
Tenancy over Distributed Workflows using Blockchain Technology exemplified by Network Slicing

Master-Arbeit

Jordi Bisbal Ansaldo

KOM-M-634



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-634

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 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 Design	27
4.2.1	Fundamental Goals	27
4.2.2	System Architecture	28
4.2.3	Blockchain Type and Mining Strategy	30
4.2.4	Authentication System	30
4.2.5	The Vickrey Auction Model	32
4.3	Summary	36
5	Implementation	37
5.1	Design Decisions	37
5.2	Architecture	37
5.3	Interaction of Components	37
5.4	Summary	38
6	Evaluation	39
6.1	Goal and Methodology	39
6.2	Evaluation Setup	39
6.3	Evaluation Results	39
6.4	Analysis of Results	39

7	Conclusions	41
7.1	Summary	41
7.2	Contributions	41
7.3	Future Work	41
7.4	Final Remarks	41
	Bibliography	41

Abstract

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: Ethereum¹. 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, Everledger², 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 several blockchain applications in real scenarios, we observe many improvements, when users are willing to share and alter data in a distributed, secure and transparent manner. In this paper, we will focus on enterprises or organizations, suffering from supply chain challenges.

¹ <https://www.ethereum.org/>

² <https://www.everledger.ioafa>

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 handle a huge amount of information. This leads companies to suffer from considerable issues, such as scalability, data security or communication. Currently most of the 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 could 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 distributing these privileges among multiple users, which cooperate for handling such complex tasks? As a result, a direct customer-provider relation, avoiding the presence of any intermediaries, will be maintained.

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 co-ordinated 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, in the 2nd generation, the so-called smart contracts, which are driven by the Ethereum platform, will be investigated. This 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 customers and providers is crucial. Due to this importance, a well-known public negotiation mechanism will also be presented, that is, auctions.

2.1 Blockchain: A decentralized and distributed 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 centralized (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 scenario where a user wants to transfer money to another one, see Figure 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, in 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 centralized 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.

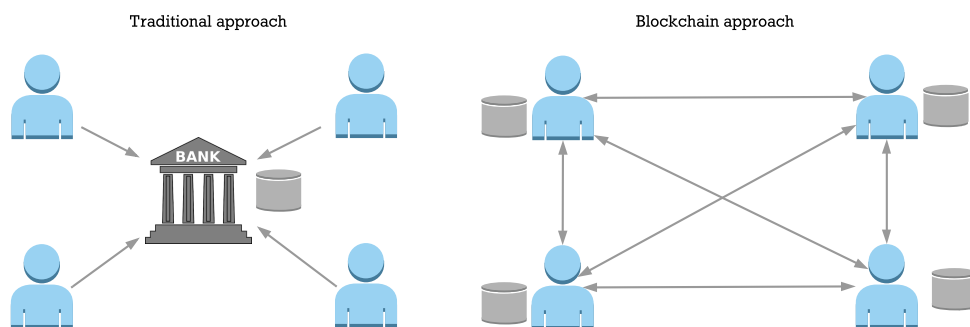


Figure 2.1: Traditional database vs blockchain approach

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. Therefore, before continuing, one statement needs to be clear: **Bitcoin is a cryptocurrency that uses the blockchain technology and cryptocurrencies are just one of the multiple blockchain applications.**

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 the end of January 2018, Coinmarketcap¹, 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 Internet², a Japanese company, will start paying parts of employees salaries in cryptocurrencies (Bitcoin).

In this paper, we will set aside whether cryptocurrencies can become true currencies or not, and also the political, social or economic impact. The only focus will be on 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 Nakamoto'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 built in the top of another network, in this case above the Internet application layer, which is controlled by its users.

From another point of view, cryptocurrencies ledgers can be interpreted as state transition systems, where there is an initial state that after a transition function (money transfer), results in a new state. In Bitcoin, this initial state is a list of mined but still not spent transactions, each containing also the address supplying BTCs. This transaction output will be the receiving address along with the amount received. For this reason, Bitcoin is considered an unspent transaction output (UTXO) data model. Imagine an scenario where user A wants to send 10 BTC to user B. The first state is A and B current balance, e.g. $\{A = 10, B = 20\}$, and the transition function will take 10 BTC from A and insert it to B's account, generating a new state, now $\{A = 0, B = 30\}$. 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 its chains 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 on average 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 the previous 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

¹ <https://coinmarketcap.com/>

² <https://www.gmo.jp/en/>

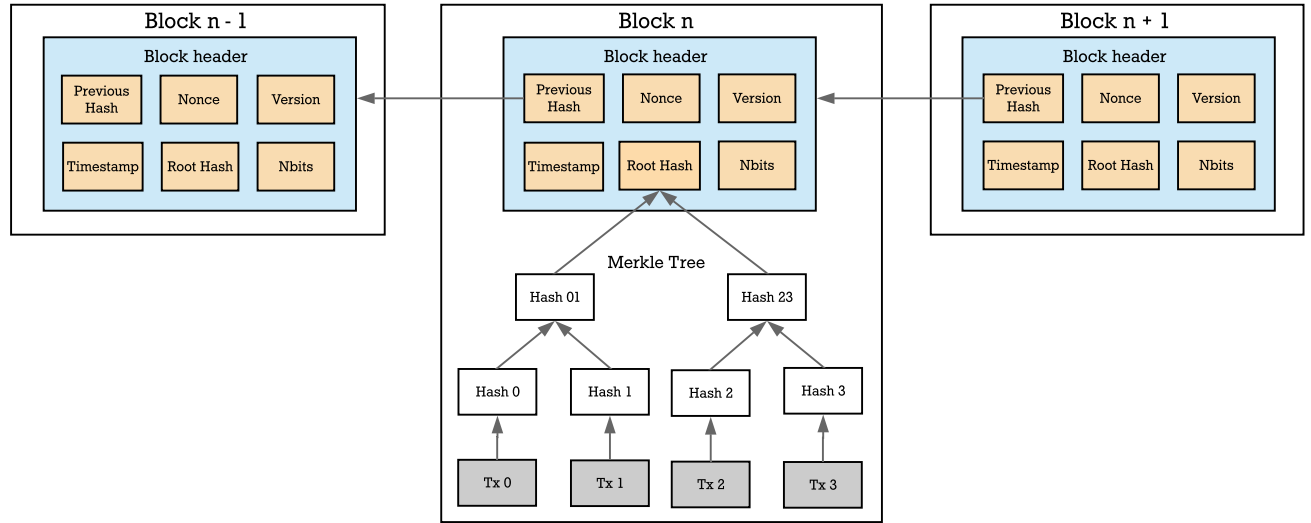


Figure 2.2: Bitcoin blockchain structure. Adapted from [HZRS17]

simple and requires computational power, if they are the first block solvers, they are also economically rewarded.

Apart from a list of transactions, each Bitcoin block contains a **block header**, whose hash is stored in the next block in order to maintain a consensus. Figure 2.2 shows how transactions are bundled into blocks, where each block header contains:

- A timestamp, to identify when the event occurs.
- A nonce, which is an arbitrary number used for the miner to create the block header.
- Previous block header hash, to keep track of already added blocks.
- A block version number that indicates the validation rules used.
- A Nbits field, whose value should be higher than the block's header hash (later explained).
- 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 Merkle tree, which is a binary tree hash [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, see (2.1), 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 a nonce (x), whose cryptographic hash (e.g *SHA256*) is less than a *target* value (non-encoded version of Nbits). 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).

$$F_d(\text{blockheader}) = F_d(c, x) = \text{SHA256}(\text{SHA256}(c|x)) < \frac{2^{224}}{d} \quad (2.1)$$

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 (ECDSA³). 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.

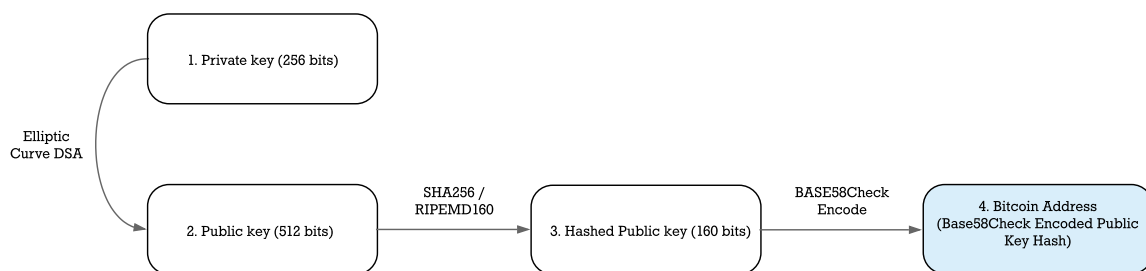


Figure 2.3: Bitcoin keys and addresses generation [Shi14]

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. Transaction examples can be found in Block Explorer⁴.

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 applications. In 2014, Gavin Wood published the Ethereum yellow paper introducing the Ethereum Virtual Machine (EVM), i.e. a blockchain with a built-in programming language. Ethereum first started as a crowdfunding project, which collects small amounts of money from a large number of users. Currently, Ethereum is the second most valuable cryptocurrency.

³ <https://www.maximintegrated.com/en/app-notes/index.mvp/id/5767>

⁴ <https://blockexplorer.com/>

Furthermore, it extends the power of decentralized transactions with a Turing-complete contract system. A Turing-complete system can be proven mathematically to be capable of performing any computation, which allows developers to create its own programs. The code of these functions are written into the so-called **smart contracts** and executed by the Ethereum nodes, each using its Ethereum Virtual Machine (EVM).

