric in production-grade environments with geo-distributed nodes. However, even in this case, the study is limited to specific use cases and does not explore performance optimization in a broader range of industrial scenarios.

The research presented in this chapter aims to address these gaps by focusing on smart contract performance in production-grade environments, offering a more comprehensive evaluation framework that is relevant to industrial applications.

4.3.3 Frameworks for performance evaluation

A variety of benchmarking tools have been proposed in the literature to assess blockchain performance, with a particular emphasis on frameworks that enable both

Table 4.2: Related works concerning blockchain benchmaring frameworks.

Work	Supported Blockchain Technologies	Automated Blockchain Deployment	Runs on a single machine	Production-grade Blockchain	Networking	Realistic Network Parameters	Geo-Distribution of nodes	Custom Smart Contract Deployment	Transaction Throughput Evaluation	Transaction Latency Evaluation	Status
[56]	Ethereum, Fabric, Sawtooth	External script	Yes	Yes	Single host	No	No	No	Yes	Yes	Released
[57]	Virtually any blockchain	External script	No	No	LAN	Partial	Yes	No	Yes	Yes	Released
[58]	Fabric v2.0	Only for Fabric	No	No	WAN	Yes	Yes	Fabric only	Yes	Yes	Released
[59]	Fabric	Manual	Yes	Yes	Single host	Yes	Yes	No	Yes	Yes	Conceptualized

blockchain deployment and network emulation rather than solely benchmarking. A summary of all our findings can be found in Table 4.2.

Many of these tools come with certain limitations, particularly when applied to the benchmarking of smart contracts in production-grade environments.

For example, Nasrulin et al. [57] present a distributed benchmarking framework that supports multiple blockchain technologies, although it lacks full network emulation capabilities, such as the simulation of network throughput limits. Moreover, their framework requires a third-party script for blockchain deployment, and it does not allow running experiments on a single machine.

In contrast, Rasolroveicy et al. [56] introduce a benchmarking platform capable of running on a single machine, providing users with the ability to customize node deployment via a script, albeit completely lacking network emulation support, and therefore being not capable of emulating a geo-distribution of production-grade blockchain nodes. Also in this case, the blockchain network deployment is offloaded to a third-party script.

Kassab et al. [58] propose and release a cloud-only benchmarking framework that leverages a chaos generator for network emulation, allowing for simulating realistic network parameters and geo-distribution of nodes. Its automated blockchain deployment system relies on a Hyperledger tool that is capable of deploying Hyperledger Fabric networks upon the description of a network topology.

Lastly, Pan et al. [59] conceptualize a benchmarking framework that runs on a single machine and also includes network emulation support. However, their work lacks an actual implementation.

Overall, the reviewed frameworks reveal a fragmentation of tools and approaches for evaluating blockchain performance, particularly in industrial contexts, as it is also noted by Touloupou et al. [60] in their review.

Most tools are focused on evaluating the blockchain infrastructure itself, rather than the performance of specific smart contracts within a production-grade environment. Additionally, no current framework provides a fully integrated solution for programmatic deployment of a production-grade blockchain, the instantiation of custom smart contracts, and the simulation of realistic network conditions on a single machine. This poses a challenge for newcomers or industry experts attempting to assess smart contract deployments' capability to meet the demands of real-world industrial applications.

To address these limitations, the research presented in this chapter aims to conceptualize a clear methodology for evaluating the performance of smart contracts in industrial scenarios. The methodology disambiguates the various approaches used by different benchmarking frameworks and provides a reference implementation based on a real-world industrial use case.

4.4 Metrics for evaluating smart contracts performance

Evaluating the performance of smart contracts involves considering a range of metrics that provide insight into various aspects of their execution. These metrics can be classified into categories such as *resource utilization*, which includes metrics like CPU, memory consumption and network usage or *performance*, which measure factors such as transaction throughput and block time [49, 61].

For the purposes of this research, we focus specifically on metrics that characterize the end-to-end performance of a smart contract when it is integrated as an independent component within an industrial system. As such, we exclude metrics related to resource utilization (e.g., CPU or memory consumption) and internal blockchain-specific metrics (e.g., block time, blocks per hour). While these metrics do contribute to the overall performance of the smart contract, they are less relevant to other

components of an industrial application and cannot be used to assess the application's overall capability or to identify performance bottlenecks.

In the context of distributed industrial applications, we have selected two key metrics as the primary Key Performance Indicators (KPIs) for evaluating smart contract performance:

1. Average Transaction Latency (ATL)
2. Average Transaction Throughput (ATT)

These metrics provide a user-centric view of performance, treating the underlying blockchain technology as a "black box". This approach allows us to focus on the actual performance delivered by the smart contract as experienced by the industrial application.

The Average Transaction Latency (ATL) is defined as the time elapsed between the moment a transaction is requested and the moment it is approved:

$$t_{\text{lat}}(\text{sec}) = t_{\text{app}} - t_{\text{req}} \tag{4.1}$$

where t_{app} is the time at which a transaction is approved, t_{req} is the time at which a transaction is requested.

The Average Transaction Throughput (ATT) is defined as the number of transactions approved per second:

$$T(\text{TPS}) = n_{\text{app}}/\text{s} \tag{4.2}$$

where n_{app} is the number of transactions approved in a time span of one second.

These two KPIs, ATL and ATT, are critical for understanding the performance of the smart contract component, particularly in terms of how well it handles requests from other components within an industrial application.

By relying on these metrics, we aim to provide a comprehensive evaluation framework that can be used to assess smart contract performance in real-world industrial applications, offering valuable insights into how

well the system performs under operational conditions. Overall, these KPIs aims to achieve the following:

- Emphasizing smart contract performance as the key interface between the blockchain and the industrial application.
- Assessing the smart contract's performance independently of the specific computations it executes, allowing the evaluation to apply across various use-case scenarios.
- Providing a technology-agnostic evaluation, meaning that the performance assessment remains relevant regardless of the particular blockchain technology or implementation in use.

4.5 Proposed methodology

As discussed in previous sections, the performance of a smart contract in an industrial application is influenced by various factors. To accurately estimate and evaluate smart contract performance in production-grade deployments, we propose a step-by-step methodology that takes these factors into account.

Each step is discussed in the following subsections, while Figure 4.1 shows a bird's eye view of the overall methodology, illustrating the hierarchical structure of the data management workflow.

4.5.1 Automated Blockchain Deployment

The first step involves defining and deploying the blockchain network used for benchmarking. To ensure repeatability and allow for different configurations, the deployment process should be fully automated.

A configuration document is essential for encapsulating the key configuration elements of a production-grade blockchain deployment, particularly those that affect smart contract execution performance. These include:

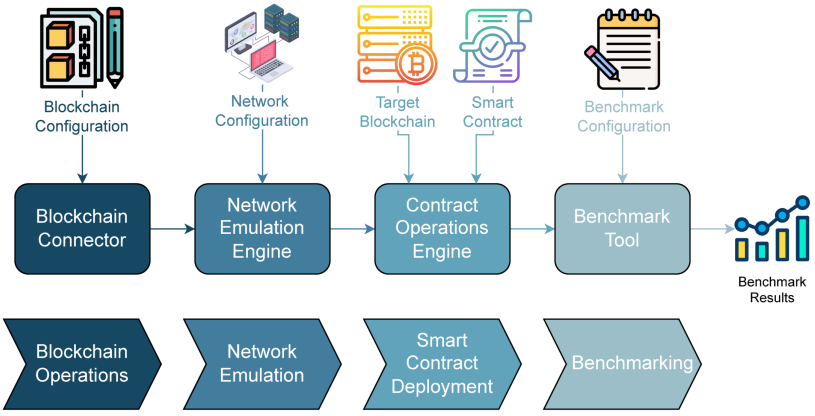

Figure 4.1: Outline of the methodology. The bottom layer shows the four main steps, while the upper part depicts the components.

- *Blockchain technology* - configuration details of the chosen technology, including the version of the consensus protocol, which impacts transaction latency based on its complexity and node interaction requirements, and the version of the client implementation deployed on the blockchain nodes, that can vary its performance based on the programming language and software version.
- *Nodes configuration* - parameters for each type of node in the network. The computational power and network capabilities of the nodes affect their execution performance. Production-grade features like communication encryption (e.g., SSL) should be enabled to mirror real-world deployments.
- *Network composition* - description of the blockchain network's topology, including the number of peer and validator nodes and their arrangement. The number and type of nodes influence the time needed for transaction endorsement and ledger consensus.
- *Network load* - parameters defining the network capabilities. If the blockchain serves multiple smart contracts, it is important to simulate the additional workload to reflect how the resources are shared among concurrent smart contracts. This affects the number of transactions endorsed per second.

