TÉCNICO LISBOA

Enabling Enterprise Blockchain Interoperability

A Bills of Exchange case study

Bernardo Carreira dos Santos

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. André Ferreira Ferrão Couto e Vasconcelos
Prof. Sérgio Luís Proença Duarte Guerreiro

Examination Committee

Chairperson: Prof. Rui Filipe Fernandes Prada
Supervisor: Prof. André Ferreira Ferrão Couto e Vasconcelos
Member of the Committee: Prof. João Fernando Peixoto Ferreira

October 2021

Acknowledgments

I would like to thank my parents for providing me with whatever I needed to be able to go through university, and for encouraging and supporting my decisions along the way.

I would also like to say thanks to my friends and colleagues who helped me throughout this 5 year journey.

Thanks to Rafael Belchior for the work that was already laid down, and for helping me with any questions I had during the dissertation.

Thanks to INCM for providing an use case, specially Alberto López and Silvia Garcia.

Last but not least, thanks to my supervisor Prof. André Vasconcelos and co-supervisor Sérgio Guerreiro for the opportunity and help with the Dissertation.

Abstract

In recent years, with the advances made in technology overall, there is a need to modernize the processes currently in place, especially in governmental organizations. This increases the number of different blockchain infrastructures that need to interact with each other, this is the reason that blockchain interoperability is so important and is a problem that needs to be solved for adoption to continue. In this document, we will study blockchain technology, as a possible fit as a solution for existing issues in these systems. A comparison between available blockchain infrastructures is shown to evaluate its applicability in real-world scenarios. We also have a comparison between the different interoperability solutions available, to be able to come to an appropriate solution regarding blockchain interoperability. The solution proposed in this thesis leverages blockchain technology to replace the current paper support model in place for Bills of Exchange with a digital model focused on the integrity and security of the data handled by the system, which is guaranteed by blockchain technology. Our solution is deployed using a blockchain infrastructure, Hyperledger Fabric, with nodes representing the multiple institutions involved. Interoperability between these systems is handled with resort to Hyperledger Cactus creating a network of relayers between the multiple blockchain infrastructures interacting. During our evaluation, we achieved a throughput of 114 Transactions per second (TPS) with an average latency of 0.15 seconds which shows that our solution would be capable of supporting the use of the current bills of exchange system. Regarding our interoperability solution, we have concluded that Cactus offers great flexibility for diverse interoperability use cases, and provides a direct way to further improve its compatibility, while in some niche use cases there might be better solutions that is the trade-off we are committing to by using Cactus.

Keywords

Blockchain; Interoperability; Hyperledger Cactus; Hyperledger Fabric; Bills of Exchange;

Resumo

Com os recentes avanços tecnológicos feitos no geral, existe uma necessidade de modernizar os processos actualmente usados, especialmente em organizações governamentais. Isto aumenta o número de infrastruturas de blockchains diferentes que necessitam de interagir entre si, esta é uma das razões pela qual a interoperabilidade é um problema tão importante que precisa de ser resolvido para que a adoção desta tecnologia possa continuar. Neste documento, iremos estudar tecnologia de blockchain, como uma potencial solução para os problemas existentes nestes sistemas. Apresentamos uma comparação entre as várias infrastruturas de blockchain disponíveis para avaliar a aplicabilidade em cenários reais. Comparamos também as diferentes soluções de interoperabilidade existentes, com o objetivo de chegar a uma solução de interoperabilidade apropriada. A solução proposta nesta dissertação tem como objetivo substituir o modelo atual de suporte em papel para Livranças, que é gerido pela INCM, por um modelo digital focado na integridade e segurança dos dados manipulados pelo sistema, garantido usando tecnologia blockchain. A nossa solução foi desenvolvida usando uma infrastrutura de blockchain, Hyperledger Fabric, com *nodes* a representar as multiplas organizações involvidas. A interoperabilidade entre estes sistemas será gerida recorrendo ao Hyperledger Cactus, criando uma rede de *relayers* entre as várias infrastruturas de blockchain a interagir entre si. Durante a avaliação conseguimos alcançar uma taxa de transferência de 114 transações por segundo com uma latência média de 0.15 segundos, o que mostra que a nossa solução será capaz de suportar o uso verificado no sistema atual de livranças. Em relação à nossa solução de interoperabilidade concluímos que o Cactus oferece uma ótima flexibilidade para diversos casos de uso de interoperabilidade, e fornece uma forma direta de no futuro aumentar a compatibilidade da solução, em alguns casos de uso mais nicho existirão melhores soluções mas esse é o compromisso que estamos a fazer ao usar o Cactus.

Palavras Chave

Blockchain; Interoperabilidade; Hyperledger Cactus; Hyperledger Fabric; Livranças;

Contents

List of Figures

List of Tables

Acronyms

API	Application Program Interface
HTTP	Hypertext Transfer Protocol
TCP	Transport Control Protocol
UDP	User Datagram Protocol
INCM	Imprensa Nacional - Casa da Moeda
NPoS	Nominated Proof of Stake
DPoS	Delegated Proof of Stake
PoS	Proof-of-Stake
PoW	Proof-of-Work
BFT	Byzantine Fault Tolerant
EVM	Ethereum Virtual Machine
dApp	Decentralized Application
IBM	International Business Machines
CCCP	Cross-chain Communication Protocol
CBCP	Cross-blockchain Communication Protocol
CC-Tx	Cross-chain Transaction
CB-Tx	Cross-blockchain Transaction
CB-dApp	Cross-blockchain Decentralized Application
CC-dApp	Cross-chain Decentralized Application

IoB	Internet of Blockchains
BoB	Blockchain of Blockchains
SPV	Simplified Payment Verification
HTLC	Hashed Timelock Contracts
ILP	Interledger Protocol
VESes	Verifiable Execution Systems
UIP	Universal Inter-Blockchain Protocol
USM	Unified State Model
HSL	Hyperservice Programming Language
API	Application Programming Interface
HTTP	Hypertext Transfer Protocol
ATA	Autoridade Tributária e Aduaneira
BP	Banco de Portugal
ABAC	Attribute-Based Access Control
PLCD	Portal de Letras de Créditos Digitais
CA	Certificate Authority
UUID	Universally Unique Identifier
GDPR	General Data Protection Regulation
DLT	Distributed Ledger Technology
GUI	Graphical User Interface
SUT	System Under Test
TPS	Transactions per second
RPS	Reads per second
NBI	Northbound Interfaces
YAML	Yet Another Markup Language

GCE Google Cloud Engine

GCS Google Cloud Services

AWS Amazon Web Services

BLP Business Logic Plugin

SSIBAC Self-Sovereign Identity Based Access Control

1

Introduction

Contents

Blockchain technology has seen a massive increase in adoption and exposure partly because of being highly associated with cryptocurrencies, but also because of the promise of providing decentralization and distribution of trust to systems where a single point of failure is the norm. This has caught the attention of organizations, especially governmental organizations that can take advantage of blockchain technology innovation to improve the efficiency, security, availability, and auditability of their systems. As these systems require interaction between each other, it creates a major concern regarding interoperability and more specifically blockchain interoperability.

Imprensa Nacional - Casa da Moeda (INCM) is the Portuguese mint and national press, owned by the Portuguese Government and administratively subordinated to the Portuguese Ministry of Finance. It is also responsible for many projects, where the focus of this thesis and the use case provided by INCM was to create a digital platform for bills of exchange using blockchain technology to first work alongside the current paper support model eventually completely replacing it, to explore the interoperability required for such a system. The need for a system like this became apparent with the recent necessity for lockdown in Portugal because of COVID preventing the use of the current system for Bills of Exchange as it required physical dislocation to the appropriate entity to obtain them.

With this in mind, we introduce a bill of exchange blockchain solution implemented using Hyperledger Fabric[1], a permissioned blockchain designed for enterprise solutions, accompanied by an interoperability solution that takes advantage of Hyperledger Cactus[22] to provide the flexibility and compatibility required associated with such a system. Our Fabric solution handles all of the life cycle associated with a bill of exchange asset, taking into account the organizations involved in this respective life cycle and permissions associated with each organization and member, this solution serves as a way to explore the multiple interoperability use cases associated with bills of exchange. As there can be several different implementations of bills of exchange systems, there is a requirement that our solution can seamlessly interact with those systems, this is provided by leveraging Cactus flexibility and compatibility as an interoperability solution.

To provide an initial evaluation of our bills of exchange blockchain solution we are going to be using Hyperledger Caliper that provides a framework for load-testing blockchain infrastructures. Regarding the evaluation of our interoperability solution, we are going to be discussing the complexity, flexibility, and compatibility that Cactus can offer.

1.1 Objectives

As with most governmental processes that have been in place for so long, there is a need to modernize these processes, in this case, deploy a digital model that serves as a database system throughout the life cycle of bills of exchange, while facilitating all of the currently supported operations. These operations

also require interoperability between other implementations of bills of exchange, as there is no standard for how these systems are implemented each country might use a different solution. This solution looks to solve the following problems that are known to affect solutions of this type:

- How can we maintain the interaction that these governmental systems and processes, such as the interaction between the Portuguese and Spanish bills of exchange platforms, require while deploying a digital model with the goal of eventually totally replacing the current model?
- What are the risks and concerns associated with these digital transformations and how can we avoid them, while providing a secure and scalable way to handle these interoperability needs?

While most widely known blockchain applications are known for their decentralized nature, the authority of the organizations involved such as INCM, Banco de Portugal, and Autoridade Tributária e Aduaneira is taken into consideration, as these require some control and management of the system. Taking this into account, there is a need to study the various approaches that can be taken and address their implications in such a system, while identifying the possible problems associated with them.

This thesis explores several diverse interoperability use cases associated with this system to analyze the flexibility provided by such an interoperability solution to show that it can be applied to multiple systems similar to the one we are working with. Our goal is to provide a base solution for future works, to entice other organizations to use blockchain technology for their systems. To do this we are going to try and answer several questions:

- Is it possible to design a interoperability solution that provides interoperability between heterogeneous blockchain infrastructures, this being permissioned and permissionless models?
- Can we provide a solution that does not require adaptation and development in the blockchain infrastructures interacting in the interoperability use case?
- Are we able to address the interoperability requirements without compromising the security and robustness of the solution?

These are questions that are crucial not only for the use cases we are going to be exploring but to also prove that it can be used for other systems, as interoperability is a key requirement for multiple different systems.

1.2 Document Structure

In this document we start by exploring the existing blockchain technology implementations, introducing both permissioned and permissionless blockchain infrastructures with resort to examples of both implementations to present their characteristics, as a way to choose the most suitable infrastructure for our use case. Then we present the state of the art regarding blockchain interoperability solutions taking into

account the context of the use case provided and focusing on solutions that have a proof of concept. The context of INCM's use case is then presented and the functional and non-functional requirements for the solution are set, with this in mind we explain our design decisions and implementation steps taken for the problem at hand. We then explain our evaluation methodology for our solution, we also present and discuss the results obtained, and lastly, we explain the following steps for our work and conclude what we have presented.

2

Background and Related Work

Contents

In this Chapter, we explore the required concepts for understanding this thesis, and the already existing array of solutions, their advantages and disadvantages and how those apply to our specific use case. We explore the the existing blockchain models those being permissioned and permissionless, and their respective consensus algorithm. And we then proceed to address the already existing blockchain interoperability solutions.

2.1 Blockchain

A blockchain is an append-only distributed ledger, focused on decentralization, transparency, and trust while taking into consideration security, privacy, and control[8].

To explain the core concepts of blockchain technology we use Bitcoin[23] as an example, as it is the first and most well-known blockchain technology project. A core concept of a blockchain is a *transaction* if we are referring to a financial application like Bitcoin or we can also call it a *record* if we are referring to a non-financial application, it consists of a collection of data. *Records* are grouped into *blocks* and those *blocks* form the *blockchain*. A *block* consists of the hash of the previous block and a timestamp that guarantees the append-only feature. For *blocks* to be appended all *records* within the block need to be validated.

A blockchain can also be seen as a network, where the participants are the computing nodes that communicate with each other in order to complete transactions and agree on the state of the network, but these nodes can be malicious. The process of agreeing on a state for the network is called *consensus* and there can be many *consensus* algorithms, in the context of Bitcoin the *consensus* algorithm is Proof-of-Work (PoW). In PoW we have transaction *validators*, which are also called *miners*, *miners* solve a difficult cryptographic puzzle to validate the hash of the transaction. Since the process of reaching consensus is expensive, the network rewards *miners* that solve the puzzle first. In the case of Bitcoin, the first *miner* that appends a valid block to the chain is rewarded with Bitcoin. This makes so attacks, where attackers attempt to append invalid blocks to the chain, are not worth it because of how expensive the process is, making the network resistant to such attacks. In Section 2.1.1 we talk about other algorithms used besides PoW, more in-depth.

Blockchains can follow two different schemes, either *permissionless* or *permissioned*, in Section 2.1.2 and Section 2.1.3 respectively we approach these two methods of implementing blockchain technology more in-depth. *Permissionless* blockchain might use different *consensus* algorithms and reward mechanisms, as in *permissionless* blockchains *miners* are not authenticated before they can partake in the network, so they rely on anonymous nodes to maintain the functionality of the network. For *permissioned* blockchains, *miners* are authenticated by a central authority before participating in the network. These two methods have different use cases depending on the desired properties for the network.

2.1.1 Consensus

Consensus algorithms for blockchain are developed as a way to solve two main problems[20], double-spending and Byzantine General Problems. Double-spending consists of reusing the currency in multiple transactions, and Byzantine General Problem which is a known problem in distributed systems, where a node that participates in the network can be attacked and start acting maliciously against the network while regular nodes need to continue obtaining consistent results regardless of malicious nodes.

PoW is the *consensus* algorithm used for Bitcoin. In PoW the fastest *miner* that solves the cryptographic puzzle gets to broadcast the block that will be appended to the chain, and collect their reward. The success of the solution of the puzzle is dependent on the previous block, the list of pending transactions in the network, and the current difficulty value of the network. The difficulty value is dynamically updated according to the current hash rate of the whole network. As the resources needed to append a block to the chain are proportional to the length of the chain and nodes trust the longest chain, this high workload is used as a way to guarantee the safety of the network. In order to interfere with the network, the attacker needs to control more than 50% of the hash rate of the network to ensure they are the first to append a block, this is known as the 51% attack.

As an example for Proof-of-Stake (PoS), we have Casper[6] which was initially designed to overlay the current PoW blockchain Ethereum, there are several variations of PoS such as Nominated Proof of Stake (NPoS) and Delegated Proof of Stake (DPoS). Contrary to PoW, in PoS physical resources are not used to append blocks to the chain, instead, blocks are agreed upon by participants based on their stake in the network, the voting power of each participant is proportional to their stake. PoS strengthens the blockchain as the value of the network increases since the process of tampering with the network becomes increasingly expensive proportionally to the network's value.

Byzantine Fault Tolerant (BFT) algorithms are another solution, popularized by the Tendermint[5, 15] project in blockchain technology. A blockchain using traditional BFT algorithms would suffer from poor scalability while offering better performance and finality than the mentioned PoW consensus algorithms[33]. BFT algorithms also require the nodes to be known by the network before participating in the consensus, needing a central authority to authenticate nodes, thus being more likely to be implemented in a Permissioned Blockchain.

The 3 consensus implementations are compared in Table 2.2, using the 3 blockchain implementations which use each consensus mentioned.

2.1.2 Permissionless Blockchains

Permissionless blockchains are public blockchains that require no authentication previous to engaging with the network. Bitcoin and Ethereum are two of the more widely adopted permissionless blockchains

and will be used as an example. These are typically self-sustainable networks without any central authority, with reward-based incentives for valid participation in the network, and normally open-sourced.

In permissionless blockchains, there are certain trade-offs between security, privacy, and performance, in most cases choosing security over the rest. For both Bitcoin and Ethereum, the consensus algorithm used is PoW, and as mentioned this type of consensus, while offering good scalability, lacks in performance. The ledger storing all the blocks, and thus all the transactions, is also available to be accessed and verified by anyone, sacrificing privacy.

Being open-source also allows for the network to develop itself through the community, in the case of Ethereum, while serving as a financial application Ethereum's goal was to also allow the development of other non-financial applications in the blockchain. The Ethereum blockchain has solidified itself as a platform for the development of smart contracts and Decentralized Application (dApp) through the use of their runtime environment called Ethereum Virtual Machine (EVM). While Bitcoin also allowed for a very primitive form of smart contracts, Ethereum improved and popularized it by creating a simple and accessible way to develop and maintain them. Smart contracts can be developed in several languages with the most common being Solidity, and then deployed to the EVM after being compiled into bytecode, a low-level programming language that can be run in EVM. Smart contracts allow the creation of certain conditions to be met for the completion of a transaction to occur, like requiring multiple signatures.

2.1.3 Permissioned Blockchains

Permissioned blockchains are private blockchains that require authentication of nodes before participating in the network, thus typically requiring a central authority. For this reason, permissioned blockchains are typically used by organizations that desire to take advantage of blockchain technology while safeguarding sensitive information. There is an obvious trade-off between control and decentralization when comparing permissioned and permissionless blockchains, the biggest advantage of permissioned blockchains being that the organization or organizations responsible for the management of the blockchain get to choose the business logic and business rules applied to the blockchain.

While permissioned blockchains can usually offer better performance, these are usually smaller networks which can result in a bigger risk for the network but also allow for easier identification of the malicious nodes involved.

A great example of a permissioned blockchain is Hyperledger Fabric[1], which will be addressed in more detail in Section 2.2. In Table 2.2 we compare both Permissioned and Permissionless projects, using the 3 blockchain implementation previously mentioned, which are the principal blockchains associated with these models.

	Hyperledger Fabric	Bitcoin	Ethereum
Consensus	Pluggable (PBFT typically)	Proof of Work	Proof of Worth (Ethash) Proof of Stake (Casper)
Accessibility	Private	Public	Public or Private
Permission Mode	Permissioned	Permissionless	Permissionless
Decentralization	Partially	Yes	Yes
Compute-intensive	No	Yes	Partially
Network-intensive	Yes	No	No
Scalability	Low	High	High
Throughput	High (for low scale)	Very Low	Low
Latency (Avg.)	100 ms (fow low scale)	10 minutes	12 seconds
Immutability	Low	High	High
Adversary Tolerance	33.33% Faulty Replicas	<25% Computing Power	<51% Stakes
Privacy	High	Low	Low
Smart Contract	Yes	Limited	Yes
Currency/Tokens	None but Tokens are possible	Bitcoin (BTC)	Ether (ETH) and Tokens are possible

Table 2.1: Comparison between Blockchain Implementations by Consensus/Permission Mode

2.2 Hyperledger Fabric

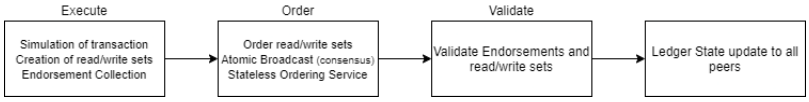

Figure 2.1: Hyperledger Fabric execute-order-state architecture

Hyperledger Fabric is an open-source blockchain platform, a project between the Linux Foundation and International Business Machines (IBM), which serves as a distributed operating system for the creation of customized permissioned blockchains[1] focused on enterprise solutions. Fabric is designed with a modular architecture that allows the use of pluggable implementations of several functions[7] offering a high degree of flexibility, confidentiality, scalability, and resiliency.

