University of Twente

Implementing and Scaling HanzeNet

Masters' Thesis

Daniël van de Giessen

2018-10-17

This page intentionally left blank.

ABSTRACT

Means of producing your own energy using technology such as solar panels is becoming available to more people every day. HanzeNet is a Dutch start-up working on a platform for trading such locally generated energy within local communities.

This thesis details the design of a software system enabling such trading of energy in a decentralized manner using blockchain technology. Our design allows consumers to directly trade energy with others in their community. This makes it feasible for many communities to run solely on locally produced energy throughout most of the day, only buying power from external energy suppliers when the local supplies are depleted. Multiple means of energy production can be tracked to allow customers to choose which type of energy they would like to consume.

Our system is designed to scale from small neighbourhoods to entire cities using a layered design to ensure optimal performance and manageability, and is shown to perform well even on lightweight server platforms. The system is released as free and open source software, allowing anyone to verify the implementation and check whether all operation are performed correctly. An extensible design allows for future additions of additional functionality without having to fork the existing blockchain network.

ACKNOWLEDGEMENTS

This thesis describes the work conducted as part of my graduation project for the master Computer Science at the University of Twente. While working on this project I've had the opportunity of learn a lot about new and emerging technologies and to expand my experience with software development, and I'm happy to see that the results are of value to the HanzeNet team which are taking this work into production.

I would like to thank dr. ir. Maurice van Keulen for his supervision and continuous feedback during this time, and for helping me stay focussed on the important academic aspects of this project.

I would like to thank Erwin Bisschop, Steven Verkuil, and Wouter den Bakker, for the many illuminating discussions about the design and implementation of the system. Together with Steven I was able to flesh out many of the important edge cases that needed to be considered. Without help from Wouter I would never have been able to find, understand and resolve some of the problems encountered while implementing the system.

I would like to thank Marco Scholten, Marius Groen, and Willem Poterman, for the valuable conversations about my design choices, and for their feedback while writing this document.

I would like to thank my friends and family for their many words of encouragement and support during this my studies and in these final months working on my thesis.

Above all I want to thank and give praise to God, my Heavenly Father, the Almighty, and to His only Son Jesus Christ, my Lord and Saviour, for providing for me every day of my life.

Enschede, October 2018.

Daniël van de Giessen

This page intentionally left blank.

CONTENTS

1 INTRODUCTION

1.1 CONTEXT

HanzeNet is a Dutch start-up in the electrical energy industry working on decentralizing energy trading. They do this by introducing small scale marketplaces where local communities can trade their energy between themselves instead of all trading with a remote energy supplier. Trading with other local marketplaces and/or remote energy suppliers is used to balance any surplus or shortage of locally available energy.

This move is enabled by a number of civilizational developments: The move to using electricity for many appliances which previously ran on other sources of energy (for example electric cars, electric heating and cooking) means electricity is being used more than ever. Technologies like affordable solar panels, battery banks and more efficient appliances result in increased decentralization: Consumers now produce, store and use their own energy. And new developments in software and digital communication now make it feasible to create smarter systems for monitoring, planning and accounting energy usage, enabling solutions for automated trading.

So far, the HanzeNet start-up has developed a small-scale working proof-of-concept in cooperation with a distribution network operator based in the Netherlands. where they run a simulation of how energy could be traded between a small number of households which both produce electricity by means of solar panels and consume it, showcasing how many of the participating households can run solely on energy produced within their community for most of the day, not needing an external energy supplier. This improves energy efficiency, helps distribute load on the central electricity grid and incentives local initiatives for cleaner energy.

With the initial proof-of-concept showing good results, the next goal is to move from a simulation run externally at HanzeNet to a working system on-premise, and scale this system to make it work for larger environments. Here lie a few open research questions for determining how such a system may best be designed.

1.2 PROBLEM STATEMENT

In this report, we describe the design of a software system that allows energy trading and management services for cases such as but not limited to HanzeNet.

Given this context we formulate our problem statement:

How can we build decentralized, scalable, digital marketplaces for electrical energy?

We aim to address both a number of high-level architecture concerns as well as lower-level technical concerns to form a broad overview for how such a system may be designed and build.

We will scope our design specifically for cases in the context of the electrical energy systems in The Netherlands, which means we will document some of our decisions to be based in Dutch-specific law and other environmental circumstances. HanzeNet will be the example case we use throughout this report.

1.3 RESEARCH QUESTIONS

To help guide the research, we formulate a number of sub-questions which we'll attempt to answer in order to answer the main question. Each of these is supplemented with a short rationale and first thoughts.

I. **How does the HanzeNet proof-of-concept work?**
The current proof-of-concept is the base upon which we'll continue our research, both to make sure the goals are attainable within the available time as well as to build upon existing knowledge available within the HanzeNet team. The proof-of-concept may also provide a basis which may be used as a testbed or for validating changes through simulation.

II. **How can we account energy usage?**
HanzeNet allows users to "trade energy". However, the system is purely acting in the administrative domain and not in control of the physical energy being transported. Thus, we need a trustworthy source of information about produced and consumed energy; preventing malicious users from trading with energy in our administrative system that doesn't actually exist in the physical world. This would likely involve making use of existing legal boundaries which prohibit tampering with electricity meters.

III. **How can we connect households for trading within the local marketplace?**
We need a way to connect households within the same local marketplace in order to make them trade energy between them. Internet-based solutions may be considered as well as simpler local radio-based connectivity. Considerations have to be made regarding what happens when there is no connectivity but the need

to trade energy is. We show through simulations and prototypes concretely how these considerations function.

IV. **How to preserve privacy of individuals within the local marketplace?**
When trading energy within a local marketplace, neighbouring households can learn a lot about each other's energy usage over time, which may cause privacy concerns. We show that using modern cryptography it is possible to reduce the amount of information one can learn about individual households. This yields some limitations as to what information is available to the system to work with however. We provide an overview of what the options for this and their respective impact to the system are.

V. **How will local marketplaces trade with external entities to resolve shortages and/or remainders?**

After local marketplaces have been trading internally to resolve all energy needs, it might happen that there isn't enough energy available within the marketplace for the current demand, or there might be a surplus. Thus, the marketplace needs to trade with external parties, such as other nearby marketplaces or centralized energy suppliers to resolve this. We consider whether individual households or the marketplace as a whole should trade, and how well these approaches fit in our system.

1.4 APPROACH

The process for this research follows the typical outline of a system design project.

We started off with an exploration of the design context where we provide a short study of relevant literature and technologies, and familiarized ourselves with the workings of energy delivery in The Netherlands. We also examined the previous prototypes and the outcome of the experiments conducted by HanzeNet to learn how these functioned.

Based on the acquired information we started the exploration and design process. We explored various design elements and discussed these with both the HanzeNet team and external experts to verify our assumptions and better understand what effects specific decisions may have. During this process we also developed a number of small prototypes to help us better evaluate the impact of decisions on software design and/or resulting functionality.

Extra attention was devoted to examine edge cases and unintended behaviour arising from the design decisions, to determine whether particular combinations of functionality could give rise to unwanted side effects in the system.

Stemming from these design decisions a system design was made and a software implementation was written. The system design was subsequently validated using a set of predetermined qualitative measures regarding functionality, security and extensibility. Using the implementation, we also assessed the systems' performance and scalability in a quantitative manner by simulating large numbers of operations in the system and measuring how it performed.

1.5 STRUCTURE

This documents' structure reflects the research approach described in the previous section. Chapter 2 contains background information consisting of related work and descriptions of technology used within this research. Chapter 3 goes in detail on the research approach and details the requirements, general solution architecture and evaluation process. Chapter 4 presents the base architecture of the system, addressing the hardware and environment questions as well as the base layer for the software implementation. Chapter 5 focusses on the system design from a conceptual and software perspective and presents the various important design decisions made within the system. Chapter 6 presents the designed implementation of the system in detail. Chapter 7 contains the evaluation of features, security and privacy properties, and performance and scalability properties of the developed system. Chapter 8 concludes this research and discusses future work.

2 BACKGROUND

2.1 RELATED WORK

2.1.1 TRIANA

TRIANA[1] is a control strategy for smart grids developed at the University of Twente as part of the PhD thesis of Vincent Bakker. It defines a three-phase strategy for identifying flexibility on the demand side of the energy grid and leveraging that information to let a centralized planner component optimize for given objectives such as reducing stress on the supply chain and maximizing economic value.

TRIANA's focus is primarily on managing flexibility and it does not handle the administrative properties we aim to handle with our system. It might however be an interesting option for future integration with HanzeNet to provide planning data for flexibility trading, something we recommend as future work.

2.1.2 PowerMatcher

PowerMatcher[2] is a distributed systems architecture and communication protocol for smart grid systems under active development at TNO in collaboration with members of the Flexiblepower Alliance Network[3]. It provides a standardized system for flexible end devices to register their interest in consuming energy at a future time, based on user-provided parameters such as a desire to consume green energy, optimize for the lowest price or to consume their energy (and subsequently perform their function) within a limited timeframe. A centralized "auctioneer" subsequently determines an equilibrium price based on the registered supply and demand, which in turn triggers the actual use of energy by devices.

PowerMatcher focusses primarily on trading flexibility and decreasing peak loads on the power grid by controlling end devices. It however doesn't provide functionality which would allow it to be used with entire households for administratively trading energy which has already been consumed or produced. It might however be an interesting option for future integration with HanzeNet as a means of flexibility trading with PowerMatcher compatible devices, something we recommend as future work.

2.1.3 USEF Framework

The USEF Framework[4] is a standard for interoperability between smart energy systems by specifying a set of rules and standards. It provides detailed business process definitions and a technical reference implementation of the various roles and interactions defined within the framework. The framework is maintained by the USEF Foundation.

While a standardized framework could be of value for interacting with other parties in a standardized matter, in its current form the USEF Framework is focussing on tackling different problems from the one we're addressing in this research, namely the business process aspects of interacting entities in the smart energy ecosystem. While it might be interesting to evaluate the USEF Framework at a later stage to allow interaction between HanzeNet and others implementing these same standard, it currently does not sufficiently address our needs for administratively trading energy based on smart meter readings for consumer households.

2.1.4 DeKo and SolarCoin

DeKo[5] is a proposed currency backed by a portfolio of electricity delivery assets in the form of physical power production assets such as fuel or power plants, as well as standardized power purchase agreements accepted as a proxy of power production value. The paper argues this diversified approach provides for a more stable retention of value compared to traditional currencies, with the additional benefit that investments may benefit development of cleaner energy solutions.

SolarCoin[6] is a digital cryptocurrency based on blockchain technology which employs the various ideas presented by the DeKo paper. It assigns a value in coins to verifiable solar energy production which can subsequently be used to trade like most cryptocurrencies. It relies on Know-Your-Customer (KYC) processes to verify these solar energy claims. The technical maintenance and claim verification processes are performed by the SolarCoin Foundation.