The deployment must occur on host machines with computational power similar to those used in production-grade environments for the final benchmark results to be accurate. This step facilitates the deployment of smart

contracts, even for existing blockchain systems, in both permissioned and permissionless blockchains.

4.5.2 Emulation of Real-World Network Properties

The second step focuses on the network layer that connects the blockchain nodes. A production-grade blockchain is typically decentralized and geographically distributed, thus presenting unique challenges for performance.

Both permissionless and permissioned blockchains benefit from geo-distribution:

- In permissionless blockchains, distributing the nodes globally helps to prevent censorship, monopolies, and security vulnerabilities by spreading consensus across a broad network [62, 63].
- In permissioned blockchains, geographically dispersed nodes reduce reliability issues, improve availability, and prevent single points of failure [64–67].

When evaluating blockchain performance, it is important to emulate real-world network conditions, especially when a true blockchain network deployment is not available. The network emulation must meet the following requirements:

- Configure fundamental network properties such as latency, jitter, throughput, and packet loss to simulate real-world conditions.
- Ensure that all blockchain nodes and clients can seamlessly join the emulated network.

The network emulation parameters must be chosen according to the features and limitations of the real-world network on which the smart contract would be deployed in production.

This step allows the assessment of network performance in non-existing blockchains by simulating realistic conditions, which are critical for accurate performance evaluations.

4.5.3 Smart Contract Deployment

In the third step of the methodology, the smart contract is deployed on the blockchain network. This operation generally involves the execution of a series of blockchain-specific commands, culminating with a transaction that deploys the smart contract making it available in the blockchain network.

Different blockchain platforms may support various types of transactions. For example, Hyperledger Fabric supports both read-only and read-write transactions:

- *Read-only transactions* involve retrieving data from the node's local ledger copy.
- *Read-write transactions* require the consensus algorithm to validate the transaction request, which consumes significantly more computational resources.

For benchmarking, it is recommended to target read-write transactions since they involve the consensus process and provide a more comprehensive view of the blockchain's performance.

4.5.4 Benchmarking

The final step in the methodology involves benchmarking the smart contract to assess its capabilities and contextualize the results within the specific industrial use case. The goal is to determine how effectively the smart contract can meet the demands of the application. Additionally, once insights are gathered, further optimization and tuning of the blockchain parameters can be performed.

During the benchmarking process, the performances of the smart contract execution must be evaluated and expressed by means of the Key Performance Indicators (KPIs) introduced in Section 4.4:

1. *Average Transactions Latency, ATL (sec)* - measures the time that occurs between a transaction request and its eventual confirmation, thus providing relevant insights into the system's responsiveness. Regardless of the specific blockchain technology, we know that a transaction is confirmed when the block containing the transaction is approved according to the consensus algorithm.
2. *Average Transaction Throughput, ATT (TPS)* - quantifies the volume of transactions the blockchain technology can effectively approve per unit of time.

Additional metrics may be considered depending on the blockchain's configuration and the specific industrial application. For example, if the smart contract is deployed on a private blockchain, internal metrics such as resource utilization and block processing time may offer further insights into the system's performance and optimization potential.

Importantly, to comprehend how efficiently the smart contract can meet the demands of the specific industrial application it serves, the benchmark scenario must reflect the peculiarities of the use-case, e.g. required minimum transaction throughput, concurrent usage by multiple applications, presence of a load balancer, etc. This can be done by editing the blockchain configuration and by simulating particular conditions by means of the network emulation layer discussed before.

Moreover, if the smart contract contains configurable parameters, a suitable number of tests must be conducted in order to explore a significant set of working points and choose the optimal one that best satisfies the served industrial application, based on the evaluation of the discussed metrics.

4.6 Final remarks

In this chapter, we delved into the design of a comprehensive methodology for estimating the performance of smart contracts in industrial applications, focusing on their behavior in production-grade environments.

At first, we explored the existing literature trends and highlighted the need for robust performance benchmarking in industrial blockchain applications. Indeed, as smart contracts become more prevalent in mission-critical systems, the ability to accurately predict their performance in real-world production-grade environments becomes a vital part of their successful integration.

The methodology developed in this chapter provides a structured approach to benchmarking smart contracts by simulating the complexities of real-world network conditions and blockchain configurations. It includes four key steps.

The first one takes into account the blockchain operations. We discussed the importance of having an automated blockchain deployment system, that is able to perfectly reproduce blockchain network deployments taking into account production-grade configurations.

The second one puts the focus on the network emulation. We discussed the need of emulating real-world network conditions when a true blockchain deployment is not available, and we underlined that the simulation parameters should be chosen to reproduce the network on which the smart contract would be deployed in production.

The third step takes into account the smart contract deployment. This step is mostly dependent on the target blockchain technology. In this context, we discussed how to properly choose the target transaction for performing benchmarks.

The last step involves the smart contract benchmarking. We discussed and selected two key performance indicators already discussed in the literature, Average Transaction Latency (ATL) and Average Transaction

Throughput (ATT), as central metrics for evaluating how well smart contracts perform in distributed environments. These metrics indeed provide a user-centric view of system responsiveness and scalability, offering valuable insights into how effectively a blockchain network can handle the transaction demands of an industrial application. We also underlined the importance of using an automated benchmarking system to ensure the repeatability of experiments.

Case-Study: Performance Evaluation of a Real-World Safety-Critical Structural Health Monitoring Application

5

The research discussed in this chapter has the aim of applying the methodology proposed in Chapter 4 on the Structural Health Monitoring (SHM) application presented in Chapter 3, thus providing a practical implementation of our methodology on a real-world case study.
This work serves different purposes:

- serve as a guideline to demonstrate how researchers can conduct experiments emulating different real-world scenarios using our methodology as a foundation;
- assess how a real-world scenario is affected when deployed in a production-grade environment, comparing it to the ideal conditions of a test network;
- provide a blueprint for forthcoming research works.

In Chapter 3 we designed a blockchain-powered framework for the secure removal of IoT-generated data from databases, with focus on a real-world Structural Health Monitoring (SHM) application on a railway bridge. The primary emphasis was on leveraging Hyperledger Fabric smart contracts to execute certified operations throughout the data management process while ensuring timely operations and tracing any malicious tampering or unintended modification.

Our methodology is applied to assess the performance of the SHM smart contract in the context of a production-grade deployment. This aims to verify its ability to meet the specified requirements and quantify its performance in a real-world setting. The resulting implementation has the architecture depicted in Figure 5.1.

The remainder of this chapter is organized as follows.
Section 5.1 discusses the implementation of the framework according to the proposed methodology.
Section 5.2 presents the experimental results of applying

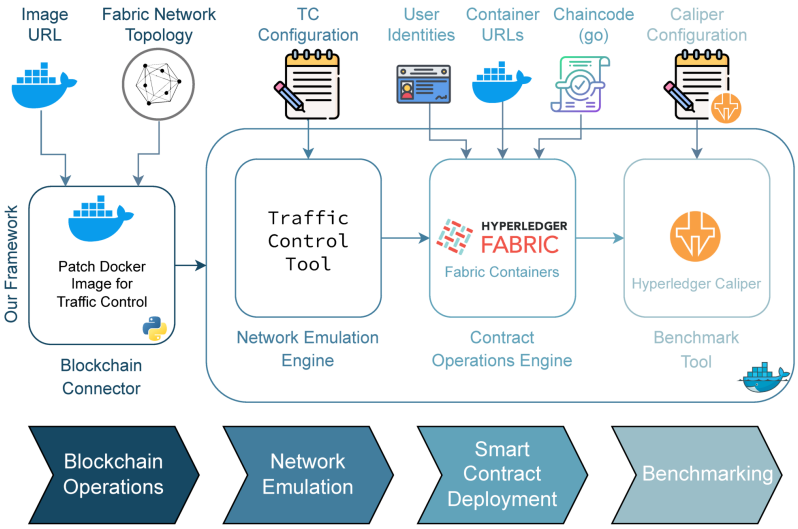

Figure 5.1: Outline of the implemented framework.

the developed framework for benchmarking the case-study smart contract.
Section 5.3 thoroughly discusses the experimental results. Section 5.4 contains some final remarks about the research discussed in this chapter.

5.1 Implementation of the software framework

As explained in Subsection 4.5.3, having a framework for setting up blockchain networks and performing benchmarks is a convenient way to achieve the experiments' repeatability and consistency.

We developed a modular framework using Python 3, leveraging on open-source components. In this way the resulting software is extendable to accommodate diverse blockchain technologies, network emulation strategies or benchmarking tools. Both the framework and a collection of tests are publicly released [68], with the repository being anonymized at the time of writing this thesis because I submitted part of my research to an academic journal with double-blind peer review.

