DECENTRALIZED HEALTH RECORD MANAGEMENT SYSTEM

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Use Case Report

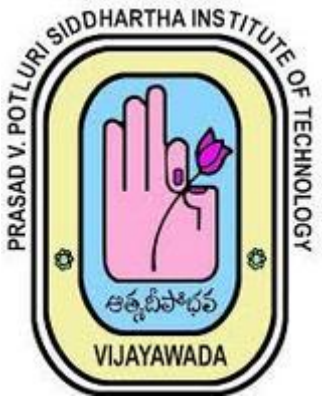
submitted by

NEKKALA CHAITRIKA

22501A05C5

Under the guidance of

Mr. A. Prashant, Asst. Prof.



Department of Computer Science and Engineering

Prasad V Potluri Siddhartha Institute of Technology

(Permanently affiliated to JNTU-Kakinada, Approved by AICTE)

(An NBA & NAAC accredited and ISO 9001:2015 certified institute)

Kanuru, Vijayawada-520 007

2024-25

Prasad V Potluri Siddhartha Institute of Technology

(Permanently affiliated to JNTU-Kakinada, Approved by AICTE)

(An NBA & NAAC accredited and ISO 9001:2015 certified institute)

Kanuru, Vijayawada-520 007

CERTIFICATE

This is to certify that the Use Case report entitled **"Decentralized Health Record Management System"** that is being submitted by **Nekkala Chaitrika, 22501A05C5** as part of Assignment-1 and Assignment-2 for the **Blockchain Technology(20CS4601C)** course in **3-2** during the academic year **2024-25**.

Course Coordinator

Mr. A. Prashant

Assistant Professor,

Department of CSE,

PVPSIT, Vijayawada

Head of the Department

Dr. A. Jayalakshmi,

Professor and Head,

Department of CSE,

PVPSIT, Vijayawada

MARKS
ASSIGNMENT-1: ____/5
ASSIGNMENT-2: ____/5

INDEX

1.INTRODUCTION

1.1 Blockchain Technology

1.1.1 What is Blockchain

Technically, Blockchain is defined as a distributed, replicated peer-to-peer network of databases that allows multiple non-trusting parties to transact without a trusted intermediary and maintains an ever-growing, append-only, tamper-resistant list of time-sequenced records.

In short, Blockchain is a type of distributed ledger that sits on the internet for recording transactions and maintaining a permanent and verifiable record-set of information.

This innovative technology was first published in 2008 by Satoshi Nakamoto (pseudonym of a person/persons as-of-yet unknown) in a white paper titled “A Peer-to-Peer Electronic Cash System.” The thought behind the design was to create a decentralized digital currency that is free from government regulation whereby two people can confidently trade directly with one another without the need for mediators or intermediaries.

In 2009, the concept became a reality when Satoshi Nakamoto implemented the first application of the Blockchain we all know as Bitcoin.

Though Bitcoin and Blockchain are often referred to interchangeably, they are not the same. Blockchain is the underpinning technology that the Bitcoin was built on. Now aside from Bitcoin blockchain, we have the Ethereum blockchain and other private blockchains that are adapted to cater to various industry applications such as supply chain, cross-border payments, and many others.

1.2 Purpose

The case study provides a systematic study to understand whether permissioned blockchain implementations could be of any benefit to managing health records in emergency situations caused by natural disasters. First, the case study presents a summary of consensus protocols in distributed systems and comprehensive investigation of the current applications of blockchain to manage EHRs. Then, it will be shown how the blockchain applied to EHRs fails to provide detailed analyses and tests.

After the design and implementation of a basic EHRs management system and the execution of a set of test cases, it will be possible to discuss the benefits and trade-offs that the system entails. The discussion will examine the performance of a permissioned blockchain for managing Electronic Health Records (EHRs). It will

compare normal and disaster scenarios using key performance indicators to gain valuable insights into how a crisis situation impacts blockchain network operations.

• Success rate: the number of successes and failures of a batch of requests. It is important to limit the number of failures caused by a surge of requests during an emergency

• Transaction commit and read latency: this refers to the time it takes to the blockchain-based system to process an access request to an EHR in a disaster situation. It is important as timeliness in getting health data, especially in emergencies, is critical [19].

• Transaction commit and state read throughput (TPS): this refers to the number of requests that can be managed by the system at the same time. Being able to access and modify a growing number or request is essential to enable everybody to interact with the system;

• Resource consumption (CPU, Memory and network IO): it is necessary to take these parameters into account as they affect all the other indicators.

1.3 Boundaries and Constraints:

The topic of blockchain in healthcare is broad and can range from health record privacy and management to data analytics for the research. The scope of this case study work is to undertake a feasibility study and address the performance of an EHR system during the response to a mass crises generated by a natural disaster. The case study presents a simple access control and storage solution on top of the Hyperledger Fabric permissioned blockchain with the main objective to execute tests and experiments.

2.BACKGROUND

Background and Problem

Before the introduction of smart contacts on the blockchain, the main discussions on Electronic Health Record (EHR) Management focused on whether to use cloud infrastructures or local centralized systems for storing and sharing EHRs. These centralized systems implied that each hospital and healthcare company would have to keep data on premise in locally managed structures and databases. However, centralized EHRs management systems present some issues as described below:

• No patient control: The patients do not own the data and have no control over it. The patients should own and control their data.

• Scattered records: As patients seek treatments in different structures, the records are replicated. The information becomes scattered.

• Limited system interoperability: Different hospitals and health facilities have different systems. Integration and interoperability issues are the consequences.

• Inconvenient secure sharing: Often times, the process of sharing health records is complex and time-consuming. In the U.S. a secure email standard called Direct is used to provide encrypted transmission between the sender (for example, an E.R. physician) and receiver.

These frameworks brought solutions to many of the challenges listed above, they still suffered from limitations especially as related to transparency, data ownership, and privacy. Moreover, the centralized model for EHRs management struggles in crisis and disaster scenarios because the response to the emergency is often unorganized and decentralized.

