Hukkinen, Jussi; Mattila, Juri; Smolander, Kari; Seppälä, Timo; Goodden, Tobias

Skimping on Gas – Reducing Ethereum Transaction Costs in a Blockchain Electricity Market Application

Published in:
Proceedings of 52nd Hawaii International Conference on System Sciences

Published: 08/01/2019

Document Version
Publisher's PDF, also known as Version of record

Skimping on Gas – Reducing Ethereum Transaction Costs in a Blockchain Electricity Market Application

Taneli Hukkinen
Aalto University
hukkinen@iki.fi

Juri Mattila
ETLA / Aalto University
juri.mattila@etla.fi

Kari Smolander
Aalto University
kari.smolander@aalto.fi

Timo Seppälä
Aalto University / ETLA
timo.seppala@aalto.fi

Tobias Goodden
Fortum Oyj
tobias.goodden@fortum.com

Abstract

In recent years, information systems have not been largely evaluated by their operating costs, but mainly by their strategic benefit and competitive advantage. As blockchain-based decentralized applications become more commonplace, representing a shift towards fully consumption-based distributed computing, a new mode of thinking is required of developers, with meticulous attention to computational resource efficiency.

This study improves on a blockchain application designed for conducting microtransactions of electricity in a nanogrid environment. By applying the design science research methodology, we improve the efficiency of the application's smart contract by 11 %, with further improvement opportunities identified. Despite the results, we find the efficiency remains inadequate for public Ethereum deployment.

From the optimization process, we extrapolate a set of general guidelines for optimizing the efficiency of Ethereum smart contracts in any application.

1. Introduction

Over the past decade, digital platforms have transformed the provision of applications in digital networks [11,33,39]. Traditionally, these platforms have been built using centrally controlled system architecture [1,4]. Recently, however, it has become increasingly popular to provide applications via decentralized blockchain smart contracts, governed by algorithmic incentives [5,36]. As the computational resources of these blockchain networks are allocated and priced according to free market mechanics [9,13,21,24], resource-efficiency and cost-optimization are placed in the center of application development.

The efficiency of information systems and business computing applications has not received wide attention in research lately. Ever since the 1980s, IT systems have not been mainly evaluated by their operating costs, but rather by their enhanced market access, product differentiation, strategic benefit and competitive advantage [20]. The systems have been largely perceived as investments with long-term effects and benefits [34], across their whole lifecycle [37] and most often emphasizing infrastructures, human resources and IT-enabled intangibles [2].

The advent of grid and cloud computing has gradually changed this long-term investment-based view to a short-term utility-based one. This change has produced some novel theoretical models on prices, revenues, and resource utilization [3,26]. Also, more general considerations on the new economic models of cloud computing [6] and grid computing [7] have been published.

As blockchain technology moves computing as utility even further, a new mode of thinking is required of software developers, with meticulous attention to computational efficiency. While some theoretical research has focused on embedded costs [15] and institutional changes [12] of blockchain, so far there has been little in the way of formal research into the optimization of blockchain-based smart contracts.

In the absence of a centralized authority, blockchain networks can consume vast amounts of electricity to maintain consensus [21]. The Ethereum smart contract platform, for example, has been estimated to consume more electricity than the country of Iceland, constituting approximately $1/1000^{th}$ of the world's electricity consumption in total [14]. Advancing the understanding and developing best practices in the optimization of blockchain-based smart contracts is important to ensure that the maximum innovation output and utility is achieved in return for

URI: https://hdl.handle.net/10125/60123
ISBN: 978-0-9981331-2-6

the vast energy consumption of such systems and their strain on the environment at large [27].

The objective of this paper is to improve and analyze the feasibility of an experimental distributed blockchain market application designed for conducting microtransactions of electricity in a nanogrid environment [19,23]. The paper applies the design science research methodology by Peffers *et al.* [29] to explore novel ways to improve the efficiency of the application's smart contract and to reduce its operating costs in the Ethereum network. During this process, we introduce a new set of general guidelines for optimizing the efficiency of Ethereum-based smart contracts.

By implementing two of the identified improvement opportunities, we benchmark the efficiency of the smart contract improved by 11 %. While not adequate for economic feasibility on the public Ethereum blockchain, we establish that further improvements are likely to be possible with more radical reformations to the source code, redefined market mechanics, and the use of an alternative deployment environment. Overall, the study demonstrates that decentralized applications should implement their own functional layers on top of the smart contract, keeping the contract as simple and as low in resource consumption as possible.

This paper is structured as follows. Section 2 provides a background for the paper by explaining some of the core technological concepts. Section 3 describes the blockchain electricity market smart contract analyzed and improved in this paper. Section 4 documents the process of optimization conducted in this study, starting with the problem identification and ending with the evaluation of the outcome. Section 5 contains some discussion on the findings. Finally, Section 6 presents the conclusions.

2. Technological descriptions

2.1 Blockchain technology

Blockchain technology enables the creation of decentralized, distributed and replicated digital ledgers. The technology itself consists of components such as peer-to-peer networking, public-key cryptography, digital tokens, a decentralized consensus algorithm and a tamper-resistant chain of blocks used to store database modifications [28,32]. Originally, the term was used solely in reference to the cryptographically concatenated data structure employed by blockchain systems such as the Bitcoin cryptocurrency network. Later on, however, the term has taken the broader meaning of the technological composition behind such systems at large, in various configurations [25].