A distributed application in Fabric consists of 2 core components, the *chaincode* which is a smart contract that is program code implementing the application logic and runs during the execution phase, Fabric also has a system chaincode, which is run in the configuration channel. The *endorsement policy* is evaluated during the validation phase and is used for transaction validation. It can specify the endorsers for a certain transaction as a subset of *peers*.

The configuration channel stores the definition of the Membership Service Provider, consensus configurations, ordering service parameters, and rules for altering the channel's configuration. Each channel has a different ledger, as each channel enforces chaincode and data isolation. Channels allow the creation of a communication network between participants, giving these participants access to the transactions they have permissions to visualize. Besides channels, Fabric offers a solution for private data, where an organization or a group of organizations isolate their data from others. Organizations with permissions to access this data can endorse, commit, and query this data which is isolated from the

channel ledger. Hashes of the private data are used to go through the orderer, and it is disseminated via the peer gossip protocol instead of via blocks.

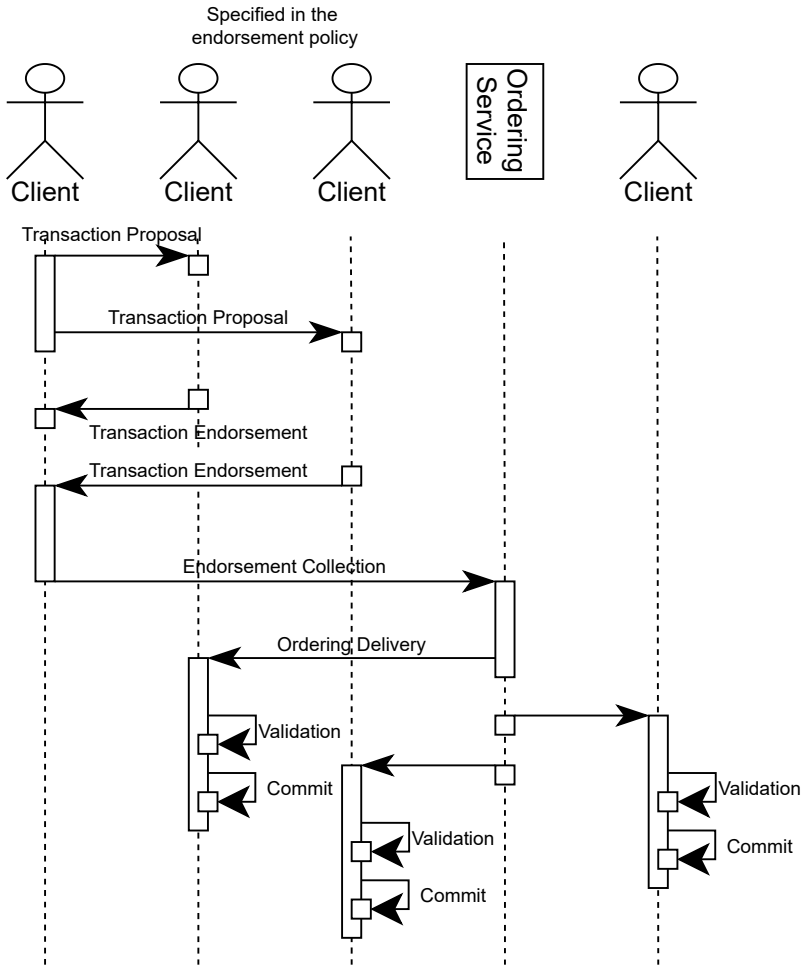

Figure 2.2: Hyperledger Fabric possible Transaction Flow from Androulaki et al. [1]

As Fabric follows an *execute-order-validate* architecture (Fig. 2.1), consensus in Fabric is broken into 3 phases as seen in Fig. 2.2, *endorsement*, *ordering*, and *validation*. A transaction proposal is created by a blockchain client, representing a member of an organization, and sent to endorsement peers, as defined in the endorsement policy. During the endorsement phase, the endorsers simulate the transaction proposal, producing a write-set, with the modified values and correspondent keys, and a read-set. This simulation is run in an isolated environment. With the endorsement created, it is sent back to the client which upon receiving enough endorsements creates the transaction and sends it to the ordering service. In the ordering phase, orderers check if the client who submitted the transaction proposal has the required permissions and produced blocks that contain the endorsed transaction ordered. The ordering allows the network to achieve consensus. The ordering service then broadcasts the blocks to peers who maintain the state of the ledger, through the ordering service or the peer gossip protocol. After this, we

get into the validation phase where peers firstly check if the transactions follow the endorsement policy. Then, for each transaction, the versions of the keys in the read-set are compared with the keys currently in the shared ledger. Transactions with non-matching versions are discarded from the block and the block is then appended to the head of the ledger[11].

Fabric supports pluggable consensus for all 3 phases. This allows applications with different requirements to use different endorsement, ordering, and validation plugins to satisfy those needs. It also does not require an associated cryptocurrency to serve as an incentive for the mining process by leveraging BFT consensus mechanisms, but a token can be implemented using chaincode, to represent an asset or value that can be exchanged between participants in the network.

The Fabric network, which is shown in Fig 2.3, is maintained by a group of *peers*, as Fabric is a permissioned blockchain, these are provided an identity by the modular *membership service provider*, this identity is created using a public key infrastructure to generate certificates which are tied to members or organizations. Each organization issues identities to its members.

Peers can take up 3 different roles, *Clients*, *Endorsers*, *Orderers*. *Clients* are those who submit transactions for execution, help in the execution phase, and broadcast transactions for the ordering phase, while also keeping a snapshot of the current state of the ledger. *Client* peers are not able to invoke chaincode functions. *Endorser* peers have access to chaincode, and when they receive a transaction proposal, they simulate the execution of the transaction in an isolated environment and based on that simulation's results they prepare a transaction endorsement to send to *Orderer* peers. *Orderer* peers receive endorsed transactions and assemble them into blocks while establishing the total order of all transactions, as each transaction contains state updates and dependencies computed during the execution phase. *Orderers* do not participate in the execution or validation of transactions, they propagate such blocks to *Client* peers, where the blocks are validated and committed to the shared ledger. Besides these 3 main roles peers can assume, Fabric also defines *anchor* and *leader* peers. *Anchor* peers act as an intermediary between their organization and an external organization. *Leader* peers assume the responsibility of distributing the transactions from the *orderer* peers to the *client* peers.

Fabric makes use of a key protocol for communication between peers, known as the *Peer Gossip*, which is responsible for broadcasting the results of the ordering phase for unsynched peers, which are peers that might have recently joined the network, or peers who had downtime. This data dissemination protocol ensures consistency and data integrity across all the nodes in the network.

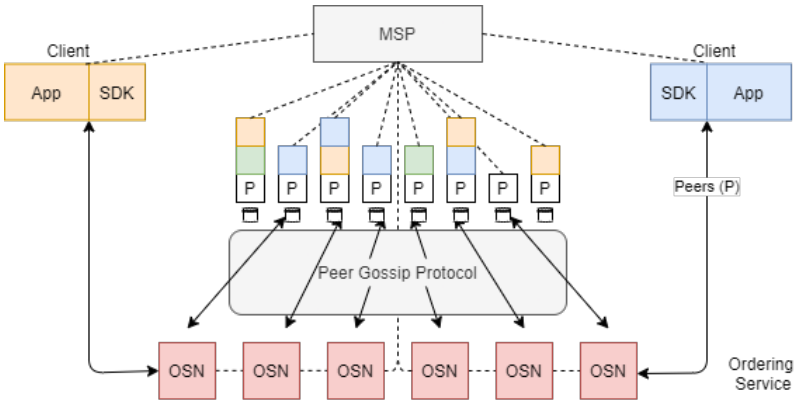

Figure 2.3: Hyperledger Fabric possible network from Androulaki et al. [1]

2.3 Blockchain Interoperability

In this Section first we define the core concepts of blockchain interoperability required to then explore the 3 different categories of solutions offering Blockchain Interoperability, these being Cryptocurrency-directed Approaches, Blockchain Engines, and Blockchain Connectors. In each of these categories, there is a different subset of solutions, this document tries to focus on those which had a proof of concept for a practical use case at the time of writing. The goal in this section is to explain how each solution attempts to solve the problem of interoperability, and its impact when it comes to interoperability to permissioned and permissionless blockchains, focusing on permissioned blockchains as that is the most common approach for blockchain infrastructure in enterprise applications of the technology.

This section explores the existing solutions while exposing the advantages and disadvantages of each solution in general and understand how certain solutions can be adequate for enterprise systems using blockchain technology, which require blockchain interoperability.

To define blockchain interoperability we first need to introduce some core terms which are required to understand and define blockchain interoperability.

Cross-blockchain communication, which involves two blockchains, a source blockchain where the transaction is initiated, and a target blockchain where the transaction will be executed. To define this process we have a *Cross-chain Communication Protocol (CCCP)* or a *Cross-blockchain Communication Protocol (CBCP)*, the difference between the two protocols is that the first specifies how two homogeneous blockchains interact to synchronize cross-chain transactions, and the latter focus on the interaction between two heterogeneous blockchains.

From this, we can define a *Cross-chain Transaction (CC-Tx)* and therefore a *Cross-blockchain Transaction (CB-Tx)*, where a CC-Tx is a transaction between homogeneous blockchains, such as transactions between EVM-based blockchains, and a CB-Tx is a transaction between heterogeneous blockchains for example a transaction between Hyperdledger Fabric and Ethereum as defined in Belchior et al. [4].

We also have *Cross-chain Decentralized Application (CC-dApp)* or *Cross-blockchain Decentralized Application (CB-dApp)*, which leverage CC-Tx or CB-Tx to implement its business logic.

An *Internet of Blockchains (IoB)* is a system "where homogeneous and heterogeneous blockchains can communicate to facilitate cross-chain transactions of value, data, and state transition", a definition from Vo et al. [32].

A *Blockchain of Blockchains (BoB)* is a system in "which a consensus protocol organizes blocks that contain a set of transactions belonging to CC-dApps, spread across multiple blockchains. Such a system should provide accountability for the parties issuing transactions on the various blockchains, as well as providing a holistic, updated view of each underlying blockchain", a definition from Belchior et al. [4].

Interoperability has several different layers, in blockchain interoperability, we are focused on technical and semantic interoperability. IoB uses *CBCP* to deliver transactions, enabling CC-dApps, thus granting technical interoperability, while the BoB approach allows CC-dApps to provide interoperability at the value level, by using *cross-blockchain dApp protocols* to provide consensus over a set of *cross-chain transactions*, granting semantic interoperability. We will then use the definition of Blockchain Interoperability from Belchior et al. [4] as follows: "The ability of a source blockchain to change the state of a target blockchain, enabled by cross-chain or cross-blockchain transactions, spanning across a composition of homogeneous and heterogeneous blockchain systems, the IoB."

2.3.1 Cryptocurrency-directed Approaches

In this section, we explore several approaches that are commonly applied to blockchains with an associated token, cryptocurrency. There are three main solutions in this area, being sidechains, notary schemes, and hashed timelocks.

Sidechains are typically a secondary chain that is connected to a blockchain, normally called the Mainchain, using a cross-chain communication protocol, for example, a *two-way peg* that allows for the exchange of assets between the mainchain and the sidechain. Sidechains are normally an approach with the goal of interoperability and scalability for the mainchain, a good example being Bitcoin and RSK[1], where Bitcoin is the mainchain and RSK the sidechain. RSK is a sidechain that not only improves scalability but also adds smart contracts to Bitcoin. RSK has its own token named RBTC which is pegged one to one to Bitcoin.

There are 3 different types of a *two-way peg* protocol, a *Simplified Payment Verification (SPV)*, a *centralized two-way peg*, and a *federated or multi-signature two-way peg*[4, 30]. A *two-way peg* consists of sending tokens from the mainchain to a specific address, those same tokens are locked up in

[1] https://www.rsk.co/

the mainchain and the same amount is then created on the sidechain, which can now be used in the sidechain. In order for the tokens in the mainchain to be released, the tokens in the sidechain need to either be locked up or destroyed. In *SPVs*, the transactions correspondent to the whole exchange of assets between the mainchain and sidechain are verified by a lightweight client[27] without the need for a global state of the chain, instead, it only requires the block headers and provides proof through the use of Merkle trees. This solution eliminates the need for a central authority providing decentralization. Contrary to SPVs, *Centralized two-way pegs* trust a central authority to verify those same transactions and the facilitation of the funds exchanged between chains. These are typically used by centralized Exchanges such as Binance[2], which facilitate the buying and selling of cryptocurrencies for fiat currencies or other cryptocurrencies, such as Bitcoin. While this solution benefits from efficiency it also introduces a single point of failure and centralization. Exchanges can also be seen as *Notary schemes*, which will be addressed later.

Federated or multi-signature two-way pegs are an attempt to reduce the centralization that comes from *centralized two-way pegs* by having a group responsible for the process of locking and unlocking the tokens, instead of a single central authority. This still benefits from similar efficiency as *centralized two-way pegs* but still has the downside of centralization, even if reduced.

As mentioned, RSK is one of the solutions in this area. In order to obtain the token associated with RSK, RBTC, a user would need to lock up the same amount of BTC in Bitcoin's chain. It makes use of *federated two-way pegs*, where the federation is composed of multiple respected members of the RSK community. RSK also uses a consensus protocol known as DECOR+ and a technique called merge mining, which aims to solve a conflict known in Bitcoin as forks, where miners are mining different blocks at the same chain height. DECOR+ solves these conflicts in a way that maximizes revenue for all miners, regardless of whether they were involved or not in the conflict. In merge-mining, rewards are shared between miners which are mining the correct block, to reduce competition between miners.

Notary schemes opposed to sidechains are not attached to a certain blockchain. A notary or group of notaries monitor several blockchains and act as an intermediary for users who wish to transact between chains. Notaries monitor events that occur on a chain in order to trigger transactions on another chain, so users rely on notaries for exchanging assets between chains. While notary schemes can't be seen as fully decentralized solutions, as notaries are typically centralized entities, different notary schemes have different degrees of decentralization, depending on the number of entities acting as a notary.

Notary schemes are technically a simple solution to implement but the security of this solution is dependent on the trustworthiness of the notaries involved[27]. Centralized cryptocurrency exchanges are one of the most common applications of notary schemes, like Binance or Coinbase[3]. In these ex-

[2] https://www.binance.com/en
[3] https://www.coinbase.com/

changes, users can purchase cryptocurrencies with fiat currencies or other cryptocurrencies. Buyers and sellers create transactions associated with the token being sold or bought and the exchange provides a matching service between buying and selling transactions, known as arbitrage services. Users of a centralized exchange have a wallet in the exchange which is owned by the exchange itself, this is a custodial wallet, which will hold the funds bought by a certain user. While decentralized exchanges exist, such as Uniswap[4], centralized exchanges still hold most of the market share, as they provide a higher degree of comfort and liquidity which decentralized exchanges still can't provide. Decentralized exchanges can be implemented using *hashed timelocks*, but currently, decentralized exchanges lack the adoption necessary to have liquidity levels similar to centralized exchanges. Adoption and development are in part driven by past events that occurred in centralized exchanges, resulting in the loss and robber of an enormous amount of funds from exchanges[9].

Hashed timelocks or Hashed Timelock Contracts (HTLC) use hashlocks and timelocks to require cryptographic proof from the user before an established deadline. The required proof isn't necessarily tied to 1 specific blockchain which enables cross-chain atomic swaps[12, 21]. Hashlocks can also trigger different actions, it can for example enable multiple payment outputs from a single hashlock. Hashed timelocks were initially designed for use in decentralized exchanges but it has also enabled other applications, such as Lightning Network[25], which is a second layer application that provides faster throughput and improved scalability to Bitcoin. In lightning network transactions happen off-chain and are later bundled into one transaction that is broadcast to Bitcoin's chain. This is possible by using hashed timelocks to create micropayment channels between nodes that take part in the network, funds are tied up in these channels until the correspondent transactions are broadcast.

Cryptocurrency-directed approaches like Sidechains mainly focus on improving the scalability of the main network by offloading and batching transactions, reducing the resources required on the mainchain. If oriented for permissioned blockchains, this approach can help to improve the scalability problems associated with private blockchains, and also isolate the mainchain from attacks to the sidechain. Sidechains provide limited interoperability and are dependent on certain configurations of the blockchains involved such as the consensus mechanism used, this limits the development of more complex applications. There is also a significant trade-off between security, performance, and decentralization, as which increased decentralization, we get better security but lower performance.

These solutions can also have undesired problems for instance centralization in the case of notary schemes increasing the security risks associated with blockchains, as acquiring the private keys of the notaries endangers the funds they are responsible for. In the case of sidechains that require a different

[4] https://uniswap.org/

consensus mechanism, the network is constrained by the slowest consensus between chains, thus stalling assets and lowering liquidity.

Notary schemes while easier to implement and providing higher levels of interoperability are typically associated with centralized applications such as centralized exchanges. Hashed timelocks of these 3 solutions are the first to provide full asset exchange between chains while keeping a higher level of decentralization. HTLCs allows for the creation of trustless payment channels and do not require a direct connection between trading nodes, as there can be chained channels acting as the middleman between source and destination nodes. In HTLCs there is the risk of fund retention and unfair trade as the nodes that act as a channel can choose to not provide proof of payment tying the funds for a certain amount of time. Cross-chain transactions using HTLCs can also incur high cost as it might require multiple transactions, with each having fees associated.

For enterprise systems, cryptocurrency-directed approaches are not the best solutions as these mainly focus on interoperability between homogeneous systems, and on providing interoperability for financial systems, with associated cryptocurrencies.

2.3.2 Blockchain Engines

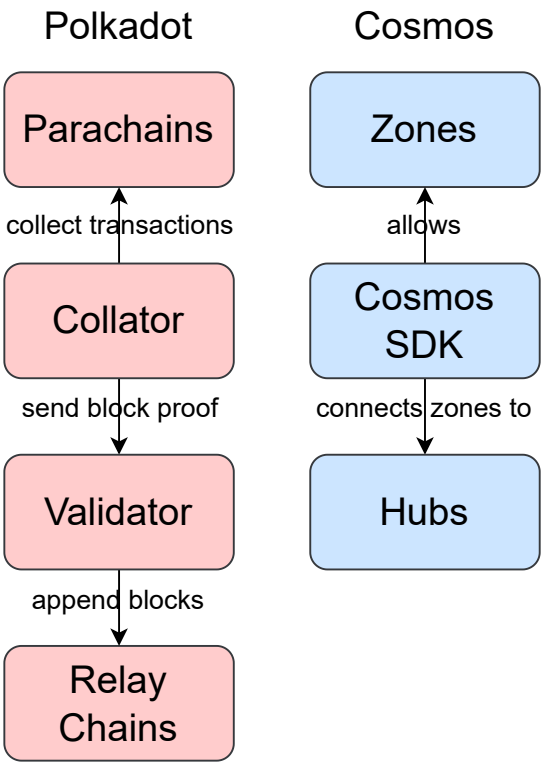

Figure 2.4: Comparison between Cosmos and Polkadot architecture

Blockchain engines focus on providing interoperability to heterogeneous systems contrary to the approach explored before which focuses on homogeneous systems, in this case, cryptocurrency-related use cases[4]. Blockchain engines such as Cosmos and Polkadot have different approaches but both aim to provide a framework for the creation of customizable blockchains and deployment of decentralized applications, which can interoperate with each other.