The concepts of DeKo and their implementation in SolarCoin are useful on a global scale and focusses on solar power, but for smaller scale trading of energy including different sources of power it is less suitable. It might however be an interesting option for future integration with HanzeNet to allow participant to trade their green energy with external parties in the SolarCoin community.

2.1.5 Energy Web Blockchain

The Energy Web Blockchain[7] is an open source blockchain platform based on Ethereum designed specifically with applications in the energy market in mind, while a number of closed-source applications is developed to support trading and regulatory operations. A permissioned blockchain test network currently exists but trading of energy is not yet possible. The blockchain and supporting applications are developed by the Energy Web Foundation.

The Energy Web Blockchain focusses on providing a generic platform with consideration for regulatory requirements for permissioned use. It doesn't provide any trading capabilities of itself. If expanded with such functionality in the future it may be an interesting option for integration with HanzeNet to allow traders from both platform access to a larger community of energy producers and consumers.

2.1.6 PowerPeers

PowerPeers[8] is a Dutch energy provider which allows its customers to administratively share/trade their produced energy with other customers, as well as providing real-time insights in energy use and which customer administratively supplied the energy being used. The functionality available to end-users is similar to what we aim to provide through HanzeNet, however the implementation and platform are closed-source and not available for external parties to work with, and additionally relies on customers switching to PowerPeers as their energy supplier.

2.1.7 Blockchain Technology in the Energy Ecosystem

In her masters' thesis[9], Amber Voets demonstrated the economic and strategic viability of a blockchain based trading systems in the Dutch energy ecosystem. This serves as a confirmation of the business viability as assessed by the HanzeNet team for further development of a decentralized energy trading system. As an explorative study its work focusses mainly on the impact on economic and business ecosystems, but does not provide a more detailed look at the system implementation questions we aim to address. The general insights into the functioning of the current energy ecosystem in The Netherlands proved valuable during the initial exploration of our problem domain.

2.2 Technology stack

2.2.1 Blockchain

Blockchain is a technology which at its core allows forming an append-only list of data in a distributed fashion. This makes it an interesting technology to examine for our use case, because an immutable history means we can verify all past trade transactions and being distributed aligns well with our requirements.

The first distributed blockchain was described in the paper detailing the design of the Bitcoin[10], a digital "cryptocurrency" using the blockchain as a core technology providing a distributed ledger of transactions. Since then many other blockchain-based applications have sprung up.

Blockchain is not as much of a radical new technology as often presented in media and marketing. It builds upon well-known techniques in distributed networking and cryptography. What sets it apart is its smart use of a consensus algorithm combined with a chain-like append-only data structure. As such, we believe blockchain can often be described as a paradigm in distributed ledger system design as much as a technology in itself: It is a design pattern that allows easy creation of a distributed system, similar for example to how working with big data was made accessible for many by the introduction of the map-reduce paradigm.

In a blockchain, all data is inserted using “transactions”, which are contained in a so-called block. A block contains a limited amount of transactions in a certain time span. Blocks are linked together by referring to the previous block to form a chain of blocks, the block chain. In most blockchains, all this happens in distributed fashion, meaning that at any time there may be multiple chains starting from the same block with different transactions appended. In such scenario’s, to ensure only a single version of the data is maintained, the longest available chain of blocks is considered by all parties in the network to be the canonical and valid list of transactions; ignoring shorter chains.

Creating a new block is often referred to as “mining”, and the party creating the block as a “miner”. Not all nodes in a network necessarily need to be miners.

To append a block to the blockchain, a miner finds the longest chain and mines a new block which is linked to latest block in that chain. The new block is shared with the rest of the network. The resulting chain is one block longer than the previously known longest chain, so other nodes in the network will begin accepting it as the new longest chain.

If all this happens concurrently in a distributed network, there is the possibility that multiple new blocks are mined at the same time, and thus multiple contenders for “the longest chain” arise. The blockchain is “forked”, with different miners working on different chains. However, as time progresses and more new blocks are mined, one of these chains will be the first to grow even longer than the other chains. When that happens, it becomes the uncontested longest chain. The other chains are discarded by all other nodes. Since data in blocks in those other chains did not make it into the new longest chain, nodes will retry to append their data to the next block based on the new chain.

Linking a block to a previous block is done using the output of a cryptographic hash function. This ensures that the previous block cannot be modified without detection, as it will have a different hash value than the one stored in the current block. As a recursive result, the integrity of the entire chain of previous blocks is protected by the hash contained in the current block.

When mining a block, the miner checks than the data contained in it is valid. It then calculates a proof of validity. This proof is most commonly a set of cryptographic signatures from trusted parties (called proof-of-stake), or a solution to a computation-intensive problem (called proof-of-work), although other techniques exists. These signatures or solutions can be checked by all nodes in the network to ensure they are correct, preventing anyone from inserting invalid blocks into the blockchain. For distributed blockchains it also means that the creation of new blocks happens in a controlled manner, which reduced the number of blocks that are mined at the same time and resulting forks of the blockchain.

These properties also make it hard for anyone to change data in previous blocks, since the only way to do so is by trying to create a longer chain than the current chain. This would mean mining a lot of blocks, which would in most cases require the trusted parties to sign these, or to be able to solve these computation-heavy problems faster than the rest of the network combined in a so-called 51% attack, both of which are generally hard to pull off without a majority of the network actively or passively colluding.

Data stored in a blockchain is thus verified by consensus to be correct and stored on every node participating in the network. As the blockchain grows, data in historic blocks effectively becomes immutable due to the increasing cost of modifying it. It thus is a append-only, distributed data structure.

Most of the first generation public blockchains make use of the mentioned computation-intensive problem solving in the mining process, which eliminates the use of trusted parties. This methodology has drawbacks, such as the huge waste of computational energy required to keep the network alive and the associated penalties to performance, scalability and costs. As such, for non-public or “permissioned” blockchains the trusted party setup is often preferred.

2.2.2 Cryptocurrency

Cryptocurrency is the application of blockchain technology to keep a distributed ledger of financial transactions.

Typically, public keys used to sign transactions are referred to as addresses or wallets and are associated with a balance often referred to as “coins”. These are not actual coins, but rather the sum of all transactions associated that wallet. Transactions consists of an input, which is a list of (not previously spent) transactions, and an output, which is a list of wallet addresses and a ratio specifying how the funds should be distributed over these wallets.

In most cryptocurrencies, to incentivize miners to mine blocks, the miner may specify a wallet address which receives a predetermined amount of coins as a reward for mining the block. This amount is usually determined by a curve which ensures new coins are gradually introduced into the market over time, also referred to as the difficulty curve, and is adjusted over time to optimize the speed and efficiency of the cryptocurrency[12].

2.2.3 Smart Contracts

Smart contracts are contracts which are specified using a digital protocol instead of using natural language, allowing them to be verified and executed programmatically. Although conceptualized in 1996[13], the term nowadays most often refers to the implementation in cryptocurrencies where arbitrary (often Turing-complete) computations can be run as part of the smart contract computer code. This allows for complex contracts to be governed purely by code, providing many advantages such as automated validation and execution of actions defined in the smart contract. In cryptocurrencies, smart contracts are usually implemented by committing the smart contract code and subsequent transitions of its state machine on a blockchain. Ethereum[14][15] is currently the best-known blockchain providing such functionality.

2.2.4 Zero-Knowledge Proofs

A zero-knowledge proof[16] is a method to allow proving correctness of a statement without any other information about the statement being revealed. A "prover" can provide a statement while keeping its details secret, while a "verifier" can still verify that the statement is indeed correct. Zero-knowledge proofs often are interactive, requiring the prover and verifier to interact multiple times.

A zero-knowledge proof satisfies three properties: Completeness: If the statement is true, the prover is (eventually) able to convince the verifier of this fact. Soundness: If the statement is not true, the prover will not be able to convince the verifier of this fact. Zero-knowledgeness: Apart from the fact that the statement is true, the verifier learns nothing about the statement from the prover.

The properties of zero-knowledge proofs make them a useful tool in the context of digital identity systems, helping tackle various problems relating to security, confidentiality and privacy.

A notable application of zero-knowledge proofs is in the digital cash systems such as Zerocash[17], a cryptocurrency which uses a specific form of zero-knowledge proofs, zk-SNARKs, to provide completely anonymous and private transactions, and Monero, which uses ring signatures and transactions[18] for the same purpose by hiding information about the source and content of all transactions.

2.3 The HanzeNet prototype

The first prototype for HanzeNet had as goal to demonstrate the viability of the HanzeNet trading concept on a small scale. It also serves as a visual explanation of how the HanzeNet system is intended to be used.

In this first prototype, energy usage data was acquired via the energy supplier from a set of households which had agreed to be part of the experiment. These households were equipped with solar panels as a means of producing energy locally. The energy data was collected via the regular channel of the energy supplier, meaning it was delivered to HanzeNet with a few days delay. The data

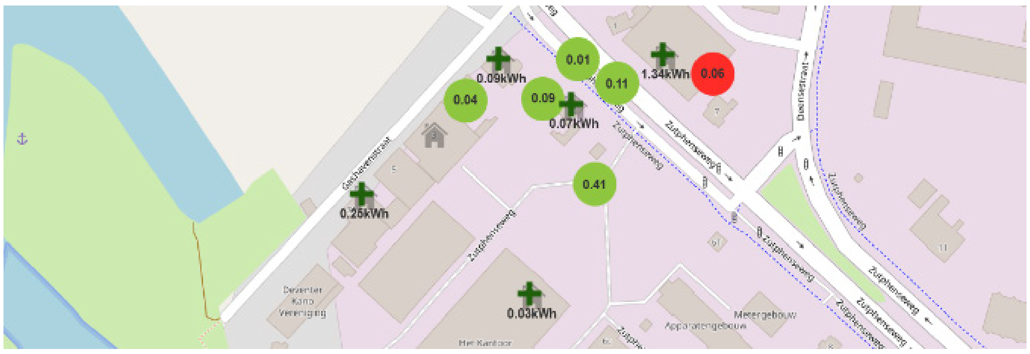


Figure 1: The HanzeNet prototype visualization

consisted of energy counter deltas per 15 minutes, indicating whether a household had consumed or produced energy in that period of time.

Upon receiving it, the data was entered in a prototype blockchain application built upon a heavily modified version of the Dragonchain platform[19], which supported a basic set of smart contracts written in Python. It consisted of an access control smart contract, and a generic transaction smart contract which could transfer energy from one address to another, or by specifying a hardcoded special case source address could introduce new energy to simulate production or consumption.

After the production and consumption was entered into the prototype, all energy was then redistributed between all households using transactions using a minimization function trying to eliminate all negative balances. When no sufficient energy was available within the network, extra energy would be added to signify energy being bought from an energy supplier.

The entire prototype functioned as a simulation showcasing how energy could theoretically be traded between parties. Its output was visualized to demonstrate how energy trade occurred between these households[20]. In Figure 1 we see a snapshot of the visualization animation. It shows six households on a map, represented by the icons of a house. Five of these households are producing energy, which is indicated by the green plus sign. The opaque circles are animated to move from one household to another and represent the energy flow between the households. Green circles indicate locally produced energy, while the red circle indicates energy acquired from an external supplier. Using all these elements, the visualization shows how locally produced energy is used over time.

The experiment demonstrated the general viability of a system for trading energy by showing how locally produced energy could efficiently be consumed by neighbouring households. It also generated interest by external parties such as energy suppliers and housing corporations to explore the implementation of a trading system for their customers, indicating the economic viability of further developing this concept. The prototype itself is not well suited for this, as it doesn’t consider any of the desired security, privacy and scaling properties as well as being limited in implemented functionality. The system we design as part of this project is aimed to address these questions and provide a solid basis for further development.

3 APPROACH

3.1 DESIGN REQUIREMENTS

Through various brainstorm and refinement meetings with the HanzeNet team we identified a number of requirements which we consider of importance for designing the system. Some are based on business requirements, creating opportunity for the system to be deployed in varying environments, some are based on limitations imposed HanzeNet by Dutch law or physical limitations in energy systems, and some are based on the desire to put the user first and allowing them to choose what they do and what they do not want to do.

3.1.1 Base functionality

The system should fulfil the basic functions for trading energy. It should be able to be supplied with an input of energy usage data, and this data should be bound to time data to allow the system to account for changing energy prices over time. Users should be able to trade among each other. Preferably, we want users to be able to choose what type or source of energy they want to use (for example from the solar panels of their neighbour or from the coal-fired power plant a few miles away). Relevant administrative management functions should be available for the party managing the system.

3.1.2 Security

Since the system will be used for tracking and trading in energy, there is monetary value attached to the systems' actions, and thus, there is the possibility some malicious users will attempt to manipulate the system for personal gain. The system should thus be secure against fraud by both end users and trusted parties, by either preventing such actions in the first place or providing the ability to detect and act upon such activity. The system should be reliable and safeguard against improper operations. Historical data should be immutable and any new changes must always be authenticated to ensure proper access control.

3.1.3 Privacy

The system will work with data which may be considered personal to some degree, which means we will have to take adequate measures to make sure this data is properly protected. At the same time, since we're building a marketplace some information inherently needs to be shared. There should be options available to deal with multiple degrees of information sharing, both inside and outside the system (by protecting information in the system itself, or limit how information in the system may be linked to individuals outside it).

3.1.4 Scalability

The system will be deployed for a large number of households. To maintain the benefits the system can provide, it has to scale so its performance does not suffer when being used by such large numbers of users. We will evaluate this by benchmarking the amount of operations the application can process in a limited timeframe, as well as showing how these results can reliably be used to reason about performance of larger deployments.

3.1.5 Extensibility

The goal of HanzeNet is to develop an open platform both we and third parties can energy-related application upon. The system should thus allow for easy expansion of core functionality, and allow external systems to connect to it to interchange data between them. Design decisions made in the first iteration of the system should not be unnecessarily limiting functionality so future use cases may extend upon it.

3.2 Design approach

To address the requirements, we aimed to designed a software solution that enables measuring energy use and using that information to let user's trade in energy. We will first explore and design a base architecture in the form of a distributed system based on blockchain technology. This platform will serve as a base which addresses common required functionalities such as acquisition of energy usage data, connectivity, immutable data storage and exchange, and secure user authentication.

Upon this base architecture we will design the core of our system which is concerned with all the domain-specific design decisions for enabling energy trade. We may design and prototype a number of different options for resolving specific design problems while iterating over our overarching system design. This can for example include different approaches for handling trade, various ways to split up functionalities in smaller, access-controlled operations and so forth.

After the iterative design process is complete we will develop an implementation of the core system design upon the base architecture in the form of a set of smart contracts on the blockchain. This implementation will be used to demonstrate the design as well be used for the validation of the system. Additional tooling we require during the development or testing process will be developed in this phase as well.

While the end-users of the system will include consumers, how these users practically interface with the system is not within the scope of this research and we will thus not be conducting significant interface and interaction design.

3.3 EVALUATION APPROACH

The result of this work will be evaluated using a two different of measures, assessing both qualitative and quantitative concerns.

The qualitative requirements defined in section 3.1 will be validated for the designed system. We will be providing a short analysis of how specific components of the system address the requirement or summarize how the requirement is met as a result of the overarching system design.

A verification of the performance and scalability properties is supported by a quantitative evaluation of the implementation, by running a benchmark of the system in which a large number of operations will be simulated, which will involve every component of the designed system. By combining these performance measurements with a design that ensures these numbers scale well to larger deployments we argue that the system is well-performing and scalable.

4 Base architecture

This chapter discusses the base architecture of the system, including the considerations made on the hardware and environmental aspects of the design.

4.1 Hardware & accounting

4.1.1 Measuring energy usage in The Netherlands

The first stage in the HanzeNet system is acquisition of energy production/consumption data. This data will subsequently be used for trading, so it is important we are able to acquire it accurately and with little delay. For this purpose, we aim to integrate the functionality we require into pre-existing smart energy meters. The advantages of this approach are multiple: Energy meters are already the legally required method in The Netherlands for measuring energy[21], and additionally are protected by that same law against tampering with the meter. That means that instead of having to develop various security countermeasures specifically to protect our entry point of data, we don't have to introduce such complexity and rely on existing legal frameworks for protecting any operations within the boundary of the smart meter. Integrating with smart meters also means we will have immediate access to reliable energy data without significant measurement errors or delays that would occur when using external or older types of energy meters.

Smart meters are just one class of meters and end users are not required by law to specifically use a smart meter. However, this will be a requirement for HanzeNet, as without the data from a smart meter it will not be possible to participate in trading of energy.

Smart meters in The Netherlands already have the ability to transmit usage data. Measurements from the smart meter are transmitted via the grid operator to EDSN, Energy Data Services Netherlands[22], which in turn shares that data with energy suppliers and other certified companies. This functionality is actually used by all active energy suppliers to calculate energy use and bill the end user accordingly. The main disadvantage is its huge delay: It takes days for the data to be delivered this way, and in practise due to bureaucracy delays of several weeks have been observed.

4.1.2 Acquiring reliable measurements

All smart meters are equipped with a P1-port, a RJ-11 connector which provides live energy measurements directly from the meter about every 10 seconds. This kind of data is ideal for use with HanzeNet: It comes directly from the (trusted) energy meter, so our numbers will match with the numbers used by the grid operator / energy supplier, something that wouldn't be guaranteed if we were to

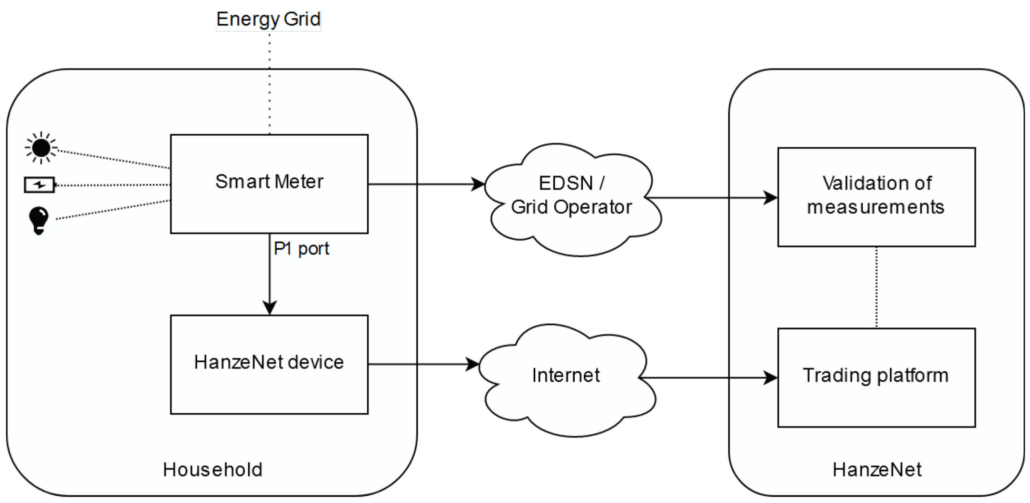


Figure 2: Overview of setup for acquiring energy measurements

use a separate energy meter. The disadvantage of using the P1 port is that it is an externally connected device, and thus not protected by the legal boundaries which apply to the smart meter itself. As such, it would be easier to supply our device with false information or just disconnect it occasionally when the user thinks that would yield a benefit to him/her.

Integration within a smart meter is thus our end goal, however, developing a customized smart meter including getting it certified and producing it on scale is out of scope for this research. HanzeNet will for the time being focus on building the blockchain platform. We will use a dualistic approach for collecting energy measurements: We will use the P1-port for real-time data, and also request that same data via the grid operator. When we receive the data from the grid operator, we will compare this to the data we acquired via the P1-port, and if there turns out to be any difference between the energy the user traded based on P1-data and the actual energy reported via the grid operator, we will simply send a bill to the end user, using the energy prices from the external supplier. This ensures that no matter how prices change, it will always be cheaper for the end user to keep its HanzeNet device connected and operational because we assume that either the external energy supplier is more expensive than local energy production, or if somehow energy from the external energy supplier is cheaper than locally produced energy, our system would buy its energy there regardless, eliminating the incentive to disconnect the device in any case.

During this first stage, our device connected to the P1-port would not be protected by the legal protection of a smart meter. Using the same approach to handle disconnecting the device we can counteract tampered data inserted by a malicious user, and additionally we will implement a few limits on input data to detect and prevent unreasonable data from entering the system.

Unintentional disconnects are handled in the same way, as distinguishing between these is not feasible during our first stage. This does mean that the possibility exists that users will pay the slightly higher prices from the external energy supplier in cases of a technical problem. We decided that that is an acceptable cost during this first stage, as given the stability of smart meters it is unlikely the connection will fail unpredictably, and because we detect disconnections we can always decide to refund the user in case we are able to confirm it was caused by a technical issue outside the influence of the user. In the second stage, when integrating directly into the smart meter, the problem of disconnection will by definition no longer exist.

4.1.3 Connectivity platform

For connectivity we chose to primarily focus on standard Internet connections via standard Ethernet or Wi-Fi for the first versions. These are standard and very widely deployed technologies and we can reasonably assume their availability. Other connectivity options were considered, such as LoRa mesh networks or LoRaWAN[23]. To limit complexity and keep initial costs and risks of implementation low we opted to not further develop these options. However, these are promising options for future development, especially when deploying HanzeNet in local communities with relatively small distances between households. In our system design we ensured implementation of such alternative connectivity technologies is possible without major changes to other parts of the system.

4.2 BLOCKCHAIN-BASED SOFTWARE PLATFORM

As a basis for the HanzeNet system we opted to go with a blockchain-based architecture. Coinversable had already been working on a new, open source blockchain platform called “Validana”. Since Coinversable is also one of the partners in HanzeNet, we choose to adapt this new blockchain platform for the HanzeNet system.

Validana is written in TypeScript, a high-level language that compiles to JavaScript. The core platform, including all improvements that were contributed during this research, has been open sourced under the AGPL-3 license[24].

The Validana platform acts like a permissioned blockchain with a single miner, called the “processor”[25], and multiple nodes mirroring the blockchain. It has full support for complex smart contracts. The single processor acts as a central trusted party, the only one that can accept transactions and thus decide what data gets on the blockchain. While it can choose to accept or reject transactions, it cannot modify them (as this would violate the signatures from the submitter of the transaction) nor make any unnoticed changes (since all data is still mirrored across all nodes and all nodes can still check that the processed blocks are correct).

The primary advantages of the single processor approach in Validana for our system are the performance properties and the ability of update smart contracts running on the blockchain. Validana doesn't use a computation-intensive mining process but only requires a valid signature by the processor for a block to be validated. The performance of the blockchain can thus be scaled by scaling up the processor, something we will look into further during our system validation. Additionally, since there is only a single processor controlled by a trusted party, this trusted party can control which smart contracts it accepts and which ones it does not. By choosing the trusted party appropriately, this allows us to deprecate smart contracts as we want to replace them with newer versions, for example to allow patching issues, adding functionality or changing the underlaying logic entirely without having to fork the blockchain and update all nodes accordingly.

Validana uses public key cryptography to secure all its operations, based on the Bitcoin protocol and thus inheriting its security properties. All identities on the blockchain are represented by a private/public keypair. The public key acts as a public address to reference to the identity, and the corresponding private key is used to sign invocations of smart contracts.

All operations on the Validana blockchain are governed by smart contracts: There is no functionality besides calling a smart contract with a set of parameters. Smart contracts can access and modify a global, shared state. Smart contracts thus are trusted code, and as such only the processor, the single trusted party, can add new smart contracts. All existing contracts can be called by any other parties, and thus smart contracts should check whether the invoker of the smart contracts is allowed to perform the operation it is attempting.

In the scenario of HanzeNet we already have a trusted party: The grid operator. The grid operator is required by law to operate the electricity grid and perform duties such as determining energy use. Since its role is embedded in Dutch law, they are expected to continue existing even as business models change, and thus are a good candidate to fulfil the role of trusted party in our blockchain by operating the processor. That way, they can set the rules by which the system operates, and their legal responsibility ensures that they should not misuse that power lest they be punished by law.

We chose to build upon this platform for the HanzeNet system since its properties make it an excellent fit for what we aim to do. The single processor approach ensures we'll have good performance, and is not a problem since we will have a trusted party in our system either way, and all nodes mirroring the data will be able to check the work done by this trusted party. The contract-based system makes it easy to develop and prototype new functionality, and the open source nature and TypeScript language makes it easily accessible to third-party developers wishing to develop upon the HanzeNet platform. The ability to deprecate contracts is very useful to have if we ever want to transition to newer versions of those contracts; given that direct trading of energy is still young and may change as the markets, technologies and government regulations mature it is important that we can adapt to those changes. There exist no other existing blockchain platforms out there which offer us the same set of functionalities.

With the Validana blockchain as the base for the HanzeNet system, most functionality will be implemented using smart contracts. The trusted party, the grid operator, can manage which smart contracts are accepted in the blockchain. The HanzeNet start-up aims to build closed-source applications exclusive to paying customers to sustain its business model, while allowing anyone to build their own applications upon the open source blockchain platform and smart contracts.

5 SYSTEM DESIGN

This chapter will detail a number of the design decisions which were in development of the system, as well as discuss rationale and alternative options. At the end of the chapter, we give a short overview of the design choices and rejected alternatives in Table 1.

5.1 SEPARATED CONTEXTS

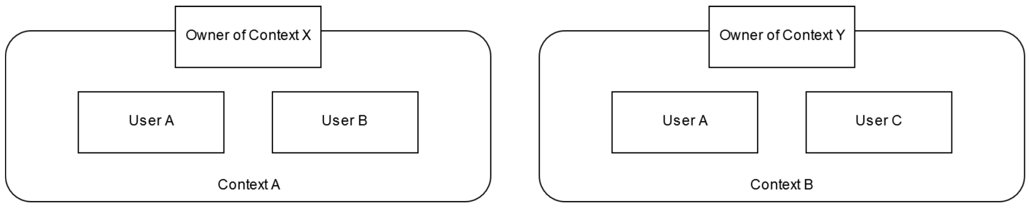
Users can only participate in a HanzeNet marketplace with permission. For this purpose, all smart contracts in HanzeNet operate within a context, which is a separate identity on the blockchain. The public address is used as an identifier for the context, and the private key is held by the context owner, allowing him/her to perform administrative duties within his/her context. All smart contracts take a "context" parameter, and before executing any functionality will check if the identity calling the smart contract is within a list of allowed addresses. The context owner can add and remove addresses to these lists using specialized smart contracts.

A context thus functions as a closed system in which all operations take place. No information is shared between contexts; actions in one context will never affect any other context. Any valid identity can represent a context, and an identity can be part of any number of contexts.

See Figure 3. Here we see a visual representation of two contexts, X and Y. The context owner is represented by the top block in each context. We see a total of three users: A, B and C. User A is added to both contexts, meaning it can perform operations in either one. But the only thing user A in context X and user A in context Y have in common is that they share a blockchain identity: They have the same address, and sign their transaction with the same private key. For all other purposes, they can be regarded as completely separate users.

This separation of contexts allows us to have multiple marketplaces run upon the same blockchain platform without having to maintain completely separate blockchains for them. This ensures we can minimize the costs for running the system for multiple marketplaces and conveniently have data for multiple

Figure 3: Two separate contexts with three blockchain identities as users



contexts available in one place when cross-context operations become relevant in future versions of the system.

5.2 KEEPING BALANCE

Once a user is allowed to operate within a context, in its most general form, HanzeNet allows him/her to do two things: Account energy consumption/production and trade. For that purpose, HanzeNet keeps a balance for each user in the context. The balance is mutated whenever one of those two basic functions is performed.

We can model this similar to a bank account: We sum the all amounts a user has produced, consumed, received through trading, or transferred away, resulting in a single amount. This amount is a users' balance.

When our energy meter shows a flow of energy, it is recorded and the "energy" smart contract is called. When the yield is a positive amount (energy was produced) the balance of the user will be increased, while a negative amount (energy was consumed) will decrease the users' balance.

These amounts (or derivatives thereof) can then be used for trading: Users can transfer amounts to other users by calling a "transfer" smart contract.

Consequently, if a balance is negative, a user has "used" more energy than they provided, and thus need to pay for the remaining energy. Likewise, if their balance is positive, they have provided more energy than they've used, and need not pay for anything as of that moment. This principle of balance is used in all HanzeNet smart contracts.

An additional limitation we maintain for HanzeNet is that users cannot trade more than they own, meaning they cannot make their balance go negative by trading. The only way a users' balance can become negative is by them consuming energy. This measure is implemented to disincentivize trade not directly linked to already existing energy production or consumption.

5.3 TRADING WITH EXTERNAL PARTIES

5.3.1 Layered trading

One of the goals of the HanzeNet team has always been to enable trading in small marketplaces, for example for a local neighbourhood, as this provides the most opportunity from an economic standpoint. We thus designed the system to function well for such a small, local marketplace, without requiring local marketplaces as the only way to run the system.

We conceptualized a local marketplace as a platform where people could trade, and envisioned these marketplaces consequently having to deal with local shortages / surpluses by, only when no internal resolution was possible, trading with external parties.

From a technical perspective, this meant figuring out how we could support this. Two different approaches were considered: We could choose between having individuals within the marketplace trade with external parties, or having the marketplace as a whole trade with other marketplaces.

For the scope of this research we made the design decisions to let local markets trade as a whole. Having individual members of the marketplace trade with external parties would require introducing some additional complexity to deal with the marketplace not being a closed system anymore, and it also doesn’t align with our goal of prioritizing local trade and only trading with external parties when local trade can no longer resolve the trade deficit.

Since a context represents a single marketplace, we opted to allow the context owner handle trade with other marketplaces on behalf of the marketplace. The context owner has total control over its local marketplace, including being able to mutate balances of participants. This enables the context owner to handle external trade by adding or removing balance from participants when these amounts represent trade with external parties. As an example, a context owner could use a mutation at the end of the month to bring all participants’ balances to zero, signifying that the energy was supplied by an external party, and bill those users for the corresponding amount of energy.

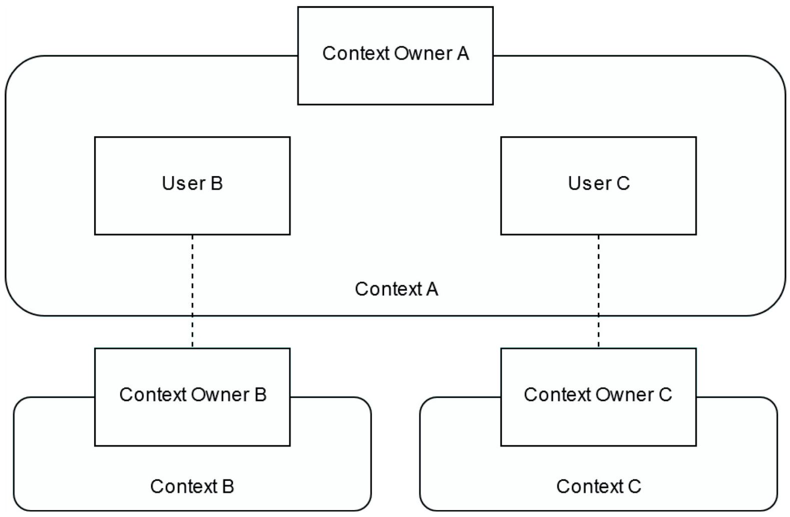


Figure 4: Two contexts represented by their context owner in a higher layered context

5.3.2 Stackability

An important design decision was to allow the design to stack, meaning that for layered trade between marketplaces we use the exact same contracts as are used within the local marketplaces themselves. This way, the entire local marketplace can be modelled as a single participating identity within a larger marketplace. See Figure 4 for a schematic representation.

Aside from reducing complexity by reuse, the stacking approach also helps us address another important issue for the system: its scalability. From an organizational perspective, it allows us to use the system in multiple, isolated layers, each administered by potential separate parties, while still allowing high-level trade between them. For example, a system in which each local neighbourhood is represented by a context could be encapsulated into a larger, city-wide context. From a technical perspective, it allows us to split up the entire market into small marketplaces which can be run separately, which we will not have a single, global processor processing every transaction, but are able to have multiple processors, one per local marketplace, making the systems performance only dependant on the size of the local marketplace, not on the size of the entire global market.

5.4 Representing value

One of the first issues to tackle while designing the smart contracts for HanzeNet was what users would be able to trade with them. For HanzeNet, it was clear we wanted to “trade energy”, but we weren’t sure how we would model that. We could either directly trade energy, represented by a unit of energy such as kWh, or use another base unit which could be traded.

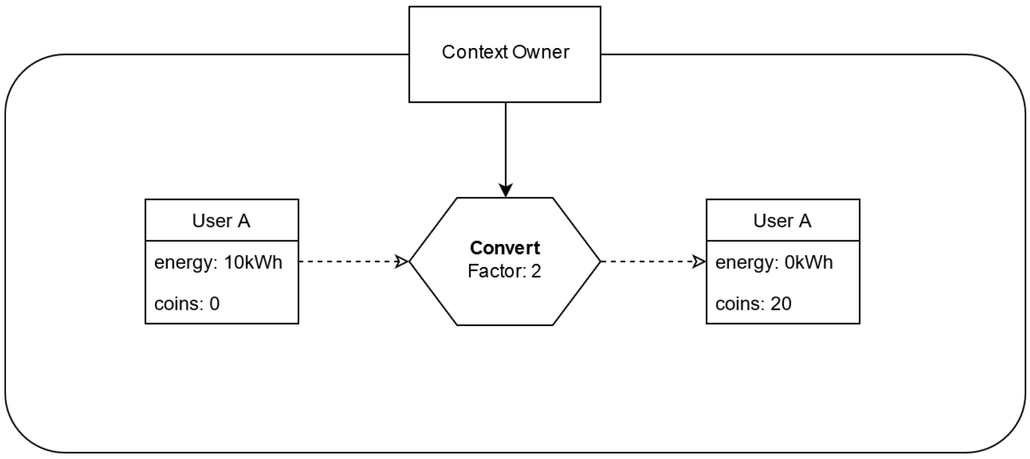


Figure 5: A context owner converting energy for user A

We concluded at some point we would have to represent the trade in terms of monetary value, since most consumers will be concerned with how much their energy use or production is costing or benefitting them. Thus, energy in our system would not have a constant value: 1kWh of energy today would have a different value than 1kWh in a few hours, a day, a month or even a years' time, as energy prices change constantly.

An important thing to note is that in our system, the amount of energy is determined at the edge, for example at each household's smart energy meter. This means that any transportation costs are not explicitly represented within our system. In real life, there is always a small loss when transporting energy. We decided not to model this in our system to reduce the complexity of the overall system. Similar to how the current energy suppliers include the costs of transport as a factor in their prices, we expect the market to attach the correct value to the energy being traded in our system including any loss component.

5.4.1 Conversion to coins

The first approach we designed used two separate units. The first was energy, produced and consumed in kWh, and stored in the users' energy balance together with the block number it was produced/consumed in. Users could not directly trade energy; it needed to be converted into "coins", the other unit of value within the system. Converting energy into coins was done using a specialized smart contract called "convert". The context owner would call this smart contract, specifying a previous block number and a "conversion factor". The amount of energy for every user produced in that specific block would be multiplied by the conversion factor and given to the user as coins. These coins could subsequently be traded by users.

With this design, the context owner could determine the value of energy per block number it converted. Since blocks are deterministically mined every few seconds, this effectively uses the blockchain to securely record when the energy was produced. The conversion factor provided control over the value energy was assigned for a given timeslot, allowing the context owner control over the valuation of energy.

5.4.2 Labels

While the conversion factor provided some control over valuation of energy, we wanted to go further and differentiate between energy from specific sources. To facilitate this, we introduced the concept of "labels". Coins have zero or more labels. Each of these labels is a string identifier, describing the coin has a certain property. An example of a label could be "solar", denoting the fact the coin was produced with solar energy.

Traders can trade coins with specific labels. They can also remove a label from a coin using the "reduce" smart contract, but they cannot add any labels to their coins. As a result, labels should only increase the value of a coin, since otherwise a trader would just remove the label to make the coin's value increase.

Each user has a set of labels which will be applied when their energy is converted to coins. They have no control over which labels get applied at conversion; this is completely controlled by the context owner.

Since coins are partitioned by their labels, removing a label is also essential to be able to use the coin for to even out a negative balance. For example, consider a user which has consumed energy, which is converted to a balance of -10 unlabelled coins. If they subsequently buy 10 "solar" coins from another user, these would not automatically balance out their negative coins, since they have a different set of labels. By removing the "solar" label from these 10 coins, they too become unlabelled coins and it bring the total coin balance to zero.

5.4.2.1 Laundering labelled energy

An important consideration is how these labels are assigned to prevent the problem of "energy laundering". We have to make a distinction between two different classes that can produce and consume energy: a "device", a singular piece of equipment such as for example a solar panel, and an "aggregation", which is an energy system consisting of a multitude of devices, such as a household. It is tempting to conclude that since a household is equipped with solar panels, any energy it produces therefor deserves a label like "green" or "solar". However, assigning such a label to an aggregation is problematic, since we cannot say for sure that energy produced by an aggregation is in fact all coming from the same source.

As an example of how a customer may launder energy, consider a homeowner who also drives an electric car. Now if this customer charges his or her car at a fossil fuel power station, drives home and connects the car to the household, draining its batteries, as far as HanzeNet is concerned, there is energy being

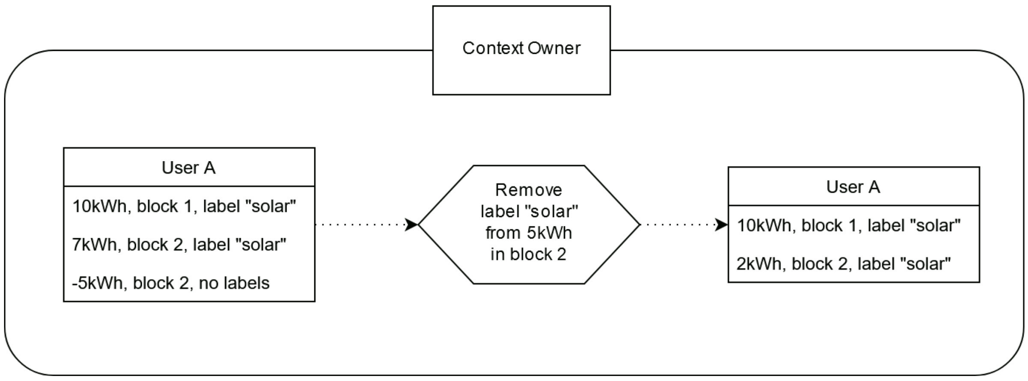

Context Owner
User A
10kWh, block 1, label "solar"
7kWh, block 2, label "solar"
-5kWh, block 2, no labels
Remove label "solar" from 5kWh in block 2
User A
10kWh, block 1, label "solar"
2kWh, block 2, label "solar"

Figure 6: Using the "reduce" smart contract to remove a label from energy in a specific block number

"produced" by the household, and it will be given the labels specified for that household. Clearly, this energy cannot be considered "green" or "solar" energy, and thus for HanzeNet we will not be assigning such labels to aggregations such as households.

5.4.3 Coinless approach

After introducing labels into the systems valuation was split into two steps: The conversion done by the context owner, and the additional value users would attach to energy with specific labels. In a subsequent iteration of our design we decided to unify this valuation and have energy be traded directly between users.

The conversion step was removed and energy now was partitioned both by block number and set of labels, but could be traded directly between users. In this version of the smart contracts, a real-world value is attached directly to the energy based on when (determined by the block number) and how (signified by the labels) it was produced.

In Figure 6 we see an example of a user with a balance spread over two different block numbers, block 1 and block 2, and two different labels: "solar" for all its produced energy, and no labels for the energy it has consumed. The user then uses a smart contract to remove the "solar" label from its energy, specifying the amount of energy (5kWh) and the block number (2) of the energy. By doing so, five unlabelled kWh's were created in block 2, which subsequently cancelled out the pre-existing negative balance in block 2 have been cancelled out and has effectively disappeared from our balance. Note that the balance for block 1 is unaffected by any of this.

Energy prices in this iteration of our model are completely left to the market to determine: Unlike the previous iteration the context owner does no longer determine a base valuation based on when the energy was produced. The block number recorded in each energy transaction allows the market to determine

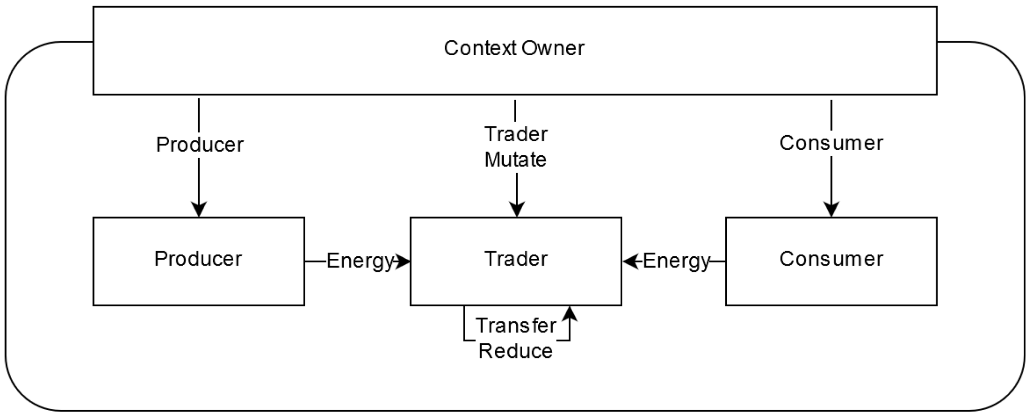


Figure 7: Multiple roles within a context and the smart contracts they use to interact

when that energy was produced or consumed, and determine the prices for it accordingly.

An additional advantage of this "coinless" approach is that it makes the use of stacked marketplaces more straightforward: Instead of revaluating coins into energy before energy can be traded at a higher layer, we can now remove energy from one market and introduce it into another without a conversion step. Additionally, labels and timings specified by block numbers can be preserved between marketplaces.

5.5 Permissions and Roles

To provide finer-grained access to the system we split up production, consumption and trading to different roles assigned by the context owner. A producer is allowed to produce energy by calling the Energy smart contract with positive amounts of energy, a consumer can likewise consume energy by calling the Energy smart contract with negative values. Finally, a trader is able to remove labels from energy (using the Reduce smart contract) and trade it with other traders using the Transfer smart contract. A single address may be both producer, consumer and/or trader, or this may be split in multiple addresses. Producers/consumers are linked to a specific trader, which means all energy produced/consumer by them will be recorded on the balance of that specific trader. See Figure 7 for a visual overview of the different roles and the contracts they use to interact.

This move to separated roles helps improve the security of the system by separating permissions. Besides being able to limit if a user is allowed to produce, consume and/or trade, it also has the practical implication that we are able to split up the private keys used for signing these transactions: We can now have a separate key in the smart energy meter which has the ability to produce and consume energy on the blockchain, and is protected by the legal boundaries surrounding this smart energy meter. The user can still trade energy using his/her own private key, but no longer has access to the key needed to produce energy, preventing malicious users from submit false energy production transactions using the key they have access to. It also protects energy production/consumption transactions for non-malicious users in case their key is compromised.

From a business perspective the move to separates roles enables a number of new ways to model business structures directly in the blockchain by having different parties control the trader keys. For example, one could imagine a leasing construct in which a user leases a set of solar panels from a company, for which we can then link the production directly to the trader address of the leasing company.

Finally, this separation can enable users to maintain a multitude of trader addresses, similar to how in cryptocurrencies users usually split their balance over a large number of wallets. This can help improve privacy as the link between who owns which trader address does not have to be made public, allowing users to trade without exposing their identity. Because unlike various cryptocurrencies we do not have full traceability of energy by following the chain of transactions that introduced them, this can be an effective tool to help protect individual's privacy within the system.

5.6 CRYPTOGRAPHIC PRIVACY

The last design improvement we looked at were various options to use modern cryptographic techniques to make certain information on the blockchain hidden while still being able to confirm the correctness of the system as a whole.

The two things that are most interesting to apply this to are to replace the public key approach we use for signing transactions, and replace the energy balances including their label functionality. Since all information on the blockchain is linked to a public key which signed the transaction, this public key provides a direct mapping of various actions performed by the same actor. Amounts and labels are privacy-sensitive in themselves as their reveal a lot about the energy patterns of the associated users.

To apply techniques for hiding balance we considered replacing the balance model of HanzeNet to be the best way forward since that way we can benefit from existing research and validation conducted in the context of privacy-preserving cryptocurrencies such as Zcash and Monero instead of having to develop and verify our own methodology.

Instead of storing the balance as a concrete amount, we would represent an addresses' balance as a sum of transactions, signed with a one-time spendable key available to only the user. These could subsequently be used in transactions between traders. Both sender, contents and receiver could then be secured with the use of various zero-knowledge techniques (see section 2.2.4) such as zk-SNARKs or ring signatures and transactions[18]. This allows third parties to be able to prove the total input and output of the transaction are equal, thus validating its correctness, while obscuring the exact amounts that were transferred between the parties.

A drawback of encrypting all transaction information is that the information will also no longer be visible to the context owner. The context owner, representing the grid operator or another administrating entity, would still need insight into the end result of trade to determine whether additional energy would need to be supplied from an external supplier and whom in the network should be billed for that. There are two possible options to address this:

1. Give the context owner direct insight into the transactions, for example by supplying the context owner with a read-only key for the transaction contents.
2. Let the context owner set up a special trader address for each of the households in the local marketplace which receives all consumption, and requiring them to transfer energy into that traders' balance to account for their energy use.

A drawback of option two is that administrative functions such as mutation would be limited in such a model; the only available operation feasible would be to erase a balance completely, unaware of its actual value.

A disadvantage imposed by replacing the simple public key setup with a system making the transactions unlinkable to users is that these links are still required by the administrating entity to allow them to bill or reimburse the correct user. In the simplest cases this can be remedied by instead of directly interacting with the balances of its users having the context owner redirect consumed energy to a trader address it controls itself, and requiring the user to transfer the required amounts or be billed accordingly. This does however not work in more complex setups which already delegate consumption to a different trader for different purposes.

This functionality was explored from a theoretical standpoint by means of literature study (see chapter 2) and a few small prototype tests with the available cryptographic primitives. This implementation proved troublesome in the existing software foundation used in the system. With the additional limitations in functionality described above and the required additional engineering effort to resolve these, we concluded that further implementing this cryptographic support was out of scope for this research project and limited our efforts to a theoretical exploration. We recommend this as an important piece of future research. In the final implementation, we will implement the contracts as described in 5.5, forgoing any cryptographic additions described in this section.

Table 1: Smart contract design decision overview

Decision area	Selected option	Rejected options
Trading model	Balance per user, negative values only caused by consumption, buys production to get to zero (5.2)	Previous transactions as input for new transactions, allowing negative balance
External trading	Via context owner (5.3), preferring trade with higher layers (5.3.1)	Having users trade directly with external parties outside context boundaries
Value representation	Energy per label (5.4.2) and time of consumption / production (5.4.3)	Conversion to intermediate "coins" (5.4.1), based on direct origin instead of label
Administrative functions	Context owner controls access (5.5), available labels (5.4.2) and can mutate balance of any user (5.2)	Uncontrolled access for traders

6 IMPLEMENTATION

The Validana platform is written in TypeScript and runs on the Node.JS platform. The source code of all core components is available on GitHub[26] and includes a set of Docker files to make it easy for anyone to run any or all of the components the system consists of.

6.1.1 CLI interface

While the entire core system is available on GitHub, the only way to interface with it was using a client library which needed to be integrated in a program. As part of this research we implemented a command-line interface program for executing common operations: *validana-cli.* It is made available as open source under the AGPLv3 license[27].

The program is designed using a structure of sub-commands to access its various functionalities, including the generation of keys, listing, creating, deleting and executing smart contracts, awaiting transactions completion and reporting its outcome. As input it accepts both parameters and files, including streaming from the standard input. All of its output is JSON. These properties allow for its commands to be composed using standard shell redirection techniques, allowing the user to easily execute more complex behaviours. For more information about the *validana-cli* program, we refer to its documentation[1] which is available in the official npm repository[28].

6.2 SMART CONTRACTS

All smart contracts are written in TypeScript, as this is the language the Validana platform itself was developed in as well and thus provided a good fit for integrating the smart contracts with it. The TypeScript code is compiled into JavaScript before being signed and submitted to the blockchain, where they are made available for execution.

[1] https://www.npmjs.com/package/validana-cli

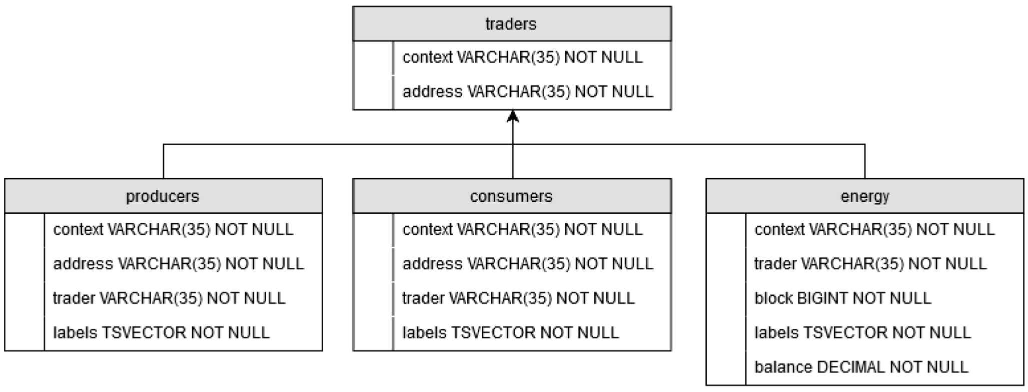

traders
context VARCHAR(35) NOT NULL
address VARCHAR(35) NOT NULL
producers
context VARCHAR(35) NOT NULL
address VARCHAR(35) NOT NULL
trader VARCHAR(35) NOT NULL
labels TSVECTOR NOT NULL
consumers
context VARCHAR(35) NOT NULL
address VARCHAR(35) NOT NULL
trader VARCHAR(35) NOT NULL
labels TSVECTOR NOT NULL
energy
context VARCHAR(35) NOT NULL
trader VARCHAR(35) NOT NULL
block BIGINT NOT NULL
labels TSVECTOR NOT NULL
balance DECIMAL NOT NULL

Figure 8: Diagram of database tables

All smart contracts in Validana have access to a PostgreSQL database via a reduced SQL interface. We will use this interface for recording a shared state accessible by all smart contracts. Figure 8 denoted the tables used for this purpose.

The final set of smart contracts described in this chapter are conforming the final iteration of design choices described in chapter 5.5. Other variations of the smart contracts are available as source code published alongside this document.

6.2.1 Producer

The Producer smart contract is used by the context owner to manage the list of allowed energy producers within its context.

6.2.1.1 Input

- The context address.
- The address of the producer.
- A boolean to determine whether that address is allowed or disallowed access.
- An address indicating which trader will receive the energy produced by this producer.
- A set of labels the energy will be associated with.

6.2.1.2 Logic

First the specified context address is compared to the submitter of the transaction to determine if the contract was called by the correct context owner.

If the allowed field is false, the producer specified by the address will be removed from the producers table. If the allowed flag is true, a check is performed to make sure the trader is in the list of allowed traders, and then all this information is stored in the producers table.

6.2.2 Consumer

The Consumer smart contract is used by the context owner to manage the list of allowed energy consumers within its context.

6.2.2.1 Input

- The context address.
- The address of the consumer.
- A boolean indicating whether that address should be allowed or denied.
- The address of the trader from whose balance the consumed energy will be subtracted.
- The labels used for subtracting that energy.

6.2.2.2 Logic

First the specified context address is compared to the submitter of the transaction to determine if the contract was called by the correct context owner.

When false is passed for the allowed field, the consumer matching the address will be removed from the consumers table. Else, the smart contract checks that the specified trader is allowed within the context, and then adds or updates the consumer data in the consumers table.

6.2.3 Trader

The Trader smart contract is used by the context owner to manage the list of allowed traders within its context.

6.2.3.1 Input

- The context address.
- The address of the trader.
- A boolean to indicate whether to add or remove the trader specified by the address.

6.2.3.2 Logic

First the specified context address is compared to the submitter of the transaction to determine if the contract was called by the correct context owner.

When a trader is requested to be added, its address is added to the traders table. When a trader is requested to be removed, first a check if performed to check if there are any producers or consumers linked to the trader. If that is not the case, the trader is deleted from the traders table, and by cascade any energy balances the trader had are removed as well.

6.2.4 Energy

The Energy smart contract is called by producers and consumers to register their energy production or consumption in the system.

6.2.4.1 Input

- The context address.
- The amount of energy as a floating-point number representing the amount of energy they want to record.

6.2.4.2 Logic

If the amount of energy is positive, it is considered production. The smart contract checks if the submitter of the transaction is in the list of allowed producers, and then adds the amount of energy to the balance of the trader specified in the producers table together with the specified labels and the current block number.

If the amount of energy is negative, it is considered consumption, and a similar process is executed. The submitter is checked against the list of valid consumers, the energy is subtracted from the associated trader for the given labels and the current block number.

If the amount of energy is zero (thus positive nor negative), the transaction is rejected.

6.2.5 Transfer

The Transfer smart contract is used by traders to transfer energy from to other traders within the same context.

6.2.5.1 Input

- The context address.
- The address of the receiving trader.
- The block number from which to transfer energy.
- The labels of the energy to transfer.
- The amount of energy to transfer as a floating-point number.
- An optional string description field.

6.2.5.2 Logic

The smart contract first performs a number of checks to ensure the both the trader and the receiver are allowed to trade within the context, that the trader and the receiver are not the same entity, and that the amount to transfer is a positive amount. It then looks up the balance of the trader and checks if the trader has a sufficient balance for the given block number and labels.

If all these checks succeed, the specified amount of energy is subtracted from the traders' balance and added to the receivers' balance.

Energy balances are stored in the energy table per context, trader, block and label set. This means a trader can have a large number of smaller balances at a time. The aggregation of those balances is the traders' total energy balance.

6.2.6 Reduce

The Reduce smart contract is used by traders to remove labels from their energy. As detailed in 5.4.2, this is an important operation to be able to combine distinct into a single balance, which traders can use to cancel out negative balances or to trade the requested type of energy with another party.

6.2.6.1 Input

- The context address.
- A block number.
- The set of original labels.
- The amount of energy to reduce as a floating-point number.
- A set of labels to remove.

6.2.6.2 Logic

The smart contract first checks if the trader is an allowed trader within the context, and if the amount of energy to reduce is positive. It then checks if the set of labels to be removed is a non-empty subset of the set of original labels. Finally, a check is performed to determine if the trader has a sufficient balance for the given block number and set of original labels.

Once all these checks completed successfully, the specified amount is subtracted from the balance of the trader for the given set of original labels, and then the balance for the relative complement of the set of labels to remove in the original set of labels is increased with the specified amount.

Labels in the HanzeNet database are stored as a text search vector type, which was chosen since it allows us to do all set operations on the database and makes it easy for applications to search for balances with specific sets of labels using built-in functionality of the PostgreSQL database system.

6.2.7 Mutate

The Mutate smart contract is used by the context owner to change the energy balance of any trader within its context. Its operation is similar to the Trade smart contract and be used for a variety of administrative purposes.

6.2.7.1 Input

- The context address.

- The address of a trader.
- A block number.
- A set of labels.
- An amount of energy as a floating-point number.
- An optional description string.

6.2.7.2 Logic

The contract checks if the submitter is the context owner, and if the trader is a valid trader within its context, and then mutates the balance of the given trader for the given block number and label set with the given amount.

6.3 EXAMPLE OF INTENDED USE

We provide a few step-by-step examples to demonstrate how the system functions, clarify how the various components work together, and how its abilities may be used in an actual deployment.

First, we set up a context by adding two traders, a producer, and a consumer:

Caller	**Contract**	**Parameters**	**Result**
Context Owner	Trader	Context: [context], address: [trader A], allowed: true	OK
Context Owner	Trader	Context: [context], address: [trader B], allowed: true	
Context Owner	Producer	Context: [context], address: [producer], allowed: true, trader: [trader A], labels: [“solar”]	OK
Context Owner	Consumer	Context: [context], address: [producer], allowed: true, trader: [trader B], labels: []	OK

We now have a context with four identities able to execute specific contracts. The producer and consumer each are linked to a trader and have a set of associated labels.

Next, we will insert some energy data into the system by having the producer and consumer call the Energy smart contract:

Caller	**Contract**	**Parameters**	**Result**
Producer	Energy	Context: [context], amount: 10	OK
Consumer	Energy	Context: [context], amount: -15	OK

We can now query the state of the system and we will see that the balances of the traders have been changed:

Trader	Block number	Labels	Amount
Trader A	100	“solar”	10
Trader B	100	(none)	-15

Since trader A was linked to the producer, it has received its production. Likewise, trader B was linked to the consumer and has now received its consumption.

Trader A and B make a deal, resulting in the transfer of some energy from trader A to trader B.

Caller	Contract	Parameters	Result
Trader A	Transfer	Context: [context], receiver: [trader B], block number: 100, labels: [“solar”], amount: 10	OK

Since trader A has trader away its entire balance for block number 100, it no longer appears on our balance overview. The balance for trader B looks as follows:

Trader	Block number	Labels	Amount
Trader B	100	“solar”	10
Trader B	100	(none)	-15

Trader B now has two separate balances for block number 100, since they have different sets of labels. Trader B wants to cancel out its negative balance with the energy it just acquired and will call the Reduce smart contract to remove the “solar” label from it:

Caller	Contract	Parameters	Result
Trader B	Reduce	Context: [context], block number: 100, original labels: [“solar”], amount: 10, labels to remove: [“solar”]	OK

All labels are now removed from the energy, turning it into unlabelled energy and combining it with the pre-existing amount trader B already had. The balance now looks as follows:

Trader	Block number	Labels	Amount
Trader B	100	(none)	-5

Now trader B is left with a deficit balance it can no longer resolve within its own context because there is no other trader left able to trade it some energy for that specific block. At this point, the context owner can initiate trade with an external energy supplier and use the Mutate smart contract to resolve the outstanding balance of trader B.

Caller	**Contract**	**Parameters**	**Result**
Context Owner	Mutate	Context: [context], trader: [trader B], block number: 100, labels: [], amount: 5, description: “Energy supplied by external party, you will be billed for this at the end of the month”	OK

Since it knows the identity of every trader, it can bill trader B for the amount of energy that was supplied from external sources.

7 EVALUATION

With the design of the system complete as outlined in chapter 4 and 5 and implementation of the smart contracts completed in chapter 6, we evaluate whether this proposed design and implemented set of functionalities is a suitable solution for HanzeNet.

7.1 GENERAL FEATURES

We designed a blockchain-based system for trading electrical energy. It implements basic trading capabilities using a balance-based system in which selected participants can trade energy. All energy balances and associated trade is part of a context, and we allow for multiple of these contexts to co-exist without impacting each other or requiring multiple blockchains to be operated (see 5.1).

Energy usage is measured using the standardized smart meters available in The Netherlands (see 4.1.1) and collected via both the EDSN network and via the P1 port directly from the energy meter. Collected energy data is timestamped upon transmission into the blockchain so timebound price differences are supported in the system and energy prices can change over time as the market determines.

We introduced labelled energy (see 5.4.2) as a way of tracking different sources of energy such as produced by solar panels, wind turbines, et cetera. Users can choose to specifically buy energy which has certain labels, allowing them to have direct control over sources they support through their purchasing of energy. It also allows users to choose to buy their energy from sources more specifically than they'd be able to with traditional energy suppliers, which usually only allow users to choose between greenly produced energy or just any energy, and specifically support local initiatives, specific techniques et cetera.

Administrative functions are provided within each context. The creator and owner of the context can choose which users can produce, consume and/or trade energy as well as change the balance of any user within that context (see 5.2). This fulfils the requirements for basic functionality as described in 3.1.1.

The design choice to make the entire system stackable (see 5.3.2) and use layered trading (see 5.3.1) both solves the problem of trading with external parties while keeping the local marketplaces small and scalable.

The application we built is easily extendable. The open source blockchain basis allows easy integration of third-party application, something we showcased with the development of the CLI interface for working with the blockchain platform (see 6.1.1). Additionally, in the design of all our smart contracts we specifically left room for alternative implementations in future iterations such as flexibility trading, something which is made possible by the ability to deprecate and replace smart contracts in Validana (see 4.2).

Our system is complementary to existing systems for smart grids and energy trading such as TRIANA, PowerMatcher, SolarCoin and others (see 2.1). Data from our system can fed into those systems to optimize using market data or trade energy externally, while information from those systems can fed back to enable users of our system to benefit from prediction capabilities and improved decision making.

7.2 Security and privacy

Using an open source blockchain as the basis of its system gives us guarantees about correctness which can be checked by any interested party in a decentralized manner. This helps build trust in the system as participants have full visibility in how it functions and can verify the trusted party conforms to the agreed rules.

It also provides us with a number of security properties required as per 3.1.2. Since all transactions are cryptographically signed and stored in a distributed fashion on all participating nodes, once a transaction is accepted it is impossible for it to be tampered with without any of the nodes detecting and refusing these changes.

Additionally, our blockchain platform allows us to deprecate smart contracts which allows us to replace them as any security issues are found. By enforcing this ability through a trusted miner this also does not suffer from some of the common vulnerabilities in contract upgrade patterns[29].

The separation of roles (section 5.5) provides an additional layer of security by having separate keys for different operations within the blockchain, which limits the impact of a key being compromised.

On the privacy front (see 3.1.3), the primary protection is provided in the form of the ability to have a multitude of producer, consumer and trader addresses. Only the context owner has to know to which real-life entities these addresses map (see 5.5). Using common metadata hiding techniques such as splitting the energy transactions over a number of transactions from multiple producer addresses with amounts modulo a fixed amount, specific usage could be obscured further. The same technique applies to consumption and trading. This makes it hard for outsiders to see how much energy is being used by a specific household as the link between addresses on the blockchain and households cannot be made, protecting the privacy of the users in those households.

The system design allows for expansion with additional cryptographic techniques to further improve privacy and hide user data as explored in 5.6.

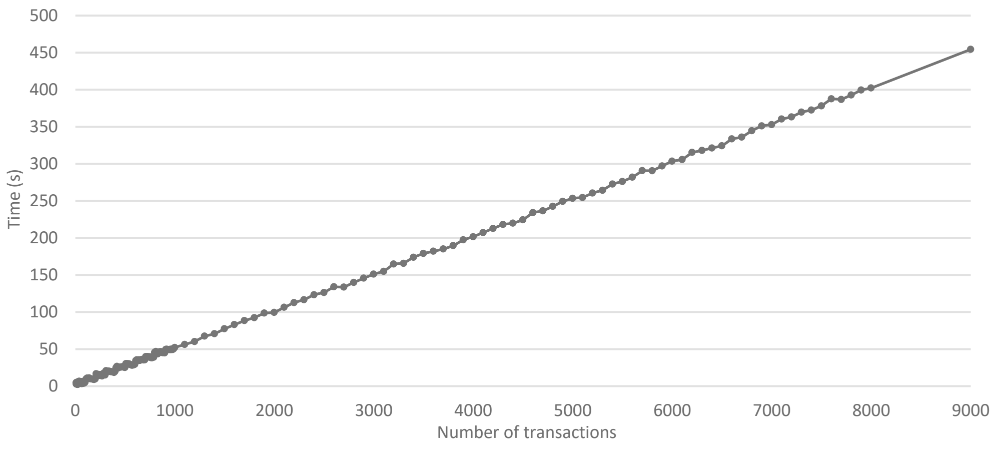


Figure 9: Time to process number of transactions

7.3 Performance and scalability

To evaluate the performance of our system, we set up a new virtual private server at a commercial datacentre and set it up with a complete copy of the system. We created the smart contracts on it and then generated large amounts of transactions for all those contracts. We submitted these transactions for processing and measured precisely how long it took until the last transaction was done processing.

The system chosen was a single core, 2Ghz CPU, 2GiB memory, VPS running Ubuntu 18.04.1, one of the cheapest offerings available in Europe at the time of writing. By specifically opting to benchmark on such a low-performance system we argue that if our implementation can run. The Validana blockchain was running in its default configuration in Docker, and all transactions were first generated locally (as to not have the machine do both generation and processing of transactions simultaneously which could impact our measurements) and then we recorded the time from when we started transmitting the first transaction to the Validana server until the last transaction was reported by the server to have been processed.

See Figure 9. This shows the time to process a batch of transactions over averaged over five test runs. Scaling up the number of transactions correlates linearly with the time until all transactions are processed. In this experiment we were able to process on average 37.86 transactions per second, which includes the overhead of waiting for the transaction to be reported as processed. Given that our primary and fastest data source is the P1-port (see section 4.1.2), which delivers measurement data once every ten seconds, this is equivalent to connecting over 378 households, which is well within the expected size of a "small, local marketplace" which the system is aimed to provide for. Thus, this experiment indicates we are well within our performance budget, especially considering the relative low specifications of the testing environment.

Additionally, since the smart contracts are designed to function as a stackable system, we can scale horizontally by splitting different contexts onto different processors, meaning that by keeping the local marketplaces small enough to run on a single processor scaling is only limited by how many processors we are able to run. Since the philosophy behind HanzeNet already focusses on building small scale marketplaces for local communities this is expected to be a good fit for applying this type of scaling.

8 Conclusion

This research started off based on the previous work by the HanzeNet start-up to explore possibilities for digital trading of energy in small communities. This research provides HanzeNet with a system design for a blockchain-based energy trading system. Insight is provided into the various design choices which are to be considered for an energy trading platform. Additionally, a working core implementation of the described system is provided. The resulting system enables users to trade their energy securely with others in their community. Produced energy is timestamped and can be labelled to allow economic value to be attached to energy produced. A stackable design allows the system to be deployed at multiple layers, allowing households to trade in small local communities, which can trade as a whole within their city, which can trade with neighbouring cities, and so forth. This design keeps the focus for its users on local communities while allowing the architecture to perform at practically any scale.

At the residential level, once energy is produced or consumed we acquire usage data via both local readouts of smart energy meters and remote readouts through EDSN. This data is transmitted to our blockchain and securely linked to the current time and a set of labels and is subsequently counted towards the balance of a predefined trader which represents the household. This trader can exchange energy with other households in the local community. The labels allow them to choose from which energy source to buy their energy, allowing customers to make choices to for example buy environmentally friendly produced energy or support local initiatives with their energy consumption. The context owner enables users to also trade with external parties, whether that be traditional energy supplier companies or neighbouring communities.

We observed that the software implementation scales well to practically any size and the feature set of the overall system is considered a good fit for the goals of start-ups like HanzeNet, allowing them to bringing added value to local communities. The core implementation of the system is made available entirely as open source software[2] for the benefit of the larger community. Detailed instructions are available to aid readers in trying our core implementation for themselves, as well as additional tooling[3] to make it easy to run this software on all popular and some unpopular platforms and operating systems.

[2] https://github.com/DvdGiessen/hanzenet-smart-contracts

[3] https://www.npmjs.com/package/validana-cli

8.1 Future research

There is a number of open questions resulting from this work which can be looked into by future research.

Currently we build upon standard Internet connections for our connectivity. While this is a feasible option in most cases in The Netherlands, there might be use cases where availability of Internet comes with additional costs prompting the exploration of other options. Local radio-based transmission of transactions between nodes using readily available technologies as using LoRa might be an interesting option for those use cases. We expect some optimization will be required to reduce the amount of data exchanged; future research could look into about alternative trading models optimized for low-bandwidth peer to peer trading.

One direction that was theoretically explored but was out of scope to implement and iterate upon was the use of an alternative balance implementation for the smart contracts, making use of a sum of transaction similar to how many cryptocurrencies operate. While this model in itself only introduces unwanted complexity, it has the potential to be used in combination with homomorphic encryption techniques such as employed in Zcash and Monero, which may allow us to hide the entire contents of transactions while still maintaining the ability to have decentralized correctness checking. This will likely introduce some limitations on the trading model, so a trade-off will have to be made whether this is worth the improved privacy properties.

Integration with existing systems for smart grids can help bring additional information into the system to help consumers choose their energy smarter and potentially help increase the efficiency for both participants and grid operators. Future research may look into how these gains can be made available to allow traders in our blockchain to better determine the value of their and their peers' energy. Smart systems can also greatly benefit from flexibility trading. Allowing users to register intent to use energy in future blocks and trade in that flexibility might be a useful addition to our system which complements the prediction capabilities of existing systems for smart grids by linking these abilities to a real-time marketplace.

References

[1] Vincent Bakker. 2012. *Triana: a control strategy for Smart Grids: Forecasting, planning & real-time control.* DOI= https://doi.org/10.3990/1.9789036533140

[2] Kok, J. K., Warmer, C. J., & Kamphuis, R. 2006. *The PowerMatcher: Multiagent Control of Electricity Demand and Supply.* IEEE Intelligent Systems, 21(2), 89-90.

[3] The Flexiblepower Alliance Network. Retrieved 2018-09-07. https://flexible-energy.eu/

[4] Universal Smart Energy Framework. Retrieved 2018-09-07. https://www.usef.energy/app/uploads/2016/12/USEF_TheFrameworkSpecifications_4nov15.pdf

[5] Nick Gogerty, Joseph Zitoli. 2011. DeKo: An Electricity-Backed Currency Proposal. DOI= https://doi.org/10.2139/ssrn.1802166

[6] SolarCoin documentation. 2014. Retrieved 2018-09-07. https://solarcoin.org/en/solarcoin-documentation

[7] About the Energy Web Blockchain. Retrieved 2018-09-07. https://energyweb.org/blockchain/

[8] About PowerPeers. Retrieved 2018-09-07. https://www.powerpeers.nl/about

[9] Amber Voets. 2017. *Blockchain Technology in the Energy Ecosystem: An explorative study on the disruptive power of blockchain technology in the Dutch energy Ecosystem.* http://resolver.tudelft.nl/uuid:ce37e094-cf9e-4976-afde-2e0ecdf8880a

[10] Satoshi Nakamoto. 2008. *Bitcoin: A Peer-to-Peer Electronic Cash System*. https://bitcoin.org/bitcoin.pdf

[11] Ittay Eyal, Emin Gun Sirer. 2013. *Majority is not Enough: Bitcoin Mining is Vulnerable.* https://arxiv.org/abs/1311.0243

[12] Surae Noether, Sarang Noether. 2014. *Difficulty Adjustment Algorithms in Cryptocurrency Protocols.* Retrieved 2018-09-07. https://www.overleaf.com/articles/difficulty-adjustment-algorithms-in-cryptocurrency-protocols/ytcxbjvzrpbp/viewer.pdf

[13] Nick Szabo. 1996. *Smart Contracts: Building Blocks for Digital Markets.* Retrieved 2018-09-07.

http://www.alamut.com/subj/economics/nick_szabo/smartContracts.html

[14] Ethereum Blockchain App Platform. Retrieved 2018-09-07. https://ethereum.org/

[15] Vitalik Buterin. 2013. *A Next-Generation Smart Contract and Decentralized Application Platform*. Retrieved 2018-09-07. https://github.com/ethereum/wiki/wiki/White-Paper

[16] Shafi Goldwasser, Silvio Micali, Charles Rackoff. 1985. *The Knowledge Complexity of Interactive Proof Systems.* DOI= https://doi.org/10.1137/0218012

[17] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, Madars Virza, *Zerocash: Decentralized Anonymous Payments from Bitcoin*, proceedings of the IEEE Symposium on Security & Privacy (Oakland) 2014, 459-474, IEEE, 2014. http://zerocash-project.org/paper

[18] Shi-Feng Sun, Man Ho Au, Joseph K. Liu, Tsz Hon Yuen. 2017. *RingCT 2.0: A Compact Accumulator-Based (Linkable Ring Signature) Protocol for Blockchain Cryptocurrency Monero.* DOI= https://doi.org/10.1007/978-3-319-66399-9_25

[19] Walt Disney. *Dragonchain blockchain platform open source release.* https://github.com/dragonchain/dragonchain

[20] HanzeNet visualisation. Retrieved 2018-09-07. https://hanzenet.com/

[21] Autoriteit Consument en Markt. *Netcode elektriciteit, artikel 2.1.3.1.* https://wetten.overheid.nl/BWBR0037940#Hoofdstuk2_Paragraaf2.1_Sub-paragraaf2.1.3_Artikel2.1.3.1

[22] Energie Data Service Nederland. *About EDSN.* Retrieved 2018-09-07. https://www.edsn.nl/ons-verhaal/

[23] LoRa Alliance. *Technical Overview of LoRa and LoRaWAN*. https://www.lora-alliance.org/resource-hub/what-lorawantm

[24] Coinversable. *About Validana.* Retrieved 2018-09-07. https://validana.io/

[25] Coinversable. *The Validana Processor.* Retrieved 2018-09-07. https://github.com/Coinversable/validana-processor/wiki

[26] Coinversable. *Open source release of Validana.* https://github.com/coinversable/validana-core

[27] GitHub. Open source release of *validana-cli.*
https://github.com/DvdGiessen/validana-cli

[28] npm Inc. *About npm.* Retrieved 2018-09-07.
https://www.npmjs.com/about

[29] Josselin Feist. *Contract upgrade anti-patterns.* Retrieved 2018-09-07.
https://blog.trailofbits.com/2018/09/05/contract-upgrade-anti-patterns/