Institute of Architecture of Application Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Masterarbeit

Architectural Analysis of Blockchain-based Approaches to a Decentralised Energy Management System

Philipp Kraus

Course of Study:	Softwaretechnik
Examiner:	Prof. Dr. Dr. h. c. Frank Leymann
Supervisor:	M.Sc. Ghareeb Falazi *University of Stuttgart* Dr. Thomas Brenner *OLI Systems GmbH*
Commenced:	June 14, 2018
Completed:	December 14, 2018

Abstract

Due to the energy transition, the traditional power grid turns from a centralised grid, with single big power plants, into a heterogeneous one, with many unpredictable and volatile electricity producers and consumers. Moreover in the future, common households not only consume energy, but also feed their own energy with photovoltaic plants, batteries or heat pumps, into the grid. This creates a demand for a management system which allows forecasting, planning and tracing electricity usage from producers to consumers and is extendible to the participants' needs. Utilising blockchain technology allows one way of mapping the heterogeneous and decentralised characteristics of the future power grid into a suitable decentralised energy management system. This work analyses possible decentralised concepts and already existing blockchain solutions in order to propose a generic architecture solution for a decentralised energy management system. In addition, the doability of the proposed architecture is shown with a possible implementation, using the Ethereum platform. Furthermore, the implementation is realised in a minimal prototype. Conclusively, the architecture and its implementation are evaluated against possible requirements of a decentralised energy management system, which verifies their suitability for the given problem. This work is not only proposing a solution for an energy-related problem, but also shall help to find the true benefits of the blockchain technology and its current ecosystem.

Kurzfassung

Aufgrund der Energiewende wandelt sich das traditionelle Stromnetz von einem zentralen Netz, mit einzelnen großen Kraftwerken, in ein heterogenes Netz, mit vielen unvorhersehbaren und volatilen Stormerzeugern und -verbrauchern. Darüber hinaus verbrauchen zukünftig gewöhnliche Haushalte nicht nur Energie, sondern speisen auch eigene Energie mit Photovoltaikanlagen, Batterien oder Wärmepumpen in das Netz ein. Dadurch entsteht eine Nachfrage nach einem Energiemanagementsystem, welches die Prognose, Planung und Rückverfolgung des Stromverbrauchs von Erzeugern zu Verbrauchern ermöglicht und auf die Bedürfnisse der Teilnehmer erweiterbar ist. Durch die Verwendung der Blockchain-Technologie können die heterogenen und dezentralen Merkmale des zukünftigen Stromnetzes in einem geeigneten dezentralen Energiemanagementsystem abgebildet werden. Diese Arbeit analysiert mögliche dezentrale Konzepte und bereits vorhande Blockchain-Lösungen, um eine generische Architekturlösung für ein dezentrales Energiemanagementsystem vorzuschlagen. Darüber hinaus wird die Machbarkeit der vorgeschlagenen Architektur, unter Verwendung der Ethereum-Plattform, mit einer möglichen Implementierung gezeigt. Zusätzlich wird die Implementierung in einem mininalen Prototypen realisiert. Abschließend wird die Architektur und ihre Implementierung anhand möglicher Anforderungen eines dezentralen Energiemanagementsystems bewertet, was deren Eignung für das gegebene Problem belegen soll. Diese Arbeit schlägt nicht nur eine Lösung für ein Problem im Energiekontext vor, sondern soll auch dazu beitragen die wahren Vorteile der Blockchain-Technologie und ihres aktuellen Ökosystems zu finden.

Contents

List of Figures

List of Tables

List of Listings

List of Abbreviations

DApp decentralised application. 27

DEMS decentralised energy management system. 37

ECS Ethereum Client Specification. 33

EEA Enterprise Ethereum Alliance. 33

EMS energy management system. 28

EVM Ethereum Virtual Machine. 49

EWT Energy Web Token. 89

FR functional requirement. 38

IMS Intelligent Metering System. 69

NFR non-functional requirement. 38

SDK software development kit. 30

SML Smart Message Language Protocol. 86

VPN virtual private network. 72

Wasm WebAssembly. 49

1 Introduction

The current electricity market houses a variety of providers with different energy sources and output amount. Moreover, the source and amount of electrical energy a household uses play an important role for various environmental and financial reasons. In the past, the electricity grid only consisted of traditional large producers such as coal plants or atomic power plants. The present grid more and more emerges to a heterogeneous one consisting of many independent small producers. Due to the energy transition, this process will inevitably change the current electrical power systems and replace them with a much more complex and decentralised system. Many households will not only consume a constant predictable amount of energy anymore but will also feed energy into the grid with their own photovoltaic panels, batteries or electric vehicles. This turns common consumers into unpredictable prosumers. Because many of these new producers produce their electricity from renewable sources such as sun, wind and water, those strongly fluctuating sources will lead to highly varying prices for certain time periods [MSN+16]. To this end, an *energy management system* which allows forecasting, planning and tracing electricity usage from power plants to end users and permits customers with preferences regarding the source of electricity and their management policies, is needed. The organisation of five prosumer households within an energy management system would already result in an efficiency increase of 98 % [SAG18].

1.1 Motivation

Centralised energy management systems that control all the devices of a power network allow to achieving these goals. Nevertheless, the current power system develops into a too complex one where such a top-down approach does not fulfil the participant's needs, anymore [SAG18]. With an increasing amount of actors in such a system, a centralised energy management system would suffer from scalability, maintenance and trust issues [EHS15]. This makes it necessary to develop a decentralised energy management system which has the capability of addressing these concerns. One way of constructing such a system is based on the blockchain technology. Blockchain technology [Nak08] facilitates the creation of decentralised networks of untrusting nodes that share a common state in the form of an immutable append-only ledger of events. Such a network can easily scale and does not have a single point-of-failure. However, the blockchain ecosystem provides several approaches which differ, for example, in the transaction verification, storage principles and capabilities of the architecture [YMRS18]. Thus, finding a suitable technology for a given use case can be challenging. In order to find a capable blockchain architecture for the requirements of a decentralised energy management system, it is necessary to conduct an architectural analysis and evaluate existing blockchain solutions. Having an architecture evaluation enables benefits like making the right business and technology decisions, controlling the product quality or managing the evaluation and migration of existing software systems [KN16]. Considering adequate research as one way to leverage a technology in the Gardner Hype Chart [Pan17] away from its hype climax onto the

plateau of productivity, this architectural analysis should not only be in the scope of a decentralised energy management system, but should also help finding the true benefits and advantages of the blockchain technology in its current ecosystem [Pan17].

1.2 Goal

It is the goal of this thesis to analyse possible blockchain-based architecture solutions for a decentralised energy management system. In the light of this analysis, recommendations regarding a generic blockchain-based architecture that tackles these kinds of problems should be provided. Moreover, it is the goal to show the feasibility of the architecture by discussing its applicability to an existing approach and implement a minimal prototype capable of utilising it. Finally, the proposed architecture concept shall be evaluated together with its recommended implementation and the realised prototype.

1.3 Outline

After the introduction in *Chapter 1 - Introduction*, *Chapter 2 - Background Knowledge* describes topics that are necessary for the understanding. Major topics are fundamental concepts to understand blockchain, the blockchain concept itself and its decentralised architecture stack. Furthermore, chapter 2 describes energy management and energy management systems in more detail and gives an introduction to operating system development for embedded devices. *Chapter 3 - Related Work* introduces two related topics wrote upfront this thesis. Subsequently, *Chapter 4 - Decentralised Energy Management System* introduces the need for a decentralised energy management system and defines its consequential requirements. Also, it is analysed whether blockchain technologies are an appropriate tool for proposing an architecture approach for a decentralised energy management system. In *Chapter 5 - Analysis of Suitable Blockchain Architectures and Technologies* several suitable technology concepts and existing solutions are examined and checked whether they could help to fulfil the requirements of a decentralised energy management system. With the knowledge gained in the previous chapters *Chapter 6 - Concept for a Generic Architecture Proposal* proposes an architecture which tries to fulfil all the identified requirements. *Chapter 7 - Implementation* then proposes a possible implementation of the introduced concept, together with the realisation of a minimal prototype. In order to clarify whether the concept and the implementation fulfil the identified requirements of chapter 4, an evaluation against the described requirements is carried out in *Chapter 8 - Evaluation*. Conclusively, *Chapter 9 - Conclusion* summarises the work done during this thesis and gives an outlook for possible future works.

2 Background Knowledge

In order to understand the following work to its full extend certain fundamental background knowledge is necessary, which needs to be described beforehand. The following chapter introduces basic concepts about central, distributed and decentralised architectures and the basics of hashing and its application. Subsequently, the fundamental concept of the blockchain is introduced and the emerging decentralised architecture stack is presented. The last to sections of this chapter are about the energy-related context of this work and about the development of operating systems for embedded devices, such as smart devices in power grids.

2.1 Concepts

The blockchain technology and its developed ecosystem are based on basic concepts. In the following, the fundamental ones are explained.

2.1.1 Central, Distributed, Decentralised

One of the key ideas behind blockchain protocols is the idea to develop a decentralised system. Therefore it is necessary to understand what decentralisation means. First of all system architectures can be divided into three types, *centralised*, *distributed* and *decentralised*. S. Raval [Rav16] defines centralised systems as the most common type of traditional systems. *Centralised* systems are systems where every action is controlled by a single central entity and interaction with the system is always dependent from such a central entity. Figure 2.1 (a) shows a visualisation of a centralised system with its dependencies. *Distributed* systems are the second type of system architectures which try to spread computational effort over multiple nodes in order to reduce latency and dependency from a central entity. Still, they consist out of a single point of control, but with the respect to distributing the resulting tasks over multiple nodes. Figure 2.1 (b) shows such a system. The third and aimed type by most of the blockchain systems is a decentralised system. *Decentralised* systems have no single entity which directs tasks to other nodes. Each node can be seen as an equal entity within such a system [Rav16]. Therefore a decentralised system has no single point of control and no single point of failure [FK17]. Figure 2.1 visualises the interconnection of equal entities.

In addition V. Buterin [But17b] classifies the meaning of decentralisation into three different types, *architectural* decentralisation, *political* decentralisation, and *logical* decentralisation. Thus, a system can have centralised characteristics not only to its technical extend. *Architectural* centralisation means only the system architecture itself. A system is architectural decentralised if there are enough nodes which prevent the system from a single point of failure. *Political* decentralisation, however, focuses on the governmental decisions done by the individuals controlling the system. If there is only one single entity which takes decisions, the system is still (political) centralised. Finally *logical*

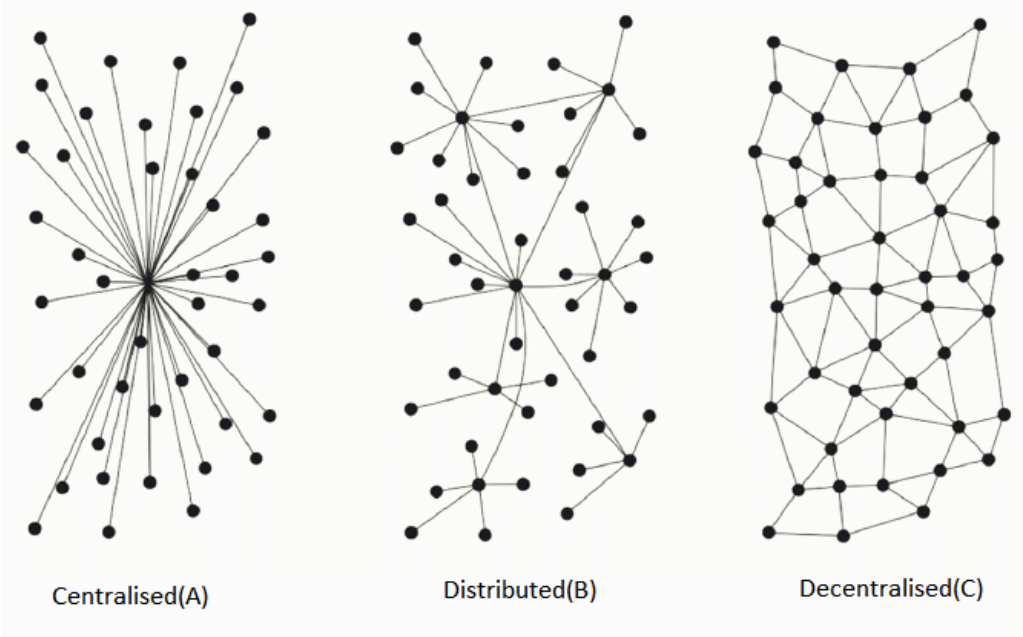


Figure 2.1: Visualisation of the differences between centralised, distributed and decentralised system architectures [Hoe14].

decentralisation means the relationship between interfaces and data structures. If the system consists of many nodes governed by individual independent parties but the interfaces and data structures are defined in a monolithic way the system is still (logical) centralised. [But17b]

2.1.2 Hashing

Another concept which is often used together with blockchain technologies is the concept of hashing. A hash-function is a function $h(e)$ which maps an element e to a position f within a field of N entries. The function h follows several conditions. First, N has to be *finite*. Second, h needs to be *definite*, which means that for the same input e, $h(e)$ is always the same f. Lastly, h has to be mostly *collision free* and provides a sufficient *spreading* of the input values. This means, that for most of the inputs e_1 and e_2 $h(e_1) \neq h(e_2)$. Furthermore, the position f of two input values which differ from each other only significantly shall be spread apart from each other in the resulting field. In the best case the probability P for an element e to be mapped to a specific field f is for all possible inputs $P = \frac{1}{N}$. This reduces the probability of collisions significantly. [SS04]

Hashing functions allow to efficiently map an input value to a specific field in a map but not necessary to reproduce the input value from the specific field. Reproducing the input values can take a significant amount of time. This happens because, although the input values might be finite in the real world the hashing function is a function for an infinite number of inputs. As long the number of inputs is small enough to prevent the hash-functions from collisions executing the reversed function $h^{-1}(e)$ of the hash-function might result in an infinite number of possible inputs e_n where $h(e_n) = f \; \forall e_n$ with $n \in \mathbb{N}$. Therefore finding the correct field f of an input e is easy, but finding

the original input value e_i of a field f is difficult, which makes hash-functions to possible *one-way* functions. One-way functions are often used when one wants to identify an input with a unique identifier but not necessarily wants the identifier to be traced back to the input value. [Sch08]

Merkle Trees

A Merkle tree is an infinite, mostly binary tree of one-time signatures introduced by R. Merkle [Mer88]. A one-time signature is the result value of a hash function for a specific input which is hardly or not at all recoverable. Each node of the tree signs its leaves into a new one-time signature. The resulting hash can then be treated as a single hash for an input consisting of multiple child hashes. Therefore the root of the tree is the recursive signature of all child nodes and their child nodes. Merkle trees enable to combine data together to a single signature. Figure 2.2 shows how hashes of multiple data packets can be combined in a single one-time signature. [Mer88]

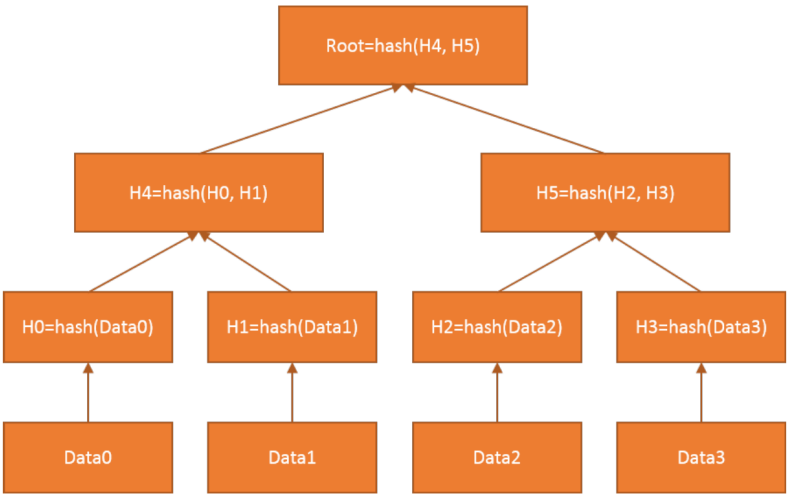


Figure 2.2: A Merkle tree with the aggregated data in the leaves of the tree and their signatures [YMRS18].

2.2 Blockchain

Blockchain technologies aim to provide a decentralised, persistent anonymous and auditable technology [ZXD+17]. Which means that every action made up does not rely on a single party. That every action accepted by the decentralised system is stored permanently with no chance of roll-back or deletion. That each user can interact with the system in an anonymous or at least a pseudonymous way. That every action persistent in the system is traceable and transparently tracked [ZXD+17].

In order to do so, blockchain technologies are divided into seven key components defining the architecture of a blockchain. These components are [YMRS18]:

1. Distributed Ledgers
2. Transactions
3. Addresses and Address Derivation
4. Asymmetric-Key Cryptography
5. Blocks
6. Chaining of Blocks
7. Utilisation of hashing technologies

On its fundamental basis, a blockchain is a technology for distributing deterministic state changes in a *ledger data structure*. Each node participating in the system strives to hold a copy of this ledger. Every state change is stored as a *transaction* which defines the action necessary to get from the previous state into the actual state. When a consensus within the nodes is reached, each new transaction is propagated over all nodes and is added to the ledger. Figure 2.3 shows a network of nodes distributing state changes in a ledger. Pending transactions are transactions which are not yet added to the ledger and consensus is not yet found. Defining distributed ledgers as a part of blockchain technologies means that distributed ledgers themselves are a superset and blockchain technologies a subset of distributed ledger technologies. [YMRS18]

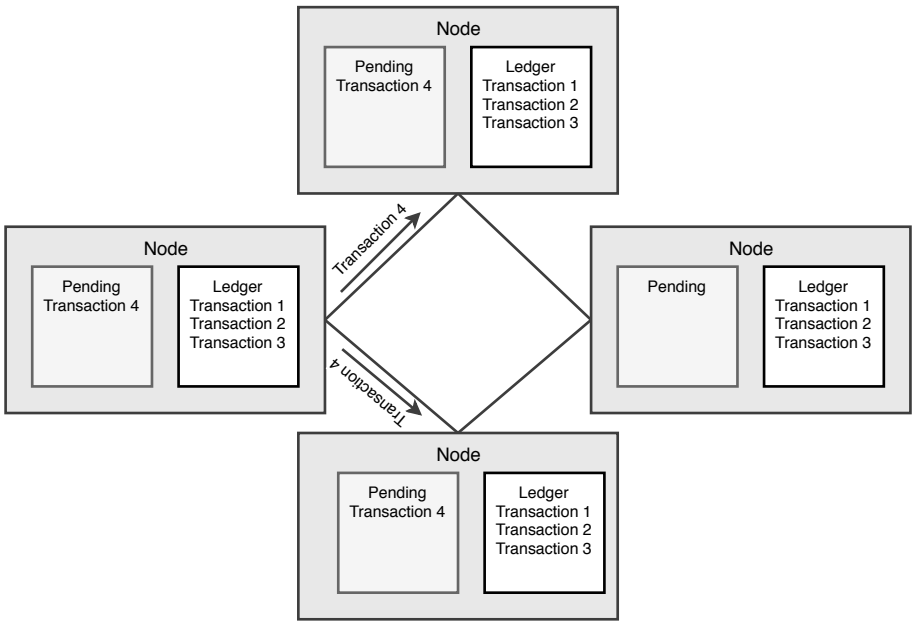


Figure 2.3: Network of nodes, each holding a copy of a ledger. Transaction 4 is being propagated through the network to every node. [YMRS18]

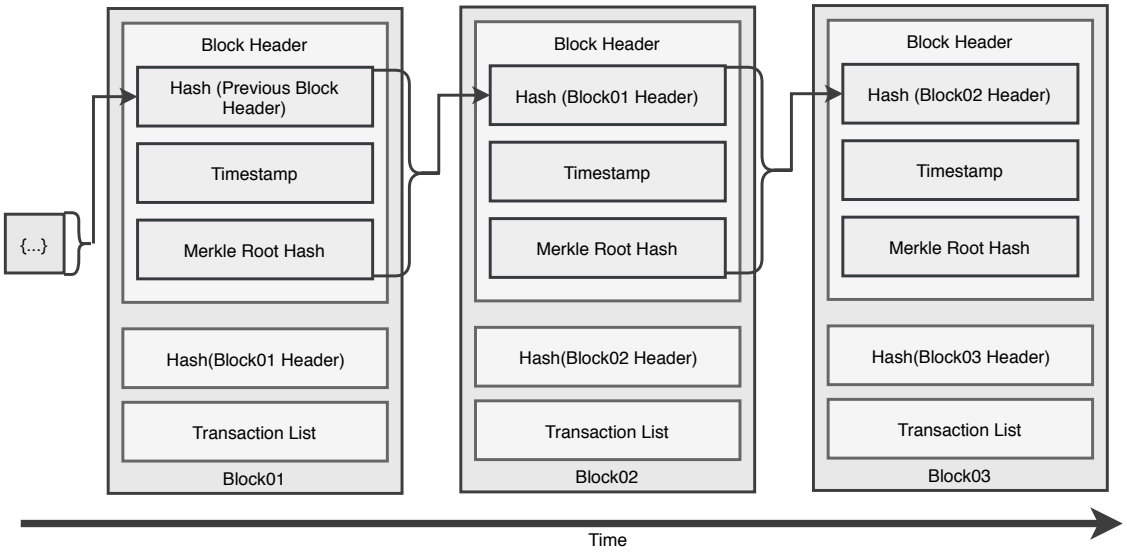


Figure 2.4: Possible data structure of a blockchain [YMRS18].

Most blockchain architectures are transferring assets as state changes from one participant to another. To implement this, *addresses* which identify the participants of a transaction with the help of *asymmetric-key cryptography*, are established. [YMRS18]
Asymmetric-key cryptography makes use of two separate keys, a private and a public key. Messages encrypted with a public key can only be decrypted with the correspondent private key and vice-versa. Because of one of the keys, ideally, the private key is always kept secret the other key can be published to whoever shall be able to encrypt messages with it. [Kum13]
Every address is associated with a private and a public key-pair. By using the keys associated with the address participants can prove the actual possession of the address by signing the transaction with their private key. By verifying this, every other participant of the ledger can validate the integrity of the transaction and therefore the validity of the new state changes. [YMRS18]

Changes are propagated in form of *blocks* over the network. Blocks are a number of aggregated state changes which shall be appended to the local state of each node. Hence every new block gets chained onto the last block which forms a traceable log of all state changes done, which is called the blockchain. In general, every block consists of its block number and the aggregated transactions. To ensure the authenticity of each copy of the blockchain and the distributed blocks *hashing technology* is used. Each block appended to the blockchain gets hashed which creates a unique digit representing the block. No changes on the data of the block can be made without changing the resulting hash of the block. Each block additionally has its current block hash and the block hash of the previous block. Because storing every single hash of each transaction in the block is inefficient, a Merkle tree data structure is used in which only the root hash of the Merkle tree will be stored in the block. Figure 2.4 shows a schematical overview of how blocks and their chain could look like. [YMRS18]

2.2.1 Classification of Blockchain

Blockchain technologies differ in several aspects. Therefore six characteristics for classification can be identified. The level of permission and access restrictions, the concept of transaction consensus, the level of central regulations, immutability of data and the way identity is treated within the blockchain.

Consensus Determination

In a network where all nodes are holding the same copy of a ledger, a mechanism for finding the next transaction for appending onto the ledger has to be conducted. This mechanism is called consensus determination. It is the main goal of a consensus mechanism to prevent Byzantine nodes from being able to write into the ledger or propagate faulty blocks. Byzantine nodes are nodes which try to manipulate the blockchain out of economical or destructive motivations against single participants or the network itself. Several different consensus mechanisms have been proposed and implemented by blockchain technologies. [YMRS18]

The metaphor of being *Byzantine* finds its origins in the Byzantine General Problem, where a fictive situation is explained with an army camping outside of an enemy city. It is the goal of the generals to coordinate an attack against the city, which can only be captured if all generals attack together. In the scenario, the generals can only speak with each other over messages and cannot trust each other. All generals are competing against each other for the favour of the king. If one general sends a message to the others to attack, a Byzantine general might reply with a commit message but will not commit the fight in order to weaken his rivals. [LSP82]
In the metaphor, a Byzantine node acts maliciously out of similar motives as a Byzantine general, which is why one cannot trust any single node at all.

Proof of Work Proof of work is the most prominent consensus mechanism and relies on a puzzle which needs to be solved. The node which solved the puzzle first has the right to decide which block shall be appended onto the blockchain next. It is the idea, that the participant who solved the puzzle proves that he spend a sufficient amount of computing power into solving it [Woo18]. This difficulty makes it indeterministic to predict which node will solve the puzzle first and therefore unlikely for Byzantine nodes to modify the latest block. In advance proof of work is used as a protection for potential spammers who would need to invest, too many costs in order to write many entries onto the blockchain. This characteristic can also be a drawback because many nodes participating in the consensus mechanism result in more difficult puzzles. The more difficult a puzzle gets the more computational power and therefore energy has to be invested to solve it. This leads to unreasonable high energy costs.

Proof of Stake Proof of stake is based on the assumption, that participants putting much economical effort into a system are most likely interested in its correct execution. In proof of stake participants deposit money, called a stake, within the system which gets locked in case of misbehaviour. The more money a participant stakes, the more likely he can sign the next block. After a correct signing of the block, the participant will get a reward for taking the risk of depositing his money and keeping the network up and running with his participation. [But17a] [YMRS18]

Proof of Authority The proof of authority consensus is based on a blockchain setup in which participants hand over their permission for validating blocks to a small group of trusted authorities. The most simple implementation of proof of authority will then select a node for being the node which adds the next block to the chain. This can happen, for example, in a Round Robin scheduling [YMRS18]. One disadvantage of such an implementation is that the new block will not be verified by other parties. Therefore, a malicious node would be able to chain blocks with illegitimate transactions. To prevent this a modification can be added in which finality is only reached once a number of validators proofed, that the last added block is legitimate and validators only will add blocks to the last proven block [HHH+18]. Proof of authority has the advantage, that its upkeep is less expensive and its transaction throughput is increased [YMRS18].

Practical Byzantine Fault Tolerance Practical Byzantine fault tolerance describes an application of the Byzantine Generals problem and its solution as an algorithm for achieving consensus with possible Byzantine nodes. M. Castro and B. Liskov propose a Byzantine fault-tolerant algorithm which finds consensus for a group of nodes in which a maximum number f of nodes can be Byzantine with $f = (n-1)/3$ where n is the number of nodes in thy system. The general idea is, that a current primary node waits for the same messages of $f + 1$ nodes. Because at least one of the nodes sending the messages is not faulty, the given message is the correct result. Practical Byzantine fault tolerance is a very efficient way of finding consensus but has the drawback of only supporting $(n-1)/3$ Byzantine nodes. [CL02]

Permission and Access Restrictions

Blockchain networks can either be public and for everyone available or restricted for a selected group of participants. While in public or permissionless networks every participant can read and write onto the blockchain, in permissioned networks theses access rights are restricted [YMRS18]. Permissioning can be implemented by setting up the network in a private network or establish a token which needs to be paid in order to write on the blockchain. The token then only resides in the wallets of participants allowed to write onto the chain. This results in a network in which everyone can read all the transactions but only a defined group of people can write into it. In some cases, permissioned blockchains are also called private blockchains when access rights are restricted to a closed group of participants which is completely isolated to the outside.

Central Regulation

Although the blockchain protocol tries to be as decentralised as possible several implementations still come with centralised regulations. Such central aspects can be the consensus algorithm, the transparency of development, the location or ownership of the nodes which host the blockchain and the governance policies defined by responsible parties. While some of these aspects might be acceptable in some use cases in order to achieve other advantages, having a system with many central regulations challenges the purpose of using a blockchain protocol for ones use case implementation.

Immutability

Immutability classifies blockchain technologies into the difficulty of tampering the stored data. If data stored on the blockchain is easy to tamper the blockchain has small immutability characteristics. [ZXD+17]

Identity

Identity classifies blockchain protocols into how they treat identities of the participants. Blockchain protocols can treat participants as uniquely identifiable entities, as pseudonym entities or as anonymous entity. Blockchain technologies do not have to tie themselves on one principle and rather could provide a solution for each of the possibilities. [ZXD+17]

Efficiency

Efficiency describes the latency and throughput of transaction validation and their propagation through the network. Efficiency can suffer from a large network or a slow consensus algorithm. [ZXD+17]

2.2.2 State Machine and Smart Contracts

It is the most principal feature of a distributed ledger (or blockchain) protocol to distribute state changes over the network. Such state changes are propagated in form of transactions and describe the necessary process from getting an application from its present state to the newly distributed state. Such an application is called *state machine*. Whereas some blockchains only implement a single state machine the Ethereum platform encourages participants to implement their own state machines within the platform. Therefore a concept called *smart contracts* has been developed. Smart contracts are arbitrary code statements which are executed within a transaction [Woo18]. While execution those code statements can alter the state of the contract itself or even issue new transactions. Because validators, when validating blocks, have to execute all these statements as well, blockchain protocols define a limited amount of computation allowed within a transaction. In Ethereum this amount is called *gasLimit*, where *gas* is the unit of computational measurement [Woo18].

