Revista de Informática Teórica e Aplicada - RITA - ISSN 2175-2745
Vol. 33, Num. 1 (2026) 99-110

RESEARCH ARTICLE

Improving the Resilience of Smart City Systems During Network Outage

Incremento da Resiliência dos Sitemas de Cidades Inteligentes Durante a Indisponibilidade de Rede

Leidmar Magnus Festa[1]*, Daniel Fernando Pigatto[1], Luiz Celso Gomes-Jr[1]

Abstract: This work presents an architecture that allows Information Systems to function even in moments of network disconnection, using a Healthcare Unit (HCU, from the Portuguese Unidade Municipal de Saúde - UMS) as a use case. The architecture uses a blockchain as a distributed database and one of its nodes *(proxy node)* is physically located inside an HCU, connected to its Local Area Network (LAN). During moments of disconnection between the LAN and the Internet, the *proxy node* continues handling query requests made by authenticated users on the LAN, functioning as a fog layer. It is also possible to generate new data and, with the help of a reliable oracle, write it to a local database. In these disconnection periods, the architecture explores the blockchain node's behavior and an oracle's functionality to record newly generated data. This approach makes it possible to guarantee the authenticity of the queried data. It is also possible to aggregate enough information about the new generated data to guarantee its integrity. Moreover, this information ensures the traceability and auditability of this data. An architecture prototype was implemented and used for execution tests. The test results show that the proposed architecture can maintain the information system working properly during a network outage. Therefore, the architecture increases the system resilience, being its application suitable for a smart city scenario, where data availability is desired.
Keywords: Blockchain — Fog Computing — Off-chain data — Oracle — Smart City

Resumo: Este artigo apresenta uma arquitetura que mantém o funcionamento dos sistemas de informação de uma cidade inteligente mesmo em períodos de indisponibilidade da Internet, utilizando como caso de uso uma Unidade Municipal de Saúde (UMS). A arquitetura utiliza uma blockchain como banco de dados distribuído e um de seus nós *(proxy node)* está fisicamente localizado dentro da UMS e conectado a uma rede local (LAN). Nos momentos de desconexão entre a LAN e a Internet, o *proxy node* continua respondendo às requisições feitas pelos usuários autenticados em sua LAN, funcionando como uma camada de névoa. Também é possível gerar novos dados e, com a ajuda de um oráculo confiável, escrever estes um banco de dados local. Nos momentos de desconexão a arquitetura explora o comportamento de um nó da blockchain, bem como as funcionalidades de um oráculo para gravar novos os dados gerados. Desta forma é possível agregar informações sobre estes novos dados e garantir a sua autenticidade, integridade, rastreabilidade e auditabilidade. Um protótipo da arquitetura foi implementado e utilizado na execução dos testes. Os resutados destes mostram que a arquitetura proposta consegue manter um sistema de informação funcionando mesmo durante a indisponibilidade da Internet. Portanto a arquitetura aumenta a resiliência do sistema, sendo sua aplicação adequada para um cenário de cidade inteligente, onde a disponibilidade de dados é desejada.
Palavras-Chave: Blockchain — Computação em Névoa — Dados fora da cadeia — Cidade Inteligente

[1] *Programa de Pós-graduação em Computação Aplicada (PPGCA), Universidade Tecnológica Federal do Paraná (UTFPR), Brasil*
***Corresponding author**: leidmar@gmail.com
DOI: http://dx.doi.org/10.22456/2175-2745.150232 • **Received:** 15/09/2025 • **Accepted:** 26/12/2025

1. Introduction

Information Systems (IS) resilience is important in several scenarios of smart cities. In these cities, the demand for edge computing resources and the need to keep applications running even in periods of network outage are growing. These two needs are added to the volume, types of data, and speeds with which they are produced, as well as the maintenance of services provided to citizens. In this context, Information

Systems resilience becomes relevant and can be achieved through applications that use modern technologies such as blockchain, fog computing, and cloud computing.

Features such as availability, reliability, and immutability of data, as well as sharing services and computing resources are relevant in the context of smart cities, which use ICT (Information and Communication Technologies) to achieve their goals of developing sustainable urban environments and improving the citizen's life quality [1]. In scenarios that follow this pattern, the occurrence of a catastrophic event can cause the temporary disconnection of some network segments in a smart city. Disconnection is a time period in which the network segment does not have access to the Internet. In this situation, the IS of the entities located inside a disconnected segment may stop working and interrupt public services in parts of the city, causing inconvenience to the population [2].

This work has adopted the Healthcare Unities (HCU) as use case and describes an architecture (sec. 3) that allows an IS to work even during a network outage. Here the blockchain is used as a distributed database, which has a node called *Proxy Node (Pn)* physically located inside an HCU and connected to its Local Area Network (LAN). When a network disconnection occurs, authenticated users connected to the LAN can continue to consult blockchain data through *Pn*. To perform the inclusion of new records, a reliable oracle module is used, which allows to encrypt and to include new records in a local database. Later, these records will be checked and made effective on the blockchain. The system uses a fog layer that communicates with users and blockchain in its normal operation. During periods of outage, the *Pn* node behaves as a temporary fog layer to keep users connected to basic services within the LAN.

In the tests performed (sec. 4), a Hyperledger Fabric blockchain network was used as a distributed database. The network had four nodes that, initially, were all available. Later, network unavailability was simulated and three of these nodes were disabled, leaving only the proxy node working normally. The architecture proved to be useful to create a resilient virtual environment, and its application is suitable for the scenario of smart cities, where data availability is desired.

2. Basic concepts and related work

2.1 Blockchain

Blockchain is a data structure conceived and implemented by Satoshi Nakamoto to solve the double payment problem in a virtual currency without the need for the intermediary of a reliable regulatory institution. It is decentrally managed in a peer-to-peer network [3]. To implement the blockchain, several existing concepts were used, with an emphasis on cryptography, Finite State Machine (FSM), and hash. In cryptography, codes and ciphers are used to hide the message's content [4]. Modern cryptography uses codes called *keys* to encrypt and decrypt messages, making it impossible to reverse their encryption without knowing the key [5]. The two main ways to use the keys are *symmetric* and *asymmetric*. In the *symmetric* form there is a single key that is shared by the sender and receiver. This key is used to encrypt the message on sending and decrypt on receipt [6]. In the *asymmetric* form there are *public* and *private* keys. Here, encryption can be used with the purpose of *maintaining the privacy of the message*. For this, the sender encrypts its message using the *public key* disclosed by the receiver. In this situation, the receiver can decrypt the message using his *private key*. Another objective, central to blockchain's operation, is to use the keys to create a *digital signature*, in which a user (the sender) signs a transaction with their *private key*. Thus, any receiver who has the public key can decrypt the message and confirm the sender's signature[5]. These signatures provide the sender's non-repudiation, since he cannot deny the message's signature, as well as the guarantee of the message's authenticity to the receiver, according to [7, 5].

The FSM is a model that can be used in the representation of computational systems, which has a finite number of states [8]. In blockchain, the distributed ledger can be seen as an FSM, where each valid transaction causes a transition from the current state to a new state.