Cosmos[16] uses a token called Atom, and the network works with 2 main components, *hubs*, and *zones*. *Zones* are independent blockchains using Tendermint as the consensus algorithm, which was already explored in this document, and the *hub* is the first zone of the Cosmos network, called Cosmos Hub, which provides a connection between different zones[26]. All communication between *zones* and *hubs* is made using an inter-blockchain communication protocol similar to a virtual User Datagram Protocol (UDP)/Transport Control Protocol (TCP) adapted to blockchains. Cosmos Hub keeps a record of the tokens and token type in each *zone*, with token transfers between connected zones being required to go through the hub. In Cosmos participants can act as Validators or Delegators, validators use their stake in the network to process blocks, validators also act on behalf of delegators in this process, both earning transaction fees.

Polkadot[34] uses a token called Dot and is built on top of Substrate. The network has 3 core components, the *Parachains*, *Relay chains*, and *Bridges*. *Parachains* are independent blockchains, *Relay chains* manage transaction consensus and delivery, while *Bridges* connect *parachains* and *relay chains*[26]. In the Polkadot network, participants can have 4 different roles these being Validators, Nominators, Collators, and Fishermen. Validators are responsible for ratifying blocks for a certain nominated parachain and subsequently ratifying relay chain blocks, validators are rewarded by acting correctly and as they are bonded to the network, acting maliciously puts their bond at risk. Nominators hold a stake in the network and contribute to the security bond of a validator, thus depositing their trust in a certain validator or group of validators, they are rewarded or penalized based on the actions of the respective validator. Collators are ”full-nodes” of a certain parachain and are in charge of proposing new blocks to validators responsible for that specific parachain. Fishermen oversee validators and report illegal activity to the network, being rewarded by the bond reduction of the malicious validator or validators. Polkadot offers a hybrid consensus that decouples the finality gadget and block production mechanism, using GRANDPA as the finality gadget algorithm and BABE as the block production algorithm[34]. It also uses a cross-chain message-passing protocol for communication between different parachains and relay chains.

Blockchain engines focus on interoperability between instances of the same platform, in the case of Cosmos, between Tendermint based platforms, and in the case of Polkadot between Substrate-based platforms. These solutions act as a router, processing outputs from a source blockchain into inputs of another blockchain, as different chains have different configurations such as consensus mechanisms, communication protocols, and so on. Both Cosmos and Polkadot use similar mechanisms to pegged sidechains or hashed timelock contracts.

Although blockchain engines offer a simple way to achieve a certain level of interoperability, blockchain engines do not interoperate with each other, requiring users to choose between existing solutions. In the case of Cosmos, there is some flexibility in the way a zone is configured, for Polkadot this is more restricted. Blockchain engines also rely on transaction fees as an incentive to keep the network operating, which in the case of enterprise solutions these are 2 undesirable problems that out weight the accessibility of these solutions. While further research could be made to find a solution that allows interoperability between blockchain engines this would require splitting the efforts of interoperability research and development, which is not the goal.

2.3.3 Blockchain Connectors

Blockchain connectors can be divided into 4 different sub-categories, *Trusted Relays*, *Blockchain Agnostic protocols*, *Blockchain of Blockchains*, and *Blockchain Migrators*[4]. Trusted relays are an approach that requires a trusted party to relay transactions between source and target blockchain, similar to how Cosmos Hub, which was discussed previously. Blockchain agnostic protocols offer interoperability regardless of the blockchain-technology involved, but typically do not provide backward compatibility, meaning that existing blockchains might require changes. Blockchains of blockchains aim to offer a way to create cross-chain dApps that operate on multiple blockchains. Blockchain migrators allow the migration of data across blockchains, usually requiring a centralized party.

Trusted Relays redirect transactions from a source blockchain to a target blockchain, requiring trusted parties to do so. Hyperledger Cactus[22] is a project in this domain that works by allowing a trusted party or parties to publish transactions on several ledgers, using a network of validators to monitor and validate cross-chain transactions, and trusted escrow parties to realize transactions. Validators, in this case, are participants in both source and target blockchains that facilitate cross-chain transactions, for these transactions to be deemed valid they must be signed by a quorum of validators. For this solution to have a higher degree of decentralization, the trusted parties could be replaced with decentralized parties.

Besides centralization, there are also several concerns regarding trusted relays as these require *a priori* knowledge of interoperating network's identities and configurations, and these solutions don't fully solve the concern of a relay service acting maliciously, apart from replication which doesn't fully solve the risks associated. Some of these concerns could be solved by implementing a decentralized blockchain registry service and replacing the trusted parties by a decentralized party possibly achieving a trustless relay. In enterprise systems, where typically we are using centralized applications, the concerns with this type of solutions regarding decentralization and the requirement of *a priori* knowledge of the participants of the network wouldn't apply as we are typically working with permissioned blockchains where participants are already previously identified by the network.

Blockchain Agnostic protocols enable cross-blockchain communication between multiple different blockchains, regardless of their configurations. The Interledger Protocol (ILP)[31] was initially designed as a decentralized peer-to-peer payment network, working similarly to hash locking schemes discussed before, using escrow accounts to facilitate cryptocurrency transactions between chains. The current implementation of the Interledger is a blockchain agnostic protocol, inspired by the Internet, the ILPv4 protocol, which uses packets to send payment information between a network of connectors to ease cross-chain transactions[5]. Participants in the network can act as a sender, those who originate the payment, as a receiver, those who receive the payment, and as a connector, trusted by the sender and receiver to act as an intermediary for their payment. Accounts in ILP are shared between 2 participants, with the respective balance record between those 2 participants, the underlying ledgers correspondent to the blockchains which use ILP are only used for funding and rebalancing of the ILP accounts. Accounts in ILP are maintained by the owners of the account.

The ILPv4 has 3 packet types, these being Prepare, Fulfill, and Reject similar to request, response, and error messages. Connectors forward a Prepare packet from the sender to the receiver, and then they also relay Fulfill or Reject packets from the receiver to the sender depending on the outcome of the payment. A sender initiates a payment by sending a Prepare packet to the receiver. When the receiver obtains the Prepare packet, it sends a Fulfill packet to the sender, if the transaction was successful, or a Reject packet otherwise. All these packets are relayed by the connectors between sender and receiver.

There are currently several implementations of the Interledger Protocol, Hyperledger Quilt[6] is an implementation of the protocol using Java. Quilt implements several Interledger primitives such as payment pointers, interledger addresses, ILP-over-HTTP, simple payment setup protocol, STREAM protocol, and ILPv4. Quilt also offers interoperability between other Interledger implementations.

While blockchain agnostic solutions are a great approach in regards to offering interoperability to existing and to-exist blockchains, they still require adaptation from the blockchain's side, not granting

[5] https://interledger.org/rfcs/0027-interledger-protocol-4/
[6] https://wiki.hyperledger.org/display/quilt/Hyperledger+Quilt

backward compatibility. As the current Interledger Protocol is inspired by web communication protocols, efforts could be made to provide backward compatibility to existing blockchains, positioning blockchain agnostic protocols as a great solution for interoperability. Interledger mainly focuses on providing exchange of value between blockchains, this is a limitation that has very specific use cases, mostly in the area of financial systems.

Blockchain of Blockchains are solutions that aim to provide a platform for the development of CC-dApp, which operate on multiple blockchains. While providing accountability for those issuing transactions of various blockchains, as well as providing a comprehensive, and updated view of the underlying blockchains[4].

Hyperservice[19] is an open-source project which provides a programming framework to develop CC-dApp across heterogenous blockchains, thus working with permissioned and permissionless blockchains. Hyperservice consists of 4 core components, the dApp Clients which allow interaction between dApps and the Hyperservice platform, and the *Verifiable Execution Systems (VESes)* which compile the dApps given by the dApp client into blockchain executable transactions. Clients and VESes run the underlying Universal Inter-Blockchain Protocol (UIP) to record actions as proof in order to enforce accountability. The Network Status Blockchain, a blockchain of blockchains, that keeps a record of blocks from target blockchains that contain transactions related to a cross-chain application. The Insurance Smart Contracts act on those records to verify correctness or violation of dApp executions in a trust-free manner.

The programming framework comprises the Unified State Model (USM) and the Hyperservice Programming Language (HSL). The USM is a blockchain-neutral model that abstracts the underlying compatible blockchains, and the HSL is the hyperservice domain-specific language in which cross-chain dApps are written.

Block Collider[14] is another open-source project to ease cross-chain application development and enables cross-chain communication by bridging the latest blocks of participant chains into one block, called the *base tuple*. Thus, block collider acts as a decentralized unifying chain between bridged chains.

Block collider uses a different approach similar to Proof of Work as a consensus algorithm, called Proof of Distance, which uses string edit distances between obtained hashes, a miner is required to find a hash that is within a minimum edit distance between the hashes of member chains, its transactions, and its blocks.

Blockchains of blockchains are a novel approach to facilitating the development of cross-chain applications, in the case of Hyperservice advances are made to abstract cross-chain dApps from the consensus layers of blockchains in which those dApps partake. Currently, these solutions do not fully solve potential forks on the underlying blockchains.

Blockchain Migrators currently allow for the migration of the state of a blockchain to another, this is still limited to moving data across blockchains, but smart contracts are predicted for the future.

Frauenthaler et al. [10] is a solution that provides a framework for blockchain interoperability, with 3 core components, the *Monitoring Component* which surveys compatible blockchains for calculation of metric values. The *Blockchain Selection Algorithm* uses the calculated metric values to select the most beneficial blockchain and the *Switchover Component* which provides the possibility of switching blockchains in runtime. These metrics are weighted and both metrics and weights are defined by the end-user, with a specific objective in mind, that being either cost or performance. This solution is open-source and it is run by the end-user as a centralized application.

Scheid et al. [29] proposes a policy-based blockchain agnostic framework that relies on an Application Programming Interface (API) based on OpenAPI and a Policy-based Management model to provide agnostic and transparent cross-chain communication. Policies can be defined to optimize the cost associated with storing data to another blockchain or to optimize performance. The blockchain used for storing data will be chosen based on these policies, in the case of optimizing costs, the blockchain with the cheapest cost for writing data is chosen. In the case we are giving priority to performance, a max cost can also be defined as a limit.

Blockchain migrators offer a good solution for data backup for applications that require redundant systems, such as enterprise applications. A blockchain view would help when it comes to migrations between blockchains[2]. A problem with these solutions is that, as mentioned, while they provide data migration across a small number of blockchains, they do not allow the reproduction of the events that lead to the current state of the blockchain through the use of smart contracts as that would require a smart contract translator[4].

Overall the blockchain connector category has one critical limitation as most solutions in this category do not support forks, and lack a solution for potential forks. This is a more severe problem for public blockchains as in private blockchains forks are less common. This can affect the dependability of cross-chain dApps, as with forks it can lead a cross-chain application to be in an inconsistent state. In the case of blockchain-agnostic protocols, building an API that works similarly to how web protocols like Hypertext Transfer Protocol (HTTP) work, could lead to a good interoperability solution while offering backward compatibility.

Approach		Leading Projects	Strong Points	Weak Points	Summary
Cryptocurrency-directed	Sidechains	RSK[18]	Simple solution, relying on verifying block headers	Limited functionality	These solutions are focused in providing performance improvements to homogeneous systems with interoperability coming as a consequence
	Notary Schemes	Binance, Coinbase	Simple way to facilitate cross-chain transactions	Limited applications, centralization	
	Hashed Timelocks	Lightning Network[25]	Increases Bitcoin performance	Susceptible to timelock expiration exploits	
Blockchain Engines		Polkadot[34], Cosmos[16]	Simple way to provide interoperability between instances of the same platform (different systems within the same organization)	Restrictive and limited interoperability	Solutions focused on enterprise systems but do not provide interoperability between different blockchain engines or other blockchain infrastructures
Blockchain Connectors	Trusted Relays	Hyperledger Cactus[22]	Effective in facilitating cross-chain transactions (including smart contracts)	Centralization, a priori knowledge of participants required	Most of the solutions lack do not address potential forks, which can lead to applications being stuck in an inconsistent state, affecting the dependability of these solutions
	Blockchain Agnostic	Interledger[7], Hyperledger Quilt[8]	Provide decentralized payment channels implemented in common programming languages	Lack of backward compatibility, designed for exchange of value	
	Blockchain of Blockchains	Hyperservice[19], Block Collider[14]	Environment for development of cross chain dApps	Lack solutions for potential forks	
	Blockchain Migrator	Scheid et al., Frauenthaler et al.	Good solution for data backup for redundant systems	Do not allow migration of smart contracts, and reproduction of events that led to the current state of the blockchain	

Table 2.2: Comparison between existing blockchain interoperability solutions

2.4 Hyperledger Cactus

In this section we are going to define Hyperledger Cactus core components which provide a way to achieve interoperability between multiple blockchain infrastructures.

2.4.1 Cactus Core Components

The core components of Cactus include the connectors, which allow establishing a connection between the blockchain infrastructures, which in this case is Hyperledger Fabric and Ethereum. The validators are effectively a node in both of the involved blockchains with permissions to publish transactions in both networks. The business logic plugin defines the logic of the interoperability use case between both infrastructures, such as the transaction flow and asset changes. We also have the verifier which is responsible for verifying and accepting results from validators.

Cactus offers a core framework for achieving interoperability between multiple Distributed Ledger Technology (DLT), which include blockchains and other more traditional DLT. To this core framework, by using the mentioned components cactus offers a pluggable way to achieve interoperability between heterogeneous system architectures. The plugins in cactus are effectively an abstraction layer on top of Cactus core source code.

This offers an enormous amount of flexibility, while new DLTs will require more development, that will only require additive development to what Cactus already offers to support interoperability use cases across those DLTs. This is important for backward compatibility purposes.[22]

2.4.1.A Validators

For each specific ledger Cactus consortium associates a group of validators, which effectively act as a secondary network that actively monitors the state of the underlying ledger network. Validators run a consensus algorithm separate from the connected ledgers to agree on the state of the underlying

network. Upon agreement on the state, the proof of state is produced and signed by several validator nodes according to the consensus in use.

Validator nodes depend on ledger-specific plugins, consequently, a smart contract on the connected blockchain can enable the ledger-specific functionalities necessary for a validator node to observe the ledger state to finalize a proof of state. When providing the results, validators associate their digital signature, by using their key, with the results for verifiers to be able to certify the produced results.

2.4.1.B Verifier

Verifiers are responsible for verifying the results and produced proof of state provided by validators. Verifier nodes can request and register the public keys of the validator nodes of a blockchain network that they want to connect to. Therefore, they can verify the signed proofs of the state of the blockchain since they have the public keys of the validator nodes. This implies that the verifier nodes trust the validator nodes and consequently they trust the Cactus consortium operating the validator nodes.

2.4.1.C Business Logic Plugin

The business logic plugin executes business logic and provides integration services that are connected with multiple blockchains. It is composed of web applications or smart contracts on a blockchain. It is a single plugin and is required for executing Hyperledger Cactus applications.

It should be developed with a specific use case in mind, by implementing the business logic associated with such use case to interact with ledger plugins respective to each involved ledger.

It offers multiple interactions with the ledger plugins, such as submitting a transaction request at the targeted ledger, querying the targeted ledger, or receiving event messages associated with those transactions requests and queries.

Cactus offers several business logic plugin samples that implement multiple different use cases which can be used as guidance or even adapted for new use cases, these include implementations with Graphical User Interface (GUI) and non-GUI implementations which is what we have chosen to use.

2.4.1.D Connectors

Cactus connectors are ledger-specific plugins that are composed of validators and verifiers, to communicate the previously mentioned business logic plugin with each involved ledger. By using validators and verifiers, connectors provide a way for the business logic plugin to operate and monitor the ledger behind them.

There are several connectors already developed for more common blockchain technologies such as Hyperledger Besu, Corda, Fabric, Iroha, Quorum, Ethereum, and several others are being developed. The more existing connectors available more blockchains can be included in interoperability use cases using Cactus.

2.5 Bills of Exchange

Bills of Exchange are a paper-written contract that involves 3 parties, the drawer which is the party that is in debt, the payee to whom the drawer is in debt, and the drawee which accepts the payment of the drawer's debt. Once the bill of exchange is approved by all parties, the drawee is legally bound to pay the drawer's debt to the payee on behalf of the drawer within a set deadline, so the debt between the drawer and the payee is extinguished[24]. This paper-based model doesn't work in the current situation of a global market where suppliers and customers aren't next-door neighbors, rendering Bills of Exchange unpractical to use. As creating a digital model for such a process has certain requirements, such as ensuring that such a document has legal validity and a non-tampering warranty, distributed ledger technologies such as blockchain technology is seen as a good fit.

Traditional distributed ledger technologies can also be used to implement a bills of exchange solution, which has its problems. Taking into account that bills of exchange data is sensitive it should not be handled by a single entity, which in solutions implemented using such technologies decentralization is not the focus.

As blockchain has seen improvements in recent years, different solutions regarding a digital bills of exchange model have been in development. These solutions, such as Billex [9] and DigiBoE [24] resort to blockchain technology and more specifically smart contracts to implement the whole life cycle of bills of exchange. The core operations regarding the bill of exchange life cycle include the issuing of the bill by the drawer, the acceptance of such bill by the drawee, the payee receiving the bill, and redeeming it receiving the payment from the drawee. The current life cycle of a bill of exchange varies between countries, that is one of the reasons that blockchain interoperability is a requirement for such a system.

By using smart contracts it facilitates the creation of a digital platform that handles the life cycle of the bills, as the platform is able to keep a record of issuing the smart contract and its current owner, the issue can also fill the details of the bill such as the payee's public key, the amount to be paid, its maturity date and then sign it with his private key providing the required non-tampering warranty. For certain platforms such as Billex redeeming the bill can even be automated by the account associated with the smart contract if its balance allows.

[9] https://billex.club/bill-of-exchange/

3

Bills of Exchange' Blockchain Solution

Contents

To explore existing blockchain interoperability technologies, we propose a blockchain implementation for Bill's of Exchange to replace the current paper model in use in Portugal, which allows us to explore a specific blockchain interoperability use case between different Bill's of Exchange implementation providing interoperability among heterogeneous blockchains. The solution for this use case provides a database system capable of issuing, signing, registering, and executing all transactions and operations related to Bill's of Exchanges securely, using a permissioned blockchain.