Token Contract

The possible most famous smart contract is the Ethereum ERC20 token standard [VB15]. With the ER20 token standard developers can create arbitrary cryptocurrency, called tokens, within an existing blockchain system. This gives participants the possibility not only to use the primary cryptocurrency of a blockchain platform but also to create their own digital asset in the form of a token. Because the token is implemented in a standardised way other application developers can re-use any token for their needs. The ERC20 token standard describes a mandatory interface which has to be implemented. Listing 2.1 shows this interface. [Wik17]

Listing 2.1 Interface for the ERC20 token [Wik17]

```
contract ERC20Interface {
    function totalSupply() public view returns (uint);
    function balanceOf(address tokenOwner) public view returns (uint balance);
    function allowance(address tokenOwner, address spender) public view
        returns (uint remaining);
    function transfer(address to, uint tokens) public returns (bool success);
    function approve(address spender, uint tokens) public returns (bool success);
    function transferFrom(address from, address to, uint tokens) public
        returns (bool success);
    event Transfer(address indexed from, address indexed to, uint tokens);
    event Approval(address indexed tokenOwner, address indexed spender, uint tokens);
}
```

The *totalSupply* function describes the number of available tokens. The *balanceOf* function returns the amount of tokens a given address owns. With the allowance function, a spender address can check how many tokens the address is allowed to withdraw from the given owner address. With the *approve* function a spender can withdraw tokens. To send tokens from the contracts supply to an address the *transfer* function is used. Furthermore, to send tokens from a specific owner address to another address the *transferFrom* function has to be used. This should be only possible if the callee of the function has the correct access rights to send tokens from one account to another. One trivial access right is, that the caller is the owner of the from account address. Additionally, the transfer functions and the approval functions issue *events* on the blockchain which can be caught by other participants. [Wik17] [VB15]

2.2.3 Light Node and Full Node

While a full node contains a full copy of the blockchain and is able to validate new transactions, the goal of a light node is to store and compute as less as possible and still be able to trustless communicate with the blockchain [YMRS18]. Because most light node protocols retrieve their data from full nodes one disadvantage of light nodes is, that a light node always has to be connected to the network in order to read blockchain data. Therefore a network cannot only exist out of light nodes and light nodes need a sufficient way of proving, that the data they retrieved is the correct blockchain data.

2.3 Decentralised Architecture Stack

Anonymous e-cash protocols are already well known for several times and have been invented in the early 1980 and 1990 [But13]. The first concept of creating money by solving a computational heavy task and therefore add value to the system was first introduced by W. Dai [Dai98]. Nevertheless, all those solutions lack a decentralised consensus and are dependent from single centralised parties. Although there have been decentralised consensus mechanisms proposed those solutions assumed a fix and known number of participants within the network [But13]. S. Nakamoto [Nak08] then in 2008 introduced a decentralised peer-to-peer cash-system - Bitcoin. The main advantage of

Bitcoin is, that its decentralised consensus algorithm is not limited for a fixed number of possible Byzantine participants, but for a number of Byzantine nodes [But13]. Because gaining the power of the majority of nodes of a network is much harder, then just simulating multiple participants, the proposed consensus mechanism was suitable for the public. With the concepts of Bitcoin as its fundamentals V. Buterin [But13] then in 2014 introduced Ethereum as a platform providing consensus for arbitrary decentralised applications [But13].

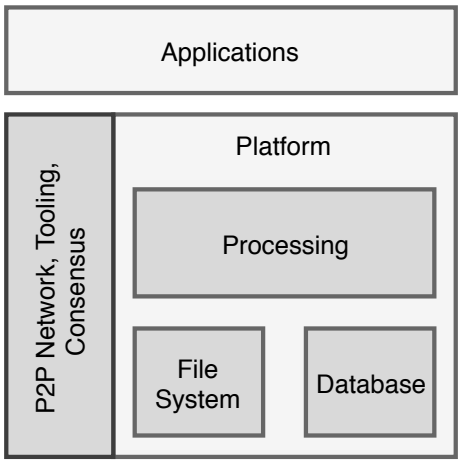


Figure 2.5: Components which need to be decentralised for a fully decentralised technology stack [Big18]

Since the development of Bitcoin and Ethereum, more and more concepts for a decentralised ecosystem have emerged. The developers of BigchainDB [Big18] describe the evolution of a traditional centralised cloud application, e.g. on Amazon Web Services, to a fully-decentralised technology stack [Big18]. This technology stack consists of a decentralised platform such as the Ethereum platform. Within this platform, a computing component, a file system and a database component are located. Figure 2.5 shows this technology stack. Following the base concepts for decentralising, each component is explained.

Decentralised Platform A decentralised platform should take care of network capabilities, tooling and consensus mechanisms for shared states. It can be seen as the base component on which all other components are built upon. In a centralised technology stack, a cloud provider such as Amazon Web Services could be the central platform. In contrast, Ethereum could be the platform for a decentralised technology stack.

Decentralised Processing While in central technology stacks participants trust the calculations of one particular machine a decentralised technology stack tries to avoid this trust issue. The concept of decentralised processing tries to transparently and trustless compute arbitrary application logic. In order to decentralise computation, challenges like the incentivation of the participating machines, the verification of computed results and the deterministic computability of the application logic have to be solved.

Decentralised File Storage A decentralised file storage is a distributed file system based on a single peer-to-peer network. It is the goal to distribute files redundant not over some single entities in a network but rather over each peer. This shall be done without any single point of failure and any trust in any third party [Ben18]. To do so every node of the network stores files or copies of other files voluntarily or incentivised. Other nodes can access these files by calling their unique address, mostly computed by the Merkle hash of the file. Using unique Merkle hashes not only makes files identifiable but also verifiable for the receiver whether he got the correct files from his requested Merkle hash. Furthermore, no file can exist twice with two different identifiers. This makes redundancy even simpler to establish. [Ben18]
Today decentralised file storage mostly differ to past solutions with the provisioning of an incentive for storing and distributing files.

Decentralised Database Blockchain protocols itself suffer from scalability issues when it comes to transaction throughput. Therefore storing all data on a blockchain is not the best idea. Consequently, the means to store large structured data outside of the blockchain and still be able to integrate with the underlying platform is needed. This is where decentralised database systems are used. [Big18]

Decentralised Applications - DApp Applications can be built on top of the decentralised technology stack. Those applications are called decentralised application (DApp). A DApp utilises all underlying technologies in order to build an application which combines all principles of the decentralised ecosystem. A typical DApp is open source, monetises itself from the platform cryptocurrency, achieves decentralised consensus and has no central point of failure [Rav16].

2.4 Energy Management System

Having enough knowledge about *energy management* is important in order to find an architecture which is derived from requirements and special needs and not from the pressure the blockchain hype puts on possible use cases. Energy management means to systematically produce, convert, distribute and use energy in a given environment, for example, companies, municipals or households. The key goals of energy management are to fulfil all needs of the involved participants with the respect to ecological and economic conditions [Ver18]. With the introduction of the ISO 50001 [DIN18] norm in Europe reducing energy consumption got even one of the main tasks a participant should fulfil within his energy management [KKK+12]. This means an established energy management helps in saving energy, reducing CO_2 emission and supports the use of renewable energies [Mic13].

To fulfil this policy several different energy strategies can be conducted [Kal10]:

- Having a *passive strategy* means just doing the most necessary things to achieve basic management policies which are mostly enforced by law. This strategy should not be used if other strategies can be applied, as well.
- Enforcing a *strategy of short reward* comes with decisions which can be implemented in a short range of time. The energy management strategy is not planned in long terms and only based on the current state of the energy household.
- In contrast to short-term decisions the *strategy of long-term rewards* focuses on including all possible factors into the strategy. Possible measures can affect the established policies over a long period of time.
- A third strategy could be to *offensively enforce* energy management policies. This means to implement energy measures although it might not maximise the revenue of a company or actor.
- The *maximum strategy* focuses on establishing a strategy to actively contribute to climate protection rather than just save energy and money.

Furthermore, Kals [Kal10] describes four different layers in energy management. First the global layer, second the layer of national economies, third the layer of several industrial or non-industrial sectors and fourth the layer of single companies [Kal10]. Although microgrids in households or municipals are not considered in this classification theses parties can be found in the fourth layer as well. Figure 2.6 shows the arrangement of the several layers. Energy management has a wide range of usage areas the focus during this work lays in the area of energy management in power supply, especially for the fourth layer. The fourth layer is most considerable for energy management because companies, households and municipals can group up within their microgrid and benefit from each other.

In order to find a systematic way of implementing energy management policies an *energy management system (EMS)*, is needed. Such a system takes responsibility of collecting data of energy flows and tries to fulfil the given needs based on the collected data [KKK+12]. A good EMS should always ease the process from just having a passive energy strategy onto a long-term or maximum strategy. One important key point hereby is the automation of the whole process, which reduces complexity, workload and costs for the establishment of an EMS.

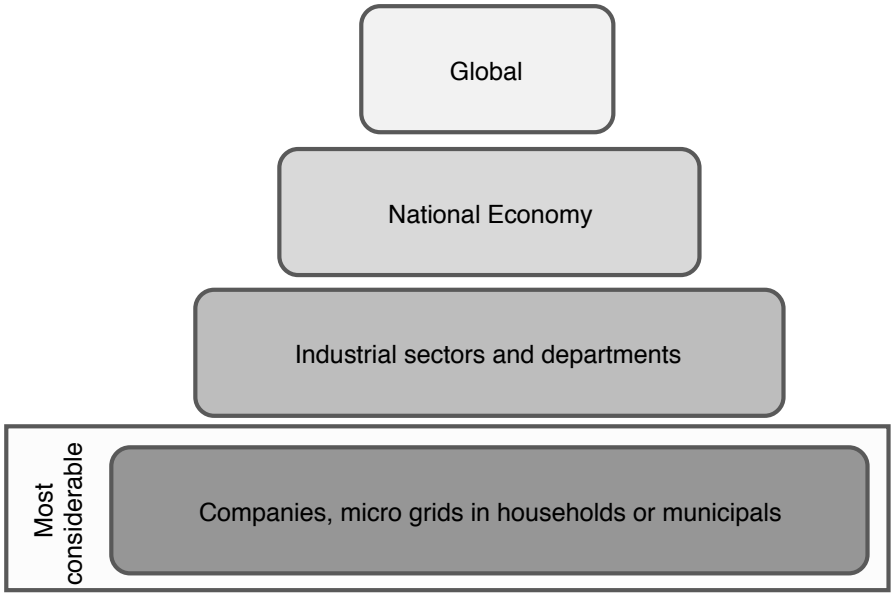


Figure 2.6: Layer of energy management(after [Kal10])

2.4.1 Smart Grids

In order to balance power consumption, production and supply and benefit from each other, several actors can form a grid, which tries to implement energy management policies in a smart way [DKE10]. Such a smart grid can have several different stakeholders. The first major stakeholders are the owners of a *household*. A household either takes part as a single actor in the grid or groups up with multiple other households. Households can either produce or consume energy, or do both. Such actors are called prosumers [RJ10]. It is the main interest of a household to gain mostly an economic advantage from its participation in a smart grid with a possible renewable energy mix. Another stakeholder are companies. *Companies* are large consumers who have mostly a static high energy intake. Although some companies also have their own power plants and try to feed their surplus of produced energy into the grid whenever possible. Companies not only have an economic motivation but also want to achieve energy management policies which could be reflected positively by their customers. A good example is Google, who only purchase energy from renewable sources in order to increase customer satisfaction and awareness [Pyp18]. A third stakeholder are *municipals*. Municipals have mostly governmental concerns and try to promote their idea of how a smart grid should be managed and setup. Nevertheless, they also could have a municipal power plant which produces energy for the grid. A traditional stakeholder are conventional *power plants* such as coal or atomic plants but also big renewable power plants. Those actors have the goal to sell their energy for the best price as possible also beyond the borders of the smart grid. Finally, there are multiple *responsibility parties* within the smart grid such as the grid operator or stakeholders responsible for energy balancing. Additionally, there are application providers who develop applications in order to make the grid smarter.

Smart Grids have the needs for an EMS which helps to achieve all the different concerns and energy management policies of the stakeholders. Because most of the actors are volatile producers, consumers or prosumers they especially can profit from interchanging their assets with the help of

an EMS. All of the actors are connected with each other over the grid. Each actor has his own energy management policies. While trying to fulfil these policies the actor benefits from the interconnected exchange over the grid.

2.5 Development of Embedded Devices with Yocto

In order to make power grids smart, they rely on the interconnection of many lightweight embedded devices. Embedded devices are computational weak devices, which are cheap in production and power consumption. Because of the limited technological specifications and its mostly not common hardware architecture, the operating system of such embedded devices needs to be adapted to its special needs. This can be done with the Yocto development environment.

The Yocto Project[1] is an open-source project which helps building own custom Linux based operating systems especially for lightweight embedded devices. It is based on the OpenEmbedded[2] project and BitBake [PLB14] task execution engine.

2.5.1 Cross-compilation with OpenEmbedded and Bitbake

OpenEmbedded is a framework for cross-compiling software packages or even complete Linux kernels. OpenEmbedded also comes with an open and extendible library of build scripts, so-called recipes, written for cross-compiling packages. Cross-compilation is necessary because native compilers only compile source code for the system they reside on. If one wants to compile source code for a system which is different then the system on which the code is being compiled, a cross-compiler is needed. Situations like the one explained appear when one wants to compile code for systems which are not capable of running a compiler, due to hardware limitations or if the system does not exist, yet. This is especially the case when compiling a Linux kernel for a custom embedded device. The recipes provided with the OpenEmbedded framework provide a detailed machine-readable explanation on how to cross-compile a given source code for a preconfigured device type. Such a recipe defines the source locations, their checksums, license and the instructions for building it. The Bitbake task execution engine then is able to execute the recipe. As there are many software projects naturally there are many recipes describing their cross-compilation. Therefore the OpenEmbedded framework is divided into multiple layers, which can also depend on each other. Each layer is combining recipes of the same context. [Rif15]
Figure 2.7 shows an example of possible layers in the OpenEmbedded Framework and their hierarchy. The Yocto Layer and the OE-Core Layer are the two basic layers on which most of the other layers depend. Additional Layers can be Hardware Enablement Layers, Kernel Enablement Layers, Software Layers and Custom Layers. [Fla18]

[1] Yocto Project: https://www.yoctoproject.org/
[2] OpenEmbedded project: http://www.openembedded.org/wiki/Main_Page

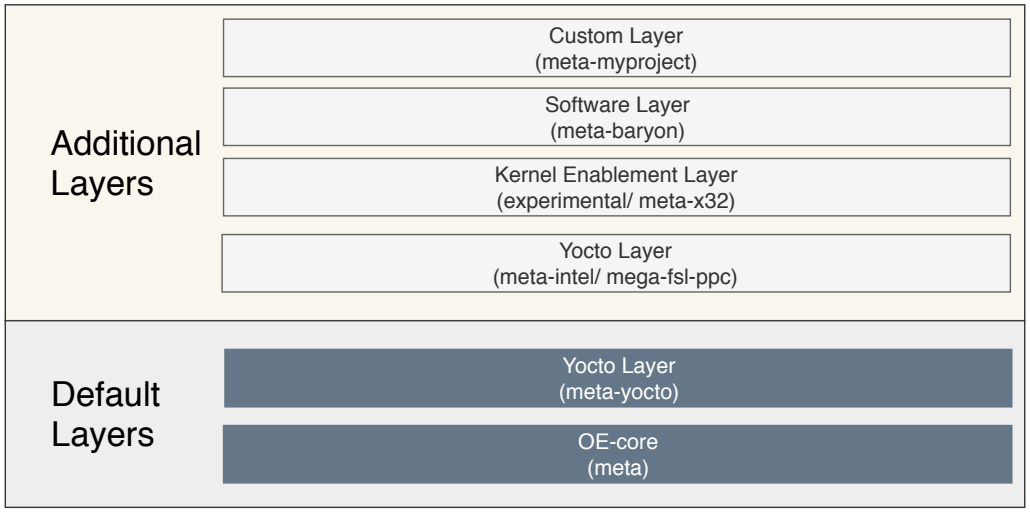


Figure 2.7: Possible Layers in the OpenEmbedded framework [Fla18].

2.5.2 Yocto Development Environment

On top of OpenEmbedded and the BitBake engine, the Yocto Development Environment [Rif15] is built. The Yocto Development Environment describes a process from configuring and describing the build process until the final operating system image. Figure 2.8 shows this process. First, shown on the left side of the figure, the developer starts with configuring the build process. In order to run a build user-specific configuration, such as the underlying architecture of the operating system where Yocto is executed, has to be defined. Second, a Meta configuration is defined. This configuration defines all Bitbake recipes and layers included in the target system. The third task specifies board specific configurations such as the architecture of the target system. With the fourth configuration task, the build can be configured further and be adapted further such as the configuration of the system initialisation, a device manager or additional libraries. Afterwards, this configuration is supplied into the build process. The build process starts with fetching the sources from various locations such as upstream sources, local files or source code management tools. Based on the configuration patches can be applied to the fetched source code before the Bitbake recipe is executed and the code cross-compiled. Following the resulting binaries are packed in the desired package formats and a quality analysis is run over the build results to ensure everything is built according to the build configuration. Before the built packages are packed into a deployable operating system image all packages are copied to a central destination. This destination is called the package feeds. Additionally, to an operating system image, the Yocto build process also creates a software development kit (SDK), which includes cross-compilers and tools supporting the development of Bitbake recipes for the configured target machine. [Rif15]

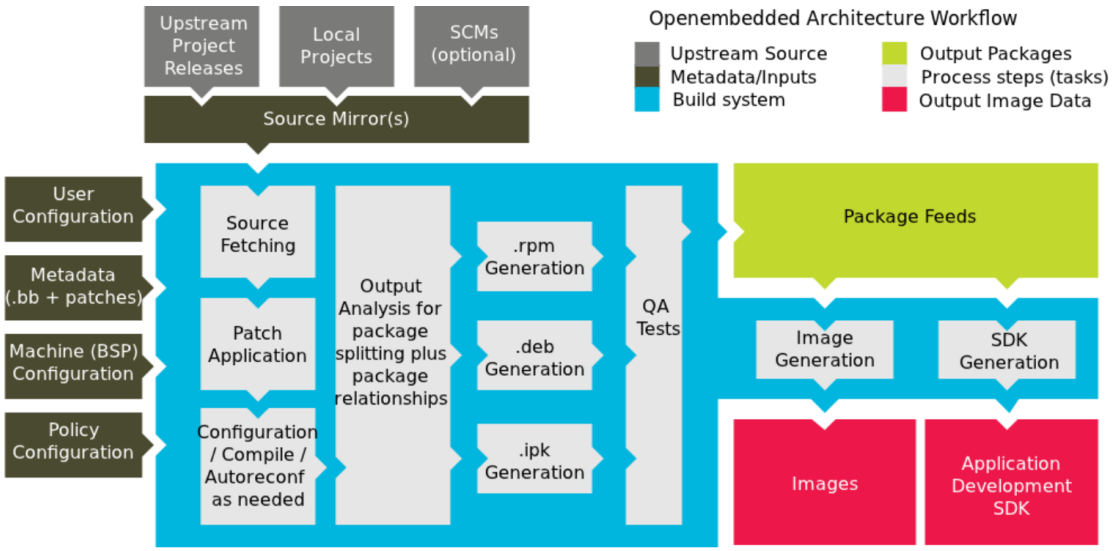


Figure 2.8: Yocto Project Development Environment [Lin15]

3 Related Work

There are two previous works related to this thesis. First a generic architectural proposal for enterprise use cases, by the Enterprise Ethereum Alliance (EEA) [BCNN18]. Second a blockchain based application in a local area grid between households in order to efficiently use shared resources without the involvement of external dependencies [SAG18]. Even though there are several already existing solutions using blockchain technology for solving energy-related use cases those solutions are part of the architectural analysis and will be discussed in chapter 5.

3.1 Enterprise Ethereum Client Specification

The EEA [BCNN18] defines the Ethereum Client Specification (ECS) as a protocol stack for an enterprise utilisation of the Ethereum platform. Several concepts, which have emerged since the development start of Ethereum, are explained and concrete protocol implementations are proposed. The results of the specification aim to be a recommendation for an Ethereum client which supports advanced scalability, security and privacy requirements required for enterprise use cases. Figure 3.1 shows the proposed protocol stack. The stack consists out of a *Network layer*, a *Core Blockchain layer*, a *Privacy and Scaling layer* and a *Tooling layer*. Each layer is divided into the already existent Public Ethereum protocols, protocols proposed in the Ethereum Yellow Paper [Woo18] and additional protocols required for being a suitable Enterprise Ethereum client. [BCNN18]

For the Network layer, to be suitable for enterprises, the EEA proposes an enterprise peer-to-peer network which can be restricted for a group of participants. The Core Blockchain layer adds concepts such as private state stored persistently on the blockchain, execution of computational logic only by trusted parties and the possibility of choosing other consensus algorithms than the proof of work consensus. Within the Privacy and Scaling layer, the EEA proposes additional concepts for executing transactions aside the blockchain protocol but still being trusted by other participants. For being able to not only store private state on the chain but also alter state privately the concept of private transactions is proposed. On the Tooling layer concepts like permissioning and authentication of participants, the deployment of enterprise management systems and oracles are introduced. Oracles are a concept for smart contracts to use data from outside of the consensus mechanisms of the blockchain without relying on a single entity sending the data to the smart contract. Additionally, the Application layer resides on top of the layers, which consists of tools, specifications and already available decentralised applications to support application development. [BCNN18]

The EEA mainly proposes requirements and ideas for future protocol specifications. Therefore, those recommendations still have to be defined and implemented. Nevertheless, developers realising the concept of the EEA all implement identical sets of interfaces and concept ideas, which makes the Ethereum Client Specification an important artefact in the heterogeneous blockchain ecosystem.

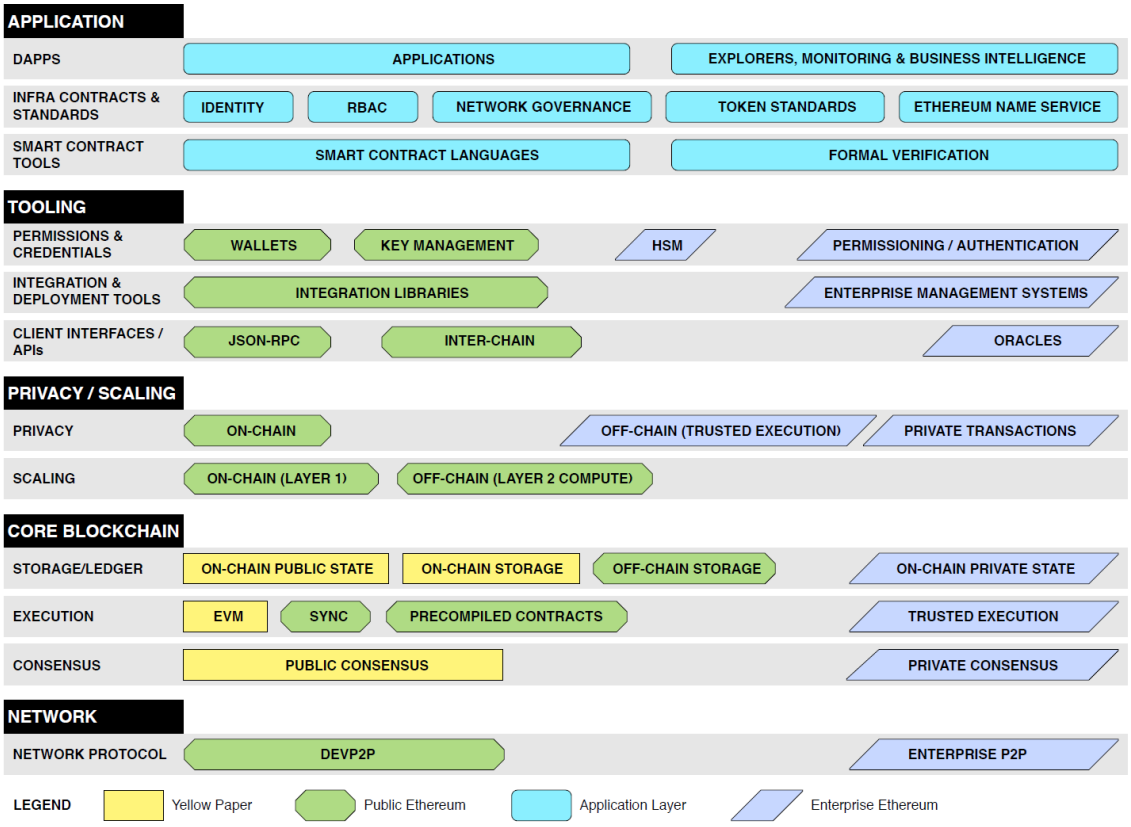


Figure 3.1: Enterprise Ethereum Architecture Stack [BCNN18]

3.2 ETHome: Open-source Blockchain Based Energy Community Vontroller

J. Schlund et al. [SAG18] describe a possible blockchain based management application for a local energy community. It is the idea to establish communication and exchange of assets directly between the households of the community instead of a traditional top-down approach. The authors hope to thereby increase efficiency which will lead to cheaper energy costs for consumers and larger profit margins for producers. In the proposed implementation the local energy community consists of households with a photovoltaic rooftop supply, electricity consuming devices, a battery and a Raspberry Pi[1] which runs a full node of a custom Ethereum blockchain and provides the distributed energy management application. Each household has its own grid connection. Figure 3.2 shows the topology of the concept. [SAG18]

For the implementation Schlund et al. [SAG18] provide a custom proof of authority Ethereum instance. All Raspberry Pies host an Ethereum full node or light node, an execute the proof of authority consensus algorithm. A smart contract deployed on the chain organises the production and consumption loads of the households. The application itself is written in Python. With the

[1] A Raspberry Pi is a small cost and power efficient device with minimal computing and storage capacities. https://www.raspberrypi.org/

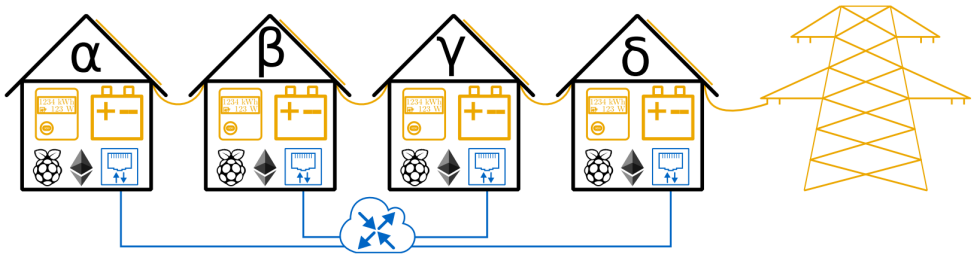


Figure 3.2: Ethome concept of interconnection of metering, consumption and production devices and the Ethereum client running on a Raspberry Pi [SAG18].

proposed concept the authors could increase the self-sufficiency of each household from 51,10% to 59.20% and the efficiency of the whole local energy community from 71.86% to 75.56%. The presented proof of concept reflects the changing energy ecosystem and does not need any central entity to be provided. [SAG18]

The Ethereum Enterprise Alliance does not focus on energy-related use cases and a generic utilisation of decentralised concepts, whereas Schlund et al. focus only on utilising the Ethereum blockchain core protocol and do not spend interest into other decentralised concepts. Therefore, an architectural analysis for energy management related use cases, which focuses on the utilisation of decentralised concepts, rather than only the core blockchain protocol, and gives a conceptual approach which may not only be implemented by the Ethereum protocol stack is needed.

4 Decentralised Energy Management System

To gain an architecture decision which is driven by the requirements of an energy management system it is important to examine the current state of such systems and identify their essential use cases, features and requirements. Due to the energy transition in Germany, the grid not only consists of several large producers but also of many small producing parties and prosumer, which act as consumer as well as producer. Because most of the new producers obtain their energy from renewable sources, the energy system will get more and more dependent on environmental conditions such as the weather. Despite having an overall constant level for power prices, changes in weather lead to highly varying prices for certain time periods [MSN+16]. The search for the current best power prices results in an economic dispatch problem, which needs to be solved [EHS15] [HAS14]. Our current energy system is driven by a relation between grid operators and customers [HHH+18]. Having many small producing parties is not suitable in the current grid architecture. Those parties would have to be aggregated into a single producer which then can be handled by the grid operator, but loose in flexibility and efficiency [HHH+18]. Therefore it is necessary for an EMS gather information from all included parties about their energy production, consumption and conception values. Every single grid member needs to know these values, or at least the resulting power price in order to map electricity flows, reduce energy costs, balance load inconsistencies and share their assets with other members.