Finally, hash algorithms map large and variable-size data to small and fixed-size data. The result is a summary; thus, it does not reveal the full original data content [5]. Hashing is critical for linking blocks together—as each block contains the hash of the previous one—creating an immutable chain.

These algorithms increase the security in the transmission of messages and ensure their integrity since the information received is exactly what was sent. Regarding the participation of nodes in the network, a *blockchain* can be *permissionless* (public) or *permissioned* (private or consortium). This classification is sometimes expanded to consider data readability (public vs. private) and write permissions (permissionless vs. permissioned) as two independent axes. On a permissionless *blockchain* any node can join, access all recorded transaction data, and act as a miner. Two examples are the *Bitcoin* and *Ethereum* networks [9, 10]. In permissioned networks, only authorized nodes can join. Authorizations are granted through identities and the *blockchain* has access control tools to maintain the confidentiality of transaction data [10]. The distinction between private and consortium networks is related to the form of governance. A private network is managed by a single organization, whereas a consortium network is managed by a group of organizations. [11, 12].

The main blockchain concept is related to a chain of data blocks known as a *distributed ledger*, which can be used to record transaction data, known as assets [13]. In addition to the data, each block keeps the timestamp of its creation, the block number, and the hash of the previous block – except for the first block, called *genesis*, which does not have a parent block and is common to all network members. The grouping of the timestamp, block number, and hash is called a *header*. By storing the hash in the header, a link is established between the blocks, which creates a chain with immutable records. This immutability occurs because if any data from

previous blocks is changed the hash is also changed and thus invalidates the block with conflicting information. With all this, in a public blockchain, any member of the network have access to the list of blocks and is able to verify the status of the data and transactions carried out [14].

To decide whether a transaction will be included in a block, members of the peer-to-peer network, called miners, participate in a consensus process. Within this process, miners receive several transactions and verify that they are valid. Invalid transactions are discarded, and all valid transactions are aggregated into a block. This block is sent to all miners and, if accepted, becomes part of the blockchain [15].

One of the most well-known blockchain networks is *Bitcoin*, which is used exclusively for transferring financial assets between user accounts [3]. However, there are other networks such as Ethereum that can be used for processing various types of assets. This network uses smart contracts (or chaincodes), which are programs stored and executed by the distributed ledger. The output of a smart contract proves the completion of a transaction and remains stored in the ledger [16].

2.1.1 Reliable Oracles in Blockchain

Smart contracts do not have access to data external to the blockchain (off-chain data). This can be restrictive for some types of applications that use data related to asset prices, and identity verification, among others [17]. Furthermore, according to [18], one of the challenges in the context of blockchain networks is the storage limitation. This is often overcomed by storing large sets of raw data in off-chain databases and keeping only metadata on the blockchain, as in [19, 20, 21]. In contexts like these, an oracle is a component that allows the interaction of smart contracts with off-chain data securely and reliably [20].

According to [22, 23], it is possible to classify an oracle based on four characteristics: *trust model* that can be centralized or decentralized; *data source* that describes the data origin (software, hardware, or humans); *interaction* indicates the direction in which information flows in relation to the blockchain, being inbound or outbound; *design pattern* describes the data availability model and can be request-response (for off-chain data requested in small pieces), publish-subscribe (for volatile data) or immediate read (for momentary data).

2.1.2 *Hyperledger Fabric*

The *Hyperledger*[1] was started by *Linux Foundation* in 2015. It is a set of *open source* projects, whose focus is on creating an ecosystem based on *blockchain* that encompasses many types of industries. There are currently six stable projects and nine more in incubation. The *Hyperledger Fabric* is one of the stable projects, being a platform for creating permissioned *blockchain* networks [24, 25]. For a node to be able to join a *Fabric* network, it must receive an identity provided and managed by the *Membership Service Provider (MSP)*. If the node does not have the identity, it will be ignored by the entire network [26]. Another detail about *Fabric* networks is that they work by executing *smart contracts* (or *chaincodes*), which are programs that define the business rules and carry out manipulation tasks for network assets [10]. A task can be recording or querying data on the network, recording social interactions, or recording purchases and sales of goods, among others [13]. *Fabric* was the first *blockchain* platform to accept *chaincodes* written in general-purpose programming languages and without the implementation of a virtual currency native to the network [27].

A *Fabric blockchain* network is formed by a consortium of organizations that carry out transactions with each other. Every organization has a set of nodes that perform one or more functions [10, 27], among them the role of *Certificate Authority (CA)*, which is responsible to delivers identities to nodes that requested authorization to join the network. By default, every organization participating in a *Fabric* blockchain has a CA and the set of all CAs in the network forms the MSP [28]. Another set of nodes is called *peers* and is in charge of maintaining the distributed ledger, executing the transactions proposals, and validate these transactions. This last task is performed by some of the *peers* called *endorsers*, which verify that the transactions meet the requirements, including the signature of the applicant, and then generate the endorsement of each transaction [27]. There are also *orderers* nodes, which receive the proposed transactions sent to the network and determine the general execution order of all transactions at *Fabric* [29, 9]. By default, each network organization has a node of this type, and the set of all *orderers* forms the *Orderer Service* (OS). some nodes are called *clients* and perform tasks for end users. Among these tasks are the transmission of proposed transactions for *orderers* and communication with *peers* [9].

Normally a transaction on a *Fabric* network follows a basic flow in which a *client* node sends a transaction proposal *W* to *peers* by invoking a *chaincode*. Each *peer* that executes the *chaincode* produces a result called *endorsement*, which is appended to the transaction *W* and submitted to OS. Then the OS bundles the *W* transaction, and others sent to the network, in a block that is transmitted to all *peers*. These validate the transactions and record the new block in the ledger [30]. In this described flow, all data involved are public and available to the entire consortium. This raises privacy issues, especially when transaction data is sensitive and should be visible to only a few organizations, maintaining secrecy between the other [28]. For this situation, it is possible to use the *Private Data Collection* (PDC) feature in *Fabric*. With PDC the *hash* of transactions is included in the ledger, but only authorized organizations have access to the data of these transactions [31, 10].

2.2 Fog Computing

Fog computing was originated by Cisco and is between the edge layer, where the devices are, and the cloud. The main fog function is to extend the cloud functionalities, bringing it closer to the end user [32]. Fog services have a high vir-

[1] ⟨https://www.hyperledger.org⟩

tualization degree, low latency, storage capacity, real-time interactions, heterogeneity, and geographically distributed devices, among other features [33].

In this way, a fog computing layer can handle a large volume and variety of data, also following the speed at which this data is generated by Internet of Things (IoT) devices. In this regard, it manages to reduce the deficiencies found in cloud computing to handle this type of situation, quickly meet user demand [34, 35] and decrease user response latency [36, 37].

2.3 Related Work

Related works were classified according to the way in which they increase resilience in smart cities: (i) general solutions for resilience, (ii) resilience to fog, and (iii) blockchain oracles.

Regarding (i) general solutions for resilience, the related works cover smart grids for energy distribution, analyzing opportunities, threats and proposing solutions that use machine learning, IoT and blockchain to manage the energy network, or even to trade excess available energy [38, 39, 40, 41]. Furthermore, [42, 43] lead some research focusing on blockchain and smart energy grids. A theoretical model that brings more efficiency to public administration was presented by [44] and the practical and theoretical parameters for the mitigation of urban risks were approached by [45].