In the end, smart contracts are basically Ethereum scripts that whenever they are called, the function programmed inside of it is executed. A simple example of a contract is the logic inside a vending 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 the Coke. This function, in the smart contract context can be seen as:

```
1 contract Products {
2
3     function getItem(bytes32 buttonPressed, uint amount) returns(bytes32 item) {
4         if (buttonPressed == 'coke' && amount == 2) return coke;
5         return null;
6     }
7 }
```

Therefore, smart contracts allow anyone to write their own functions, in just a few lines of code. Despite the fact that Bitcoin and Ethereum have similar features, such as decentralization or being 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 differences, there are other inequalities that need to be highlighted:

- Ethereum's block time is shorter, specifically it is around 14s. Remember that Bitcoin's mining time was around 10 minutes. Hence, as a consequence, Ethereum needs to handle large amounts of transactions per seconds, which means that the network can easily get separated in different subchains. To solve these scalability issues, *Casper*, a protocol from the new *Proof-of-Stake* (PoS) mechanism has been recently proposed [Ros17].
- Thus, Ethereum pretends to move from the *Proof-of-Work* to the *Proof-of-Stake* consensus algorithm. In contrast to PoW, where the miners that first solve a mathematical problem are rewarded, in PoS, validators propose and vote the next block. The voting power is directly proportional to the validator's stake (deposit). To become validators, users need to lock up their ether into a deposit. Therefore, in PoS there are validators rather than miners.
- Bitcoin's blocks and transaction sizes are indicated in bytes, however, in Ethereum, this depends on the contract complexity. This complexity is expressed in terms of *gas*, which is the amount of *ETH*⁵, used for executing contracts in the Ethereum blockchain.
- 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.

Also, a miner does not receive a block reward, but instead he earns all gas used for executing the smart contract. 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 identify users while 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

⁵ *ETH* or *Ether* is the digital currency used by Ethereum

- **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.

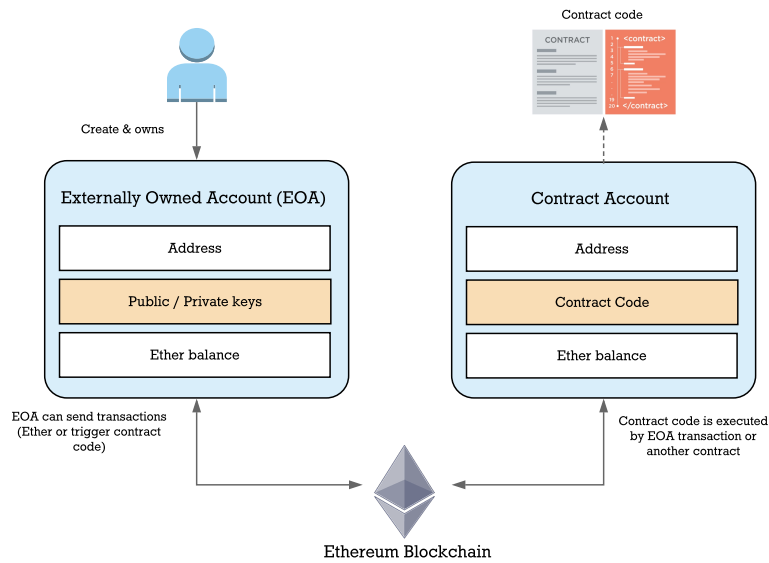


Figure 2.4: Ethereum accounts: Externally owned accounts and contract accounts

However, what happens if this contract code results in an infinite loop? Ethereum solves this using 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 F is given by:

$$F = P \times L + V - R \quad (2.2)$$

Firstly, the gas limit L is the maximum amount of gas to spend on a transaction, which is by default 21000. Secondly, on Ethereum the gas price P is measured in units of Gwei, which is equivalent to $1e^{-9}$ ETH. Thirdly, the value V is the amount of ether transferred to the other EOA. Fortunately, all unused gas R during a transaction is refunded to the sender. Nevertheless, if a transaction returns an error, e.g: gas provided not enough to execute a contract, this provided ether will never be refunded.

In conclusion, a smart contract 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] decentralized applications DAPPs are split into three types. The first group includes financial applications, where users only want to manage money transactions, e.g. a customer paying the provider through a blockchain application. The second group is formed by semi-financial applications, in which money is involved but it is mixed with other requirements. For instance, an application where users receive incentives when solutions to computational problems are provided. 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 DAO⁶, 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.

2.2 Network Virtualization

In the last few years, network virtualization has started to grow in popularity since it enables simulating baremetal networks. These virtual networks, in contrast to physical networks, 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).

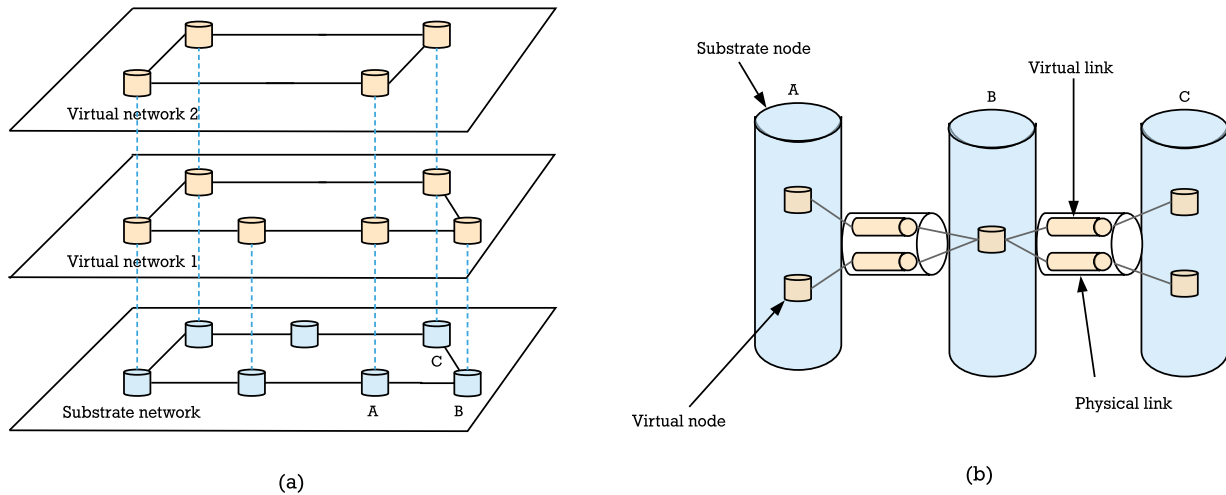


Figure 2.5: Network virtualization model and its basic elements [CJ09]

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. This is done without the need to purchase physical network equipment.
- **Virtual network providers (VNPs):** Are brokers, which can optionally serve as intermediaries between the infrastructure providers and service providers.

The network slicing process with the presence of a broker typically starts with the VNP obtaining the necessary network information from the SP. To optimize the software costs, the SP's virtual network request will be divided into groups of nodes (subgraphs). Afterwards, the modified network is transmitted

⁶ <https://ethereum.org/dao>

to the multiple InPs, who are in charge of embedding the virtual nodes from the request in their physical resources.

The presented process is coordinated by an actor, the VNP, who stores all the network information to successfully allocate the incoming resources. However, there are other approaches, where the InPs do not want to rely on a central entity, since its confidential information is being disclosed and hence, known by the VNP. Conversely, the SPs and InPs make use of consensus mechanisms to perform the virtual resources trading. Thus, the network virtualization task is accomplished in a more decentralized manner, since no middle-party exists. 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 the buyer's goal is to find the desired service at the lowest price, the providers or seller's goal is to sell the service at the highest "possible" price. Thus, an auction should 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 can be (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 network cost models. 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** has the particularity of corresponding to a *sealed-bid second-price* auction [Vic61]. 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, imagine a bidding for product A, where the winner has bidden $\{A = 15\}$ and the second highest bid is $\{A = 10\}$. Then, this winner ends-up paying only $\{A = 10\}$. In addition, it is shown that a Vickrey auction is strategy-proof, meaning that each bidder quotes the true cost of the service, and if they do so, the resulting total values are maximised [Vic61]. From the bidder's point of view this means that he can safely bid the real 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 reviewing the preceding analysis regarding auction types, a ***one-sided reverse Vickrey*** auction seems to be the most suitable for brokerless virtual network embedding since it provides confidentiality and the biggest aggregate profit among sellers and buyers. This will be later investigated in chapter 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 started with Bitcoin, the blockchain technology can be used as a software connector in multiple scenarios. Due to smart contracts, arbitrary applications could be implemented in the context of Ethereum blockchains, enabling developers using the blockchain concept in their own environment. In the next chapters, we will investigate a multi-provider virtual network embedding scenario where infrastructure providers are not willing to disclose internal network cost models.