5.1.1 Automated deployment of production-grade Hyperledger Fabric networks

The first pillar of our methodology is to have an automated system for configuration-based deployment of production-grade blockchain networks.

Since the target smart contract is designed for Hyperledger Fabric blockchain, we have developed a framework to facilitate establishing and operating production-grade Hyperledger Fabric blockchains. This framework operates on a definable topology, allowing for a descriptive representation of the network structure as detailed in Section 2.4. Thanks to this approach, we can declare multiple Fabric Organizations along with a customized number of Peers. Also, we can setup an Orderer Service that can be tailored to include a custom number of Orderers.

All the Fabric components are spun up as Docker containers and configured as production-grade nodes, with TLS enabled and Raft consensus algorithm.

The framework is responsible for orchestrating the setup process in a systematic sequence. This process entails the sequential establishment of Fabric Certificate Authorities, followed by the integration of Peer Organizations and their respective Peers, and finally resulting in the configuration of the Orderer Organization and its Orderers.

5.1.2 Emulation of Real-World Geo-Distributed Network

The second pillar of our methodology is to provide either a real-world deployment on an already running blockchain network, or a network emulation that takes into account real-world emulation parameters to mimic real geo-distributed networks.

After launching the Peer and Orderer containers, we design our framework to trigger the Traffic Control tool. This action aligns the network parameters, as specified in the configuration, ensuring that the intended network conditions are met throughout the operational phase.

Network emulation strategy

We decided to exploit the *Traffic Control* (*tc*) tool to emulate a network connection between nodes.

Traffic Control (tc)[1] is a networking tool commonly used in Linux operating systems to manage and control network traffic. It provides various mechanisms for shaping, prioritizing, and controlling the flow of data packets within a network interface. *tc* enables administrators to enforce quality of service (QoS) policies, manage bandwidth, and regulate network behavior according to predefined rules. It manipulates the Linux kernel's network packet queuing and transmission mechanisms. It primarily uses the *qdisc* (queueing discipline) framework to control the queuing and scheduling of packets. In particular, it allows defining different queueing disciplines, which determine how packets are organized and treated in queues. Each *qdisc* has specific characteristics that control factors like bandwidth allocation, delay, and packet dropping. Among all the possibilities, administrators can use *tc* to build *netem* queue disciplines. Those kinds of *qdisc* provide network emulation functionality for emulating the properties of real-world networks. In particular, this queue discipline provides many network impairments to packets, including delay and jitter, packet loss, and throughput limitation. Leveraging queueing disciplines allows administrators to perform network emulation, simulating network conditions like latency, packet loss, and jitter.

1: See the official Linux manual page.

The target smart contract is based on a containerized deployment of Hyperledger Fabric. Inspecting the official Docker[2] images, we can notice that the blockchain components (CA, peer, orderer) are built on top of the

2: See the official Fabric's Docker Hub profile.

3: hub.docker.com/alpine

alpine:3.16 [3] Docker image of the Alpine Linux distribution. Therefore, we built new Docker images for the Hyperledger Fabric components to support the Traffic Control tool inheriting from the following official images: *fabric-peer:2.4.9*, *fabric-orderer:2.4.9*, *fabric-ca:1.5.6*.

Choice of real-world network emulation parameters

To emulate an authentic network environment according to the presented methodology and coherently with the selected use case, we considered that this kind of smart contract is typically hosted on virtual machine servers offered by cloud providers. Indeed, for industrial applications, the blockchain nodes are most likely deployed using BaaS (Blockchain as a Service) platforms or manually by leveraging virtual machine servers to spin up the blockchain nodes. In both scenarios, the effectiveness of network communications is related to the performance of the underlying connection links that interconnect the nodes within the Cloud Provider's infrastructure.

The performance report [69] presents the network performance exhibited in the years 2019-2022 by the foremost public cloud providers: Amazon Web Services (AWS), Microsoft Azure, and Google Cloud. It also encompasses inter-region and inter-AZ measurements. We channel these insights to emulate a permissioned blockchain network deployed across inter-regional host machines, choosing Microsoft Azure as the reference cloud service purely to present a realistic example.

The chosen network emulation tool, *tc*, allows us to assign a *delay* value to the packets received by each Docker container, thus by each one of the blockchain nodes. As a result, the measured ping between two nodes will equal the sum of their respective delay values: $p = d_1 + d_2$.

We analyzed the data reported by ThousandEyes regarding Azure inter-region ping, finding that the data can be mocked as a normal distribution with a mean

value of $\mu_{\text{ping}} = 148\text{ms}$ and a standard deviation equal to $\sigma_{\text{ping}} = 78\text{ms}$.

tc allows randomizing the delay value per each packet according to a Gaussian jitter distribution: $d - j_i \leq d \leq d + j_i$. We can exploit the properties of normal distributions to observe that the linear combination of the normal distributions of delays gives as output normally distributed values; in particular, we can set *tc* to randomize the delay of nodes using a normal distribution with a mean value equal to:

$$\mu_{\text{delay}} = \frac{\mu_{\text{ping}}}{2} = 74\text{ms} \tag{5.1}$$

and a standard deviation equal to:

$$\sigma_{\text{delay}} = \frac{\sigma_{\text{ping}}}{2} = 39\text{ms} \tag{5.2}$$

As a result, the ping between random couples of containers will follow a normal distribution, reproducing the real Azure inter-regional servers.

In estimating network performance parameters, we need to acknowledge that different virtual machine choices lead to varying capabilities regarding throughput performance. In the context of a moderate to low-scale deployment, which aligns seamlessly with the requisites of running a solitary Hyperledger Fabric node tailored for a private blockchain configuration, we can take as a reference target the "Standard_E8s_v4" virtual machine offered by Microsoft Azure, which boasts an average inter-regional throughput measure amounting to 2.92Gbps according to the 2022 report authored by Cockroach Labs [70].

5.1.3 Deployment of Fabric Chaincode

The third step of our methodology expects the automatic deployment of the smart contract.

Therefore, in our case, the automation framework needs to deal with instantiating the Fabric chaincode (i.e. Fabric smart contract) in the network.

The first operation that the framework must carry out is to initialize a Fabric channel, and to include all the nodes in it.

Then, the specified smart contract can be deployed onto the established channel leveraging the Fabric SDK.

5.1.4 Benchmarking

The last phase of our methodology concerns benchmarking the deployed smart contract, leveraging the metrics discussed previously.

Accordingly, we looked for a benchmarking tool able to interact with the target running smart contract and measure the performances leveraging the Average Transaction Latency (ATL) and the Average Transaction Throughput (ATT).

To the best of our knowledge, the most suitable tool for this purpose is Hyperledger Caliper [4] by Linux Foundation.
It consists of an open-source blockchain benchmarking solution that facilitates the quantification of smart contract performance by executing benchmark runs based on predetermined configurations. The transactions load can be set to be constant (fixed-rate), variable according to a predefined law or variable according to the backlog queue (fixed-load).
Upon execution, Hyperledger Caliper generates a comprehensive performance report, encapsulating the performance indicators we mentioned before.

4: hyperledger.github.io/caliper

It can accommodate many blockchain platforms, including Hyperledger Fabric, Besu, and Ethereum. Notably, its open-source nature ensures its adaptability and potential for expansion, enabling developers to extend its support to other blockchain technologies.

5.2 Experimental results

This section illustrates the results of two distinct benchmark runs performed against the case-study smart contract.

5.2.1 Preliminary setup

The host machine is powered by a 16-core Intel Xeon (Cascadelake) @ 2.394GHz, 120GiB RAM. The Fabric topology is composed of two Organizations, including two Peer nodes each and an ordering service with three Orderer nodes, for a total of seven nodes as in the case-study application described in Chapter 3.

As the methodology suggests, we limit the deployed Docker containers to avoid computational power bottlenecks so that each Peer and Orderer can access the computational capabilities of two cores.

Considering the smart contract chosen for benchmarking, we identified *AddChunkWithPolicy* as the read-write transaction to target. Moreover, we identified *Chunk Length* as a parameter that can be tuned to modify the size of the chunk of data that gets processed by the system and, consequently, the number of transactions requested per second to satisfy the industrial application.

The size of the transaction request may be expressed in KiB using a conversion coefficient. The conversion can be made starting from the chosen *Chunk Length* applying the formula provided in Equation 3.7. Therefore, we can perform different tests varying the *Chunk Length* while expressing the results in terms of transaction size for a better understanding.