Hyperledger Fabric provides the underlying permissioned blockchain functionality, which provides some of the features that we associate with blockchain technology, such as *data integrity* and *non-repudiation*(where an entity cannot refuse its participation in transactions related to a bill). To manage the required operations supported by Bill's of Exchange we are using the Chaincode provided by Fabric, this includes registering an asset that represents the bills on the blockchain with a standard data structure. As a blockchain is append-only and bills of exchange operations require updating certain attributes of the bill, each update requires a new state with the result of the updated asset, this also means that we keep the history of changes made to a certain bill on-chain. This system will be used jointly with the Portal de Letras de Créditos Digitais (PLCD) platform which handles the user interface for the end-user from which the Bill's of Exchange blockchain ingests data and execute the transactions associated with each operation. Users have their authentication in the PLCD platform linked to the authentication credentials from Fabric's on-chain cryptographic identity management service.

The Bill's of Exchange blockchain solution is crucial to evaluate the central questions regarding blockchain interoperability in this thesis: how can we enable interoperability between permissioned and permissionless blockchains and what is the main problem with providing blockchain interoperability? In this chapter, we are only addressing the blockchain solution which allows us to explore the interoperability use case. Starting by introducing the requirements for the Bill's of Exchange system and then presenting the implementation and design choices for Bill's of Exchange blockchain.

3.1 Requirements

The solution was developed and implemented as a database system capable of issuing, signing, registering, and executing all transactions and operations contemplated by the law for parties who rely on bills of exchange to obtain and provide funds. The solution should be able to interact with solutions from other countries, regardless of the infrastructure and technologies used in different implementations of this system. INCM has several non-functional requirements for this system:

1. *Availability:* The solution developed should be available 99.9% of the time.
2. *Scalability:* The solution should be able to support both a high and low number of users or nodes in the network.

3. *Security:* The solution should provide a robust system for safely issuing, signing, and execute transactions or operations throughout the bills of exchange life cycle.

4. *Testability:* The solution should provide the possibility of being tested in a non-production environment. (i.e., an environment where operations such as issuing, signing, and execution of transactions can be simulated using similar characteristics to the production system).

5. *Privacy:* Information regarding bills of exchange should only be accessible to those who have the proper privilege rights to access that information.

6. *Auditability and Traceability:* The solution should provide ways for authorities (such as the Tax Authority, the Central Bank, and the Justice Sector) to have access to logs of the system in order to trace, prevent or act upon malicious or illegal activity.

7. *Interoperability:* The solution requires interaction with other bills of exchange systems to be able to support the current operations.

As for functional requirements, the solution is supposed to be integrated within the domain of INCM's projects and should facilitate all of the features currently allowed by the original model in use. This includes several operations regarding bills of exchange, such as:

- Issuance of bills of exchange
- Signature of bills of exchange
- Registration of bills of exchange
- Execution of all transactions and operations regarding bills of exchange
- Offer interoperability between other blockchain implementations handling bills of exchange

The system ingests data generated by each transaction executed by the users in the PLCD platform, which is the platform that gives access to the user to all operations contemplated by the applicable legislation in the life cycle of bills of exchange, among others, issuance, acceptance, endorsement, discount, payment, and more. It should also take into consideration other environmental data required for the execution of the business logic and reports.

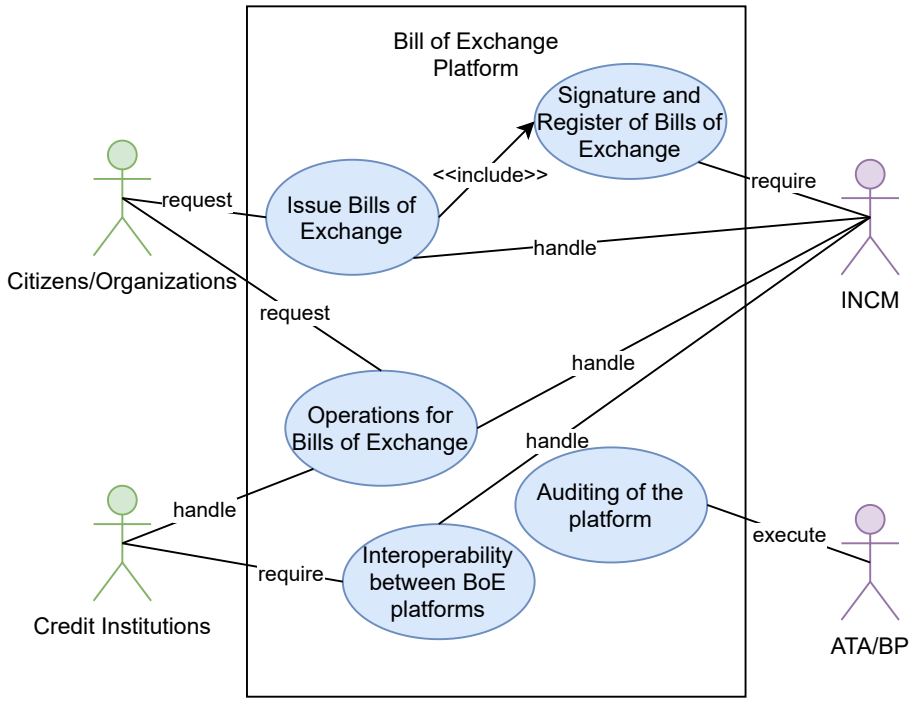

Figure 3.1: Bills of Exchange Use case Diagram

3.2 Preliminaries on Bills of Exchange Blockchain

In this section, introduce the conditions and environment in which the bills of exchange system operates by exploring the existing solution that is currently in use in Portugal, this allows us to leverage this knowledge to be able to develop and implement a solution that works in these same conditions and still meets the mentioned requirements.

The use case discussed in this thesis presents several characteristics:

1. Participants are willing to cooperate but have limited trust in each other
2. Bill's of Exchange assets are a responsibility of all stakeholders (this includes INCM, Autoridade Tributária e Aduaneira (ATA), Banco de Portugal (BP), and the participant banks)
3. Multiple organizations should be able to have an administrative role in the network (such as INCM, ATA and BP)
4. End users only have permission to access bill's related to them (where they are one of the entities represented in the bill)

Regarding the first two characteristics, consensus mechanisms used in blockchains already provide the needed decentralization to ensure that no single entity controls the blockchain, for permissioned blockchains such as Fabric, this remains true. For the third point, Fabric also allows for decentralized

management of the network, allowing for multiple administrators. While 3 different organizations were mentioned, even if INCM was the only administrator, we could represent it using several nodes, one for each team interacting with the ecosystem. Lastly, for the fourth point, Fabric allows for the delegation of different levels of access to specific participants using a traditional Attribute-Based Access Control (ABAC) system.

Besides these characteristics, there are problems associated with blockchains, such as the possibility of not all of the participating entities conscientiously collaborating during the consensus protocol, which can lead the network to an erroneous state. While this risk is much higher in public blockchains, with the possibility of leading to a 51% attack, these are reduced by using a permissioned, private blockchain such as Fabric, where participating entities are known. As mentioned previously, Fabric also allows for modular consensus protocol, essentially leading to the decoupling of the consensus algorithm and the security model, as Fabric utilizes an endorsement policy to validate transactions, the stricter this policy, the more resilient the network is.

A Blockchain participant represents an entity that participates in the blockchain. As mentioned, we assume that participants have limited trust in each other and participate in the same channel. Participants control peer nodes that maintain the ledger and may endorse transactions. We also assume that at least one participant is capable of running a blockchain client to commit transactions to the blockchain-based on the information present in the PLCD platform and that machines running nodes are physically separated, meaning that an organization has no access to another organization's node. In case of attack, it is assumed that one or more nodes, which compose the minority, are colluding to make alterations to a bill. It is given that, communications within the blockchain ecosystem are done using secure channels such as SSL/TLS.

There are 3 actors who take part in the network:

- *Network Administrator*. Network Administrators are responsible for managing all of the blockchain configurations. They also create identities and manage the participants in the network. The organization responsible for the network is in most cases the Administrator (in this case INCM), if there are several Administrators, a quorum will make the decisions instead.

- *Forwarder*. Forwarders are oracles responsible for interpreting operations related to bills of exchange executed in the PLCD platform and committing the respective transactions associated with those operations to the blockchain. These act as a blockchain client and oracle.

- *Auditor*. Auditors audit the life cycle of bills of exchange, such as the operations executed throughout its lifespan. In this use case, ATA and BP, would be the auditors.

The authentication of all network participants relies on public-key cryptography (PKI) [28], and public-private key pairs, by leveraging Fabric's Certificate Authority (CA). We assume that the private key is

only known by their owner, and that participant's public key is known by everyone in the network.

3.3 Bills of Exchange Data Model

In this section, we explain the data model that addresses the business concerns about issuing and handling bills of exchange. While handling bills of exchange, different operations require a different set of permissions, in this section, we define what permissions are associated with each participant in the network. We also define the asset and its attributes, and how each operation makes alterations to that asset.

As this use case is being used mainly to explore blockchain interoperability solutions, the main goal is to keep the implementation of the operations and assets representing bills of exchange as simple as possible, while still being able to achieve the requirements. The asset representing a bill of exchange is stored as a JSON Object which is composed of several attributes:

- *ID*. ID is a unique identifier composed by the keyword *"boe"* followed by an identifier created according to the Universally Unique Identifier (UUID) standard [17] that is used to easily identify each bill of exchange. It is represented using a String type attribute.
- *Expiration Date*. Expiration Date represents the date on which the bill becomes invalid for use, this is also used for certain operations and is represented using a Date type attribute.
- *Initial Amount*. Initial Amount represents the initial amount the bill was issued with, this means the value that the entities owe or are owed, and is represented using a Float type attribute.
- *Remainder Amount*. Remainder Amount represents the amount that is still to be fulfilled, as bills can be paid in multiple installments, this value will always be below the Initial Amount. Similar to the Initial Amount attribute it is represented using a Float type attribute.
- *Drawer, Drawee, Payee*. These similarly to the ID attribute are a unique identifier composed by the keywords *"drawer", "drawee", "payee"* followed by an identifier created according to the UUID standard. It is represented using a String type attribute. These should be linked to the PLCD platform and could be changed to use something such as the Chave Móvel Digital which is used in Portugal for digital authentication as a Portuguese citizen, but this is not handled in this thesis.
- *Previous Drawers, Drawees, Payees*. These are a list storing the Strings corresponding to the previous entities associated with the bill.
- *State*. State corresponds to a String type attribute that stores a value associated with the current State of the bill of exchange.

All of the operations that are to be executed in the life cycle of a bill of exchange make the necessary updates to the asset representing that bill, modifying the respective attributes associated with the operation in question. For each operation, there are changes made to certain attributes, this includes in most cases updating the Remainder Amount or State, or change the entities referred to in the bill, such as the Drawer, Drawee or Payee and by consequence of that updating the list for Previous Drawers, Drawees and Payees. The Expiration Date and Initial Amount attributes are mostly used for certain verifications during the bill's life cycle.

We are using a ABAC system to guarantee that only participants with the required permissions can commit certain operations, for each organization or user there are the following permissions:

- *INCM*. As INCM is acting as a forwarder between the PLCD platform and the blockchain network, they have permissions to issue transactions corresponding to every operation contemplated for Bills of Exchange.
- *ATA* and *BP*. ATA and BP are acting similarly to an auditor, so they have permissions to view data related to the full length of bills of exchange in the network.
- *Banks*. Banks are allowed to issue transactions for the Discount operation, as they are a required entity for this operation.

The entities listed on a bill of exchange, or the end-user, are not contemplated in this list as they don't directly interact with the blockchain, they interact with the PLCD platform and the connection between the platform and the blockchain is then handled by INCM.

3.4 Technologies

In this section, we address why we are using blockchain to implement the Bill's of Exchange digital solution, instead of other alternative existing technologies such as traditional or distributed databases.

Traditional databases, like MySQL, rely on roles to define a set of users who can insert or update data within the database. Furthermore, Administrative roles can modify the contents of the database independently of their decentralization. So, due to its nature, it is expected that compared to a permissioned, private blockchain such as Fabric, traceability, verifiability, transparency, security, and privacy are usually more fragile.

As distributed databases, we have Amazon's DynamoDB or Azure's CosmoDB as an example, which are cheaper and allow support for multiple non-trusting writers. Blockchains still allow achieving what distributed databases cant offer, such as the distribution of trust, with the transparency and auditability between involved parties that blockchain can offer. Distributed databases also rely on other technologies to achieve the proper configuration of nodes between the network for specific functionalities, which

blockchain can achieve on its own. For our use case, as mentioned, we have multiple restraints regarding the organizations:

1. Participants have limited trust between each other.
2. Trust and responsibility to maintain the proper flow of transactions in the network belongs to every organization.
3. INCM should be able to administer the network.

As bills of exchange involve multiple organizations, participating entities are not willing to delegate full control of the system, as the information carried in a bill can affect multiple entities, avoiding its manipulation is essential. For that reason, each entity should be able to monitor the network and ensure the proper flow of transactions. The requirements mentioned match the features offered by a blockchain. Consensus mechanisms ensure that no single entity holds control over the blockchain, and a blockchain such as Hyperledger Fabric allows the delegation of specific levels of control to each different participant. This shows that a blockchain solution is suitable to address the bills of exchange problem.

We should also consider the trade-offs between a permissionless and a permissioned blockchain. As there is information that should be accessible by certain organizations or certain entities, a Permissionless blockchain solution raises some privacy concerns due to information leakage. While it is possible to restrict access to information, there will always be the possibility of information leakage. As we are storing information regarding users' bills of exchange, this problem is also scaled when we take into account the more recent General Data Protection Regulation (GDPR) laws. This system's main goal is to provide a secure digital solution for the life cycle of bills of exchange for both the user and the organizations responsible for maintaining that same life cycle. Information regarding bills should only concern the entities involved in their life cycle and their administration. When mentioning public permissionless blockchain solutions, we also have to mention that these were designed with cryptocurrency in mind and that these blockchains are maintained by miners who collect transaction fees as an incentive to maintain the network.

By using a permissionless blockchain INCM would incur those same transaction fees, which is not intended for our use case. This shows that public blockchains are not a suitable solution to address the proposed problem.

With the requirements from Section 3.1 in mind, a permissioned blockchain is a promising infrastructure for our use case. Permissioned blockchains offer not only reduced risk of participants intentionally introducing malicious code through smart contracts, or acting maliciously against the system, but also offer more control over the network. This comes from permissioned blockchains requiring participants to be authenticated being attributed an identity within the network before being able to participate. In Hyperledger Fabric, transactions also follow several rules which help ensure the security of the network,

such as following the endorsement policies defined by the consortium for the network. It also keeps track of all transactions committed, valid or invalid. This helps auditors identify offenders or possible offenders and use preventive measures in future occurrences.

Taking all this into consideration we opt to use Hyperledger Fabric as our blockchain infrastructure, which as mentioned before is a permissioned blockchain focused on enterprise solutions.

3.5 Bills of Exchange Blockchain Architecture

In this section, we introduce the architecture for the bills of exchange blockchain solution. The assets being stored in the blockchain are the bills of exchange generated by the PLCD platform, which is the information system that allows the end-user to issue operations on their bills. This proposed solution provides scalability in terms of the participating organizations. The information system mentioned and the blockchain for the bills of exchange are independent, this is possible due to the modularity offered by Hyperledger Fabric. This thesis doesn't address the information system and assumes it is already in place, instead we focus on the blockchain infrastructure which will be used to explore the interoperability use case. The blockchain component includes the ledger, permission management, transaction validation and definition, and the data model. This architecture is represented in Figure 3.2, our thesis focuses mainly on the blockchain side of it, and addresses the different types of clients required.

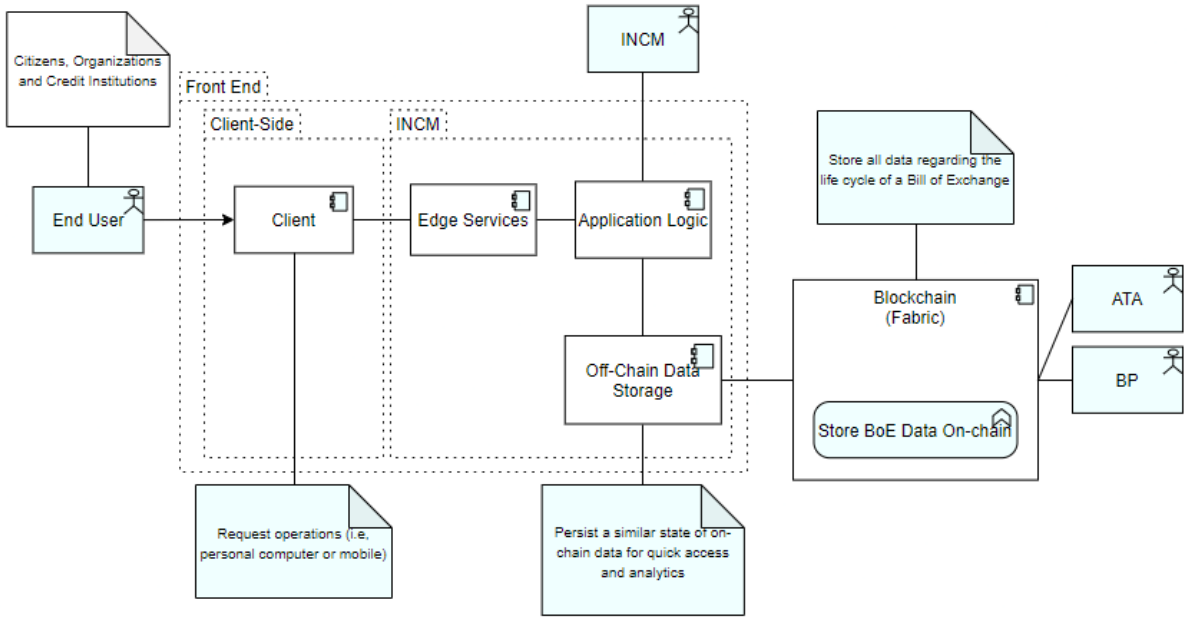

Figure 3.2: Architecture of the Blockchain Platform for Bills of Exchange

The blockchain component store all the data associated with the assets respective to each bill of exchange, while also enforcing the defined blockchain configurations regarding each different organization

and participant in the network. The blockchain client ensures that participants can issue operations on the network concerning bills of exchange. In the following Sections, we approach both the Blockchain Component and Clients with more detail, Section 3.5.1 and Section 3.5.2.

3.5.1 Blockchain Components

Implementing Fabric's blockchain infrastructure requires different types of nodes or how they are called in Fabric, peers: Orderer peers (orderers), Endorser peers (endorsers), Committing peers (peers). Orderers are assigned in the configuration file which can be deployed by the network administrators, and provide delivery guarantees of blocks containing transactions while assuring atomic communication, consensus. A node can be both an endorser and a peer node at the same time, endorsers host and execute instances of the chaincode to run simulations of transactions, peers host instances of the ledger. Apart from these 3 we also have Anchor peers which are responsible for inter-organization communication, within the blockchain, to ensure peers from different organizations are known to each other [1]. For simplification, anchor peers are not present in our solution, as they are not essential for our use case.

It is important to note, that in Fabric, there is the possibility to attribute different levels of trust to each peer using roles. Each different role has different permissions on the operations that can be applied by participants on the ledger, this includes having limited operations on bills of exchange depending on your role. And as mentioned before, Fabric offers a modular system that decouples the trust system from the consensus algorithm, this is done using endorsement policies that can assign a higher or lower degree of trust to specific subsets of endorsing peers in the network.