In (ii) resilience to fog, we highlight related research akin to [46, 47], where the Challenged Networks problem is addressed. In these networks, a device may remain disconnected from the fog layer for a long time and needs to use a replication strategy for its data until it is delivered to the fog. Distributed frameworks are proposed by [48] and [49]. The first replicates data from a faulty node to its replacement in the fog structure. The second performs shared processing between fog nodes using load balancing.

Finally, regarding (iii) blockchain oracles, there are distributed oracle projects as [20, 50] that have source verification modules and consensus among their nodes. Authors in [51] present an oracle that performs the search for data through HTTPS, while the authors in [52] describe *Aeternity*[2] a blockchain platform with native oracles that are used in the consensus process. Finally, [53] describe *Chainlink*[3] as a computational service provider to smart contracts. Table 1 provides a summary of the related works and highlights the novelty of our proposal. While existing research addresses resilience in specific domains like smart grids [38] or fault tolerance in fog computing [48], our work presents a unique synthesis of technologies aimed at ensuring the complete operational continuity of an information system at the local level during a total network outage. Unlike solutions for challenged networks [46], which focus on eventual data delivery, our architecture provides immediate read access to historical data and secure write capabilities for new data through a Proxy Node that combines a blockchain node, a fog layer, and a trusted oracle. The use of an oracle to digitally sign and temporarily store new off-chain records locally is a key differentiator, providing a robust mechanism for data integrity and auditability that is not explicitly addressed by other approaches in the context of network isolation.

[2] ⟨https://aeternity.com/⟩
[3] ⟨https://chain.link/⟩

3. Increasing resiliency for smart cities' information systems

3.1 Use case

To understand the proposed architecture better, we chose the use case of a general Healthcare Unit (HCU). The HCU operates a Digital Vaccination Card (DVC) application that must remain functional, even during a network outage – when the HCU network remains without Internet and loses communication with external nodes to its Local Area Network (LAN). Keeping the DVC operational in this scenario mitigates errors in the subsequent transcription of data into the system, as well as the application of duplicate vaccine doses or even fraud in the HCU´s stock.

The DVC is a web application that sends requests to a fog layer which, in turn, executes them using a blockchain as a distributed database. The DVC data relationship diagram is shown in figure 1. There are represented, on the left side, the *HcuRegister* which maintains the HCUs data, the *VaccinesTypes* which lists the existing types of vaccines, and the *CitizenRegister* which has the personal citizens data. On the right side are *VaccineStock* which maintains the record of vaccine stocks at the HCUs and *TakenVaccines* which lists the vaccines already applied.

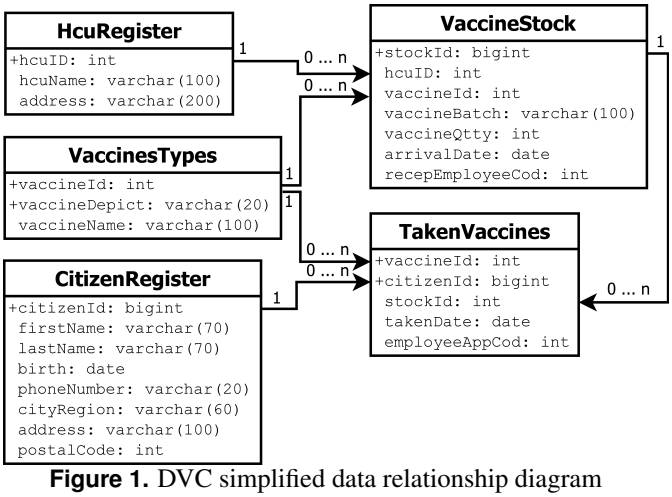

Figure 1. DVC simplified data relationship diagram

The requirements to keep the DVC functioning in the disconnection moment are: (a) maintain access to citizens' registration data, vaccines available at HCU, and vaccines already applied to each citizen; (b) keep the registration of new vaccines applied to each citizen and the updates of citizens data; (c) store the new records made during the disconnection period for later inclusion in the blockchain; (d) tracking/auditing

Table 1. Comparison of Related Works with the Proposed Architecture

Work	Resilience Approach	Technology Used	Offline Capability	Data Integrity Method
[38, 40, 39, 41, 42, 43]	Resilience in Smart Grids and Energy Systems	IoT, Machine Learning, Blockchain	Not the primary focus	Blockchain-based / ML-based Security
[44, 45]	Urban Hazard Mitigation / Resilient City Models	Theoretical Models, Policy	Contextual (Policy-based)	Theoretical Integrity / Governance
[46, 47]	Data replication for challenged networks	Replication strategies	Yes, focuses on eventual delivery for disconnected devices	Assumes eventual delivery
[48]	Fault tolerance in fog nodes	Data replication among fog nodes	No, focuses on node failure/replacement	Data replication / Fault tolerance
[49]	Optimal Workload / Load Balancing in Fog	Fog Computing, Load Balancing	No, requires connectivity	Shared processing assurance
[20, 50, 53]	Decentralized Oracles and Oracle Networks	Distributed Oracle Networks, Blockchain	No, assumes connectivity and consensus	Oracle consensus / Reputation
[51]	Oracle with Authenticated Data Feed (Town Crier)	Trusted Execution Environment (TEE), Oracle	No, relies on authenticated data feed	TEEs, Data Authentication
[52]	Native Oracles for Blockchain Consensus (Aeternity)	Aeternity Blockchain, Native Oracles	No, used in consensus process	Blockchain Consensus
This Work	**Local system continuity during outage**	**Blockchain, Fog, Centralized Oracle**	**Yes, full read/write locally**	**Oracle's digital signature for offline data**

the new data.

3.2 Proposed architecture

The use case selected (sec. 3.1) requires blockchain data to be available to users authenticated to the HCU LAN during the network outage. This research uses the concepts of blockchain, fog computing, and the logic of a centralized oracle in order to make this possible. Figure 2 shows the main components of the proposed architecture. At the top, the blue rectangle highlights the blockchain network structure. Inside of it, *P1*, *P2*, and *P3* represent peers that are geographically located anywhere in the national territory and are connected by a wired network. Furthermore, *P4* and *P5* represent peers connected to the blockchain by a wireless network. These two peers can be intentionally disconnected from the blockchain at any moment, taken to remote places where there is no network structure, and provide local access to edge devices. After a certain period, when they have an Internet connection again, *P4* and *P5* can reconnect to the blockchain network.