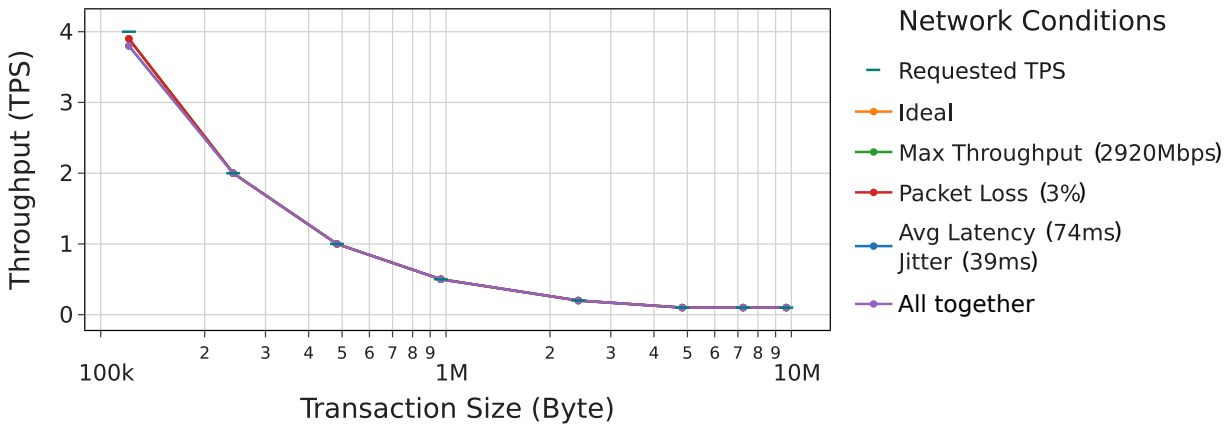

Figure 5.2: Results of the first benchmark run in logarithmic scale: throughput varying transaction size. Different colors represent the desired throughput and the different emulated network conditions.

5.2.2 First run: evaluating the performances beyond idealized environments

The first benchmark run, according to the methodology, has the aim of evaluating the performances in non-ideal deployment environments.

Specifically, we evaluated the target smart contract under various scenarios listed below:

1. *ideal conditions* with no disruptions;
2. *real-world latency*, introducing network constraints to emulate the latency and jitter values discussed in Subsection 5.1.2;
3. *real-world throughput limitations*, replicating the maximum throughput constraints detailed in Subsection 5.1.2;
4. *real-world packet loss*, introducing network constraints to emulate the packet loss values discussed in Subsection 5.1.2;
5. *real-world network*, incorporating all the constraints simultaneously.

The input transaction request rate reflects the use-case requirements for managing the industrial application.

The results of this benchmark run are depicted in Figure 5.2 and Figure 5.3. We can observe how the smart contract, when subject to a realistic deployment, is capable of satisfying the proposed use-case requirements in

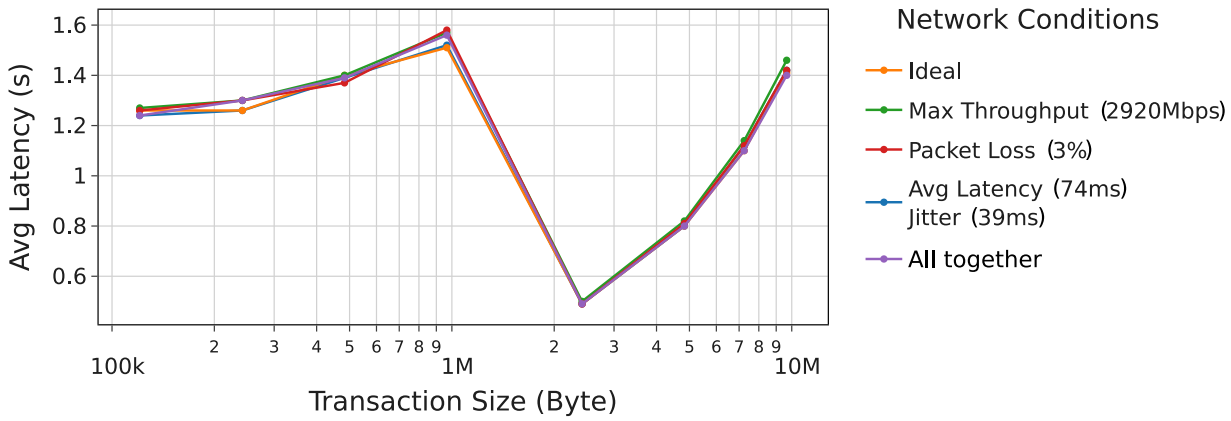

Figure 5.3: Results of the first benchmark run in logarithmic scale: average transaction latency varying transaction size. Different colors represent different emulated network conditions.

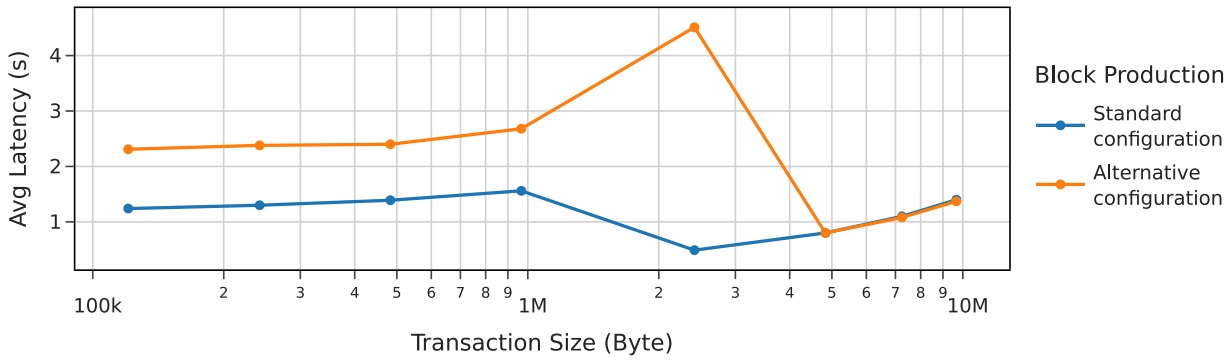

Figure 5.4: Comparison of average latency obtained using two different block generation configurations. Logarithmic scale.

terms of Average Transaction Throughput (ATT). The network conditions simulated according to the proposed methodology have a negligible impact on the smart contract performances. However, we can observe that the resulting Average Transaction Latency (ATL) may be up to 6% higher than the ideal conditions.

In this context, given that the industrial application in question processes a continuous data stream, we have the flexibility to adjust the batching size, thereby modifying the size of the processed transactions that carry the data chunks. Examining the chart in Figure 5.3, it becomes evident that whenever we select the batch size value, the system will adapt, necessitating a correspondingly different throughput, thereby impacting the ATL. In particular, it is possible to identify a specific configuration that yields an optimal latency value.

In our deployment using Hyperledger Fabric, we hypothesized that these latency differences stem from the block creation algorithm configuration. Subsequently, we validated this assumption by conducting the same experiment with an alternative block production configuration, the results of which are illustrated in Figure 5.4.

This experiment allows us to state that the proposed industrial application can be successfully run in real-world conditions with minor differences in ATL with respect to an ideal deployment. Moreover, the obtained insights can be leveraged to identify an optimal configuration that offers the lowest ATL while still satisfying the application's needs. In addition, the experiments shed light on the impact on ATL given by the block configuration of the underlying blockchain, suggesting that the latter can be customized to optimize the smart contract's performance further.

5.2.3 Second run: evaluating the scalability of the smart contract over multiple industrial applications

Industrial applications are commonly designed with scalability as a foundational consideration, anticipating the need to handle increased workloads and concurrent operations. This approach is justified by the dynamic and evolving nature of industrial processes, where the ability to scale ensures adaptability to growing demands and technological advancements.