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 point of view, and secondly, we will analyze the pros and cons of the existing VNE related work, in order to observe the 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 represented a major challenge for many 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 in the following way: (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, a lot 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 Skuchain¹, 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), involves different stages [Sta15], in which multiple parties modify the product's data, see Figure 3.1.a. 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

¹ <http://www.skuchain.com/>

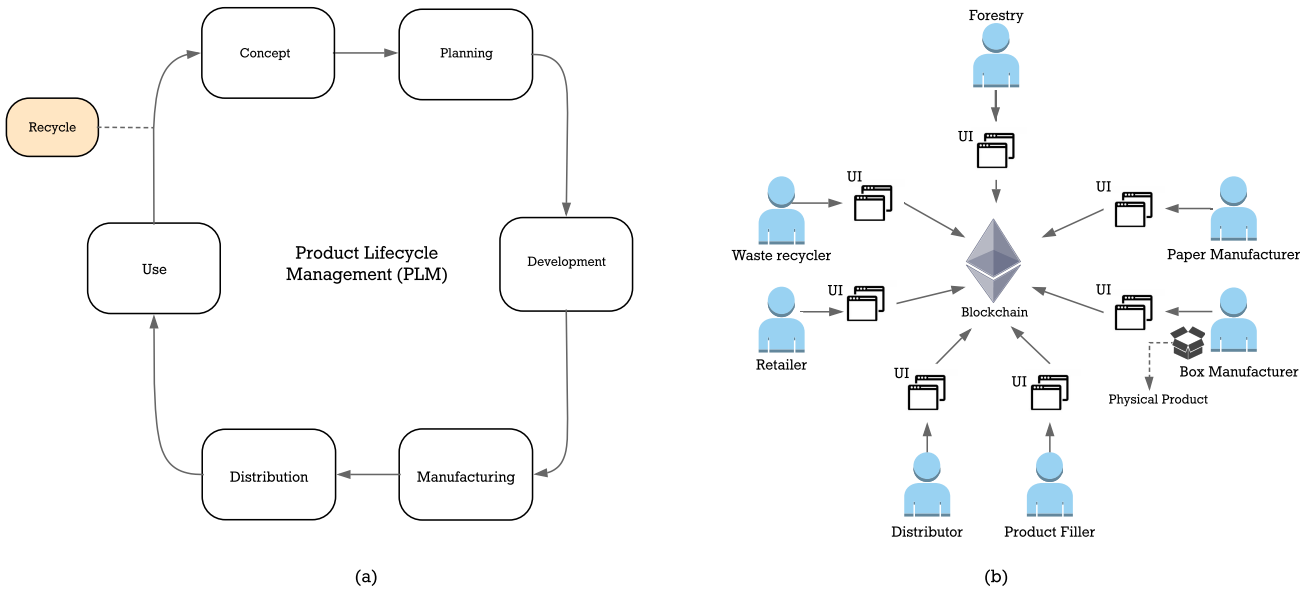


Figure 3.1: PLM stages and cardbox blockchain application example [Sta15]

can access or modify the information. To access or alter this data, parties must authenticate themselves signing the requests with their private keys.

During the lifecycle management, a product is owned by an entity, for instance the 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 ensures that in each phase, only the granted user modifies the contract.

At the end of [AM16], an example to better explain the mentioned approach is also presented. The application consists of the product lifecycle of a cardboard box, starting from the trees cut down until the box is recycled. 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 is allowed to alter it through the user interface (e.g. modify the box quality). Once finished, he will transfer the package (signed) to the next actor, in particular, the product filler, who will follow the same steps.

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 thesis, 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 [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. 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 VNE², is decomposed 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 software cost of the virtual network setup.
- **Inter-connection between subgraphs:** Establish paths between the previously created subgraphs.
- **VN segment mapping:** Embed the virtual resources (subgraphs) to the InPs physical nodes.

Furthermore, in such scenarios exists two types of communication [ZXB10]:

- **Horizontal communication** is the one established between infrastructure providers to guarantee the most efficient end-services. These relations can be: *public relations* established using a market mechanism (e.g: an auction), or *private relations* that are previously arranged, because the InPs already know each other. In both cases, the relation arises from the need to negotiate and cooperate to serve the SPs.
- **Vertical communication**, which emerges from the negotiation between SPs and InPs, where the former is willing to lease some virtual resources from the latter. This communication is typically facilitated by a third party (VNP).

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 optimize the network software costs.

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 [ama18], which announces the attributes (CPU, memory, cost) of different instances types.
- **Substrate network topology:** Most of network's 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 the [DRP15] implemented framework, the slice embedding is achieved by subsequently performing 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 thesis).

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

² Inter-domain is between multiple InPs and intra-domain is inside a single InP.

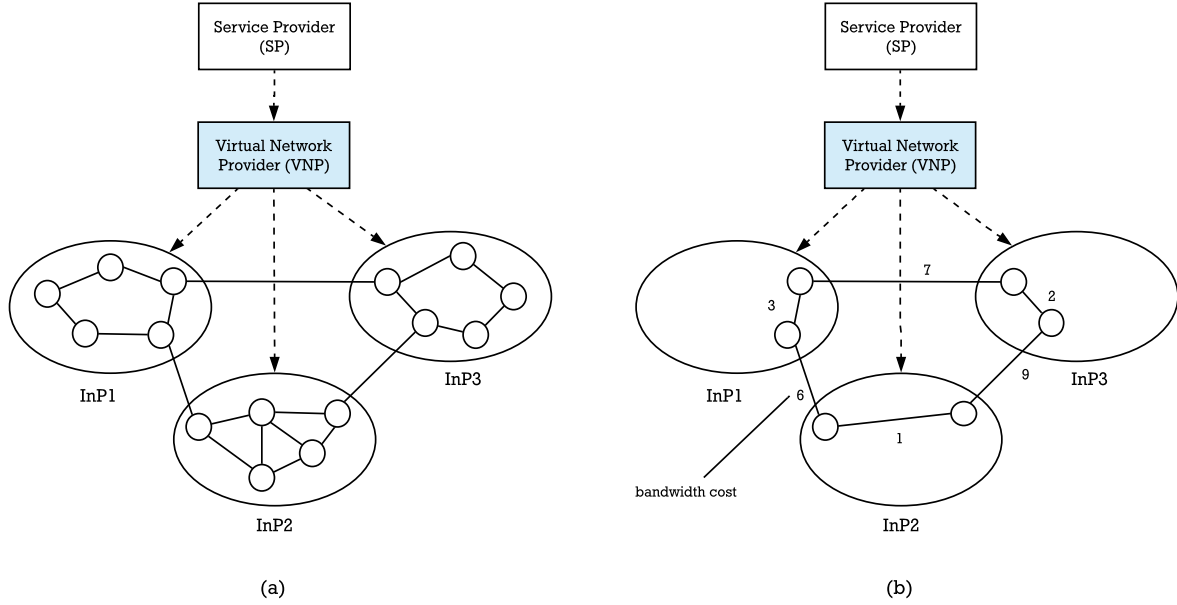


Figure 3.2: (a) Virtual network request across multiple InPs using a VN Provider. (b) VNP's view on the substrate network topologies. [DRP15]

the number of slices requested and later allocated. Despite the positive findings, this scenario suffers mainly from scalability, since everything relies on a single centralized authority: the virtual network provider. In addition, it assumes that InPs 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 the 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, forwarded decisions are location-based.

For example, in Figure 3.3 each virtual node $\{A, B, C, D\}$, has a preferred 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 needs to 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 need to know at least 1 InPs can lead to an application poor performance.

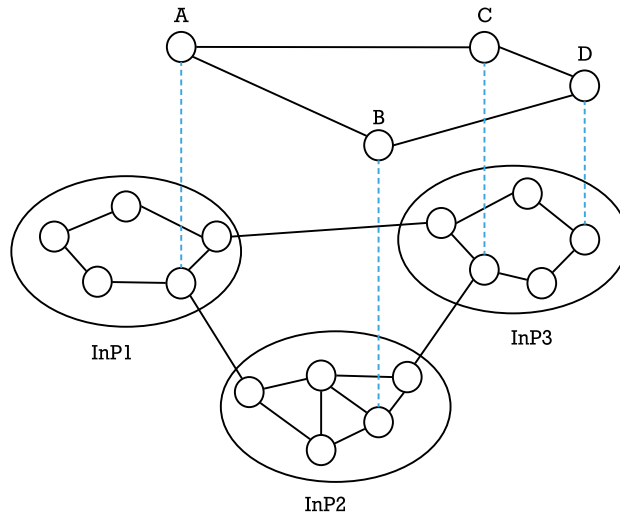


Figure 3.3: Multi-provider virtual network embedding [CSB10]

- 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 could cause again confidence issues. Nevertheless, the main V-Mart shortcoming is that it does not solve the LID problem, since it does not enforce inter-domain policies.

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 blockchain's potential mainly 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 contemplated:

- **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 presented example, users were able to transfer ownership, just signing the product's contract.
- **Data management and redundancy:** Blockchain helps us in coordinating, validating and storing the distributed data in a decentralized manner. It also provides automatically redundancy since users own the same data. Reversely, in centralized systems, the users need to explicitly create a backup of the databases, for preventing the loss of data if the system crashes.
- **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 already mentioned example, all actors were able to coordinate its services through the PLM process, which before would have been a complex task 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. A comparison for each of the approaches presented before is shown in Table 3.1.

First of all, we can observe that decentralized systems provide obviously a better scalability because the application does not rely on a single point of failure. 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 offering a sealed-bidding between users, although with the use of a trusted third party. 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 the main challenges 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 throughput 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. 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 innovative 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.
- Large scenarios performance.
- Computational and economic low costs.

Therefore, the main focus of this thesis will be to investigate and later implement an approach that gets rid of the VNP, and which fulfills the above-mentioned virtual network embedding demands. Here, we foresee that blockchain could play a significant role.