For our specific use case, and to keep it simple as our main goal is to explore the interoperability solutions, we have a single instance of chaincode which defines all of the life cycle of the bills of exchange, this includes all of the operations contemplated in the law regarding bills. Both endorsing peers and committing peers are required to have this chaincode installed.

Regarding the authentication of participants, Hyperledger Fabric offers a built-in CA to issue identities to each participant, but a custom CA could be used if INCM chooses to do so. These certificates are important as they are used by participants to sign transactions, ensuring non-repudiation of transactions.

In our use case, there is limited trust between participants, and INCM is acting as the network administrator, so a single orderer is being used. For decentralization purposes, the network could be deployed with orderers from multiple organizations.

In this scenario, the organizations interested in auditing the network, such as BP and ATA, should maintain at least a peer node holding an instance of the ledger, also called a committing peer, allowing for these organizations to assure bills are not being tampered and retrieve data associated with bills of exchange for auditing purposes. The organization responsible for the network, INCM, should have an endorser peer, as endorsers are responsible for assuring the validity of transactions, so in this case,

assuring transactions associated with bills of exchange are valid. Banks which participate in the network should help to maintain the network by running a committing peer as they are a crucial part of the bills of exchange life cycle, and are responsible for the discount operation. Regarding the privacy associated with accessing data concerning bills of exchange, we have to define permissions for each auditor if that is the case, which can be done by the network administrator.

With respect to data privacy between different organizations, Fabric offers multiple solutions which all have their advantages and disadvantages. Banks for example shouldn't be able to access the same data INCM, BP, or ATA can access. We have 3 possible solutions:

1. Creation of multiple Channels, one for each organization
2. Using private data, a Fabric's feature
3. Using the previously mentioned ABAC system

We could create 2 separate channels, one for INCM, BP, and ATA, and one where banks would be included. By creating different channels we are also effectively creating different ledgers as ledgers are shared within channels, which would ensure data privacy, but would also prevent all participants to see all the transactions in the network. Fabric offers a private feature where certain organizations are given the ability to endorse, commit, and query private data without having different ledgers. This solution adds a significant amount of overhead to transactions. Lastly, by tuning the ABAC[13] system through chaincode we can define which organizations have permissions to access data. For our solution, we have chosen the last option as it offers the best performance and ease of implementation while being able to address the restrictions in our use case.

In the end, as the goal was a simple and easy to implement solution, the architecture we end up with has a single channel where we have the 3 main organizations, being INCM, BP, and ATA, in the future Banks should be represented as they are crucial for the bills of exchange life cycle. We also have a single applicational chaincode that implements the logic behind all of the operations regarding bills of exchange. Lastly, to address the privacy concerns we end up tuning an ABAC system to meet our needs as mentioned before. There are already solutions to address the problems associated with a traditional ABAC system regarding the centralization concerns and limitations, such as the Self-Sovereign Identity Based Access Control (SSIBAC)[3].

3.5.2 Blockchain Clients

While our solution does not focus on a full-stack application, as the goal is to have a simple prototype to help explore the interoperability solutions, it is important to address the goal of the blockchain clients, as they are important to establish the connection between INCM's front-end application and the blockchain.

Blockchain clients allow organizations to request operations to the blockchain, for different needs we should have different clients as each different client is able to request different operations depending on their permissions and roles.

In our use case, we require 3 different blockchain clients, as we have INCM acting as a network administrator and forwarder between the PLCD platform and the blockchain, we have ATA and BP acting as auditors requiring access to data respective to bills of exchange, and lastly, we have banks which directly interact with a specific operation in the bills of exchange life cycle. These 3 blockchain clients are defined as follow:

1. Forwarder Client: After receiving transaction requests from the PLCD platform, those are processed and verified before a blockchain transaction being committed to the network, triggering the updates necessary on the respective asset.
2. Audit Client: Responsible for committing transactions on behalf of the auditors, ATA and BP, these transactions are in fact requests from the auditors to query the blockchain ledger regarding data from bills of exchange assets in the blockchain network.
3. Bank Client: This will give banks the capability to commit transactions in respect to the Discount operation in which banks are directly involved, to make the necessary changes to the bills of exchange asset.

In addition to acting as a bridge between the PLCD frontend application and the blockchain, blockchain clients also provide authentication in the blockchain network. Members of the participant organizations can link their credentials inside the organization with the client to have their blockchain identity and organization identity linked, this provides a way to keep track of who requests certain operations, allowing for traceability.

3.6 Bills of Exchange Blockchain Implementation

For the implementation of the blockchain architecture mentioned, we used a modified version of Fabric's samples to meet the needs for our specific use case. This implementation includes establishing the blockchain infrastructure and configuration regarding the participant organizations, defining the ABAC system rules and attributes for access control, and the chaincode for transactions and queries associated with the blockchain.

3.6.1 Blockchain Components

As mentioned Fabric uses a certificate authority to generate the necessary cryptographic information, such as the key pairs, to enroll different participants in the network. The CA server can be configured

to define what implementations are used for the generation of the certificates, we have opted to use Fabric's default implementation.

We have implemented the solution with 3 default organizations which represent INCM, ATA and BP. Each node representing a different organization has different specific permissions regarding which data can be accessed and which operations can be requested, using the ABAC system. Nodes from INCM, the network administrator, have a higher level of permissions allowing them to validate and execute smart contracts and transactions, ATA and BP nodes can validate transactions and access the ledger as an auditor. These configurations and permissions allow enforcing the necessary privacy and confidentiality of data.

The chaincode responsible for all of the operations in a bill of exchange life cycle, and its auditing were implemented using NodeJS and following the data model previously mentioned in Section 3.3.

3.6.2 Blockchain Client

The blockchain client is written in NodeJS, it allows communication with the blockchain, providing an entry point to the developed chaincode mentioned before. Taking into account the permissions defined for the authenticated client, the chaincode will then access the distributed ledger and execute the requested operations: such as register a bill of exchange, protest a bill of exchange, or audit the network.

While possible to develop a graphical user interface, the clients made were used through the command line. A script was used to simulate the population of the blockchain with bills of exchange assets.

4

Blockchain Interoperability Solution for the Bills of Exchange Use Case

Contents

In this section, the goal is to explore interoperability solutions between heterogeneous blockchains such as a permissioned blockchain like Fabric which we used for our bill of exchange solution as it was outlined in the previous Chapter 3, and a permissionless blockchain which in this case we have chosen Ethereum.

4.1 Interoperability Use Case Implementation

In this section, we address the interoperability use cases explored and their respective implementation, as well as mention the advantages and disadvantages of cactus during our implementation. As far as the architecture used, as mentioned, we are using our Bills of Exchange solution outlined in Chapter 3 and an Ethereum Testnet Network sample provided by Cactus as shown in 4.1. Apart from these networks required to explore the interoperability use case, we are using a validator to monitor both networks and commit transactions associated with the different interoperability use cases being explored which are implemented using a business logic plugin, for this the connectors respective to each network are required.

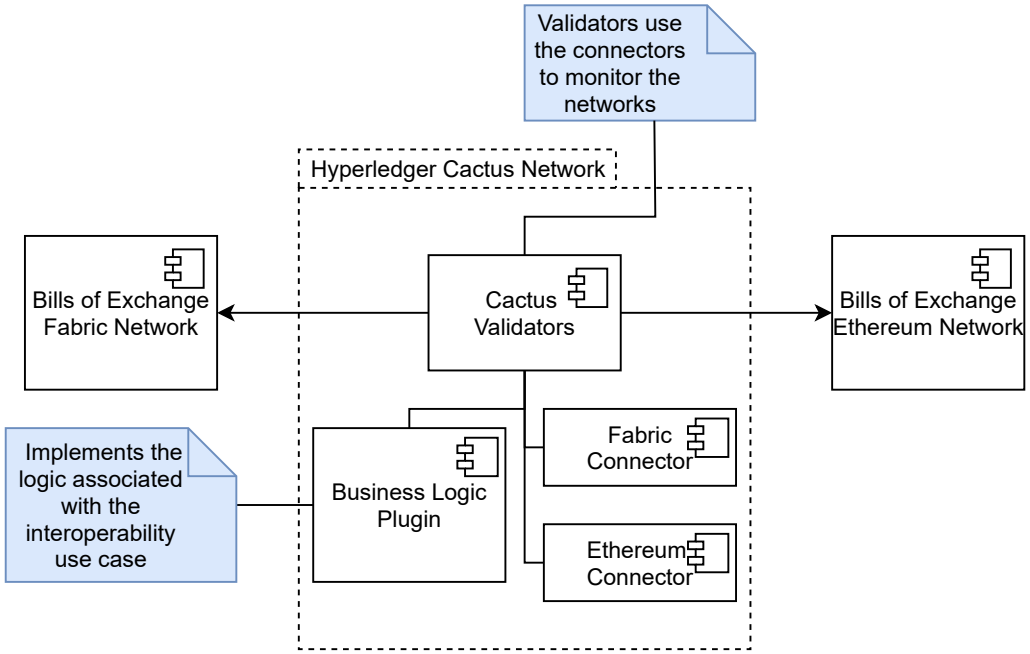

Figure 4.1: Architecture of the Cactus interoperability solution

While it would be of interest to explore interoperability use cases with already existing Bills of Exchange solutions that were mentioned previously such as Billex, it is hard to have access to these solutions in a test environment that suits our needs. The Ethereum Testnet solution offers a lower degree of complexity to explore several interoperability use cases, as we can mold the network to suit our

needs. To note that Cactus already offers connectors for both Fabric and Ethereum which helps to focus only on the interoperability use cases being explored without needing to develop a whole new connector, which would be the case if we were using Billex's network to explore interoperability, as it would require a connector for Stellar.

When it comes to bills of exchange, there are several interoperability use cases that can be explored with Hyperledger Cactus. These include replicating the assets between multiple implementations of bills of exchange, replicating transactions based on a specific trigger such as having a bill of exchange being registered in the Portuguese network for bills of exchange that involves an entity from Spain, you can replicate the transactions associated with this specific asset on the Spanish network for bills of exchange. We could also use a foreign currency or even a cryptocurrency to liquidate a bill of exchange, which would require interoperability between our solution and that cryptocurrency blockchain.

While there is no standard for how to implement a bill of exchange blockchain solution, every bill of exchange implementation, regardless of the country it is associated with, requires a core set of operations such as registering or paying the bill. Taking this into account we define the required transaction flow associated with each interoperability use case to achieve the needs for that specific use case, which we will go more in-depth throughout this Chapter.

Most of the work being done is associated with designing business logic plugins that define the transaction flow for each interoperability use case. We will outline that work for each interoperability use case explored, while also mentioning the difficulties and where it could have been improved.

4.1.1 Network Replication

The goal with this specific use case is to replicate every transaction requested in one of the involved networks, we have our solution developed on Hyperledger Fabric and a sample solution using Ethereum, whenever a transaction is requested on one of the networks it should trigger Cactus to request the corresponding transaction on the other network as presented in Fig 4.2.

For this to happen, on the business logic plugin we need to associate transactions from the solution which uses Fabric and the solution using Ethereum, for the business logic plugin there is complete abstraction from how these blockchains work, it is not required for them to behave similarly. While both solutions might have different operations for Bills of Exchange, both should have the core operations required for their life cycle, in this case, if a register operation is requested in Fabric's solution, the respective register operation should be triggered by cactus on Ethereum's solution.

As mentioned cactus has nodes monitoring the blockchain's state, using this we can react to updates to this state, such as new transactions being committed, which in this case helps us replicate the same state of each asset on the blockchain to other solutions, working similarly to an oracle. In figure 4.3 we can see exactly how this behavior happens between the different blockchains involved, their users, and

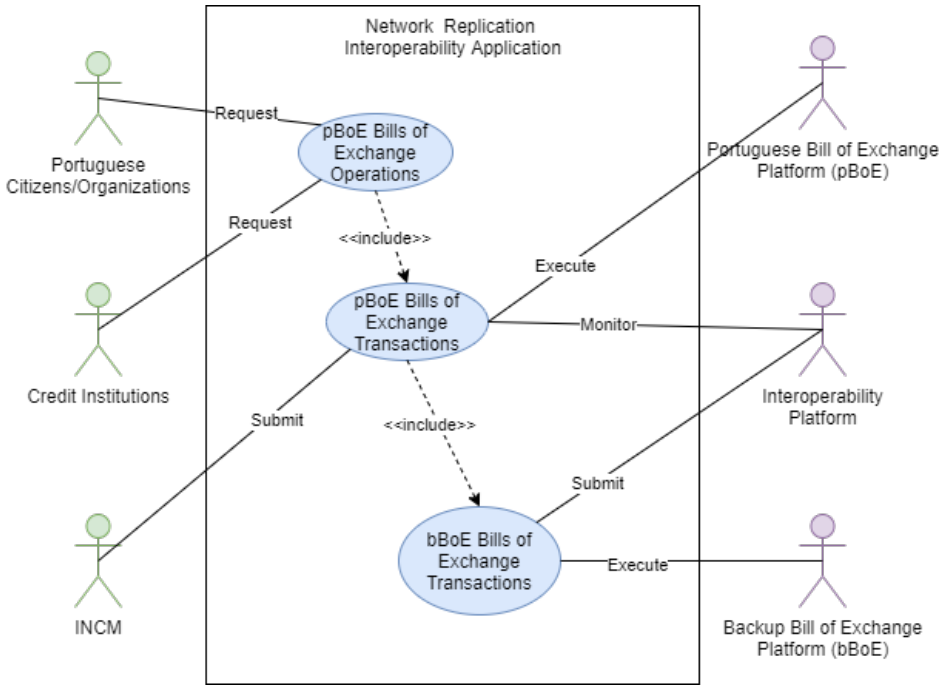

Figure 4.2: Cactus Use Case 1 Diagram

the cactus auxiliary network.

A user commits a transaction to Fabric's network, in this case, registers a bill of exchange, the transaction gets processed in Fabric's network which triggers a new state of the network. Cactus validators network by monitoring the network are aware of this new state and publish the respective transaction to the Ethereum network which then similarly to Fabric ends up creating a new state of the network. We have to be aware of this because new states are what triggers Cactus validators, so we need to have a condition where transactions committed by Validators do not trigger a transaction request on Cactus end.

For this specific use case, there are simpler solutions if the goal is only the replication of the network for the purpose of redundancy which we went over before. One of the problems of solutions that focus on replication only is that it is hard to achieve the transaction history that led to that specific replicated state of the network, Cactus offers a way to accomplish that by actually replicating each transaction requested individually.

4.1.2 Cross-country Asset Replication

As pictured in Fig 4.4 this use case works somewhat similarly to the previous use case explored in the sense that we are effectively reproducing certain transactions between the involved networks. Similarly, we have our solution previously outlined in Chapter 3 using Fabric, and an Ethereum sample solution,

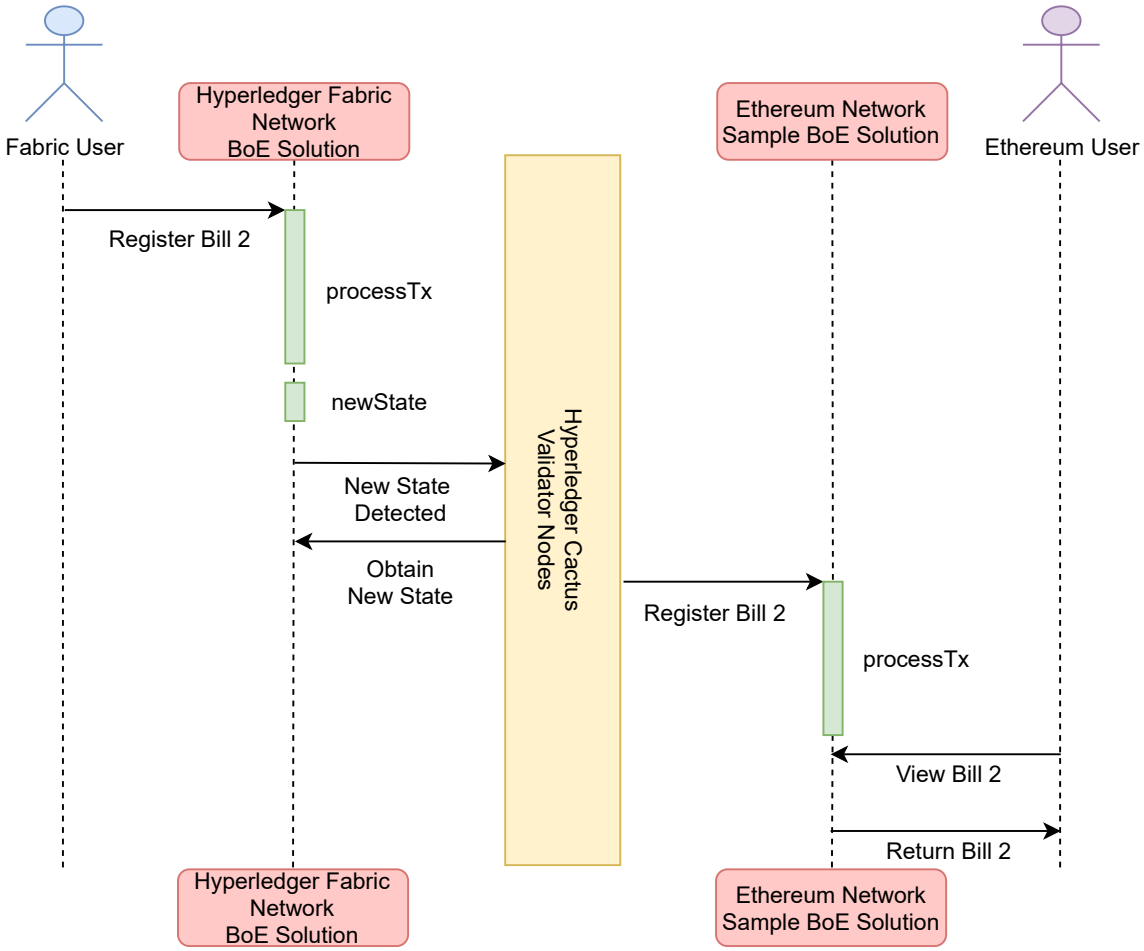

Figure 4.3: Cactus Business Logic Plugin transaction flow for Use case 1

both running implementations of the bills of exchange use case.

The goal in this use case is to replicate transactions for assets that involve entities from different countries, lets say our Fabric solution and the Ethereum sample solution are running implementations for Portugal's and Spain's respective bills of exchange systems, if a certain bill involves entities from Portugal and Spain, we want to reproduce transactions associated with this specific asset in both networks.