Although natural disasters are rare events, they introduce new challenges as the healthcare sector should be prepared and able to respond to the crisis promptly. In fact, flooding, tsunamis, and earthquakes could potentially disrupt facilities and infrastructures, thereby limiting the access to records and patient information. This is one of the arguments that demonstrate how decentralizing the management of EHRs and replicating and distributing the information can assure better performance and availability in disaster situations, compared to centralized models.

A decentralized system is a distributed network where no party has the full control over the data and the operations, but the decisions are made collectively through a consensus process. The parties forming the network are called nodes and communicate through message passing [3]. Generally speaking, by sharing and replicating the information, the network provides availability and robustness especially in case of extensive failures. Moreover, Peer-to-peer systems (P2P) can even provide data ownership as the private information can be stored and requested only to the proprietary node. However, reaching consensus while preserving

confidentiality, security, and correctness despite failures has been a challenging problem [3]. The introduction of blockchain made possible to achieve it while preserving confidentiality and providing security.

A blockchain is a data structure where the records are stored in a linked sequence of blocks. This sequence forms a distributed ledger, which means it is replicated in multiple machines, called nodes, that communicate with one another. The nodes form a peer to peer network where every update to the ledger must be accepted by the network using a consensus protocol. The consensus protocol assures that everybody has the same view on the status of the system .

However, it is important to analyze if and how a blockchain system designed for health records would behave in such scenarios. This implies that an accurate assessment of security and resiliency, as well as transaction throughput, must be done to be sure that the system would meet the security and performance requirements. Among the requirements, it is possible to list: the ability to coordinate and cooperate through a secure mean of communication; the ability to continuously reach life-saving information; and the ability to let NGOs and rescuers join and access medical records without compromising privacy and confidentiality. In this context, blockchain is potentially the best solution because the operations during a disaster are inherently decentralized and the help must be coordinated and on time: this is essential when the life or well-being of someone is at stake.

3. BLOCKCHAIN BASICS

3.1 Blockchain

A blockchain is a data structure where the records are stored in a linked sequence of blocks. This sequence forms a distributed ledger, which means it is replicated in multiple machines, called nodes, that communicate to one another. The nodes form a peer to peer network where every update to the ledger must be accepted by the network using a consensus protocol. The consensus protocol assures that everybody has the same view on the status of the system.

The technology was introduced in 2008 as the foundation of a cryptocurrency called Bitcoin, with the aim to solve the double spending problem[6] and allow value exchange between peers without the mediation of a third-party central authority.

Besides being distributed, blockchain is secure by design and resilient to node failures, misbehaviors or malicious acts[6][7]. This and other characteristics made it appealing for many applications, such as medical records and identity management; supply chain management; assets insurances; luxury product anti-counterfeiting and provenance.

3.1.1 Active and passive replication in distributed systems

In distributed systems, replication is mainly used to provide fault tolerance. In particular, there are two types of replication: active and passive. Active, or state machine replication, requires the processes to be deterministic so that every node that runs the application gets the same final output. This is necessary to keep the state consistent. Determinism also requires the updates to be sent to everybody in the same order. This is possible thanks to an atomic broadcast protocol with delivery guarantees. Active replication allows the system to work well even in the presence of Byzantine faults [5].

In passive replication, only one node, or a subset of nodes, processes a request and sends the new state to all the others. Passive replication could also be used for non-deterministic processes but tends to be less resilient in case of faults [5].

3.2 Blockchain models

As mentioned above, a blockchain is a distributed network open to anyone. This definition usually relates to a particular model known as permission-less or public. In the public model, any participant can join and leave at will because no rule restricts access and interaction. Therefore, the data stored in a public blockchain (i.e., Bitcoin [6] or Ethereum [7]) is accessible by anyone unless encryption and smart contract logic are employed. Besides the public model, blockchain can also be employed in a restricted network where the participants' identities are known. This restricted model is usually referred to as permissioned or consortium. The model of participation has a significant influence on how the consensus is reached by the network.

3.2.1 Permissionless model

In the permissionless model, identities are either anonymous or pseudonymous, and everybody is allowed to participate. Any user can generate a set of keys and an address that enables her to interact with other entities in the blockchain network. Therefore, everybody has the right to read data, create transactions and append information to the ledger. This model also allows to install a blockchain node and participate in the transaction validation process known as consensus. Examples of such networks are Bitcoin and Ethereum. In the latter, the user can create and install code, known as smart contract, that is public and invokable by anyone. The smart contract is identified by an address and runs in an environment called Ethereum Virtual Machine (EVM).

The early permissionless blockchains entail a consensus protocol known as Proof of Work (PoW) where entities called miners compete in order to find the solution to a computational intensive mathematical problem. The novelty of this mechanism lies on the ability to reach agreement in a network with no formal barrier by substituting them with economic ones. It means that the effect of a single node in the consensus process is directly proportional to the computing power that it can produce.

3.2.2 Permissioned and consortium model

A permissioned blockchain is a closed system where the participants have identities and know one other. It is built to allow a consortium or a single organization to securely and efficiently exchange information. As proof of the fact that anonymity of participants is not always a desirable property, the permissioned model is gaining interest among enterprises because it allows secure interactions in a network of businesses with common goals but which do not fully trust each other [1]. One of the most prominent work is Hyperledger Fabric, an open source project hosted by the Linux Foundation. Fabric's modular and extensible architecture is designed to fit different enterprise use cases.

In the implementations mentioned above, privacy and confidentiality are managed by trusted parties, called membership services. In Fabric, this service is known as Membership Service Provider and has the role maintain all the identities in the system. It is responsible for issuing credentials used for authentication and authorization. In general, each organization has a local implementation of the service that is used to generate certificates and public keys for its members [1]. The credentials are necessary to participate in the network activities as every message and transaction must be signed. This, in turn, increases the privacy and security of the network as well as its participants.

3.3 Consensus mechanisms

Blockchain presents all the characteristics of a distributed system because the computation is done concurrently by different entities, without the assistance of a global clock. In addition, any process can fail at any point during the execution. According to the theory of distributed systems [3] there are two types of failures: crash (or stop) failure and Byzantine failure. The former is the simplest to assess and consists of a node crashing without resuming, thus leaving the network. The rest of the processes can spot a faulty node because it suddenly stops replying to messages.