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, in particular, the blockchain type, the mining strategy, the authentication 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 the blockchain the only way of approaching scalability, security or data handling between organizations? Or why not use a traditional database rather than a blockchain? The truth is that there is not a single solution or better technology than the others, as it depends on user's requirements. Thus, it is important to understand in which scenarios the 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, the suitable BC type. We will also follow the diagram to decide if our defined VNE scenario demands match the blockchain use case.

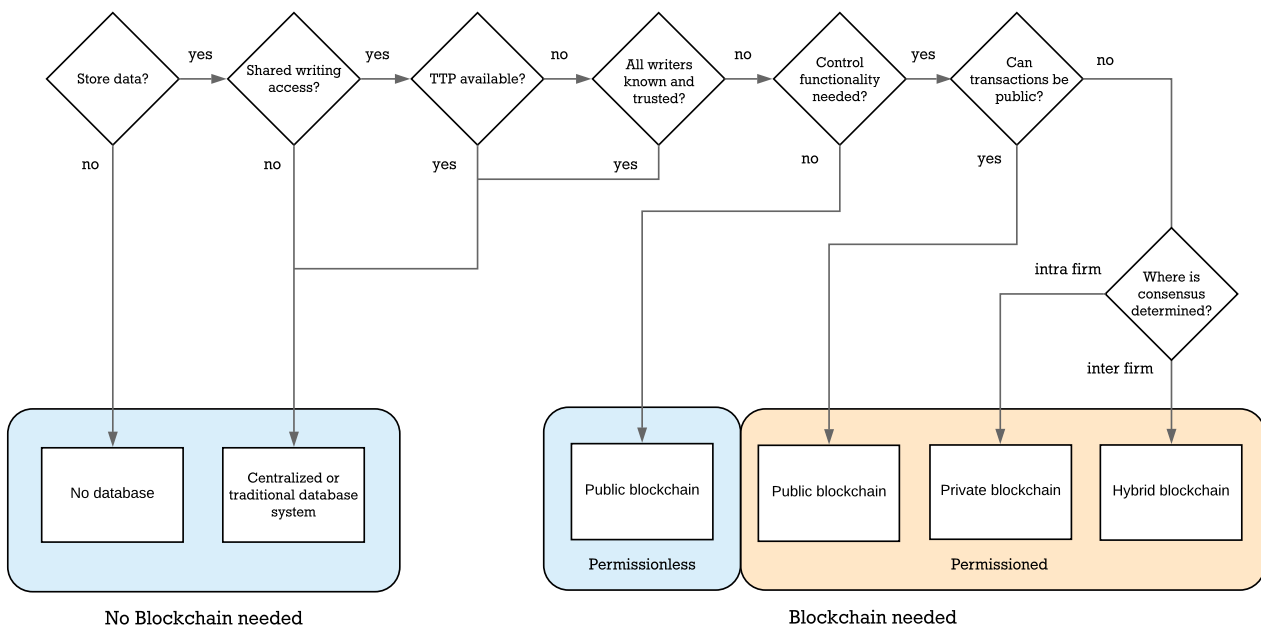


Figure 4.1: Flowchart to determine whether blockchain is the suitable solution. Adapted from [WG17]

Before starting with the flowchart, one statement needs to be clarified: 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 better positioned. 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.

In Figure 4.1, we also 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, the main networks of Bitcoin and Ethereum are permissionless, 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. 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 [bit09])	Transactions in business networks (Hyperledger Fabric [hyp17])

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

Furthermore, as stated in Figure 4.1, permissioned blockchains are 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 (Figure 4.1) 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 is controlled by a pre-established set of entities (inter-firm). In this model, each of these firms coordinates and ensures 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 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. The blockchain also organizes data into blocks, which is updated only appending new data. 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 some research in network virtualization have been conducted, to the best of our knowledge, there is still no approach under LID that gets rid of the broker (VNP), providing 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 network cost models to an intermediate actor.

Observing the last mentioned requirements, we foresee that blockchain could be a suitable solution for our VNE scenario suffering from LID. However, an appropriate blockchain type still needs to be chosen. 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 should address 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 platform 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 [Tea17]:

- **Settlement agreement:** Since smart contracts can be personalized, it enables that participants can reach any kind of 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 later we will implement a prototype, the blockchain platform chosen needs to fulfill the following prerequisites [MLTJ17]:

- **Easy to build, use and learn:** A low effort in setting the platform is essential. Thus, the smart contracts should be written or derived from common languages in the world of programming, such as JavaScript or Python.
- **Support and documentation:** A consolidated platform, which is not in its early stages and with a high-maturity level achieved, 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.
- **Blockchain 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. Thus, that the predefined main network type matches the desired one, is not a must-have feature.

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, in our opinion, is depicted. Since in our application a scripting language with smart contracts is preferable, and Bitcoin and Ripple by default do not incorporate them, these platforms are automatically discarded. On the other hand, Hyperledger Fabric and R3 Corda are popular used platforms, whose main network perfectly fits our requirements. However, they provide customized blockchains with a high configuration effort, since its specifications need to be carefully designed. Thus, as a platform with a straightforward setup and usage is required, Ethereum seems to be the better positioned.

Ethereum is a mature and easy to build project based on smart contracts with a large number of developers behind, who are actively discussing and improving its functionality. Thanks to this, the

creation of new business models is extremely facilitated for newcomers. Furthermore, Ethereum smart contracts are written in Solidity and Serpent, which are similar to JavaScript and Python respectively. Despite the fact that Ethereum's main network (public permissionless), is not the ideal for our scenario (permissioned and hybrid), the simple and easy set up of the platform's environment, will allow us to quickly deal with these issues. Finally, the Ethereum Virtual Machine (EVM) is a Turing-complete system, which helps us in obtaining the maximum benefit from the smart contract's potential.

Overall, thanks to the blockchain, more precisely Ethereum and smart contracts, the VNP could be replaced with a secure, flexible and coordinated 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 Design

Based on the requirements and assumptions defined in the previous section, the design of a multi-provider virtual network embedding system using the blockchain technology is presented in the following. At first, the fundamental goals that are expected from the system are defined, followed by a general overview of the architecture and later on, the essential design features are discussed.

4.2.1 Fundamental Goals

In general, the fundamental goals for network virtualization relies on simulating baremetal networks, providing improvements in terms of flexibility, manageability, and costs. In this scenario, the SPs are willing to embed virtual nodes across different InPs to create customized virtual networks. This embedding task consists first of the VN request partitioning and later, of the corresponding mapping to the InP's physical nodes, such that virtual node and link requirements are fulfilled. However, the problem appears when InPs are not willing to disclose detailed information about their resource availability or network topologies, which is indispensable for the VN partitioning efficiency [DRP15]. In this thesis, the VN partitioning is treated as a cost minimization problem, where the service provider pursues the minimum VN embedding cost¹. In a real world, there are more parameters that could be taken into consideration, such as quality of the service or the proportioned capacity.

In a multi-provider scenario with full information disclosure (Figure 4.2.a), since the infrastructure provider's internal information is publicly available, the VN request partitioning is efficiently computed. Conversely, the lack of data in a limited information disclosure scenario (Figure 4.2.b), directly affects the VNE throughput. Nevertheless, there is some information not considered confidential by the InPs, such as the location or the virtual node types offered by the peering nodes. For instance, in Figure 4.2.b, the InP1 has two peering nodes with locations in Germany and Switzerland, and virtual node types $\{A, B, D, E\}$ and $\{F, G\}$ respectively. These virtual node types, as Amazon EC2 [ama18] does, are used to classify the resources in different groups, each having common attributes, such as CPU, memory, storage and network capacity. Thus, in this thesis, new embedding strategies and approaches to facilitate the service negotiation process between the InPs and SPs, will be developed.

Moreover, in section 3.3, it has been contemplated that decentralized approaches tend to use auctions to negotiate the network workflows. Regardless of the auction model applied for our VNE framework, there are several objective metrics that need be considered:

- **Efficient:** The virtual network embedding must be efficient in terms of synchronization, communication, and data sharing between the participating users.

¹ From now on, we use the term cost to refer the value that the SP must pay for embedding the virtual nodes across the InPs. This value is derived from the InP bids

- **Equality:** All participants, namely service providers and infrastructure providers, must have the same role's privileges. In other words, all SPs must be able to request the same amount of virtual networks over time and InPs bids must be treated equally in the application.
- **Market fairness:** The auction model requires that the SPs find the desired service at the lowest price and that the InPs obtain the highest possible revenue for the offered service.
- **Data confidentiality:** At the moment of bidding, the bids should be kept in secrecy in order to not alter the auction progress. In the end, the winners can be publicly known.
- **No trusted auctioneer:** Since confidentiality is required, a middle party or auctioneer should not be capable of managing the auction workflow and thus, accessing the private bidding values.
- **Automated:** The virtual network embedding process needs to be automated with a friendly and easy-to-use framework.