While cumbersome and often more expensive to operate than centralized systems, blockchain networks can be useful due to their tamper-resistant and non-hierarchical quality. Built on public open-source protocols, they can also help foster the growth of digital ecosystems with a bottom-up approach different from conventional centralized platforms. For this reason, cryptocurrency is an essential component of any permissionless blockchain. It can be used to facilitate economic incentives for the pseudonymous participants in the network to collaborate with one another and to maintain the integrity of the shared blockchain database [11,21,24].

2.2 Smart contracts

Blockchain technology has enabled the creation of decentralized execution environments for smart contracts. In comparison to conventional digital contracts, blockchain-enabled smart contracts expand the digital contracting space by enabling tamper-proof storage and decentralized algorithmic execution [10]. Moreover, diverging from contracts concluded in the form of action, speech or writing, a smart contract is characteristically a computer program built in code [5].

The concept of a smart contract is best explained by an example. A vending machine takes coins and a push of a button as inputs and dispenses change and a product as outputs. The vending machine always acts deterministically according to the same set of instructions. Inserting coins into a machine is seen as a sign of agreement with the terms of the vending machine's embedded contract. The vending machine is able to autonomously manage the process of handling a customer's money and selling a product without an external adjudicator [31]. Much in the same way as vending machines, digital smart contracts can essentially be characterized as cryptographic "boxes" containing value that only unlocks upon the fulfilment of the preconditions determined in their design [5]. Thus, they are able to handle the fulfillment of the contractual clauses embedded in their software without human intervention. Furthermore, they are able to penalize breaches of contract and prevent any unauthorized changes to their code [22,31].

For this paper, we define smart contracts as digital programs that: a) are written in computer code and formulated using programming languages, b) are collectively stored, executed and enforced by a distributed blockchain network, c) can receive, store, and transfer digital assets of value, and d) can execute with varying outcomes according to their specified internal logic [22].

2.3 Ethereum

In recent years, blockchain technology has helped in overcoming the obstacles smart contracts previously faced. One such milestone was the launching of the decentralized application platform Ethereum in 2015. It offers a Turing-complete programming language for writing smart contracts and allows the deployment of smart contracts into its blockchain [5].

Ethereum smart contracts can serve as a back-end for decentralized applications. The benefits of using an Ethereum smart contract instead of a new blockchain include faster and easier development, bootstrapped security, and being able to communicate with other decentralized applications deployed in the Ethereum blockchain [5].

Ethereum utilizes a transaction fee system to prevent denial-of-service attacks and to incentivize efficient smart contract deployment. A transaction fee—or gas consumption—is determined by the amount of computational work, network bandwidth and storage space the transaction consumes [5]. This type of a fee system, instead of a simpler model, such as the one in Bitcoin, is required due to the Turing-complete programming language in which Ethereum smart contracts are implemented. The fee system must be able to charge on a per computational step basis in order to avoid the execution of infinite loops with infinite resource expenditure.

The contracts' code is run on Ethereum Virtual Machine (EVM) to ensure that the execution environment is always identical and hardware-independent. Every operation executed in the EVM is executed on every full Ethereum node, as nodes must validate new blocks before appending the blockchain.

3. A blockchain-based electricity market application

This section describes the Ethereum smart contract of the blockchain electricity market application analyzed in Section 4. The application provides a decentralized marketplace where nanogrid participants can sell excess electricity to one another. Since the marketplace is implemented as an Ethereum smart contract, it has no need for a central authority that could censor offers, steal users' deposits, or do front-running. In addition to providing a platform for making contracts and trading electricity, the application also inherently facilitates payment processing, using Ethereum's native cryptocurrency, ether.

For a smart contract to be useful for energy trading, a system composing of the following components is required: 1) a smart contract facilitating the marketplace; 2) a small-scale electrical network for delivering electricity (*i.e.* a nanogrid); 3) smart meters serving as access points to the nanogrid; and 4) a reputation system for assessing the trustworthiness of smart meters. However, in this paper, we do not specify the non-software components or the reputation system involved.

3.1 Electricity markets today and in the future

Due to the non-storable nature of electricity, supply and consumption must be constantly balanced in electrical grids. With multiple producers and consumers interacting with the grid, determining the price for each instance where supply and demand meet is vital for grid balancing. In most developed markets, price formation occurs at power exchanges, such as Nord Pool, where a range of power delivery contracts are used to balance the supply and the demand [35].

In EU countries, the percentage-share of renewable energy in gross final energy consumption has risen from 9 percent to 16.7 percent in a ten-year time span between 2005 and 2015 [16]. Solar photovoltaic generation and many other renewable systems allow distributed generation near the points of demand, reducing transmission losses [38]. Such localized power production could transform the current vertical, centralized energy system into a more horizontal and distributed one. Conventional power generation, such as coal-fired power plants, allow power output to be steered to better match electricity demand. Therefore, given the price inelasticity on the demand side, grid balancing has traditionally taken place at the supply side [8]. However, with the growing share of intermittent renewable energy sources, future energy systems may require demand-side flexibility. Real-time pricing has been shown to affect demand and can be used to reduce curtailment [17].