While doing this with a central instance this instance would lead to high additional costs and needs to be maintained by a single provider. Furthermore, involved parties would not be able to enforce their own policies on top of the system. An example could be, choosing energy sources with respect to environmental conditions. This results in the need of a decentralised energy management system (DEMS) which aims to reduce additional costs and can be maintained by every participant. In conclusion, a DEMS is needed, which can automate all the discussed tasks without including an inefficient, costly and to the underlying grid architecture inappropriate central entity.

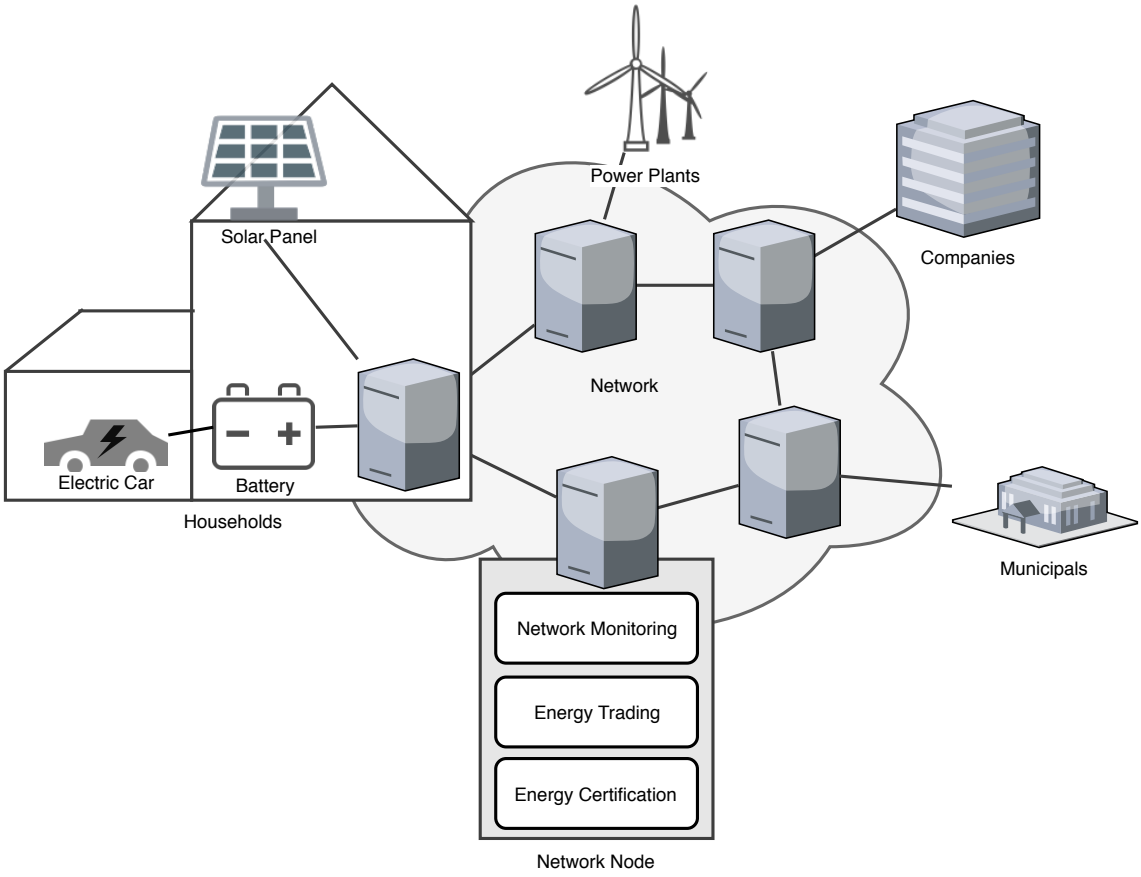


Figure 4.1: Smart Grid overview

4.1 Requirement Implications

Establishing an energy management system in a decentralised grid of different actors results in several requirement implications for such a system. The following section describes the functional requirements (FRs) and non-functional requirements (NFRs) this system comes with.

4.1.1 Functional Requirements

Figure 4.1 shows how the resulting energy management system could look like. The example shows, that the power grid could consist of several households, possibly prosumers, a central power plant, a big consumer such as a company and some municipal buildings. An underlying DEMS should be capable of:

1. Connecting all participants with each other in a network
2. Collecting and storing all necessary energy data (focus on power consumption) in a consistent and transparent way
3. Providing an extendible and open platform for implementing individual energy management use cases such as network monitoring, energy trading or certification

FR1: Connecting participants

In order to share assets as power consumption or production, the interacting participants have to be connected with each other. In conventional grids with a small number of big producers and no nodes acting as prosumers, participants had to collaborate with a small number of nodes. With every single household possibly acting as a producer and feeding the grid, consumers now have to connect with many nodes at once. As a participant who wants to only gather power from renewable sources or react to specific demand-side changes in the grid, this participant now has to monitor the energy production and consumption of many different nodes. This also means having the possibility of interacting with connected devices in order to reduce their energy consumption or production in favour of the needs of the participants. Thus, the DEMS has to give everyone the possibility to collaborate with others, to achieve their defined energy management policies.

FR2: Data Storage and Collection

Dependent on the energy management policies of the participants it is the need to store and collect data fed into the grid. Having this data gives the possibility of monitoring the system, forecasting future behaviour or sharing the data as financial assets. The energy management system has to provide a way to store this data in a transparent and consistent way. Only if every participant has unrestricted access to all the data he needs, it is possible to adapt his own energy management strategy based on the behaviour of the grid. Due to the energy transition, the grid gets more and more heterogeneous. Therefore, the stored data has to be in a standardised and for everyone readable data format. In total, the energy management system should support the mapping of the energy data, transaction and flow of the real world on to a digital system.

FR3: Open and Extendible Platform

Actors of the EMS shall not perform energy management use cases on their own. Rather use cases which involve many participants shall be integrated into the system and extend it. This use cases shall be executed in an automated manner with less user interaction as possible. Furthermore, it should be possible for every one of the grid to take part in several provided use cases and also implement their own applications on top of the platform. The energy transition lives from small stakeholders who contribute their energy values into the grid. This should be also the case when it comes to making the grid smart and intelligent. In conclusion, the decentralised energy management system shall be an open and extensible platform on which everyone can collaborate based on its fundamental functionalities and knowledge.

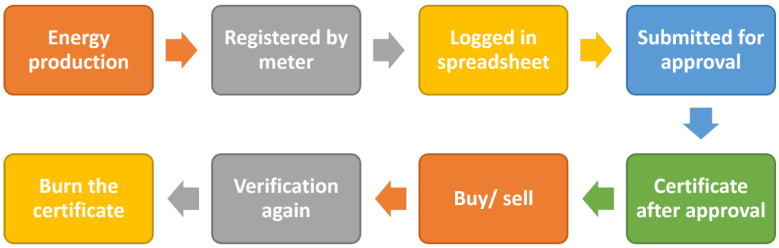


Figure 4.2: Workflow of creation, and distribution of the proof of origin certificates [Red18] after [Yah18]

Example Use Case - Monitoring and Certification One possible use case, implemented on such an open and extendible platform could be the monitoring and certification of power origins and their destinations. Although there are many different sources of power production, the electricity delivered at the consumer site always stays the same. During the energy transition in Germany, consumers actively want to control the source of their power consumption. Thus a mechanism for proving the origin of the obtained power is needed. This is enabled by monitoring all power production and their origin certification. Certificates can be created by the power producer and bought by the power distributors. Power distributors can now give a certificate to one of their consumers in order to prove that the supplied energy comes from certain sources, such as renewable sources. Figure 4.2 shows this process. [Umw17]
Currently, the creation storage and distribution of these certificates is a centralised and inefficient process.

In order to eliminate this disadvantages, J. Schmid [Sch18] proposes a concept for realising origin certification with the aid of a blockchain protocol [Sch18]. Furthermore, M. Jahja [Yah18] proposes a DApp which monitors production and consumption values of participants and provides a means for visualising the origin of produced energy assets [Yah18]. Additionally, certificates in the form of tokens can be issued for each produced energy unit. Those tokens can then be traded with consumers who then can certify, that they only purchased electricity from a certain source [Sch18]. For now, there is no suitable platform which provides things like a decentralised storage mechanism, an established peer to peer network, trusted clients or integrated stakeholder concerns. Running this use case on the proposed energy management system would solve these missing requirements and therefore make it to a possible reference system for such use cases.

4.1.2 Non-functional Requirements

The functional requirements of the proposed energy management system come with several non-functional requirements. The following section describes the most important ones.

NFR1: Decentralisation

As mentioned at the beginning of the chapter, several aspects lead to a decentralised distribution of the grid. This characteristic should also be reflected in the overlaying energy management system. Thus, no single trusted party which could result as a bottleneck shall operate and maintain

the system and no single point of failure shall be possible. This leads to a fault-tolerant network of empowered actors which communicate with each other overview third parties as possible. In addition, decentralisation shall also be reflected in the governance and development of the system. This means giving key insights into the system and preventing the development team from becoming a central component by publishing all development artefacts.

NFR2: Availability

In general, the system has to be available every time participants want to benefit from the provided platform or contribute to it. A temporarily unavailable system would lead to possible inconsistencies in representing the data of the real world with the aid of the energy management system. Especially prosumers could use the system to rapidly check for the conditions to efficiently produce, consume or store energy for later. With many participants doing this in parallel the system has to be available all the time. Furthermore, if one actor might be disconnected from the grid, due to connectivity issues, the grid has to provide the functionality of supplying the data the actor has missed. Thus, all in all, the system has to be 24/7 available plus providing compensation activities for disconnected nodes.

NFR3: Scalability

There are two scalability issues, which should be fulfilled by the DEMS. First, scalability in terms of the number of participants and second, scalability in terms of data storage. To optimise ones own energy exchange, one needs to connect with multiple other participants. Every actor who consumes produces or stores energy should be part of the grid and the platform itself. By using the EMS only in small grids such as municipals, scalability issues might not be that important. Still, it should be possible to extend the platform over larger areas in order to give everyone the possibility to benefit from it. Furthermore, the system shall be capable of storing a large amount of data.

NFR4: Performance

Currently, responsible parties of the participants and the grid itself only get energy data from the other participants in large intervals which can size up to once in a year reports [Fis18]. The system should remove this latency by providing the possibility to interchange data at least all 15 minutes. In general, this means reading the dataset as often as possible and writing at least every 15 minutes. This requires a high transaction throughput which rises depending on the number of participants.

NFR5: Data Protection, Security and Privacy

The system should support the possibility of handling secured and private communication between participants. Communication which is done between view members of the grid and not changing the data state of global applications do not necessarily require public and transparent communication flows. Therefore supporting encrypted communication opens a wide range for applications and enables the EMS to conform with possible data policies of the local government.

NFR6: Data Persistence

Once data packets are sent and verified, these packets should be persisted within the system. Thus, a shut-down of the actor which sent the data should never lead to data loss. Furthermore, being disconnected from the network does not mean not having any access to maybe temporary available data. This means the verified data shall be persisted on the device of the sending actor, as well as on the system as a whole.

NFR7: Auditability

Some applications require the possibility of tracing back communication flows transparently. Thus, communication between participants shall be traceable whenever possible. In the example of the explained use case in requirement FR3, power consumers shall be able to trace back who owned the certificate before and who issued it. Therefore they can be sure of purchasing power from a specific source or are able to blame misbehaviour of the issuer.

NFR8: Hardware Requirements

In order to reduce power usage and production costs, the current environment consists of lightweight embedded devices which act as electricity meters, gateways or communication devices. Currently, there is no stable infrastructure in which these devices are installed. Thus connectivity availability and bandwidth can highly differ. Furthermore, the devices have to be stable against load peeks in power supply such as lightning strikes or power grid imbalances and passive weather effects like hot or cold temperatures. Because the devices are exposed to actors who could have an interest in exposing it, all clients have to be resilient against possible manipulation threads.

NFR9: Self-Sustainability

Self-sustainability of the system means, that the system shall not rely on single parties maintaining and paying the upkeep costs of the system. The system rather should generate a common interest for all participants to keep it up and running. If participants invest economical or mental costs into the system, they shall be rewarded by the other participants benefit through them. Such an incentive system prevents the system from being operated by just a single entity.

4.2 Suitability of the Blockchain Technology for a DEMS

This work aims at analysing existing or equivalent blockchain solutions for a decentralised energy management system. In order to constrain the analysis on blockchain based solutions, it is important to show that the concept of blockchain technology is the best way to use it as an underlying technology. Projects based on blockchain should not be driven by the technology itself and rather by the requirements of the project which fit the technology. This means, that the projects needs can only be solved best by the advantages of the blockchain technology and not only because of the need for another blockchain use case.

The two major parts in investigating the suitability of a blockchain technology should be to decide whether a central solution with a normal database would be enough. If there is at least one evidence, that this is true, one should think about using existing centralised database technologies instead of spending a large number of resources to drive a blockchain use case which might not fit the underlying requirements. For answering this question, Greenspan [Gre15] identified eight conditions which need to be fulfilled in order to be a use case which fits the underlying blockchain technology [Gre15]:

Shared Database: The foundation of a blockchain can be seen as a database which shares its state with all participating parties. If the project does not need the overhead of a database to persist its state, a blockchain technology might not be the best choice.

Use case fulfilment: As it is the main requirement to share and persist energy data over the platform, this constraint is fulfilled by the use case.

Database has Multiple Writers: The database underlying in the blockchain supports multiple writes at the same time. If the project is implementable with a single writer, solving this with a blockchain is not necessary.

Use case fulfilment: Because every participant wants to write his energy data independent from other entities into the database, multiple writers are necessary. Therefore, this condition is fulfilled.

All the Writers are Non-trusting: Blockchain technology supports writes on the database from parties which are not trusting each other at all. If the project only consists of parties which trust each other one could use a regular database, as well.

Use case fulfilment: One participant of the platform cannot trust any others participants to correctly execute his scheduled transaction. The energy system relies on many single parties trying to interchange values with the highest reward as possible. Furthermore, participants are interconnected with a large number of other participants they never knew before. Therefore interchanging assets with a participant one might not know and who could misbehave for his own advantage shall still be possible. Consequently, the condition is fulfilled and a regular database is no possible option.

Intermediaries are no Possible Option: The support for the absence of trust can be solved by introducing a trusted intermediary. The key benefit of blockchain technology is generating trust without introducing such an intermediary. This makes it possible to write onto the database directly, mostly with the trade-off of performance and scalability issues in data propagation and finality. If the project implementation is fine with the introduction of an intermediary these trade-offs should not be considered by using a blockchain technology.

Use case fulfilment: The digitisation of power flows shall not be managed by an intermediary party. Single administrative parties fall into the risk of taking advantage from their monopoly role and might be a single point of failure within the system. Furthermore, adding intermediaries for a small number of participants just to enable the interchange of assets between them is not suitable and only creates high administrative costs. Additionally, some countries rely on regulations where power consumption and production are not allowed to be controlled by a single entity anymore. Therefore a system without the presence of a single intermediary is needed.

Transactions Depend on each Other: A blockchain supports transactions which depend on other transactions or on the current state. If the project does not have interacting transactions, multiple databases instead of one shared database might be better.

Use case fulfilment: In order to react to data changes in the database, or transfer energy-related assets from one party to another, transactions have to be dependent on each other. Although this can be done with multiple databases, a decentralised solution in which data responsibilities are not out-sourced to single databases solves this requirement more adequate.

Rules for Transaction Execution: In order to make multiple untrusted writers and dependent transactions possible, several rules have to be set up. These rules have to define constraints like who is validating the transactions, what are the preconditions and postconditions of a transaction or how are simultaneous modifications on the same data set treated. If the use case is to complex to define such rules onto it might be better to use a traditional database which does not need such constraints.

Use case fulfilment: In general, the digitisation of power flows can be represented by the trade of digital assets. Transacting digital assets from one participant to another is a known problem and has been defined and solved, for example in the banking sector, several times. Additionally, many countries have already defined and regulated the transaction of energy assets within their borders. Therefore modelling power flows in a digital system is easy and a predestined use case for decentralised concepts such as the blockchain.

Authoritative Final Transaction Log is Needed: The blockchain technology provides a final transaction log which makes it possible to determine the current state from the first transaction on with the prevention of transaction conflicts. Furthermore, nodes can be unavailable for a certain time. To carry such a transaction log is highly costly and should only be done if the overlaying use case requires it.

Use case fulfilment: Many parties are located in rural areas with bad internet connectivity. Furthermore, it is not enough to just share the current data state. Participants have to be able to retrace the past transaction history and verify that every interchanged asset was transferred correctly. Therefore an authoritative final transaction log is needed.

Real World Representation of Assets: The last condition says something about asset-based blockchain projects. If the blockchain is used to track assets. the implementation should specify the real world representation of the tracked asset. What represents the asset in the real world, who is concerned about its transactions between different parties. If this is not possible, representing an asset within a blockchain technology might not be recommended.

Use case fulfilment: In order to implement several energy management policies, it is one of the main use cases of the platform to track energy-related assets. Examples could be the produced and consumed amount of power or certification tokens for the production of renewable energies. All these assets are representable in the real world and respectively already exist.

The given platform for a decentralised energy management system fulfils all conditions which are important for using the blockchain technology. Thus, the blockchain technology is a possible technology for implementing such a platform and potentially is the most suitable one because traditional database management systems cannot fulfil the given requirements.

5 Analysis of Suitable Blockchain Architectures and Technologies

In order to make a suitable architecture proposal one has to be familiar with helpful concepts and technologies of the ecosystem in which the solution resides in and with already developed equivalent architecture solutions. The blockchain ecosystem comes with many different ideas, concepts and technologies. Section 2.3 describes the emerging decentralised technology stack with its key components. Using this technology stack can be important in order to fulfil the requirements of a decentralised energy management system. To find an architecture which fits the use case, the utilisation of technologies implementing the concepts from such a decentralised technology stack shall be analysed. In addition, already existing blockchain solutions, which claim to solve an equivalent problem as the problem of a decentralised energy management are examined. Based on the experiences gained, a generic architecture solution for a decentralised energy management system will be proposed.

5.1 Blockchain technology stack

Each of the concepts from the decentralised architecture stack is implemented by multiple technologies. The following section investigates the most important implementations of each component.

5.1.1 Platforms

Several technologies emerged lately which offer a solution for distributing a shared state over peer-to-peer networks and finding consensus. The following technologies act as an underlying platform.

Ethereum

The most popular platform is Ethereum[1]. Ethereum is a project for building a protocol implementation providing a platform for all possible transaction-based state machine concepts. It is the main concept of Ethereum to provide a decentralised value-transfer system which also enables the execution of computational logic (smart contracts) during a transaction [Woo18]. Ethereum consists of a peer-to-peer network protocol, a distributed ledger protocol and a shared-state-machine called the Ethereum Virtual Machine. While using Ethereum either the public or a private enterprise platform can be utilised. In general, there are two main protocol implementations available. The

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

Parity client and the Go-Ethereum (Geth) client. Although both can be used for the public platform, using them in a private network does not guarantee interoperability of both clients, especially when consensus algorithms other than proof of work are used.

Tendermint

Tendermint[2] is a platform offering the replication of state changes made by deterministic state machines. State changes are ordered by using a Byzantine fault-tolerant consensus algorithm, called Tendermint [BKT18]. In contrast to the Ethereum platform, the Tendermint platform does not come with its own state machine. Participants rather have to implement their own state machine and distribute its state changes over a uniform interface to the Tendermint platform. One advantage of the Tendermint consensus algorithm, promised by the developers, is the high transaction throughput of more than a thousand transactions per seconds. Tendermint is mainly for enterprise solutions and there is no public network available.

Hyperledger (Fabric & Indi)

Hyperledger[3] is a project which focuses on creating an enterprise suitable distributed ledger framework. Firstly developed by IBM, Hyperledger was given to the Linux Foundation in order to satisfy the transparency and open mindset of the blockchain environment. Hyperledger Fabric is a sub-project of the Hyperledger framework and adds the functionality of running smart contracts to its implementation. Hyperledger Fabric supports several consensus algorithms. The default one is a Byzantine fault tolerance based algorithm [Cac16]. Together with the Hyperledger Indy project, the Hyperledger framework is one of the only frameworks supporting privacy features to its full extent. Indy enables decentralised identifiers, pointers to resources outside of the ledger and zero-knowledge-proofs. Zero-knowledge-proofs are a cryptographic method which, for example, enables the verification of hidden claims, such as certificates, to a third party without having them exposed to the other party. [BSC18]

IOTA

IOTA[4] claims to be a cryptocurrency for the industry of the Internet-of-Things. In contrast to other distributed ledger protocols, IOTA is based on a tangle which is a directed acyclic graph. In order to issue a new transaction, a participant has to prove two other transactions. Therefore the sent transaction is accepted by the system after a sufficient amount of approvals. [Pop18]
It is important to mention that IOTA does not support the distribution of program sequences for deterministic state machines such as smart contracts.

[2] Tendermint platform: https://tendermint.com/
[3] Hyperledger Project: https://www.hyperledger.org/
[4] IOTA cryptocurrency: https://www.iota.org/

5.1.2 Processing

While protocols like IOTA or Tendermint only support transacting state changes from a machine outside the distributed ledger protocol, there are also concepts for executing arbitrary machine code within a transaction. Decentralised processing can be helpful especially when a participant of the decentralised energy management wants to transparently and trustless execute arbitrary application logic. In the following, technologies supporting such decentralised processing are introduced.

Ethereum Virtual Machine

One of the first concepts for establishing decentralised processing in a distributed state machine is the Ethereum Virtual Machine (EVM). The EVM allows submitting code, in form of smart contracts, which then can be executed by issuing a transaction. By using Solidity as the programming language the computational effort for each contract can be deterministically calculated. Computational effort is calculated in gas, which is an abstraction to the executed statements by the Ethereum Virtual Machine. Machines computing the smart contract are incentivised by getting a specific amount of value, the gas price, for each gas. Each block of the blockchain and its containing transactions have a limited amount of computational power, the gas limit. After a successful computation of the smart contract, the result is submitted in the transaction and written in a block with the underlying consensus mechanism. [Woo18]

TrueBit

TrueBit[5] tries to offer correct, trustless and decentralised computing while using the possibility of calculating computations aside from the blockchain. Therefore, a layer on top of the Ethereum protocol is proposed, which rewards honest participants correctly performing computational tasks. One main advantage of TrueBit is the execution of code with the WebAssembly (Wasm) engine. The Wasm engine allows the pre-compilation of many common programming languages in Wasm readable bytecode. Because of the calculation aside from the blockchain protocol, the computations executed over TrueBit are not limited to a maximum block size. [TR17]

5.1.3 Decentralised File Systems

Decentralised file systems help to distribute static content over the network in a decentralised way. Having an application which uses the blockchain to read and write data may require a graphical user interface. This could be distributed over a decentralised file system. Because one does not want to have a central entity providing all static files of an application one could make use of the decentral aspects of the energy management system.

[5] TrueBit: https://truebit.io/

Currently, two important projects proposing a protocol for a decentralised file storage exist. The Swarm[6] and the IPFS[7] project. Swarm is mainly used for distributing Ethereum static files such as decentralised application code and their data files. IPFS sees itself as the one file system replacing content distribution in the conventional web. Swarm is written with the Go programming languages and integrates nicely with the Go-Ethereum client. IPFS is a standalone implementation independent from any platform and is also written with the go programming language.

Incentive

In order to encourage the user to store copies of data with their Swarm or IPFS node incentives have to be built on top of the protocol. The basic idea behind the incentive is the most economical reward for participants storing files not being their own. IPFS is using the Filecoin[8] protocol on top of the decentralised storage layer. Filecoin is a decentralised market where participants can earn money by providing storage for clients [Pro17]. Swarm provides its own incentive protocol called the SWAP protocol [TFN+16].

Nameservice

In addition to a decentralised file storage, a nameservice can be added. With a nameservice file, distributors can register their file hashes with uniquely identifiable names. Advantages are a better discovery of files or the abstraction of identifiers and their concrete files. If the version of the application changes distributors do not necessarily have to distribute all the updated Swarm-Hashes and only need to update the registered hashes under the identifier. Although IPFS provides its own native nameservice, both Swarm and IPFS support the possibility of adding a nameservice on top of the Ethereum Platform. This nameservice is called the Ethereum Nameservice[9].

Ethereum Nameservice The Ethereum Nameservice consists of a *Registry* and a *Resolver*. Both are contracts deployed on an Ethereum based blockchain. Participants can register domains or sub-domains in the form of nodes or sub-nodes. The owner of a domain can then add a Resolver contract which points to a file hash, smart contract or wallet address. Figure 5.1 shows an example. [Joh]

5.1.4 Decentralised Database Systems

Although a blockchain, in fact, is a large database, using it as a store for structured data is costly and inefficient. Decentralised file systems are good in storing large binary objects but not in storing structured data. Thus, decentralised database systems help to store structured data and might be necessary for a decentralised energy management system.

[6] Website of the Swarm project hosted over a decentralised file system: https://swarm-gateways.net/bzz:/theswarm.eth/
[7] IPFS: https://ipfs.io/
[8] Filecoin: https://filecoin.io
[9] Ethereum Nameservice: http://ens.domains

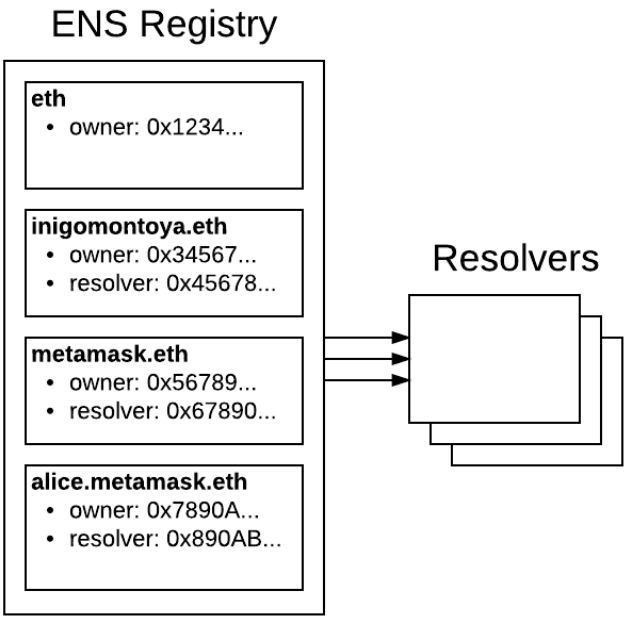


Figure 5.1: Example overview of the Ethereum nameservice contracts. [Joh]

BigchainDB

BigchainDB[10] offers a decentralised database solution by providing a common database system as the underlying state machine for the Tendermint protocol. While every node is owned and operated by several different entities, the network gets a decentralised network of databases sharing the same state. [Big18]

Bluzelle

Whereas BigchainDB offers a solution in which every node stores all the data, Bluzelle[11] partitions the data over multiple nodes and only replicates partially. Because the data is not distributed over the whole network, consensus only has to be found within the replicas of a data set. Additionally, several mechanisms preventing participants from attacking the database system have to be provided. Such mechanisms are the requirement of a participation stake, random verification of requests, redundant execution of data manipulations by multiple nodes or random challenges in order to prove that the target node has the correct data. [BM17]
Bluzelle is a distributed system rather than a decentralised one because data is distributed over a set of nodes which makes several nodes unequal from each other.

[10] BigchainDb: https://www.bigchaindb.com/
[11] Bluzelle: https://bluzelle.com/

TiesDB

Same as Bluzelle, TiesDb[12] also partitions the data redundant over the network. Consensus is achieved by letting all participants read the data but only a small group of trusting participants write to defined data sets. Trusted groups are defined with a permission management. TiesDB integrates into existing blockchain implementations, for example, by enabling the permission management over smart contracts or providing an incentive system. In contrast to BigchainDB, data can be deleted and a more powerful request language has been added. TiesDB is a decentralised database made publicly available trough permission management. [FK17]

5.1.5 Scaling, Privacy or Security

Recent blockchain protocols suffer from scalability issues in transaction throughput or do not give an answer to the question of data privacy and account security. Because those issues are part of the requirements of a decentralised energy management system it is worth taking a look at suitable concepts and technologies which try to solve those problems.