The latter, on the other hand, is more complicated to model and address because a component is not able to definitely infer if another has failed. In fact, a Byzantine failure has no restriction: a component can crash, be delayed, or produce both correct and wrong messages; it can also act maliciously against the network and appear both functioning and failed to different observers. The term originates from the Byzantine generals problem described in [8].

In blockchain, the consensus is needed to agree on the total order of transactions and the block that must be added to the chain. The equivalence between the Proof of Work protocol proposed in Bitcoin and the consensus in distributed systems was formalized in the work of Garay [8]. The consensus is the mechanism by which the state machines agree on the order of messages sent. This mechanism is usually referred to as atomic broadcast and is needed by the state machines to be able to produce the same output, given the same input. The output can be the same only if the logic is deterministic and identical for each machine. The determinism is also necessary to detect faults: in fact, a replica that produces a different output can be considered as misbehaving.

In literature, it is possible to find consensus implementations that fall either below the crash-tolerant family or the Byzantine fault tolerant (BFT) family . In general, an algorithm must satisfy the following requirements to be considered correct [4]:

• **validity:** if all the processes propose v, then all processes decide v

• **termination:** every correct process eventually decides a value

• **agreement:** all the correct processes agree on a common value

• **integrity:** "Every correct process decides at most one value, and if it decides some value v, then v must have been proposed by some process" [4].

3.3.1 Proof of Work

The full description of a proof of work consensus was first introduced as the fundamental way to agree on the order of transactions in the Bitcoin network. It needs a distributed timestamp server to demonstrate that a transaction has come before another and that an entity has not spent the same amount of money in other transactions [6].

The approach to solve this problem consists of finding a value that, when hashed with an algorithm like SHA-256, presents an output with a number of leading zero bits. In the context of a block, the goal is to increment a nonce so that the block's hash has a number of leading zeros greater than the one required by the system. It has been proved that there is no efficient algorithm to solve this mathematical puzzle. Therefore, the only approach is to increment the nonce until the output satisfies the requirements. Finding the solution proves that some work has been done by a CPU and guarantees that a block cannot be modified without repeating the process [6][7].

The time required to solve the puzzle grows exponentially with the number of required zeros. In addition, the longer the chain, the more difficult becomes to change a block because it requires to prove the work for that block and all the subsequent ones. It would be possible only if a set of malicious attackers has a computing power greater than the rest of the network.

The difficulty of PoW is continuously adjusted to control the block formation frequency and reduce the number of forks. When a fork happens, all the blocks forming the shortest chain are pruned and their transactions are considered invalid. This heavily impacts the overall consensus latency as transactions are required to have a minimum amount of subsequent blocks (known as block confirmations) to reduce the probability to be pruned or removed.

3.3.2 Proof of Stake

Another set of consensus algorithms is the one represented by Proof of Stake (PoS). It was initially proposed as an energy-efficient alternative to Bitcoin's PoW to reach consensus in public blockchains. In fact, PoW has proven to be incredibly inefficient in terms of energy used as all the nodes in the network must prove the work, but only one is eventually able to add a new block.

To solve the problem while ensuring security and decentralization, PoS takes advantage of a group of validators (a subset of the blockchain network) taking turns to propose, vote, and add new blocks to the chain. To become a validator, one has to send a specific transaction that locks its coins into a vault. The vault opens only once the validator has managed to add a block. Therefore the algorithm requires the participants to hold coins (value in the form of a cryptocurrency) to put at stake to join, as well as to track them [1].

At periodic intervals, all the validators participate in a consensus process to determine the following blocks to add. Although there is no single rule to decide how to reach an agreement, two approaches are, in general, the most common: the random approach and the Byzantine Fault Tolerant approach [1]. In the former, a validator is randomly selected and is given the right to create a new block linked to the end block of the longest chain. The latter is achieved through a multi-round process where each member votes for one of the proposed blocks and the group eventually converge to a collective decision. This is possible since the number of participants is known, and the votes can be linked to specific identities (in the form of addresses) that remain the same throughout the whole voting process [1].

3.3.3 PBFT Consensus

In the context of permissioned blockchain networks, Byzantine fault-tolerant protocols are the most promising to ensure the security in the presence of faulty components. However, these protocols were originally based on synchronous models that made them unsuitable for network applications. In addition, the scalability of nodes was hindered by the communication overhead growing exponentially with the number of participants to the consensus [7]. These characteristics made BFT algorithms less employed than the crash-tolerant counterparts (i.e., Paxos) until a practical BFT (PBFT) algorithm was introduced in 1999 . The algorithm reduces the overall response time by decreasing the communication overhead from exponential to polynomial. Moreover, it is designed to work in eventually synchronous environments that make it a good fit for systems that communicate over Internet protocols [2][7].

The algorithm consists of five steps as illustrated in figure 3.1:

• **Request:** the client sends a request to the master node.

• **Pre-prepare:** the master node forwards the request to the other nodes which decide whether to accept the request or not.

• **Prepare:** in case the nodes accept the execution, they send a preparation message to all the other nodes. Upon receiving at least 2f + 1 messages, the nodes start the commit phase if the majority has accepted the request.

• **Commit:** each node sends a commit message to all the other nodes in the system. When a node receives 2f + 1 commit messages, it executes the logic to fulfill the request because it infers that the majority of the nodes has accepted the request.

• **Reply:** finally, the server node replies to the client that waits until the reception. If any message is delayed, the client triggers a timeout and resends the request to the master node [4].