With photovoltaic systems and wind energy systems becoming accessible and affordable, energy generation may shift from large energy companies to smaller organizations or consumers themselves. This could spur the rise of small-scale electrical networks, microgrids and nanogrids, that can be isolated from the main grid thanks to local generation, consumption and control. In such circumstances, a distributed market mechanism would be beneficial for nanogrid balancing and enabling interconnectivity between separate nanogrids.

3.2 The process of transacting electricity

In the smart contract's logic, the process of an electricity transaction unfolds as follows (see Figure 1).

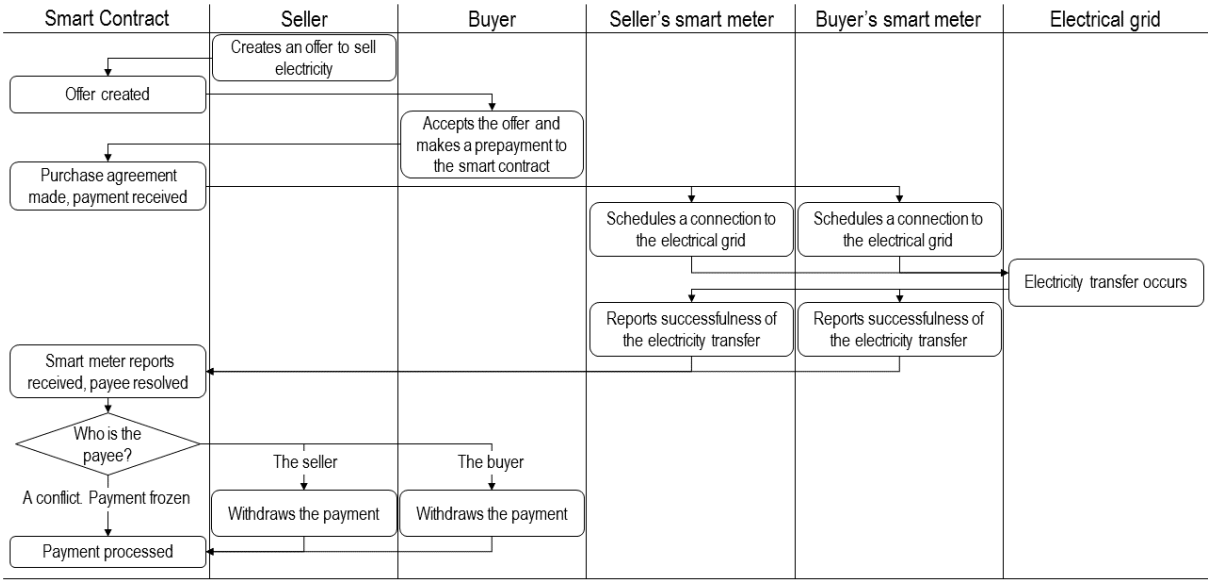

Figure 1. A flowchart of the process of trading electricity in the marketplace facilitated by the smart contract. Each actor's actions are presented in a column of their own.

First, a user wishing to sell electricity makes a transaction into a smart contract situated in the blockchain, constituting a selling offer and specifying the terms of a proposed agreement. Any other user wishing to accept the offer and its terms may do so by submitting a prepayment for the purchase into the smart contract, thus expressing their intent to enter into a contract. The prepayment, at this stage, is not yet transmitted to the seller, but rather held by the smart contract facilitating the trade. This way, it would be unwise for the buyer to back out of the trade, already having irrevocably committed to making the payment. The seller, however, may fail to provide enough electricity, thus breaching the contract. Thus, a reputation system, that can penalize economically, is needed to prevent producers from doing so.

In the smart contract facilitating the transaction, two timestamps are specified: one for the starting time of the electricity transfer and one for the end of it. When the starting time occurs, the seller's and the buyer's smart meters allow the buyer and the seller to access the nanogrid and to transfer electricity. The smart meters act as gatekeepers for the nanogrid, preventing unauthorized electricity consumption or overload from occurring. The smart meters also record electric current and voltage during transmission, and using that data, they are able to tell whether the electricity transfer was successful.

After the time period allocated for electricity transfer is over, the two smart meters autonomously create blockchain transactions, reporting on the successfulness of the electricity transfer on their owner's side. Based on the reports, the smart contract decides who can withdraw the prepayment made earlier by the buyer. The desirable outcome is that both reports implicate a successful trade, and the seller may withdraw the payment.

Sellers can populate the electricity market smart contract with multiple offers and buyers can accept any offer that suits them best. All offers and agreements are processed fully transparently. As a result, a fair market price should be discoverable by the users.

3.3 Nanogrid characteristics

The electricity market smart contract has no notion of transmission costs or transfer losses in the electrical grid. Therefore, it is only suited for a compact nanogrid network where transmission costs of electricity are small enough to render them irrelevant when considering who to transact with. The electricity market application examined in this report assumes a fully interconnected nanogrid topology. In other words, each access point is directly reachable from every other point without having to proxy through another access point. The nanogrid needs to have a commonly agreed voltage and current type (alternating or direct current).