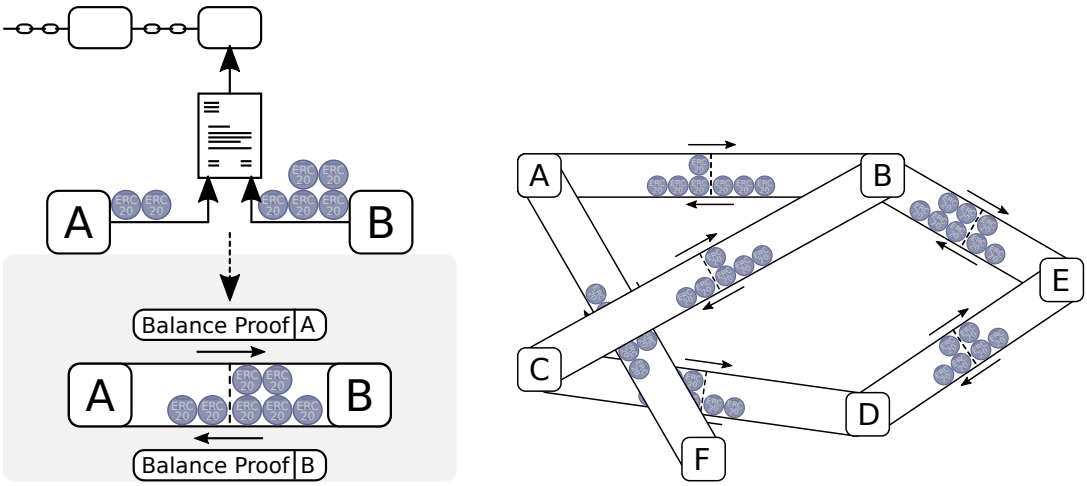
State Channels - Raiden

State channels are a way of aggregating transactions aside from the blockchain in order to only issue one single aggregated transaction. The metaphor of state channels is used because two participants decide on opening a channel in order to issue an arbitrary amount of transactions. In this channel, they are manipulating a common state change which then is written on the blockchain with the shut down of the channel.

The most popular state channel project is the Raiden protocol. Raiden is a state channel protocol for issuing microtransactions of digital assets. Raiden not only allows the creation of channels between two participants, but it also allows building a network of state channels all aggregating state changes to a minimum amount of blockchain transactions. In the Raiden protocol, two participants deposit an amount of a digital asset on a smart contract which is then handled as a balance proof. As long as the participant can proof, that he has enough balances deposited on the smart contract micropayments can be issued within the state channel. Figure 5.2a visualises these payments. As long as the same type of asset, such as an ERC20 token, is exchanged transactions can be transitively aggregated. Figure 5.2b shows how such a network of state channels could look like and how tokens are flowing beyond two participants. [Bra18]

State channel protocols, like the Raiden protocol, not only address the scalability issue of blockchain protocols but also address privacy concerns and reduce transaction fees. Due to the aggregation of payments, the participant's payment behaviour is not publicly written to the blockchain. Currently, there is only a Python implementation available for the protocol.

[12] TiesDB: https://tiesdb.com/



(a) Simple peer-to-peer channel. **(b)** Network of Payment Channels

Figure 5.2: Raiden Payment Channels. [Bra]

Peer-to-Peer Messaging - Whisper

Peer-to-peer messaging is a concept of utilising the underlying peer-to-peer network of the blockchain protocol to exchange short messages between network participants. This helps to bring machine-to-machine communication away from the blockchain and only creates transactions where transparency, consensus and validity are important. A good example is the negotiation of a price for a digital asset between two participants. The negotiation does not need any proof of consensus and can be done apart from the blockchain over a peer-to-peer messaging protocol. Only the eventual trade should be validated, proven and may be transparently published.

One popular peer-to-peer messaging protocol is the Whisper protocol, which runs on top of the Ethereum network layer. Within Whisper nodes broadcast data packages, called *envelopes*, which contain possibly encrypted *messages*. Messages can be either sent to a specific participant or a specific *topic*. A topic is a classification of the message on which other participants can listen to incoming messages. Because participants may not be directly connected with each other messages eventually arrive at their destination as long as all participants are connected to the network. There are two possible implementations. First the Go-Ethereum and second the Parity Whisper implementation. [Eth18]

Multi Signature Wallets

A multi-signature wallet allows multiple entities to authenticate an address on the blockchain. Only if all or a defined part of the entities supply their keys the address can be authenticated. Therefore if one key gets exposed or lost the address is still secured and accessible. [BN73]

Parity offers a possible implementation of multi-signature wallets which provides a more secure access to the owner's wallet.

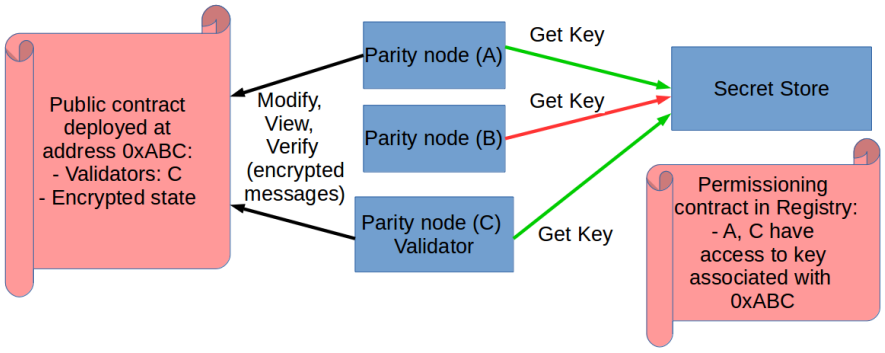


Figure 5.3: Private Transactions [Parc]

Private Transactions

The EEA defines two disjunctive requirements for the transmission of private state on the blockchain. Private state shall either be transmitted and readable only directly by the participants of a transaction, or the private state shall be transmitted to all nodes of the network but only readable by the participants of the transaction [BCNN18]. The concept of private transactions fulfil the second requirement and lift private encrypted state onto the blockchain. This enables the interchange of assets or manipulation of contract states in a private and encrypted way and still have the advantages of the blockchain. Since privacy requirements are part of the decentralised energy management system it is worth investigating Private Transactions. Possible implementations for transacting private state are Parity Private Transactions or Hyperledger Channels[13].

Parity enables private transactions by deploying an encrypted *private contract* code within a common *public smart contract*. This public contract defines a list of *validators* and participants which can decrypt the state of the private contract. Validators validate encrypted, so called *private, transactions*, sent by the participants. In order to find a common key to encrypt and decrypt content, the participants define a *secret store* which holds those keys. The secret store needs to be deployed on nodes of the network and is then able to serve needed keys. Permissioning for access right for given keys can be managed via a *permissioning contract*. Figure 5.3 shows an overview of all the components included in a private transaction. Private transactions are only private with respect to the validators in which all participants have to trust, at least in the group of validators. [Parc]

Interchain Communication

There are several possible scenarios where more than one blockchain exists. Interchanging content over two separated blockchains can be difficult because consensus mechanisms and transaction validation are only available within a single chain. Especially when digital assets have to be exchanged a solution has to be found on how to send an asset from one chain to another. Possible challenges are the destruction of an asset on one chain at the creation of the same amount of assets on the other chain in a decentralised way.

[13] Hyperledger Channels: https://hyperledger-fabric.readthedocs.io/en/release-1.3/channels.html

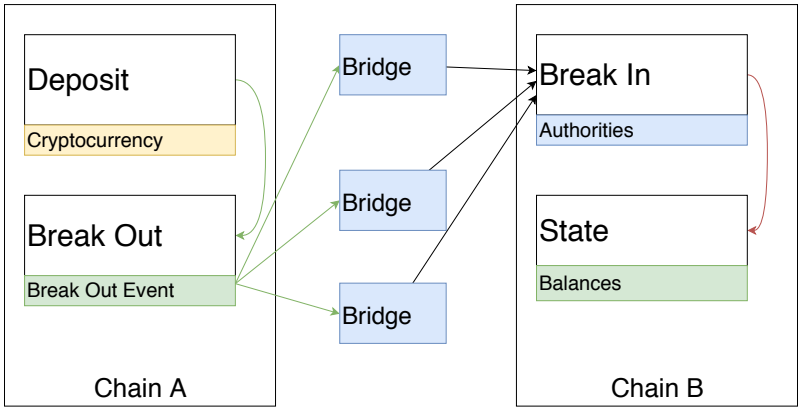


Figure 5.4: Structural overview of the Parity Bridge [Par18]

A simple protocol for interchanging ERC20 tokens is proposed by Parity. The protocol allows the deposition of cryptocurrency on one chain and the creation of tokens of the same amount on the other chain. In order to achieve consensus and validity as decentralised as possible, a group of authorities is defined which run the protocol. When cryptocurrency is deposited on a *bridging smart contract* at chain A, a *Break Out Event* is issued. This event is caught by the authorities which independently send a transaction of the desired state changes to the bridging contract on chain B. If a sufficient part of the authorities submitted the same transaction the contract issues tokens for the desired accounts. Withdrawing cryptocurrency in exchange of tokens is done vice versa. Figure 5.4 shows a structural overview of the concept explained. [Par18]

Another protocol of interconnecting multiple chains in a network of blockchains is the Polkadot[14] network. Still heavily under development, this concept might once solve the problem of interchain communication and replace workarounds such as bridging contracts.

[14]Polkadot: https://polkadot.network/

5.2 Most Promising Existing Solutions

The following section gives an overview of several promising solutions which implement a use case related to a DEMS with the support of decentralised technologies. The analysis has the focus to examine the underlying architecture and technologies and how the solutions can be classified into the distinctive features of blockchain solutions described in section 2.2. Additionally, features like the main business case and the development state of the solutions are evaluated. Development state evaluates the solutions against, whether they are producing a prototype (prototypical), already did proof their concept and board the first customers (semi-productive) or they are fully productive and scale their solution out to many customers (productive). Table 5.1 gives an overview of the solutions and shows the classification of the examined blockchain solutions. It is the goal to gain knowledge about how these solutions solve an equivalent problem such as a platform for a decentralised energy management system and what can be reused or made better when it comes to proposing a new architecture approach.

5.2.1 Oli Systems

Oli Systems[15] is doing blockchain laboratories for energy-related use cases. Laboratory means prototyping several possible energy blockchain solutions. One major use case is the establishment of a peer-to-peer network of smart devices which meter the current energy flow of producing or consuming actors within the network. Common participants are households, companies, power plants or municipals. The network itself is based on a permissioned Ethereum blockchain which is validated by a consortium of authority nodes. Thereby two possible networks come into place. First, the Energy Webchain [HHH+18] and second a variation of the Energy Web Chain hosted by Oli Systems itself. Use cases are developed and tested on the hosted chain and if needed made publicly available on the Energy Web Chain.

Energy Webchain The Energy Webchain is a proof of authority network hosted by the Energy Web Foundation. The Energy Web Foundation is a consortium of responsible companies in the energy sector. Currently, the Energy Web Foundation consists of 70 energy and blockchain affiliates from which ten nodes are hosting authority nodes. With the utilisation of proof of authority consensus, the Energy Web Foundation tries to reduce block times and make the chain more sustainable. In addition to the pure Ethereum network, the Energy Webchain provides private transactions, permissioning of the network, an Ethereum nameservice, identity management solutions and Oracle services. Furthermore, the Energy Web Foundation tries to be the first available solution supporting not only the EVM but also the Wasm Engine. The Energy Web Foundation also comes up with a whole concept on how to establish governance onto the network. This governance protocol shall be open, transparent, secure, able to react quickly, adaptable, cost-effective and scalable to the real world. [HHH+18]
It is remarkable that the Energy Web Foundation hereby describes a setup which is not only concerned with the technology realisation but also tries to include the values of decentralisation into the governance structure of the Energy Web Foundation.

[15] Oli Systems: https://www.my-oli.com

	Oli Systems	**Powerledger**	**Grid+**	**Slock.IT**	**Enerchain**
Main Business Case	Blockchain Laboratories	Platform for various electricity use cases	Enable blockchain based electricity market and beyond	IoT in the Blockchain	Energy wholesale market
Development state	Prototypical	Prototypical/ Semi-Productive	Productive	Semi-Productive	Semi-Productive
Base Technology	Ethereum	Ethereum	Ethereum Public Chain	Mainly Ethereum	Tendermint
Permission	Permissioned	Private (Public)	Public/ Permissioned	Public (so far)	Private
Consensus	Consortium	Consortium (PoW)	PoW	Chain dependent/ Watchdogs	Tendermint
Central regulations	Development, Own chain, Oli Boxes, Additional Services	Development	Grid+ Smart Agent	None	Development, Nodes
Identity	Pseudonymity	No information	Agents and Wallets identifiable	At least Pseudonymity	Client Anonymity
Immutability	Tobalaba hard to tamper/ Own chain easy to tamper	Dependent on Network size	After each State Channel seal hard to tamper	Chain dependent	Dependent on network size
Efficiency	Bottleneck for large number of participants	Multiple Private Chains, State Channels	State Channels	Implementa-tion context dependent	Tendermint as high throughput consensus

Table 5.1: Classification of the blockchain solutions.

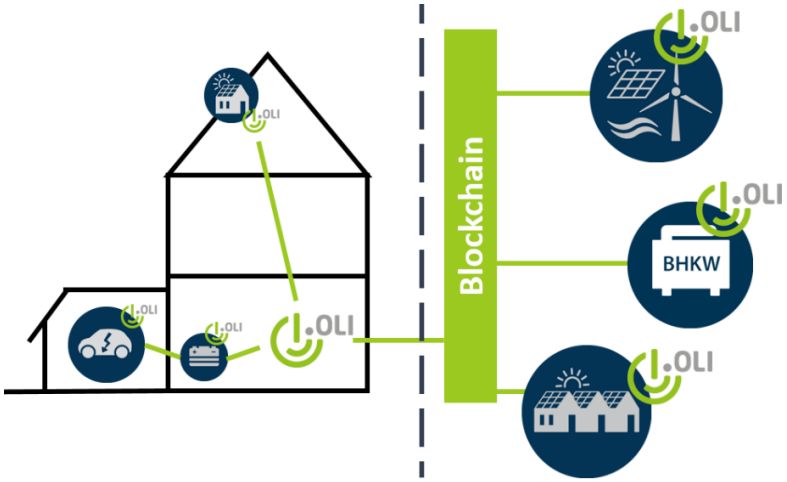


Figure 5.5: Interconnecting a household with other participants over the blockchain [För17].

Within the blockchain laboratory, Oli Systems strives to implement use cases such as a decentralised market for energy-related assets, grid balancing or the certification of the production of renewable energy. The whole system is based on a peer-to-peer blockchain network which connects every participant with each other. To do so, participants have to install a network node in their facility. This node, on the one hand, connects to the network to access the transactions sent over the blockchain and on the other hand, reads the electricity production and consumption of the household. The network node is based on a Raspberry Pi 3 and is certified according to European policies.
The so-called Oli ecosystem [För17] differentiates between an internal and external area. Every participant has his own internal network of devices which are connected to the network node and also may be connected with each other. A single household, for example, could consist of an electric car, a battery and a solar panel. The household can now track all of its power production and consumption values and send them onto the network. After the data is sent to the network, all other participants can access it. This is automated by the implementation of DApps on the network node. [För17]
Figure 5.5 shows a schematic overview of connecting a household with other participants in the external area. Additionally, Oli Systems provides participants with off chain services based on the collected data. Possible services are broadcasting weather forecasts, current energy prices or the identification of suitable peer-to-peer trade partners.

Classification

Oli Systems provides solutions mainly based on the Ethereum technology stack. The underlying blockchain protocols are permissioned and are using the proof of authority consensus. Central regulations appear when it comes to the development of the Oli Box, own code and the hosting of the own permissioned blockchain. To avoid this the code is shared and open source solutions are used as often as possible. Other central components are the additional central services provided. Still, it is worth mentioning that not everything has to be done on the chain to be a decentralised architecture. Since only the wallet addresses of the participants are used for communication, the system is pseudonym for participants. Still, the system is identified in the perspective of Oli Systems

because every customer gets a single Oli Box with a predefined key. When classifying the solution against immutability it has to be divided into the investigation of the Energy Web Chain and the own chain of Oli Systems. Because there are only a view nodes hosting the blockchain protocol tampering the own chain of Oli Systems is easy. Tampering the Energy Web Chain, however, is hard, due to the multiple authorities and the already well-established network. Even though all of the authorities might get tampered or the network breaks down, there will be nodes hosting a copy of the Energy Web Chain ready to serve as a backup. Since Oli Systems is in both times using a permissioned consortium chain latency time for waiting until a transaction is getting minded is reduced. Still, it depends on the use case implemented on the chain and how many other participants are using the chain. Because the Energy Web Chain is used by many participants this could get a bottleneck in the future.

Requirement Fulfilment

With the prototypical implementation of several use cases, Oli Systems is already building a platform for a decentralised energy management system. Therefore, requirements like interconnecting participants, the collection and storage of data and the extendibility of the platform are fulfilled. Using a proof of authority blockchain also fulfils the requirement of decentralisation, availability, scalability and performance to some extent. Yet, Oli Systems does not provide a solution for a large number of participants, a collusion of validators on the self-hosted chain, and the protection of the privacy of participants or sensual data. This can not be done with a single permissioned blockchain solution and requires other concepts to be integrated into the architecture. In respect to the hardware requirements, defined in requirement NFR8, the lightweight Oli Boxes can be a possible candidate. However, they do not resist against high power peaks, strong weather effects and possible manipulation threads.

5.2.2 Powerledger

Powerledger[16] serves as a platform for several different use cases, all located across the regulated electricity network or in private grids. It is the goal to provide the platform as well as the use case implementations. The platform is Ethereum based and consists of the utilisation of two blockchains. First a private consortium chain and second the Ethereum public network. Application examples are peer-to-peer trading across and behind the grid, electric vehicle metering or autonomous asset management. [Pow18]

The architecture consists out of four layers, the *Public Layer*, the *Powerledger Core Layer*, the *Consortium Blockchain Layer* and the *Powerledger Applications Layer*. Figure 5.6 shows an overview of the architecture. The Public Layer utilises the Ethereum public chain. The Consortium Blockchain Layer utilises a custom permissioned consortium chain. Powerledger uses two different tokens. A public standardised *POWR* token and an internal *Sparkz* token. The Sparkz token is used as a digital asset within the permissioned chain. In order to obtain Sparkz tokens, one exchanges them with POWR tokens bought at a public exchange. POWR tokens are for traders or application providers. Sparkz tokens are meant to be provided by application providers to their customers and

[16]Powerledger: https://www.powerledger.io/

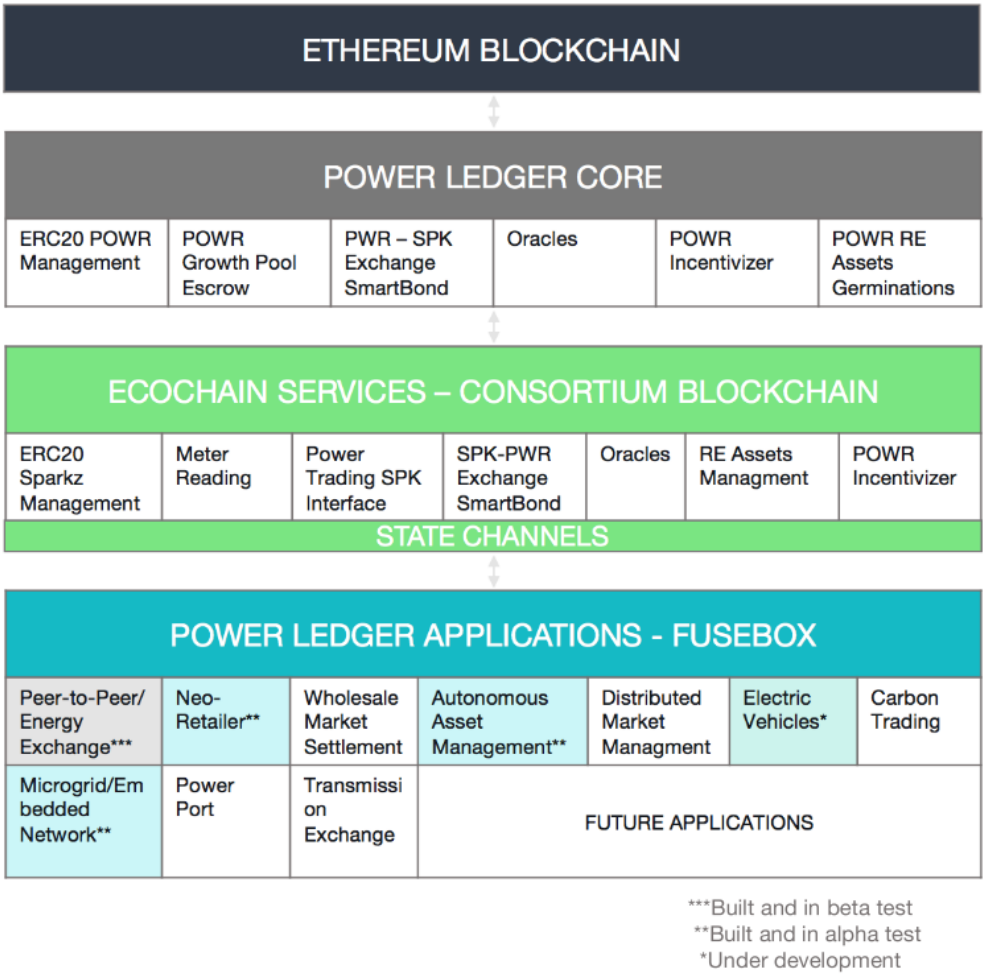


Figure 5.6: The Powerledger architecture [Pow18]

are bound to the local currency. Furthermore, POWR tokens are rewarded to participants only trading with renewable energies in order to incentivise the reduction of carbon dioxide emission. The PowerLedger Core Layer provides smart contracts for the token management and exchange between Sparkz tokens and POWR tokens. The exchange between those tokens is done with the help of so-called *Smart Bonds*. POWR tokens are deposited in an Ethereum Smart Bonds and therefore exchanged against Sparkz tokens. The locked POWR tokens can only be withdrawn if a sufficient amount of Sparkz tokens is supplied in return. The consortium blockchain layer provides the permissioned consortium chain, as well as smart contracts for token exchange and smart contracts for meter readings. For scalability, the Consortium Blockchain layer utilises state channels. It is the idea of having the Consortium Blockchain Layer deployed for each region Powerledger is used onto. The application layer consists of the applications running on the Powerledger platform. [Pow18]

Classification

The Powerledger platform is an Ethereum based solution. In general, it is a private blockchain where application providers give access to their customers. The POWR tokens are public and grant participation for an application provider or just token trader. The whitepaper does not give enough information about how the platform especially the private chains are hosted. The research of such use cases has shown, that this is mostly the case when validators are hosted centrally and not distributed independently over the stakeholders. The Powerledger whitepaper also does not explain how the interchain communication works. If it is a single party executing each smart contract on the separate chain this is a centralised chokepoint. Furthermore, there is no publishing of source code or configurations, but a transparent communication with participants. Powerledger does not provide any information on how keys and wallets are stored or how the concrete identities are managed. In respect to immutability, the manipulation of a single consortium network is dependent on the network size. The application parts on the Ethereum public chain are immutable stored and will not get tampered. Powerledger uses multiple private chains and state channels as a scalable approach against the expensive and inefficient public Ethereum chain although the sustainability of the public Ethereum chain is used in order to trade POWR tokens.

Requirement fulfilment

Same as Oli Systems, Powerledger already proposes a platform for running decentralised energy management applications which fulfil all functional requirements. However, NFR1, NFR6 and NFR7 are only partially fulfilled. The governance and development of the platform are centralised due to the lack of information on concrete implementation details. Data stored on the private chains is temperable and transactions might not be auditable anymore due to the utilisation of state channels. Nevertheless, with the trade-off for reduced audibility Powerledger improves privacy concerns about sensitive data. Furthermore, Powerledger fulfils the requirements of NFR2, NFR3 and NFR4. With multiple private chains, Powerledger reduces the number of participants on a single blockchain and leverages transaction throughput trough state channels. This provides the potential for a highly available, scalable and performant platform. Powerledger does not provide a solution for NFR8 and does not give any information which could evaluate the fulfilment of NFR9.

5.2.3 Grid+

Grid+[17] is developing a consumer-focused smart agent which holds cryptocurrencies in order to buy and sell electricity from electricity providers. It is the goal to not only enable a blockchain based electricity market but also to enable several use cases on top of the Grid+ infrastructure. Grid+ aims to provide an understandable solution for core users by abstracting their underlying architecture. Grid+ solely runs on the public Ethereum chain. Moreover, Grid+ is using Raiden state channels in order to reduce transaction costs and latency. An internal BOLT token is used as a digital asset which is coupled to the U.S. dollar currency. [Con17]

In order to make trades possible, Grid+ provides a *Smart Agent* which hosts an Ethereum Light Client in order to sign transactions. It is the main use case of the device to pay the electricity costs of a customer via cryptocurrency in nearly real time. Every smart agent's registry number and the owner's wallet address is registered in a separate smart contract. This ensures that only owners of a Grid+ Smart Agent can take part in the peer-to-peer market. Figure 5.7 shows a typical household holding a Smart Agent. Behind the electricity meter producer and consumer devices such as a battery or a photovoltaic panel could be connected with each other. The Smart Agent can read the electricity meter and automatically pays or receives money for the consumed and produced energy. The Agent can be controlled over a smartphone connected to the WiFi of the household. In addition, the Smart Agent can be locked with a multi-signature wallet which allows having keys not only on the device but also stored on an external *Signing Agent*. In contrast to Oli Systems, wallet keys are only generated when the Smart Agent has been delivered to the customer and therefore are only known by the customer. The Grid+ platform shall not only be used for energy trading, features like using the agent as a cryptocurrency wallet, a proof of stake signer or an Ethereum gateway for IoT devices are also planned in the future.[Con17]

Classification

Grid+ is operating on the public Ethereum chain. Only participants owning a Grid+ Smart Agent can participate in the peer-to-peer market. By running on the public network, consensus is done by the proof of work protocol. Same as for Oli Systems, a central component within Grid+ is the necessary Smart Agent. All of the software and hardware components important for using the Grid+ application are only provided by Grid+ itself. However, all of the used code is open sourced by Grid+. Yet, direct development participation is no part of the current plan. The Grid+ system relies on the full identity of the users because every Smart Agent and its wallet is registered on a smart contract. Immutability is only given after each state channel seal. Afterwards, the data cannot be tampered due to the utilisation of the public Ethereum network. To reduce the disadvantages of the Ethereum public chain in transaction throughput and costs, Grid+ is using Raiden as a state channel technology.

[17] Grid+: https://gridplus.io/

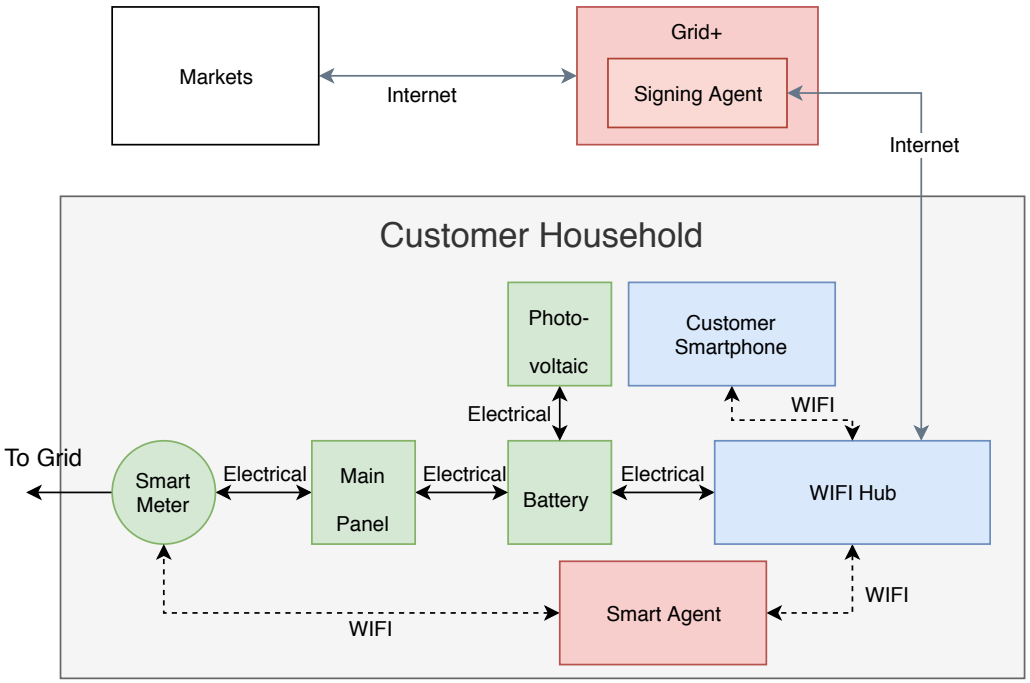


Figure 5.7: Architectural setup of a multi signature wallet within the Grid+ architecture [Con17]

Requirements fulfilment

Grid+ connects all participants within a peer-to-peer network but does not provide intercommunication between participants (requirement FR1). Currently, the platform is only used to ensure payments with the electricity provider and not with single peers in the network. Although, due to the extendibility (requirement FR3) of the system an application doing so might be implemented in the future. The Grid+ system also does not provide a means to collect and store energy data to provide them for the grid (requirement FR2). With the utilisation of the public, Ethereum network and state channels, Gird+ creates a highly available, scalable and performant solution. Regarding requirement NFR9, the solution is not sustainable due to the execution of all transactions on the public proof of work Ethereum chain. Nevertheless, Grid+ focuses on utilising proof of stake once it is available on the public Ethereum network. The required anonymity of NFR5 is not provided at all. All participants of the Grid+ Platform can be traced back to their wallet address and therefore to the identifier of the Smart Agent, they purchased. In the trade-off of auditability Grid+ strengthens the requirement of privacy by using state channels. Compared to Oli System the Smart Agent even solves more of the hardware requirements (NFR8). The Smart Agent is lightweight and tries to provide a means against manipulation threads. Security is also improved with the help of multi-signature wallets.

