Tenancy over Distributed Workflows using Blockchain Technology exemplified by Network Slicing

Master-Arbeit

Jordi Bisbal Ansaldo
KOM-M-number

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Fachbereich Elektrotechnik
und Informationstechnik
Fachbereich Informatik (Zweitmitglied)

Fachgebiet Multimedia Kommunikation
Prof. Dr.-Ing. Ralf Steinmetz

Tenancy over Distributed Workflows using Blockchain Technology exemplified by Network Slicing
Master-Arbeit
KOM-M-number

Eingereicht von Jordi Bisbal Ansaldo
Tag der Einreichung: 30.05.2018

Gutachter: Prof. Dr.-Ing. Ralf Steinmetz
Betreuer: Dr.-Ing. Amr Rizk und Prof. Paul Müller

Technische Universität Darmstadt
Fachbereich Elektrotechnik und Informationstechnik
Fachbereich Informatik (Zweitmitglied)

Fachgebiet Multimedia Kommunikation (KOM)
Prof. Dr.-Ing. Ralf Steinmetz

Erklärung zur Abschlussarbeit gemäß § 23 Abs. 7 APB der TU Darmstadt

Hiermit versichere ich, Jordi Bisbal Ansaldo, die vorliegende Master-Arbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Mir ist bekannt, dass im Falle eines Plagiats (§38 Abs.2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.

Bei der abgegebenen Master-Arbeit stimmen die schriftliche und die zur Archivierung eingereichte elektronische Fassung überein.

Darmstadt, den 30.05.2018

Jordi Bisbal Ansaldo

Contents

1 Introduction **3**
- 1.1 Motivation . . . 3
- 1.2 Problem Statement and Contribution . . . 4
- 1.3 Outline . . . 4

2 Background **5**
- 2.1 Blockchain: A decentralized and distributed trustless ledger . . . 5
  - 2.1.1 Blockchain 1.0: Cryptocurrencies . . . 6
  - 2.1.2 Blockchain 2.0: Smart contracts . . . 8
- 2.2 Network Virtualization . . . 11
  - 2.2.1 Auction Mechanisms . . . 12
- 2.3 Summary . . . 13

3 Related Work **15**
- 3.1 Blockchain in Supply Chain Management . . . 15
  - 3.1.1 Blockchain Ready Manufacturing Supply Chain . . . 15
- 3.2 Network virtualization . . . 16
  - 3.2.1 Multi-Provider Virtual Network Embedding with Limited Information Disclosure . . . 17
- 3.3 Analysis of Related Work . . . 19
- 3.4 Summary . . . 21

4 Design **23**
- 4.1 Requirements and Assumptions . . . 23
  - 4.1.1 Virtual Network Embedding with LID scenario . . . 25
  - 4.1.2 Smart Contracts and Ethereum . . . 25
- 4.2 System Overview . . . 27
  - 4.2.1 Fundamental Goals . . . 27
  - 4.2.2 Architecture . . . 27
  - 4.2.3 Blockchain Types . . . 27
  - 4.2.4 Mining . . . 28
  - 4.2.5 Authentication System . . . 28
  - 4.2.6 Notification System . . . 28
  - 4.2.7 Vickrey Auction Model . . . 28
- 4.3 Summary . . . 28

5 Implementation **31**
- 5.1 Design Decisions . . . 31
- 5.2 Architecture . . . 31
- 5.3 Interaction of Components . . . 32
- 5.4 Summary . . . 32

6 Evaluation **33**
- 6.1 Goal and Methodology . . . 33
- 6.2 Evaluation Setup . . . 33
- 6.3 Evaluation Results . . . 33

6.4 Analysis of Results 33

7 Conclusions **35**
7.1 Summary 35
7.2 Contributions 35
7.3 Future Work 35
7.4 Final Remarks 35

Bibliography **35**

Abstract

The abstract goes here... (BITCOIN influence -> Blockchain). Captivate readers attention.

1 Introduction

Blockchain (BC) has been considered as one of the most promising disruptive technologies during the last years. Many market-leading companies, experts and global innovators have referred it as the "Next Generation of the Internet" [Cla17], succeeding the World Wide Web era. After evaluating its potential benefits, different banks and major enterprises, such as UBS, Microsoft or IBM, have already accomplished important investments in such innovative technologies.

The revolution started in 2008, with a whitepaper publication by Satoshi Nakamoto [Nak08], who introduced a new digital payment protocol called Bitcoin. Satoshi Nakamoto was just a name used by an unknown person or group of people to first reference the performed work. Nowadays, Bitcoin's creator still remains a mystery.

In 2009, a deployed software version based on the paper was launched. Bitcoin uses an alternative virtual currency, to make trusted transactions between different peers. This system relies on a kind of distributed database allocated on the Internet, the blockchain. Blockchain uses a peer-to-peer architecture model combined with secure algorithms, such as public key cryptography, which intentionally removes the presence of middle-parties or intermediaries. Thus, in the case of Bitcoin, it eliminates the agent responsible for transactions: central banks.

At the beginning, the blockchain architecture was restricted to only one application: online payments. However, after observing its advantages and possible use cases, an improvement of Bitcoin emerged: Ethereum[1]. By contrast, Ethereum extends the power of decentralized transactions with a Turing-complete contract system. A Turing-complete system can perform any computation, with just writing a few lines of code, in order to create Ethereum scripts: smart contracts. A smart contract can be generated with non-restrictive and user-friendly programming languages, allowing developers to easily learn and benefit from them. Therefore, it brings to the user the opportunity to develop their own applications. Smart contracts are currently used to implement decentralized apps, which unlike normal web apps, they may not be allocated in a central server. In other words, they use blockchain technology to retrieve and store data instead of a database.

1.1 Motivation

Nowadays, blockchain is becoming a trending topic in the business world. Thousands of articles, research papers and books, such as: How the technology behind bitcoin is changing money, business and the world [TT16] or Blockchain: Blue print for a new economy [Swa15], are catching the public eye. Nevertheless, as it is an emerging technology, a necessity to look towards new horizons exists. Through the use of the above mentioned Ethereum smart contracts new possibilities to approach existing problems are opened up. Hence, blockchain technology can be used in many applications beyond currency.

Many scenarios are currently investigated from a blockchain perspective, e.g: candidate's voting or asset tracking [AM16]. In the former, voters send signed and encrypted ballots to the blockchain contract, who immediately verifies them. Simultaneously, it also preserves confidentiality, since the ballot can only be emitted from its owner. In the latter, each physical asset could be encoded in the blockchain enabling a fast and transparent tracking. For example, Everledger[2], a startup company from London, tracks diamonds storing each diamond's digital identity on the BC. Thus, diamond theft could be effectively prevented.

After investigating the implementation of blockchain in real scenarios, one top-level application is clear: it can be used as a software connector. Blockchain contributes to face security, storage's prob-

[1] https://www.ethereum.org/

[2] https://www.everledger.ioafa

lem, communication and coordination between users. In this paper, we will focus on enterprises or organizations, suffering from such problem, referenced from now on as supply chain challenges.

1.2 Problem Statement and Contribution

Supply chain management is the process of linking organizations through information flows, in order to achieve a competitive strength or advantage, which will maximise customer value. Supply chain activities go from the design or development of a product, up to its return on investment (ROI). Thus, a good coordination during these activities is extremely needed.

Nowadays in the Big Data era, enterprises must handle a huge amount of information. This leads companies to suffer from considerable issues, such as scalability, data's security or communication. One could think that currently most of these companies rely on third parties, which help them on the mentioned problems. But what if this process could be efficiently accelerated in a secure and decentralized manner? Here, is where Blockchain can play a crucial role.

During the paper, the focus will be on IT companies facing this dilemma. The scenario will include in one side different customers, and in the other organizations acting as providers. For example, eBay, one of the biggest multinational e-commerce corporations, acts as an intermediate for the product's purchase-sale. Thus, eBay is responsible for managing all this data. However, can a user/company always safely trust third-parties? Why not using BC as the main responsible for handling such complex tasks? This will result in a direct customer-provider relation, avoiding the presence of any intermediaries.

For this scenario, a current application example that consists of the embedding of virtual networks (network virtualization) between different Infrastructure Providers (InPs), will be further investigated [DRP15]. This process can also be called network slicing. In this example, Service Providers (SPs) want to embed virtual nodes among different InPs in order to provide wide-area network services. Nevertheless, Infrastructure Providers are not willing to publicly disclose its internal network topology, along with its resources availability and costs. In such cases, brokers, usually known as VN Providers (VNP) try to perform the embedding under the mentioned limited information disclosure (LID) problem. As a result, it can be clearly observed that blockchain can solve this interaction, providing: secure sensitive data storage, customer-provider negotiation without third-parties (without VNP) and finally maintaining a coordinated process. The negotiation between the involved parties will be based on a time-limited auction system, where each virtual network request will be stored as a contract on the blockchain network.

Therefore, a good question for the theses could be: How blockchain takes advantage of distributed workflows providing an agile and secure environment? Exemplifying workflows, with the network slicing example. At the end, a decentralized application approaching the mentioned problem will be deployed. In addition, the app will include a user-friendly front-end in order to guide users through the process.

1.3 Outline

This thesis is structured as follows. In Chapter 2, relevant background on blockchain, network virtualization and auction mechanisms is given. Afterwards, in Chapter 3 an overview and analysis of existing research about blockchain in supply chain management and multi-provider virtual network embedding is presented. Based on these investigation, in Chapter 4 an application system design to solve the virtual network embedding problem using blockchain is exposed. Then, the implementation of the proposed design is discussed in Chapter 5. The created application along with its technologies are tested from different perspectives in Chapter 6. Last but not least, in Chapter 7, the conclusions of this thesis are exposed as well as possible improvements for future work.

2 Background

In this section, an overview of the blockchain technology evolution will be provided. It starts with the 1st blockchain generation related to cryptocurrencies, with Bitcoin as a leading representative. Then, the second or smart contracts generation will be investigated, where Ethereum extends the idea of money transfers, to any other application that can be writable as a piece of code.

Afterwards, the network virtualization concept will be introduced, in which resource negotiation between customer and providers is crucial. Due to this importance, a well-known public negotiation mechanism will also be presented: the auctions.

2.1 Blockchain: A decentralized and distributed trustless ledger

A blockchain is a decentralized distributed ledger, which stores the entire history of transactions on the network. In other words, it is a simple database distributed among a network of computers, where each computer has an identical copy of this database. This contrasts with traditional (e.g. SQL) databases that are controlled by a single entity. Thus, in a blockchain, there is no central server or agent in the middle of the communication. For example, imagine a user willing to transfer money to another one 2.1. In a centralized system, the transaction will go first to the bank, who will update its internal database and subsequently perform the operation. In contrast, by a decentralized system, each user is able to directly transfer the money, since it possesses an updated copy of the database. Another example to replace a centralize design could be in the healthcare environment. There, patient records are stored in multiple databases, which always leads to a costly exchange of information between them. In this scenario, the blockchain could improve the process, preserving patients confidentiality in a secure and decentralized manner.