Comparably to the previous solution, we still have to associate corresponding transactions between both networks. The difference in this use case is that instead of replicating every transaction that produces a new state in the ledger, we are being more restrictive by only reproducing transactions for specific assets. In addition to cactus validators nodes to react to changes in the blockchain state, this state update will trigger a verification process to assess if the assets being changed involve entities from each network, in this case, if there are entities from Portugal and Spain associated with this asset. If this is the case then the validators will request a transaction in the opposite network to replicate the state of this asset. In figure 4.5 we can see the flow between the actors in this use case, in both cases, where

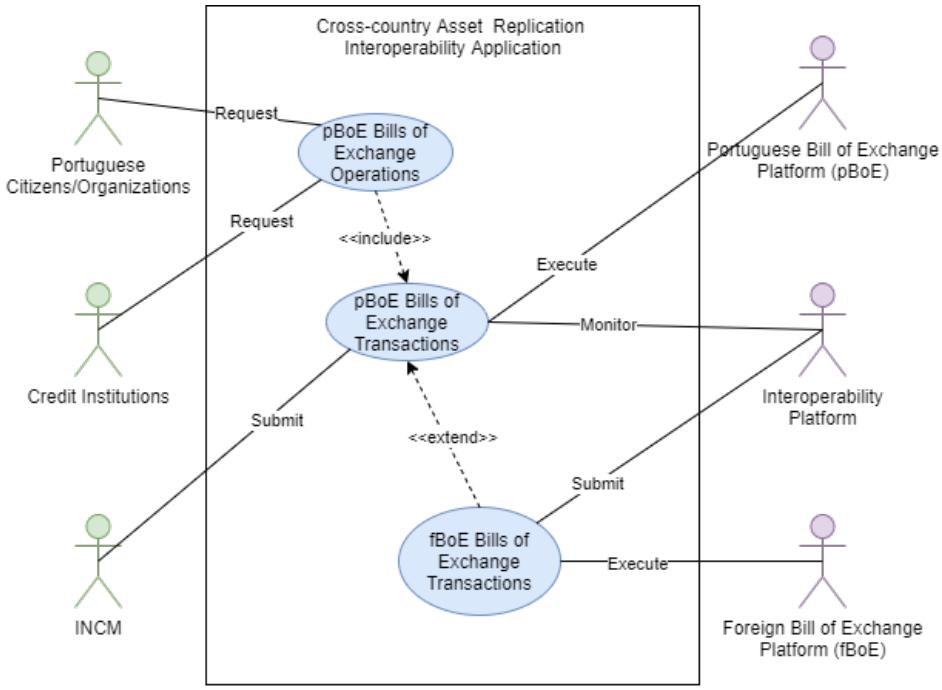

Figure 4.4: Cactus Use Case 2 Diagram

an asset involves entities from both countries, and when it does not.

In the case where the transaction does not involve entities from both networks' respective countries, the validator still detects a state update, since the validator network is always monitoring the networks, but after verifying the entities associated with the asset it will not trigger any transaction request on the opposite network.

When the transaction does update an asset that has entities from multiple countries associated with, after detecting an updated state and verifying the entities involved in the transaction the validators will then request the respective transaction in the opposite network, effectively replicating the same life cycle for this specific asset in both networks.

Similarly to the previous use case, cactus offers a way to replicate the exact transactions that led to a specific asset current state, keeping its transaction history, in this case, we are not going for full replication of the network but focusing on single assets that respect certain constraints, which in this case Cactus offers a very good solution as the verification for those constraints occurs in the validator nodes without any need for changes in the blockchains interoperating.

4.1.3 Cross-blockchain Payment

With countries starting to accept cryptocurrencies as legal tender, such as El Salvador, and allowing cryptocurrencies to be seen as a country's currency, this use case becomes more interesting to ex-

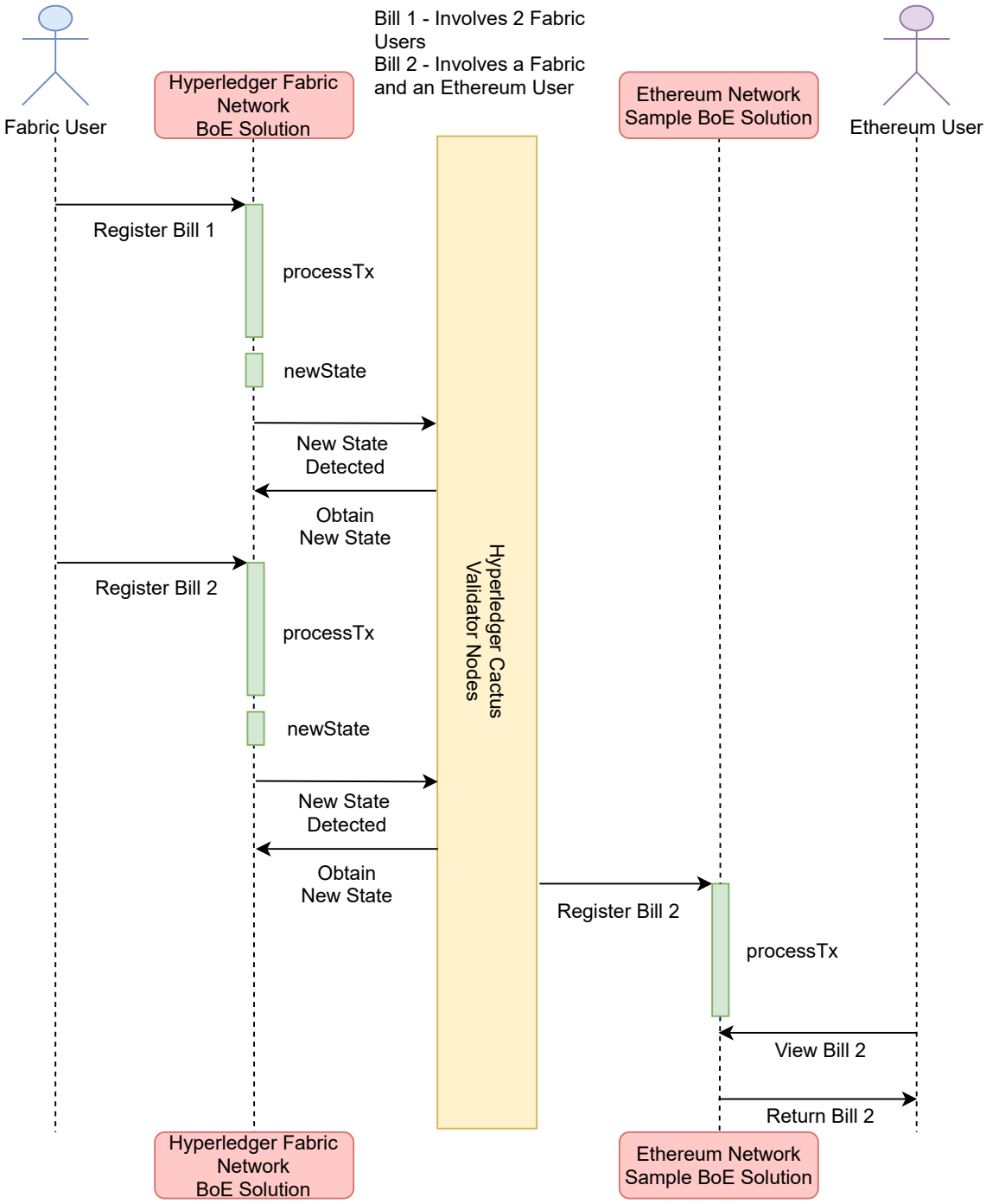

Figure 4.5: Cactus Business Logic Plugin transaction flow for Use case 2

plore as this could be a potential need in the future. The goal of this use case is to use an existing cryptocurrency to liquidate a bill of exchange asset.

As far as the architecture used, similarly to the previous use cases we are still using our Fabric

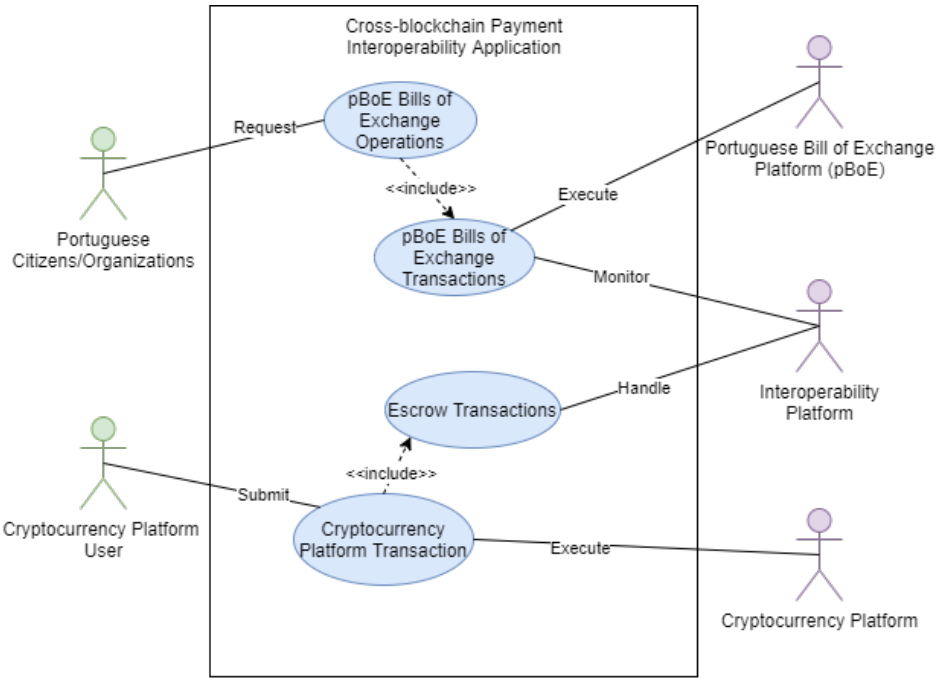

Figure 4.6: Cactus Use Case 3 Diagram

solution outlined in Chapter 3, but in this use case the Ethereum network used is not running a sample implementation for bills of exchange, it is simply being used to run Ethereum transactions as a payment option for bills of exchange to show that if in the future more countries start to accept cryptocurrencies as a currency, this is a possibility.

For this use case instead of having to match the transactions representing operations respective to bills of exchange, we have to find a way to associate, in this case, an Ethereum address to a certain bill of exchange asset or an entity related to this specific bill. The simplest way for this to be done is in the case where the bill of exchange solution is developed with this in mind, and the asset itself can have an Ethereum address associated with it for payment purposes.

Another problem with this use case is the need for an escrow account as shown in Fig 4.6, as the payment operation on the Fabric network can only be processed after confirmation of payment, but the payment can only be sent after the payment operation has been processed. In this case, Cactus validators can act as escrow where they receive the payment from the Ethereum network and hold it until the payment operation in Fabric's network has been processed, finally, they send the payment to the final address associated with the bill.

In figure 4.7 we can see the whole transaction flow which differs from the 2 previous use cases where Cactus validators were reacting to changes in the networks, wherein this use case a request to Cactus to act as a middleman is being made. The figure shows both a case where the payment operation on Fabric's network succeeds and a case where it fails, where the validators escrow account is important.

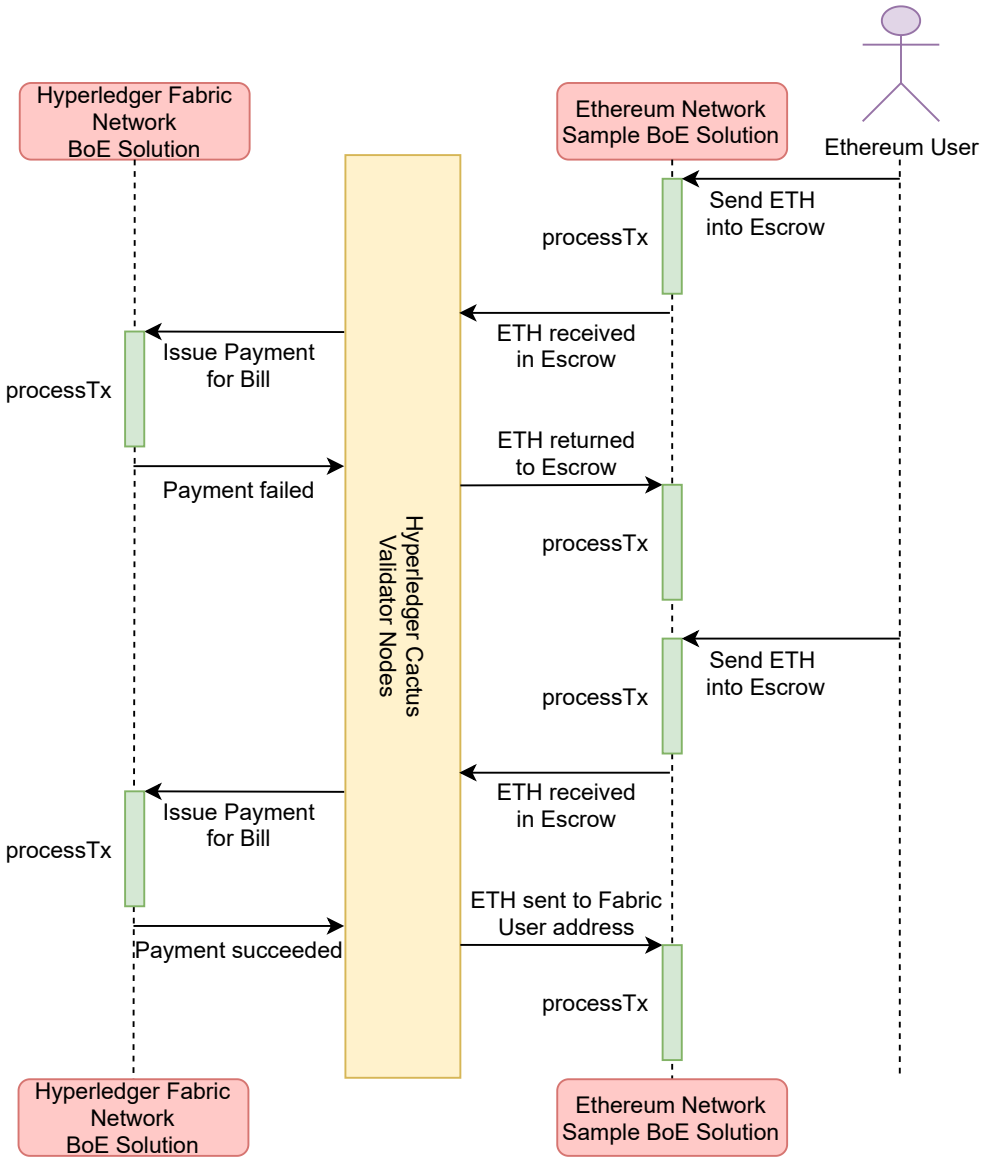

Figure 4.7: Cactus Business Logic Plugin transaction flow for Use case 3

To see the importance of using Cactus validators as an escrow in this use case, we can see the difference in the transaction flow between the first set of transactions and the second, where the first one ends up failing, with the second one succeeding. If for any reason the transaction for the payment of the bill fails in Fabric's network, the Ethereum that had already been provided as payment is returned to the original address it was sent from instead of being sent as payment to the Fabric's user address associated with the bill being liquidated. This keeps the funds safe until there is confirmation that the bill's payment succeeded, it also prevents the bill from being paid without the funds being secured for the owner of the bill.

It is important to explore such a use case as the future seems to be approaching a state where

cryptocurrencies might be seen as an actual currency and start being adopted as such, this also shows that Cactus can be quite flexible in terms of the problems it can solve in a somewhat simple way. It also shows how future-proof Cactus is. While we explored this use case using Ethereum as the cryptocurrency being used, multiple other ones could have been used as long as Cactus already has a connector developed for it, otherwise, a connector could also be developed.

5

Evaluation

Contents

In this chapter we are going to evaluate both the Bills of Exchange Blockchain solution, developed using Hyperledger Fabric, and the Hyperledger Cactus solution for the interoperability use cases explored. As far as evaluating the blockchain solution, we are going to describe the environment used for testing the system, the metrics used, and the methodology. We also define how this evaluation is performed, by defining the metrics and goals, which are then collected, analyzed, and discussed. Regarding the interoperability solution, we are not evaluating its performance as the solution's performance is bottlenecked by the 2 interoperating systems, in this case, our Hyperledger Fabric blockchain solution and an Ethereum network used for the examples provided. We evaluate and discuss the complexity of the solutions, their limitations, and advantages.

5.1 Evaluation Methodology

In this section we define our evaluation methodology and approach regarding our Bills of Exchange solution, explaining how we are going to measure its performance.

The focus of our evaluation is the costs of assuring that the assets representing bills are protected, as that is the main goal of our solution. We are going to evaluate the performance of all operations regarding bills of exchange as all of them directly interact with the asset, producing a new state in the network by requiring an update to the asset. While a new state is produced by every operation, some operations require changes to more attributes, having a heavier impact in terms of performance.

The goal of this evaluation is to answer 3 main questions regarding our solution: i) What is the maximum throughput it can achieve, this includes how many bills of exchange operations can be executed per second?, ii) What is the latency at the maximum throughput achieved, what is the time window between an operation being processed, and its results being secured?, and lastly iii) In terms of storage, and protection, what is the cost associated with our solution, compared to the existing solution, is our solution scalable? By answering such questions we can conclude if our solution suits the needs of our use case or not.

5.1.1 Metrics

Regarding the evaluation of a blockchain solution, there are certain fundamental concepts. For evaluating a blockchain's performance a typical configuration is composed of the *test harness* which is the software and hardware, including clients that can introduce or invoke work from the system, mainly this client can be a load-generating client which will be addressed in the following Section. It is also comprised of the System Under Test (SUT) which is the hardware, software, networks, and respective configuration of the blockchain solution being tested, in our case, the bills of exchange blockchain solution.

This configuration and the key performance metrics are highlighted by the Hyperledger Foundation[1].

Regarding the key performance metrics, the ones outlined are the following:

- *Read Latency*.

  Read Latency = Time When Response Received – Submit Time

  Read latency is the time between when the read request is submitted and when the reply is received.

- *Read Throughput*.

  Read Throughput = Total Read Operations / Total time in seconds

  Read throughput is a measure of how many read operations are completed in a certain time period, expressed Reads per second (RPS). While this is an important measure, Transaction Throughput is typically more important, as there can be external systems facilitating reads to the blockchain network.

- *Transaction Latency*.

  Transaction Latency = (Confirmation time @ network threshold) – submit time

  Transaction Latency is the time between the point where a transaction is submitted to the point it is usable across the network. This is the time it takes for a transaction to not only be processed and confirmed but also the time to spread it across the network reaching a certain threshold of nodes that guarantee the finality of a transaction according to the consensus mechanism being used.

- *Transaction Throughput*.

  Transaction Throughput = Total committed transactions / total time in seconds @ #committed nodes

  Transaction Throughput is the rate at which valid transactions are committed by the blockchain in a certain time period. This includes transactions being spread across the network, and not only being confirmed, as previously explained. To note that the invalid transactions that may occur during testing should be subtracted from the total transaction number to obtain the transaction throughput, expressed as Transactions per second (TPS).