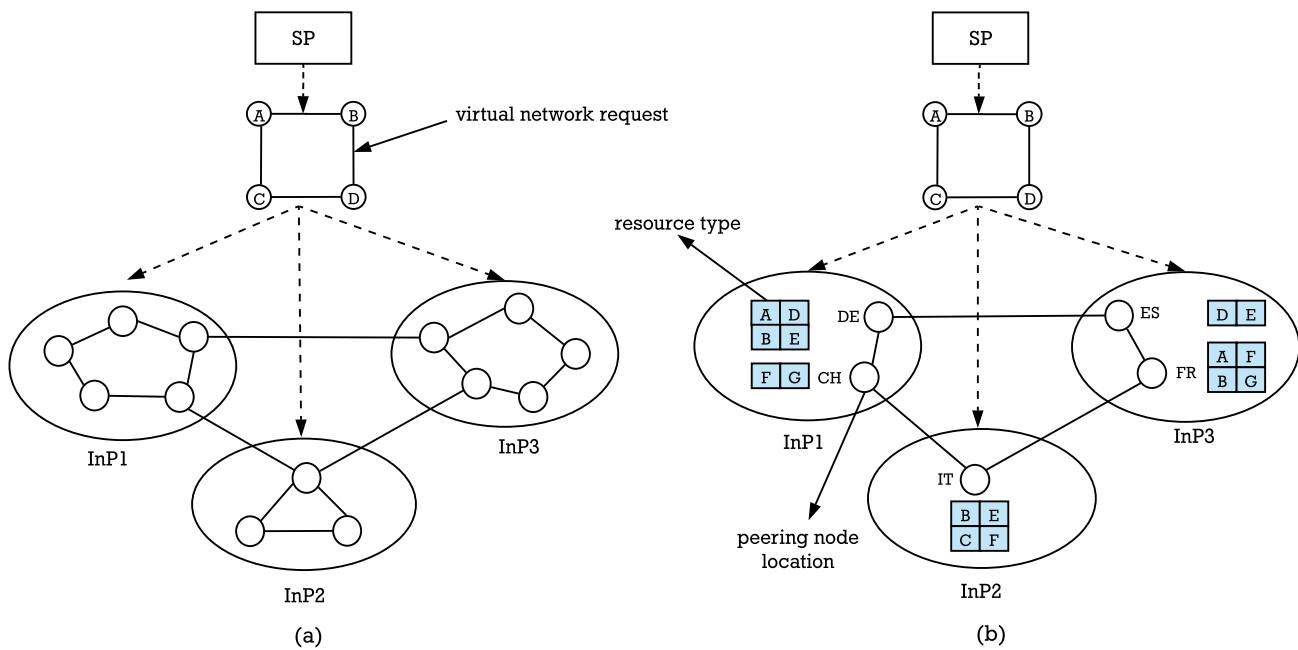


Figure 4.2: Substrate network visibility. (a) Full information disclosure (FID). (b) Limited Information Disclosure (LID) with virtual node types and the corresponding prices. Adapted from [DRP15].

4.2.2 System Architecture

After defining the main VNE problem that needs to be addressed, and describing the goals and metrics required for the design of our scenario, a system architecture is now presented, see Figure 4.3. Firstly, the proposed design shows the main components of the system, which are:

- **Private group of users:** The system is composed of a private group, in which participants can be infrastructure providers or service providers. Since in private groups only certain members are allowed to access and alter the data, in this design, new users must ask for permission before joining the network. Furthermore, once entering the group, the newcomer can optionally act as a miner or just use the application without the need of maintaining a blockchain node.
- **User interface (UI):** Through an API (for automation), each member will efficiently interact with the blockchain application. Depending on the user's role, i.e infrastructure providers or service providers, the API will provide different functionalities. For example, SPs have the ability to request virtual networks and InPs to bid for the proposed virtual nodes, but not the other way around. Thus, the available operations must be restricted to the user's privileges.

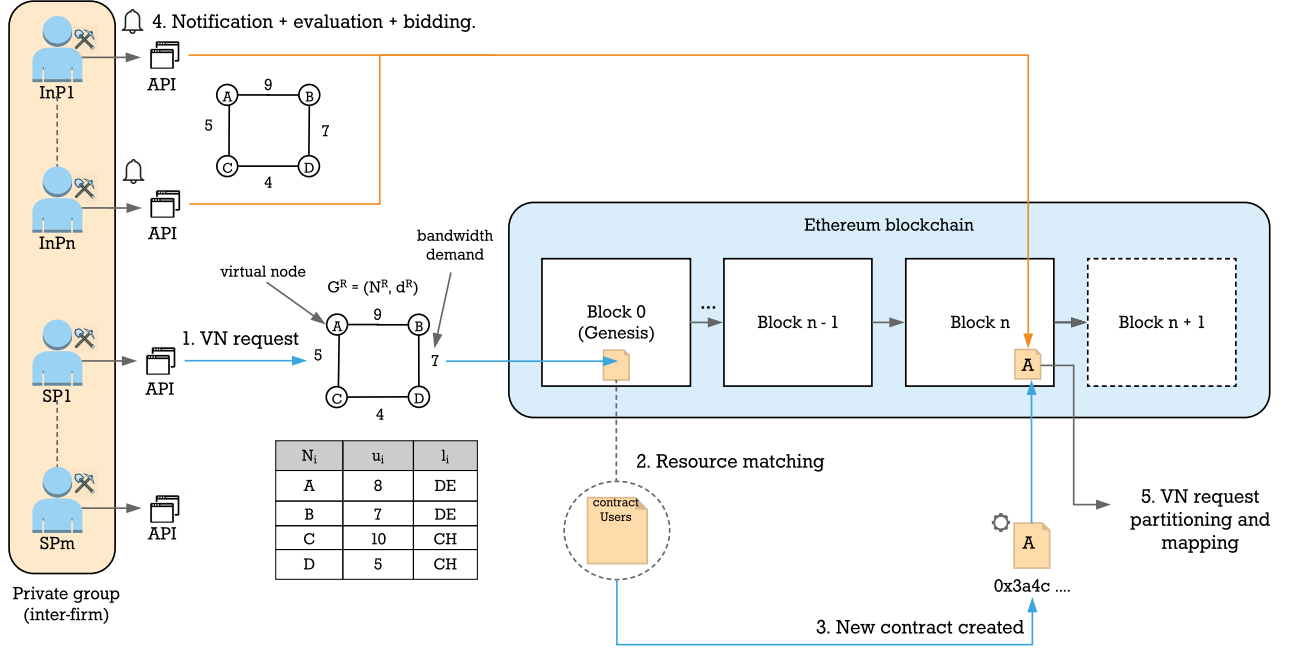


Figure 4.3: Our design for the system architecture. u_i states for upper bound cost and l_i for location

- **Ethereum blockchain:** One of the main differences between the Bitcoin and the Ethereum blockchain is that the first is transaction-based while the second account-based. For this reason, apart from a list of transactions, each Ethereum's block also stores the most recent state, i.e. the current account list with the corresponding balances. Nevertheless, every blockchain starts with the block 0, also called *Genesis block*. Since it is the first block, it is the only one not pointing to a predecessor and it is typically hardcoded in a file called *genesis.json*. This file contains some of the blockchain specifications, such as the difficulty or the gas limit. By modifying the value of these parameters, users are allowed to create blockchains based on their requirements. More details about the genesis file content used in our system will be discussed in chapter 5.
- **Smart contracts:** Above, we have mentioned that each Ethereum block stores a list of accounts with their corresponding balances, among others. These accounts can be externally owned accounts (users) or contract accounts (smart contracts). In the case of smart contracts, they enable reflecting user's needs into code and can be created or called either from another contract or simply by an EOA. In our system, a static contract called *Users* is contained, which will be permanently requestable. This contract will store all the participant's information, and whenever a new user registers, its state will be modified. In addition, this contract includes the InPs entered information about their substrate networks, such as the peering nodes or the substrate node types, which will be used to match the virtual network requests.

To provide a clearer view of the last mentioned components, the VNE functionality needs to be introduced. This VNE process entails the following steps (Figure 4.3):

1. **VN request:** A service provider requests a virtual network, which can be represented in a graph as $G^R = (N^R, d^R)$, where N^R consists of a set of k virtual nodes, and d^R the set of all bandwidth demands between the virtual nodes $i, j \in N^R$. Here, the superscript R stands for request. In addition, each virtual node $N_i \in N^R$ is formed by a collection of attributes, such as the desired location l_i or the upper bound cost u_i , where the last term may be the maximum amount of money that the SP is willing to pay for virtual node N_i . Otherwise, if an upper bound cost for the whole network is specified u_G , the one for each virtual node will be treated as $u_i = \frac{u_G}{k}$.

2. **Resource matching:** Before the auction starts, first, the associated set of attributes (e.g. location) of the requested virtual nodes N^R are matched with the InPs entered data in the *Users* contract. For instance, virtual nodes $\{A, B\}$ and $\{C, D\}$ will be matched with InPs possessing physical nodes in Germany (DE) and Switzerland (CH) respectively. In case no location l_i is specified for the nodes N^R , the VN request will be broadcasted to all the registered infrastructure providers.
3. **Auction start:** Using the previous information, the corresponding smart contract, i.e. a *limited time auction*, is created. This contract contains: (i) the data related to the virtual network nodes and links, (ii) the address of the SP willing to lease virtual nodes, (iii) a list of all the InPs addresses matched and (iv) an *end-time* field. Since the bidding must terminate, in the first step, the SP must also add this *end-time* field to the request.
4. **Notification and bidding period:** After the new auction contract creation, the included InPs, which are the only users with the ability to bid for the resources, are notified. Despite having a full view of the VN request and knowing the other matched InPs, each InP is only allowed to bid for their paired resources. For instance, InPs from Germany can only bid for the resources $\{A, B\}$. Before bidding, these users typically evaluate the request requirements into their substrate network, depicted as $G^S = (N^S, L^S)$, such that N^S and L^S are the set of physical nodes and physical links respectively. Although InPs are interested in serving requests at maximum profit, a single InP can barely serve all the virtual nodes requirements. As a result, the virtual network just may be supplied by different infrastructure providers. More details about the bidding mechanism will be presented in section 4.2.5.
5. **VN request partitioning and mapping:** Once the auction has finished, each virtual node will be assigned to the winning InPs so that the VN embedding cost is minimized. These winners will be publicly known by all the other participating users and thereupon, the service provider will contact them to ultimate the VN setting up, i.e. to perform the VN segment mapping.

In a nutshell, the presented system design uses the blockchain technology, in particular, smart contracts, to enhance the VN partitioning efficiency in a multi-provider scenario with limited information disclosure. Nevertheless, up to now, this is only a conceptual design with a lot of features that must be further discussed.

4.2.3 Blockchain Type and Mining Strategy

The Ethereum's main network, as Bitcoin, is public and permissionless, where all the users have access to the transferred values between accounts, since value transfers cannot be blinded. However, since our described scenario has a small number of users, the fully public decentralized approach will directly affect the application performance. Thus, as concluded in section 4.1, our focus is on **permissioned hybrid blockchains**, where a set of inter-firm entities control and maintain the blockchain functionality.

On the other hand, the default Ethereum's consensus algorithm will be used, which is at the time of writing, the **Proof-of-Work** protocol. Nevertheless, since the blockchain can be customized, in chapter 6, other mining strategies will be further investigated on their suitability.

In summary, although the Ethereum main network is public and permissionless, it can run as a private or hybrid blockchain, with the technology mechanisms and protocols remaining unaltered.

4.2.4 Authentication System

In the following, we will discuss how a user is registered and authenticated to the system, see Figure 4.4.

First of all, in order to deploy and use the presented design, a single or a group of InPs need to set up the environment. Afterwards, this consortium of InPs or a public authority, such as a regulatory agency, is responsible for inviting others users (InPs or SPs) to join his network, i.e. to create an Ethereum account (EOA) for the application.

However, once these users are registered, the communication between them must be secured. For this reason, the blockchain technology applies the **public-key cryptography**, also called asymmetrical cryptography, which generates a public and private key pair. The first, as the name suggests, it is publicly disclosed, and the second must remain confidential to its respective owner. Since these keys are mathematically related, the information encrypted with a private key can only be decrypted with a public one, and vice-versa. In addition, the Ethereum address is derived from this key pair, more specifically, from the last 160 bits of the $SHA3-256^2$ hash of the public key. Thus, once a user creates a new Ethereum account, the blockchain generates a public and private key pair for this user.

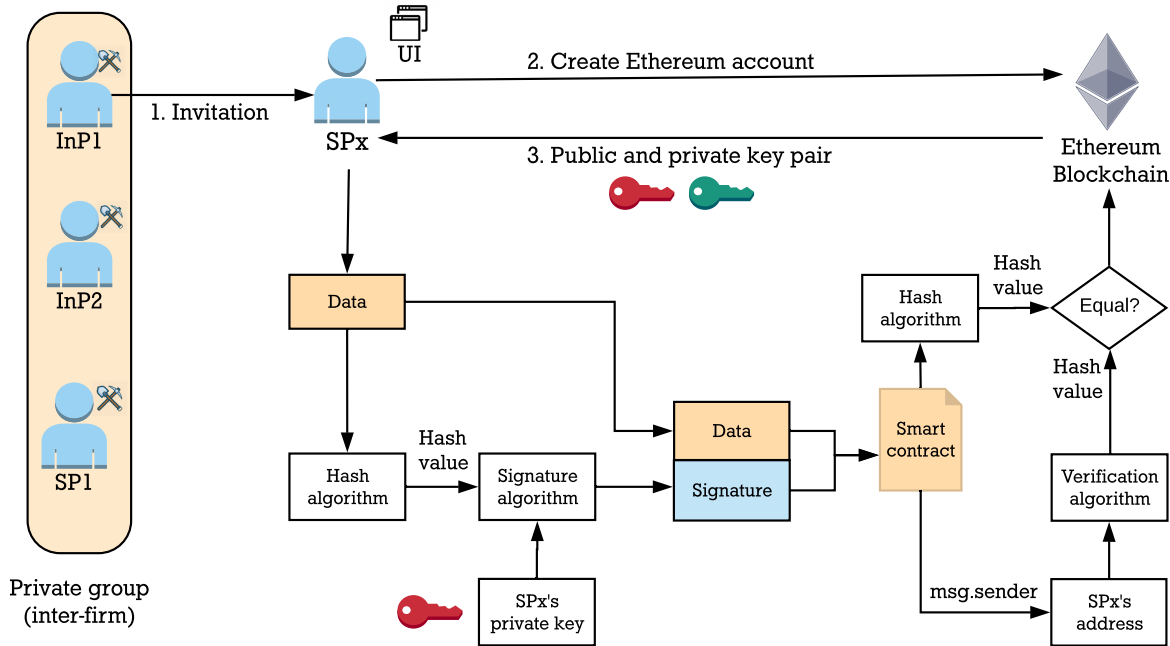


Figure 4.4: Our authentication design and used public-key cryptography

Furthermore, to validate the origin and integrity of the messages, users must always digitally sign their operations, by providing a **digital signature**. In particular, Ethereum uses the *Elliptic Curve Digital Signature Algorithm (ECDSA)*. In this process, see Figure 4.4, a sender uses his private key to sign a hashed version of the data, which produces the signature. This signature, together with the original data, is sent to the receiver, who uses the sender's public key and the data to verify that the information has been signed by who claims to be the owner and that the content has not been altered (hash value comparison).

It is important to note that smart contracts, if well-defined, also provide security in terms of access control and confidentiality. Despite the fact that Ethereum users can create and own smart contracts, the content is not always accessible or alterable. For instance, when users are not willing to broadcast the contract's information or simply when they want to avoid that every party is able to alter the state of the contract. Thus, these operations can be restricted only to a set of users, whose addresses need to be known by the contract account. In our VNE scenario, the matched InPs are the only users allowed to bid on specific resources.

Although the blockchain is a decentralized and distributed technology, where the actors do not need to know or trust each other, thanks to cryptography, a secure and unalterable communication between all the participants is achieved. In addition, since Ethereum uses smart contracts, the data access can also be restricted.

² <https://en.wikipedia.org/wiki/SHA-3>

4.2.5 The Vickrey Auction Model

In section 2.2.1, after reviewing different auction types, we have concluded that a **one-sided reverse Vickrey auction** seems to be the most suitable for brokerless virtual network embedding. In the following, a more thorough examination in adapting this auction model to the multi-provider VNE scenario with LID will be presented.

Firstly, in this scenario, we seek an efficient virtual network partitioning where the requested virtual nodes are assigned to the participating InPs, such that the VNE costs are minimized. Note that, despite an auction could use a multi-dimensional process, i.e. more parameters, in this report, only the price quotes of the notified InPs will be used to determine the minimum VNE cost. Hence, the InPs act as sellers, who submit bids for the services requested by the SP, and once the auction finishes, the virtual network will be splitted between the winning InPs. In this section, our focus will be on the techniques and auction models applied for the VN partitioning.

Symbol	Description
G^R	virtual network request formed by a set of virtual nodes N^R and a set of bandwidth demands d^R
N^R	set of requested virtual nodes
d^R	set of bandwidth demands between the virtual nodes $i, j \in N^R$
N_i	virtual node $N_i \in N^R$
N_i^t	virtual node N_i type
d_i	virtual node demand for node N_i
d_{ij}	bandwidth demand $d_{ij} \in d^R$ between the requested virtual nodes $i, j \in N^R$
P_l	l virtual nodes groups
u_i	upper bound cost for the virtual node N_i
u_G	upper bound cost for the virtual network G
l_i	desired location for the virtual node N_i
$b_n(N_i)$	n 's bid value for the virtual node N_i
$b_n(G)$	bid value for the virtual network G
$v_n(N_i)$	n 's real cost for the virtual node N_i
$v_n(G)$	n 's real cost for the virtual network G
$C_n(N_i)$	cost of virtual node N_i when assigned to InP n
$C_n(P_l)$	cost of InP n for P_l virtual nodes group
$C_n(G)$	cost of virtual network G when assigned to InP n
$C^*(G)$	minimum cost of virtual network G
$\gamma_n(N_i)$	InP n profit margin at the time of bidding on virtual node N_i
$\gamma_n(G)$	InP n profit margin at the time of bidding on virtual network G
G^S	substrate network formed by a set of substrate nodes N^S and a set of links L^S
N^S	set of substrate nodes
L^S	set of substrate links between the substrate nodes $i, j \in N^S$
$\alpha_i(N_s^*)$	dynamic cost for embedding N_i in the paired substrate node N_s^*
$\beta_{ij}(L_s^*)$	dynamic cost for sharing the paired substrate link L_s^*

Table 4.3: Notations

Since in our scenario, the SPS request the embedding of different virtual nodes $N_i \in N^R$, we are facing a multi-unit auction. Thus, before defining the minimum VN cost $C^*(G)$ using the *one-sided reverse*

Vickrey auction, it is important to note that $n = 1, \dots, N$ InPs can submit bids per virtual node $b_n(N_i)$ or for a group of virtual nodes $b_n(P_l)$, where P_l are the $l = 1, \dots, 2^k - 1$ possible virtual nodes combinations.