5.2.4 Slock.IT

Slock.IT[18] tries to connect every single IoT device with the blockchain, building a payment platform for humans and devices with any interaction possible. In order to connect as many devices as possible, they introduced a very lightweight client which can connect to multiple chains. This client is called the *Incubed Client*. Although it is possible to connect to chains of different technologies, the Ethereum stack is mainly used. [Jen18]

The Incubed Client is a lightweight stateless client with the only purpose issuing a transaction to an *external node* in order to be signed. The external node is part of a network of registered nodes which deposit a sufficient amount of tokens on a smart contract. The network itself is a separate established peer-to-peer network called the *IN3 network*. The smart contract is a contract deployed on any chain, Incubed Clients want to communicate with. If the external node does not sign the transaction correctly the node loses all of its deposited tokens. If the external nodes do sign the transaction correctly it gets an incentive paid by the Incubed Client. In order to find falsely signed transactions *Watchdogs* validate already signed transactions and punish a node in case of misbehaviour. Watchdogs are randomly selected external nodes of the IN3 network which double check the already signed transactions. Since the Incubed Client does not have to sign transactions on its own and therefore only needs to be capable of sending and receiving RPC calls to a list of known nodes, the technical specification for an Incubed Client can be very low. This is an advantage especially in the emerging world of interconnected things. Figure 5.8 shows the functionality of the Incubed Client within a network of external nodes. [KJJD18]
Slock.It is also considering to integrate state channels into the concept. Because external nodes are selected randomly for each transaction of the Incubed client, a network of state channels, such as Raiden, can also be utilised.

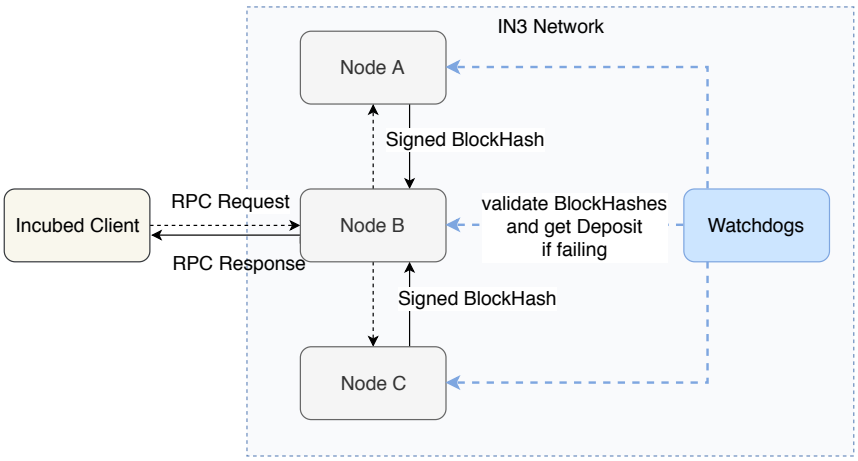


Figure 5.8: Functionality of the Incubed Client [Jen18]

[18] Slock.IT: https://slock.it/

Classification

Slock.It mainly develops for the Ethereum platform but offers platform independent concepts. So far, the IN3 network is public and accessible to everyone. The consensus is achieved by the underlying blockchain technology. Watchdogs validate requests within the network. In fact, there are no central regulations within the IN3 network. The development is open and transparent for everyone and the network itself does not rely on any centralised drawbacks. With respect to identity, Incubed Clients at least can be identified as pseudonym actors although this might be chain dependent. The IN3 network itself has no drawbacks when it comes to efficiency as long the network is big enough for the number of Incubed Clients, transactions will be processed without any bottlenecks. Yet, the transaction throughput is defined by the underlying blockchain protocol, the Incubed Client wants to communicate with.

Requirement Fulfilment

Slock.It itself is no platform and therefore does not fulfil any functional requirement. Nevertheless, integrating the Incubed Client into an architecture solution could solve several non-functional requirements especially when it comes to the demanded hardware requirements from requirement NFR8. As mentioned above, the IN3 network is fully decentralised. Thus there is no single point of failure and no single point of responsibility or control. Furthermore, all the development is public and transparent which helps to fulfil the demand of requirement NFR1. All the other non-functional requirements rather depend on the blockchain protocol integrated with the Incubed Client than on the Slock.It solution itself. One drawback of the Incubed Client is that it cannot react directly to events on the blockchain due to the lack of a direct connection to the blockchain protocol.

5.2.5 Enerchain - WRMHL Framework

Enerchain[19] is a blockchain application for business-to-business power and gas trading in the energy wholesale markets provided by Ponton. The Enerchain project is currently a proof-of-concept and based on the internally developed *WRMHL (wormhole)* framework which uses Tendermint as the blockchain technology [Pon18]. It is the goal to provide a distributed marketplace with market access costs as low as possible. The framework states to be process independent, customisable, modular, resilient and secure. The WRMHL framework is divided into two separate environments. The *Hosted Environment* and the *Business Environment*. The architecture itself consists of three layers. A *blockchain layer*, a *middleware layer* and an *application layer*. Figure 5.9 shows an architectural overview of the WRMHL framework. The blockchain layer consists of nodes hosting a peer-to-peer network based on the Tendermint protocol [BKT18] technology. Since Tendermint only comes with a network and consensus protocol *Node Adapters* extend the core technology with functions like caching and transaction validation. On the client side, *Client Adapters* connect the clients to the network and manage the identity of each client. Client Adapter and Node Adapter together form the blockchain middleware. Finally, use cases, such as the Enerchain use case, using the platform are then implemented in the application layer with *Vertical Clients*. The network itself

[19] Enerchain by Ponton: https://enerchain.ponton.de/

can be hosted on any machine and by any participant of the WRMHL framework. For the proof of concept of the Enerchain project, the nodes are hosted by Ponton. Each node of the peer-to-peer network consists of a Tendermint node and a Node Adapter. [Pon18]

Classification

The Enerchain together with the WRMHL framework is based on the Tendermint protocol. Therefore, the Enerchain project is based on a consortium Byzantine fault tolerant consensus algorithm. The Enerchain project itself is a private chain where currently most of the nodes are hosted by the providers. All changes are incrementally published in close consultation with the participants. Still, all technical decisions are implemented only by the developers itself. It is the plan to found the Enerchain legal entity in the future [Mer18]. The Enerchain legal entity is a governmental union of all participants in order to discuss future decisions. During the proof of concept and until the operation of the legal entity provisioning and maintenance are established by Ponton itself. As long as participants are not involved in a trade their identities are anonymised and only the exchanged values within a transaction are visible. All participants involved in a trade can see the identities of the other trade partners. During the proof of concept, many nodes are currently hosted by the provider. Therefore immutability is only given if the provider itself is not Byzantine in a way that he does not manipulate the data or seeks an advantage in using the data. For the productive network, it

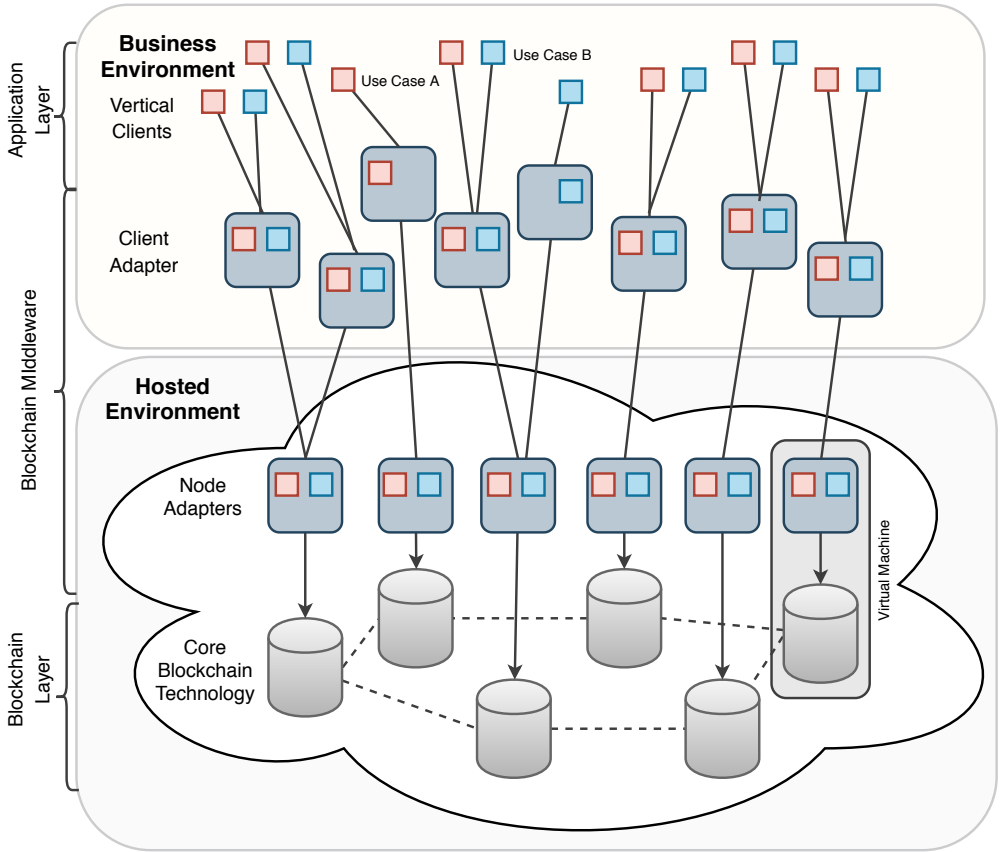


Figure 5.9: WRMHL architecture diagram [Pon18]

is planned to host the nodes within the members of the Enerchain legal entity. This will make the network immutable, as long the limit of Byzantine nodes is not reached. Ponton is operating its framework with a block time of approximately one second per block. It is the goal to not exceed the limit of 15 minutes per trade in order to fulfil the requirements of the current most liquid energy market segments in Germany [Mer18].

Requirement Fulfilment

Enerchain and especially the WRMHL framework fulfil all functional requirements of section 4.1. Especially requirement FR3 is achieved in a way the other solutions do not. With the WRMHL framework application, application logic can be implemented on all layers of the framework. This enables developers to utilise the core blockchain protocol on their demand. Yet, the Enerchain proof of concept has some centralised characteristics. First, most of the nodes are still hosted by the developers. Second, due to the utilisation of Tendermint, the developers had to implement their own state machine with many features already existing in the community. This creates a central entity because now only the WRMHL developers understand and maintain the code base. With the establishment of the Enerchain legal entity and the release of the Enerchain project, the first centralised characteristic most probably will be repealed. Due to the Tendermint protocol, the WRMHL framework is a highly available, scalable and performant application. Moreover, the Enerchain project has high data protection and privacy requirements, which are also implemented in the underlying state machine. Therefore, values transferred within a transaction can be seen but without relieving the origin and destination of the assets [Mer18]. Because the Enerchain project runs on normal machines without any requirements for lightweight clients, the Enerchain project or the WRMHL framework do not have a solution for requirement NFR8.

6 Concept for a Generic Architecture Proposal

The following chapter shows a generic architecture proposal for a decentralised energy management system. The proposed approach utilises suitable concepts from the decentralised technology stack, analysed in chapter 5, as well as useful architectural decisions met in already existing solutions. It is the goal to combine all found aspects into a generic concept which tries to fulfil most of the requirements identified in chapter 4. The architecture proposal is described in a way, that it can be implemented with any present or future technology supporting the underlying protocols.

6.1 Assumptions

There are several assumptions made up before proposing an architectural concept for a decentralised energy management system.

1. Reading out metering data is an already known and solved problem. There is no need of proposing equivalent solutions.
2. The process from reading out the metering data until transaction creation on the blockchain is kept secure. This is already a well-known problem and several solutions have been proposed [How93] [ZR10].
3. It is assumed that all technology concepts utilised in the architectural concept are in their state as proposed by the technology providers and developers. Thus it is the goal of the implementation of this architecture to figure out the reliability of the applied concepts.

6.2 Architectural Overview

The proposed architecture consists of three main components. First a peer-to-peer network, second a platform providing the blockchain running on the peer-to-peer network and third an application layer of decentralised energy management applications. In addition, the architecture is governed by a governance structure. Figure 6.1 shows an architectural overview of the system and table 6.1 a short summary of each component. These components together build a decentralised energy management system deployable in a local area grid, for example, in the fourth layer of energy management.

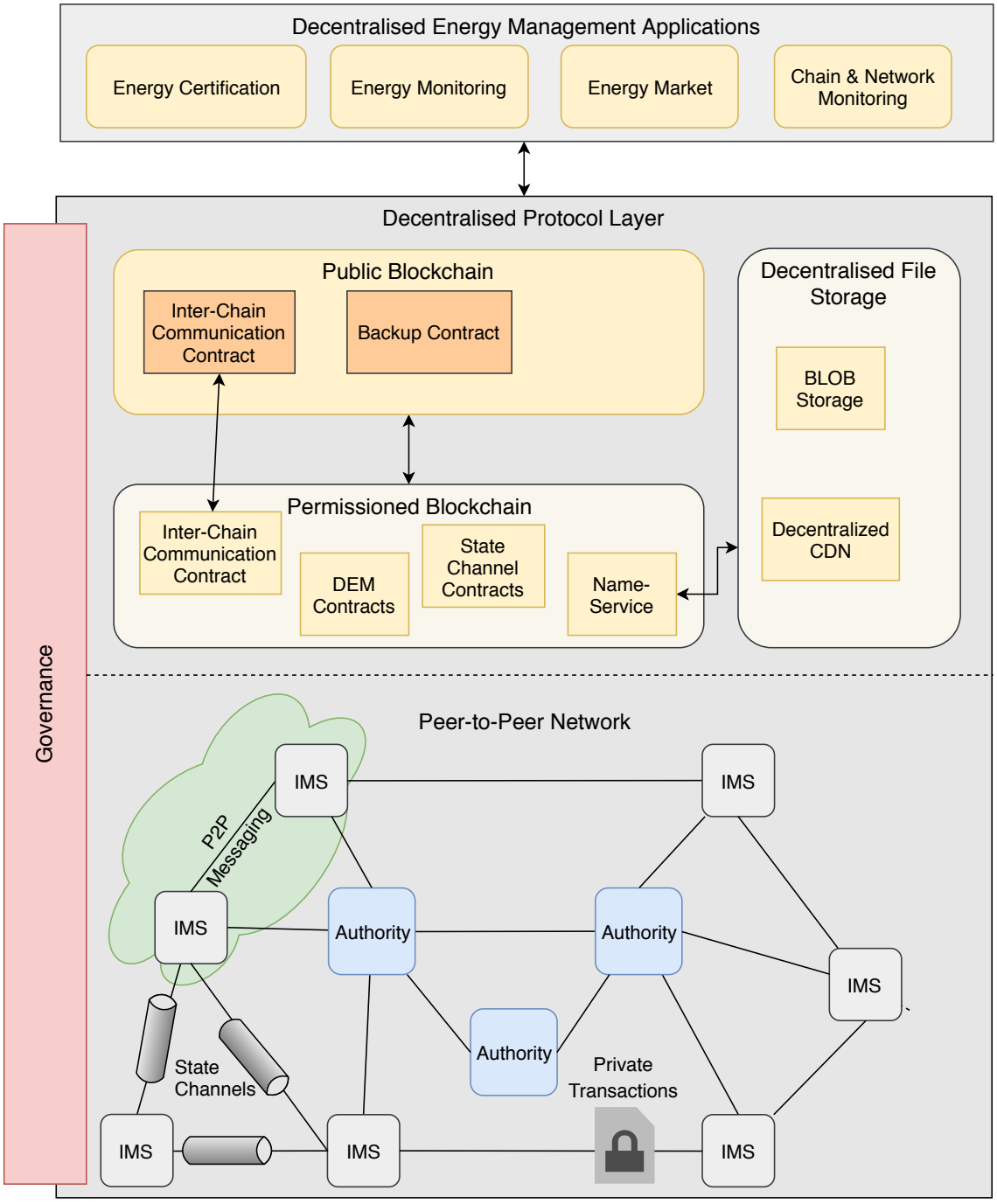


Figure 6.1: Architectural Overview of the generic architecture proposal. IMS means Intelligent Metering Systems, CDN Content Delivery Network and DEM Decentralised Energy Management.

Decentralised Energy Management Applications	
Chain & Network Monitoring	Applications for monitoring the network status and underlying blockchain protocols
Energy Related Applications	Energy management applications deployed on the architecture fulfilling energy related use cases
Decentralised Protocol Layer	
State Channels	Scalable trustless off-chain communications
Peer-to-Peer Messaging	Independent private machine-to-machine communication
Permissioned Blockchain	Custom permissioned blockchain instance reducing latency and economical costs. Hosts contracts of application layer.
Private Transactions	Transacting private encrypted state on the permissioned blockchain
Public Blockchain	Used for backup hashes, inter-chain communication, transferring financial assets
Decentralised Storage	Distributes application and data files, interconnects to a nameservice on the permissioned chain
Peer-to-Peer Network	
Intelligent Metering System	Metering device capable of hosting lightweight blockchain technologies and decentralised application parts
Authority	Server hosts all protocols, provides consensus, communicates with other grids and blockchains,
Governance	
A governance structure for selecting peers, managing participants an publish development artefacts.	

Table 6.1: Summary of each component within the proposed architecture

6.2.1 Peer-to-Peer Network

The peer-to-peer network is the foundation of the energy management system. It solves the requirement of interconnecting all participants in a local area grid and sets the necessary prerequisites for running any decentralised protocol on top of it. The peer-to-peer network focuses on integrating devices, weak in computational and storage capacities, such as Intelligent Metering Systems (IMSs), into an interconnected network of participants. This network consists of several IMSs deployed in the facilities of each participant. All IMSs of a community of participants are forming a grid and are connected which each other over a peer-to-peer protocol. The community decides on several authorities inside the grid capable of hosting blockchain nodes. Authorities can be any participant taking part in the grid. Each node in the network has to implement the proposed architecture stack or parts of it. This is comparable to the network protocol stack [MH17], where each device of the network architecture implements the stack dependent from its role within the network. Connected nodes can also be normal machines provided by application developers or other stakeholders of the underlying power grid.

In order to be decentralised and high available, the network needs a sufficient network size regarding the number of participants. Nevertheless, the network should not exceed a maximum size in order to still scale in the matter of transaction throughput and governance effort. For a large number of

participants, several networks in different areas or industrial sectors can be deployed. Instead of the approach by Powerledger, these networks should not be isolated and rather should be able to intercommunicate with each other.

Because the network is meant to transact sensible data which concerns only a closed number of participants, the network should also provide a means for this. One possibility of closing the network for uninvited access is a deployment of the network inside a virtual private network (VPN) which does not expose its IP addresses to the outside. Another possibility would be to restrict the clients from connecting with nodes they do not know. This can be done by IP whitelisting or an authentication handshake over asynchronous key encryption. However, this would affect the flexibility and decentralisation of the network significantly. There is also a mixture of those two possibilities presented by J. Poon [Poo18] in which several VPNs can connect which each other by calling a registrar node. After a successful asynchronous key authentication, this registrar then publishes nodes which act as entry points for the VPN [Poo18]. Nodes acting as entry points are also called bootnodes. Therefore the registrar node is called Bootnode Registrar. Figure 6.2 shows a schematic overview how two separate VPNs can be connected. This method is especially helpful if several groups of participants, each sharing a VPN together, want to connect with each other. [Poo18]
Another possibility is proposed by the Enterprise Ethereum Alliance [BCNN18]. The EEA proposes the deployment of contracts on the overlying blockchain protocol which grant permission for nodes to take part on the network. Therefore nodes which want to connect to the peer-to-peer network first have to ask for access on one of the other nodes. If the nodes' identical identifier is allowed to connect to the network, access is granted [BCNN18]. Within this process, concrete node identifiers are stored in a central contract readable for every participant on the network, which might be a drawback when it comes to the fulfilment of privacy requirements.

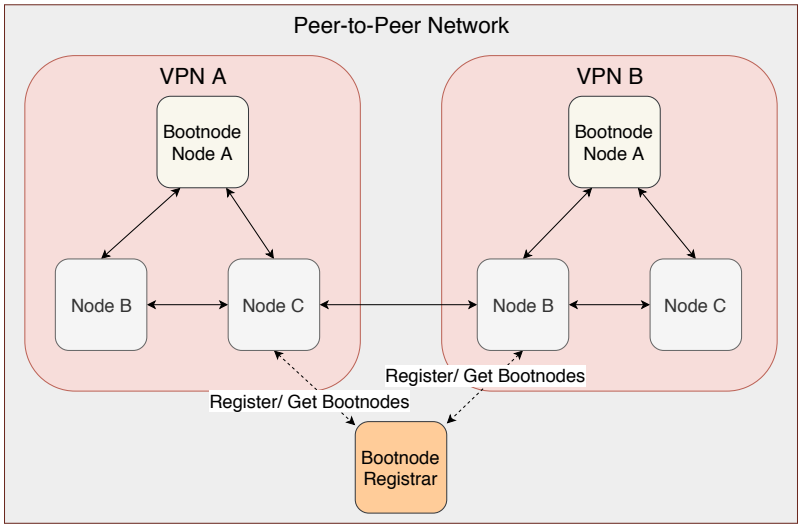


Figure 6.2: Interconnecting two virtual private networks to a single restricted peer-to-peer network [Poo18].

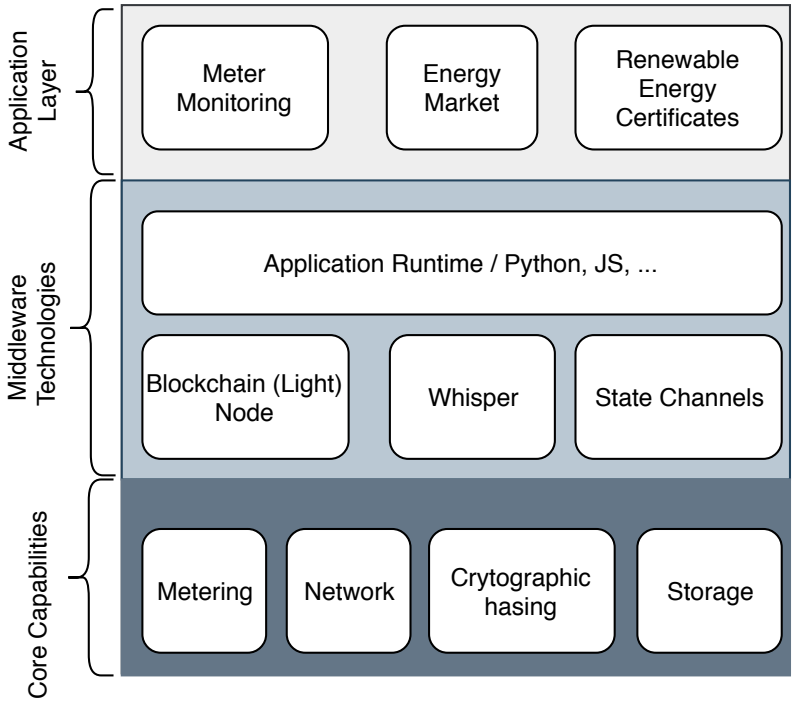


Figure 6.3: Overview of a Blockchain Ready Intelligent Metering System.

Intelligent Metering System

Intelligent Metering Systems are the key to make the underlying power grid a smart and intelligent one. Same as done by Grid+ they are nodes capable of hosting lightweight blockchain technologies, but also can meter electricity consumption and production of the participant. Although Slock.It is proposing a solution for devices not necessarily hosting blockchain protocols, those devices cannot react to events on the blockchain. Therefore the solution of an Incubed Client equivalent device is not a possible option, because reacting to events issued by other participants is important for some use cases. Thus, additionally to the metering functionality, an IMS has to be able to connect to a network, has the computational power to execute cryptographic hash functions and has adequate storage. The network connection is needed in order to connect to the peer-to-peer network. To send a transaction and access the account of the user cryptographic hashing power is needed. The storage is needed because storing the blockchain and the required applications need space and at least some I/O-power. On top of these *core capabilities* decentralised protocols can be executed, for example, a lightweight blockchain client, a peer-to-peer messaging protocol or state channels. Furthermore, the device has to be able to provide not only a runtime for compiled applications but also for interpreted languages like Python or Javascript. The *application layer* is then utilising these *middleware technologies* together with the core capabilities in order to run applications on the IMS. The most basic application could be a monitoring application for metering data. More complex functions like participating in an energy market or accessing renewable energy certificates must be possible on the meter, as well. A meter having all these capabilities may be called a *Blockchain Ready* device. Figure 6.3 shows the three technology layers described above.

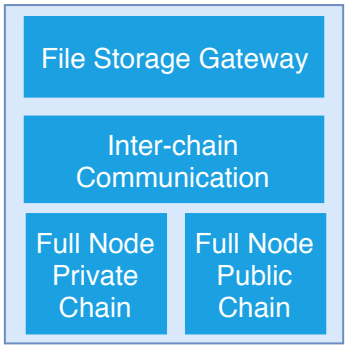


Figure 6.4: Overview of a validator node.

Authority

Due to the fact, that the Intelligent Metering Systems are only hosting light nodes, additional nodes have to be integrated into the network which are capable of hosting full nodes. These authority nodes have the task of hosting a full node of each blockchain part of the protocol layer and providing consensus for the permissioned chain. In order to communicate between several blockchains, the authority node needs to provide a way for inter-chain communication. This enables the possibility to interchange assets, send data backups to a more persistent blockchain or access common contracts and applications of a public blockchain. Furthermore, this gives a single local area grid the possibility to communicate with other grids, in order to exchange for example electricity certificates over the grid borders. Communication between multiple grids over authorities can be possible as well.

Participants need the means to get access to possible decentralised energy management applications. This can be done by the authorities hosting a decentralised file storage with possible applications. Because the storage capacity and computational power of the IMSs might be too low in order to host a file storage node by their own, authorities need to provide a gateway to access these applications. To fulfil all of these tasks authority nodes need much more computational power and storage capacities than the Intelligent Metering Systems. This leads to authority nodes being servers hosted by selected participants of the grid. Figure 6.4 shows an overview of the protocols implemented within the node.

6.2.2 Decentralised Protocol Layer

The decentralised protocol layer consists of several decentralised protocols running on top of the peer-to-peer network. It holds a permissioned blockchain, a public blockchain, a state channel and peer-to-peer communication protocol and a decentralised file storage. These three components shall help to enable the peer-to-peer network in order to build decentralised energy management system applications on it.

State Channels

Currently, most blockchain technologies suffer through the issue of scalability. Thus supporting only a limited amount of transaction throughput relative to a large number of participants. This issue can only be solved by a trade-off against the degree of decentralisation or other blockchain characteristics. Therefore the architecture needs the means to solve this issue by outsourcing workload outside of the blockchain protocol but still being able to trustless transact state. As explained in section 5.1.5 state channels or similar concepts could do this by aggregating many transactions between several parties to a small number of transactions being executed on the blockchain. Therefore similar to the solution presented by Grid+ or Powerledger in section 5.2, the network needs to support the usage of a state channel protocol or other off-chain aggregation mechanisms for transactions.

Peer-to-Peer Messaging

State channels present an off-chain solution supporting the transaction pattern for state changes within the state machine of a blockchain, mainly for digital assets defined on the blockchain. In some cases, it is necessary to just send information completely independent from a transaction log or a state machine. Therefore, the architecture needs the means to support messaging between devices part of the peer-to-peer network. This can be established by utilising a peer-to-peer messaging protocol introduced in 5.1.5. Because most peer-to-peer protocols encrypt the sent message, these protocols can also be used to enable private communications.

Permissioned Blockchain

Although state channels solve one of the main issues, the issue of scalability, state channels come with some drawbacks as well. One disadvantage is the lack of immutability as long as the state channel is not closed. Furthermore, all transactions happening during a state channel lifecycle might not be reflected and therefore not traceable in the distributed ledger. This might not be an issue for two parties trusting each other but becomes important when more than one parties are involved and transactions are dependent on each other. A good example is a decentralised market for energy-related assets. Although state channels are necessary for applications with high transaction throughput and less traceability requirements, also a means for a more traceable and more immutable scaling solution is required.

A public blockchain could solve this issues, but comes with several problems as well. Public blockchains, for example, have high transaction fees, small transaction throughput in relation to the number of participants and are energy expensive if proof of work is used as a consensus algorithm.