[1] https://www.hyperledger.org/wp-content/uploads/2018/10/HL_Whitepaper_Metrics_PDF_V1.01.pdf

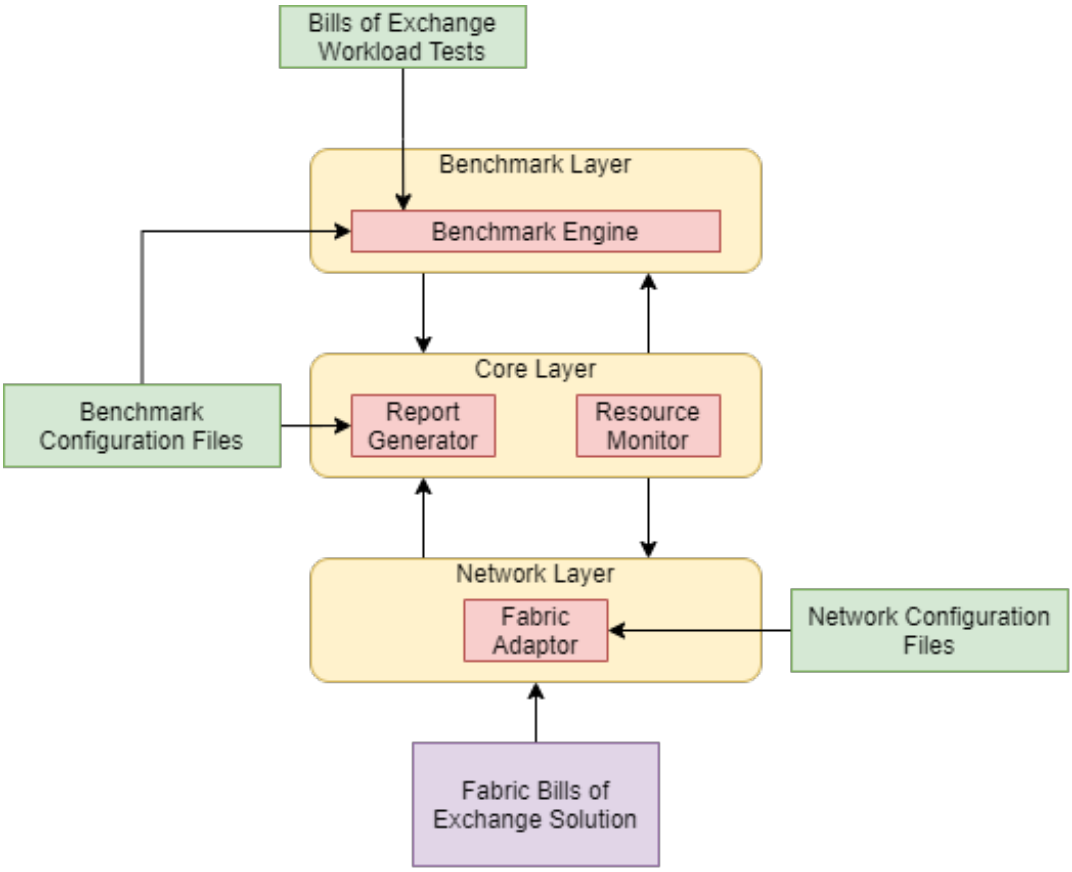

Figure 5.1: Testing Bill of Exchange solution with Hyperledger Caliper

5.1.2 Hyperledger Caliper Load-generating client

As mentioned in the previous section we are going to study performance metrics, additionally, we are also going to measure resource consumption, which includes mainly the storage requirements for storing bills of exchange assets. To obtain data regarding these metrics we need a client to serve as a workload input.

We are going to use Hyperledger Caliper[2] as our load-generating client, Calipers framework goal is to facilitate the evaluation of multiple blockchains solutions built on different infrastructures such as Hyperledger Technologies blockchains, which include Fabric, and others such as Ethereum. Caliper serves as a load-generating client by running tests based on configuration files, which helps to replicate the architecture of the solution's network, to run tests.

As seen in Figure 5.1 Caliper has several layers, the Network layer, Core layer, and Benchmark layer, with each layer having its own purpose. The network layer allows the framework to integrate and connect with different blockchain infrastructure, which in our use case is our bills of exchange Fabric solution. The following layer, the Core layer, provides an abstraction between the adaptor and caliper's

[2] https://www.hyperledger.org/use/caliper

framework by using Northbound Interfaces (NBI) which are common blockchain interfaces that facilitate the interaction with the backend of blockchains, such as querying the ledger or deploy smart contracts. This layer also includes the report generator which produces the reports with the data obtained from the tests, and the resource monitor, which can both be tuned by the configuration files. The Benchmark layer contains the tests that Caliper will use to test the SUT in question, in this case, it includes the tests we developed for our solution.

The context in which the tests are run is specified in a Yet Another Markup Language (YAML) configuration file, used by the benchmark engine to initialize the test environment and run the tests. In this configuration file, you can tune Caliper to suit the needs for each different test, by using *rate controllers* that allow us to control the transaction flow of the system for each test. The 3 most common *rate controller* types, *fixed rate controller*, *fixed feedback controller* and *fixed backlog controller*. A *fixed rate controller* will keep the transaction flow constant depending on the specified flow in TPS, a *fixed feedback controller* works similarly to a *fixed rate controller* but it takes into account the unfinished transactions per client, if a certain threshold is reached it stops submitting transactions for a certain period of time, with these controllers you can emulate the regular use of the system. A *fixed backlog controller* will keep a certain number of transactions in backlog, this means that you specify a number of unfinished transactions per client and the controller will try to keep that value constant, this effectively achieves the maximum TPS value.

5.2 Bills of Exchange Blockchain Solution Evaluation

5.2.1 Setup and Test Environment

To emulate a real production environment where we have several distributed nodes, we are using a Google Cloud Engine (GCE) machine set up in Amsterdam, Netherlands with a 16vCPU and 256GB of memory. This helps to keep the hardware used for the test environment used to run the tests consistent across every round of tests.

As far as configuring the test environment, we are using an Hyperledger Fabric version 2.2.1 running a simplified network with a single channel with 3 organizations representing INCM, ATA and BP each with one peer and 1 CA. The consensus algorithm being used is solo orderer which is designed for testing purposes where it effectively bypasses the consensus process. We are also using the default network configuration provided, with a maximum block size of 128MB, a batch timeout of 250ms and the number of transactions per block is 10.

The peers, orderers and CA are being run on top of Docker containers running Docker version 20.10.8, running the base image of Hyperledger Fabric. The state database used is the database provided by Fabric, LevelDB.

5.2.2 Throughput and Latency

As previously mentioned we are going to use throughput and latency metrics to understand scalability, this means how many transactions per second can our solution handle and how fast are they considered valid. These are important measures to assess the suitability of our solution for the use case INCM provided.

While there is not enough data regarding the use of the system that is currently in place, according to the small amount of data we have the system achieves a peak use of 7000 operations monthly, which is a really small amount of transactions per month for such a system. Taking this into account if we can achieve an average TPS of 5, we should be able to cover much more than the current use of the system that we currently see.

In the following tests, we are going to issue transactions at a constant rate of 5 TPS. We are also going to vary the number of transactions issued in total between 1000, 2000, 4000, and 8000 transactions. Furthermore, we are going to vary the number of blockchain clients between 1, 2, 4, 8, and 16 clients submitting transactions, in our context these clients act as our *forwarders*. We did test all operations but for the data used, as every operation is somewhat similar in terms of performance we are using the average of the data for all operations.

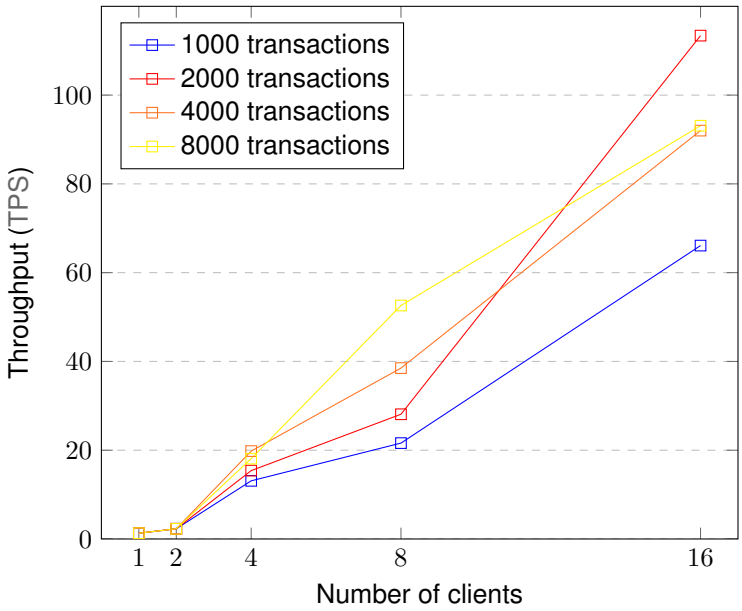

Figure 5.2: Throughput variation with different number of clients, using a fixed-feedback controller

The above graphic (Figure 5.2) shows the variation of the throughput with the number of clients, which in our tests we can see that for a number of clients of 1 or 2 the throughput is equal independent of the number of transactions, but we see more variation with increasing numbers of clients which is a more likely scenario in regular use of the system.

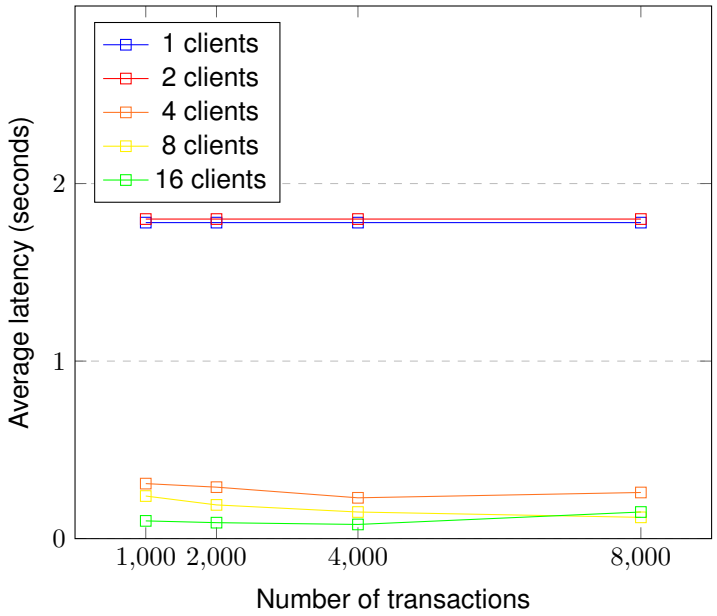

Figure 5.3: Average latency variation with different number of transactions, using a fixed-feedback controller

In the above graphic (Figure 5.3) the variation of the average latency per number of clients is shown, with transactions being submitted at a constant rate of 5 TPS similarly to the previous graph. Usually the more clients the bigger the latency as the peer nodes have to return answers to more clients before a transaction is considered valid, in our testing that is not what we have verified as we have 16 clients with the lowest average latency.

During these tests, something seems to have occurred when we were testing with 1 and 2 clients submitting transactions as the results from those cases seem to be too far apart from what is expected in both throughput and average latency.

5.2.3 Storage

In regards to Storage evaluation, the goal is to estimate how much digital storage such a solution would take, as the current system uses a paper-based solution and there is not enough data to estimate how much storage would be required for the current system in terms of digital storage.

To estimate the storage required the transactions being issued consist on the registration of a defined amount of bills of exchange assets, as the rest of the operations do require updates to the asset but does not increase the storage required for that asset, the only operations which would add to this requirement would be the discount and endorse operation but it would be negligible. Bills of Exchange assets in these tests are still generated randomly which can still create some difference between those same tests.

Apart from data regarding storage, using Caliper we also obtain data regarding CPU usage, disc writes and reads, and network traffic which could be of interest. These tests, similarly to the throughput

and latency tests have 2 variables, being the number of transactions issued (with all of them being register transactions as mentioned) and the number of clients.

Name	CPU% (max)	CPU% (avg)	Memory (max) [MB]	Memory (avg) [MB]	Traffic In [MB]	Traffic Out [MB]	Disc Write [MB]	Disc Read [B]
peer0.org3.example.com	6.87	3.09	151	146	13.0	3.92	22.1	0.00
peer0.org2.example.com	10.06	3.31	234	229	18.0	12.0	22.1	0.00
peer0.org1.example.com	10.02	3.38	213	209	18.1	12.0	22.1	0.00
orderer.example.com	3.43	0.36	205	198	10.4	29.0	24.9	0.00
dev-peer0.org2.example.com	2.31	0.20	60.1	59.9	4.25	2.12	0.00	0.00
dev-peer0.org1.example.com	2.31	0.19	59.4	58.9	4.26	2.12	0.00	0.00

Table 5.1: Bills of Exchange Storage Testing: Issuing 1000 register operation respective transactions with 1 blockchain client

While all this data was collected, for this specific use case the only crucial data is the data regarding memory usage as that is what concerns digital storage costs, from the 2 metrics shown regarding memory usage we are going to focus on the average memory usage instead of the maximum one, and we are going to use the mean value between the 3 peers to produce the graphics and further on an estimate from the data we have what is the storage cost per bill of exchange asset.

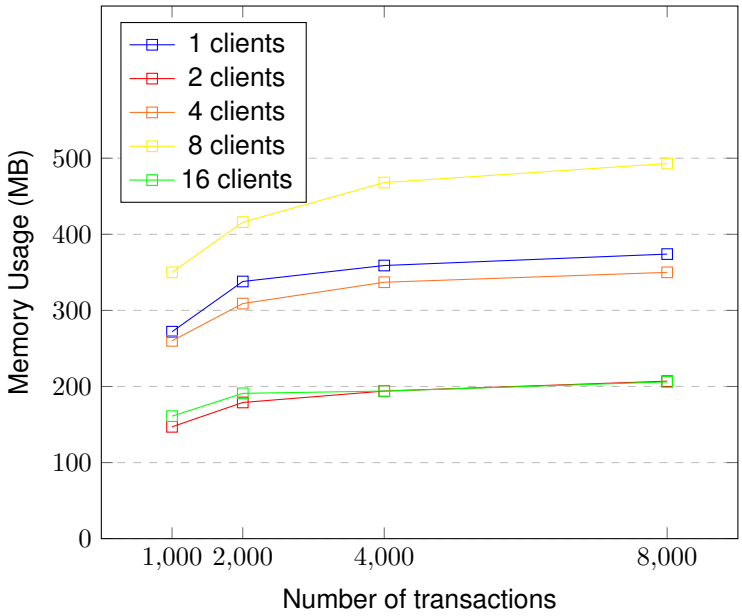

Figure 5.4: Average memory usage variation with different number of transactions

While there is some discrepancy between the storage requirements depending on the number of clients used seen in the graphic above (Figure 5.4), we can see that across all clients the variation of the memory usage according to the number of transactions follow a similar curve.

We are going to explore what would be the best case scenario being the 2 clients curve, the worst case scenario which would be the 8 clients curve, and the average between all of the clients. This should give us enough information regarding what would be the resources usage and cost associated with it on

the most likely scenario which would be somewhere between these 3 scenarios we are going to explore.

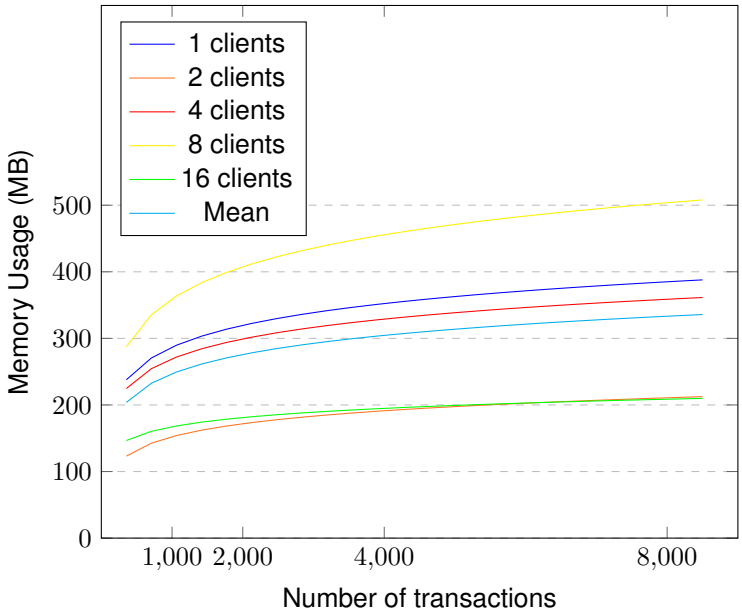

Figure 5.5: Logarithmic regression trendlines for the retrieved data for memory usage

To be able to estimate the cost of storage we need to be able to estimate how much memory it would cost past the values used during our testing, for that we are using logarithmic regressions to define the function that better describes our data, from that we can effectively estimate how much memory would be used at a certain number of transactions, in this at a certain number of registered bills of exchange assets. For the 3 scenarios, we are going to explore we obtained the following equations:

- *Best case scenario - 2 Clients*.

$$S_{2Clients} = -41.8 + 28.1 * ln(x)$$

- *Worst case scenario - 8 Clients*.

$$S_{8Clients} = -120 + 69.4 * ln(x)$$

- *Average of all clients*.

$$S_{Mean} = -39.7 + 41.5 * ln(x)$$

To estimate the cost for these 3 scenarios we need to associate the storage we are using with a defined price, for that we could use some distributed cloud storage solution such as Amazon Web Services (AWS) or Google Cloud Services (GCS), for this specific use case since we were already using GCE for our testing environment we are going to estimate the storage costs using GCS.

The average price for a standard storage bucket in GCS using an European located server is around $0.021 per GB, this price comprises only the storage cost and does not include operations made on top of the stored data such as updates to data points for each operation there is added costs which are negligible, if we estimate the price using the peak operations achieved in the current system monthly and we assume all those operations are a register operation, we would have 7,000 register operations monthly, resulting in 84,000 assets being stored early.

Using the previous equations we can estimate how much storage would be required in our current solution for our 3 scenarios, and consequently what is the cost associated with it:

- *Best case scenario - 2 Clients*. We would have a storage requirement of 276 MB for the 84,000 assets, which would amount to roughly $0.006 monthly per Peer used in the network
- *Worst case scenario - 8 Clients*. We would have a storage requirement of 667 MB for the 84,000 assets, which would amount to roughly $0.014 monthly per Peer used in the network
- *Average between all clients*. We would have a storage requirement of 431 MB for the 84,000 assets, which would amount to roughly $0.009 monthly per Peer used in the network

The storage costs associated with this solution are effectively negligible, which can be due to the fact we are using quite a simple data structure to represent the assets as we are mainly using this solution to explore interoperability use cases associated with such a system. If the solution were to become more complex it would certainly have a bigger impact on the costs associated with it.

5.2.4 Discussion

It is important to note that this specific solution was developed with the main goal of exploring the interoperability use cases associated with such a system, there are multiple decisions made throughout the implementation of the solution with this in mind. The goal was to keep the development of such a solution with as little complexity as possible, which makes some compromises in terms of the overall solution.

Regarding our performance evaluation, we have achieved a peak TPS of 113.4 with a latency of 0.15 seconds in the same test, this peak occurred for a test where 2000 transactions were submitted and 16 clients were being used, while it is likely that fewer clients will be used on a real scenario, during our tests we can see that for any number of clients above 4, we can achieve decent performance without compromising latency. Tests were run multiple times to assure that the results were correct.