In crafting this benchmark run, our aim was to stress test the smart contract's performance at edge conditions by inputting the maximum transactions per second possible. For this purpose, we leveraged the fixed-load Caliper configuration, which varies the input TPS dynamically, ensuring that the backlog queue of undone transactions is always non-empty. This deliberate approach enables us to comprehensively assess the smart contract's maximum capabilities within the parameters

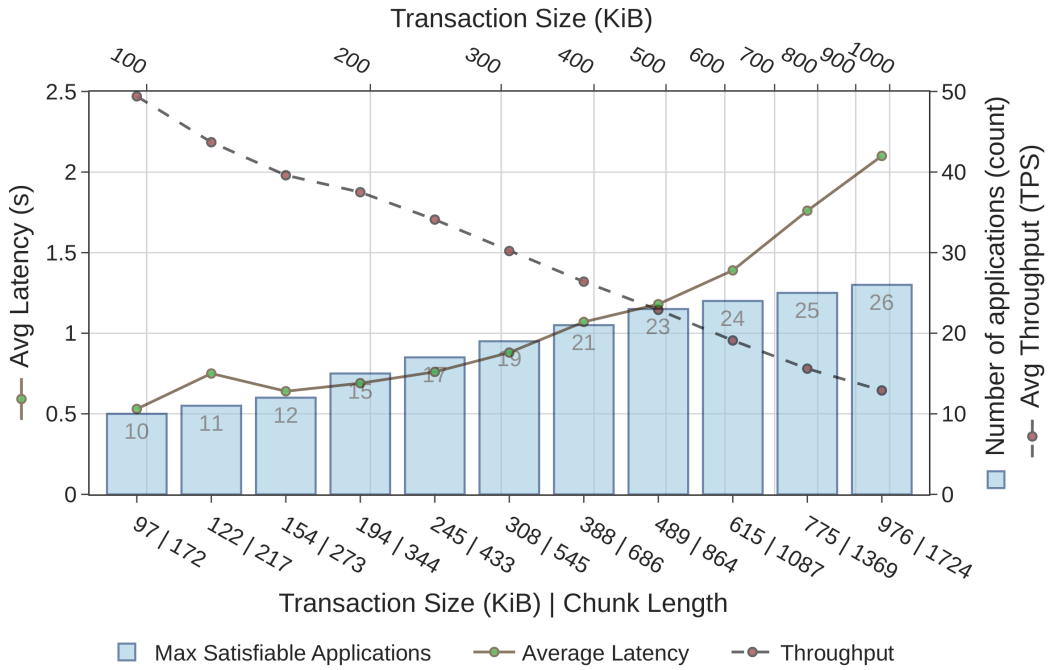

Figure 5.5: Results of the second benchmark run. The line plots show the maximum average latency and throughput that the system under test is able to provide when using different transaction sizes. The histogram shares the same horizontal axis and shows the number of concurrent industrial applications that the smart contract deployment can satisfy for different configurations of Chunk Length. Logarithmic scale.

of a production-grade deployment, as delineated by the methodology presented.

As explained before, we can modify the *Chunk Length* parameter of the use-case smart contract to vary its working point. Indeed, varying the *Chunk Length* implies a change of the *Transaction Size*.

The outcomes of the second benchmark run are presented in Figure 5.5. The relationship between ATT and Transaction Size is highlighted: the former decreases as Transaction Size increases, whereas Average Latency demonstrates the opposite trend.
Given N as a set of similar industrial applications running on the use-case smart contract, we can state that the required throughput is equal to the throughput requested for running one application multiplied by the number of concurrent applications:

$$\mathrm{TPS}_{\mathrm{Req}} = N * \mathrm{TPS}_{\mathrm{App}} \tag{5.3}$$

Starting from the experiment results, we can therefore calculate the maximum number of satisfiable applications as:

$$N = \mathrm{TPS}_{\mathrm{Res}}/\mathrm{TPS}_{\mathrm{App}} \tag{5.4}$$

As we can observe in Figure 5.5, the number of concurrent industrial applications that the smart contract deployment can satisfy varies with the Transaction Size. More specifically, the latter smart contract parameter can be adjusted to accommodate more concurrent use cases at the cost of a higher ATL.

For instance, with a chunk length of 864, resulting in a transaction size of 489 KiB, the average throughput is approximately 23 TPS. This configuration can effectively serve 23 concurrent Structural Health Monitoring (SHM) applications while maintaining an average latency of less than 1.5 seconds.

We can conclude that the proposed smart contract can run a variable number of concurrent industrial applications in real-world conditions. The system can be tailored to serve additional use cases at the expense of increased latency. The ultimate choice depends on the use-case's timing constraints.

5.3 Discussion

The obtained experimental results show that the simulated real-world network conditions exert a minimal impact on smart contract performance, suggesting the resilience of the case-study industrial application to production-grade network environments.

Applying the proposed methodology allowed us to estimate the smart contract performance under realistic deployment conditions. Therefore, we can observe that:

- the application is capable of meeting the use-case requirements in terms of Average Transaction Throughput (ATT);
- the Average Transaction Latency (ATL), which causes delays in the application's functionalities, is significantly influenced by the transaction size and the block configuration. As a consequence of these observations, tuning the smart contract to optimize ATL empirically is possible.

A further benchmark run sheds light on the smart contract's scalability to accommodate more than one concurrent industrial application in real-world conditions. Thanks to the proposed methodology, we are able to quantify both the maximum number of satisfiable concurrent applications and the resulting ATL under a production-grade deployment. Therefore, we can choose the optimal trade-off, taking into account the use case's timing constraints.

Clearly, these results are useful to evaluate the smart contract under test, and experimentation on other smart contracts with the same network conditions could lead to different numerical results based on the blockchain peculiarities.

5.4 Final remarks

The methodology proposed in this thesis is intentionally designed to be agnostic regarding the choice of blockchain technology for deploying smart contracts in industrial applications. We illustrate the application of our methodology through a real-world industrial case study, which utilizes Hyperledger Fabric smart contracts and is available in the literature [20].

To achieve this, we have developed a framework consisting of technology-agnostic core abstractions complemented by specific implementations tailored for Hyperledger Fabric. This approach ensures that our framework is both adaptable to various blockchain technologies

Table 5.1: Comparison with reviewed blockchain benchmarking frameworks. Focus on the first two steps: Blockchain Operations, Network Emulation.
(1): this is the supported blockchain technology in the reference implementation; however, the methodology is technology-agnostic.

		1. Blockchain Operations				2. Network Emulation		
Work	**Status**	**Supported Blockchain Technologies**	**Automated Blockchain Deployment**	**Isolation of nodes resources**	**Runs on a single machine**	**Networking**	**Realistic Network Parameters**	**Geo-Distribution of nodes**
[56]	Implemented	Ethereum, Fabric, Sawtooth	External script	No	Yes	Single host	No	No
[57]	Implemented	Virtually any blockchain	External script	No	No	LAN	Partial	Yes
[58]	Implemented	Fabric v2.0	Fabric only	Fabric only	No	WAN	Yes	Yes
[59]	Conceptualized	Fabric	Manual	No	Yes	Single host	Yes	Yes
Ours	**Implemented**	**Fabric**[1]	**Yes**	**Yes**	**Yes**	**Single host**	**Yes**	**Yes**

Table 5.2: Comparison with reviewed blockchain benchmarking frameworks. Focus on the final steps: Smart Contract Deployment, Benchmarking.

		3. Smart Contract Deployment	4. Benchmarking		
Work	**Status**	**Custom Smart Contract Deployment**	**Evaluation of Transaction Throughput**	**Evaluation of Transaction Latency**	**Execution of Parallel Workloads**
[56]	Implemented	No	Yes	Yes	No
[57]	Implemented	No	Yes	Yes	No
[58]	Implemented	Fabric only	Yes	Yes	No
[59]	Conceptualized	No	Yes	Yes	No
Ours	**Implemented**	**Yes**	**Yes**	**Yes**	**Yes**

and grounded in practical, case study-specific requirements.

5.4.1 Comparison with related works

When compared to existing benchmarking frameworks referenced in Table 5.1 and Table 5.2, it becomes evident that existing frameworks do not fully satisfy our specific requirements. To the best of our knowledge, our framework is unique in its capabilities, in that:

- It allows experiments to be conducted on a single machine and supports automatic configuration-driven deployment of production-grade blockchain nodes and automatic deployment of custom smart contracts;

- It lets the user set up a realistic network simulation according to the network parameters detailed in our methodology;
- It isolates the resources of these nodes to prevent load-balancing between CPU cores, thus mirroring real production environments;
- It can emulate multiple deployments of the same application, as is often the case in industrial contexts.

Finally, it allows benchmarks to be run against the smart contract transaction chosen according to the proposed methodology, making it suited to address the needs discussed in this thesis.

5.4.2 Extending feature estimation

Although we focused on ATL and ATT KPIs, our work allows us to estimate many additional features or simulate new scenarios, such as using a load balancer. Below, we present some meaningful examples.

Estimation of the percentage of transaction success rate under particular conditions. This parameter may be evaluated by leveraging the ratio between successful and unsuccessful transactions. It could reveal specific situations in which the transaction requests could not be satisfied.

Simulation of the presence of a load balancer with overload protection. This can be accomplished by configuring the benchmarking tool to send requests at a constant rate while maintaining a defined maximum backlog queue of unfinished transactions. This benchmark allows evaluating the need for implementing and sizing a load balancer system that is able to enqueue, monitor and eventually resubmit transaction requests to ensure robust operation when the application is run in a production environment that imposes a particular workload.

Simulation of varying loads. This can be accomplished by configuring the benchmarking tool to send requests at variable rates. This benchmark allows the evaluation of

the smart contract's response to changing workloads, an essential aspect when considering its application in dynamic industrial settings.

5.4.3 Application to further blockchain technologies

Expanding the proposed pilot framework to showcase the methodology for accommodating various blockchain platforms is possible.
To achieve this goal, the application's core needs to be adapted to facilitate the deployment of the intended blockchain nodes within a controlled environment.