All participants of the nanogrid are connected to the nanogrid via a smart meter. Furthermore, although energy storing devices are not required from users, an ample use of batteries is expected locally behind the smart meters. To prevent outages in an interconnected nanogrid, each individual electricity transmission must happen exactly as planned. Throughout the timeframe specified in the smart contract, the seller is required to supply the network with the specified amount of

power. In a parallel circuit, however, the lack of a consumption load on the buyer's side is non-critical.

3.4 Sale and purchase agreements

The contract includes the four specifiers necessary for any electricity transfer contract: delivery period (start and end time), recipient (seller's smart meter's Ethereum address), size of the transfer (amount of electricity), and price. An id field is also added, containing a 256-bit unique identifier for the contract. When a buyer accepts a seller's offer, the buyer's Ethereum address and their smart meter's Ethereum address are added into the contract. Finally, when the scheduled transmission of electricity is over, the seller's and the buyer's smart meters report whether in their view the transmission was successful or not.

3.5 Smart meters

Ethereum smart contracts only have access to data stored in the Ethereum blockchain. For this purpose, a smart meter is required for every user to act as an oracle (*i.e.* a trusted data feed into the blockchain) and to control the connection to the nanogrid. The smart meters either need to run their own Ethereum nodes or connect to an external trusted Ethereum node. For creating transactions, they must have their own Ethereum address and have access to their private key. The owners are expected to top up their smart meter's account with ether to maintain their ability to pay transaction fees.

Users of the electricity market application should not have full control of their smart meter, the software installed on it, or the private key of the smart meter's Ethereum address. Users should also not be able to bypass the smart meter and draw electricity directly from the nanogrid, as this would allow them to steal electricity and to manipulate results of the smart meter reports. Therefore, smart meters should be issued by a trusted party, presumably a company that manufactures or installs the smart meters.

3.6 Smart contract

The electricity market application involves two smart contracts written in the Solidity programming language. The `SmartMeters.sol` contract contains a mapping of smart meter addresses and their owners which the `ElectricityMarket.sol` contract is able to look up. In the `ElectricityMarket.sol` contract, there are five public, state changing functions: 1) `makeOffer`, used to signal the willingness to sell electricity; 2) `acceptOffer`, used to accept standing offers; 3) `buyerReport` and 4) `sellerReport`, used by the smart meters to report the successfulness of a transaction into the blockchain; and 5) `withdraw`, used to withdraw the deposit of a specified offer.

3.7 Scalability

A successful trade in the electricity market smart contract requires five Ethereum transactions, consuming a total of over 400 000 gas. According to the website etherscan.io, on the 12th of June 2018, the average gas limit per block in the public Ethereum chain was 7 996 828 and the average block time was 14.68 seconds. As a point of comparison, on the 13th of October 2017, the gas limit was 6 712 392 and the block time 31 seconds.

Were the public Ethereum chain to exclusively process transactions that call functions of the electricity market smart contract, the blockchain would be able to handle roughly one trade per second. A throughput like this is easily enough for a single community's nanogrid. However, it is not enough for widespread adoption of the application, processing trades of multiple nanogrids along with the transactions of all the other smart contracts on the public Ethereum chain.

Some method of increasing the scalability of the system is clearly needed. Either the application needs a significant reduction in gas consumption, or it needs to be executed in an environment other than the public Ethereum blockchain.

4. Applying the design science research methodology

In this section, design science research methodology by Peffers *et al.* [29] is applied to the electricity market smart contract. Design science is a suitable research approach when an innovative, purposeful artifact is created and evaluated for a special problem domain [18]. In our study, we built an artifact and evaluated it to ensure its utility for the problem. By using the design, we demonstrate why the general blockchain solution must be improved in this particular problem domain. We selected design science as our methodology because it offers a rigorous method for designing, building and evaluating the artifact.

The methodology consists of a process model involving six activities: 1) problem identification and motivation, 2) defining the objectives for a solution, 3) design and development, 4) demonstration, 5) evaluation, and 6) communication. Since communication, as described by Peffers *et al.* [29],

encompasses the entire research article, we will address activities 1–5 in this section.

4.1 Problem identification and motivation

In any public blockchain, a system of transaction fees is needed to arrange transaction priority, to prevent denial-of-service attacks, and to create an incentive for running the network and maintaining its consensus.

To complete a successful trade in the electricity market, the seller needs to spend roughly 290 000 gas on three transactions (calling the functions `makeOffer`, `sellerReport` and `withdraw`) and the buyer roughly 110 000 gas (calling the functions `acceptOffer` and `buyerReport`). From 13th of October 2017 to 28th of August 2018, according to etherscan.io, the transaction cost of a trade to the seller varied between $1.07 and $23.45 (USD), and to the buyer between $0.41 and $8.90, depending on the fluctuations of the gas price and the ether price during the observed time interval (see Table 1).

Table 1. The cost of completing a trade in the electricity market at various points in time.

Date	*Gas price (Gwei)*	*Ether price (USD)*	*Total cost (USD)*
13 Oct 2017	26.49	$302.89	$3.21
14 Jan 2018	58.39	$1385.02	$32.35
15 Feb 2018	21.57	$920.11	$7.94
12 Jun 2018	14.04	$531.15	$2.98
28 Aug 2018	12.80	$288.02	$1.48

The cost of performing a trade sets a lower bound for the amounts of electricity that can be transmitted and thereby limits the possible uses of the application. For instance, it can never be profitable to sell electricity for one dollar, if the seller needs to pay two dollars in transaction fees.