- **Per virtual node:** In a Vickrey auction with multiple items, the simplest form to bid is submitting a quote for each unit. In our scenario these units are virtual nodes, and in contrast to standard auctions, the **lowest bid** wins the service, inasmuch as the SPs are the ones who pay for the services offered by the InPs. Thus, to ensure not ending up with inflated prices, the SPs denote either an upper bound cost per virtual resource u_i , or a global one for the virtual network u_G , where $u_G = \sum_{i=1}^k u_i$. For instance, let $v_n(N_i)$ and $b_n(N_i)$ be, respectively, bidder n 's real cost and bid value for the virtual resource N_i . Since it is a *low-bidding second-price auction*, the cost $C_n(N_i)$ that the service provider pays to InP n , is given by (4.1). This value must also include the link costs, which after being evaluated by the InPs, are subsequently added to the quoted price $b_n(N_i)$. In this thesis, we assume that the internal link costs within an InP are negligible and that the external links, due to the lack of information of the other InP's substrate topologies, must be roughly estimated by the InPs. On the other hand, it is important to note that in our system, when $b_n(N_i) = b_j(N_i)$, the first bidder will be the provisional winner. For instance, in an English auction the same decision is taken, since when a bidder overcomes a price, the next participant must always bid a higher value than the current one. Thus, the InP's goal is to smartly quote $b_n(N_i) = v_n(N_i) \times [1 + \gamma_n(N_i)] \leq u_{N_i}$, where the symbol $\gamma_n(N_i) \in [0, 1]$ represents the profit margin that InP n takes at the time of bidding for N_i , knowing that the lower $\gamma_n(N_i)$ is, the higher the probability of winning, although at a smaller revenue. The cost $C_n(N_i)$ that a SP pays to InP n is given by:

$$C_n(N_i) = \begin{cases} \min b_j(N_i), & b_n(N_i) < b_j(N_i) \leq u_i : \forall j \neq n, \exists b_j(N_i) \neq \infty \\ b_n(N_i), & b_n(N_i) < b_j(N_i) \leq u_i : \forall j \neq n, \nexists b_j(N_i) \neq \infty \\ \infty, & \text{otherwise} \end{cases} \quad (4.1)$$

- **Per group of virtual nodes:** The Vickrey auction model behaves differently when a seller offers a package of items at a lower price, also called package pricing. In this context, rather than ending up paying the second price of the auction, the winner pays the **opportunity cost** for the units won, or in other words, the "damage" that his bids impose on the rest of the bidders. Formally, it is described as follows. Let x be a vector of outcomes again with $n = 1, \dots, N$ players bidding $b_n(X_n)$. The mechanism will seek for the outcome $x^* \in \argmax_{x_1, \dots, x_N} \sum_n b_n(X_n)$ that maximises the efficiency. Then, the cost for bidder n will be $p_n = \sum_{j \neq n} b_j(x_{-n}^*) - \sum_{j \neq n} b_j(x_j^*)$, where the term x_{-n}^* refers to the most efficient outcome without the presence of bidder n . Generally speaking, this expression can be seen as the result of the auction when bidder n is not participating in the auction, minus the total cost for the other bidders when n participates. However, in [AM⁺06], it has been proven that the Vickrey auction for multiple items suffers from serious weaknesses, e.g. zero revenues or unfair prices, which results in a poor performance. Due to the high degree of complexity, in this thesis, we investigate the package pricing when an InP bids for the whole VN network $C_n(G)$, rather than for partial biddings $C_n(P_l)$. For instance, as shown in Figure 4.5 example, we will just consider bids for the entire network $b_n(ABCD)$ and not the other possible combinations, such as $b_n(AB)$ or $b_n(AB) + b_n(ABC)$. Hence, in this case the cost $C_n(G)$ will be treated exactly as in the per virtual node auction, represented as:

$$C_n(G) = \begin{cases} \min b_j(G), & b_n(G) < b_j(G) \leq u_G : \forall j \neq n, \exists b_j(G) \neq \infty \\ b_n(G), & b_n(G) < b_j(G) \leq u_G : \forall j \neq n, \nexists b_j(G) \neq \infty \\ \infty, & \text{otherwise} \end{cases} \quad (4.2)$$

Finally, an algorithm that combines individual bidding and package pricing, and calculates the minimum VN cost $C_m(G)$, will be designed. Firstly, in case no bid per package is provided, the cost for the VN network can be generally expressed as $C^*(G) = \sum_{i=1}^k C_n(N_i)$, which is the sum of all minimum virtual nodes costs. However, when the two bidding types are used, the minimum cost will be calculated as follows 4.3.

$$C^*(G) = \min\{C_n(G), \sum_{i=1}^k C_n(N_i)\} \quad (4.3)$$

The last expression shows that in the end, the system first computes independently an individual and a package auction. Afterwards, the resulting costs for these two tenders are compared, whose minimum value gives us the final VNE cost $C^*(G)$. In case $C_n(G) = \sum_{i=1}^k C_n(N_i)$, the virtual network will be embedded to the InP offering package pricing, since the performance is always better within a single InP. Moreover, the common InPs bidding strategies and the encountered weaknesses of the Vickrey model for package pricing, need to be investigated in this design. Algorithm 1 shows the pseudocode of the InP n resource matching and subsequently bidding strategy, where N_i^t corresponds to the virtual node type of N_i .

Algorithm 1: InP n resource matching and bidding strategy

```

1: if  $N_s \leftarrow \forall l_i \wedge \forall N_i^t$  then
2:   return  $b_n(G) = v_n(G) \times [1 + \gamma_n(G)] \leq u_G : v_n(G) = \sum_{i=1}^k v_n(N_i), \gamma_n(G) \in [0, 1]$ 
3: else
4:   for  $N_i \in N^R$  do
5:     if  $N_s \leftarrow l_i \wedge N_i^t$  then
6:        $b_n(N_i) = v_n(N_i) \times [1 + \gamma_n(N_i)] \leq u_{N_i} : \gamma_n(N_i) \in [0, 1]$ 
7:   return  $b_n(N^R)$ 

```

Firstly, according to Algorithm 1, an InP bids either for the entire virtual network $b_n(G)$ or for individual virtual nodes $b_n(N_i)$ depending on the amount of paired resources. In other words, if all the virtual nodes locations l_i and types N_i^t are matched with certain InP substrate nodes N_s , then the system enables that the infrastructure provider bids for the entire virtual network. If not, an InP is only allowed to quote prices for the paired virtual nodes.

To better explain the bidding strategy of Algorithm 1, it is worth taking a look at the example from Figure 4.5. There, four InPs are competing to win virtual nodes $\{A, B, C, D\}$, each using profit margin values of $\{\gamma_1 = 0.6, \gamma_2 = 0.1, \gamma_3 = 0, \gamma_4 = 0.2\}$. In this example, the profit margins are identical for each N_i and G . At the end of this auction, since the individual quotes are lower than the package pricing offer $C^*(G) = \min\{30, 29\}$, the virtual network is partitioned across different InPs, i.e. $A \rightarrow InP_1, B \rightarrow InP_1, C \rightarrow InP_4, D \rightarrow InP_3$. In addition, the prices that the SP must pay to each InP are the ones resulting from the *low-bidding second-price auction* $C_n(N_i)$.

In the above-mentioned example, is important to note that although the InP_1 applies the highest profit margin, it is the one with the best outcome. Thus, since the virtual node and virtual network bids are determined first by $v_n(N_i)$, which is also used in $v_n(G) = \sum_{i=1}^k v_n(N_i)$, and second by $\gamma_n(N_i)$ and $\gamma_n(G)$, these values need to be examined:

- **Real cost $v_n(N_i)$:** Since network virtualization is a real-time process where the demand and availability of the resources are constantly changing, our approach encourages that users apply **dynamic pricing models**. A dynamic pricing model, in contrast to fixed pricing, is a strategy where the prices are flexibly adapted to the current market demands, rather than always charging the same costs. In our approach, InPs could use algorithms that take into account the SP demands and the current