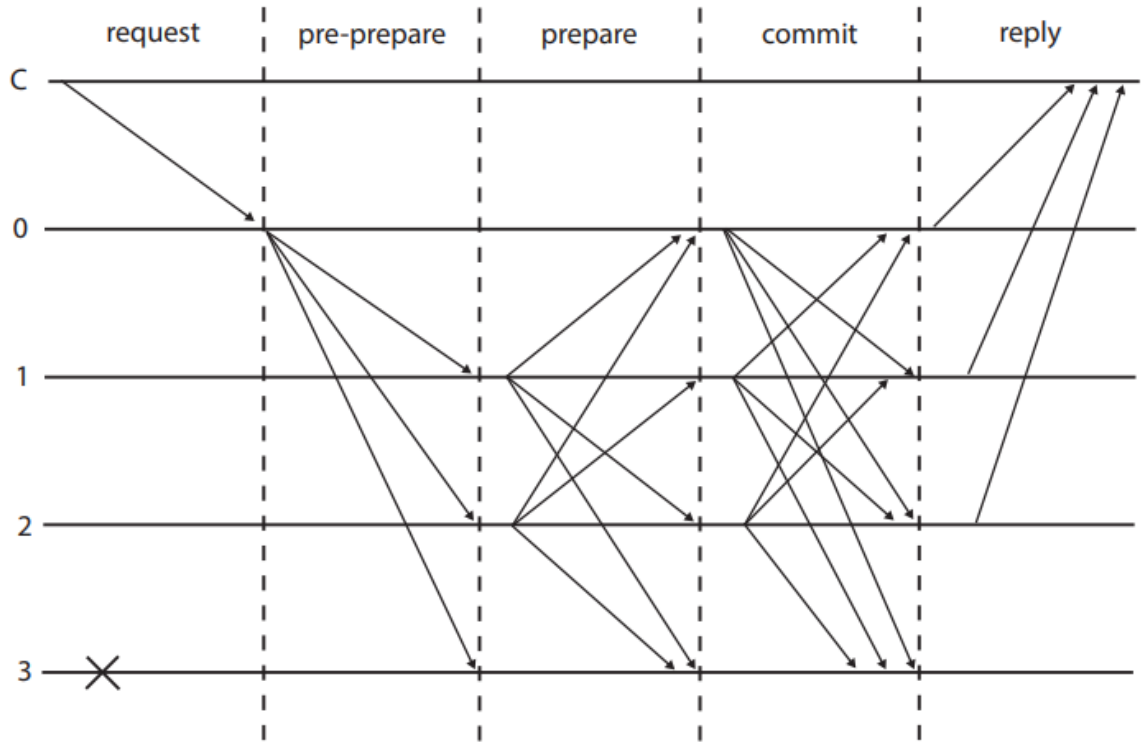


Figure 3.1: PBFT communication

Five steps of the communication. Image adapted from [9].

This algorithm needs a leader to be elected at each round and can be seen as an extension of the Paxos and VSR family [2] designed to tolerate Byzantine failures. A system employing this implementation is proved to tolerate up to n/3 failures, given n the total number of nodes. PBFT algorithms are also proved to handle up to 80000 messages per seconds. After this implementation researchers have designed new algorithms to implement the performance and scalability. Examples of such improvements are the Randomized BFT like Hybrid BFT solutions or XFT.

3.4 Hyperledger Fabric

As introduced in section 3.2.2, Hyperledger1 a project founded in 2015 by the Linux Foundation to grow and improve blockchain-based technologies through cross-industry interaction. It is a fully open source initiative to promote the adoption and advancement of blockchain thought a community process with the final objective to generate common technological standards. One of the outcomes of this initiative is Hyperledger Fabric (HL Fabric), a modular permissioned blockchain.

The following three sections give a basic overview of the most important notions that one should have in order to understand design, qualities, and drawbacks of the thesis project.

3.4.1 Privacy and identification

Identification is not only essential to control who can access the network, but also the fine-grained permissions that an entity has over the resources. HL Fabric adopts a Public Key Infrastructure (PKI) model where every active entity must hold a public-private key pair to sign and verify messages, and an identity in the form of a digital certificate to access and interact with the system. A digital certificate is a document binding a user public key to its actual identity. It contains user's information like name, employer organization or company division, as well as technical details such as public key, signature algorithms, and validity. The X.509 standard2 is the most common type of certificate and is the default one adopted in HL Fabric as it can encode more structured information. A digital certificate is issued and signed by a Certificate Authority (CA). By signing the certificate, everyone that trust the CA also trusts the bind between someone else's public key and identity.

An HL Fabric network usually has multiple CAs for fault tolerance, control and performance purposes. A CA instance can be either a Root CAs or Intermediate CAs. This duality allows creating flexible and complex certification infrastructures where each organization holds its own CA (or chain of CAs) that issues certificates for its members. Intermediate CAs certificates are valid across multiple organizations since they are issued by other Intermediate or Root CAs thereby allowing a distributed chain of trust security against faults.

Once certificates, keys, and CAs are set up, the mechanism that enforces the rules and checks the identities is the Membership Service Provider (MSP). It identifies which CAs are trusted to define the members of a domain [1]. Every to sign messages and transactions, authenticate users, define roles, and verify permissions and privileges within the context of a channel. node in the network maintains a local MSP, a software component that enables the node

3.4.2 State and roles

As mentioned above, a network is formed by different organizations that have control over a set of nodes. In HL Fabric there are three types of nodes: peer, client, and ordering-service-node. Each node has a specific set of functions that were designed to achieve modularity, flexibility and better performance and scalability.

Peer nodes are the most fundamental element of a Fabric system because they hold instances of one or more ledgers, and copies of smart contract source codes. The number of ledgers held by a peer depends on the number of channels joined by the organization that controls it. Each HL Fabric ledger consists of two data structures: the world state and the transaction log. Each peer holds a full copy (instance) of these two structures. The former is the current ledger state; it contains the latest variables values in a key-value database. The database is pluggable and can be either Apache CouchDB3 or LevelDB4. It is designed for fast access to the data without the need to traverse the entire linked list of blocks to compute the current value. The latter has the

role to preserve the blockchain history by recording all the values updates to the keys used in the network. Applications must connect to a peer to access the ledger state and invoke smart contracts.

The peers cooperate with the ordering-service-nodes, or just "orderers". These are the elements forming the ordering service, which has the goal to establish a total order of transactions: the transactions endorsed by a peer are sent to the ordering service, which collects them, puts them into a block and broadcasts it to all the peers in the channel. The peers then verify and commit the updates.

3.4.3 Consensus in Hyperledger Fabric

Unlike the permissionless implementations, the consensus in fabric does not correspond to a well-defined algorithm like PoW but is an overreaching process that goes beyond the simple agreement on the order of transactions.