Implementing an electricity market as a blockchain application has several advantages compared to a centralized service but these advantages can be thwarted by high operating costs. The problem identified in the electricity market smart contract is that, at the estimated cost level, the public Ethereum blockchain would not be a viable deployment environment for the electricity market application.

4.2 The objectives for a solution

In Ethereum, transaction cost is simply gas price multiplied by the amount of gas consumed by the transaction [36]. To reduce transaction costs, at least one of these two values should be lowered. Gas price should generally not be controlled by Ethereum smart contracts, but instead selected by the user at the time of creating a transaction. This allows users to maintain the ability to choose between cheap transactions and getting their transaction included in the blockchain quickly. The amount of gas consumed, on the other hand, is a variable that can and should be optimized by smart contract developers. Ethereum's transaction fee system is built with the idea, that any use of computational, bandwidth or storage resources costs gas. Thus, making the contract less resource-intensive in any of these aspects will also reduce its gas consumption, therefore marking our objective.

4.3 Design and development

Trying to find inefficiencies in the smart contract's gas consumption, we approached each of the contract's five public functions individually. We first considered if there was a viable way of executing the function off-chain. If not, we examined if another method of improvement would be applicable. During this process, we identified and applied the following principles:

Avoid a design pattern where many new smart contracts need to be deployed, for instance, on a per-user basis. At a cost of 32 000 gas, contract creation is the most expensive EVM operation.

Keep the amount of transactions needed to interact with the smart contract low to diminish the impact of the transaction base fee of 21 000 gas. Design an interface with fewer functions that do more actions, rather than more functions that do fewer actions.

Optimize the smart contract's use of storage space. Whenever possible, use memory instead of persistent storage. Storing a word in persistent storage costs 20 000 gas, whereas storing a word in memory only costs 3 gas plus a memory expansion fee, whenever more memory is required. The memory expansion fee scales quadratically as more memory is needed, so memory should be used densely.

When the use of persistent storage is necessary, *consider if the stored data could be replaced with its cryptographic hash* on-chain, and the data itself could be stored off-chain.

Delete contracts and data stored in persistent storage that are not needed, in order to gain gas refunds.

Make use of off-chain transactions, using the blockchain only as an arbiter in case disputes happen.

During the analysis, it was identified that the `makeOffer` function can be removed from the smart contract entirely. Instead of announcing sales offers in the blockchain, offers can be cryptographically signed by their creator and sent to potential buyers in an off-

chain communications channel, e.g. a peer-to-peer network. If a buyer later decides to accept the offer, the buyer must then include the sales offer along with the seller's digital signature as a parameter in their `acceptOffer` function call. This way, unaccepted offers do not needlessly bloat the blockchain yet a buyer can prove the authenticity of the offer by its digital signature.

The expected reduction in gas consumption from implementing off-chain offers is approximately 21 000 gas per a successful trade. This is due to not having to execute the `makeOffer` function anymore, and not needing to pay its transaction base fee. In addition to this saving, off-chain offers have the effect that offers that are never accepted by a buyer also never create a blockchain transaction, rendering them entirely free. This can be expected to enhance the efficiency of the electricity market due to much more efficient price discovery and lower transaction costs.

The functions `acceptOffer`, `buyerReport` and `sellerReport` do not seem as straightforward to execute off-chain. An on-chain `acceptOffer` call is necessary to make sure that only one buyer can accept a given offer and to prove that the offer was accepted before its expiration. The smart meter report functions have a strict deadline before which they must be submitted. An on-chain function call is a simple way to prove that this deadline has been met.

The `withdraw` function needs to be executed on-chain in order for the funds to be transferred on-chain. It was identified, however, that the current design where a separate call needs to be made for each transfer of electricity is not optimal. If users were allowed to withdraw funds from multiple electricity transfer contracts using a single `withdraw` call, fewer transactions would be required and less gas would be spent on transaction base fees. With this modification, we estimate the saving per electricity transaction to be $21000 - \frac{21000}{n}$ gas, where 21 000 is the Ethereum base transaction cost, and n is the number of trades from which a user can withdraw funds using a single `withdraw` call.

4.4 Demonstration

During the design and implementation work, three artifacts were produced. Each artifact is a variant of the electricity market smart contract with some attempted improvements implemented. Artifact 1 implemented off-chain offers, as discussed earlier in Section 4.3. In Artifact 2, the `withdraw` function was modified so that it takes an array of trade IDs as argument and attempts to withdraw funds from all of the listed trades. Artifact 3 combines the changes implemented in Artifacts 1 and 2, with both off-chain offers and an improved `withdraw` function implemented.

A benchmark use case was executed on the artifacts, measuring the respective gas consumptions (see Tables 3–5). The same benchmark use case was also executed on the original electricity market smart contract, to be used as a reference (see Table 2). The benchmark use case was crafted so that it represents typical use of the smart contract where a number of trades are completed successfully: 1) as a seller, create n number of offers; 2) as a buyer, accept all created offers; 3) as the seller's smart meter, report all trades to have been successful; 4) as the buyer's smart meter, report all trades to have been successful; and 5) withdraw all deposits to the seller's address.