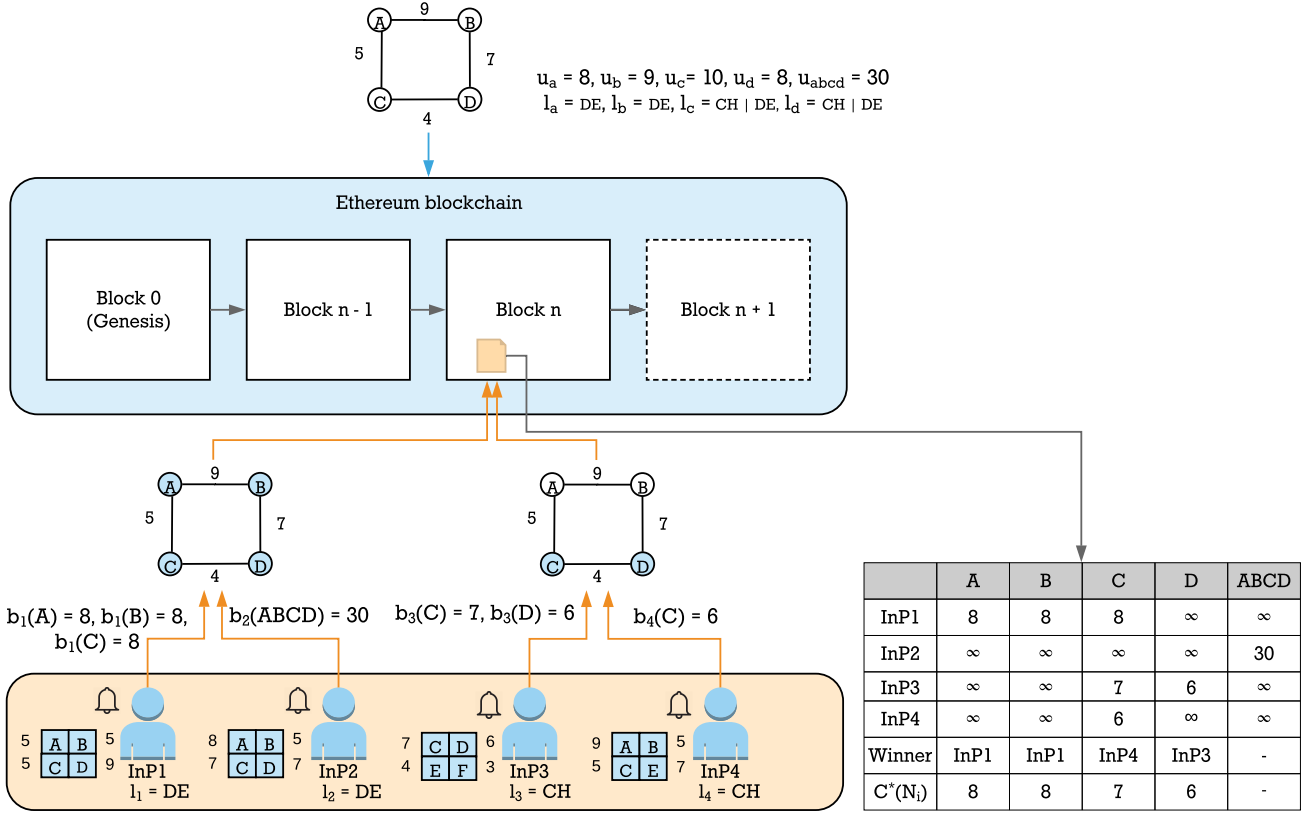


Figure 4.5: Designed auction combining the Vickrey model with package pricing example

availability of the resources, to better define their product prices, and hence, to have a better market positioning or to open up new markets. For instance, they could use the demand and supply principle in microeconomics, which states that if the demand for a product short in supply is high, the price increases. In our scenario, this demand is related either with the computing or the bandwidth requirements, and the supply with the resource utilization. Thus, the real cost $v_n(N_i)$ is given by (4.4), where $\alpha_i(N_s^*)$ and $\beta_{ij}(L_s^*)$ are the dynamic prices for the paired substrate node N_s^* and link L_s^* , and whose values will be updated according to the resource availability (see chapter 6). Apart from demand and supply, InPs could also use automated processes based on other factors, such as competitive pricing or customer behaviors. In the former, since the reserved prices for the virtual nodes are publicly known when an auction finishes, it allows InPs to learn from competitors in order to be more proficient in incoming auctions. In the latter, the virtual node upper bound costs entered by the SPs on their VN requests, can also be used to adjust the InP prices on customers needs. Overall, due to the high elasticity of the prices, InPs should recalculate periodically their costs, in order to compete in the modern market. This cost estimation can be performed based on many other factors, which can be thoroughly examined in future research.

$$v_n(N_i) = \alpha_i(N_s^*) + \sum_{j=1}^k \frac{\beta_{ij}(L_s^*)}{2} \quad \forall j \in N^R, \exists d_{ij} \neq \infty : i \neq j \quad (4.4)$$

- **Profit margins $\gamma_n(N_i)$ and $\gamma_n(G)$:** The profit margin is a measure to indicate the profitability of a product, which is extracted from the net profit divided by the cost. Equations (4.5) and (4.6), give the values of the profit margins for a virtual node and for the entire network. It is important to note that these values are provisional (at the time of bidding), since if a second lowest quote exists, as it is a *second-price auction*, the values $b_n(N_i)$ and $b_n(G)$ will be replaced by $C_n(N_i)$ and $C_n(G)$ respectively.

Typically, a high-profit margin produces a higher revenue, although at a smaller probability of winning. Nevertheless, from the example in Figure 4.5, we can observe that if an InP performs a good market research (most probably InP_1), it can also obtain virtual nodes through high-profit margins. Thus, in chapter 6, we will examine how the risk taken by the InPs at the time of determining the profit margins, affects the VN embedding outcome.

$$\gamma_n(N_i) = \frac{b_n(N_i) - v_n(N_i)}{v_n(N_i)} \quad \gamma_n(N_i) \in [0, 1] \quad (4.5)$$

$$\gamma_n(G) = \frac{b_n(G) - v_n(G)}{v_n(G)} \quad \gamma_n(G) \in [0, 1] \quad (4.6)$$

Lastly, we will analyse the weaknesses of the Vickrey auction model for package pricing in our design [AM⁺06], and observe how the potential attacks in Internet auctions could hinder our system performance [BM00]:

- **Bidder non-monotonicity:** Adding other bidders in a *Vickrey auction* with package pricing can result in low or zero revenues, such that bidders pay the opportunity cost. Hence, the system has potential vulnerabilities that the bidders can exploit. In our design, as the virtual network and the corresponding virtual nodes are treated as single independent items, adding more bidders will only increase the resource competition.
- **Multiple bidding identities:** In many Internet auctions, a single bidder can use multiple pseudonyms or accounts to bid and increase their profits. In our system, as users can only be registered once (through an invitation), it is not possible for an InP to have multiple Ethereum accounts to compromise the final outcome. Thus, this is a controlled system where InPs, such as Deutsche Telekom or Swisscom, are identified by a single account.
- **Bid shielding:** It consists in the withdraw of bids in the last minute. Since our application interacts with a blockchain, where submitted data is immutable, this has no influence on the system.
- **Bid sniping:** The auctions started by the SPs have a fixed duration, after which no further bids are accepted. In this case, an InP could enforce a denial of service attack, by sending multiple last-minute bids, in order to prevent that the system computes other bids and hence, winning the auction. For this reason, in our system, each InP will have to first pay his bid to the smart contract, and a percentage of this total quantity will be refunded after the sealed bid is opened.

4.3 Summary

Virtual network embedding (VNE) enables SPs to embed virtual nodes across different InPs to create customized virtual networks. Typically, these InPs are not willing to disclose detailed information about their resources availability or network topologies, which directly affects the VNE performance. In addition, the service providers seek to minimize VNE embedding costs, such that the VN partitioning is treated as a cost minimization problem.

In this thesis, an approach to improve the VNE throughput under the limited information disclosure scenario has been introduced. More precisely, a complete system using the Ethereum blockchain has been presented, which thanks to smart contracts, enables the creation of new embedding strategies to facilitate the service negotiation between the InPs and SPs. In our case, an algorithm adapted from the Vickrey auction model has been designed, which improves the VN partitioning efficiency. In the following chapters, the implementation of this design will be evaluated.

We do not claim that our approach is the only way of performing VN partitioning. However, it enables brokerless VN embedding in a distributed environment. Furthermore, to the best of our knowledge, this is the first approach using blockchain in the VNE context, and thanks to smart contract's flexibility, it can serve as a starting point for more upcoming investigations.

5 Implementation

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.

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 (contracts + our system). 4. Component relations in the source code 5. Code examples 6. Implement private main network (check geth ENODES number and file where to modify. Also check a dynamic way to do it). Data in contracts in bytecode, contracts not predefined structure (Check this). 7. Mining 8. Vickrey (fake value), [BM00] and who opens the bids? (paper explanation). 9. Implementation workflow

Explain that other parameters (not cost) could be used to perform the service negotiation. 2. A well-design UI, should be designed with a focus on usability and efficiency. 2. Genesis.json content must be identical for all the other mining nodes in order to work on the same blockchain, 3. Investigate web3.eth.sign(web3.eth.accounts[0], address), the address should be unlocked. How this is possible? 3. solidity ecrecover, msgHash, v, r, s. To create it, they must register to the application entering the following fields: (i) an email, (ii) the user's role, where users can select in a dropdown menu InP or SP, and (iii) a password. This will create an Ethereum owned account (EOA), whose address need to be stored. Remember that if a user forgets the address, all the information regarding the account, e.g. ETH earned, will be lost. Invite People to the private BC. Ethereum account in the application with some fields. Users fixed amount of Ether. 7. For the implementation, like in Bitcoin at the beginning, a supernodes or mining users will be placed. This users, will be the only responsible for mining in the blockchain.

9. Implementation workflow (1. InP registers from invitation, 2. InP add his peering nodes + virtual node types info, 3. SP request, 4. VN partitioning).

5.1 Design Decisions

5.2 Architecture

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. Which 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 chosen 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. To evaluate: 1. transaction time/speed (considering propagation), GENI maybe, if a node goes down, which is the synchronisation time, better algorithms for consensus -> performance.

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. 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. 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.
- [ama18] Amazon ec2 instance types. <http://aws.amazon.com/ec2/instance-types>, 2018. Accessed: 2018-01-16.
- [B⁺14] Vitalik Buterin et al. A next-generation smart contract and decentralized application platform. *white paper*, 2014.
- [bit09] Bitcoin is an innovative payment network and a new kind of money. <https://bitcoin.org/e/>, 2009. Accessed: 2018-03-26.
- [BM00] Colin Boyd and Wenbo Mao. *Security issues for electronic auctions*. Hewlett-Packard Laboratories Bristol (UK), 2000.
- [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 Ameer, 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.
- [hyp17] Business blockchain frameworks hosted with hyperledger. <https://www.hyperledger.org/>, 2017. Accessed: 2018-03-26.
- [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-Thorold, 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.
- [Tea17] Roster Team. Blockchain at berkeley. <https://blockchain.berkeley.edu>, 2017. 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.