These nodes must be set up to allow for the operation of a network emulation tool on the connections between the nodes and clients. The network emulation tool we chose works on Linux environments, making it compatible with all kinds of blockchain nodes that could be deployed as Docker containers built on Linux images. Additionally, selecting a benchmarking tool compatible with the specific target blockchain is essential.
Notably, the tool used in our benchmarking framework is open-source and designed for easy extension, allowing for the integration of plugins tailored to various blockchain environments.

Supplementary Research Projects 6

Throughout the course of my doctoral studies, I engaged in several side projects that, while not forming the core of this thesis, are closely related to the themes of blockchain technology and industrial applications. This chapter presents these supplementary research works, highlighting their relevance to the main subject of the thesis.

6.1 Smart Device for Shipping Monitoring

In collaboration with *TIM S.p.A.*, the company sponsoring my Ph.D. scholarship, I participated in the conceptualization of a smart device designed for monitoring the condition of goods during the shipping and delivery process. The primary goal of this project was to enhance the security and transparency of the shipping lifecycle by leveraging advanced technologies, including blockchain.

The proposed device is intended to be placed inside shipping containers or boxes, where it continuously monitors environmental conditions such as temperature, humidity, shock, and orientation, and estimates the damages of the contained items. Additionally, it is capable of communicating with other devices, such as wearable technology used by handlers and automated delivery boxes. This interconnectivity allows for real-time tracking and monitoring of the goods' status throughout the entire delivery chain.

Given that multiple stakeholders are involved in the shipping process - ranging from senders and carriers to recipients — and these parties may not fully trust each other, blockchain technology emerges as a suitable

solution for ensuring transparency and accountability. By recording each operation or state transition of the delivery lifecycle on a blockchain, we can provide an immutable and verifiable ledger of events that is accessible to all authorized participants.

In this project, my contributions included designing the blockchain-related workflows and specifying the device features required for seamless integration with blockchain transactions. This involved outlining how the device would interact with the blockchain network, the type of data to be recorded, and the protocols to be implemented. The work aligns with the main themes of this thesis by exploring the application of blockchain technology in an industrial context, specifically in the logistics and shipping sector, and addressing challenges related to data integrity, trust, and security.

As part of the project, I also participated in formalizing and drafting a patent application to protect the intellectual property associated with the device and its blockchain integration. Due to confidentiality agreements and the pending status of the patent, further technical details cannot be disclosed at the time of writing this thesis.

6.2 Enhancing Workplace Safety through Operator Area Networks

Workplace safety is a critical concern across various industries, prompting ongoing research into effective methods for preventing accidents and ensuring compliance with safety regulations. One prevalent issue is the improper use or removal of Personal Protective Equipment (PPE), which can lead to severe injuries or fatalities. To address this challenge, I contributed to the development of an *Operator Area Network (OAN)* system designed to monitor and enforce the correct usage of PPE in a non-intrusive and privacy-preserving manner.

The OAN system leverages wearable devices equipped with Bluetooth Low Energy (BLE) capabilities to create

a local network around the operator. By measuring the Received Signal Strength Indicator (RSSI) between the PPE devices worn by the operator, the system can infer whether the equipment is being worn correctly. The novelty of this work lies in its ability to provide robust PPE monitoring without compromising user comfort or privacy. By avoiding intrusive sensors or cameras, the system respects the operator's personal space while still delivering real-time safety enforcement. The implementation of the post-processing algorithm reduces false positives by approximately 80% and detecting improper PPE usage within seven seconds.

In this project, my primary role involved the design and implementation of the firmware for the wearable devices. I focused on implementing the BLE functionalities in order to estimate the relative proximity of the devices. I also contributed to developing the machine learning model which processes the RSSI data to detect anomalies indicative of incorrect PPE usage. This project intersects with the main subject of the thesis by presenting an industrial case study where technology is used to improve safety and operational efficiency. There is potential to enhance this system by incorporating blockchain technology to certify correct PPE usage and possibly implement incentive mechanisms, such as rewarding workers for compliance.

6.3 Device Authentication Using Physical Unclonable Functions (PUFs)

When devices are required to participate in sensitive operations such as blockchain transactions, then security becomes a concern in IoT networks. Ensuring that devices are uniquely identifiable and authenticated is critical for maintaining the integrity of the network. In this context, I explored the use of *Physical Unclonable*

Functions (PUFs) as a means to enhance device authentication mechanisms.

PUFs exploit the inherent manufacturing variations in electronic components to generate unique fingerprints for each device. These fingerprints can be used in challenge-response authentication schemes without the need to store secret keys in non-volatile memory, thereby reducing the risk of key extraction through physical attacks.

In this project, I developed a PUF-based authentication scheme on the *ESP32-WROOM* SoC, utilizing a portion of its internal SRAM as the inherent entropy source for the PUF. The process involved an enrollment phase, where the device's unique PUF response was recorded, followed by authentication phases where the device could be challenged and its responses verified against the enrolled data.

The successful implementation demonstrated that such devices could be authenticated securely and efficiently. However, a much better approach would be to generate unique cryptographic keys at runtime. This approach has significant security benefits with respect to storing keys in flash memories, as the keys are regenerated as needed, making them resistant to tampering and extraction. Moreover, with respect to challenge-response schemes, these keys could be used to implement all the most common PKI authentication schemes.

The relevance of this work to the main thesis lies in the potential application of PUFs for enhancing the security of devices participating in blockchain networks. By ensuring that devices have a secure and unique identity, they can safely generate and manage cryptographic keys required for blockchain transactions, such as producing *secp256k1 ECDSA* signatures used in Ethereum. This capability is essential for integrating IoT devices into blockchain applications, where trust and authentication are fundamental.

7 Conclusions

The integration of blockchain technology into industrial applications has introduced new ways to address critical challenges, such as ensuring data integrity, securing transactions, and enhancing transparency. Across various industries, from structural monitoring to logistics, public and private actors face the challenge of managing vast amounts of data while being constrained by limited storage capacity. By combining continuous monitoring systems with blockchain, we have developed a framework that reduces the burden on storage infrastructure while ensuring that critical data remains safeguarded.

In Chapter 3, we presented our proposed system architecture, which includes a Policy Smart Contract and a Policy Watchdog (PWD), to effectively manage data by setting retention policies that determine the relevance of stored information. Non-essential data is safely deleted, while critical data remains intact, ensuring the long-term sustainability of the database. We validated this approach using data from a real-world railway bridge monitoring system, demonstrating that the smart contract could distinguish between high-value data, such as data from train passages, and less relevant information. This initial validation under ideal conditions confirmed that our system works and addresses the use-case problem, showcasing its potential benefits and feasibility.

However, as we considered the practical deployment of this system in real-world settings, we recognized that more investigation was necessary. The complexities introduced by production-grade deployments—such as the choice of blockchain technology, the degree of network decentralization, network speed and latency, blockchain topology, and other operational factors—could significantly impact the performance and applicability of our smart contract solution. Testing in ideal conditions does not capture these real-world variables, which are critical

for understanding the true capabilities and limitations of the system.

This realization led us to delve deeper into the benchmarking of production-grade deployments of smart contracts to serve industrial applications. In Chapter 4, we developed a comprehensive four-step methodology to assess smart contract performance under realistic conditions. This methodology includes:

- *Automated Blockchain Deployment* - Setting up the blockchain network with production-grade configurations in an automated and repeatable manner.
- *Emulation of Real-World Network Conditions* - Simulating network latency, jitter, throughput, and packet loss to mimic real-world network environments.
- *Smart Contract Deployment* - Deploying the smart contract onto the configured blockchain network, ensuring it operates under realistic conditions.
- *Performance Benchmarking* - Measuring key performance indicators such as Average Transaction Latency (ATL) and Average Transaction Throughput (ATT) to evaluate the smart contract's performance.

By systematically following the methodology, we can accurately measure the impact of real-world variables on smart contract execution, enabling us to configure test deployments that mimic actual operating environments.

Finally, in Chapter 5, we applied this methodology to perform a proper evaluation of the smart contract developed earlier. By testing the smart contract under production-grade conditions, we were able to optimize its performance to meet the stringent demands of the Structural Health Monitoring (SHM) use case. Through fine-tuning of the smart contract parameters and blockchain configurations, we reduced Average Transaction Latency (ATL) while ensuring that the Average Transaction Throughput (ATT) remained sufficient for the application's needs. This validation highlights the adaptability of our system in managing multiple concurrent

applications and emphasizes the importance of performance optimization in real-world environments.