The variable n in step one translates to the number of electricity trades completed in the use case. The test case was executed with different values of n to see how different implementations perform when varying amounts of transactions are created.

We measured the combined gas consumption of all the transactions created in the execution of the use case. The gas consumption of the transactions was inquired from the TestRPC instance using the `eth_estimateGas` function of the Ethereum JSON RPC API before the sending of each transaction. This function makes a transaction and returns its gas consumption but does not add the transaction to the blockchain. In a TestRPC configuration like the one used, the `eth_estimateGas` call is made to a blockchain of exactly the same state as its corresponding actual transaction, so the returned gas consumption estimate is equal to the true consumption.

The measurements were run on an Ubuntu 16.04.3 LTS machine. The Solidity smart contracts were compiled using the Solidity compiler solc version 0.4.18. A local instance of TestRPC version 6.0.1 was used to simulate an Ethereum blockchain. TestRPC was configured to create a separate new block for each transaction. A block gas limit of 90 000 000 was configured.

4.5 Evaluation

The results collected from the first artifact show that the implementation of off-chain offers did reduce gas consumption of the smart contract in the selected use case. In measurements where a few trades were made, roughly a 5 percent decrease in gas consumption was achieved. When more trades were made, this percentage gradually decreased. That is, however, due to the smart contract becoming more populated with data and certain phases in its execution having to spend

Table 2. Reference measurements from the original electricity market smart contract.

Amount of trades (n)	*Gas consumed*
1	400 318
2	787 210
3	1 175 676
4	1 565 716
5	1 957 330
6	2 350 518
7	2 745 280
8	3 141 616
16	6 368 968
32	13 125 496
64	27 848 152
128	62 128 792

Table 3. Measurements from Artifact 1 that has off-chain offers implemented.

Amount of trades (n)	*Gas consumed*	*Difference to reference (%)*	*Difference to reference (gas)*
1	379 534	-5.19	-20784
2	745 578	-5.29	-20816
3	1 113 260	-5.31	-20805
4	1 482 516	-5.31	-20800
5	1 853 346	-5.31	-20797
6	2 225 750	-5.31	-20795
7	2 599 728	-5.30	-20793
8	2 975 280	-5.29	-20792
16	6 036 168	-5.23	-20800
32	12 459 832	-5.07	-20802
64	26 517 208	-4.78	-20796
128	59 466 520	-4.29	-20799

Table 4. Measurements from Artifact 2 that implements the renewed withdraw function.

Amount of trades (n)	*Gas consumed*	*Difference to reference (%)*	*Difference to reference (gas)*
1	402 564	+0.56	2246
2	762 007	-3.20	-12602
3	1 123 024	-4.48	-17551
4	1 485 615	-5.12	-20025
5	1 849 780	-5.49	-21510
6	2 215 519	-5.74	-22500
7	2 582 832	-5.92	-23207
8	2 951 719	-6.04	-23737
16	5 959 480	-6.43	-25593
32	12 276 826	-6.47	-26521
64	26 121 121	-6.20	-26985
128	58 645 051	-5.61	-27217

Table 5. Measurements from Artifact 3, off-chain offers and renewed withdraw function.

Amount of trades (n)	*Gas consumed*	*Difference to reference (%)*	*Difference to reference (gas)*
1	381 780	-4.63	-18538
2	720 397	-8.49	-33407
3	1 060 652	-9.78	-38341
4	1 402 481	-10.43	-40809
5	1 745 884	-10.80	-42289
6	2 090 861	-11.05	-43276
7	2 437 412	-11.21	-43981
8	2 785 537	-11.33	-44510
16	5 627 010	-11.65	-46372
32	11 611 844	-11.53	-47302
64	24 791 563	-10.98	-47759
128	55 985 573	-9.89	-47994

gas on iterating that data. The absolute gas consumption savings achieved from off-chain offers do not seem to be reliant on the number of trades made, ranging narrowly from 20 784 to 20 816 gas per trade. This roughly equals to the base transaction fee of 21 000 gas, which was the expected saving from not having to call the `makeOffer` function

Implementing the ability to withdraw funds from multiple trades using a single transaction also led to savings in gas consumption, at best by over 6 percent. Surprisingly, with all *n* values other than 1, Artifact 2 created larger savings than the initially estimated $21\,000 - \frac{21\,000}{n}$ gas per electricity transaction. With *n* values greater than 4, the savings of the artifact exceeded the base transaction fee of 21 000 gas, which was anticipated to be the maximum gas saving opportunity for the artifact. We hypothesize that the extra savings at least partly originate from Artifact 2 only calling the Solidity `send` function once, while the reference implementation calls it *n* times. As a result, slightly less EVM code needs to be executed.

Artifact 3 was a combination of the changes in Artifacts 1 and 2: off-chain offers and the improved `withdraw` function. It is noteworthy that the gas consumption savings in Artifact 3 were almost exactly equal to the sum of savings gained in Artifacts 1 and 2. In other words, there was virtually no overlap in combining the two improvements.