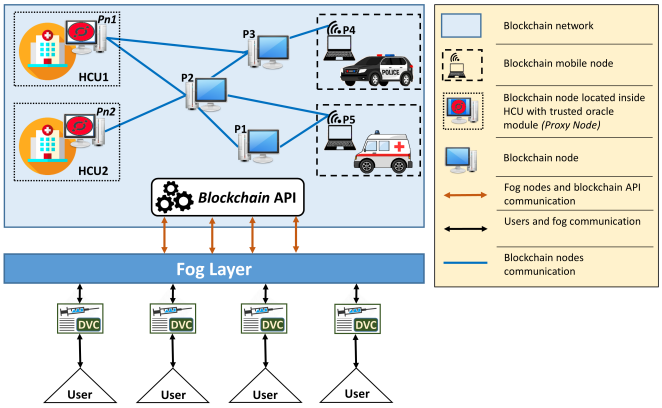

Figure 2. Architecture to increase the resilience of information systems in a smart city

The highlight of this study rests on the two *Proxy Nodes* (*Pn1* and *Pn2*) physically located within *HCU1* and *HCU2* and both are equipped with the oracle module, a local database (LDB), and a ledger copy with the citizen and vaccine data. In addition, the *blockchain API* executes the requests received from the *fog layer* which is responsible for storing and handling requests. Just below this layer are *DVC instances*, which are running on edge devices by users. These make read-and-write requests through the DVC. The Proxy Node (PN) is a critical component of this architecture and its inclusion is deliberate. It is more than a standard blockchain node; it is an integrated, lightweight component designed to operate within the resource constraints of an HCU. The PN serves as the local fog layer during an outage, hosting the API endpoint, the trusted oracle module, and the LDB for temporary storage. This combination of functionalities in a single, localized node avoids the need for a more resource-intensive full blockchain peer and provides a dedicated, streamlined solution for offline operation.

With this architecture, during periods of Internet outage in a HCU, users with authenticated devices on the HCU LAN can send read requests to a *Pn* through its local address (IP and port). Hence, the pre-existing data on the blockchain remains available on the local HCU network. During these periods it is also possible to generate new off-chain data, i.e. data outside the blockchain. To avoid tampering with off-chain data, an oracle module is used to encrypt the data with its *private key* and store it in a LDB. Later, for this data to be recorded on the blockchain, the oracle verifies its signature with its *public key*. Asymmetric keys were chosen due to the possibility of using a digital signature, which simplifies the verification of data saved in the LDB.

The choice of the oracle applied in the project considered the limited computational resources available on the LAN. That's why the oracle is *centralized trust*, which consumes fewer resources, and its *data source* is from *software*, as it queries only *Pn* data. About *interaction* it is *bidirectional*, as it can behave like *inbound* or *outbound*. Finally, regarding the *design pattern*, the oracle is classified as *publish-subscribe*.

To illustrate how the architecture works, figure 3 shows a fog computing layer working together with a blockchain network. Here, it is assumed that each node (from the fog and from the blockchain) is located in a geographically distinct location and that the *Pn* node is physically located within the HCU. At this first moment, all users use their edge devices to connect to the internet and make data requests to the fog layer nodes. Fog nodes communicate with the blockchain through an API and return incoming data to users.

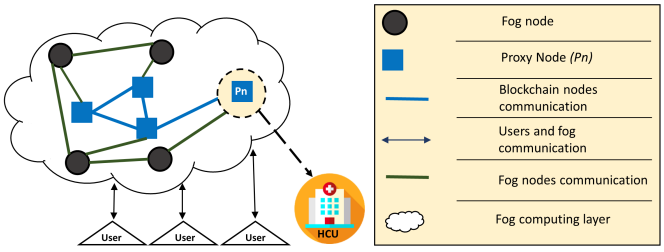

Figure 3. Fog layer working with blockchain and handling the requests

Figure 4 shows the moment when an event causes the network outage for the segment where HCU is located. It is now possible to observe that some users remain connected to the part of the fog layer that continues to function normally (left side of the figure). However, the network segment where the *Pn* node is located is disconnected from the rest of the network. The blue dotted rectangle in figure 4 represents the LAN in which the *Pn* node is inserted. This LAN is formed by a router connecting the blockchain *Pn* node and the edge devices that are used to send read/write requests. In similar scenarios, it is possible to use the proposed architecture where the *Pn* node starts to behave like a fog layer (for this reason, it was represented inside the red cloud).

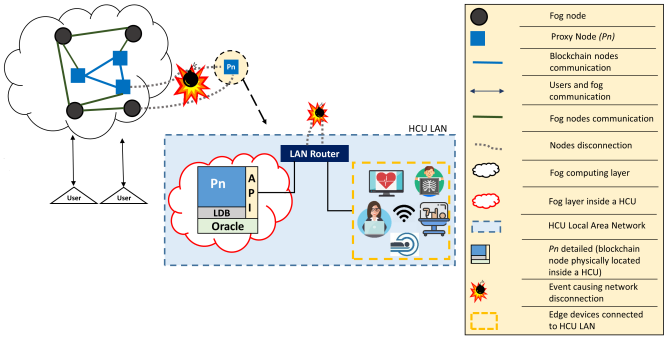

Figure 4. Event causes the network disconnection and the HCU LAN is now isolated

The main objective is to keep data available to DVC users even after disconnection. For this, whenever a disconnection is detected, the dynamics of DVC operation becomes as exemplified in Figure 5. As an example, consider that Alice wants to consult the personal data and vaccines applied to a citizen. To do this, she makes the query request 1, which is sent to the local address of the *Pn* node API as shown in 2. Then, Alice receives the API response and registers a new vaccine applied to a citizen 3. The registration request is sent to the oracle's local address 4, which uses its private key to encrypt the data and requests its recording in *Pn*'s LDB 5.

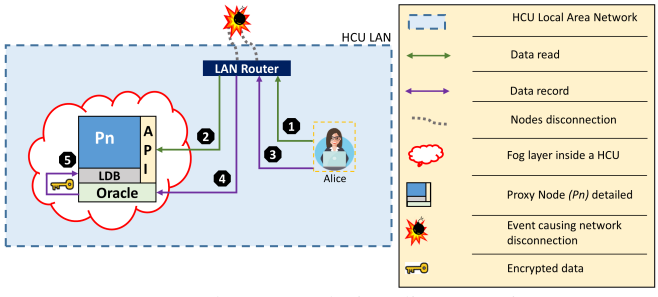

Figure 5. User send requests during disconnection moment

After some time, the LAN internet connection has been reestablished, as shown in Figure 6. In the indication of 6 we can see that *Pn* is now able to communicate with the blockchain and is available again to receive data requests from fog nodes. In this situation, the oracle starts reconciling the information, that is: sending the new data stored and encrypted in the LDB to the blockchain. For this, the oracle reads the data from the LDB 7, verifies it using its private key, and sends a request to write this data to the *Pn* 8 node API. After that, the request is sent to the blockchain 9 and the response received by the API is passed on to the oracle that requests its recording in the LDB.

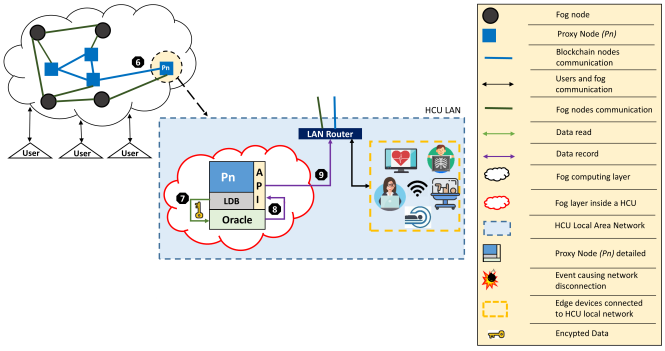

Figure 6. Network reconnection and off-chain data record to the blockchain

The responses received by the oracle can be the hash indicating data record success or an error message that must be handled. In case of error messages, they can be mainly due to unavailability, blockchain error, or even duplicated records, which can indicate attempts to use more than one dose of vaccines. Whether the answer is a success or an error, it is possible to compare it with the ledger to track and/or audit the records.