Moreover, we introduced the concept of the Blockchain Elaboration Ratio (BER), which provides a means of understanding the system's performance in terms of data processing and deletion. By conducting simulations and analyzing the blockchain size, we demonstrated that with well-configured policies, our system can significantly reduce storage requirements, up to 75% in some cases, while maintaining data security. For example, in a scenario where the Policy Gateway (PGW) operates with a 1-minute period and the Policy Watchdog (PWD) with a 3-minute period, our system could shrink the blockchain's transaction data size to 8 GiB over a year, compared to the 1.7 TiB required without data deletion.

A major takeaway from this research work is acknowledging the importance of not only developing innovative blockchain-based solutions for industrial applications, but also thoroughly evaluating their performance under realistic conditions. The progression from initial development and feasibility testing to comprehensive performance benchmarking ensures that such systems are truly ready for deployment in real-world settings.

Future work can delve deeper into exploring more advanced policy algorithms for data management in safety-critical systems, such as redundant data removal or autonomous policies that evolve over time. Additionally, for benchmarking production-grade smart contracts, expanding the methodology to include more Key Performance Indicators (KPIs) like transaction success rate or simulating variable loads with more complex algorithms can provide deeper insights. The proposed methodology and the related benchmarking framework, demonstrated using Hyperledger Fabric, have significant potential to be extended to support more blockchain platforms, serving as blueprints for future research.

Bibliography

[1] Alexandros Bousdekis et al. 'A Review of Data-Driven Decision-Making Methods for Industry 4.0 Maintenance Applications'. In: *Electronics* 10.7 (Mar. 2021), p. 828. DOI: 10.3390/electronics10070828 (cited on page 6).

[2] Clive Lafferty. 'Sustainable Industry 4.0: Product Decision-Making Information Systems, Data-driven Innovation, and Smart Industrial Value Creation'. In: *Journal of Self-Governance and Management Economics* 7.2 (2019), pp. 19–24 (cited on page 6).

[3] C. Farrar and et al. *Structural health monitoring: a machine learning perspective*. John Wiley & Sons, 2012 (cited on page 7).

[4] J. Lynch and et al. 'Structural health monitoring: technological advances to practical implementations'. In: *Proceedings of the IEEE* 104.8 (2016), pp. 1508–1512 (cited on page 7).

[5] H. Li and et al. 'Reviews on innovations and applications in structural health monitoring for infrastructures'. In: *Structural Monitoring and Maintenance* 1.1 (2014), pp. 1–45 (cited on pages 7, 8).

[6] Zonzini F. and et al. 'Structural Health Monitoring and Prognostic of Industrial Plants and Civil Structures: A Sensor to Cloud Architecture'. In: *IEEE Instrumentation Measurement Magazine* (2020) (cited on pages 7, 15).

[7] A. Sabato and et al. 'Wireless MEMS-based accelerometer sensor boards for structural vibration monitoring: a review'. In: *IEEE Sensors Journal* 17.2 (2016), pp. 226–235 (cited on page 8).

[8] Z. Tang and et al. 'Convolutional neural network-based data anomaly detection method using multiple information for structural health monitoring'. In: *Structural Control and Health Monitoring* 26.1 (2019), e2296 (cited on page 8).

[9] S. Jeong and et al. 'Sensor data reconstruction and anomaly detection using bidirectional recurrent neural network'. In: *Sensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems 2019*. Vol. 10970. 2019, pp. 157–167 (cited on page 8).

[10] M. Reyer and et al. 'Design of a wireless sensor network for structural health monitoring of bridges'. In: *2011 Fifth International Conference on Sensing Technology*. IEEE. 2011, pp. 515–520 (cited on page 8).

[11] Imran Bashir. *Mastering blockchain*. Packt Publishing Ltd, 2017 (cited on page 9).

[12] J. Bonneau and et al. 'Sok: Research perspectives and challenges for bitcoin and cryptocurrencies'. In: *2015 IEEE symposium on security and privacy*. IEEE. 2015 (cited on page 9).

[13] M. Bartoletti and et al. 'A journey into bitcoin metadata'. In: *Journal of Grid Computing* 17.1 (2019), pp. 3–22 (cited on page 9).

[14] Nick Szabo. 'Formalizing and securing relationships on public networks'. In: *First monday* (1997) (cited on page 9).

[15] V. Buterin and et al. *Ethereum: A next-generation smart contract and decentralized application platform*. 2014 (cited on page 9).

[16] A. Gervais and et al. 'On the security and performance of proof of work blockchains'. In: *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 2016, pp. 3–16 (cited on page 10).

[17] E. Androulaki and et al. 'Hyperledger fabric: a distributed operating system for permissioned blockchains'. In: *Proceedings of the thirteenth EuroSys conference*. 2018 (cited on pages 10, 12).

[18] M. Castro and et al. 'Practical byzantine fault tolerance'. In: *OsDI*. Vol. 99. 1999, pp. 173–186 (cited on page 10).

[19] Tobias Guggenberger et al. 'An in-depth investigation of the performance characteristics of Hyperledger Fabric'. In: *Computers & Industrial Engineering* 173 (2022), p. 108716. DOI: https://doi.org/10.1016/j.cie.2022.108716 (cited on page 12).

[20] Nicola Elia et al. 'Smart Contracts for Certified and Sustainable Safety-Critical Continuous Monitoring Applications'. In: *Advances in Databases and Information Systems*. Springer, Aug. 2022, pp. 377–391. DOI: 10.1007/978-3-031-15740-0_27 (cited on pages 14, 38, 66).

[21] A. Borghesi and et al. 'ExaMon-X: a Predictive Maintenance Framework for Automatic Monitoring in Industrial IoT Systems'. In: *IEEE Internet of Things Journal* (2021) (cited on pages 17, 21).

[22] C. Aguzzi and et al. 'MODRON: A Scalable and Interoperable Web of Things Platform for Structural Health Monitoring'. In: *2021 IEEE 18th Annual Consumer Communications Networking Conference (CCNC)*. 2021 (cited on pages 17, 19).

[23] L. Gigli and et al. 'Blockchain and Web of Things for Structural Health Monitoring Applications: A Proof of Concept'. In: *2022 IEEE 19th Annual Consumer Communications Networking Conference (CCNC)*. 2022 (cited on page 17).

[24] Md Ashraf Uddin et al. 'A survey on the adoption of blockchain in IoT: challenges and solutions'. In: *Blockchain: Research and Applications* 2.2 (2021), p. 100006 (cited on page 18).

[25] Endale Mitiku Adere. 'Blockchain in healthcare and IoT: A systematic literature review'. In: *Array* 14 (2022), p. 100139 (cited on page 18).

[26] K. Christidis and et al. 'Blockchains and smart contracts for the internet of things'. In: *Ieee Access* 4 (2016), pp. 2292–2303 (cited on page 18).

[27] A. Reyna and et al. 'On blockchain and its integration with IoT. Challenges and opportunities'. In: *Future generation computer systems* 88 (2018), pp. 173–190 (cited on page 18).

[28] H. Dai and et al. 'Blockchain for Internet of Things: A survey'. In: *IEEE Internet of Things Journal* 6.5 (2019), pp. 8076–8094 (cited on page 18).

[29] K. Griggs and et al. 'Healthcare blockchain system using smart contracts for secure automated remote patient monitoring'. In: *Journal of medical systems* 42.7 (2018) (cited on page 18).

[30] S. He and et al. 'BoSMoS: A blockchain-based status monitoring system for defending against unauthorized software updating in industrial Internet of Things'. In: *IEEE Internet of Things Journal* 7.2 (2019), pp. 948–959 (cited on page 18).

[31] K. Košťál and et al. 'Management and monitoring of IoT devices using blockchain'. In: *Sensors* 19.4 (2019), p. 856 (cited on page 18).

[32] P. Helebrandt and et al. 'Blockchain adoption for monitoring and management of enterprise networks'. In: *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*. IEEE. 2018, pp. 1221–1225 (cited on page 18).

[33] Michail Sidorov et al. 'A Public Blockchain-Enabled Wireless LoRa Sensor Node for Easy Continuous Unattended Health Monitoring of Bolted Joints: Implementation and Evaluation'. In: *IEEE Sensors Journal* 20.21 (2020), pp. 13057–13065 (cited on page 19).

[34] Qiheng Zhou et al. 'Solutions to Scalability of Blockchain: A Survey'. In: *IEEE Access* 8 (Jan. 2020), pp. 16440–16455. DOI: 10.1109/ACCESS.2020.2967218 (cited on page 19).

[35] Turki Ali Alghamdi, Rabiya Khalid, and Nadeem Javaid. 'A Survey of Blockchain Based Systems: Scalability Issues and Solutions, Applications and Future Challenges'. In: *IEEE Access* (May 2024). DOI: 10.1109/ACCESS.2024.3408868 (cited on page 19).