While we were able to reduce the gas consumption of the electricity market smart contract in this study, the reduction was a little over ten percent at best. Assuming that the market participants are using batteries to ensure their capability to make successful trades, a typical trade in the electricity market could be estimated to be in the same order of magnitude as the capacity of a large car battery. A 12-volt 100 Ah battery could theoretically output 1.2 kWh of electricity. Assuming a price of $0.1/kWh, the electricity transferred in a typical trade would be worth $0.12. In section 4.1, we showed that at the market prices over the past year or so, the total transaction cost of a trade in the original smart contract, deployed in the

public Ethereum blockchain, would have varied between $1.48 and $32.35. Even with the gas savings achieved, the transaction costs remain disproportionate, and in fact orders of magnitude too high compared to the value of a typical use of the application. This would suggest that the implemented optimizations are not adequate to make the application economically feasible on the public Ethereum chain, at least for transactions as small as suggested and at the examined price levels.

5. Discussion

While we were not able to improve the efficiency of the electricity market smart contract to the point of economic feasibility, this study demonstrates how blockchain-based smart contracts require a new kind of utility-centric focus on resource management in software development. Any Ethereum smart contract in any application should always seek to perform the absolute minimum set of tasks required from it. Whenever possible, decentralized applications should implement their own functional layers on top of the smart contract to keep the smart contract as simple and as low in resource consumption as possible.

The guidelines we produced for optimizing gas consumption of Ethereum smart contracts were successfully applied to pinpoint and fix inefficiencies in the electricity market smart contract. We estimate that the drafted guidelines are perfectly applicable for similar optimization tasks of any other Ethereum smart contract as well.

Although we used price ranges based on the price variation over the past year or so to estimate economic feasibility, it should be acknowledged that due to the chaotic nature of the system, future gas price dynamics are difficult to predict. While deemed unlikely, any radical drops in the real price of gas would require the feasibility findings of this study to be re-evaluated.

While having little impact on gas consumption in the benchmark use case, it was recognized that implementing off-chain offers could enable new types of price discovery mechanisms, potentially useful in other contexts. In a use case involving heavy use of selling offers that are never accepted, off-chain offers alone could reduce gas consumption significantly more than the 11 % measured.

It is also quite possible that other improvable inefficiencies exist in the smart contract which were simply not identified or pursued in this paper. For example, a large share of the application's gas consumption is due to the use of persistent storage. We anticipate that significant gas savings could be achieved by only storing the hash of a sales and purchase agreement instead of its full details in the blockchain. The actual data could be hosted elsewhere, *e.g.* the Interplanetary File System (IPFS).

As an alternative to deployment in the public Ethereum blockchain, the Plasma child blockchains proposed by Poon and Buterin [30], for example, may provide a viable platform for deployment in the future. A Plasma child chain could provide a similar execution environment connected to the Ethereum main chain, but with a lower demand for transactions, implying lower transaction costs. Another option would be to create a separate Ethereum blockchain instance entirely. While this would enable transactions at a mere fraction of the cost of the canonical Ethereum chain—or even completely free of transaction fees altogether—the lack of support for the ether cryptocurrency and for the security of the canonical chain could turn out to be problematic.

6. Conclusions

In this study, we explored ways to analyze and improve the feasibility of an experimental distributed blockchain market application designed for conducting microtransactions of electricity in a nanogrid environment [19,23]. By applying the design science research methodology by Peffers *et al.* [29], we managed to pinpoint inefficiencies in the design of the smart contract and to reduce its gas consumption by 11 %. From this process, we formulated a set of general guidelines suitable for optimizing the efficiency of any Ethereum-based smart contract.

While the improvement achieved in efficiency was not adequate for economic feasibility on the public Ethereum blockchain, we established that further improvements are likely to be possible with more radical reformations to the source code, redefined market mechanics, and the use of an alternative deployment environment.

Further research is encouraged on the recognized improvement opportunities where additional efficiency gains could be achieved. We also invite the exploration of other new ways to improve the efficiency of Ethereum-based smart contracts. Furthermore, the price dynamics of gas and ether, and their effects on application feasibility, would benefit from a more structured delineation.

7. References

[1] C.Y. Baldwin, C.J. Woodard, The Architecture of Platforms: A Unified View, in: A. Gawer (Ed.), Platforms , Mark. Innov., Edward Elgar Publishing, Cheltenham, UK, 2009: pp. 19–44.

[2] A.S. Bharadwaj, A Resource-Based Perspective on Information Technology Capability and Firm Performance: An Empirical Investigation, MIS Q. 24, 2000, pp. 169–196.
[3] H.K. Bhargava, S. Sundaresan, Computing as Utility: Managing Availability, Commitment, and Pricing Through Contingent Bid Auctions, J. Manag. Inf. Syst. 21, 2004, pp. 201–227.
[4] K.J. Boudreau, A. Hagiu, Platform Rules: Multi-Sided Platforms as Regulators, in: A. Gawer (Ed.), Platforms, Mark. Innov., Edward Elgar Publishing, Cheltenham, UK, 2008, pp. 1–29.
[5] B.V. Buterin, Ethereum White Paper - A Next Generation Smart Contract and Decentralized Application Platform, 2013.
[6] R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg, I. Brandic, Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility, Futur. Gener. Comput. Syst. 25, 2009, pp. 599–616.
[7] R. Buyya, D. Abramson, S. Venugopal, The Grid Economy, Proc. IEEE. 93, 2005, pp. 698–714.
[8] T. Bye, P.V. Hansen, How do Spot prices affect aggregate electricity demand?, Statistics Norway Discussion Papers No. 527, 2008.
[9] C. Catalini, J. Gans, Some Simple Economics of the Blockchain, MIT Sloan Research Paper 5191-16, 2016.
[10] L.W. Cong, Blockchain Disruption and Smart Contracts, 2018.
[11] P. Constantinides, O. Henfridsson, G.G. Parker, Introduction—Platforms and Infrastructures in the Digital Age, Inf. Syst. Res. Articles, 2018.
[12] S. Davidson, P. De Filippi, J. Potts, Economics of blockchain, 2016.
[13] P. De Filippi, B. Loveluck, The Invisible Politics of Bitcoin: Governance Crisis of a Decentralised Infractructure, Internet Pol. Rev. 5, 2016, pp. 1–28.
[14] Digiconomist, Bitcoin Energy Consumption Index, 2018.
[15] D. Easley, M. O'Hara, S. Basu, From mining to markets: The evolution of bitcoin transaction fees, 2017.
[16] European Environment Agency, Eurostat - Share of renewable energy in gross final energy consumption, 2017.
[17] P. Finn, C. Fitzpatrick, Demand side management of industrial electricity consumption?: Promoting the use of renewable energy through real-time pricing, Appl. Energy. 113, 2014, pp. 11–21.
[18] A.R. Hevner, S.T. March, J. Park, S. Ram, Design Science in Information Systems Research, MIS Q. 28, 2004, pp. 75–105.
[19] T. Hukkinen, J. Mattila, J. Ilomäki, T. Seppälä, A Blockchain Application in Energy, ETLA Reports No. 71, 2017.
[20] B. Ives, G.P. Learmonth, The information system as a competitive weapon, Commun. ACM. 27, 1984, pp. 1193–1201.
[21] J.A. Kroll, I.C. Davey, E.W. Felten, The Economics of Bitcoin Mining or, Bitcoin in the Presence of Adversaries, in: Proc. WEIS, 2013: pp. 1–21.
[22] K. Lauslahti, J. Mattila, T. Hukkinen, T. Seppälä, Expanding the Platform: Smart Contracts as Boundary Resources, in: Springer Transl. Syst. Sci. Ser., Collab. Value Co-Creation Platf. Econ., 2018.
[23] J. Mattila, T. Seppälä, C. Naucler, R. Stahl, M. Tikkanen, A. Bådenlid, J. Seppälä, Industrial Blockchain Platforms?: An Exercise in Use Case Development in the Energy Industry, ETLA Working Papers No 43., 2016.
[24] J. Mattila, T. Seppälä, Distributed Governance in Multi-Sided Platforms – A Conceptual Framework from Case: Bitcoin, in: Springer Transl. Syst. Sci. Ser., Collab. Value Co-Creation Platf. Econ., 2018.
[25] J. Mattila, The Blockchain Phenomenon. The Disruptive Potential of Distributed Consensus Architectures, 2016.
[26] S.C. Misra, A. Mondal, Identification of a company's suitability for the adoption of cloud computing and modelling its corresponding Return on Investment, Telecommun. Softw. Eng. Emerg. Methods, Model. Tools. 53, 2011, pp. 504–521.
[27] S. Murugesan, Harnessing Green IT: Principles and Practices, IT Prof. 10, 2008, pp. 24–33.
[28] S. Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System, 2008.
[29] K. Peffers, T. Tuunanen, M.A. Rothenberger, A Design Science Research Methodology for Information Systems Research, J. Manag. Inf. Syst. 24, 2007, pp. 45–77.
[30] J. Poon, V. Buterin, Plasma: Scalable Autonomous Smart Contracts, 2017.
[31] N. Szabo, Formalizing and Securing Relationships on Public Networks, First Monday, 1997.
[32] D. Tapscott, A. Tapscott, Blockchain revolution: How the technology behind bitcoin is changing money, business, and the world, Penguin, 2016.
[33] A. Tiwana, Platform Ecosystems: Aligning Architecture, Governance, and Strategy, Morgan Kaufmann, Waltham, MA, 2014.
[34] P. Weill, The Relationship Between Investment in Information Technology and Firm Performance: A Study of the Valve Manufacturing Sector, Inf. Syst. Res. 3, 1992, pp. 307–333.
[35] R. Weron, Modeling and Forecasting Electricity Loads and Prices - A Statistical Approach, Vol. 407, John Wiley & Sons Ltd, Chichester, England, 2007.
[36] G. Wood, Ethereum: A Secure Decentralised Generalised Transaction Ledger, 2013.
[37] D.G. Woodward, Life cycle costing—Theory, information acquisition and application, Int. J. Proj. Manag. 15, 1997, pp. 335–344.
[38] World Nuclear Association, Renewable Energy and Electricity, 2018. http://www.world-nuclear.org/information-library/energy-and-the-environment/renewable-energy-and-electricity.aspx (accessed June 13, 2018).
[39] Y. Yoo, O. Henfridsson, K. Lyytinen, The new organizing logic of digital innovation: An agenda for information systems research, Inf. Syst. Res. 21, 2010, pp. 724–735.