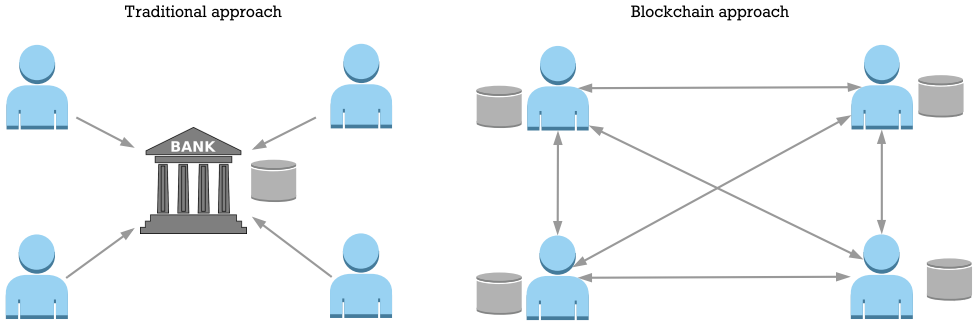

Figure 2.1: Traditional database vs blockchain approach

At the beginning, the terms Bitcoin and blockchain were sometimes interchanged, as these words were used to refer: (i) the technology, (ii) the protocol for making transactions and (iii) the cryptocurrency (money). Therefore, before continuing, one statement needs to be clear: Bitcoin is a cryptocurrency that uses the blockchain technology. Hence, Bitcoin is just one of the multiple applications that use blockchain.

However, when a new technology appears, the first user's goal is normally to exploit its economic potential. For this reason, money transaction through cryptocurrencies was its first application. In the next subsection, we will understand what cryptocurrencies are, followed by an explanation of the Bitcoin's design architecture.

2.1.1 Blockchain 1.0: Cryptocurrencies

At of the end of January 2018, Coinmarketcap[1], a cryptocurrency market tracker, lists more than 1,400 cryptocurrencies with an aggregate value approaching USD 700bn. But what are cryptocurrencies? Cryptocurrencies are a variety of digital currencies pretending to work as a medium of exchange, such as Euro or USD does. As the name suggests, apart from being a virtual currency, they use cryptography to secure and verify its transactions. The main difference with traditional currencies is that they do not have any physical equivalent in the real world. Nevertheless, they can be used to pay goods and service, with the advantage of not being constrained with geographical or political borders. For example, GMO Internet[2], a Japanese company, will start paying parts of employees salaries in cryptocurrencies (Bitcoin).

In this paper, we will set aside whether they can become true currencies or not, and also its political, social or economic impact. The only focus will be the technology behind it, as blockchain can be extended to much more than digital currencies. Thus, we will start from the genesis of the technology, with Bitcoin as its revolutionary innovator.

Bitcoin

Bitcoin took the world by surprise in 2008, after Satoshi Nakotomo's white paper publication [Nak08] and later its software release. Bitcoin (BTC) is a cryptocurrency used for making secure transactions across a peer-to-peer (P2P) network. In addition, Bitcoin uses its own protocol that operates in an overlay network, the blockchain. An overlay network is a computer network build in the top of another network, in this case above Internet application's layer, which is controlled by its users (no central authority).

From another point of view, the Bitcoin ledger can be interpreted as a state transition system, where there is an initial state, which after a transition function (money transfer), results in a new state. Imagine an scenario where user A wants to send 10 BTC to user B. The first state is A and B current balance and the transition function will take 10 BTC from A and insert it to B's account, generating a new state. But what happens if A sends exactly the same payment to two different addresses (B and C) at the same time? This scenario is the so-called *double spending attack* and consequently, a transaction always needs to be verified by miners before being confirmed.

Mining and Transactions

Miners are specific blockchain users, responsible for monitoring and verifying all the transactions between users. And how all these miners cooperate efficiently? The answer to this question is one of the most remarkable Satoshi innovation key factors, which consists of the communication between nodes through a simple decentralized consensus protocol. This consensus protocol consists of multiple algorithms (e.g. Proof of Work), used by the miners in the Bitcoin network.

Therefore, Bitcoin needs to combine the state transition system with a consensus protocol, in order to synchronize the order of all transactions among the users. There, new transactions are stored in the last block of the blockchain, and a new block is mined every ten minutes. Over time, this creates an ever-growing chain of blocks, which are constantly updated. Thus the name: blockchain. Additionally, a complete history of the transactions is kept, so everyone can verify the last money movements. For instance, a blockchain can be compared to an endless domino game, where all the pieces are placed in vertical one after the other. Each of these pieces references a block, and if one block is removed (e.g. transaction error), all the subsequent ones will be affected. Hence, as miner's task is not simple and requires computational power, if they are the first block solvers, they are also economically rewarded. Additionally, if any transaction has a positive balance at the end, it is also destinated to the miner.

Figure 2.2 shows how transactions are bundle into blocks, where each block contains a:

[1] https://coinmarketcap.com/
[2] https://www.gmo.jp/en/

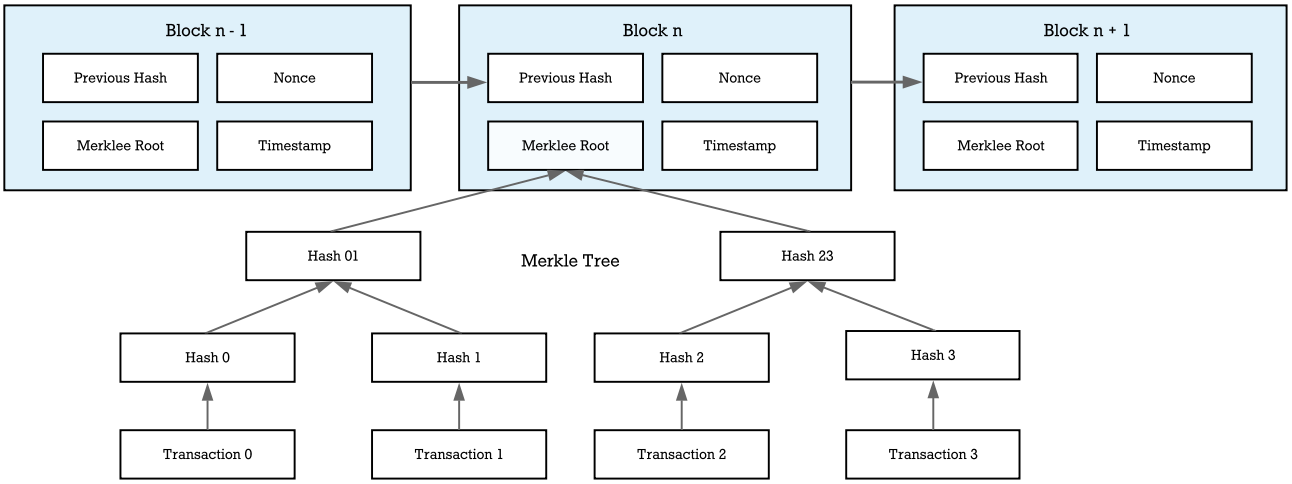

Figure 2.2: Blockchain transactions. Adapted from [HZRS17]

- Timestamp, to identify when the event occurs.
- Nonce used to avoid malicious nodes from flooding the blockchain.
- Hash of its previous block, to keep track of already added blocks.
- A list of all the transactions that have been created since the previous block. To save storage, this transaction list is typically stored in a a Merkle tree [Mer87].

Furthermore, in Bitcoin, each block and transaction is restricted to a size of 1MB and 250-300bytes respectively, and for a block being valid it needs to satisfy the following requirements:

- Previous block referenced exists.
- Block's timestamp is greater than previous block one.
- Proof-of-work in this block is valid.
- If any transaction from the transaction lists returns an error, exit.
- If all previous steps confirmed, store state at the end of the block and return true.

In the third step of the block validation process, appears the term *Proof-of-work*. Bitcoin uses the Hashcash proof of work algorithm to prove that a block miner spent some computational time creating a block. More precisely, Bitcoin protocol demands that miners find an input, which is a valid $blockheader$ formed by a random value (c) and nonce (x), whose cryptographic hash (e.g $SHA256$) is less than a determined value. This value is obtained from d, the blockchain difficulty. Then, the only way to create a valid block is simply trial and error until a valid Proof-of-Work is generated. Proof-of-work is also used in other contexts, such as for limiting email spam or denial-of-service attacks (DoS).

$$F_d(\,blockheader\,) = F_d(\,c,x\,) = SHA256(\,SHA256(\,c|x\,)\,) < \frac{2^{224}}{d}$$

Key management

In Bitcoin, each user is identified by a single public address. However, how is this Bitcoin address generated? This process involves the following steps:

1. A random 256-bit private key is created. Since this key will be used to sign the transactions, it must be kept secret.

2. A 512-bit public key is generated from the private key, using the Elliptic Curve Digital Signature Algorithm (ECDSA[3]). This key is used for verifying private key signatures.
3. This public key is hashed to 160 bits using SHA-256/RIPEMD. Apart from size constraints, the reason for hashing the public key is that if there is a vulnerability in elliptic curves, user's money can still be safe, since only the hash is known.
4. Finally, the public key is encoded in ASCII using Base58Check. This output is the resulting Bitcoin address.

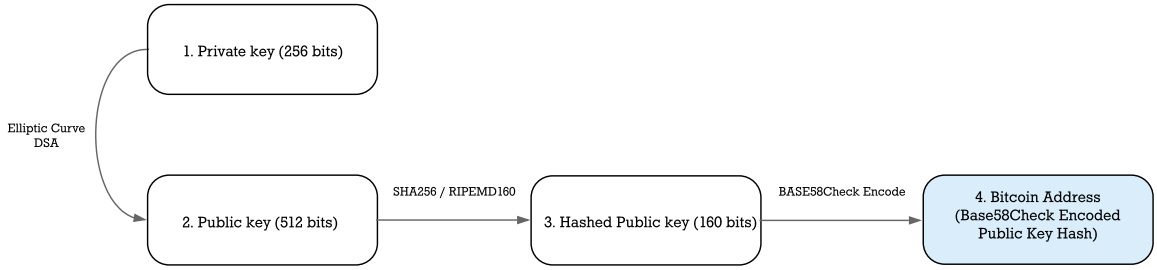

Figure 2.3: Bitcoin keys and addresses generation [Shi14]

After that, users are able to send transactions using Bitcoin. For example, imagine the last example where user A wants to send 10 BTC to user B. Firstly, the user creates a transaction willing to transfer 10 BTC to user address B. Secondly, he signs the transaction with his private key, attaching also his public key, and sends it to the blockchain. There, miners will verify the signature using A's public key and also that its hash matches user A address. If both, the signature and the hash, are correct, the transaction is accepted and added to the next block. Transactions examples can be found in Block Explorer[4].

2.1.2 Blockchain 2.0: Smart contracts

The Blockchain 1.0 had numerous limitations since essentially it only approaches the decentralization of money and payments. However, the architecture implemented by Bitcoin is extensible beyond financial uses cases.