[36] Nicola Elia et al. 'Estimating Smart Contracts Performance'. In: *6th Distributed Ledger Technologies Workshop, DLT2024. Oral communications*. 2024 (cited on page 36).

[37] Bhabendu Kumar Mohanta et al. 'An Overview of Smart Contract and Use Cases in Blockchain Technology'. In: *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. IEEE, 2018, pp. 10–12. DOI: 10.1109/ICCCNT.2018.8494045 (cited on page 37).

[38] Randa Kamal et al. 'Care4U: Integrated healthcare systems based on blockchain'. In: *Blockchain: Research and Applications* 4.4 (Dec. 2023), p. 100151. DOI: 10.1016/j.bcra.2023.100151 (cited on page 38).

[39] McSeth Antwi et al. 'The case of HyperLedger Fabric as a blockchain solution for healthcare applications'. In: *Blockchain: Research and Applications* 2.1 (Mar. 2021), p. 100012. DOI: 10.1016/j.bcra.2021.100012 (cited on page 38).

[40] Mohammed Zia. 'B-DRIVE: A blockchain based distributed IoT network for smart urban transportation'. In: *Blockchain: Research and Applications* 2.4 (Dec. 2021), p. 100033. DOI: 10.1016/j.bcra.2021.100033 (cited on page 38).

[41] Iakovos Pittaras et al. 'Secure smart contract-based digital twins for the Internet of Things'. In: *Blockchain: Research and Applications* (Nov. 2023), p. 100168. DOI: 10.1016/j.bcra.2023.100168 (cited on page 38).

[42] Nikos Fotiou et al. 'A privacy-preserving statistics marketplace using local differential privacy and blockchain: An application to smart-grid measurements sharing'. In: *Blockchain: Research and Applications* 2.1 (Mar. 2021), p. 100022. DOI: 10.1016/j.bcra.2021.100022 (cited on page 38).

[43] Vittorio Capocasale et al. 'A Blockchain, 5G and IoT-based transaction management system for Smart Logistics: an Hyperledger framework'. In: *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*. 2021, pp. 1285–1290. DOI: 10.1109/COMPSAC51774.2021.00179 (cited on page 38).

[44] Lorenzo Gigli et al. 'Blockchain and Web of Things for Structural Health Monitoring Applications: A Proof of Concept'. In: *2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 2022, pp. 08–11. DOI: 10.1109/CCNC49033.2022.9700679 (cited on page 38).

[45] Sin Kuang Lo et al. 'Evaluating Suitability of Applying Blockchain'. In: *2017 22nd International Conference on Engineering of Complex Computer Systems (ICECCS)*. IEEE, pp. 05–08. DOI: 10.1109/ICECCS.2017.26 (cited on page 38).

[46] Rifat Sonmez et al. 'Blockchain in project management: a systematic review of use cases and a design decision framework'. In: *Journal of Ambient Intelligence and Humanized Computing* 14.7 (July 2023), pp. 8433–8447. DOI: 10.1007/s12652-021-03610-1 (cited on page 38).

[47] Catherine Mulligan et al. *Blockchain Beyond the Hype*. [Online; accessed 8. Mar. 2024]. Mar. 2018. URL: https://www.weforum.org/publications/blockchain-beyond-the-hype (cited on page 38).

[48] Marko Vukolić. 'The Quest for Scalable Blockchain Fabric: Proof-of-Work vs. BFT Replication'. In: *Open Problems in Network Security*. Springer, May 2016, pp. 112–125. DOI: 10.1007/978-3-319-39028-4_9 (cited on page 39).

[49] Giampaolo Bovenzi et al. 'Blockchain Performance in Industry 4.0: Drivers, use cases, and future directions'. In: *Journal of Industrial Information Integration* 36 (Dec. 2023), p. 100513. DOI: 10.1016/j.jii.2023.100513 (cited on pages 40, 44).

[50] Kaushal Shah et al. 'Exploring applications of blockchain technology for Industry 4.0'. In: *Mater. Today:. Proc.* 62 (Jan. 2022), pp. 7238–7242. DOI: 10.1016/j.matpr.2022.03.681 (cited on page 40).

[51] Tiago M. Fernández-Caramés and Paula Fraga-Lamas. 'A Review on the Application of Blockchain to the Next Generation of Cybersecure Industry 4.0 Smart Factories'. In: *IEEE Access* 7 (Mar. 2019), pp. 45201–45218. DOI: 10.1109/ACCESS.2019.2908780 (cited on page 40).

[52] Murat Kuzlu et al. 'Performance Analysis of a Hyperledger Fabric Blockchain Framework: Throughput, Latency and Scalability'. In: *2019 IEEE International Conference on Blockchain (Blockchain)*. IEEE, July 2019, pp. 536–540. DOI: 10.1109/Blockchain.2019.00003 (cited on pages 41, 42).

[53] Tobias Guggenberger et al. 'An in-depth investigation of the performance characteristics of Hyperledger Fabric'. In: *Computers & Industrial Engineering* 173 (Nov. 2022), p. 108716. DOI: 10.1016/j.cie.2022.108716 (cited on pages 41, 42).

[54] Vittorio Capocasale et al. 'Comparative analysis of permissioned blockchain frameworks for industrial applications'. In: *Blockchain: Research and Applications* 4.1 (Mar. 2023), p. 100113. DOI: 10.1016/j.bcra.2022.100113 (cited on pages 41, 42).

[55] Canhui Wang et al. *Performance Characterization and Bottleneck Analysis of Hyperledger Fabric*. IEEE Computer Society, Nov. 2020 (cited on pages 41, 42).

[56] Mohammadreza Rasolroveicy et al. 'BlockCompass: A benchmarking platform for blockchain performance'. In: *figshare* (Mar. 2023). DOI: 10.36227/techrxiv.22299634.v1 (cited on pages 42, 43, 67).

[57] Bulat Nasrulin et al. *Gromit: Benchmarking the Performance and Scalability of Blockchain Systems*. IEEE Computer Society, Aug. 2022 (cited on pages 42, 43, 67).

[58] Ayham Kassab et al. 'C2B2: a Cloud-native Chaos Benchmarking suite for the Hyperledger Fabric Blockchain'. In: *2022 18th European Dependable Computing Conference (EDCC)*. IEEE, Sept. 2022, pp. 89–96. DOI: 10.1109/EDCC57035.2022.00024 (cited on pages 42, 43, 67).

[59] Haochen Pan et al. 'BBB: A Lightweight Approach to Evaluate Private Blockchains in Clouds'. In: *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*. IEEE, Dec. 2020, pp. 1–6. DOI: 10.1109/GLOBECOM42002.2020.9322354 (cited on pages 42, 43, 67).

[60] Marios Touloupou et al. 'A Systematic Literature Review Toward a Blockchain Benchmarking Framework'. In: *IEEE Access* 10 (July 2022), pp. 70630–70644. DOI: 10.1109/ACCESS.2022.3188123 (cited on page 43).

[61] Caixiang Fan et al. 'Performance Evaluation of Blockchain Systems: A Systematic Survey'. In: *IEEE Access* 8 (2020), pp. 126927–126950. DOI: 10.1109/ACCESS.2020.3006078 (cited on page 44).

[62] Satoshi Nakamoto. 'Bitcoin: A peer-to-peer electronic cash system'. In: *Decentralized business review* (2008) (cited on page 48).

[63] Vitalik Buterin et al. 'A next-generation smart contract and decentralized application platform'. In: *white paper* 3.37 (2014), pp. 2–1 (cited on page 48).

[64] Martin L Shooman. *Reliability of computer systems and networks: fault tolerance, analysis, and design*. John Wiley & Sons, 2003 (cited on page 48).

[65] Elena Dubrova. *Fault-Tolerant Design*. New York, NY, USA: Springer, 2013 (cited on page 48).

[66] Daniel Ford et al. 'Availability in globally distributed storage systems'. In: *OSDI'10: Proceedings of the 9th USENIX conference on Operating systems design and implementation*. USENIX Association, Oct. 2010, pp. 61–74. DOI: 10.5555/1924943.1924948 (cited on page 48).

[67] Abdelfatah A Tamimi, Raneem Dawood, and Lana Sadaqa. 'Disaster Recovery Techniques in Cloud Computing'. In: *2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT)*. 2019, pp. 845–850. DOI: 10.1109/JEEIT.2019.8717450 (cited on page 48).

[68] *fabric-benchmarking*. June 2023. URL: https://anonymous.4open.science/r/fabric-benchmarking-C7A5 (cited on page 54).

[69] *Cloud Performance Report: 2022 Edition*. Tech. rep. ThousandEyes, Cisco Systems Inc., 2022 (cited on page 57).

[70] *2022 Cloud Report*. Tech. rep. Cockroach Labs', 2022 (cited on page 58).