As mentioned above, the core of the consensus process consists in the atomic broadcast offered by the ordering service. In the past years, the HL Fabric research team proposed different implementations for the service: the version v0.6 employed a Byzantine Fault Tolerant consensus in the form of PBFT, whereas the versions after v1.0 present an ordering service based on Apache Kafka6 and Zookeeper, which uses replication to provide strong consistency and high availability. However, this implementation supports crash faults but not Byzantine faults. To overcome these problems, researchers have proposed a mechanisms based on the BFT-SMART state machine replication7 and consensus library.

Regardless the implementation of the ordering service in HL Fabric, the consensus is a process that follows an execute-order-validate pattern which requires the transaction flow to be divided into three respective main steps: proposal, packaging, and validation. Therefore, to really understand the consensus in fabric, it is necessary to describe the entire transaction flow.

Transaction flow

The system needs to run at least a peer node and an ordering service (that would form the minimum viable fabric blockchain distributed network). The transaction creation and the steps are orchestrated by the fabric client, an SDK to interact with the network currently available for Java and NodeJs runtime environments. Before interacting with the fabric, the user/application must have created and stored the certificates; joined a channel and connected with a peer. The certificates will be necessary for authenticating the entity and signing the transactions.

As illustrated in figure 3.2 and described in [1], the client, using the SKD, initiates the process by (1) sending a transaction proposal to the network. The proposal in nothing more than a smart contact function invocation. The SDK signs the proposal with the user’s cryptographic material and forwards it to the peers via gRPC.

The contacted peers receive the proposal and check it (2). The checking includes signature and ledger state verification. After the checking, each peer executes the chaincode function and saves two states: one is the READ set which holds the key-values accessed by the function call while the second is the WRITE set which holds the key-values that have been added or modified. This two sets as well as the peers signatures are sent back to the client as proposal response.

Upon reception of a proposal response, the client checks both the signature and that the responses are the same (3). If the endorsement policies have been fulfilled, the client broadcasts the transaction and the endorsements to the ordering service (4). The ordering service has the sole scope of atomically broadcast the transaction to the entire network. To do so. It takes the transactions in chronological order, puts them into a block and forwards it.

The block is sent to the entire network using a gossip protocol. Each receiving node validates each transaction by enforcing the policies and checking the read and write sets (5). Then each peer appends the block to the ledger and updates the state database with the new key values (6). Finally, an event is emitted to inform the client that the transaction has been committed/not committed to the ledger

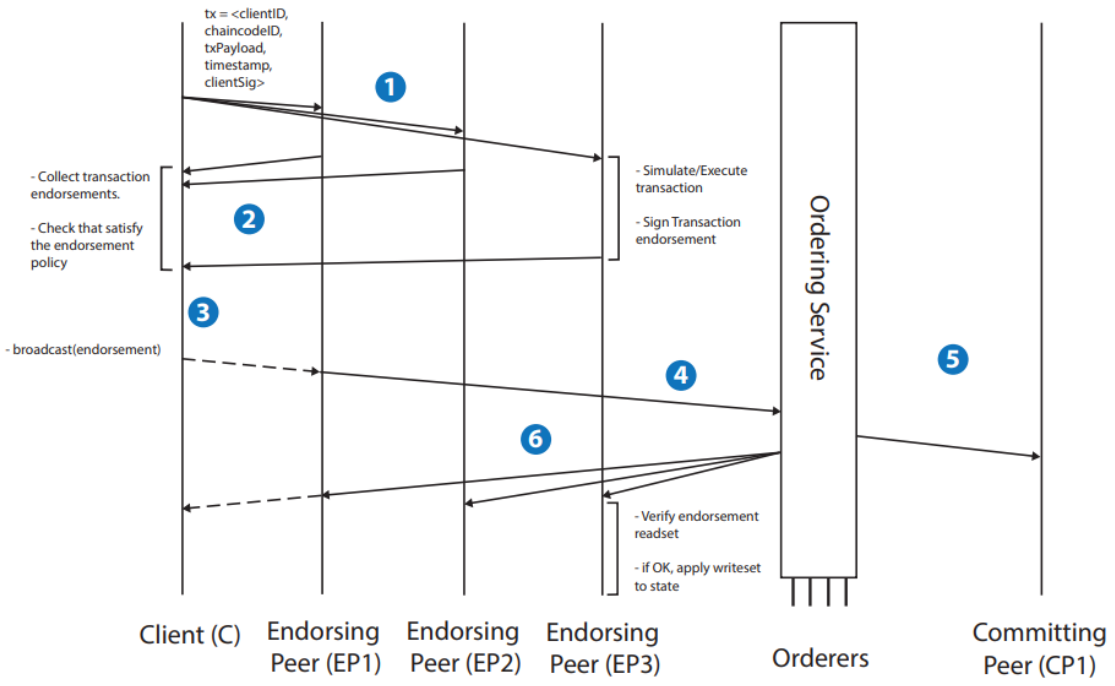


Figure 3.2: Hyperledger Fabric transaction flow.

Image adapted from[10].

4. USECASE OVERVIEW

Electronic Health Records (EHRs) play a critical role in modern healthcare by storing patient information, including medical history, diagnoses, prescriptions, and treatment plans. However, accessibility, security, and reliability of EHRs become major concerns, especially in emergency situations like natural disasters (earthquakes, hurricanes, and floods).

Traditional EHR systems are often centralized, relying on local servers or cloud-based infrastructures. In large-scale crises, these centralized systems can fail due to network disruptions, power outages, or cyberattacks, leading to delayed medical response, misinformation, or complete loss of data access.

To address these issues, permissioned blockchain technology, particularly Hyperledger Fabric, offers a decentralized, tamper-resistant, and failure-resistant approach to managing EHRs. This case study explores the feasibility, benefits, and limitations of blockchain-based EHR systems in handling health records during emergency situations.