As the focus of this study is on data availability improving system resilience, there was no implementation of a protocol that verifies the fog disconnection. For this functionality, it was assumed that when performing two consecutive attempts to communicate with the fog, the system considers that there is a temporary disconnection. To verify the reconnection, it was also assumed that the system sends a data query request at each time interval and, if it receives a response from the fog, it considers that there has been a reconnection. Regarding secure reconnection, it always refers to the blockchain node that was isolated. The *Fabric* nodes already have a secure reconnection protocol and data update of the network whenever a disconnection occurs.

3.3 Proposed architecture benefits

The availability of pre-existing data on the blockchain for consultation and maintenance of writing off-chain data, both provided by the architecture in network outage periods, results in some direct benefits. One of these is the DVC functioning maintenance, which avoids the citizens service interruption and reduces the need to make manual notes (which mitigates transcription errors of these annotations to the system). Another benefit observed is to avoid the duplicate doses application of the same vaccine in the citizen, since the vaccine history continues available for queries and additions of new

records. Other less evident benefits arising from the features provided by the architecture are greater control of the HCU vaccine stock and also the mitigation of fraud in this stock.

Furthermore, the architecture uses two types of data to facilitate records traceability and auditability: *origin* data and *blockchain response* data. The *origin* is the raw data plus date and time information of new off-chain data generated using the oracle. The *blockchain response*, as indicated in section 3.2, are the possible responses when the oracle sends data for blockchain recording. One of those responses is*duplicate records*, which may indicate attempts by a citizen to receive more than one dose of the same vaccine. Whether the response received from the blockchain of success or error, it is possible to compare it with the ledger to track or audit the records, enabling tampering verification. All the aspects listed above create a resilience mechanism that helps keep an information system functioning in network outage moments. As a result, basic services provision is guaranteed, avoiding inconveniences to the smart city population.

4. Experiments and results

4.1 Implementation details

To understand if the proposed architecture is feasible, some experiments with employing a traditional blockchain architecture as a source for writing and reading. [4] The peers used were allocated in the form of *virtual machines* in service providers. The standard adopted was machines with 1 or 2-core processors, 1GB or 2GB RAM memory, and 15GB to 60GB HD SSD - always according to availability existing. To record data in the virtual machines, the local MySQL Server database was used, and to develop the test tools, the *Node js* language was used.

To prepare the machines and implement the blockchain network, *GoFabric*[5] was used, a permissioned blockchain network orchestrator based on Hyperledger Fabric technology. This tool has options for creating and changing networks, as well as data backup. The *chaincode* used was *GoShare*, which is an existing standard for *GoFabric*. In addition, *Grafana Monitor*[6], with the *Hyperledger Fabric Monitor 1.4* module, was used to observe the behavior of the hardware and network during idle and usage times.

4.2 Blockchain Network Deployed for Testing

For the execution of the tests carried out, a blockchain was implemented with two organizations: Org1 representing the UHS (Unified Health System) and *Org2* representing any other institution. Regarding the configurations adopted for the blockchain, the rule-defined endorsement allows only the UHS organization to carry out data inclusion. This same organization was given read-and-write access to all data, enabling its nodes to perform operations through the API. Another configuration performed was to enable each node located within a HCU as an API. Hence, they can receive and respond to data reading requests in a standardized way. In this work, it is assumed that to use HCU systems, users must perform authentication in a device that is connected to the HCU LAN. It is also assumed that the oracle is trusted and its private key is securely stored.

The reason for using a permissioned blockchain with two organizations is to adopt a realistic scenario, in which the UHS has read and write access to any data on the blockchain. *Org2* represents an organization that can only have access to a limited set of data on this same blockchain. A situation like this can occur when, for example, a company wins a bid to carry out a study for the government. This study is related to some profiles of citizens who received any vaccine and/or medicine. Therefore, the company can only have access to a specific set of data needed to carry out the aforementioned study. For the creation of this type of access control in a Fabric blockchain it is possible to use a Collection of Private Data (CDP). In addition, each of the organizations has the following configuration:

- a node as *orderer* (being part of the *Orderer Service*), network peer and also API;
- a node as CA (*Certificate Authority*) and also a network peer.

The deployed network is shown in figure 7. The node's geographic location is represented by the countries' flags. The choice of these locations, geographically distinct and distant, simulates a real situation where there would be nodes of the blockchain spread across several places in Brazil and also mitigates the risk of downtime network due to the joint unavailability of nodes in a given geographic region. In this same figure, some native services of Hyperledger Fabric are also shown separately, such as the Certificate Authority (CA), the Membership Service Provider (MSP), and the Orderer Service (OS). Also displayed are the API nodes (*API Org1* and *API Org2*), through which it is possible to access all the functionalities to act on blockchain data [28].

The tests were carried out to verify the availability of Orgs, access stress, and data reading/writing through the API. During the Internet outage simulation, requests to the API are made through the */query/search* route, which performs searches without the need to register on the blockchain. Using *GoFabric*, the blockchain assets previously shown in figure 1 were created.

The fog layer was implemented with two nodes, exchanging messages with network users. This layer is responsible for communication between users and API nodes. It receives requests from users, sends the data to the API, and returns the response to users. The fog also checks if the API nodes are available. If any node is unavailable, it redirects the request to an active node. Finally, at the bottom of Figure 7 the users

[4] To promote reproducibility, the source code and implementation artifacts are available from the corresponding author upon reasonable request.

[5] ⟨http://gofabric.io/⟩

[6] ⟨https://grafana.com/⟩

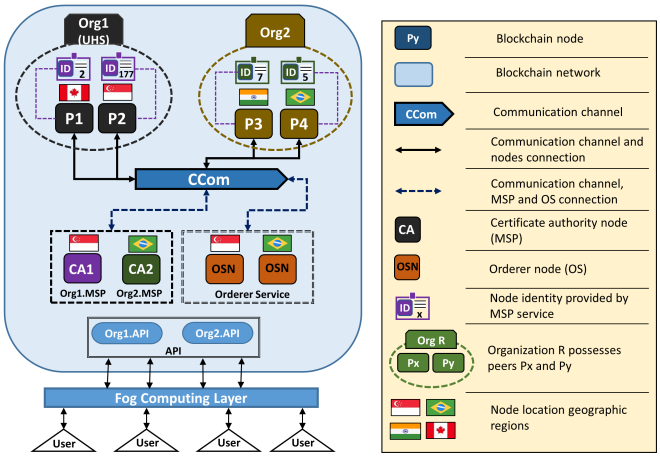

Figure 7. Deployed blockchain to perform tests

are represented. They access the fog layer through various device types, request the data reading/writing transactions on the blockchain, and wait for responses. Regarding the secure reconnection mentioned in section 3, it is a Hyperledger Fabric native implementation. Its nodes already have a secure reconnection protocol and network data update whenever a disconnection occurs.