Furthermore, the transparency of state changes in an industrial use case is only wanted between the participants and may not be visible to the whole world. Therefore, a permissioned blockchain with an authoritative consensus algorithm is used which even brings more advantages. This leads to an architecture concept which consists of a permissioned chain which is hosted and validated by the validator nodes within the peer-to-peer network. Because of the smaller size of participants and the more performant consensus algorithm, state channels are only needed for frequent peer-to-peer transactions and not for complex, interdependent, transaction mechanisms. Moreover, a permissioned blockchain solves several privacy issues because only participants allowed to interact with the blockchain can access and write onto the blockchain.

Private Transactions

Sometimes it may be necessary for a participant to issue transactions which are only readable by a subset of the network participants. An example could be a market use case in which price offers for an energy asset are kept secret until the auction is over. Some private transaction concepts explained in section 5.1, require a trusted third party or at least a group of peers executing consensus in order to check the validity of the transaction. Because trust is already established by the authorities of the peer-to-peer network, these authorities could form a group of validators which are then allowed to read the encrypted data and validate it against existing state changes. To do so, these validators are not allowed to directly take part in the overlying use case and are only allowed to act as a passive party. Otherwise, they could earn advantages from being able to decrypt incoming state changes.

Public Blockchain

Because a permissioned blockchain comes with the disadvantage of having only a small amount of nodes hosting the blockchain, a solution for having a sustainable ledger of undoubtedly immutable entries is needed. This can be any public blockchain such as the Tobalaba or the Ethereum network.

Hashed data backups are stored onto the public blockchain frequently which makes it hard for a Byzantine blockchain to alter data which already has been written. Moreover, in case of a network breakdown, the past state of the applications can be proven and transferred to a different network as long as there is one node left, hosting the blockchain. This also reduces the risk of having a vendor lock-in for having a persistent and validated copy of the data stored on the permissioned blockchain.

In addition to the sustainability advantage, the interconnection with a public network can also be used for transferring data from the public to the permissioned blockchain or vice versa. This is important when several grids with their own permissioned blockchain want to communicate with each other. Especially transferring financial assets can be useful, due to the underlying infrastructure provided by a public network, for example, established tokens, exchanges and a large participant number. Using the concept of bridging contracts such as the Parity Bridge helps in achieving this.

Decentralised Storage

Having protocols on top of the peer-to-peer network for exchanging, computing or calculating assets and data objects, these protocols reach their limits when it comes to handling large data files. Furthermore, most current blockchain technologies set a block size limit and transferring large data over messaging is no possible option as well. However since participants, for example, application providers and their applications, need the means to distribute large files over the network, this should be provided by the architecture. A possible solution is a decentralised file storage protocol hosted by the authorities or other nodes providing a sufficient amount of storage capacity. This file storage then could function as a content delivery network for static files. Authorities or application providers provide applications, which a household can install on its Intelligent Metering System over the decentralised storage. In order to have a better way of discovering distributed files a nameservice deployed on the blockchain can be used. This nameservice points to the latest version of a stored file on the distributed file storage.

The used permissioned blockchain protocol can be utilised even more. One possible way is a deployment of a versioning contract, which keeps track of the changes of several files and helps to validate the correctness of the retrieved files by the file storage. This is of particular importance, because IMSs do not have enough storage capacity to host an instance of the distributed file storage protocol on their own. Solving this issue requires a file storage gateway provided by the authorities. The versioning contract then gives nodes which do not host a storage, the possibility to validate the files retrieved from the gateway. This reduces the trust required to a single authority. In order to distribute the static files over the network, an incentive with value pledged tokens can be established. Nodes hosting files of, for example, application providers or authorities, receive a small amount of token value. This is not necessary but strengthens the network and the availability of the files. A trivial working solution anyway can be a decentralised file storage without any incentive but having the authorities replicating files on their own will.

6.2.3 Governance

Similar to the governance mechanisms presented with the Energy Web Foundation in section 5.2.1 the proposed architecture concept must also have a governance structure. This governance structure helps to organise the participants, select authorities and builds the foundation for a self-sustainable provision of the architecture. All governance decisions explained can either be defined in a code of conduct or implemented within smart contracts. In the following, the fundamental governmental decisions necessary for the architecture are explained.

First of all, participants of a grid need to identify possible authorities who are capable of hosting the permissioned chain. The authorities have to have enough technical knowledge about the core technologies used on the platform. Dependent on the network size an acceptable amount of authorities has to be found. This is especially important for finding consensus without having a single party involved and having an adequate amount of full nodes serving the blockchain for multiple light nodes. An amount which is too small might lead to a centralised system. A number of authorities which is too big can lead to a system which has high maintenance and administration overhead. The authorities have to be heterogeneous and independent from each other. Therefore it is the goal to find authorities which represent each entity of the underlying grid. A smart grid, for

example, could exist out of a group of households, a big consumer such as a company, a municipal, a power plant and the distribution operator. All these entities should have the possibility of hosting an authority node. Additionally, at least one authority should be elected by all participants.

If demanded people other then authorities, capable of hosting full nodes, also can host nodes within the peer-to-peer network. This creates a more sustainable network and gives participants the possibility not to only rely on the authority nodes when it comes to reading the blockchain or issuing transactions.

All parties involved in the development of the decentralised energy management system and its applications must transparently publish their work. Only if the participants can verify whether the development is done without any party trying to secure an unfair advantage, the participants can use the platform without any trust into the developers.

Conclusively, all governmental decisions taken by responsibilities should only be executed with the consent of all other participants or at least the consent of the authorities. In the best case, this is implemented within the blockchain protocol, for example in a smart contract.

6.2.4 Decentralised Energy Management Applications

On top of the architecture resides the decentralised energy management applications layer. Application providers can utilise the platform in order to extend it with their own applications. As described in figure 6.1 and section 4.1 possible applications could be an application for an energy certification use case, a monitoring application or a market for energy-related assets. In general, an application consists of three components. First, the application component which resides on the IMS, second the component of the application which is done on the blockchain and third graphical interface component which makes the application accessible for the user. This principle is similar to the way the WRMHL-Framework, introduced in section 5.2.2, is providing applications.
It is not necessary for the application provider to implement a new graphical user interface. The data provided by the network and the smart contract can also be integrated into an already existing application. This can be, for example, an application implemented by the network provider or the provider of the Intelligent Metering System. Still, it is important to note, that the data then should still be accessible from the IMS, if possible.

Figure 6.5 shows an example of the interaction of the components for a possible monitoring use case between the Intelligent Metering System deployed on a household and the grid provider interested in its energy usage and production data. The graphic is divided into the deployment of the application onto the architecture (a) and the interaction of the participants after deployment (b). First, the application is enabled by the grid provider who provides the GUI over the decentralised storage (1) and deploys the contract (2) with his own node. The application component on the Intelligent Metering System then reads the metering data (3) and sends it to the second component (4), a smart contract on the blockchain, which stores the data (5). The third component, the graphical user interface, can then be retrieved by the participant through the file storage gateway which then reads (4) the data out of the blockchain and makes it human readable (6). It is important to note, that although the user interface is provided by the file storage gateway, the blockchain data is provided and validated by the light client on the IMS. The participant reading the data stored on the blockchain does not necessarily need to be a participant with an Intelligent Metering System. It also can be the grid providers' node, for example, which then reads the data (7) to execute further logic, like

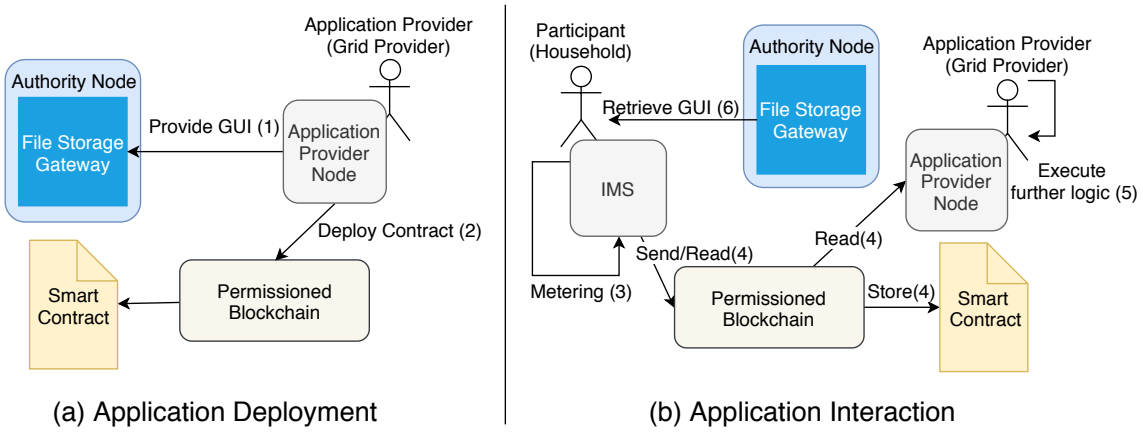


Figure 6.5: Interaction of the participants with Intelligent Metering System, blockchain, file storage gateway and application provider.

balancing the networks power usage (8). TheIMS does not necessarily only need to send data to the smart contract. The smart contract can also be used to invoke procedures at the devices connected to the Intelligent Metering System, for example, to manipulate power consumption or production.

Applications can use the underlying protocols and peer-to-peer network by their demand. One example could be a peer-to-peer trading application which uses peer-to-peer messaging in order to negotiate a price for an energy asset between two households and then utilises state channels to trade energy. The blockchain is then used after the state channel is closed. Figure 6.6 shows a possible peer-to-peer trade with the utilisation of state channels. The graphical user interface for this application again can be provided by the application provider, for example, the market operator, over the file storage gateway.

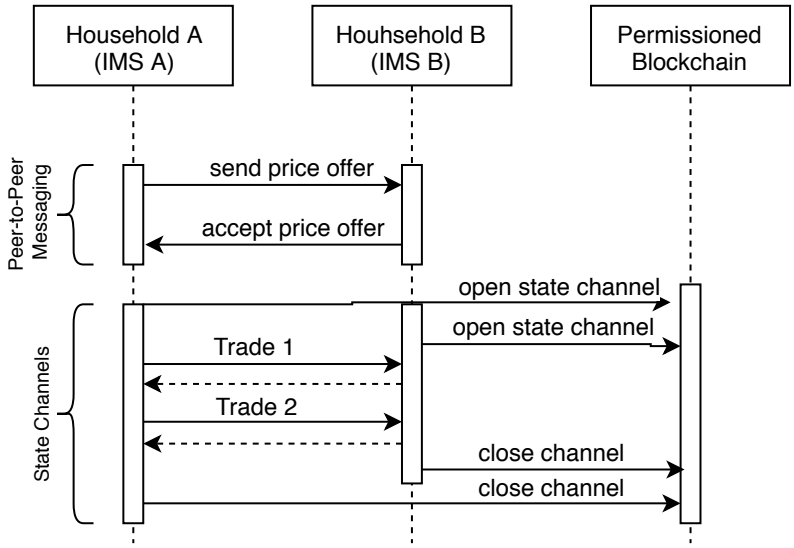


Figure 6.6: Process sequence of a peer-to-peer trade using peer-to-peer messaging and state channels.

7 Implementation

The concept shown in chapter 6 is not bound to any specific technologies and shows a way of utilising decentralised technology concepts into an architecture for a decentralised energy management system. Therefore, the following chapter shows a possible implementation of the concept described in chapter 6. Ethereum solves most of the requirements defined in section 4.1 and is the current state of the art blockchain platform technology stack. Thus, Ethereum is used as the main technology stack for implementing the proposed architecture. It is worth to mention, that the underlying technology stack not necessarily has to depend on the Ethereum stack. As long as the protocols support the core concepts proposed in chapter 6, any stack can be used. One possible alternative is Hyperledger. An unsuitable protocol stack is the native IOTA protocol because it does not support distributed processing like smart contracts. In addition to a possible implementation of the proposed concept, this chapter also shows what actually is implemented in a minimal prototype. Table 7.1 shows a summary of all technologies and protocol implementations used. In addition, it lists whether the proposed technologies have been implemented in the minimal prototype. In the following, each of those concepts and their implementation will be explained in detail.

Concept	Implementation
Peer-to-Peer Network	Ethereum Stack (Parity)
	Access Restriction via VPN
Intelligent Metering Systems	Metering Device + Embedded Gateway
Authority nodes	Two Authorities (Oli Systems and TUM) Configurable up to 6 Authorities
Peer-to-Peer messaging	Parity Whisper
State Channels	Raiden State Channel Network
Private Transactions	Parity Private Transactions (Contract & Client)
Permissioned Blockchain	Parity Proof of Authority Chain
Public Blockchain	Energy Web Chain
	Oracle Backup Contract, Parity Token Bridge
Decentralised File Storage	Go-Ethereum Swarm
	Ethereum Nameservice, Versioning Contract
Decentralised Energy Management Applications	Energy Browser

Table 7.1: Overview of possible technologies for each concept and their implementation in the prototype. The green cells show what is implemented in the prototype, the yellow ones what is implemented partly and the white ones what is not yet implemented.

Figure 7.1: Network status monitor of the peer-to-peer network

7.1 Peer-to-Peer Network

For interconnecting the participating nodes into a network, the peer-to-peer network protocol implementation of the Ethereum stack is used. Currently, there are two established Ethereum implementations, the Go-Ethereum client and the Parity client. For the peer-to-peer network, the Parity implementation is used. New nodes can find other nodes in the network by using the trusted authorities as an entry point. After connecting to an authority node, the node broadcasts its peers to the new participant. Because authority nodes are trusted parties, this way of initially connecting to the network does not harm the decentralisation of it any further. Using a different trusted node other than an authority node is also possible.

For the prototype, a small network consisting of one IMS and four authority nodes are established. Three authority nodes are hosted by the Oli Systems GmbH and one is hosted by the Technical University of Munich. In order to be able to monitor the status of the network, a network status monitor is deployed on one of the authority nodes. The monitor shows metrics like the current number of connected peers, the uptime, the propagation time for data packets through the network, or the latency of each node. Additionally, the monitor can show the current status of the permissioned blockchain, explained in section 7.2.2. The screen of the status monitor is shown in figure 7.1.

7.1.1 Access Restriction

It is possible to connect all Intelligent Metering Systems with a restricted VPN hosted by the authority nodes. Hence, a way of hosting a VPN like the concept shown in section 6.2.1 can be implemented. Owner of an IMS and authorities group up into several VPNs which then share the addresses of some nodes in order to interconnect with each other. A well known VPN implementation is the

OpenVpn[1] implementation. In order to interconnect both networks deploy a Static Node Registrar[2] such as proposed by J. Poon can be deployed [Poo18]. The network of the prototype has not been restricted yet but can be restricted as described.

7.1.2 Intelligent Metering Systems

Currently, households have many different metering devices installed. It is the idea not to necessarily replace these devices and rather add an additional device which can read power values processed by any metering device. This device then acts as a gateway for participating in the peer-to-peer network of the smart grid and comes with adequate computational power to do so. Therefore, an Intelligent Metering System is composed of two devices. Once the metering device itself and second a gateway.

Gateway

The gateway itself offers a serial interface for connecting embedded devices such as an electricity meter with it. A serial interface is a transmission interface for data over a port which supports sending and receiving bits, each bit at a time [Axe07]. In order to reduce power usage, minimise production costs and offer a system with standardised hardware and software specifications, according to requirement 4.1.2, a lightweight embedded device is used. Table 7.2 shows the technical specification of the gateway. Furthermore, the gateway has to be resilient against load peeks in power supply, weather conditions such as high humidity, heat or cold.

For the prototype, a microcontroller developed by an established manufacturer is used. The device is able to resist high load peaks and fluctuating weather conditions. For having a more stable environment, the NAND-Flash of the device is divided into two separate partitions. Each partition is hosting a copy of the operating system. In the case of a non-restorable system failure, the system can boot the other partition. This redundancy can also be used to update the operating system by updating each partition after another. After the final development of the prototype, the device will be certificated for the secure usage in the application area of the energy field.

Operating System The operating system running on the gateway has to be adapted to the technical specification of the microcontroller. For this purpose, the Yocto Open Embedded build system is used. The Yocto build system is explained in section 2.5. The operating system is based on a Linux 4.9.26[3] ARMv7 kernel. The system is a minimal system which comes with basic packages like firmware for the interfaces, networking capabilities, a bootloader for the two separate operating systems or an SSH server for remote connection to the device. In order to be able to run the proposed Protocol Layer on the device, several Middleware Technologies, defined in section 6.2.1, need to be added to the device. Therefore, two separate OpenEmbedded layers are added to the Yocto build configuration. The first layer is a generic layer defining all packages needed for running

[1] OpenVpn: https://openvpn.net/
[2] A sample implementation of a Static Node Registrar [Poo18]: https://github.com/EthereumEx/bootnode-registrar
[3] Release of the Linux 4.9.26 kernel: https://lkml.org/lkml/2017/5/3/382

CPU	NXP i.Mx6 UltraLite ARM Cortex-A7 Core ARMv7 Architecture Single Core, 528 MHz
RAM	256 Mbyte DDR3L SDRAM Bus clock 400 MHz Transfer rate 800 MT/s
Boot-Flash	4 Mbyte SPI NOR-Flash Hosts Barebox Bootloader
Flash	512 Mbyte NAND-Flash SLC-Technology Hosts two seperate Linux kernel
Interfaces	Micro SD-Card 2x 10/100BASE-TX Ethernet 1x USB-OTG1 Device 1x USB-OTG2 Host 2x UART

Table 7.2: Technical specification of the gateway.

decentralised blockchain protocols. The second layer is a system specific layer adding packages only needed for the gateway. Figure 7.2 shows a graphical overview for each layer with their included packages.

The decentralised protocols layer adds all protocol implementations necessary for the device. For the blockchain protocol, two implementations of the Ethereum light node protocol are added. First the Go-Ethereum and second the Parity implementation. Go-Ethereum, as well as Parity, provide an implementation for a peer-to-peer messaging protocol as well as a peer-to-peer protocol. Although Go-Ethereum and Parity implement the same protocol specifications they are not compatible any more. Thus one has to decide of setting up a, for example, the peer-to-peer network, the messaging, or the blockchain protocol either by using the Go-Ethereum or the Parity implementation. For this implementation example, the Parity implementation shall be used. Anyway, both clients are added to the Operating System to ensure flexibility in case the peer-to-peer network or blockchain implementation changes. During this implementation using the Raiden state channel network is proposed. Although it is possible to cross-compile the client implementation of the Raiden protocol cross-compiling all necessary dependencies would have been out of the scope of this work. Therefore Raiden is not added to the minimal prototype yet but is proposed as a possible implementation for the state channel concept.

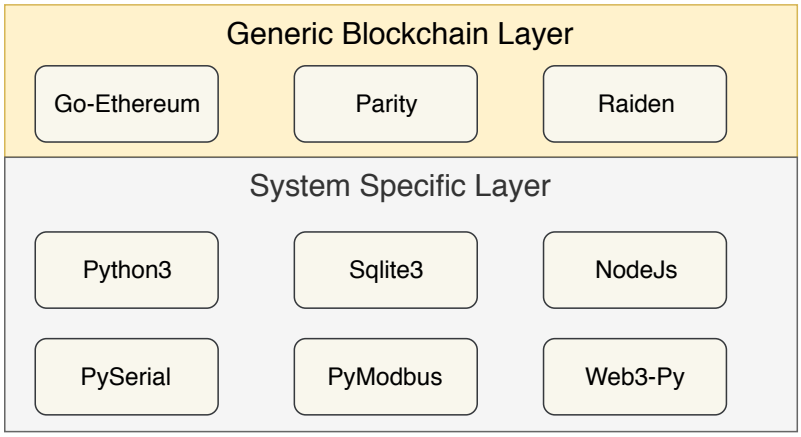


Figure 7.2: Overview of the packages added within the two additional OpenEmbedded layers

Additionally, to the blockchain layer, packages for communicating with the serial interface, running application code or intercommunicating between applications on the device are added. For an easier development and distribution of applications the Python[4] and NodeJs[5] runtimes are added. Python is available in both versions, Python 2 and Python 3. To communicate with the serial interface the Python packages *pyserial* and *pymodbus* are added. To intercommunicate with other applications or just temporary store values the *Sqlite3*[6] database engine is added. In order to have a SDK which supports the interaction with the Parity/ Go-Ethereum node, the Python library *web3-py*[7] is proposed. Although it is possible to cross-compile the library, doing so would have been out of the scope of this work. Same as Raiden, this library is not added to the prototype, yet, but is proposed as a possible implementation for an Ethereum SDK.

Software Development The easiest and fastest way to implement software for the gateway is using Python or NodeJs. It is important to mention, that each dependency used in the application has to be cross-compiled for the gateway. Because the minimal prototype does not yet support an Ethereum SDK, for the Python or NodeJs runtime the Go-Ethereum library is proposed. This library comes together with the Go-Ethereum client implementation and can be cross-compiled with the toolchain of the Go programming language[8]. One advantage of the Go programming language is that the toolchain supports the compilation of source code for any known system architecture. Hence it is easy developing native applications on the developers own machine and cross-compiling it for the architecture of the gateway.

[4]Python Runtime: http://python.org
[5]NodeJs runtime: https://nodejs.org
[6]Sqlite3 database engine: http://sqlite.org
[7]Web3-Py Package on Pip: https://pypi.org/project/web3/
[8]The golang programming language and toolchain: https://golang.org/

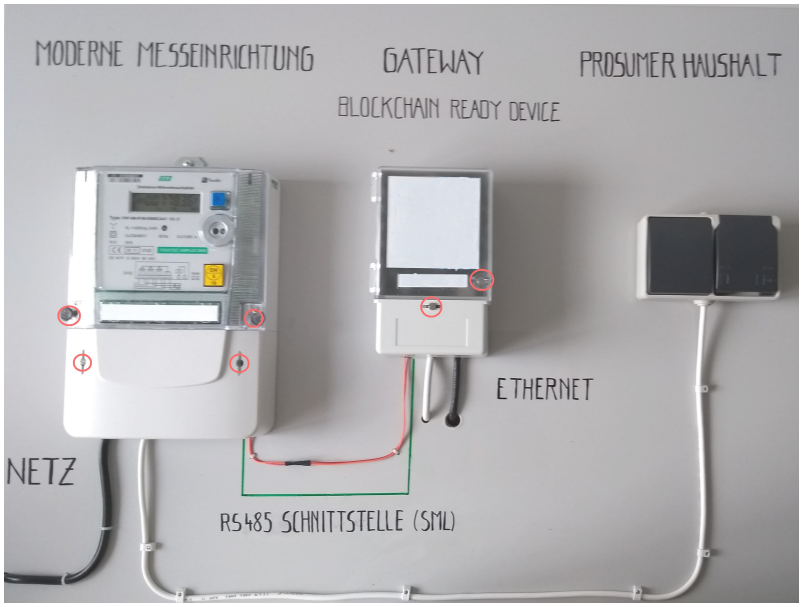


Figure 7.3: The implemented prototypical IMS consisting out of a metering device and a gateway. The red circles point the seals of the devises.

Metering Device

The metering device itself can be any common electricity meter found in a typical household. The only constraint is the support of a serial interface and a standardised communication protocol. It is important, that the metering device supports the secure sealing of the connection between the device and the gateway. Having a secure connection between the device and the gateway allows using the metering data as verified and trustworthy data. This is especially important when use cases such as peer-to-peer trading are established on top of the energy management system. Another requirement for the meter is its broadcast frequency of metering data. This should happen at least every 15 minutes.

For the prototype, the Froetec Simplex ZN91 meter [DrN] is used. It is supporting the RS485 serial standard [Kug08] and sends data packets based on the Smart Message Language Protocol (SML) [Wis08]. The SML is a communication protocol for heaving a lightweight and standardised data structure for the intercommunication of embedded devices [Wis08]. The meter is sending new metering data every 15 minutes. Furthermore, as required above a secure sealing of the connection with the gateway can be established. This is done by adding a case which is connected with the meter and the gateway which is sealed with leads. Figure 7.3 shows the connection of the gateway and the meter and how they can be sealed. On the left side, a power connection to the energy grid is established. The red wire connects the RS485 interfaces of the devices. Additionally, the figure shows a socket which either contains a producing or consuming device of a possible prosumer household. The background represents a panel under which the wires can be installed. This panel also can be sealed, thus the whole connection between the metering device and the gateway can be considered as secured.

CPU	RAM	Storage	OS
> 3.3 GHz (Single Core)	> 1GB	> 40 GB	Linux (x64 Architecture)

Table 7.3: Minimal system requirements for an authority node. Adapted from the system specification of a AWS t2.micro instance [Ama18].

7.1.3 Authority Nodes

In practice, the authority nodes can be any Linux machine, which is powerful enough to host the decentralised protocol layer. Although most of the protocol implementations are platform independent, or at least can be compiled for Windows machines, as well, the practical implementation has shown, that the implementations run far more stable on Linux machines than on a Windows machine. The minimum amount of computing power should not reach under the standard specification of a minimalistic cloud virtual machine. Table 7.3 shows an example specification which is at least required to run all protocols. Besides the authority nodes should be able to expose their IP address into the network. If this is not possible Intelligent Metering Systems will not find the node and will not be able to join the network. On top of this machine, a Parity client for the peer-to-peer network and the messaging protocol, a client for the decentralised file storage and a state channel client are deployed. The specific technologies for the messaging, decentralised file storage and state channel protocols are proposed in the succeeding section. Additionally, to the Parity client, another Parity or Go-Ethereum client should run on the node which can then connect to the public blockchain. It is recommended to host the clients as a system service which restarts automatically during a possible system failure.

For the prototype three virtual machines, meeting the minimum requirements, are hosted on the Amazon Web Service Cloud[9] by Oli Systems. The other machine is a dedicated server running Ubuntu at the Technical University of Munich with at least has the same hardware specifications. Currently, there are only the Parity client for the peer-to-peer network, the messaging protocol and a client for the decentralised file storage protocol running. Additionally, the network monitor and a blockchain explorer of the permissioned blockchain is running on one virtual machine of Oli Systems.

[9] Amazon Web Services: https://aws.amazon.com

7.2 Decentralised Protocol Layer

The decentralised protocol layer consists of several protocols proposed in the generic architecture concept. Following, possible implementations of these protocols and their application will be described.

7.2.1 Peer-to-Peer Messaging

Peer-to-peer messaging relies on a peer-to-peer network, hence it is worth to utilise the already established peer-to-peer network. The Parity client comes with an implementation of the Ethereum Whisper protocol, which runs on top of the Ethereum network. Enabling Whisper on the Parity clients allows the communication over subscribed topics which get broadcasted over the network. All the nodes in the network of the prototype have the whisper protocol enabled. However, there is no final interface definition yet for developers. Thus, the functionality of the protocol can be shown, but integrating Whisper in applications is not possible, yet.

7.2.2 Permissioned Blockchain

For the permissioned blockchain, a proof of authority blockchain has been implemented. Based on the experiences made during the implementation, a too small number of authorities will lead to a network breakdown and a too large number of authorities will lead to high latencies in block propagation and validation. Therefore an arbitrary amount of possibly six authorities has been selected, which is not too small and not too large. Listing 7.1 shows a shortened version of the chain specification file for the permissioned blockchain. Every participant connecting to the network and syncing the permissioned blockchain needs the chain specification file. The specification defines a chain named *olichain* with the Parity native *Aura consensus algorithm* [Para]. This consensus algorithm aims to choose an authority from the *list of authorities* in order to seal a new block, every three seconds. If the current validator is not available, the engine will choose the next validator from the list in a Round Robin procedure [Para]. Therefore, in an ideal scenario, the block time would always be around three seconds. A smaller *step duration* should not be set, as this can lead to multiple blocks sealed in parallel due to latency issues in the network. In the prototype, four authorities are set up. This can be extended up to six authorities. The *Genesis block*, the very first block in the chain, is defined as mined by a proof of authority consensus with a *gasLimit* around 8.000.000. The *networkID* is defined in the parameters section. The *networkID* is important for the underlying peer-to-peer network protocol in order to find nodes running on the same network. Several initial *accounts*, including the accounts of the validators, have been defined, which start with an initial amount of Ether. This is also the maximum available amount of Ether in the network because the Aura algorithm does not supply a block reward out of the box. For the prototype, an amount of ten Million Ether has been set. Ether is just used as a participation token. Whoever has Ether can use it to write on the blockchain. This means that all authority nodes only accept transactions with a *gas price* set greater than zero. In the prototype, the minimum gas price is set to 40. Therefore, the underlying cryptocurrency can be named as *POA (Proof Of Authority)* as it is only necessary for writing onto the blockchain and executing the consensus algorithm. In future, this may be changed by a currency which is coupled to some value in order to pay upkeep costs of the network and prevent participants from too many transactions. The last section of the chain specification

file describes the nodes serving as the entry points for the peer-to-peer network. Only with these nodes, a new node can connect to the existing network. In order to make development easier and to have a generic insight on the database of the permissioned blockchain, a central Blockchain Browser is deployed. This shall just be for development purpose generating an easier way to debug and overview the data transmitted on the blockchain. Figure 7.4 shows the deployed Blockchain Browser.