4.1 Goal

The case study aims at designing a prototype for a simple blockchain based application for EHRs that satisfies some requirements like information privacy, traceability, secure information access and sharing in a decentralized fashion. The prototype will be implemented with Hyperledger Fabric1 (HL Fabric) and will serve also as an access control system to manage identities, provide traceability and preserve the privacy of users and patients. The system could potentially be used as a backup for health companies and be used in extreme situation to ensure the continuity of treatment while preserving privacy and confidentiality.

The system will be tested in a simulation of a real disaster: the Haiti earthquake of 2010. The goal of the tests is to demonstrate how the blockchain network can tolerate disruption and partitions. Analyze how this system performs and reacts to an extreme situation where a part of the nodes in the network crashed or start presenting a Byzantine behaviour caused by server failures. The system must allow the patients and practitioners to share and access EHRs and be able to detect and react to the crisis situations by changing the network policies and allowing new nodes representing the rescuers and humanitarian help. Moreover, it will need to behave correctly in the presence of malicious nodes

4.2 Scope

The scope of this case study focuses on the feasibility, advantages, and challenges of using permissioned blockchain (Hyperledger Fabric) for managing Electronic Health Records (EHRs) during emergency situations caused by natural disasters. It encompasses the following key areas:

4.2.1 Technological Scope

- Blockchain Integration: Exploring how permissioned blockchain can enhance security, privacy, and accessibility of EHRs.
- Hyperledger Fabric Implementation: Using this framework to develop a prototype for decentralized EHR management.
- System Performance Analysis: Evaluating throughput, latency, memory, and CPU usage under real-world stress conditions.
- Security & Fault Tolerance: Assessing the system's resilience to malicious nodes and unauthorized access attempts.

4.2.2 Healthcare Application Scope

- EHR Accessibility in Emergencies: Ensuring uninterrupted access to critical patient data during crises like earthquakes, hurricanes, and floods.
- Data Integrity & Privacy: Maintaining confidentiality while allowing authorized access to multiple healthcare providers (hospitals, pharmacies, clinics).
- Decentralized Decision-Making: Enabling real-time collaboration among doctors, paramedics, and hospitals for better crisis response.

4.3 Architecture options and design decisions

This section discusses the decisions that should be made when it comes to design a blockchain system for the healthcare. The primary design decision is where to store the data, which, in this case, corresponds to how and where to store health records. The solutions are storing the record entirely on the blockchain (on-chain) or outside the blockchain (off-chain).

The decision to store records off-chain depends on different reasons. First of all, the data can grow indefinitely thus making the replication across all the nodes expensive and memory intensive; therefore, it is simpler to propose a solution that integrates and interface with current EHR management systems rather than change to an entirely new method to store data. These solutions employ blockchain to control the accesses and keep track of modifications and interactions: the ledger stores hash pointers that are signed with the owner's private key. The pointers point to the location where the file/record is actually stored. The main drawback of this type of solutions is that they

cannot guarantee the accessibility of the records: in fact, if the server that stores a record is under maintenance or has crashed, the information cannot be accessed.

The other extreme is the on-chain storage: this allows to have full control over the records; achieve higher resilience and availability as a result of full replication; lower request latency; and higher security because the application does not rely on external and potentially not secure, storage systems.

Assuming that redundancy is critical in disasters scenarios where the source of the data might be destroyed or off-line, the proposed prototype stores the metadata and textual information of an EHR on the blockchain. Media files including images, videos, x-rays are saved in external storage (handled by one or more hospitals or health organizations) and referenced to by using URI or links. By keeping files with large extensions off-chain it is possible to reduce the network traffic and the overall blockchain disk space usage. Moreover, defining the resource location(s) in a unique place will prevent unnecessary replication and increase the data management capabilities. Although the media files could become unreachable, yet this solution allows accessing critical textual information that could be life-saving in case of emergencies. This decision can be considered hybrid since it combines the desirable characteristics from the off-chain and on-chain solutions.

4.3.1 How to detect disasters

As introduced above, an EHRs management system based on blockchain needs to work both during the day to day operations and the emergency situations. Each of these cases has a different set of security and performance requirements. Therefore, there should be a mechanism to change policies and functionalities of a blockchain network depending on the state of the system. Before that, it is necessary to define the possible states that a system can present as well as the criteria and rules to change from one state to another.

Although different states can come in succession during the life of a system, the simple prototype implemented in this case study presents two primary states: normal and emergency. In the former, each user needs to be authenticated and needs to have the authorization to access an EHR associated with a patient. The authorization must be asked directly to the patient through the application. Thanks to the system, the patient has the full ownership of her record and can decide who can access it. When someone changes the information contained in a record, the transaction needs to be approved by at least two organizations in the blockchain network.

In the latter state, new organizations need to be able to join the network despite the failure of other nodes; the requests need to be processed promptly and securely. In such scenario, each authorized party has the right to access the EHRs and the modifications need to be approved by at least the 50% of the network or by at least one hospital. This set of policies allows protecting the data from free modifications by

the new organization (aid, volunteers, non-governmental or humanitarian organizations) with the necessary approval of at least a hospital or the government.

4.3.2 Data Model

Electronic medical record systems have been object of legal and regulatory initiatives carried by different nations and institutions. This regulatory initiatives have the aim to achieve interoperability of electronic health record systems and cover multiple aspects and stakeholders. For example, the European Commission is carrying a so-called eHealth Action Plan since 2008 that established many initiatives including the eHealth European Interoperability Framework (eEIF).

Many reports provide rules and guidelines on architectures, technical and semantical standards as well as interoperability requirements.

5. IMPLEMENTATION

In order to model and test an health network during disasters, it is first necessary to model the day to day operations of a normal system. In fact, the prototype is built keeping in consideration the number of active hospitals in Haiti before the earthquake and creating a scaled network of 5 organizations. Each of these five is represented by an HL Fabric organization holding two peer nodes, one orderer node and a custom CA to issue certificated to its members. The different components are processes running in each hospital's infrastructure. Furthermore, there is a 6th entity representing the Haitian Ministry of Health (figure 5.1). The hospitals can use the prototype both as a backup mechanism, to be integrated with their legacy systems, and as a full operating system, meant to substitute the current solutions. For the sake of the thesis, the former way will be considered.