4.3 Results

The writing tests mainly consisted of registering batches of 2,000 new records (writing) of citizen data through the organizations' APIs. In the first scenario, all network nodes were kept online, thus the blockchain was working normally. The dispersion of the response time among the 2,000 writing requests can be seen in figure 8. It shows the existence of a baseline below 5,000 ms formed by 1,878 requests, where most of the requests had a shorter duration at this time. Between the times of 5,001 ms and 10,000 ms, there were 11 requests and 111 requests with a response time greater than 10,001 ms.

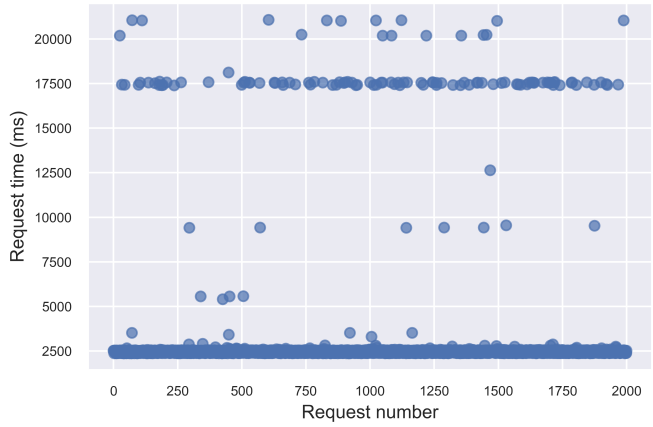

Figure 8. Blockchain write request response time

The observed times are acceptable according to the user's experience [54] when considering the configuration of the virtual machines, the geographical distance between the peers, the fog, and the database processing time. The chart in Figure 9 was created using *Grafana*, collecting information directly from the blockchain. It is possible to observe that the longest database processing times are close to 1 second, which represents around 30% of the time for most responses.

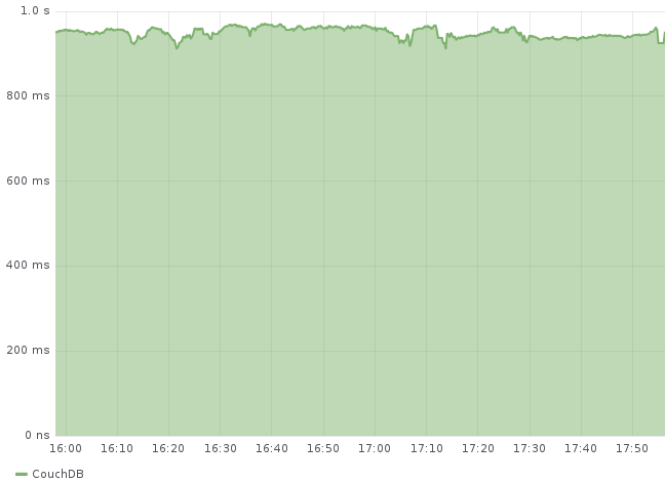

Figure 9. CouchDB processing time during blockchain write request

During simulations of the local network outage, P2, P3 and P4 peers were turned off, leaving only P1. When performing local read requests, node P1 sometimes returned the message *"Error: Failed to connect before the deadline URL:grpcs://[peer ip]:7051"*. If this error occurs, it is necessary to repeat the query. This error occurs because in a Hyperledger Fabric network there is a list of peers that run the API [28]. In this way, when a query request arrives, it will be directed to one of the peers in the list. If it is down, the error message is returned. When resending the query request, another peer in the list will be responsible for responding. In our experiments, two peers were configured to run the API. In this context, 70.4% of the requests returned successfully and 29.6% with error.

Figure 10 segments the response times for reading data from the blockchain and writing data to MySQL (through the trusted oracle), both on the local network and simulating network outage. In addition, it also displays the blockchain writing time through the Internet, considering the network working normally. Using the standard deviation of this data, it can be seen that writing on the blockchain, represented by the color orange, takes about 74.6% of the time of the entire process performed at times of disconnection. The 25.4% of the remaining time refers to the oracle process (reading the *Pn* node and writing to the MySQL database).

Figure 11 shows a box plot for each time measurement category presented in figure 10. In this figure it is possible to observe that, in the local network, the *Blockchain Read* operations are represented by a longer rectangle which reaches higher values than the *MySQL Write* operations. This means

Figure 10. Segmented request response time

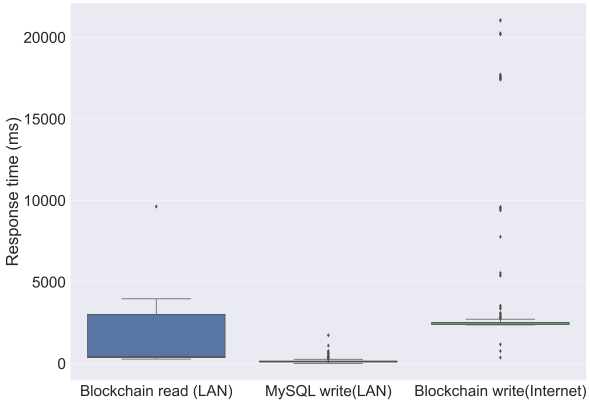

Figure 11. Response time range

that the *Blockchain Read* have more varied and longer times, mainly due to the need to redo the reading on the blockchain that returns an error. On the other hand, *Blockchain Write* operations carried out over the Internet show a smaller variation, focusing on times below 5,000 ms, with several larger outliers and only three smaller ones. This shows that most requests were answered in a shorter time.

The results obtained in the measurements of times show that 93.9% of the requests had an adequate response time, remaining within the appropriate limits to keep the users' attention [55]. Therefore, in addition to the proposed architecture achieving the objective of keeping an information system working during a period of disconnection, the results demonstrate that it is viable from the point of view of usability related to the response time for the user.

5. Conclusion and Future Work

This paper presented a novel architecture to enhance the resilience of smart city information systems, specifically addressing the challenge of network outages. By integrating a blockchain, a fog computing layer, and a trusted oracle within a local Proxy Node (PN), our solution ensures that critical applications, such as the Digital Vaccination Card (DVC) in a Healthcare Unit (HCU), remain fully functional offline. The architecture allows for continuous access to historical data stored on the local ledger copy and enables the secure recording of new data, which is signed by an oracle and synchronized with the main blockchain once connectivity is restored. Our experiments, conducted on a geographically distributed Hyperledger Fabric network, demonstrated the viability of the proposed approach. The results showed that even during a simulated network outage, read and write operations could be performed locally with acceptable response times, maintaining system usability. This confirms that the architecture can successfully guarantee data availability and integrity, thereby preventing service interruptions and ensuring the continuity of essential public services. For future work, we identify several promising directions. A primary area of investigation is to address the limitation of data synchronization between multiple, simultaneously disconnected HCUs. Exploring ad-hoc connectivity mechanisms, such as Mobile Ad-hoc Networks (MANETs) or 5G-based peer-to-peer communication, could enable PNs to exchange critical updates even without full internet access, mitigating risks like duplicate vaccinations. Additionally, we plan to develop more sophisticated, automated protocols for detecting network disconnection and managing the reconnection and data reconciliation process. Finally, the proposed architecture could be generalized and applied to other critical smart city scenarios, such as traffic management or emergency response systems, where operational continuity during network failures is paramount.

Acknowledgements

The authors would like to acknowledge the partial support from *Banco do Brasil* through an internal scholarship. The authors are grateful to *GoLedger* for providing free access to *GoFabric*[7] platform and technical support.

Conflicts of Interest

The authors have no financial or other conflicts of interest.

Author contributions

Leidmar M. Festa: Writing (original draft), Software, Formal Analysis, Conceptualization, Methodology, Visualization. **Daniel F. Pigatto:** Supervision, Conceptualization, Writing (review & editing). **Luiz C. Gomes Jr.** Supervision, Conceptualization, Writing (review & editing). All authors named in the manuscript have made a significant contribution to the writing, concept, design, execution, or interpretation of the work represented. All authors agree with the author's list that appeared in the manuscript.

[7] ⟨https://goledger.io/⟩

References

[1] XIE, J. et al. A Survey of Blockchain Technology Applied to Smart Cities: Research Issues and Challenges. *IEEE Communications Surveys and Tutorials*, IEEE, v. 21, n. 3, p. 2794–2830, 2019. ISSN 1553877X.

[2] BARON, M. DO WE NEED SMART CITIES FOR RESILIENCE. *Journal of Economics And Management*, v. 10, p. 32–46, 2012.

[3] NAKAMOTO, S. A peer-to-peer Electronic Cash System. *Nakamoto Institute*, 2008. Disponível em: ⟨https://nakamotoinstitute.org/bitcoin/⟩.

[4] MICROSOFT. *Understanding Public Key Cryptography*. 2014. Data de acesso: 23/08/2021. Disponível em: ⟨https://docs.microsoft.com/en-us/previous-versions/tn-archive/aa998077(v=exchg.65)?redirectedfrom=MSDN⟩.

[5] COULOURIS, G. et al. *Sistemas Distribuídos - Conceitos e Projetos*. Porto Alegre/RS: Bookman, 2013. ISBN 978-0-13-214301-1.

[6] CHANDRA, S. et al. A study and analysis on symmetric cryptography. *2014 International Conference on Science Engineering and Management Research, ICSEMR 2014*, 2014.

[7] ZHOU, J.; LAM, K. Y. Securing digital signatures for non-repudiation. *Computer Communications*, v. 22, n. 8, p. 710–716, 1999. ISSN 01403664.

[8] SOUZA, E. F. D. Geração de casos de teste para sistemas da área espacial usando critérios de teste para máquinas de estados finitos. *Ministério da Ciência e Tecnologia - Instituto Nacional de Pesquisas Espaciais (INPE)*, 2010. Data de acesso: 22/07/2021. Disponível em: ⟨http://mtc-m16d.sid.inpe.br/col/sid.inpe.br/mtc-m19@80/2010/02.06.20.39/doc/publicacao.pdf⟩.

[9] VALENTA, M.; SANDNER, P. Comparison of Ethereum, Hyperledger Fabric and Corda. *Frankfurt School Blockchain Center*, v. 8, n. June, p. 8, 2017. Disponível em: ⟨https://medium.com/@philippsandner/comparison-of-ethereum-hyperledger-fabric-and-corda⟩.

[10] WANG, S. et al. On private data collection of hyperledger fabric. *Proceedings - International Conference on Distributed Computing Systems*, v. 2021-July, p. 819–829, 2021.

[11] DIB, O. et al. Consortium blockchains: Overview, applications and challenges. *International Journal on Advances in Telecommunications*, v. 11, n. 1&2, p. 51–64, 2018.

[12] FURTADO, F. R. *L7SP: Serviços para otimizar o gerenciamento e o desempenho de Blockchain Privados Dissertação*. 118 p. Tese (Thesis) — Unisionos RS, 2019. Disponível em: ⟨http://www.repositorio.jesuita.org.br/handle/UNISINOS/8706⟩.

[13] SWAN, M. *Blockchain: Blueprint for a new economy*. [S.l.]: O'Reilly Media Inc, 2015.

[14] CHRISTIDIS, K.; DEVETSIKIOTIS, M. Blockchains and Smart Contracts for the Internet of Things. *IEEE Access*, IEEE, v. 4, p. 2292–2303, 2016. ISSN 21693536.

[15] FREY, D. et al. Dietcoin: Shortcutting the Bitcoin verification process for your smartphone. *arXiv*, v. 18, n. March, 2018. ISSN 23318422.

[16] ISO. *Blockchain and distributed ledger technologies – overview of and interactions between smart contracts in blockchain and distributed ledger technology systems*. [S.l.], 2019. TC 307 - I.

[17] AL-BREIKI, H. et al. Decentralized access control for IoT data using blockchain and trusted oracles. *Proceedings - IEEE International Conference on Industrial Internet Cloud, ICII 2019*, v. 5, n. Icii, p. 248–257, 2019.

[18] XU, X. et al. The blockchain as a software connector. *Proceedings - 2016 13th Working IEEE/IFIP Conference on Software Architecture, WICSA 2016*, p. 182–191, 2016.

[19] LONE, A. H.; MIR, R. N. Forensic-chain: Blockchain based digital forensics chain of custody with PoC in Hyperledger Composer. *Digital Investigation*, Elsevier Ltd, v. 28, p. 44–55, 2019. ISSN 17422876. Disponível em: ⟨https://doi.org/10.1016/j.diin.2019.01.002⟩.

[20] ADLER, J. et al. Astraea: A Decentralized Blockchain Oracle. In: *Proceedings - IEEE 2018 International Congress on Cybermatics: 2018 IEEE Conferences on Internet of Things, Green Computing and Communications, Cyber, Physical and Social Computing, Smart Data, Blockchain, Computer and Information Technology, iThings/Gree*. [S.l.: s.n.], 2018. p. 1145–1152. ISBN 9781538679753.

[21] AZARIA, A. et al. MedRec: Using blockchain for medical data access and permission management. *Proceedings - 2016 2nd International Conference on Open and Big Data, OBD 2016*, p. 25–30, 2016.

[22] PASDAR, A.; DONG, Z.; LEE, Y. C. Blockchain Oracle Design Patterns. *arXiv*, p. 1–25, 2021. Data de acesso: 26/11/2021. Disponível em: ⟨http://arxiv.org/abs/2106.09349⟩.

[23] AL-BREIKI, H. et al. Trustworthy Blockchain Oracles: Review, Comparison, and Open Research Challenges. *IEEE Access*, v. 8, p. 85675–85685, 2020. ISSN 21693536.

[24] SARAF, C.; SABADRA, S. Blockchain platforms: A compendium. *2018 IEEE International Conference on Innovative Research and Development, ICIRD 2018*, IEEE, n. May, p. 1–6, 2018.

[25] Hyperledger Foundation. *Hyperledger*. 2016. Disponível em: ⟨https://www.hyperledger.org/⟩.

[26] GORENFLO, C. et al. FastFabric: Scaling hyperledger fabric to 20 000 transactions per second. *International Journal of Network Management*, v. 30, n. 5, p. 1–18, 2020. ISSN 10991190.

[27] ANDROULAKI, E. et al. Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains. *Proceedings of the 13th EuroSys Conference, EuroSys 2018*, v. 2018-Janua, 2018.

[28] Hyperledger Fabric. *Hyperledger Fabric Docs*. 2020. Data de acesso: 06/03/2022. Disponível em: ⟨https://hyperledger-fabric.readthedocs.io/en/latest/whatis.html⟩.

[29] BRANDENBURGER, M. et al. Rollback and Forking Detection for Trusted Execution Environments Using Lightweight Collective Memory. *Proceedings - 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2017*, p. 157–168, 2017. Disponível em: ⟨https://cachin.com/cc/papers/lcm.pdf⟩.

[30] BRANDENBURGER, M. et al. Blockchain and trusted computing: Problems, pitfalls, and a solution for hyperledger fabric. *arXiv*, 2018. ISSN 23318422.

[31] BROTSIS, S. et al. On the Security and Privacy of Hyperledger Fabric: Challenges and Open Issues. *Proceedings - 2020 IEEE World Congress on Services, SERVICES 2020*, p. 197–204, 2020.

[32] CISCO. Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are. *White Paper*, p. 1–6, 2015.

[33] BONOMI, F. et al. Fog computing and its role in the internet of things. *MCC'12 - Proceedings of the 1st ACM Mobile Cloud Computing Workshop*, p. 13–15, 2012.

[34] SAURABH; DHANARAJ, R. K. A Review Paper on Fog Computing Paradigm to solve Problems and Challenges during Integration of Cloud with IoT. *Journal of Physics: Conference Series*, v. 2007, n. 1, 2021. ISSN 17426596.

[35] ALGHAMDI, A.; ALZAHRANI, A.; THAYANANTHAN, V. Fog Network Area Management Model for Managing Fog-cloud Resources in IoT Environment. *International Journal of Advanced Computer Science and Applications*, v. 12, n. 3, p. 482–489, 2021. ISSN 21565570.

[36] LAUTERT, F. *Distributed data provenance: fog computing and blockchain improving privacy control, trust and reliability*. 75 p. Tese (Dissertation) — UTFPR - Universidade Tecnológica Federal do Paraná, 2020. Disponível em: ⟨https://repositorio.utfpr.edu.br⟩.

[37] LAUTERT, F.; PIGATTO, D. F.; GOMES, L. Distributed data provenance: Fog Computing and Blockchains improving privacy control, trust and reliability. *Ibict Utfpr*, 2020. Disponível em: ⟨https://bdtd.ibict.br/vufind/Record/UTFPR-12_1a68cd34586065ee3bb4676ac3cccf1c⟩.

[38] BABAR, M.; TARIQ, M. U.; JAN, M. A. Secure and resilient demand side management engine using machine learning for IoT-enabled smart grid. *Sustainable Cities and Society*, Elsevier, v. 62, n. June, p. 102370, 2020. ISSN 22106707. Disponível em: ⟨https://doi.org/10.1016/j.scs.2020.102370⟩.

[39] BRILLIANTOVA, V.; THURNER, T. W. Blockchain and the future of energy. *Technology in Society*, Elsevier Ltd, v. 57, p. 38–45, 2019. ISSN 0160791X.

[40] AHMAD, T.; ZHANG, H.; YAN, B. A review on renewable energy and electricity requirement forecasting models for smart grid and buildings. *Sustainable Cities and Society*, Elsevier, v. 55, n. April 2019, p. 102052, 2020. ISSN 22106707.

[41] SANSEVERINO, E. R. et al. The blockchain in microgrids for transacting energy and attributing losses. *Proceedings - 2017 IEEE International Conference on Internet of Things, IEEE Green Computing and Communications, IEEE Cyber, Physical and Social Computing, IEEE Smart Data, iThings-GreenCom-CPSCom-SmartData 2017*, v. 2018-January, p. 925–930, 2018.

[42] WANG, Q.; LI, R.; ZHAN, L. Blockchain technology in the energy sector: From basic research to real world applications. *Computer Science Review*, Elsevier Inc., v. 39, p. 100362, 2021. ISSN 15740137.

[43] ALLADI, T. et al. Blockchain Applications for Industry 4.0 and Industrial IoT: A Review. *IEEE Access*, IEEE, v. 7, p. 176935–176951, 2019. ISSN 21693536.

[44] BATTY, M. et al. Smart cities of the future. *European Physical Journal: Special Topics*, v. 214, n. 1, p. 481–518, 2012. ISSN 19516355.

[45] GODSCHALK, D. R. Urban Hazard Mitigation: Creating Resilient Cities. *Natural Hazards Review*, v. 4, n. 3, p. 136–143, 2003. ISSN 1527-6988.

[46] CASTELLANO, G.; RISSO, F.; LOTI, R. Fog Computing over Challenged Networks: A Real Case Evaluation. *Proceedings of the 2018 IEEE 7th International Conference on Cloud Networking, CloudNet 2018*, IEEE, p. 1–7, 2018.

[47] KULATUNGA, C. et al. Opportunistic Wireless Networking for Smart Dairy Farming. *IT Professional*, v. 19, n. 2, p. 16–23, 2017. ISSN 15209202.

[48] JEONG, T. et al. Towards a Distributed Computing Framework for Fog. In: *IEEE Fog World Congress (FWC)*. [S.l.: s.n.], 2017. p. 195–210. ISBN 9781538636664.

[49] AL-KHAFAJIY, M. et al. Iot-fog optimal workload via fog offloading. In: *Proceedings - 11th IEEE/ACM International Conference on Utility and Cloud Computing Companion, UCC Companion 2018*. [S.l.]: IEEE, 2019. p. 349–352. ISBN 9781728103594.

[50] KOLINKO, T. *Orisi White Paper*. 2014. Disponível em: ⟨https://github.com/orisi/wiki/wiki/Orisi-White-Paper⟩.

[51] ZHANG, F. et al. Town crier: An authenticated data feed for smart contracts. *Proceedings of the ACM Conference on Computer and Communications Security*, v. 24-28-Octo, p. 270–282, 2016. ISSN 15437221.

[52] HESS, Z.; MALAHOV, Y.; PETTERSSON, J. Æternity Blockchain Whitepaper. *White paper*, v. 01, p. 1–10, 2017. Disponível em: ⟨https://blockchain.aeternity.com/{\OT1\ae}ternity-blockchain-whitepaper⟩.

[53] BREIDENBACH, L. et al. Chainlink 2.0: Next Steps in the Evolution of Decentralized Oracle Networks. *White paper*, p. 1–136, 2021. Disponível em: ⟨https://research.chain.link/whitepaper-v2.pdf⟩.

[54] MILLER, R. B. Response time in man-computer conversational transactions. Introductions and major concepts. *AFIPS Conference Proceedings*, v. 33, p. 267–277, 1968.

[55] NIELSEN, J. *Usability Engineering*. 1st. ed. Mountain View, California - USA: Morgan Kaufmann Publishers Inc., 1993. ISBN 9781119130536.