One of the places where there are improvements to be made is Fabric's blockchain configuration, this includes several performance improvements in terms of TPS. We could have tested several parameters such as the batch size which includes the size of the block in terms of data and also includes the number

of transactions per block, which if we used a higher value in both parameters we could possibly achieve higher values of TPS. Besides this we could also experiment with different values for the batch timeout which is the time for a block to be deemed invalid if not processed in a certain time frame, a higher value would give more time for blocks to be processed.

Fabric modularity also allows experimenting with another parameter that influences its performance, which is the consensus algorithm used. Depending on the algorithm used, the number of replicas used to form the network directly affects its performance as the messages exchanged during the consensus scale are based on the number of replicas participating in the consensus.

In relation to our storage costs assessment, one consequence of keeping the solution so simple is that it is hard to evaluate the storage costs associated with a real production scenario, as the data structure used to store the bills of exchange assets is quite simple. Due to this, the data we received from our testing results associated with the low amounts of data available to estimate what a real production scenario use would be. During our testing, using the average between all clients scenario as an example we obtained a cost of $0.009 per Peer monthly using GCS assuming we would have 84,000 bills of exchange assets registered in a year.

We did not explore the difference of using multiple cloud storage services as the value we obtained is already negligible for what would be expected. However, there are always trade-offs when it comes to choosing a cloud storage solution, this trade-off will often be between the cost associated with the storage required, the security, and the distribution of trust between, the evaluation of these trade-offs should be done by INCM.

It was expected that the number of clients submitting transactions during the tests would not affect the storage required, as the transactions respective to registering an asset do not influence the amount of information stored. During our tests, there were some discrepancies that can be associated with the fact that the assets are generated randomly which can lead to assets of different sizes. They do however affect the traffic as there are more clients connections, but that was not evaluated.

Throughout the years the storage costs will obviously scale, as each Peer has to store a copy of the blockchain ledger, with more bills of exchange assets being registered each month the storage requirements to store the ledger will keep increasing. It also increases the search complexity associated with locating an asset in the ledger, which for Fabric, similarly to most permissioned blockchains it is O(n).

5.3 Interoperability Solution Evaluation

In this section, we are going to address our evaluation for our interoperability solution implemented using the Hyperledger Cactus technology. There was no performance evaluation performed as the

performance bottleneck for an interoperability solution such as this one is associated with the blockchain infrastructures used in the explored use cases, in this case with our Bills of Exchange Hyperledger Fabric solution, and the Ethereum testnet networks used. This is due to the fact that Cactus uses a network of validators to process interoperability requests on both blockchains, which effectively act as a participant in those networks and are consequently restricted to that network's performance.

We mostly address the complexity associated with implementing an interoperability solution using Cactus, the compatibility Cactus offers when it comes to the multiple blockchain infrastructures it supports for developing interoperability solutions, and the flexibility of use cases where Cactus can be used to implement such solutions.

It is important to understand that Cactus is still in development and is not production-ready at this moment, it should be production-ready in version 1.0, and it is currently in version 0.4.2. This means that there is still development being made in the core framework, and there are still connectors to be released, this will improve the usability of Cactus as an interoperability solution.

5.3.1 Compatibility

Regarding the compatibility that Cactus offers, it is first important to understand how Cactus can implement interoperability solutions between different blockchains, this includes both permissioned and permissionless blockchains. It is also relevant that interoperability between blockchains is an urgent problem with the increasing number of different blockchain infrastructures being developed without an existing standard.

Cactus relies on validators to perform transactions on multiple blockchains to facilitate interoperability between those respective blockchains. These validators in turn require a connection to the respective blockchain in order to submit transactions, this is where Cactus connectors come into play. As each blockchain infrastructure implements a different consensus algorithm, Cactus requires a different connector for each different blockchain infrastructure.

In our specific use cases, we are using connectors that Cactus already has developed, these being connectors for Hyperledger Fabric and Ethereum. Cactus already has other connectors available such as connectors for most Hyperledger Distributed Ledger technologies such as Hyperledger Besu and Hyperledger Sawtooth with existing business logic plugins samples that show these connectors working.

In terms of compatibility the only development necessary to be done in Cactus is effectively the connectors, and with the current number of different blockchain infrastructures available it is hard to develop connectors for every existing one. However, Cactus should be able to support interoperability between most existing blockchain technologies both permissioned and permissionless, as shown in our use cases where we are using both with that being Hyperledger Fabric and Ethereum respectively. Effectively, Cactus offers backward compatibility, by not requiring extra development to the blockchain

technologies being used.

5.3.2 Flexibility

Concerning the flexibility of the solution, what we are trying to evaluate here is if the solution can be used to implement a wide variety of use cases, in our case we explored 3 different scenarios each with its own goal. This was done effectively by developing 3 different Business Logic Plugin (BLP), as explained before the BLP is what allows the logic behind a specific interoperability use case to be implemented. They define the transaction flow associated with the interoperability use case being explored, respective to each blockchain infrastructure involved in the use case defined by the BLP.

As a consequence of requiring a BLP for each different use case, it means there will be development associated with each new interoperability use case being implemented. Cactus unlike other interoperability solutions requires that whoever is implementing a specific use case has knowledge regarding the multiple applications involved.

This is required because when developing the business logic plugin you are effectively defining what triggers a certain interoperability function, what transactions will the validator submit to each associated distributed ledger depending on the function triggered, and in what order should those transactions be processed. In order for this to be possible, you need to be familiar with the applications involved in the interoperability solution, and Cactus.

The advantage that this provides is that by using Cactus as an interoperability solution you are not required to develop your blockchain solution taking into account the possible interoperability use cases it will require in the future. As all of the work regarding the necessary interoperability can be done when using Cactus to implement those use cases by developing their respective BLP.

5.3.3 Implementation Complexity

With respect to complexity, we are not going to evaluate if the implementations steps themselves are complex, we are instead going to evaluate the number of steps required to implement an interoperability solution using Cactus in the different possible scenarios. It is hard to evaluate if the implementation steps taken to achieve a certain solution are or not complex as that is subjective to whoever is implementing them, and how familiar the framework is to them.

There are effectively 2 scenarios when it comes to using Cactus for implementing an interoperability solution, something common in these scenarios is that you will always have to develop a BLP as that is specific to each interoperability use case being implemented. In addition to this, your solution may require the development of a new connector as Cactus does not have connectors available for all existing blockchain infrastructures.

In the worst-case scenario, a connector will have to be developed and tested before actually being available to use in the interoperability solution being implemented. This can be simpler if the respective blockchain infrastructure uses a similar consensus to current blockchain infrastructures supported by Cactus which can be used as an example.

5.3.4 Discussion

Cactus can be used as an interoperability solution for a vast amount of different use cases as it doesn't require adaptation and development of the blockchain infrastructures associated with those use cases. This is a problem with multiple interoperability solutions we address in Chapter 2 as they do not offer backward-compatibility, taking into account that there is already a large number of different blockchain infrastructures, requiring extra development for those for each interoperability use case they want to support is not feasible. For Cactus to support a new blockchain infrastructure it only requires a one-time development of a connector for that respective blockchain, and that development is on Cactus's side.

One of the focuses of this thesis was to explore the interoperability between permissioned and permissionless blockchains, this is a common problem because the implementation of permissioned and permissionless blockchains diverge in multiple points, which in turn hampers the development of interoperability solutions using heterogeneous blockchains. The way Cactus provides interoperability is by using an auxiliary network of Validators that respond to triggers and submit transactions on all of the involved blockchains, them being permissioned or permissionless. So, validators effectively act as participants in those networks, which solves the problem.

As mentioned, the biggest problem with providing blockchain interoperability is that there is an increasing number of new blockchains being developed, and there is no existing standard that these blockchains follow, both at the root of the blockchains that are being developed, as well as the applications which work on top of them.

If we take the 3 scenarios we implemented to show Cactus flexibility as an example, the main concern is that you are required to have knowledge of all of the applications involved in the interoperability use case being implemented, take our replication scenario as an example, you are required to know which operation has the same result in all applications. Taking an example where multiple organizations are going to develop new applications that will only require interoperability between each other, using something like Cactus might not be the best solution, something like Cosmos or Polkadot which work like an ecosystem where every application built on top of it is already able to interoperate between each other might be a better solution.

The biggest advantage of using Cactus is that it presents itself as a adequate interoperability solution for cases where your application requires interoperability with an already existing blockchain application. In general Cactus provides a good solution for interoperability, but for very niche use cases, there might

certainly be better solutions, such as the case mentioned previously where something like Cosmos or Polkadot would work better, the example would be an enterprise ecosystem where you would require interoperability between systems within the same organization.

6

Conclusion

Contents

This dissertation presents a solution that aims to replace the current Bills of Exchange system which resorts to paper support by a digital solution using the Hyperledger Fabric blockchain technology, these solutions require to be able to interoperate with solutions implemented in other countries, that is where we propose an interoperability solution using Hyperledger Cactus framework. The goal is to provide a secure and trustworthy digital solution that still achieves the requirements the current system is capable of achieving. As there is sensitive information being handled, this solution provides distribution of trust while still depending on a centralized information system.

This thesis aims to provide a way to entice enterprises to look into blockchains as a better solution for their projects, by providing a solution for the interoperability requirements that such projects require. This can also show how day-to-day systems could take advantage of blockchain-based solutions to provide a secure way to make these systems digital.

6.1 Contributions

In this thesis, we developed a proof of concept for a Bills of Exchange system using a blockchain solution with the goal of working alongside the current paper-based bill of exchange solution in place in Portugal, for this we used Hyperledger Fabric. We then used this solution to explore multiple interoperability use cases, by implement 3 different business logic plugins for Hyperledger Cactus. During our evaluation of our Bills of Exchange blockchain solution, we managed to achieve a peak of 113 bills of exchange operations per second with an average latency of 0.15 seconds, using 16 clients which ends up being a slightly more costly solution. By using a lower amount of clients such as 4 we achieve an average bills of exchange operations per second of 16 to 20 with an average latency of 0.21 to 0.30 seconds. As far as storage, since our solution uses a very simple implementation of the bills of exchange assets, we can not provide a proper estimate of how this would impact the costs, but in general, we can conclude that the storage costs will increase proportionally to the number of peers in the network. There is a clear trade-off between decentralization and trust which such blockchain-based solutions offer, and the performance and storage requirements that a traditional system provides. As mentioned by tuning Fabrics configurations we could achieve better performance but the performance we achieved, according to the available data, is enough to support the current use the bills of exchange system has.

Regarding the interoperability solution, we implemented 3 different scenarios based on our Bills of Exchange blockchain solution to explore what Hyperledger Cactus can offer. This allowed us to explore how flexible such a solution can be by implementing 3 use cases with different goals and different implementation requirements, and we also explored how Cactus is able to provide compatibility with already existing and to-exist blockchain solutions as that is a crucial focus point with interoperability solutions. We concluded that Cactus can be used for a wide variety of use cases, and while for very

niche use cases there are better solutions, Cactus is able to provide backward-compatibility by not requiring for there to be extra development to already existing blockchain infrastructures as all the work is done on Cactus side. This can be done as all the logic associated with the interoperability use case is implemented as a Cactus plugin. Concerning the compatibility, as Cactus uses a network of validators to issue transactions on all of the involved ledgers to achieve the required interoperability, which depends on Cactus connectors. These connectors can be developed for each different blockchain technology being used, this includes both permissioned and permissionless blockchains.

6.2 Future Work

The work done throughout this thesis shows that blockchain presents itself as a viable solution to complement or improve upon systems that use traditional solutions. Blockchain-based solutions are already being used in very diverse areas, apart from the most common one which is the financial sector with cryptocurrencies, such as secure data storage, supply chain and logistics, voting systems, and internet of things to name a few. The goal of this thesis is to further improve how organizations view blockchain technology by providing ways around the interoperability problem that is becoming one of the most important problems when it comes to blockchain technology as there is no standard yet in place and the number of new blockchain infrastructures is still increasing.

To further improve the existing interoperability solutions with Hyperledger Cactus, one of the main focuses should be increasing the number of supported blockchain infrastructures, which is work that is already being done and should be continued. Cactus as an interoperability solution highly depends on this as more complex interoperability use cases will eventually require that a wider number of different blockchain infrastructures interoperate. In the 3 scenarios explored in this thesis, we always explored with only 2 different networks.

Additionally, the work done in this thesis, and the solution provided could be looked into as a base for future systems in different areas where the decentralization of an information system could erase the need to use a third-party system. Looking into governmental systems such as the *segurança social*, *instituto de mobilidade e transporte*, *serviço de estrangeiros e fronteiras* and others which are systems which in some cases require interoperability between them and where blockchain-based solutions could help reduce the issues and delays associated with such systems by automatizing some of the processes where interoperability is required.

By exploring and further improving interoperability solutions similar to the one we explored, we can demonstrate how other areas which interact with each other at different levels could leverage blockchain solutions for their own systems without compromising this dependency they have on other systems, areas such as banking, education, healthcare, insurance, electronic identity, justice, and more.

Apart from Hyperledger Cactus other solutions should also be explored such as Cosmos which is a blockchain-focused on scalability and interoperability which aims to aggregate multiple different blockchains. It presents itself as a great enterprise solution as it effectively provides an ecosystem focused on interoperability. Cosmos subchains, which were explained before, are able to interoperate with each other, these subchains could be used as different projects of the same organization or even projects from different organizations which interact with each other, which could act as an ecosystem for organizations.

Bibliography

[1] Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., Enyeart, D., Ferris, C., Laventman, G., Manevich, Y., et al. (2018). Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the thirteenth EuroSys conference*, pages 1–15.

[2] Belchior, R., Guerreiro, S., Vasconcelos, A., and Correia, M. (2021a). A Survey on Business Process View Integration. *Business Process Management Journal*.

[3] Belchior, R., Putz, B., Pernul, G., Correia, M., Vasconcelos, A., and Guerreiro, S. (2020). SSIBAC : Self-Sovereign Identity Based Access Control. In *The 3rd International Workshop on Blockchain Systems and Applications*. IEEE.

[4] Belchior, R., Vasconcelos, A., Guerreiro, S., and Correia, M. (2021b). A survey on blockchain interoperability: Past, present, and future trends. *ACM Computing Surveys (CSUR)*, 54(8):1–41.

[5] Buchman, E. (2016). *Tendermint: Byzantine fault tolerance in the age of blockchains*. PhD thesis.

[6] Buterin, V. and Griffith, V. (2017). Casper the friendly finality gadget. *CoRR*, abs/1710.09437.

[7] Cachin, C. et al. (2016). Architecture of the hyperledger blockchain fabric. In *Workshop on distributed cryptocurrencies and consensus ledgers*, volume 310.

[8] Correia, M. (2019). From byzantine consensus to blockchain consensus. *Essentials of Blockchain Technology*, 41.

[9] Dragomiretskiy, S. (2018). The influence of ddos attacks on cryptocurrency exchanges.

[10] Frauenthaler, P., Borkowski, M., and Schulte, S. (2019). A framework for blockchain interoperability and runtime selection. *arXiv preprint arXiv:1905.07014*.

[11] Group, H. A. W. et al. (2017). Hyperledger architecture volume 1: Introduction to hyperledger business blockchain design philosophy and consensus.

[12] Herlihy, M. (2018). Atomic cross-chain swaps. In *Proceedings of the 2018 ACM symposium on principles of distributed computing*, pages 245–254.

[13] Hu, V. C., Kuhn, D. R., Ferraiolo, D. F., and Voas, J. (2015). Attribute-based access control. *Computer*, 48(2):85–88.

[14] Jain, A. and Schilz, P. (2017). 2019. Block Collider Whitepaper. https://www.blockcollider.org/whitepaper.

[15] Kwon, J. (2014). Tendermint: Consensus without mining. *Draft v. 0.6, fall*, 1(11).

[16] Kwon, J. and Buchman, E. (2019). Cosmos whitepaper: A network of distributed ledgers, 2019. *URL: https://cosmos.network/cosmos-whitepaper.pdf*.

[17] Leach, P., Mealling, M., and Salz, R. (2005). A universally unique identifier (uuid) urn namespace.

[18] Lerner, S. D. (2015). Rsk.

[19] Liu, Z., Xiang, Y., Shi, J., Gao, P., Wang, H., Xiao, X., Wen, B., and Hu, Y.-C. (2019). Hyperservice: Interoperability and programmability across heterogeneous blockchains. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 549–566.

[20] Mingxiao, D., Xiaofeng, M., Zhe, Z., Xiangwei, W., and Qijun, C. (2017). A review on consensus algorithm of blockchain. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 2567–2572. IEEE.

[21] Miraz, M. H. and Donald, D. C. (2019). Atomic cross-chain swaps: development, trajectory and potential of non-monetary digital token swap facilities. *Annals of Emerging Technologies in Computing (AETiC) Vol*, 3.

[22] Montgomery, H., Borne-Pons, H., Hamilton, J., Bowman, M., Somogyvari, P., Fujimoto, S., Takeuchi, T., Kuhrt, T., and Belchior, R. (2020). Hyperledger Cactus Whitepaper. Technical report, Hyperledger Foundation.

[23] Nakamoto, S. (2019). Bitcoin: A peer-to-peer electronic cash system. Technical report, Manubot.

[24] Ponza, A., Scannapieco, S., Simone, A., and Tomazzoli, C. (2020). Envisioning the digital transformation of financial documents: A blockchain-based bill of exchange. In *International Congress on Blockchain and Applications*, pages 81–90. Springer.

[25] Poon, J. and Dryja, T. (2016). The bitcoin lightning network: Scalable off-chain instant payments.

[26] Qasse, I. A., Abu Talib, M., and Nasir, Q. (2019). Inter blockchain communication: A survey. In *Proceedings of the ArabWIC 6th Annual International Conference Research Track*, pages 1–6.

[27] Qin, K. and Gervais, A. (2018). An overview of blockchain scalability, interoperability and sustainability. *Hochschule Luzern Imperial College London Liquidity Network*.

[28] Salomaa, A. (2013). Public-key cryptography.

[29] Scheid, E., Rodrigues, B., and Stiller, B. (2019). Toward a policy-based blockchain agnostic framework. In *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 609–613. IEEE.

[30] Singh, A., Click, K., Parizi, R. M., Zhang, Q., Dehghantanha, A., and Choo, K.-K. R. (2020). Sidechain technologies in blockchain networks: An examination and state-of-the-art review. *Journal of Network and Computer Applications*, 149:102471.

[31] Thomas, S. and Schwartz, E. (2015). A protocol for interledger payments. *URL https://interledger. org/interledger. pdf*.

[32] Vo, H. T., Wang, Z., Karunamoorthy, D., Wagner, J., Abebe, E., and Mohania, M. (2018). Internet of blockchains: techniques and challenges ahead. In *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pages 1574–1581. IEEE.

[33] Vukolić, M. (2015). The quest for scalable blockchain fabric: Proof-of-work vs. bft replication. In *International workshop on open problems in network security*, pages 112–125. Springer.

[34] Wood, G. (2016). Polkadot: Vision for a heterogeneous multi-chain framework. *White Paper*.