Listing 7.1 A shorted version of the chain specification for the permissioned blockchain.

```
{
  "name": "olichain",
  "engine": {
    "authorityRound": {
      "params": {
        "stepDuration": "3",
        "validators": {
          "list": [List of Authorities]
        }
      }
    }
  },
  "params": {"networkID": "0x62121", [...]},
  "genesis": {
    // Sealed by Authority Round
    "gasLimit": "0x800000"
   },
  "accounts": { [...] },
  "nodes": [
    "enode://KeyOfFirstAuthority@ip:port",
    "enode://KeyOfSecondAuthority@ip:port",
    [...]
  ]
}
```

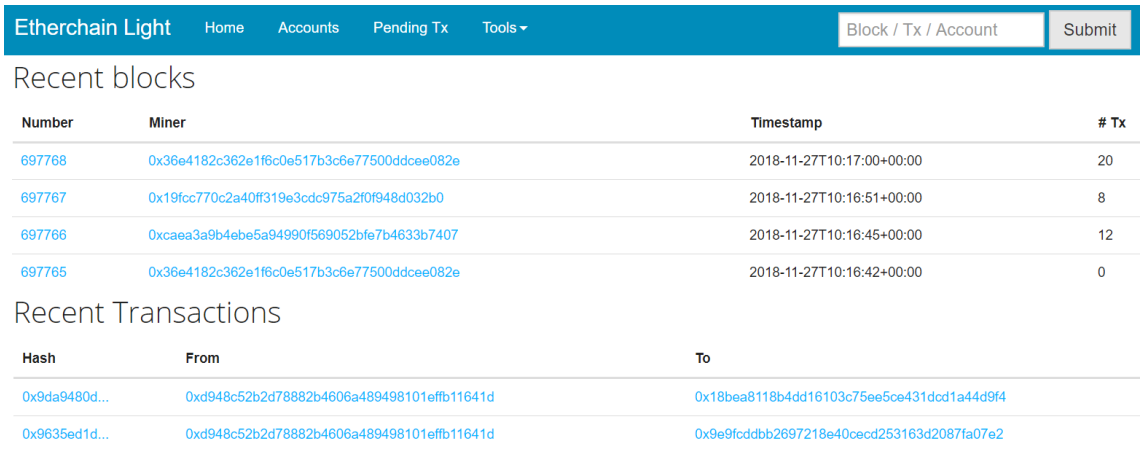


Figure 7.4: Centralised Blockchain Browser of the permissioned blockchain.

7.2.3 Public Blockchain

The concept is proposing any public blockchain sustainable and immutable enough to be hosted as a data backup. Additionally, it could be at an advantage of having tradeable established valuable tokens on the public chain. For the implementation, the proof of authority based Energy Web Chain is proposed, because it is aiming to be the standard blockchain for energy-related use cases and does not rely on a proof of work consensus. Each authority node connects with its additional running Parity client to the Energy Web Chain, using its provided chain specification file. In order to be able to write to the chain *Energy Web Tokens (EWTs)* have to be purchased. After purchasing some tokens and syncing the chain it can be used for interchain token trades or for creating data backups of the permissioned blockchain. Furthermore, the participants of the underlying grid could think about getting part of the Energy Web Foundation in order to influence decisions based on their special needs.

During the implementation of the prototype, Oli Systems is part of the Energy Web Foundation and also hosts an authority node at the Energy Web Chain. This gives the opportunity to be part of the Governance process of the Energy Web Chain while using it as the public blockchain protocol. Although the steps on how to add the public blockchain layer to the implementation of the prototype are explained, these steps have not yet been implemented in the minimal prototype.

Backup Validation

It is one utilisation of the public blockchain to make the permissioned blockchain more persistent and more tamper-resistant. This can be done by a Backup Validation contract. The Backup Validation contract is a contract which stores a hash-sum of aggregated block headers. A good example would be the aggregation of the last 100 block headers into one root hash.

Listing 7.2 shows the main functions of the contract in its Solidity code. The main function is the *setBackup* function. It takes a timestamp, the block number of the block where the aggregation starts, the block number where the aggregation ends and the block hash of the aggregated block headers. The function itself only takes a modifier which makes it executable only for a set list of *Verifiers*. The *onlyVerifiers*-modifier checks whether the transaction sent to the contract is sent by a verifier and aborts if not so. After this check, the *setBackup* function hashes the parameters into a single hash and saves it in a map of *unverifiedHashes*. Afterwards, the function checks if all verifiers have sent the same aggregated hash and if positive sets *verifiedHash* to the current hash. This type of contract is called an *Oracle* contract because only if all, or a specific amount, of verifiers, have sent the same backup the current hash is seen as the new *verifiedHash*. Doing so enables executing the smart contract in a decentralised manner. At the end of each new *verifiedHash*, the contract emits a *VerifiedBackup*-Event which can be used by the validators to again to start aggregating block headers. For the example implementation, all authority nodes can be part of the *verifiersMap* and if two third have sent the same hash, the *verifiedHash* will be executed. In case of a network breakdown or a tampered permissioned blockchain, the events can also be used to track back all issued hashes and verify still existing copies of the former permissioned blockchain.

Listing 7.2 Backup Validation with authority oracles.

```
modifier onlyVerifiers {
    require(
        verifiersMap[msg.sender] == true,
        "Only verifiers can call this function."
    );
    _;
}

function setBackup(uint timestampParm, uint blockNumberStartParm, uint blockNumberEndParm,
        string blockHashParm) public onlyVerifiers {
    verifiedDataBackup memory backup = verifiedDataBackup(timestampParm,
            blockNumberStartParm, blockNumberEndParm, blockHashParm);
    bytes32 hash = getHashedStruct(backup);
    unverifiedHashes[msg.sender] = hash;
    if(checkHashes()) {
        verifiedHash = hash;
        verifiedBackup = backup;
        emit VerifiedBackup(backup.timestamp, backup.blockNumberStart,
                backup.blockNumberEnd, backup.blockHash);
    }
}
```

The oracle concept used for the backup contract can also be used for interchanging data with the permissioned blockchains of other grids. If all authorities have submitted the same data to the Energy Web Chain, the data is accessible for other nodes of any other network. During the implementation of the minimal prototype, only the contract has been developed but not the client code issuing the transactions to the contract, yet.

Interchange of Assets

An interchange of assets can be implemented with the Parity Bridge shown in section 5.1. The Parity Bridge helps to interchange valuable cryptocurrency between the Energy Web Chain and the permissioned blockchain. The interchanges have to be validated by a group of authorities which are the same authorities than the authority nodes. Within the bridge, Energy Web Tokens get deposited on the contract of the Energy Web Chain. The bridge then issues the same amount of arbitrary ERC20 tokens on the permissioned blockchain. To do so, an authority has to deploy two bridging contracts. One contract on the Energy Web Chain and another contract on the permissioned blockchain. The deploying authority creates a *bridge.db* file which has to be shared with the other authorities. Every authority has to install the Parity Bridge client. These authorities then use the *bridge.db* file to connect with each other and validate the interchanges of assets in an authority round. Listing 7.3 shows an example of the *bridge.db* file. In order to verify that the deploying authority has deployed the correct contracts, the source code can be uploaded at the block explorers of the two chains [Parb].

Listing 7.3 Example for the *bridge.db* file shared by the authorities [Parb].

```
home_contract_address = "0xebd3944af37ccc6b67ff61239ac4fef229c8f69f"
foreign_contract_address = "0xebd3944af37ccc6b67ff61239ac4fef229c8f69f"
home_deploy = 1
foreign_deploy = 1
checked_deposit_relay = 3
checked_withdraw_relay = 4
checked_withdraw_confirm = 4
```

The Parity Bridge only handles Energy Web Tokens, thus to be exchanged into ERC20 tokens, other established tokens on the Energy Web Chain first have to be exchanged into Energy Web Tokens. The same has to be done with tokens on the permissioned chain. Because the integration of the Energy Web Chain is not implemented with the prototype the Parity Bridge is also not been implemented.

7.2.4 Decentralised File Storage

The decentralised file storage proposed in section 6.2.2 can be implemented with the Ethereum Swarm protocol. In the implementation example, each authority node hosts a Swarm node which then can be used as a file storage distributing static files. To do so, each authority node also has to provide a file storage gateway which then can be accessed by the participants. The participant then can download necessary files, for example, a graphical user interface, if he knows the hash-sum, called *swarm hash* [Eth] of the file. This has been done for the prototype as described. Listing 7.4 shows the upload of a file and its download into a separate file, with the returned swarm hash.

Listing 7.4 Interaction with one of the swarm nodes.

```
$ swarm up welcome.md
> 99863a4d8b85484c2aa63ddc6d178390c7ab0c6d4bece27717df20a14b00f1ae
$ swarm down bzz:/99863a4d8b85484c2aa63ddc6d178390c7ab0c6d4bece27717df20a14b00f1ae dwnld.md
```

Ethereum Nameservice

In order to make distributed files resolvable with an understandable identifier, Ethereum Nameservice contracts can be deployed on the permissioned blockchain. Furthermore, the contract addresses have to be linked with the Swarm clients on the authority nodes. Application providers can then register their applications and its parts. Each application gets its own node. Each subnode of this node can be seen as an application part. A possible example is one subnode for the smart contracts, one for the files necessary for the IMS and one for the files for the graphical user interface. An identical identifier for the graphical user interface of an application developed by Oli Systems, for example, could be *gui.exampleapp.oli*. This name can then be resolved over the file storage gateway

of an authority node. A nameservice also gives the participants the possibility of recalculating the Swarm-Hash and check if they received the expected files. Implementing an Ethereum Nameservice has not been done yet for the prototype.

7.2.5 State Channels

As shown in the analysis in section 5.1, state channels are a protocol for aggregating transactions apart from a blockchain protocol. The currently most famous state channel implementation is the Raiden implementation. Therefore, this implementation suggests using the Raiden protocol. As mentioned in the sections above, cross-compiling Raiden would be out of the scope of this thesis. However, this section shows how state channels could be implemented. Before being able to run Raiden, the Raiden contracts have to be deployed on the permissioned blockchain. Because Raiden only supports ERC20 tokens to be transferred over the network, an ERC20 token contract has to be deployed for every use case using the state channel protocol. This token can then be added to a new token network and used for off-chain transactions. The Raiden network can be enabled by running a Raiden client on each participating node.

7.2.6 Private Transactions

As explained in section 5.1.5, private transactions are basically a way of transacting encrypted state on the blockchain. With the Parity private transaction protocol, a public contract containing the encrypted state and encrypted bytecode of the contract which deals with that state has to be created. This contract has to be deployed by one of the participants using the contract. Therefore a public contract with its private encrypted contract inside has to be deployed on the permissioned blockchain. Inside the public contract, validators have to be defined. Because participants already trust the authority nodes, those nodes can act as validators as well. The keys needed to decrypt the content of transactions are stored in a secret store. This secret store has to be shared between the participants of the private transactions. Since participants already use the authority nodes as validators, the secret store can be deployed on the authorities, as well. After the Parity nodes have been configured for accessing the deployed contracts and using private transactions, the participants can issue an encrypted transaction. [Parc]

7.3 Decentralised Energy Management Applications

On top of the architecture, the application provider can develop applications utilising the supplied protocols. As mentioned in section 6.2.4, an application typically is divided into three parts. One part which runs on the IMS, one part which is distributed to the user and can be accessed for example with a Browser, and a third part which resides on the decentralised protocol layer, such as a smart contract. There are several libraries and application interfaces available supporting the development of the application provider. For the first part, the Go-Ethereum client library is available for the Ethereum blockchain protocol and whisper. For the Raiden protocol pure HTTP requests or the Raiden Python client code, if cross-compiled, can be used. Private transactions can currently be sent only over raw HTTP calls. However, Parity supplies some tools to encrypt the data with the senders key, sent in the private transaction. For the second part common Ethereum SDKs like Web3Py or

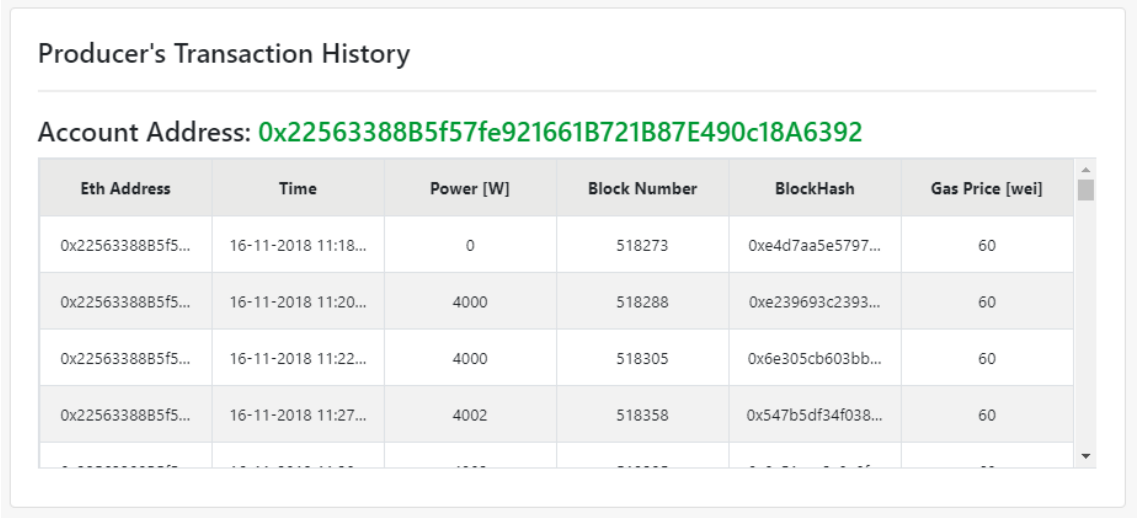


Figure 7.5: Transaction history of the production data sent by the IMS developed for the minimal prototype. [Yah18]

Web3Js can be used. These libraries cover all protocol implementations necessary. For application logic residing on the permissioned or public blockchain, Ethereum smart contracts can be deployed which are written in Solidity[10] programming language. In order to distribute their applications, application providers can use the Ethereum name service on the permissioned blockchain to register their application and its parts.

7.3.1 Energy Browser

During the implementation of the minimal prototype, an Energy Browser[11] has been deployed on top of the architecture. The Energy Browser is a decentralised dashboard for browsing energy data in a power grid developed by Oli Systems[Yah18]. The Energy Browser itself is a JavaScript implementation based on the Web3Js library. The application is reading transactions sent to two smart contracts by the participants of the power grid. Currently, the blockchain data read by the JavaScript implementation is provided by an authority node. The transactions, in general, contain the amount of energy produced or consumed. If energy is consumed by a participant, this transaction is sent to a contract storing all consumption values for each participant. If energy is produced by a participant, this transaction is sent to a contract storing all production values. The smart contracts are deployed on the permissioned blockchain. The statical files for the graphical user interface of the Energy Browser are delivered over a gateway of the distributed file storage. For the Intelligent Metering Systems, a client code is implemented. This code reads the metering data from an SQL database and sends it either to the production or consumption contract. Because the meter used for the IMS only provides new information every 15 minutes, the transactions are sent in this time interval. Figure 7.5 shows the transaction history of the production data sent by the IMS to the production contract. Figure 7.6 shows a screenshot of the live monitor of the Energy Browser.

[10] Solidity documentation: https://solidity.readthedocs.io/en/v0.5.0/
[11] Available at http://Oli-Chain.com

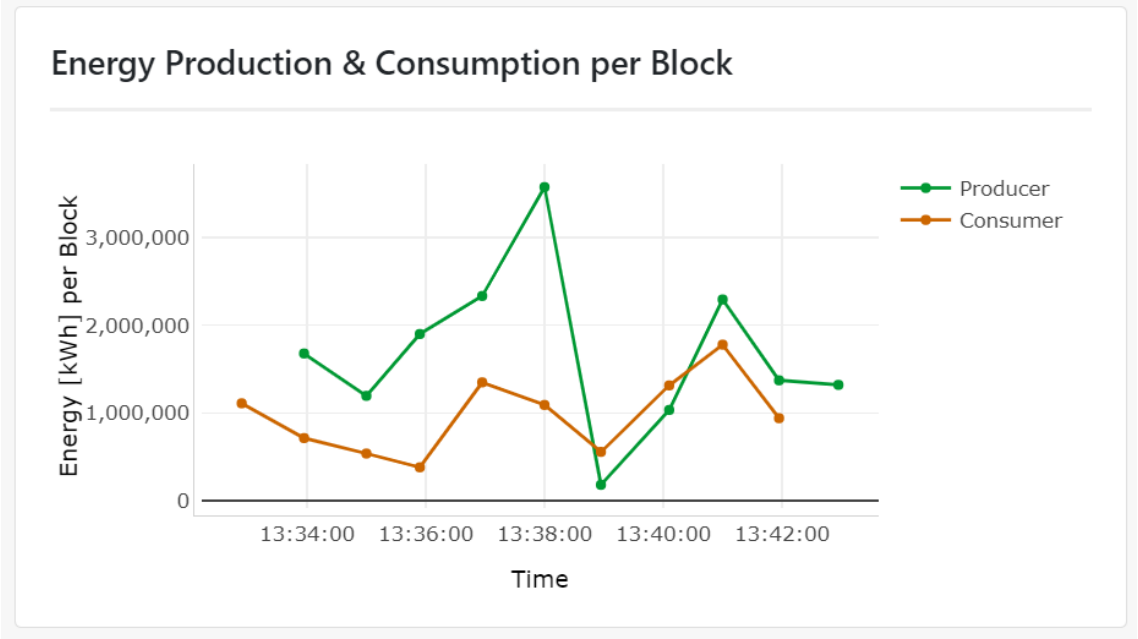


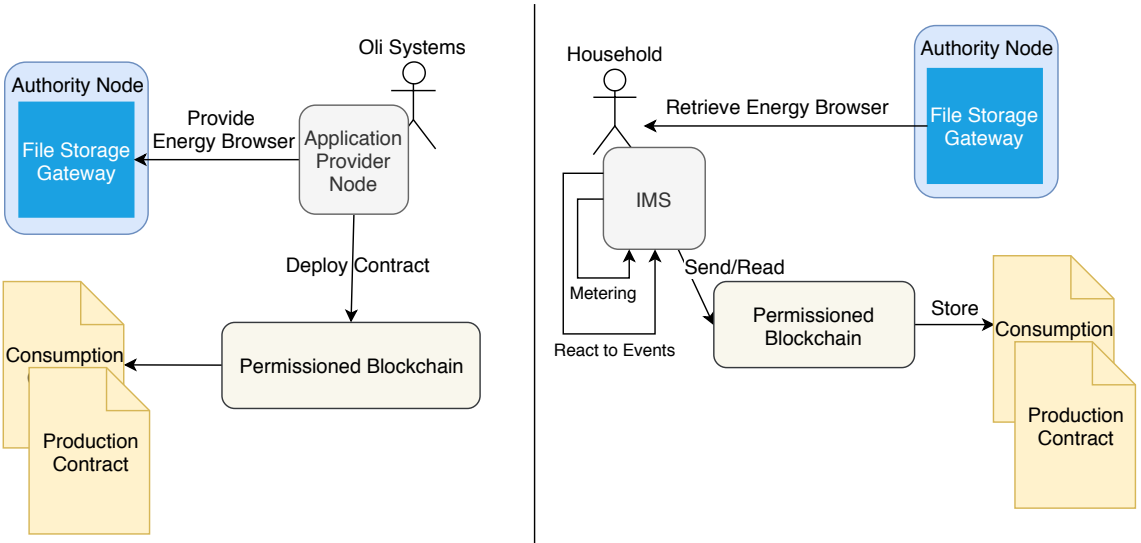
Figure 7.6: Screenshot of the live monitor of the Energy Browser [Yah18]

Furthermore, an additional client code is deployed on the IMS. In addition to storing the data sent to the contract, the contract issues an event for every new entry containing the currently submitted power values. This enables participants reading the blockchain to react to this event. The client code deployed on the IMS also does this and shows in a prototypical way how those devices could react to this events. Figure 7.7 shows a console log of the IMS where received events from the Production Contract are printed and could possibly be handled, afterwards. A possible reaction could be a message to the consuming devices of a household in order to switch them off, if the energy grid does not offer enough produced electricity. Figure 7.8 shows the interaction between the several actors from figure 6.5 for the concrete example of the Energy Browser, where Oli Systems is providing the application and a Households sends its production and consumption values to the contracts.

Additionally to the things done during the prototype, the application could also use the deployed Ethereum Nameservice for distributing the application files and the versioning contract. This gives the participants a way to verify the files they received by a file storage gateway. In favour of reducing the confidence of trust on single parties, the blockchain data supplied by an authority node could be changed into a use case in which the data is read directly from the blockchain client on the IMS.

```
_____________________________________
Receiving Event:
Block Hash: 0x705cd491785506149b29cccf91b7ffb322463f50f064674aea698170d8e0ffec
Block Number: 875017
Oli Address: 0x0ED92bbEdC9881ad8FCa2baB936aBb92040Af988
Energy Timestamp: 1544438415
Energy Amount: 1095
_____________________________________
Receiving Event:
Block Hash: 0x705cd491785506149b29cccf91b7ffb322463f50f064674aea698170d8e0ffec
Block Number: 875017
Oli Address: 0x136cB1DA7198725EEb51e23C42b0A8eA6d9002B1
Energy Timestamp: 1544438415
Energy Amount: 2954
_____________________________________
Receiving Event:
Block Hash: 0x705cd491785506149b29cccf91b7ffb322463f50f064674aea698170d8e0ffec
Block Number: 875017
Oli Address: 0x15b8E202750A76968cb4842ef71cc3fE8e2dCe8a
Energy Timestamp: 1544438415
Energy Amount: 720
_____________________________________
Receiving Event:
Block Hash: 0x9664f31e9f40eac02291c6673c391be4b9fe2358c56e80cf92a22648bf2b828c
Block Number: 875020
Oli Address: 0x22563388B5f57fe921661B721B87E490c18A6392
Energy Timestamp: 1544438433
Energy Amount: 100
_____________________________________
```

Figure 7.7: Received events from the IMS which are sent by the Production Contract of the Energy Browser.



(a) Application Deployment (b) Application Interaction

Figure 7.8: Interaction between the several actors for the concrete example of the Energy Browser

8 Evaluation

With the generic architecture, a concept has been introduced which tries to solve the problem of a decentralised energy management system. Furthermore, a possible implementation of the concept with a minimal prototype has been proposed in the previous chapters. Same as the existing solutions those artefacts shall be evaluated against the requirement implications identified in section 4.1.

8.1 Functional Requirements

The functional requirements describe the basic characteristics a decentralised energy management system should provide. Following the proposed architecture should also provide those characteristics.

8.1.1 FR1: Connecting Participants

The underlying peer-to-peer network interconnects all participants of one grid. Additionally, communication capabilities such as peer-to-peer messaging and the overlaying blockchain protocols have been proposed in the concept. This allows all participant to communicate with each other and not to rely on a third party when it comes to collaboration between several members of the grid. Furthermore, participants can address other peers over their unique address within the network or specific messaging topics. Thus, the concept provides a means to interconnect and establish communication between participants of a single network. Intercommunication over the borders of the local grid is also possible, but only with the utilisation of the public blockchain protocol. Moreover, participants need the authority nodes in order to collaborate with grid members outside of the local power grid. These features fulfil requirement FR1. The feasibility of requirement FR1 has been shown within the proposed implementation and has been realised in the prototype for a small single peer-to-peer network.

8.1.2 FR2: Data Storage and Collection

The generic architecture proposal proposes two concepts for storing and collecting data. First by using blockchain technology and second by using a decentralised file storage. These two concepts provide a means for the participants to transparently store data and make it accessible for other members of the grid. Blockchain technology is suited for mapping power values into digital assets like cryptocurrencies. Although standardised data formats have not been proposed, standardised formats can be established by deploying contracts exposing a unique interface to the user. An example of a unique interface is the introduced monitoring use case, where the interface defines the data format which all participants have to adapt to. The feasibility of requirement FR2 has been

shown within the proposed implementation by utilising the provided blockchain protocols to store energy data and visualise them with the decentralised Energy Browser. This also has been realised within the prototype. As shown during the prototype implementation, querying the blockchain might be difficult with the IMS. Reading a large number of blocks requires a high amount of IO data which might not be fulfilled by the embedded device. Therefore, the most promising energy management use cases are sending data to the blockchain and interact with the devices over smart contracts. It is recommended that querying blockchain data should be done over a remote node with more computing and IO power or over static files provided over the decentralised file storage.

8.1.3 FR3: Open and Extendible Platform

The architecture proposal itself provides an open and extensible platform for the participants of the energy management system. Application providers can choose which protocols they want to integrate within their applications without any restriction. The only requirements an application provider should fulfil is the necessary knowledge about the architecture and how decentralised applications work. By integrating the Energy Browser application into the platform the extendibility of the platform is shown. Because all the technologies used for the implementation are accessible for everyone, application developers can get familiar with those technologies. Developers can start developing for the decentralised energy management system, which fulfils FR3.

8.2 Non-Functional Requirements

The main characteristics of a decentralised energy management system and the characteristic of a decentralised energy grid implies several non-functional requirements, as well. Whether those requirements are fulfilled is explained in the following sections.

8.2.1 NFR1: Decentralisation

The generic architecture proposal tries to be as decentralised as possible and centralised as necessary. Despite using decentralised concepts and technologies the architecture comes with some trade-offs The first trade-off is that the Intelligent Metering Systems are only running light node protocols of the permissioned blockchain and are dependent on the authority nodes. Second, in order to save computational resources and increase throughput, the consensus mechanism is passed to a group of authorities. Third, the IMSs are not capable of hosting their own file storage node, meaning participants have to access one of the provided gateways. As long as the architecture is dependent on a number of nodes and not from a single or individual node, the architecture is still decentralised enough for the energy management system. With the publishing of this thesis and several development tasks, the architecture and technology decisions of the underlying platform are made transparently available. Therefore, there is no dependence from a single entity which hosts the knowledge the platform is built upon. During the implementation of the prototype, some centralised drawbacks have to be taken into account. Those drawbacks have to be removed when publishing a public and productive energy management system. The implemented prototype is still centralised because one party is hosting most of the authority nodes. The blockchain data used for the Energy Browser use case is not read directly from the IMS. Although extending the use case to a

scenario where the blockchain data is read from the node on the IMS, which reduces dependencies on single parties, the IMS has still to trust the number of authorities in the network. In conclusion, the architecture has no single point of failure and no single central party operates and maintains the system. The system reflects the decentralised energy architecture into a digitalised one.

8.2.2 NFR2: Availability

With the fulfilment of requirement NFR1 also a high available system has been established. The decentralised and independent hosting of transaction logs, applications and data files makes the system resistant against failures and makes it highly available. One disadvantage with the proposed implementation of using an Ethereum based peer-to-peer network and the proof of authority consensus mechanism is its requirement for a sufficient network size. If the amount of participants is too small single participants can lose network connection which reduces availability for the participant and the network itself. This is especially a problem if an authority node loses connection. In proof of authority, the node will still validate new blocks which results either in the creation of a side chain or in the lack of the ability for the node to reconnect to the network. This has to be considered when setting up the architecture for production use. The prototype also provides compensation mechanisms in case the Intelligent Metering System loses internet connection. In such an event the node stores the data and requests until the connection has been restored. One drawback can be the limited storage capacities of the IMS which can lead to a loss of data if the device is disconnected for too long.

8.2.3 NFR3: Scalability

State channels, Whisper and the utilisation of a permissioned consortium blockchain solve the scalability issue in terms of transaction throughput. Still, a large number of participants can be integrated into the system by deploying multiple instances of the permissioned blockchain. The implementation of the prototype has shown, that storing large amounts of data can be a problem, especially with blockchain technology. Since data is not partitioned over multiple nodes and every node has a single copy the storage upkeep for syncing the permissioned blockchain and even more, the public Energy Web Chain might increase significantly during uptime of the system. Therefore, NFR3 is only solved for the first dimension of scalability but not for data capacities.

8.2.4 NFR4: Performance

The architecture proposal tries to solve performance issues by utilising a permissioned blockchain and swapping load from the blockchain protocol to state channels or messaging protocols. As some applications do not need a persistent transaction log performance issues are solved by state channels and the peer-to-peer messaging protocol. However, if a persistent transaction log is required blockchain protocols still suffer from performance issues. During the development of the prototype, with the utilised proof of authority consensus, an average block time of *4.72* seconds with an average propagation time around *40* milliseconds and a gas limit scalable up to *80.000.000* gas per block has been achieved. This means, that the network can handle every *4.72* seconds around *500* transactions with a gas size of *160.360* every *4.72* seconds. *160.360* is the gas amount needed for sending power

and consumption values of a participant to the smart contract of the Energy Browser. Assuming that participants want to write on the blockchain at least every 15 minutes enables 100 participants to write 951 transactions each in a 15 minutes time interval. Although this expects an evenly distributed transaction throughput the number of possible transactions for each participant is still large enough to host multiple applications at ones. The following equations show the calculation of the 951 transactions per participant.

$$AvailableGas = \frac{15min * 60\frac{sec}{min}}{4.72sec} * 80.000.000gas \approx 15.254.237.288gas \tag{8.1}$$

$$AvailableTransaction = \frac{AvailableGas}{160.360gas} \approx 95.125transactions \tag{8.2}$$

$$ParticipantsTransactions = \frac{AvailableTransactions}{100participants} \approx 951\frac{transactions}{participant} \tag{8.3}$$

Performance still can be an issue if lightweight nodes want to read data included in the entire blockchain. Since the IMS is limited in its computational capacities, this might take much time and has to be considered when building applications utilising the permissioned blockchain.