Ethereum

In 2013 Vitalik Buterin, a Russian-Canadian programmer released the Ethereum white paper [B+14], where he describes an alternative platform running in a blockchain that allows building any kind of decentralized application. In 2014, Ethereum started as a crowdfunding project, which collects small amounts of money from a large number of users. Currently, Ethereum is a running project and the second most valuable cryptocurrency.

Furthermore, it is built on a Turing-complete contract system. As previously explained, a Turing-complete system allows the developer to perform any computation, which runs on the so-called **smart contracts**. These smarts contracts are executed by the Ethereum nodes, each using its Ethereum Virtual Machine (EVM), or in other words, a blockchain with a built-in programming language.

In the end, smart contracts are basically Ethereum scripts that whenever they are called, the function programmed inside of it is executed. A smart contract can also be seen as the logic inside a vending

[3] https://www.maximintegrated.com/en/app-notes/index.mvp/id/5767
[4] https://blockexplorer.com/

machine. For instance, a buyer wants a Coke bottle that costs 2 €.. Once this buyer inserts 2 € and presses the button, a small program (contract) runs inside the vending machine and supplies the user with its Coke. This function, in the smart contract context can be seen as:

```
contract Products {

  function getItem(bytes32 buttonPressed, uint amount) returns(bytes32 item) {
    if (buttonPressed == 'coke' && amount == 2) return coke;
    return null;
  }
}
```

Therefore, smart contracts allow anyone to write their own functions, in just a few lines of code. Despite Bitcoin and Ethereum have similar features, such as decentralization or transaction-based, Ethereum makes use of smart contracts, which potentially opens up real-world use cases. Bitcoin and Ethereum can be comparable to a calculator (one application) and a smartphone (multiple applications) respectively. In addition to the already mentioned, there are several differences that need to be highlighted:

- Ethereum's block time is shorter, specifically around 14s. Remember that Bitcoins mining time was around 10 minutes. Hence, as a consequence, Ethereum needs to handle extremely large amounts of transactions in seconds, which means that the network can easily get stuck with different subchains. To solve this scalability issues, Casper, a new consensus strategy has been recently deployed [Ros17].
- Bitcoin's blocks and transaction sizes are indicated in bytes, however, in Ethereum, it depends on the contract complexity. This complexity is expressed in terms of gas, which is the amount of ether [5], used for executing contracts in the Ethereum blockchain.
- Ethereum moves from the Proof-of-Work to the Proof-of-Stake (PoS) concept. In contrast to PoW, where the miners that first solve a mathematical problem are rewarded, in PoS, the block's creator is selected in a deterministic way (wealth depending or stake). Also, a miner does not receive a block reward, but instead he earns all gas used for executing the smart contract.
- Ethereum transactions compared to the mentioned Bitcoin fields consists of (i) the transaction value in Ether, (ii) recipient address, (iii) data arguments and (iv) execution cost.
- Ethereum is account-based and not transaction-based.

In the last point, we observe that Ethereum introduces a new concept called accounts. These accounts are unique 20-byte addresses in the blockchain, each of them having a balance controlled by ethers (ETHs). In Ethereum, there are two types of accounts (Figure 2.4):

- **Externally Owned Accounts (EOAs):** These accounts normally identify a user, being controlled by their own public and private keys. Only EOA can send transactions to other accounts. A transaction in Ethereum can either send Ethers or call/trigger a contract account
- **Contract accounts:** These are the accounts, where the code (smart contract) is stored. Thus, once triggered (function call from another contract), its defined code is executed.

However, what happens if this contract code results in an infinite loop? The node will get stuck the whole time executing this contract. Thus, Ethereum uses the above-mentioned term: gas. For example, suppose an EOA calling a contract account for money transfer. Apart from the ether transferred, it will also spend an additional amount of ether (gas). Thus, the ether used for a transaction, or transaction fee is:

[5] ether or ETH is the digital currency used by Ethereum

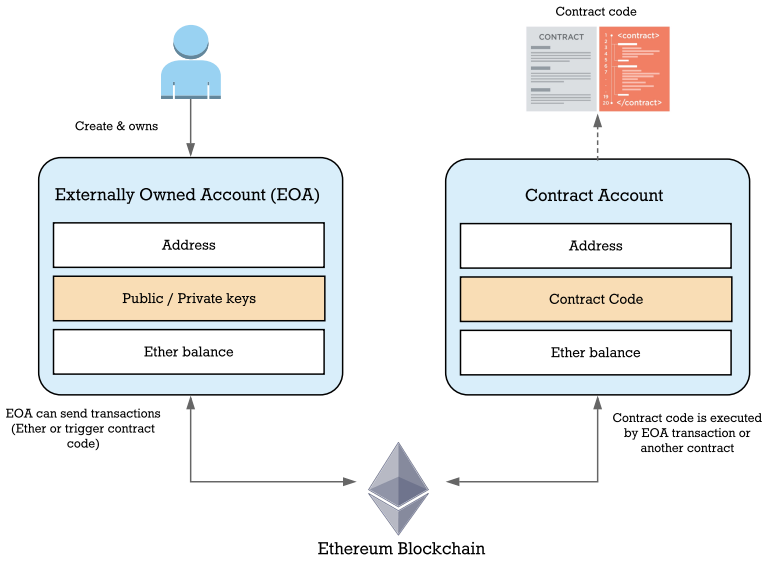

Figure 2.4: Ethereum accounts: Externally owned accounts and contract accounts

$$Transaction\ fee = gasprice \times gaslimit + value - unusedgas$$

Firstly, the $gaslimit$ is the maximum amount of gas that a sender is willing to spend on a transaction. By default 21000 is the minimum gas limit for all transactions. Fortunately, all $unusedgas$ during a transaction ($gaslimit - realgasused$), is refunded to the sender. Nevertheless, if a transaction gives an error, e.g: a fake transaction, this provided ether (gas limit) will never be refunded. Secondly, the $gasprice$ is its relation with real Ether, e.g: 40 Gwei (4e-8 ether). Thirdly, the $value$ is the amount of ether transferred to the other EOA.

In conclusion, a smart contact is just an account containing code, which lives on the blockchain and allows developers to create their own decentralized applications.

Decentralized applications

In the Ethereum white paper [B+14] dapps are split into three types. The first group includes financial applications, where users exchange ether in an efficient and distributed manner. The second group is formed by semi-financial applications, in which money is involved but it is mixed with data from the real world, for example with weather feed. And finally, in the third category, there are non-financial applications. For instance, an online voting platform providing a better transparency into elections, without compromising voter confidentiality.

Despite the blockchain is a secure technology, the created applications are constantly compromised to multiple attacks. This is because the majority of smart contracts are insecure, due to its code being prone to bugs. For example, in June 2016, the DAO[6], a distributed autonomous organization instantiated on the Ethereum blockchain, was hacked. The attack, which was a smart contract bug, had resulted in over 50 million dollars of cryptocurrency theft. Thus, smart contracts need to be protected against malicious attacks before being uploaded on the blockchain.

In the following section, an example application or process currently performed by IT companies will be introduced.

[6] https://ethereum.org/dao

2.2 Network Virtualization

In the last few years, network virtualization has started to grow in popularity since it enables simulating hardware networks, in software. These software networks are called virtual networks, and in contrast to physical networks, they provide flexibility, manageability and a low cost.

Network virtualization handles two main concepts: node virtualization and link virtualization. The former implies sharing the node's physical resources located in the substrate network (e.g: CPU, storage, memory) to multiple virtual nodes in the virtual networks, such as physical node A in Figure 2.5.a. The latter enables the transport of multiple virtual links in a single and shared physical link (Figure 2.5.b).

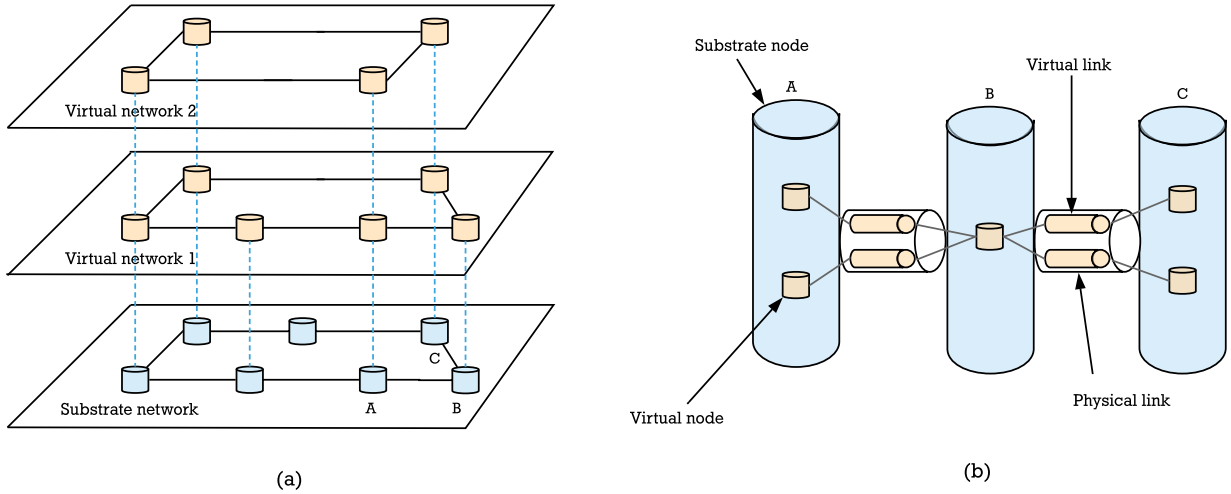

Figure 2.5: Network virtualization model and its basic elements [CJ09]

In a real scenario, there are two main actors (and one optional) participating in the network virtualization process.

- **Infrastructure providers (InPs):** Infrastructure providers (e.g: Deutsche Telekom or Telefonica) deploy and manage the physical nodes, sharing efficiently its resources to the virtual network. For InPs, this results in a significant decrease in operational and technology investment costs [DRP15].
- **Service Providers (SPs):** Service providers lease virtual resources from infrastructure providers to create its customized virtual network. All this without the need to purchase physical network equipment.
- **Virtual network providers (VNPs):** Are the brokers, which can optionally serve as intermediaries between the infrastructure and service providers.

The network slicing process with the presence of a broker typically starts with the VNP obtaining the necessary information from the InPs. This data will be used to divide the virtual network into groups of nodes (subgraphs). Thus, once the SP sends a VN request, this is immediately optimized (in terms of cost) by the VNP. Afterwards, the modified request is transmitted to the multiple InPs, who are in charge of allocating the virtual nodes from the request in their physical resources.

The presented process is coordinated by an actor, the VNP, who stores all the information to successfully allocate the incoming resources. However, there are other approaches which do not rely on a central entity. Conversely, they use a consensus mechanism between the SPs and InPs to perform the virtual resources trading. Thus, in the following, we will briefly investigate an open negotiation technique: the auctions.

2.2.1 Auction Mechanisms

Service negotiation has already been examined in many researchs, such as [HS05], [OV02], which have worked with diverse auctions platforms. An auction is a public negotiation method involving buyers and sellers that enables resource trading between peer-to-peer users. While buyer's goal is to find the desired service at the lowest price, providers or seller's goal is to catch the attention of buyers who are able to pay the highest price for its offered services. Thus, an auction must provide fairness in terms of technical and economic efficiency, for both the buyer and the seller.

Firstly, auctions can be divided into (i) *one-sided auctions* where only the buyers submit their bids (e.g: a painting auctioned in an art exhibition), or (ii) *two-sided auctions* in which buyers and sellers submit their bids. Normally, when any of the two sides (buyer or seller) cannot perform a good estimation of the service, a one-side auction is preferred. This is the case in network virtualization, where the SPs are not aware of the InPs physical nodes availability or complexity. Secondly, the auctions are (i) *open-cry* auctions, where the bids are broadcasted to all users, or (ii) *sealed-bid* auctions in which bidders first commit bid values in secret that are later revealed (e.g: physical envelopes). In our scenario, the InPs should bid in a confidential manner, because they do not want to disclose internal data. In addition, concerning auction types, our focus will be on the most common [CST80]:

- **English auctions**, also called *open-cry ascending-price* auction, is the classic bidding mechanism where the seller first sets a minimum starting value (reserve price), which buyers should overcome. Then, these buyers start to offer higher prices until nobody wants to offer a greater one. However, due to the competition, this auction model can result in a long process (e.g. low reserve price compared to real value) where users pay huge amounts.
- **Dutch auctions** are similar to the English auctions, but on the contrary, they are *open-cry descending-price* auctions. In this model, seller fixes a maximum value which is reduced slightly over time. Once a buyer accepts a price, the auction will finish. Nevertheless, to prevent ending with low prices for the sellers, they also establish the mentioned reserve price, which will be the minimum accepted amount. Dutch auctions are faster than the English ones, but buyers will also end up paying more to ensure that they win.
- **Vickrey auctions** [Vic61] has the particularity of corresponding to a *sealed-bid second-price* auction. Since it is sealed, during the bidding time buyers do not know other bids and eventually, how the auction is evolving. In addition, the Vickrey auction corresponds to a second-price tender. This means that like standard auctions, the bidders will offer a price for the service, and the highest bid will win. Nevertheless, this service will be rendered at the second highest value. For example, if the winner bids 15 € and the second highest bid is 10 €, he ends-up paying only 10 €. From the bidder point of view means that he can safely bid the true value, knowing that if he wins, he will pay less than his bid. Thus, a Vickrey auction compared to English and Dutch auctions, is considered a fair-price system since it provides a reasonable price to the buyer by motivating bidders to bid truthfully.
- **Reverse auctions** as the name states, buyers and sellers roles are switched. In contrast to ordinary auctions, where buyers compete to obtain an asset, in reverse auctions, sellers compete to gain a business. In other words, the seller bids for his own provided services, which in the end, will be given to the supplier offering the lowest price. For example, in the network virtualization process, InPs will bid for the lease of their physical nodes, and afterwards, the SP will rent the most economical ones.

Ultimately, after judging the preceding analysis regarding auction types, a ***one-side reverse Vickrey*** auction seems to be the most suitable for network virtualization since it provides confidentiality and the biggest aggregate profit among sellers and buyers. This will be later investigated in section 4.

2.3 Summary

The blockchain is a transparent, secure and robust system where users are in charge of their own accounts and transactions. Beyond money transfer, which starts with Bitcoin, the blockchain technology can be used as a software connector in multiple scenarios. Due to smart contracts, any kind of application could be implemented, enabling developers using the BC concept in their own environment. In the next chapters, the already presented network virtualization scenario will be further investigated.

3 Related Work

In the following, we explore some of the noted challenges that organizations encounter during their supply chain management (SCM) process, and how they currently approach these issues. Then, we will introduce a recent application in manufacturing, to demonstrate how blockchain could improve SCM efficiency. Henceforth, we focus on a particular supply chain management example: virtual network embedding across multiple InPs. A comprehensive overview of related work on already implemented centralized and decentralized frameworks will be presented. Finally, we will first enumerate the main blockchain benefits observed from the SCM manufacturing application, and secondly, we will analyze the pros and cons of the existing VNE related work, in order to observe the main problems and limitations that need to be addressed in our thesis.

3.1 Blockchain in Supply Chain Management

In the last years, managing the information during the lifecycle of a product represents a major challenge for all companies [KARF03], [Tut02]. This product information is constantly changing and in addition, it has to be accessible by different entities. As a consequence, this problem results in high costs for companies.

Typically, companies approach the problem: (i) storing and maintaining its data into their company-specific infrastructures, which is then communicated to the other supply chain partners, or (ii) sharing this data in a centralized database. In the former, each enterprise creates its own copy of the product data, using their own protocols and procedures. As a consequence, lots of information asymmetries are found, which ends up destroying the benefits of knowledge or data sharing. In [LB92] and [Fia05] inadequate definition of customer services, poor coordination or organizational barriers are named to be some of the pitfalls that these users suffer from. On the other hand, the latter is a great solution as long as the companies trust the party who is maintaining the database. However, what if they are not willing to place this intermediary on their operations? Here, we foresee that blockchain could improve the process supplying data distribution and storage among these companies.

Apart from the supply chain, [Hei02] states another relationship between suppliers and customers, the Demand Chain Management (DCM), where we predict that blockchain could also play a significant role. In a DCM, the aim is to provide a customer service at the least cost. Hence, the suppliers need a real-time visibility of customer situations and needs. The main difference is that in SCM, the stakeholders wait to receive the order for proceeding with a product (push), whereas in DCM the suppliers observe and immediately operate in user's petition (pull) [WG17]. Currently, companies such as Everledger or Skuchain[1], offer blockchain services to manage and improve the supply chain performance.

In the following section, a blockchain application for a manufacturing supply chain system will be presented.

3.1.1 Blockchain Ready Manufacturing Supply Chain

In this part, a blockchain example [AM16], which stores and distributes specific product information during its lifecycle, will be discussed. This task also named product lifecycle management (PLM [Sta15]), involves different stages in which multiple parties modify the product data. Figure 3.1.a shows the typical stages on this cycle. In this scenario, each actor interacts with a user interface, which is connected to the product's data from the blockchain. Each product is created in form of rules (smart contract) so that only specific users can access or modify this information. To access or insert data into the contract,

[1] http://www.skuchain.com/

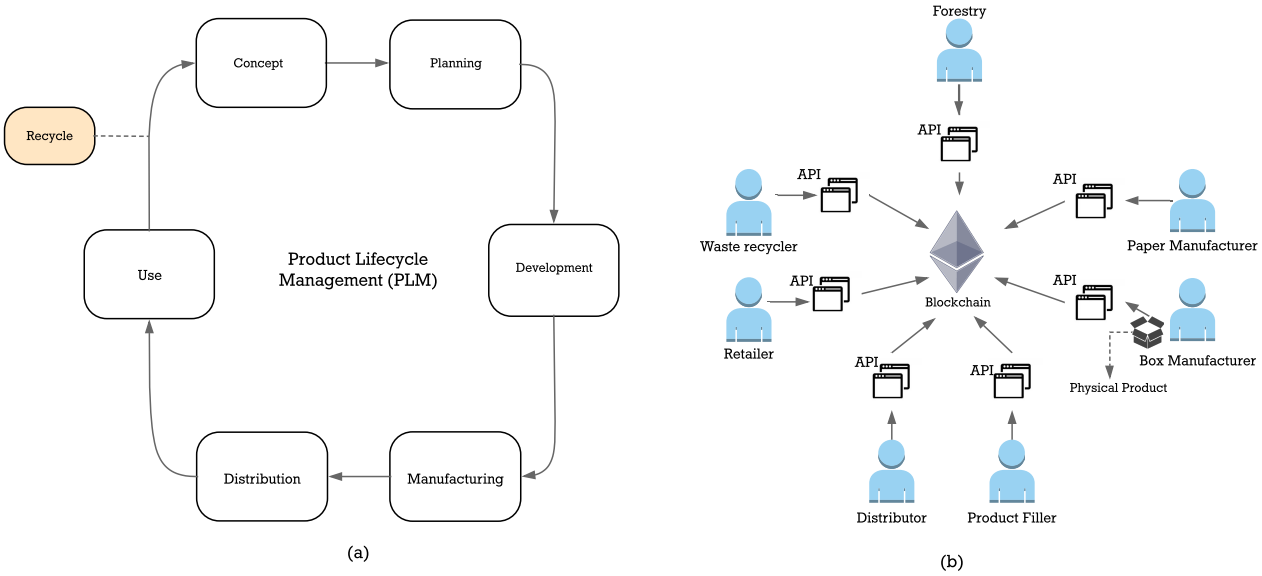

Figure 3.1: PLM stages and cardbox blockchain application example [Sta15]

parties must authenticate themselves signing the request with their private keys. During the lifecycle management, a product is owned by an entity (e.g. distributors). Moreover, when a product goes to another actor (e.g. from distributors to a consumer), both parties will sign a contract that updates product's ownership. Thus, the system guarantees that only the granted user can modify the contract in each phase.

In [AM16], an example is also presented to better explain the mentioned approach. The application consists of the product lifecycle of a cardboard box, starting from the trees cut down until the box is recycled, in which the involved parties are previously registered to the system. In Figure 3.1.b an overview of the implemented design is presented. In this business process, different actors are entailed. For instance, the box manufacturer first receives the physical object and then signs a contract to gain the product's access. Afterwards, he will be allowed to alter the product data through the software application (e.g. box quality). In the end, he will transfer the package (signed) to the next actor, in particular, the product filler.

3.2 Network virtualization

In the last years, many investigations in network virtualization have been conducted by different authors [HLAZ11], [ZZSRR08], [CRB09]. Among these, it is important to highlight topics such as the trade-offs between single [CRB09], [HLZ08] and multi-provider VNE [DARP17], or the different techniques and algorithms used for achieving VN efficiency. Typically, they decompose the VNE into two tasks: VN partitioning and VN segment mapping [FBB+13].

Each of these studies proposes new interesting features compared to the prior research. Nevertheless, in this paper, we will not contrast the different architectures (e.g. between a single or multiple InPs) or discuss the various VN partitioning and mapping algorithms. The main goal of this work will be to explore an existing network virtualization scenario, in particular, multi-provider VNE applications suffering from the limited information disclosure problem or LID [DARP17], [ZXB10], [EDPM13], [CSB10].

3.2.1 Multi-Provider Virtual Network Embedding with Limited Information Disclosure

To deploy wide-area networks, service providers are willing to embed virtual resources across heterogeneous domains belonging to different infrastructure providers (Figure 3.3). As a result, the system avoids being restricted to just a single provider topology, providing at the same time better efficiency. This process, also named inter-domain VNE[2], is descomposed in three main tasks [CSB10]:

- **VN request partitioning**: Divide the virtual network into groups of virtual nodes (subgraphs), such that virtual node and link requirements are satisfied, minimizing at the same time the cost of the virtual network setup.
- **Inter-connection between subgraphs**: Establish paths between the previously created subgraphs.
- **VN segment mapping**: Map the virtual resources (subgraphs) to the InPs physical nodes.

Furthemore, in such scenarios exists two types of communication:

- **Horizontal communication** is the one established between infrastructure providers to guarantee the most efficient end-services. These relations stated in [ZXB10], can be: *public relations* established using a market mechanism (e.g: an auction), or *private relations* already arranged. In both cases, the relation arises from the need to negotiate and cooperate to serve the SPs.
- **Vertical communication** emerges from the negotiation between SPs and InPs, the former willing to lease some virtual resources from the latter. This communication is facilitated by a third party (VNP), an intermediary that forwards SPs request to the relevant InPs.

One of the main challenges of inter-domain VNE, is that InPs are not willing to broadcast their resources information (e.g: nodes availability, cost) or their network topology to the outside world [DRP15]. Thus, the virtual resources need to be allocated in a constrained and limited scenario (LID). And if that was not enough, this InP's data is crucial to perform the partitioning of resources, in order to obtain a fair negotiation between the SPs and InPs.

Centralized Slice Embedding

In [DRP15], a centralized VNE framework is implemented (Figure 3.2.a). There, the mentioned LID partitioning problem is approached through a VPN, who accesses public information that is not considered confidential by the InPs:

- **Virtual resources availability**: In this case, virtual resources examples are obtained from Amazon EC2 [ama], which announces the attributes (CPU, memory, cost) of different instances types to give the user facility in choosing the virtual resource that better fits in its application.
- **Substrate network topology**: Most of network topology information is treated confidentially for the InPs. Nevertheless, there are certain aspects of the network which are not considered private, such as InPs peerings (including location) and its related link cost (Figure 3.2.b).

Furthermore, in [DRP15], the slice embedding is achieved by subsequently perfoming the following tasks: Firstly, the above-mentioned resource information is sent from the InPs to the VNP, who registers and stores it. Secondly, the VNP matches the service provider request with this collected data. Thirdly, the corresponding partitioning and mapping algorithms are applied (not discussed in this paper).

In addition, the results prove that this centralized VNE approach provides embedding costs not much higher than in an ideal case scenario (InPs announcing all required information) and a high ratio between the number of slices requested and later allocated. Despite the positive findings, this scenario suffers

[2] inter-domain is between multiple InPs and intra-domain is inside a single InP.

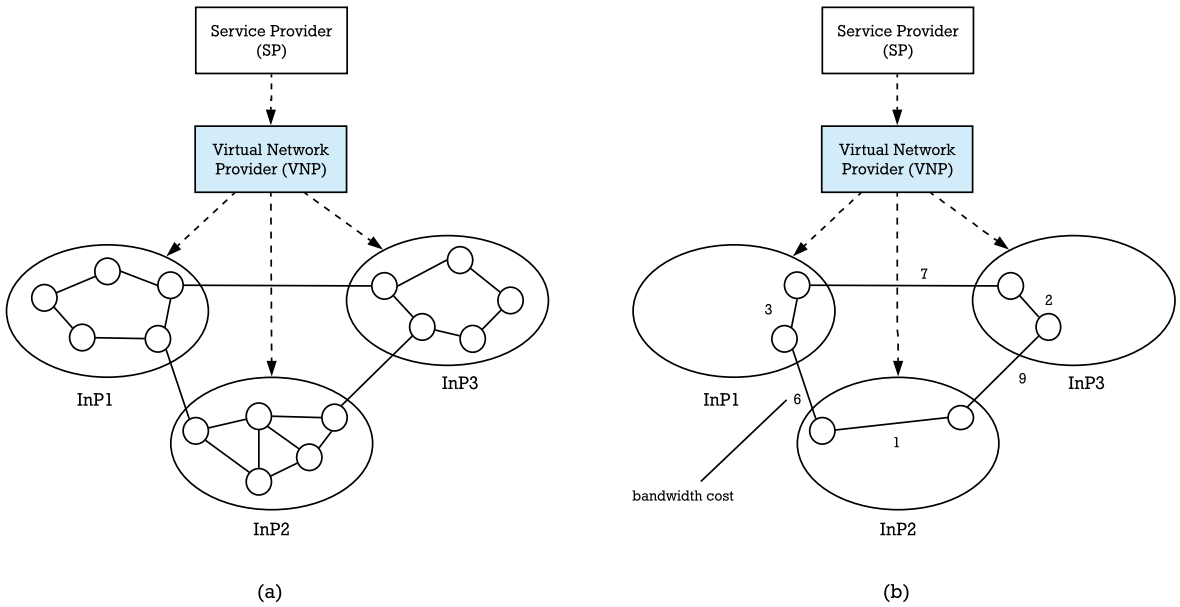

Figure 3.2: (a) Virtual network request across multiple InPs using a VN Provider. (b) VNP's view on the substrate network topologies. [DRP15]

mainly from scalability, since everything relies on a single centralized authority: the virtual network provider. In addition, it assumes that InPs (if they do so) will be constantly advertising their updated data to the VNP, which can lead to undesired costs.

Therefore, other approaches where the actors benefit from a more scalable and distributed slice embedding need to be further investigated.

Distributed Slice Embedding

Many existing solutions [HLAZ11], [DRP15], [DARP17] rely on a centralized party (VNP) that stores the InPs disclosed information and then performs the partition of the virtual network request. However, what if the SPs and InPs are not willing to trust a centralized broker? In that case, this central actor could be removed enabling a direct communication between SPs and InPs, which provides a better system scalability. Distributed network slice embedding has already been investigated in:

- In [EDPM13], a **consensus-based auction for distributed slice embedding (CAD)** is proposed. In particular, the physical nodes, which are owned by a single or different InPs, bid on virtual nodes. This value is subsequently stored in a vector b_i, where i is the physical node. Afterwards, once the bidding phase concludes, each physical node exchange the bids with its neighbors to reach an agreement for the auctioned virtual nodes. The mentioned bidding can be for a single slice (2 virtual nodes and 1 link) or on the entire slice (multiple virtual nodes and links). In the former, there is a limit on the number of biddable nodes, which although it has a great performance, it provokes multiple iterations (excessive time). Despite in a multi-provider VNE scenario, the latter is more suitable, the InPs willingness to disclose their internal information will hamper the performance. Hence, this approach provides better scalability (decentralization), but it is not proper for our scenario since it does not solve our LID problem.
- In [CSB10], a **policy-based inter-domain VN embedding framework (PolyViNE)** is presented. In PolyViNE, each InP embeds part of the VN request under its internal policies, while they cooperate with other InPs in a decentralized manner. In addition, forward decisions are location-based. For example, in Figure 3.3 each virtual node (A,B,C,D), has a prefered geolocation. After that, the

matched InPs will receive a notification for embedding the specific virtual node (e.g. InP3 with virtual node D). This whole process starts with the SP sending its VN request to multiple trusted InPs who reply back with the related embedding and its prices (bidding). Then, the SP will choose the most economical embedding. However, since there is no central broker, to ensure performance each SP must know $k^{SP} \geq 1$ InPs and each InP $k^{InP} \geq 1$ InPs. Also, most of the time a complete VN is not mappable by a single InP. In this case, the InP embeds its part and forwards the other to a known InP.
Despite, PolyViNE introduces a decentralized approach dealing with LID and uses the interesting location-assisted embedding, it has some drawbacks that need to be covered. Firstly, each InP has an extra overhead in communicating the rest of the request when the VN is not fully mapped. Secondly, the bidding to provide a competitive market is a good idea, but in this scenario, the bids can be publicly known by other InPs. Thus, a better bidding process preventing InPs from revealing its confidential data is needed. Finally, the requirement that SPs and InPs must know at least 1 InPs can lead to an application poor performance.

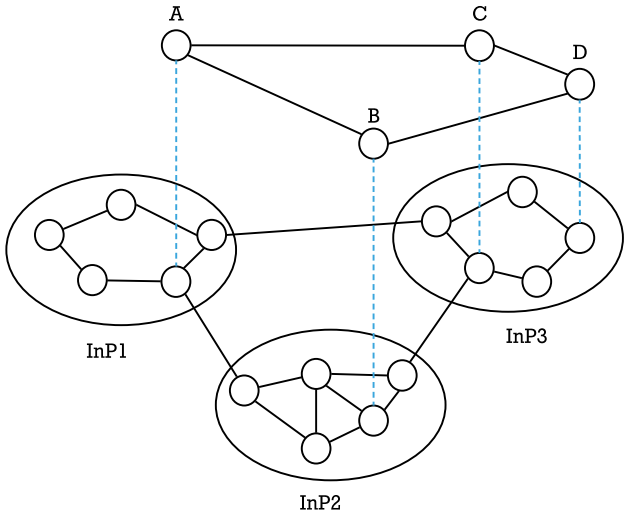

Figure 3.3: Multi-provider virtual network embedding [CSB10]

- Finally, in [ZXB10], an **automated service negotiation framework (V-Mart)** is implemented. The V-Mart is also decentralized and based on an auction mechanism, the Vickrey auction model. Though this approach ensures a fair market, thanks to the Vickrey auction model, it does not guarantee performance. The reason is that the VNE process deployed is a second-stage auction where explicitly all the InPs need to interact during the VN embedding. Thus, the VNE results in a high-demanding task. In addition, the sealed-bid is proposed to be done by a trusted 3rd party, which will cause that our application suffers again from centralization issues. Nevertheless, the main V-Mart shortcoming is that it does not solve the LID problem, since it does not enforce inter-domain policies, and the bids and the bidders are at the end known. This disclose once more information that InPs may prefer to keep as confidential.

3.3 Analysis of Related Work

Through relevant examples in supply chain management, as the presented in section 3.1.1, we can observe the extraordinary potential of blockchain in areas, where entities are willing to exchange data without trusting a third party. Despite the research does not include detailed specifications, such as the blockchain's type or mining strategy, a useful system design has been provided. In addition, some of the main benefits compared to centralized technologies are observed:

- **Disintermediation:** Thanks to the consensus mechanisms, blockchain runs without a central administrator. Thus, intermediaries are not required for realizing transactions which suppresses all the significant costs of involving middle-parties [MS15]. For instance, in the last example, users were able to transfer ownership, by signing the product's contract, without the need of a third party.
- **Autonomy:** After an owner deploys a smart contract on the BC, it is then maintained by all the participating users, and not only by its creator.
- **Data management and redundancy:** Blockchain helps us in coordinating, validating and storing the distributed data in a decentralized manner. It also provides redundancy since users own the same data, e.g. databases backup in centralized systems.
- **Unconstrained technology:** As actors can define and establish their own rules (smart contracts), new business models open up.
- **Privacy:** Thanks to cryptography, users can interact with untrusted partners. For example, in the above example, all actors can coordinate its services through the PLM process, which before would have been impossible as they do not trust each other.
- **Transparency and integrity:** All the changes that are made to the contract data are directly visible to the other users. In addition, once a transaction is executed, its immutable since it cannot be altered or deleted.

On the other hand, in the second part of this chapter, we have investigated related work on multi-provider VNE suffering from LID. Table 3.1 provides a comparison for each of the approaches presented before.

First of all, we can observe that decentralized systems provide obviously a better scalability because the application does not rely on a single point. In centralized systems, this central entity has to be trusted since it stores and manages all the information. In contrast, all the presented related work on decentralized VNE, uses a bidding mechanism to reach an agreement between the SPs and InPs.

Despite being a decentralized system, [ZXB10] uses to some extent a trusted third party (TTP). This is because, it is the only approach that offers sealed-bidding between users, although using a central entity or TTP. We also notice that in [EDPM13], [CSB10] the bids are exchanged between InPs to cooperate in the VN embedding, which results in undesired computational costs. In contrast, centralized systems [DRP15], perform the service negotiation without using auctions, since they trust a middle-party managing these operations.

Moreover, one of our main challenges in the multi-provider VNE scenario is to solve the limited information disclosure issues. We notice that [DRP15], [CSB10] deal with the problem, accessing public information not considered confidential and performing the VNE under its internal policies respectively. In addition, they provide location-assisted VNE, which facilitate the VN request partitioning across different InPs. The last, it also introduces a notification system, where the InPs gets notified upon VN request, which avoids time spending in request synchronization.

In general, centralized systems provide a better throughout in small scenarios since all the information is managed by a central entity. Hence, data sharing or user's cooperation is not needed before performing any operation. Conversely, decentralized systems benefit from large scenarios. Firstly, because the more users participating in an auction, the more competitive the market is. Secondly, all the activities are diversified, and thus, the decision-making is more efficient.

Finally, we classified costs into virtual network embedding and economic ones. Overall, we observe that VNE costs are lower when using a centralized system [DRP15], although when the parties are efficiently coordinated in a decentralized scenario, the costs are also extremely reduced [CSB10]. Concerning economic costs, they are obviously higher when a centralized system is used, as the application performance depends only on a central entity, which needs to be financed for their computational and maintaining costs.

Approach	System	Scalability	Trusted third party (TTP)	Sealed bidding	Bid exchange between InPs	LID problem solved	Location-assisted VNE	User notified	S-performance	L-performance	Low VNE cost	Low cost (€)
Multi-provider VNE framework with VNP [DRP15]	C	✗	✓	✗	✗	✓	✓	✗	✓	✗	✓	✗
CAD - Consensus-based auction for distributed slice embedding [EDPM13]	D	✓	✗	✗	✓	✗	✗	✗	✗	✓	✗	✓
PolyViNE - Policy-based inter-domain VN embedding framework [CSB10]	D	✓	✗	✗	✓	✓	✓	✓	✗	✓	✓	✓
V-mart - Automated service negotiation framework [ZXB10]	D	✓	*	✓	✗	✗	✗	✓	✗	✗	✗	✗

Table 3.1: Comparison of virtual network embedding approaches - Overview for system features implemented as *yes* (✓), *no* (✗), *to some extent* (*). Abbreviations: *D* = decentralized, *C* = centralized, *L-performance* = large scenario performance and *S-performance* = small scenario performance.

3.4 Summary

After comparing the different multi-provider VNE approaches exposed throughout this chapter, we find out that each of them introduces innovate features to deal with network virtualization. However, there is no single solution that satisfies at the same time the following requirements:

- Scalability.
- LID problem approached with data confidentiality.
- L-performance.
- Computational and economic low costs.

Therefore, the main focus on this thesis will be to investigate and later implement an approach which fulfills the above-mentioned virtual network embedding demands.

4 Design

Based on the challenges highlighted in the previous chapter, a conceptional design of a multi-provider virtual network embedding approach using blockchain will be now introduced. First, we define the requirements and assumptions used for the design process. Afterwards, we will present the fundamental goals expected from the system, followed by a general overview of the architecture along with its functionality. In addition, important features for our application will be discussed, such as the blockchain type, mining strategy, authentication and notification system, and the auction model used.

4.1 Requirements and Assumptions

Since blockchain is a disruptive technology, apparently it seems that it could improve many existing systems. However, is it suddenly blockchain the only way of approaching scalability, security or data handling between organizations? Or why not to use a traditional database rather than a blockchain? The truth of it is that there is not a single solution or better technology than the others, it just depends on user's requirements. Thus, it is important to understand in which scenarios blockchain could replace existing infrastructures. For this purpose, in Figure 4.1 we present a flowchart [WG17], which pretends to guide users in determining whether blockchain is the appropriate solution to resolve their problems, and if yes, its suitable blockchain type. We will also follow the diagram to decide if our defined VNE scenario demands match the blockchain use case.

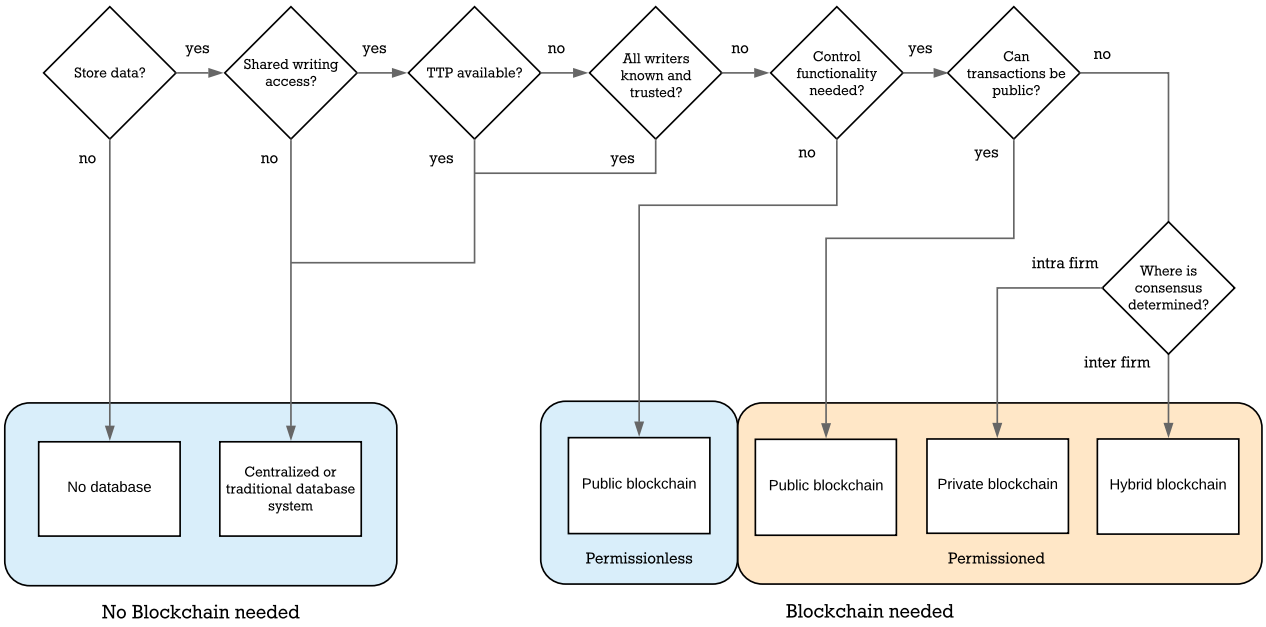

Figure 4.1: Flowchart to determine whether blockchain is the suitable solution. Adapted from [WG17]

Before starting with the flowchart, one statement needs to be clear: if the user does not need to store data, no blockchain or database is required. The discussion arises when multiple writers want to exchange data without relying on a trusted third party (TTP). In this context, a writer is a user that performs the four basic functions of persistent storage: create, read, update and delete data (CRUD). If they trust a TTP, a traditional or centralized database will always provide a better throughput. However, if this is not the case, then the user should first consider if all the writers are known and trusted. If so,

again a typical database (e.g. SQL) will be more adequate. If all the last requirements are satisfied, or in other words, multiple untrusted writers are willing to update the state of a system without trusting a middle actor, it is then that using a blockchain is a reasonable solution.

Before proceeding, in Figure 4.1 we first classify blockchains into two groups: *permissionless* and *permissioned* blockchains. To decide between these two approaches, the user should consider if he is willing to control his customized blockchain, creating his own specifications. For example, Bitcoin or Ethereum are permissionless blockchains, where users are added to the network and thus, restricted to its policies. In Table 4.1 the main characteristics of each blockchain type are summarized. Scalability is an important topic in the blockchain domain, where we distinguish: node scalability and performance scalability. The former is the capacity to add new participants in the blockchain without losing performance, and the latter is the number of transactions executed per second while suffering from latency. In addition, note that some features depend on the blockchain design decisions since permissioned blockchains capabilities are easily modifiable. For example, if consensus algorithms are used, the interaction between users should have a cost in order to ensure performance, i.e miners need to be rewarded with transaction fees.

Features	**Permissionless**	**Permissioned**
Type	Public	Public, private and hybrid blockchain
Control functionality/ Transaction processing	Fully decentralized	One or a small group of pre-selected entities
Access control	Open access	Authorized access
Number of users	High	Low
Number of untrusted users	High	Low
Writers	Any user participating	Permissioned group of people, e.g. bank customers
Centrally managed	No	(*)
Security	Consensus algorithms (e.g PoW)	(*)
Transaction fee	Yes	No (*)
Trusted	No	Yes
Node scalability	High	Low
Performance scalability	Low	High
Use cases	Cryptocurrencies (e.g. Bitcoin)	Transactions in business networks (Hyperledger Fabric)

Table 4.1: Types of blockchain: permissionless vs permissioned. Adapted from [WG17]. (*) states for depending on blockchain design decisions.

Furthermore, as stated in Figure 4.1, permissioned blockchains are then divided into:

- **Public permissioned blockchain:** Decentralized, all the participating users are allowed to read the system's state (transactions can be publicly verified), but just some of them have writing privileges.
- **Private blockchain:** Centralized, with access and writing permissions restricted to a set of users. Note that in the flowchart this corresponds to an intra-firm consensus since it is only managed by a single organization. We understand by firm a traditional company or a pre-selected group of users.
- **Hybrid/Consortium blockchain:** Partly decentralized, the blockchain management is controlled by a pre-established set of entities (inter-firm). In this model, each of the firms typically possesses a blockchain node to coordinate and ensure the system's functionality. Note that compared to the private blockchain, multiple firms maintain and serve the infrastructure rather than a single one. Thus, a consensus between these parties is also extremely needed. For example, currently, several banks deploy common blockchains to manage their shared data and reduce intermediaries costs. In this case, each bank will control a node, resulting in a better throughput in terms of scalability.

After presenting the blockchain use cases and types, we observe some similarities between a traditional shared database and a permissioned private chain, since they are both managed by a group of users, which simultaneously maintain the same confidential data. Despite private blockchains could be seen only as another term used to name shared databases, the truth of it is that each technology has its trade-offs, without mentioning scalability or cost issues. On one hand, centralized databases will always have better throughputs since, in addition to the basic transaction operations, blockchains need to perform cryptographic verifications, and ensure a consensus between the other participating nodes. On the other hand, private blockchains are less prone to malicious attacks, inasmuch as they spend a longer period in checking errors or validating transactions, than regular databases. Therefore, as already noted, if users trust each other and a high-performance is desired, using a blockchain will not be the suitable solution.

4.1.1 Virtual Network Embedding with LID scenario

In section 3.3, we have concluded that although excellent research in network virtualization have been conducted, to the best of our knowledge, there is still not an approach solving the LID problem in terms of scalability, confidentiality, and low-cost performance. Thus, after having spotted the main benefits of integrating blockchain in real-scenarios, among which disintermediation, privacy and transparency need to be highlighted, we assume that this technology could improve the VNE business process. Nevertheless, first, we should investigate whether our scenario could be integrated in a blockchain.

In network virtualization, we mainly encounter users with two different roles: Infrastructure Providers and Service Providers. Moreover, as the VNE is typically performed by the same parties across the countries, e.g. Deutsche Telekom, Vodafone or O2 in Germany, the scenario is restricted to a limited amount of users that barely changes. Therefore, we are facing an application where multiple known users but probably not trusted, want to reach a virtual network agreement, without relying on a third party. The decision of not including a TTP or VNP is mainly to avoid exchanging the InPs internal data to an intermediate actor.

Contemplating once again the flowchart in Figure 4.1 and observing the last mentioned requirements, we foresee that blockchain could be a suitable solution for our VNE scenario suffering from LID. However, the significant choice between blockchain types still needs to be argued. Obviously, the service negotiation between InPs and SPs is performed in a private environment, which automatically removes the permissionless and permissioned public blockchain options. Hence, the decision should be taken between a private or a hybrid blockchain.

One of the main requirements noticed from the previous related work is that our approach must not suffer from scalability issues. Accordingly, using a private blockchain, which is a centralized system managed by a single firm, will provoke that users rely again on a central point or trusted party, converting it to an unfeasible workaround. Thereby, the **partly decentralized hybrid blockchain** seems to be the most suitable solution. In this scenario, a firm corresponds to each interested InP or SP willing to possess and control a blockchain node, in order to ensure the network virtualization functionality.

4.1.2 Smart Contracts and Ethereum

After having spotted the most suitable blockchain type for our VNE environment, the non-trivial decision of selecting a blockchain needs to be taken. Prior to this, we need to consider whether our scenario could profit from the potential of smart contracts. Despite in subsection 2.1.2, it has been discussed what smart contracts are, it is also required to notice when they can be used. Thereupon, due to its extended dimension, a market study regarding smart contracts and its application fields will not be conducted. Nonetheless, we will enumerate the main usage examples of smart contracts in business to business processes [Tea]:

- **Non-confidential money transactions:** As blockchains are composed by multiple participating nodes, these can learn the values transferred. Though smart contracts allow restricting reading and writing functionalities, this information is still stored in the blockchain nodes, who can obtain it, e.g. accessing the disk or memory. Thus, trust on the blockchain nodes is also required.
- **Settlement agreement:** Since smart contracts can be personalized, it enables that participants reach an agreement without involving third-parties costs.
- **Frequently executed tasks:** Users regularly performing the same operations, can accelerate this process by depicting them on a smart contract code.
- **Self-operating databases:** Thanks to the blockchain, secured shared databases with low-costs are easily deployed. In addition, smart contracts can automate the interaction with this stored data, providing speed, real-time transparency and at the same time, opening new business models.

In the presented network virtualization example, as the users negotiate the lease of virtual nodes over and over to reach an agreement, this task could be perfectly reflected in a contract code. Apart from dealing with smart contracts, as a prototype for the VNE scenario will be implemented, the blockchain platform needs to fulfill the following requirements [MLTJ17]:

- **Easy to use and learn:** A low effort in setting the platform is essential. Furthemore the smart contracts should be written or derived from common languages in the world of programming.
- **Support and documentation:** A consolidated platform that is not in its early stages and has achieved a high-maturity level, is extremely needed. An active community of developers behind the project is also fundamental, e.g. with a considerable number of Github stars and forks.
- **Blockhain type:** Though it could be desirable that the main network is permissioned and hybrid or private, currently, in open source code platforms, the blockchain types can be modified.

Characteristics	Bitcoin	Ripple	Ethereum	Hyperledger Fabric	R3 Corda
Application	Payments	Payments	General purpose	General purpose	Financial services
Governance	Bitcoin developers	Ripple Labs	Ethereum developers	Linux Foundation	R3
Main network	Permissionless, public	Permissioned, public	Permissionless, public and private	Permissioned, private and hybrid	Permissioned, private and hybrid
Data model	UTXO-based	UTXO-based	Account-based	Account-based	UTXO-based
Configuration effort	Low	Low	Low	High	High
Support and Documentation	High	Medium	High	Medium	Low
Smart contract language	-	-	Solidity (JS), Serpent (Python) and LLL	Go, Java	Kotlin, Java
Currency	BTC	XRP	ETH	None	None

Table 4.2: Comparison of different blockchains platforms. UTXO stands for Unspent Transaction Outputs.

In table 4.2, a comparison of the most common blockchain platforms, at time of writing, is depicted. Since a scripting language with smart contracts is recommendable, Bitcoin and Ripple are not considered solutions. Thus, following the above requirements, Ethereum seems to be a mature project with a large number of developers behind, which facilitates the creation of new business models. In contrast, Hyperledger Fabric and R3 Corda are customized blockchains which need a longer configuration time, since its specifications need to be carefully discussed. FINISH TO EXPLAIN IT (programming-languages, build and use technology, private nw need to be done, EVM).

On balance, thanks to the blockchain, more precisely Ethereum and smart contracts, the untrustworthy VNP could be replaced with a secure, flexible and coordinated end-system. In this scenario, the reliance will now be distributed among multiple blockchain nodes rather than just on a costly single entity. However, challenges, such as the scalability or the service negotiation performance, will have to be thoroughly investigated along the thesis.

4.2 System Overview

4.2.1 Fundamental Goals

Enumerate system requirements: e.g: Flexibility, fair price negotiation, trustworthy... And explain each point

4.2.2 Architecture

A DApp has its backend code running on a decentralized peer-to-peer network. Contrast this with an app where the backend code is running on centralized servers. A DApp can have frontend code and user interfaces written in any language (just like an app) that can make calls to its backend. Furthermore, its frontend can be hosted on decentralized storage such as Swarm or IPFS.

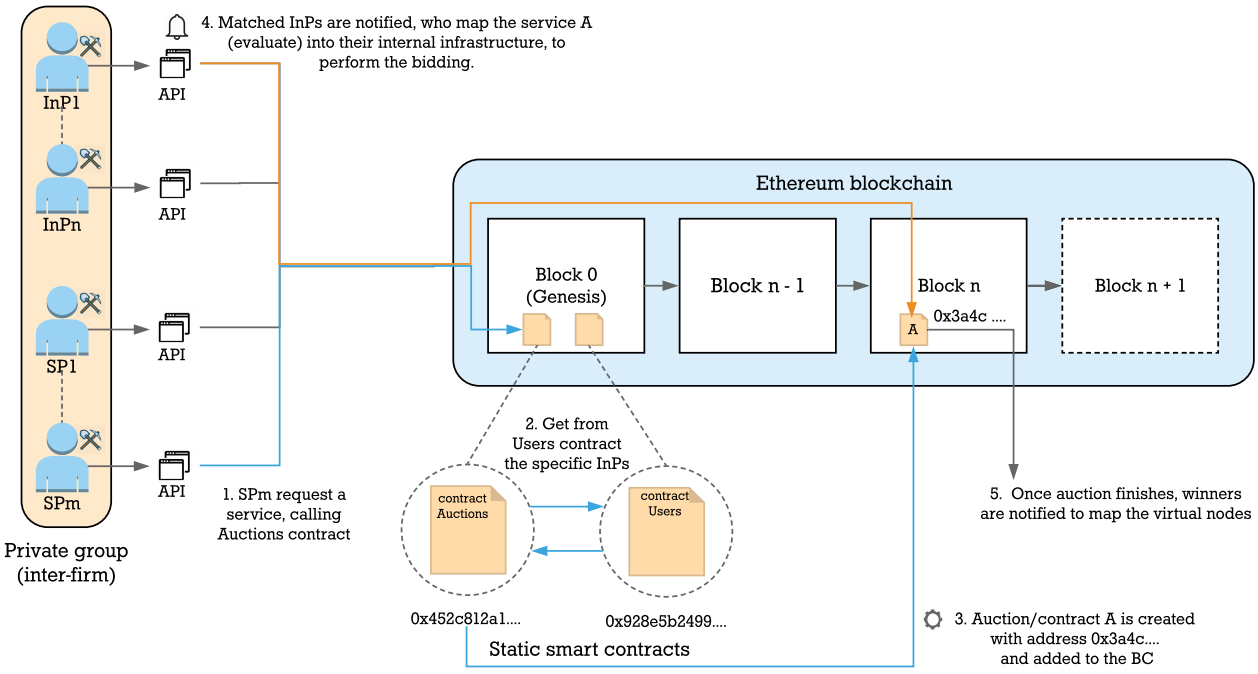

Figure 4.2: Design system architecture

Explain that scalability need to be tested. multiple ETH nodes, etc... which could be accomplished through a limited time auction. Hence, InPs disclosed information will be kept confidential (public/private key cryptography) in the blockchain and just accessible for its desired users.

4.2.3 Blockchain Types

Types of blockchain (permissionless and permissioned) economic reasons: cost, maintenance, computing power... TABLE COMPARING TYPES OF BC.

4.2.4 Mining

Mining from users or supernode? Start with supernodes (like in Bitcoin at the beginning), when more users maybe a user say he want to become a miner (gets the privileges and anonimity increased)

4.2.5 Authentication System

What can be externally extracted from a contract? Since value transfers cannot be blinded in Ethereum, anyone can see the value and therefore the highestBid. Sol: We can just see the address. That's a problem if there are just a few users. Do research on amount of users needed to solve the problem.

Invite People to the private BC. Users fixed amount of Ether (Proof of Stake). Garzik, J. 2015. "Public Versus Private Blockchains.

4.2.6 Notification System

4.2.7 Vickrey Auction Model

Study [AM+06] with equations and everything?

Assumption that intra-domain virtual links cost is almost 0. Having the virtual link cost from EC2 Amazon and the BW we could determine it...

However, the second-price concept needs to be thoroughly investigated as long as a bidder can submit multiple bids. Imagine an scenario, where a single bidder A submits two bids: {1 €,9 €} for a service with a real cost 5 €. In this case, this buyer will end up paying 1 for the service. Therefore, the second-price auction model needs always to be compared between bidders and not between bids. Suppose the last example, with a second bidder B submitting {5 €, 6 €}. At the end user A, will win the bid but paying 6 €, risking that if he would have been the only user bidding, he would have payed 9 €.

In this case, our auction procedure will start with the SP requesting a virtual network (a graph of virtual nodes) along with an upper bound, the last being the maximum amount of money the SP is willing to pay. Then, the InPs will estimate the request requirements (e.g: nodes availability or costs) in their infrastructure and thereafter submit their bids. In other words, the InPs will be sellers submitting bids and SPs the consumers requesting a service.

It is important to note that in our scenario, similar to [ZXB10], the **lowest auction** will win and not the highest. The reason is that at the end the SPs is who will pay the bid value. For instance, suppose that all the InPs make an offer close to the upper bound specified by the SP. If the highest bid wins, the SPs will always end-up paying approximately its upper bound (unfair). However, if the lowest bid wins, the system ensures that the InPs submits a bid proportional to the real cost since it will be the minimum amount of money that they could earn. Explain that an auction could use a multi-dimensional process (more parameters).

Package pricing or volume discount?

4.3 Summary

During this paper, the Ethereum blockchain will be used to perform a service negotiation (Vickrey auction) and later to track its possession throughout the supply chain process. This will be exemplified by a web application implementation, focused on network slicing. Thus, blockchain strengths and weaknesses will be further investigated.

The blockchain technology, in particular, secure smart contracts on the Ethereum platform, will be used to approach a real-world scenario. A decentralized application (third category) that enhances

supply chain performance, will be implemented. More precisely, network virtualization providers and customers will improve its communication, through the use of a web application (front-end) connected to the blockchain (back-end).

We do not claim that our approach is the best or the only way of performing VN embedding. On the contrary, it is more an add-on feature that can be merged with most of the existing solutions. In this scenario, the main goals are to remove the middleware or virtual network provider, by using a new prominent technology such as blockchain. Thus, a new decentralized, automated and secure system between the SPs and InPs will be created, which benefits from the blockchain potentials. Furthermore, to the best of our knowledge, this is the first approach using blockchain in the network virtualization context, and thanks to smart contract's flexibility, it can serve as a starting point for upcoming investigations.

Hint:

This chapter should describe the design of the own approach on a conceptional level without mentioning the implementation details. The section should have a length of about five pages.

5 Implementation

1. How a contract is compiled in to the Ethereum BC. 2. Technologies used (GETH vs ETH NW, web3, Solidity, truffle). (The Genesis block is the first block in a blockchain. It is always hardcoded in the blockchain). 3. Authentication 4. Component relations in the source code 5. Code examples

3.Investigate web3.eth.sign(web3.eth.accounts[0], address), the address should be unlocked. How this is possible? Smart contracts step by step code

```
contract Coin {

    // Declares a state variable that stores the balances for each possible address.
    mapping (address => uint) balances;

    // Constructor whose code is run only when the contract is created.
    function Coin() public {
    }
    // Function that sends 'amount' to the receiver address.
    function send(address receiver, uint amount) returns(bool isSent) {
        if (balances[msg.sender] < amount) return false;
        balances[msg.sender] -= amount;
        balances[receiver] += amount;
        return true;
    }
}
```

In the following code extracted from [1].

Hint:

This chapter should describe the details of the implementation addressing the following questions:

1. What are the design decisions made?
2. What is the environment the approach is developed in?
3. How are components mapped to classes of the source code?
4. How do the components interact with each other?
5. What are limitations of the implementation?

The section should have a length of about five pages.

5.1 Design Decisions

5.2 Architecture

[1] https://solidity.readthedocs.io/en/develop/introduction-to-smart-contracts.html

5.3 Interaction of Components

5.4 Summary

6 Evaluation

Hint:

This chapter should describe how the evaluation of the implemented mechanism was done.

1. Which evaluation method is used and why? Simulations, prototype?
2. What is the goal of the evaluation? Comparison? Proof of concept?
3. Wich metrics are used for characterizing the performance, costs, fairness, and efficiency of the system?
4. What are the parameter settings used in the evaluation and why? If possible always justify why a certain threshold has been chose for a particular parameter.
5. What is the outcome of the evaluation?

The section should have a length of about five to ten pages.

Test scenario with Real users using our application.

6.1 Goal and Methodology

6.2 Evaluation Setup

6.3 Evaluation Results

6.4 Analysis of Results

7 Conclusions

Hint:

This chapter should summarize the thesis and describe the main contributions of the thesis. Subsequently, it should describe possible future work in the context of the thesis. What are limitations of the developed solutions? Which things can be improved? The section should have a length of about three pages.

7.1 Summary

7.2 Contributions

7.3 Future Work

IOTA-TANGLE BigchainDB Hyperledger? The Front-end application could be hosted in a decentralized storage such as IPFS or SWARM. BC storing chunks of data

7.4 Final Remarks

Bibliography

[AM+06] Lawrence M Ausubel, Paul Milgrom, et al. The lovely but lonely vickrey auction. *Combinatorial auctions*, 17:22–26, 2006.

[AM16] Saveen A Abeyratne and Radmehr P Monfared. Blockchain ready manufacturing supply chain using distributed ledger. 2016.

[ama] Amazon ec2 instance types. `http://aws.amazon.com/ec2/instance-types`. Accessed: 2018-01-16.

[B+14] Vitalik Buterin et al. A next-generation smart contract and decentralized application platform. *white paper*, 2014.

[CJ09] Jorge Carapinha and Javier Jiménez. Network virtualization: a view from the bottom. In *Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures*, pages 73–80. ACM, 2009.

[Cla17] Jen Clarck. Blockchain Technology: the Next Generation of the Internet, March 2017. accessed:2017-11-27.

[CRB09] NM Mosharaf Kabir Chowdhury, Muntasir Raihan Rahman, and Raouf Boutaba. Virtual network embedding with coordinated node and link mapping. In *INFOCOM 2009, IEEE*, pages 783–791. IEEE, 2009.

[CSB10] Mosharaf Chowdhury, Fady Samuel, and Raouf Boutaba. Polyvine: policy-based virtual network embedding across multiple domains. In *Proceedings of the second ACM SIGCOMM workshop on Virtualized infrastructure systems and architectures*, pages 49–56. ACM, 2010.

[CST80] Vicki M Coppinger, Vernon L Smith, and Jon A Titus. Incentives and behavior in english, dutch and sealed-bid auctions. *Economic inquiry*, 18(1):1–22, 1980.

[DARP17] David Dietrich, Ahmed Abujoda, Amr Rizk, and Panagiotis Papadimitriou. Multi-provider service chain embedding with nestor. *IEEE Transactions on Network and Service Management*, 14(1):91–105, 2017.

[DRP15] David Dietrich, Amr Rizk, and Panagiotis Papadimitriou. Multi-provider virtual network embedding with limited information disclosure. *IEEE Transactions on Network and Service Management*, 12(2):188–201, 2015.

[EDPM13] Flavio Esposito, Donato Di Paola, and Ibrahim Matta. A general distributed approach to slice embedding with guarantees. In *IFIP Networking Conference, 2013*, pages 1–9. IEEE, 2013.

[FBB+13] Andreas Fischer, Juan Felipe Botero, Michael Till Beck, Hermann De Meer, and Xavier Hesselbach. Virtual network embedding: A survey. *IEEE Communications Surveys & Tutorials*, 15(4):1888–1906, 2013.

[Fia05] Petr Fiala. Information sharing in supply chains. *Omega*, 33(5):419–423, 2005.

[Hei02] Jussi Heikkilä. From supply to demand chain management: efficiency and customer satisfaction. *Journal of operations management*, 20(6):747–767, 2002.

[HLAZ11] Ines Houidi, Wajdi Louati, Walid Ben Ameur, and Djamal Zeghlache. Virtual network provisioning across multiple substrate networks. *Computer Networks*, 55(4):1011–1023, 2011.

[HLZ08] Ines Houidi, Wajdi Louati, and Djamal Zeghlache. A distributed virtual network mapping algorithm. In *Communications, 2008. ICC'08. IEEE International Conference on*, pages 5634–5640. IEEE, 2008.

[HS05] David Hausheer and Burkhard Stiller. Peermart: The technology for a distributed auction-based market for peer-to-peer services. In *Communications, 2005. ICC 2005. 2005 IEEE International Conference on*, volume 3, pages 1583–1587. IEEE, 2005.

[HZRS17] Ronny Hans, Hendrik Zuber, Amr Rizk, and Ralf Steinmetz. Blockchain and smart contracts: Disruptive technologies for the insurance market. 2017.

[KARF03] Mikko Kärkkäinen, Timo Ala-Risku, and Kary Främling. The product centric approach: a solution to supply network information management problems? *Computers in Industry*, 52(2):147–159, 2003.

[LB92] Hau L Lee and Corey Billington. Managing supply chain inventory: pitfalls and opportunities. *Sloan management review*, 33(3):65, 1992.

[Mer87] Ralph C Merkle. A digital signature based on a conventional encryption function. In *Conference on the Theory and Application of Cryptographic Techniques*, pages 369–378. Springer, 1987.

[MLTJ17] M Macdonald, Lisa Liu-Thorrold, and R Julien. The blockchain: A comparison of platforms and their uses beyond bitcoin. *Working Paper*, (February), 2017.

[MS15] Michael Mainelli and Mike Smith. Sharing ledgers for sharing economies: an exploration of mutual distributed ledgers (aka blockchain technology). 2015.

[Nak08] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

[OV02] Elth Ogston and Stamatis Vassiliadis. A peer-to-peer agent auction. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*, pages 151–159. ACM, 2002.

[Ros17] Ameer Rosic. What is ethereum casper protocol? crash course. `https://blockgeeks.com/guides/ethereum-casper/`, November 2017. Accessed: 2018-02-09.

[Shi14] Ken Shirriff. Bitcoins the hard way: Using the raw bitcoin protocol, February 2014. accessed:2018-02-01.

[Sta15] John Stark. Product lifecycle management. In *Product Lifecycle Management (Volume 1)*, pages 1–29. Springer, 2015.

[Swa15] Melanie Swan. *Blockchain: Blueprint for a new economy*. " O'Reilly Media, Inc.", 2015.

[Tea] Roster Team. Blockchain at berkeley. `https://blockchain.berkeley.edu`. Accessed: 2018-02-18.

[TT16] Don Tapscott and Alex Tapscott. *Blockchain Revolution: How the technology behind Bitcoin is changing money, business, and the world*. Penguin, 2016.

[Tut02] A Tuttle. Who do you trust. *Industrial Distribution*, 91(3):17, 2002.

[Vic61] William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of finance*, 16(1):8–37, 1961.

[WG17] Karl Wüst and Arthur Gervais. Do you need a blockchain? *IACR Cryptology ePrint Archive*, 2017:375, 2017.

[ZXB10] Fida-E Zaheer, Jin Xiao, and Raouf Boutaba. Multi-provider service negotiation and contracting in network virtualization. In *Network operations and management symposium (NOMS), 2010 IEEE*, pages 471–478. IEEE, 2010.

[ZZSRR08] Yaping Zhu, Rui Zhang-Shen, Sampath Rangarajan, and Jennifer Rexford. Cabernet: Connectivity architecture for better network services. In *Proceedings of the 2008 ACM CoNEXT Conference*, page 64. ACM, 2008.