As mentioned above, the disruption of the earthquake destroyed 8 hospitals and severely damaged 22: respectively the 16% and the 44% of the total (49). These percentages are reflected in the simulated failures of the prototype: 1 node out of 5 (16%) crashes and two nodes out of 5 present a Byzantine behavior (44%). The simulated disaster, as well as the real disaster, affected the 60% of the total number of organizations (figure 5.2).

Figure 5.1: HL Fabric normal network

Image adapted from [4].

Figure 5.2: Network crash and faults

Image adapted from[4].

5.1 Users and roles

The prototype allows different entities to access and operate with the system and the records. Each entity can be either an organization (hospital, health ministry or volunteer organization), someone belonging to one of the institutions (doctor, nurse, physician or volunteer) or even the patients themselves. The prototype supports the following roles:

• Hospital;

• Doctor;

• Government;

• NGOs (WHO, PAHO, field hospital);

• Patient (client)

Each entity has different importance and freedom within the system. This freedom reflects in the operations that an entity can perform: some have access and rights over the patient records, some have the faculty to modify the system policies and rules. The

HL Fabric network must ultimately approve the vast majority of these operations and rights through the consensus process. The roles can be listed as follows:

• Government: monitors the system and can change mode;

• Hospital: can create, retrieve, update, delete a record, ask for consent;

• Doctor: can read and update records, ask for consent;

• Patient: read its health record, give consent to doctor.

5.2 Operations

HL Fabric provides several mechanisms to ensure the security of its network: each organization controls the access through a CA; the policies regulate the interactions, and one or more pieces of code restrict the accesses and operations. The EHRs are expressed as JSON files, and the entities interact with them through a smart contract that represents and guard the data model. In fact, not only the contract represents the EHR but also defines the operations that can be performed over it.

The smart contract implemented with the prototype provides a set of primary functions to interact with the EHR listed as follows:

• Create a health record:

createRecord({key: value}): patientId

This function accepts a key-value asset that could be either a JSON file or Javascript object. It returns a string containing the patient identification. The identification is currently the Swedish personal identity number, a 12-digit number serving different purposes: from taxation to health.

• Retrieve a record:

getRecord (patientId): medicalRecord

Given a patient identification number, this function returns a JSON file containing the EHR corresponding to the id.

• Update a record:

updateRecord (patientId, { key : value }): boolean

This function allows a patient to modify the content of her record: it accepts the patient id and the field that must be modified, and returns the the boolean result of the operation.

• Delete a record: deleteRecord (patientId): boolean

This function allows to remove a record given the patient id.

• Request for treatment:

requestForTreatment (patientId, doctor): Promise

A practitioner (doctor, physician etc.) or an organization can request to access or create a record belonging to a patient. The patient gets notified and can either accept or decline the request.

• Treat patient:

treatPatient (patientId, diagnosis, doctor, treatment): ,! boolean

The doctor that has the right to access a record, can update its information through this function.

• Give consent:

giveConsent (patient, doctor): boolean

With this function, the patient can grant someone the right to access and modify her record. The entity can be either an organization or someone representing the organization such as a doctor or a nurse.

5.3 Switching from normal mode to disaster

As discussed in chapter 4, there are different approaches to detect and change the state of the system during an ongoing mass crisis. The prototype does not have a mechanism to identify an emergency automatically: in fact, a user under the Haitian Ministry of Health organization has to execute a particular system transaction that modifies the policies and the rules of participation. This means that the Ministry of Health can accept new members within the HL Fabric channel in case of an emergency. The channel, typically formed by all the health organizations and used for the day to day operations, can be then joined by members like NGOs and volunteers organization.

The state-changing transaction is illustrated below

setMode (govt_sign, State): boolean

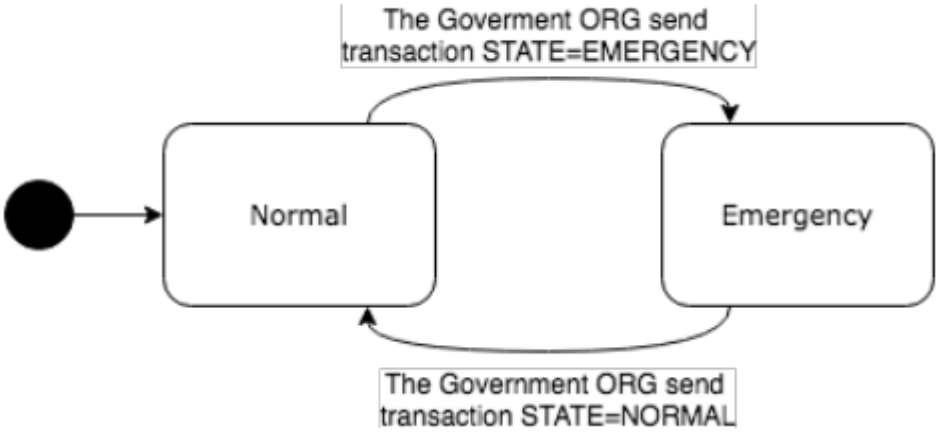


Figure 5.3 System state change

The systems switches from normal state to emergency state upon reception of a particular transaction from a user that belongs to government organization (Ministry of Health)

6. BENEFITS

Some of the benefits that a permissioned blockchain solution can provide to the healthcare sector, and to EHRs in particular, can be listed as follows:

• Security: a blockchain is secure by design. In fact, under certain conditions, the information stored on the ledger are tamper-proof ;

• Resiliency: a blockchain network is able to reach consensus and operate correctly also in case of Byzantine failures;

• EMR sharing: through encryption and digital signature, it is possible to securely share information;

• Avoid scattered information: the data is stored or referenced on the blockchain, that becomes a source of truth;

• Allow data access: the data can be accessed and modified only by trusted entities, and the modifications are approved through consensus;

• Control accesses: in the same way, the accesses to a permissioned network is regulated by policies and rules set by the parties;

• Enable doctors and rescuers to access life-saving information without compromising the privacy and security of the person and the system.

A permissioned blockchain could be used by hospitals and government both as backup system and as an operating system in case of disruptions and emergencies. It could grant easy access to information also to NGO and humanitarian help while preserving the privacy and confidentiality of patients, and keeping medical records secure and updated. It is also important to deny access after emergency to people and associations that should not have access to the information.

7.CHALLENGES

The results obtained from the local tests should be interpreted considering the technical and practical limitations. These, in fact, affected the prototype design and deployment, the test cases, and the subsequent discussion. The most important limitation derives from the difficulty to get free access to the cloud resources necessary to carry out additional tests. This deficiency influenced the extension and completeness of the testing and assessment process. Therefore the results were derived based on local simulation environments distributed in a local access network: the measured latencies do not take into consideration network delays and crashes.

Another limitation was represented by the relative novelty of HL Fabric, its composition of many different complex modules, and the lack of advanced knowledge or specific resources. These restrictions made difficult to improve the extension of the blockchain prototype as deploying and debugging was extremely complex.

Despite these limitations, it was still possible to assess how the disruption caused by a natural disaster could affect the latency and transaction throughput of an HL Fabric permissioned blockchain network.

8.CONCLUSION

In the context of blockchain for healthcare, the purpose of this case study is to provide a study to understand whether permissioned blockchain implementations could be of any benefit to managing health records in emergency situations caused by natural disasters. In fact, the current literature on blockchain lacks performance analyses, and, in particular, the literature on EHRs management systems lacks investigation of emergencies during natural disasters. After the design and implementation of a basic EHRs management system and the execution of a set of test cases, it was possible to discuss the benefits and trade-offs that a blockchain solution entails. The discussion focused on performance indicators like throughput, latency, success rate, consensus policies and network size measured in two different scenarios: one reflecting the normal operations of a blockchain network and another simulating a crisis caused by an earthquake.

The prototype for a simple blockchain based application for EHRs had to satisfy some requirements like information privacy, traceability, secure information access and sharing in a decentralized fashion. The prototype has been designed and implemented at IBM Sweden using Hyperledger Fabric to serve as an access control system to manage identities, provide traceability, and to be used in extreme situation to ensure the continuity of treatment while preserving privacy and confidentiality of users and patients. The system has been tested in a simulation of a real disaster, the Haiti earthquake of 2010, to investigate how a blockchain network (in particular, the HL Fabric network) can tolerate disruption and partitions; and analyze how it performs during an extreme situation where a part of the nodes in the network crashed or start presenting a Byzantine behavior. The prototype has proven to allow the patients and practitioners to share and access EHRs and be able to detect and react to the crisis situations by changing the network policies and allowing new nodes representing the rescuers and humanitarian help. It has also proven to behave correctly in the presence of malicious nodes.

8.1 Future work

The prototype provides a simple blockchain-based application for EHRs and satisfies some fundamental requirements. Nevertheless, in order to be employed by a network of hospitals, it needs more functionalities and roles as well as more extensive tests on cloud infrastructures. The tests on the cloud could extend the local results and provide a thorough analysis of how a more extensive network and the geographical distribution affect the performance of a blockchain. It will be also possible to investigate other states like pre-disaster or post-disaster to characterize better the response to a crisis. This type of analyses is meant as a natural continuation of the thesis work. The future work also includes the extension of the current smart contact and the addition of others to improve the lookup and support the additional features required by an EHRs management solution.

9.SDG'S ADDRESSED

SDG 3: Good Health and Well-being

How? Ensures secure, tamper-proof medical records, leading to better patient care and efficient healthcare management.

Impact: Reduces medical errors, enhances data sharing, and ensures patient data privacy.

SDG 9: Industry, Innovation, and Infrastructure

How? Uses blockchain technology to create a transparent, efficient, and decentralized healthcare system.

Impact: Improves healthcare infrastructure by reducing dependency on centralized databases and preventing data breaches.

SDG 16: Peace, Justice, and Strong Institutions

How? Enhances data security, integrity, and privacy, preventing unauthorized access and fraud in medical records.

Impact: Ensures accountability and trust in the healthcare system through decentralized governance.

10. REFERENCES

[1] Vitalik Buterin. Proof of Stake FAQ. 2016. URL: https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQ

[2] Christian Cachin and Marko Vukolic. “Blockchain Consensus Protocols in the Wild”. URL: http://arxiv.org/abs/1707.01873.

[3] George F Coulouris, Jean Dollimore, and Tim Kindberg. Distributed Systems - Concepts and Design (5th Edition). Pearson Eduction, May 2011. ISBN: 978-0132143011.

[4] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. “The Bitcoin Backbone Protocol: Analysis and Applications”. In: Advances in Cryptology - EUROCRYPT 2015. Ed. by Elisabeth Oswald and Marc Fischlin. Vol. 9057. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 281–310. ISBN: 978-3-662-46802-9 978-3-662-46803-6. URL: http://link.springer.com/10.1007/978-3-662-46803-6_10

[5] Rachid Guerraoui and André Schiper. “Fault-tolerance by replication in distributed systems”. en. In: Reliable Software Technologies — Ada-Europe ’96. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, June 1996, pp. 38–57. ISBN: 978-3-540-61317-6 978-3-540-68457-2. DOI: 10.1007/BFb0013477. URL: https://link.springer.com/chapter/10.1007/BFb0013477

[6] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf. 2008. URL: https://bitcoin.org/bitcoin.pdf

[7] A Next-Generation Smart Contract and Decentralized Application Platform. 2014. URL: https://github.com/ethereum/wiki/wiki/White-Paper

[8] Leslie Lamport, Robert Shostak, and Marshall Pease. “The Byzantine Generals Problem”. URL: http://portal.acm.org/citation.cfm?doid=357172.357176

[9] MDPI, "Advances in Sensor Technology," *Sensors*, vol. 25, no. 4, p. 1030, 2025. URL: https://www.mdpi.com/1424-8220/25/4/1030

[10] GeeksforGeeks, "Consensus in Hyperledger Fabric," *GeeksforGeeks*, Feb. 5, 2024. URL: https://www.geeksforgeeks.org/consensus-in-hyperledger-fabric/

11. APPENDIX A

https://drive.google.com/drive/folders/1-8BfyoWHNpKzjtUzMtTGPEpZwNFNc1Ac?usp=sharing