8.2.5 NFR5: Data Protection, Security and Privacy

Exchanging data in a closed network with a permissioned blockchain makes data only available to users who are part of the network. However, using a network contract which registers all participants exposes their identifiers to all other participants and may result in privacy issues. Participants can come up together about which data is shared and which data should not be shared among each other. Furthermore, each participant on his own can decide to which extent he wants to reveal his personal data. Encrypting private state and peer-to-peer messaging enable communication aside from the traceable blockchain protocol. Since the exchange of encrypted state always needs some third party peers, using concepts like private transactions can result in lack of decentralisation and should be considered. Because data is only sent with a pseudonym this can be seen as impersonal data which conform to possible regulations. Nevertheless, application providers need to be sensitive with the contact of the participants' data. Private data may be exposed which makes it necessary to only store data which is really needed on the blockchain.

8.2.6 NFR6: Data Persistence

Permissioned blockchains do have the drawback of being more tamper-prone than public blockchains. In contrast to other blockchain solutions, the proposed architecture does provide a means to make data stored on the blockchain more persistent. The implementation proposes a possible backup contract. It has to be considered, that only data hashes are stored on the public blockchain, requiring at least one node with a valid data copy to be able to restore the data. Furthermore, the proposed implementation does not cover a solution on how to restore the data automatically and how to handle reorganisations of the chain.

8.2.7 NFR7: Auditability

As long as applications only utilise the permissioned blockchain protocol, such as the Energy Browser, all transactions are fully auditable. In the case of the utilisation of state channels or whisper, auditability is not provided anymore. Therefore, out-sourcing transactions off the blockchain always come with a trade-off in auditability. Because application providers might prefer a performant and not always traceable application requirement NFR7 may not be fulfilled all the time. Moreover, NFR7 is only fulfilled in specific contexts, where auditability is more important than data privacy and transaction throughput.

8.2.8 NFR8: Hardware Requirements

The architecture concept proposes the hosting of the blockchain on powerful authority nodes which enables the usage of light nodes in the households. Therefore, lightweight devices can mostly be used within the network, which is a condition of requirement NFR8. In the implementation, the do-ability of lightweight devices reading metering data and sending it to the energy management system is shown by using a traditional meter and a lightweight gateway. By using a device for the gateway from a manufacturer who is specialised in producing devices exposed to strong weather conditions and load peeks this requirement is also met. The prototypical Intelligent Metering System realises the concept and its implementation proposal successfully. The network mostly consists of embedded devices hard to be exposed and possibly being installed in an environment with strong weather conditions. Thus requirement NFR8 is fulfilled.

8.2.9 NFR9: Self-Sustainability

The architectural concept is only proposing the creation of a governmental entity but is not proposing any economic incentive. Therefore, the only incentive provided by the architecture is the interest of the participants to keep the platform up and running. Thus NFR9 is only fulfilled if the implemented governance structure rewards application providers, authorities and other parties keeping the system alive with an economic incentive.

9 Conclusion and Outlook

In this thesis, a generic blockchain architecture for a decentralised energy management system has been proposed. With an energy management system, participants of a power grid can interconnect with each other in order to make energy consumption and production more efficient and save costs. Especially the energy transition with its rise of new participants acting as unpredictable prosumers makes an energy management system, which maps the new decentralised power grid, more and more important. Therefore, the architecture interconnects all participants of a local power grid and beyond its borders. It is able to act as a distributed state machine which can store and provide all data necessary to fulfil one's own energy management policies and therefore establishes an open and extensible platform.

After getting familiar with currently related researches and necessary background knowledge, it was the first step of the thesis to identify the most important requirements requested by a DEMS. The work also discussed whether an architecture approach which utilises blockchain technology is necessary and contrasted it against the utilisation of common database management systems.

As a second step, several possible decentralised technologies and concepts have been analysed. It was the goal to find already existing concepts which could help to fulfil the underlying requirements of a decentralised energy management system. Thus, already existing blockchain solutions implementing energy-related use cases have been examined. A classification against decentralised characteristics and an evaluation of their fulfilment of the identified requirements has been done before the development of the architectural concept.

Based on the acquired knowledge an architectural concept for a DEMS has been presented. The proposed architecture is based on a peer-to-peer network of lightweight nodes together with a group of authority nodes. The main part of the network is a permissioned blockchain, which shall fulfil the identified requirements together with other protocols responsible for scalable and private communications between the participants. In order to interconnect with other local power grids, the utilisation of a public blockchain has been introduced. Finally, the architecture is extendible with decentralised applications such as an energy monitor.

In order to show the feasibility of the concept a possible implementation of the proposed concept together with a minimal prototype have been introduced. The implementation proposes several technologies of the Ethereum platform, realising the concepts introduced in the generic architecture. In order to have a lightweight device which is cheap in production costs and energy usage, each network node consists of a gateway and a metering device. For this purpose, a minimal operating system has been implemented which runs on the gateway. To show the suitability of the proposed implementation an already developed decentral energy monitor application has been deployed on top of the architecture stack.

In the last step, as well as the preexisting solutions, the proposed generic architecture together with its implementation and minimal prototype have been evaluated against the identified requirements. While all of the functional requirements have been fulfilled, the architecture does not provide a solution in respect to scalability for data capacities and results in a trade-off between auditability and privacy requirements. Still, the architecture fulfils the needs of a decentralised energy management system and can be seen as a reference architecture for a standardised platform deployed in a local power grid.

Outlook

As blockchain technologies are still under heavy development and complex use cases are therefore difficult to bring in a productive state, this work only sets the foundation of research possible in the field of energy-related blockchain solutions. Only when a platform, such as proposed in the developed thesis, is established as the foundation for a smart power grid, application providers can start bringing production ready use cases in the energy field. Thus improving the proposed concept and focusing on a production-ready implementation which becomes the standard for underlying power grids shall be the main goal of possible future works.

The first improvement possible is the proposal of a more distinct governmental structure and rules for economic incentives within the participants. This has not yet been done by the previous work, to a sufficient extent. Thus, in the future, the architecture needs a means for fine granular permission management and on-chain governance implementations.

Another improvement could be the extension of the proposed architecture with a multi-chain technology, such as Polkadot[1]. Although bridging contracts and oracles try to enable interchain-communication, multi-chain concepts enable the way for a more efficient communication with more possible use cases.

Because the proposed concept does not provide a solution for scalability in terms of data a possible future work could be the investigation of this requirement in more detail. One possible approach would be the application of decentralised databases, whereas the permissioned blockchain is only used to store data hashes for validation. Another approach could be the utilisation of partitioning the data stored within the blockchain. Investigating possible approaches against their scalability characteristics and their trade-off against decentralisation would be valuable for a future improvement of the architecture proposal.

Finally, one could evaluate the proposed implementation with real operational data and extend the minimal prototype with the missing technologies. This will further prove the feasibility of the architecture and pave its way of becoming a standard reference architecture for not only energy-related use cases.

[1] Polkadot: https://polkadot.network

Bibliography

[Ama18] Amazon.com Inc. *Amazon EC2 Instance-Typen – Amazon Web Services (AWS)*. 2018. URL: https://aws.amazon.com/de/ec2/instance-types/ (visited on 11/27/2018) (cit. on p. 87).

[Axe07] J. Axelson. "Serial Port Complete: COM Ports, USB Virtual COM Ports, and Ports for Embedded Systems". In: *Lakeview Research* (2007). DOI: 10.1145/1348478.1348477 (cit. on p. 83).

[BCNN18] D. Burnett, R. Coote, C. Nevile, G. Noble. *Enterprise Ethereum Client Specification V2*. Tech. rep. Enterprise Ethereum Alliance, 2018. URL: http://entethalliance.org/wp-content/uploads/2018/10/EEA_Enterprise_Ethereum_Client_Specification_V2.pdf (cit. on pp. 33, 34, 54, 72).

[Ben18] J. Benet. *IPFS-Content Addressed, Versioned, P2P File System (DRAFT 3)*. 2018. DOI: 10.1145/2980137.2980141. arXiv: 1407.3561. URL: https://ipfs.io/ipfs/QmR7GSQM93Cx5eAg6a6yRzNde1FQv7uL6X1o4k7zrJa3LX/ipfs.draft3.pdf (visited on 11/29/2018) (cit. on p. 27).

[Big18] BigchainDB. "BigchainDB: The blockchain database". 2018. URL: https://www.bigchaindb.com/whitepaper/bigchaindb-whitepaper.pdf (cit. on pp. 26, 27, 51).

[BKT18] E. Buchman, J. Kwon, Z. M. Tendermint. *The latest gossip on BFT consensus*. Tech. rep. 2018 (cit. on pp. 48, 65).

[BM17] P. Bains, N. Murarka. "Bluzelle: A DECENTRALIZED DATABASE FOR THE FUTURE". In: (2017), pp. 1–42. URL: https://s3.amazonaws.com/bluzelle-craft-private-storage/Bluzelle-Whitepaper-English.pdf?mtime=20180531165140 (cit. on p. 51).

[BN73] M. Bellare, G. Neven. "Identity-Based Multi-signatures from RSA". In: *Topics in Cryptology – CT-RSA 2007*. Springer, Berlin, Heidelberg, 1973, pp. 145–162. ISBN: 978-3-540-69327-7. DOI: 10.1007/11967668_10. URL: http://link.springer.com/10.1007/11967668_10 (cit. on p. 53).

[Bra] Brainbot. *Raiden Network*. URL: https://raiden.network/101.html (visited on 11/30/2018) (cit. on p. 53).

[Bra18] Brainbot. "Raiden Specification Documentation Release 0.1". In: (2018). URL: https://media.readthedocs.org/pdf/raiden-network-specification/latest/raiden-network-specification.pdf (cit. on p. 52).

[BSC18] T. Blummer, M. B. Sean, C. Cachin. *An Introduction to Hyperledger*. Tech. rep. 2018. URL: https://www.hyperledger.org/wp-content/uploads/2018/07/HL_Whitepaper_IntroductiontoHyperledger.pdf (cit. on p. 48).

[But13] V. Buterin. *A NEXT GENERATION SMART CONTRACT & DECENTRALIZED APPLICATION PLATFORM*. 2013. URL: https://cryptorating.eu/whitepapers/Ethereum/Ethereum_white_paper.pdf (cit. on pp. 25, 26).

[But17a] V. Buterin. *A Proof of Stake Design Philosophy*. 2017. URL: https://medium.com/@VitalikButerin/a-proof-of-stake-design-philosophy-506585978d51 (visited on 10/11/2018) (cit. on p. 22).

[But17b] V. Buterin. *The Meaning of Decentralization – Vitalik Buterin – Medium*. 2017. URL: https://medium.com/@VitalikButerin/the-meaning-of-decentralization-a0c92b76a274 (visited on 08/14/2018) (cit. on pp. 17, 18).

[Cac16] C. Cachin. "Architecture of the Hyperledger Blockchain Fabric". In: *Online* (2016). ISSN: 18688969. DOI: 10.4230/LIPIcs.OPODIS.2016.24. arXiv: arXiv:1603.07351. URL: http://bytacoin.io/main/Hyperledger.pdf (cit. on p. 48).

[CL02] M. Castro, B. Liskov. "Practical byzantine fault tolerance and proactive recovery". In: *ACM Transactions on Computer Systems* 20.4 (2002), pp. 398–461. ISSN: 07342071. DOI: 10.1145/571637.571640. arXiv: arXiv:1203.6049v1. URL: http://portal.acm.org/citation.cfm?doid=571637.571640 (cit. on p. 23).

[Con17] Consensys. "GridPlus". In: *Whitepaper* (2017). URL: https://gridplus.io/assets/Gridwhitepaper.pdf (cit. on pp. 62, 63).

[Dai98] W. Dai. "b-money, 1998". In: (1998) (cit. on p. 25).

[DIN18] DIN-Normenausschuss Grundlagen des Umweltschutzes (NAGUS). *DIN EN ISO 50001 - Energy management systems - Requirements with guidance for use*. 2018 (cit. on p. 28).

[DKE10] DKE VDE. *Projekt E-Energy - Smart Grids made in Germany*. 2010. URL: https://teamwork.dke.de/specials/7/Wiki-Seiten/Homepage.aspx (visited on 06/27/2018) (cit. on p. 29).

[DrN] Dr.Neuhaus Telekomunikation. *SIMPLEX 91/92 Benutzerhandbuch*. Tech. rep. URL: https://www.sagemcom.com/fileadmin/user_upload/Energy/Dr.Neuhaus/Products/Zahler/FROETEC_SIMPLEX_ZN/Benutzerhandbuch_Simplex-Z.pdf (cit. on p. 86).

[EHS15] C. Eksin, A. Hooshmand, R. Sharma. "A decentralized energy management system". In: *2015 European Control Conference, ECC 2015*. 2015, pp. 2260–2267. ISBN: 9783952426937. DOI: 10.1109/ECC.2015.7330875 (cit. on pp. 15, 37).

[Eth] Ethereum. *Swarm Hash · ethereum/wiki Wiki*. URL: https://github.com/ethereum/wiki/wiki/Swarm-Hash (visited on 12/11/2018) (cit. on p. 92).

[Eth18] Ethereum. *Whisper PoC 2 Protocol Specification*. 2018. URL: https://github.com/ethereum/wiki/wiki/Whisper-PoC-2-Protocol-Spec (visited on 11/30/2018) (cit. on p. 53).

[Fis18] G. Fischer. *Bilanzkreismanagement Strom und Grundlagen MaBiS*. 2018 (cit. on p. 41).

[FK17] A. Filatov, D. Kochin. *TiesDB Technical description (eng) v0.1.1*. 2017. URL: https://docs.google.com/document/d/1UwaaUgunnrFpL6jetA_DdNLQsbbqBx1HLcln07kLrUw/edit (cit. on pp. 17, 52).

[Fla18] E. Flanagan. *The Architecture of Open Source Applications (Volume 2): The Yocto Project*. 2018. URL: https://www.aosabook.org/en/yocto.html (visited on 11/24/2018) (cit. on pp. 30, 31).

[För17] F. Förster. *Masterthesis Disruptive Konzepte vernetzter*. 2017 (cit. on p. 58).

[Gre15] G. Greenspan. *Avoiding the pointless blockchain project*. 2015. URL: https://www.multichain.com/blog/2015/11/avoiding-pointless-blockchain-project/ (visited on 07/17/2018) (cit. on p. 43).

[HAS14] A. Hooshmand, B. Asghari, R. K. Sharma. "Experimental demonstration of a tiered power management system for economic operation of grid-tied microgrids". In: *IEEE Transactions on Sustainable Energy* 5.4 (Oct. 2014), pp. 1319–1327. ISSN: 19493029. DOI: 10.1109/TSTE.2014.2339132. URL: http://ieeexplore.ieee.org/document/6883231/ (cit. on p. 37).

[HHH+18] S. Hartnett, C. Henly, E. Hesse, T. Hildebrandt, C. Jentzch, K. Krämer, G. Macdonald, J. Morris, H. Touati, A. Trbovich. *The Energy Web Chain*. Tech. rep. Energy Web Foundation, 2018. URL: http://www.energyweb.org/papers/the-energy-web-chain (cit. on pp. 23, 37, 56).

[Hoe14] J. Hoelscher. "Diffused Art and Diffracted Objecthood: Painting in the Distributed Field". In: 2014 (cit. on p. 18).

[How93] C. R. Howell. *Securing mechanism for an electricity metering device*. 1993 (cit. on p. 69).

[Jen18] C. Jentzsch. *Slock.it IoT Layer – Slock.it Blog*. 2018. URL: https://blog.slock.it/slock-it-iot-layer-f305601df963 (visited on 09/06/2018) (cit. on p. 64).

[Joh] N. Johnson. *Ethereum Nameservice documentation 0.1*. URL: https://docs.ens.domains/en/latest/ (visited on 11/29/2018) (cit. on pp. 50, 51).

[Kal10] J. Kals. *Betriebliches Energiemanagement: Eine Einführung*. Kohlhammer Verlag, 2010, 260 S. ISBN: 978-3-17-021133-9 (cit. on pp. 28, 29).

[KJJD18] S. Kux, C. Jentzsch, S. Jentzsch, P. Depraz. *INCUBED A trustless stateless incentivized remote node network VERSION 0.1 (draft)*. Tech. rep. 2018. URL: https://download.slock.it/whitepaper_incubed_draft.pdf (cit. on p. 64).

[KKK+12] W. Kahlenborn, S. Kabisch, J. Klein, I. Richter, S. Schürmann. *Energiemanagementsysteme in der Praxis ISO 50001: Leitfaden für Unternehmen und Organisationen*. Tech. rep. BMU/UBA, 2012. URL: www.bmu.de (cit. on p. 28).

[KN16] J. Knodel, M. Naab. *Pragmatic Evaluation of Software Architectures*. The Fraunhofer IESE Series on Software and Systems Engineering, 2016, pp. 3–148. ISBN: 978-3-319-34177-4. DOI: 10.1007/978-3-319-34177-4. URL: https://link.springer.com/content/pdf/10.1007%2F978-3-319-34177-4.pdf (cit. on p. 15).

[Kug08] T. Kugelstadt. *The RS-485 Design Guide Application Report The RS-485 Design Guide*. Tech. rep. 2008. URL: www.ti.com (cit. on p. 86).

[Kum13] A. Kumar. *Asymmetric Key Cryptography*. Tech. rep. 2013. URL: http://ssrn.com/abstract=2372882 (cit. on p. 21).

[Lin15] Linux Foundation. *Yocto Project Quick Start*. 2015. URL: https://www.yoctoproject.org/docs/2.0/yocto-project-qs/yocto-project-qs.html (visited on 11/01/2018) (cit. on p. 32).

[LSP82] L. Lamport, R. Shostak, M. Pease. “The Byzantine Generals Problem”. In: *ACM Transactions on Programming Languages and Systems* 4.3 (1982), pp. 382–401. ISSN: 01640925. DOI: 10.1145/357172.357176. arXiv: arXiv:1011.1669v3. URL: http://portal.acm.org/citation.cfm?doid=357172.357176 (cit. on p. 22).

[Mer18] M. Merz. *Enerchain Project Overview and Key Insights*. Tech. rep. 2018, pp. 1–17. URL: https://ponton.de/downloads/enerchain/EnerchainKeyInsights%7B%5C_%7D2018-03-29%7B%5C_%7Dfinal.pdf (cit. on pp. 66, 67).

[Mer88] R. C. Merkle. “A Digital Signature Based on a Conventional Encryption Function”. In: *Crypto* (1988), pp. 369–378. DOI: 10.1007/3-540-48184-2_32. URL: http://link.springer.com/10.1007/3-540-48184-2_32 (cit. on p. 19).

[MH17] D. MacMichael, T. Hudek. *Windows Network Architecture and the OSI Model | Microsoft Docs*. 2017. URL: https://docs.microsoft.com/en-us/windows-hardware/drivers/network/windows-network-architecture-and-the-osi-model (visited on 12/09/2018) (cit. on p. 71).

[Mic13] R. Miceli. “Energy management and smart grids”. In: *Energies* 6.4 (2013), pp. 2262–2290. ISSN: 19961073. DOI: 10.3390/en6042262 (cit. on p. 28).

[MSN+16] J. Mattila, T. Seppälä, C. Naucler, R. Stahl, M. Tikkanen, A. Bådenlid, J. Seppälä. “ETLA Working Papers Industrial Blockchain Platforms: An Exercise in Use Case Development in the Energy Industry”. In: *ETLA Working Papers No 43.* (2016). ISSN: 2323-2439. URL: http://pub.etla.fi/ETLA-Working-Papers-43.pdf (cit. on pp. 15, 37).

[Nak08] S. Nakamoto. “Bitcoin: A Peer-to-Peer Electronic Cash SyNakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System.” In: *Consulted* (2008). eprint: 43543534534v343453. URL: https://bitcoin.org/bitcoin.pdf (cit. on pp. 15, 25).

[Pan17] Panetta Kasey. *Top Trends in the Gartner Hype Cycle for Emerging Technologies, 2017 - Smarter With Gartner*. 2017. URL: https://www.gartner.com/smarterwithgartner/top-trends-in-the-gartner-hype-cycle-for-emerging-technologies-2017/ (visited on 06/27/2018) (cit. on pp. 15, 16).

[Para] Parity Technologies. *Aura - Authority Round*. URL: https://wiki.parity.io/Aura (visited on 12/11/2018) (cit. on p. 88).

[Parb] Parity Technologies. *Parity Bridge · Parity Tech Documentation*. URL: https://wiki.parity.io/Bridge.html (visited on 11/28/2018) (cit. on pp. 91, 92).

[Parc] Parity Technologies. *Private Transactions · Parity Tech Documentation*. URL: https://wiki.parity.io/Private-Transactions (visited on 11/29/2018) (cit. on pp. 54, 93).

[Par18] Parity Technologies. *paritytech/parity-bridge: Bridge between any two ethereum-based networks*. 2018. URL: https://github.com/paritytech/parity-bridge/ (visited on 11/30/2018) (cit. on p. 55).

[PLB14] R. Purdie, C. Larson, P. Blundell. *BitBake User Manual*. 2014. URL: https://www.yoctoproject.org/docs/2.3/bitbake-user-manual/bitbake-user-manual.html (visited on 11/23/2018) (cit. on p. 30).

[Pon18] Ponton. “WRMHL - Industries and Application- independent Blockchain Framework”. In: (2018). URL: https://ponton.de/wrmhl/ (cit. on pp. 65, 66).

[Poo18] J. Poon. *Building a Private Ethereum Consortium - Developer Blog*. 2018. URL: https://www.microsoft.com/developerblog/2018/06/01/creating-private-ethereum-consortium-kubernetes/ (visited on 11/15/2018) (cit. on pp. 72, 83).

[Pop18] S. Popov. "IOTA whitepaper v1.4.3". In: (2018), pp. 1–28. ISSN: 0028-792X. URL: https://assets.ctfassets.net/r1dr6vzfxhev/2t4uxvsIqk0EUau6g2sw0g/45eae33637ca92f85dd9f4a3a218e1ec/iota1_4_3.pdf (cit. on p. 48).

[Pow18] Power Ledger. *Power Ledger Whitepaper*. Tech. rep. 2018. URL: https://powerledger.io/media/Power-Ledger-Whitepaper-v8.pdf (cit. on pp. 59, 60).

[Pro17] Protocl Labs. "Filecoin : A Decentralized Storage Network". In: (2017), pp. 1–36. URL: https://filecoin.io. (cit. on p. 50).

[Pyp18] J. Pyper. *Google Officially Hits Its 100% Renewable Energy Target*. 2018. URL: https://www.greentechmedia.com/articles/read/google-officially-hits-100-renewable-energy-target#gs.70seXF4 (visited on 12/06/2018) (cit. on p. 29).

[Rav16] S. Raval. *Decentralized Applications: Harnessing Bitcoin's Blockchain Technology*. 2016. ISBN: 9781491924549. URL: https://www.worldcat.org/title/decentralized-applications-harnessing-bitcoins-blockchain-technology/oclc/953971624 (cit. on pp. 17, 27).

[Red18] H. Redding. *How Blockchain Can Enable Real-time Green Energy Transactions*. 2018. URL: https://www.rtinsights.com/how-blockchain-can-enable-real-time-green-energy-transactions/ (visited on 12/09/2018) (cit. on p. 40).

[Rif15] S. Rifenbark. "Yocto Project Mega-Manual". In: (2015). URL: https://www.yoctoproject.org/docs/2.3/mega-manual/mega-manual.html (cit. on pp. 30, 31).

[RJ10] G. Ritzer, N. Jurgenson. "Production, Consumption, Prosumption: The nature of capitalism in the age of the digital 'prosumer'". In: *Journal of Consumer Culture* 10.1 (Mar. 2010), pp. 13–36. ISSN: 14695405. DOI: 10.1177/1469540509354673. URL: http://journals.sagepub.com/doi/10.1177/1469540509354673 (cit. on p. 29).

[SAG18] J. Schlund, L. Ammon, R. German. "ETHome: Open-source Blockchain Based Energy Community Controller". In: *Proceedings of the Ninth International Conference on Future Energy Systems* (2018), pp. 319–323. DOI: 10.1145/3208903.3208929. URL: http://doi.acm.org/10.1145/3208903.3208929 (cit. on pp. 15, 33–35).

[Sch08] C. Schindelhauer. "Hashing". In: *Taschenbuch der Algorithmen*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 205–211. DOI: 10.1007/978-3-540-76394-9_20. URL: http://link.springer.com/10.1007/978-3-540-76394-9_20 (cit. on p. 19).

[Sch18] J. Schmid. *Konzept zur Implementierung eines Herkunftsnachweises für Strom via Blockchain*. 2018 (cit. on p. 40).

[SS04] G. Saake, K. U. Sattler. *Algorithmen und Datenstrukturen: eine Einführung mit Java*. dpunkt-Lehrbuch. dpunkt-Verlag, 2004. ISBN: 9783898642552. URL: https://books.google.de/books?id=qRwsPQAACAAJ (cit. on p. 18).

[TFN+16] V. Trón, A. Fischer, D. A. Nagy, Z. Felföldi, N. Johnson. "Swap, Swear and Swindle: Incentive System for Swarm". In: May (2016), pp. 1–30. URL: http://swarm-gateways.net/bzz:/theswarm.eth/ethersphere/orange-papers/1/sw%5E3.pdf (cit. on p. 50).

[TR17] J. Teutsch, C. Reitwießner. “A scalable verification solution for blockchains”. In: (2017), p. 50. URL: https://people.cs.uchicago.edu/~teutsch/papers/truebit.pdf (cit. on p. 49).

[Umw17] Umweltbundesamt. *Das Herkunftsnachweisregister für Strom aus erneuerbaren Energiequellen*. Tech. rep. 2017 (cit. on p. 40).

[VB15] F. Vogelsteller, V. Buterin. *ERC-20 Token Standard*. 2015. URL: https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md (visited on 12/07/2018) (cit. on pp. 24, 25).

[Ver18] Verein Deutscher Ingenieure (VDI). *VDI-Richtlinie VDI 4602*. Berlin, 2018 (cit. on p. 28).

[Wik17] Wikipedia Community Contributors. *ERC20 Token Standard*. 2017. URL: https://theethereum.wiki/w/index.php/ERC20_Token_Standard (visited on 12/07/2018) (cit. on pp. 24, 25).

[Wis08] D. M. Wisy. “Smart Message Language Version 1.03”. In: (2008), pp. 1–42. URL: http://www.emsycon.de/downloads/SML_081112_103.pdf (cit. on p. 86).

[Woo18] G. Wood. “ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER”. In: (2018). URL: https://ethereum.github.io/yellowpaper/paper.pdf (cit. on pp. 22, 24, 33, 47, 49).

[Yah18] M. Yahya. *Decentralized Green Energy Based Certification on the top of a Web3 Based Energy Geo Browser*. 2018 (cit. on pp. 40, 94, 95).

[YMRS18] D. Yaga, P. Mell, N. Roby, K. Scarfone. *Draft Blockchain Technology Overview (NISTIR-8202)*. Tech. rep. 2018, p. 59 (cit. on pp. 15, 19–23, 25).

[ZR10] N. Zivic, C. Ruland. “Security Architecture of Smart Metering Systems”. In: 2010, pp. 249–259. ISBN: 978-3-642-16282-4 978-3-642-16283-1. DOI: 10.1007/978-3-642-16283-1_28. URL: http://link.springer.com/10.1007/978-3-642-16283-1_28 (cit. on p. 69).

[ZXD+17] Z. Zheng, S. Xie, H. Dai, X. Chen, H. Wang. “An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends”. In: *Proceedings - 2017 IEEE 6th International Congress on Big Data, BigData Congress 2017*. 2017, pp. 557–564. ISBN: 9781538619964. DOI: 10.1109/BigDataCongress.2017.85 (cit. on pp. 19, 24).

Cooperation

The thesis was conducted in cooperation with the OLI Systems GmbH. OLI Systems develops and implements blockchain applications in the energy sector. Hereby, several research and laboratory projects have been initiated. This thesis is part of one of those projects. During the implementation of the minimal prototype OLI Systems provided the therefore necessary resources. The work at Oli Systems has been supported by Dr. Thomas Brenner.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature