Norwegian University of Science and Technology

Department of Engineering Cybernetics

Use of blockchain technology for settlement in a finite energy market (microgrid)

By:
Marit Schei Tundal
marittu@stud.ntnu.no

Supervisor: **Geir Mathisen**

NTNU

March, 2018

Abstract

Sammendrag

Contents

Abstract i

Sammendrag ii

Table of Contents iv

1 Introduction 1

2 Background and Related Work 2
2.1 Microgrids 2
2.2 Blockchain 2
2.2.1 The Blockchain Basics 2
2.2.2 Private, Public and Consortium Blockchains 5
2.3 Merkel Tree 7
2.4 Consensus Models 7
2.4.1 Proof of Work 8
2.4.2 Proof of Stake 8
2.4.3 Proof of Elapsed Time 8
2.4.4 Byzantine Fault Tolerance 8
2.4.5 Summery of Consensus Models 9
2.5 Security and Weaknesses in Blockchain technology 9
2.6 Blockchain Applications 9
2.6.1 Bitcoin 9

2.6.2 Ethereum 9
2.6.3 Hyperledger 10
2.7 Related work 10
2.7.1 Brooklyn Microgrid 10
2.8 Other Use Cases 10

3 Specifications **11**

4 Implementation **14**
4.1 Network Communication 14

5 Testing **16**

6 Result **17**

7 Discussion **18**

Appendices **19**

Bibliography **20**

Chapter 1

Introduction

Provides a trustworthy foundation in many applications. Saw need for this after the housing bubble burst in 2009 and the Lehman Brothers fell. There is no longer the need for a trusted institution to handle transactions - due to the blockchain technology, trust is no longer need at all to be part of transactions of value.

Distributed - not sending copy of assets. Prevents double spending of assets like money, votes, electricity etc.

Chapter 2

Background and Related Work

2.1 Microgrids

2.2 Blockchain

A blockchain is a decentralized database, distributed over a network of nodes. Transactions in the network are stored in blocks which are linked together as a chain, creating the blockchain. It was first introduced by Satoshi Nakamoto with the Bitcoin application [1]. In the following sections of this chapter, some key features about blockchains, the technology behind blockchains and, their usage will be analyzed and discussed.

2.2.1 The Blockchain Basics

The blockchain is primarely used to transfer assets between users. In contrast to previous transaction methods, there is no need for trust between the participants of the transaction, nor the need of a trusted third-party institution, like a bank or government. The trust lies in the system as a whole, and the mathematical functions behind it, and not the individual participants [2]. Due to the decentralization of the network, there is no single-point-of-failure, as the database is duplicated in every node of the network. If a new user enters the network, or if they have been offline for a while, they can obtain the most recent version of the blockchain by querying the network multiple times, until they are convinced they have the correct version.

The below description of a blockchain is based on the Bitcoin blockchain. Many of these features hold true for most blockchains. However, there are many variations of blockchain implementations, due to public/private configurations, consensus models etc. These differences will be described later in the chapter.

When a node transfers an asset to another node, it creates a transaction. The transaction contains: one or more inputs; one or more outputs; and a digital signature. An input is a previously received and unspent output, containing the amount of an asset. Multiple inputs can be added together to create a larger transaction. The inputs are used completely, so if the amount does not add up to the wanted output, multiple outputs, specifying the amount in each, can be used so that the change is sent back to the owner. This is because transactions must reference an output, and each output can only be referenced once. Thus, the exact ownership of every asset can be pinpointed to a user in the network, based on the previous transactions. In order for a transaction to occur, it must be proved that the asset has not been previously spent by the current owner. Thus, there is no way to double spend assets.

Transactions are validated through digital signatures, proving the authenticity of a messaged, by using public and private keys. Each node can obtain a random private key. Based on this, a public key is generated from an Elliptic Curve Digital Signature Algorithm (ECDSA) The recipient node sends the hash of its public key to the node initiating the transaction. The public key, together with the message is hashed together, and used to create a digital signature, thus providing a unique signature for every transaction. The transaction can then be claimed by the node in possession of the corresponding private key. Thus, in order to use a previously obtained output as an input, it must be verified by the node's private key. The use of digital signatures ensures that transactions can traverse the network without being altered, as even the smallest change will cause a signature to no longer be valid. Since every transaction is visible to every node on the network, anonymity is important as it becomes (nearly) impossible to find the true identity of the owner. A new key pair can be generated for every transaction to increase anonymity.

A miner stores multiple pending and unconfirmed transaction together to create a block. The block consists of the transactions, as well as the block header. The block header usually contains: a reference to the previous block in the chain, namely the hash of the previous block; a timestamp for when the new block is created; the merkel root of the merkel tree containing all the transactions in this block; and a nonce and a target used to calculate the hash of the new block. The merkel tree and how the hash of a block is found, will be discussed later in the chapter.

When the hash of the new block is found, the block is broadcasted to the rest of the network. Every node on the network validates the block, before they start working on the next block. Validation is an easy task, once the hash is found. If several miners find the solution to the next block at the exact same time, a fork is created. A fork means that there exsists two (or more) different versions of the blockchain in the network. Due to network latencies, nodes in the network will receive different broadcasted messages at differend times. Therefore, nodes might have different versions of the blockchain. The miners start working on the next block based on the one they received first, while storing the other version in case that fork becomes longer [1]. The fork is resolved when one side of the fork becomes longer than the other(s), which will most likely happen when the next block is created. The side with the longest chain always wins because it is backed by the most work. All the nodes in the network immediately look to the longest chain as the valid version of the blockchain. The transactions in the discarded block go back into the pool of the pending transactions, waiting to be mined in a block. Once a transaction is part of the blockchain, it is irreversible. Due to the hash functions, even the slightest alteration in a transaction will be detected, and the alterations will be deemed as non-valid. Thus, a transaction cannot be changed or reverted once it is part of the blockchain. There is however some need for caution. As previously mentioned, transactions in discarded blocks go back to being unconfirmed. If the other side of the fork contains a transaction attempting to double spend the asset, the initial transaction might no longer be valid.

Say, for instance node A sends two bitcoins to node B, while also sending the same two bitcoins to itself, only one of these transactions can be valid. Once a transaction is stored in the blockchain, the two bitcoins are referenced as a new unspent output, and the other transaction is no longer valid. This is intentional behavior in the blockchain, as it solves the problem of double spending which is usually what a trusted third-party, e.g. a bank, is needed for. The problem occurs when there is a fork. If node B receives the bitcoins as a payment for some goods, and ships the goods once the transaction is part of the blockchain, while node A is simultaneously working on another fork in the blockchain containing the transaction sending the two bitcoins to itself, node B will lose the payment for the goods if the fork node A is working on becomes longer. It is therefore a good policy to wait until the block containing the transactions has multiple successors before asserting the transaction final.

2.2.2 Private, Public and Consortium Blockchains

Blockchains are usually divided into three different categories, depending on the way they are governed. [3] Public blockchains, like Bitcoin, is a blockchain where anyone is free to participate, and is completely decentralized. Private blockchains are more centralized with e.g. a single organization controlling the blockchain. A consortium blockchain is a hybrid between the two, where participants must be authorized, but does not have a single, central authority controlling it. Another, and wider classification is permissioned and permissionless blockchains. These are categorized depending on whether the blockchain is open to anyone or if some form of permission is required for access to the blockchain. Public blockchains are categorized as permissionless, while private and consortium blockchains fall under the category of permissioned. The three main categories will be further discussed below.

Public Blockchains

A public blockchain is completely unregulated. It is open to anyone that wishes to read it; transactions are completely transparent. It is also open to anyone wishing to make transactions or participate in the consensus process for validating blocks - making it fully decentralized [3]. The only requirement is a valid transaction.

Public blockchains rely heavily on cryptoeconomics for security in an otherwise low-trust environment. Cryptoeconomics is described as "*a combination of economic incentives and cryptographic verification using mechanisms such as proof of work or proof of stake, following a general principle that the degree to which someone can have an influence in the consensus process is proportional to the quantity of economic resources that they can bring to bear*" [3].

The most important innovation from public blockchains, is the problem of who transmits first and the issue of double spending Additional benefits of the openness of a blockchain includes the resistance to censorship. A single node or a group of nodes, provided they do not control a big enough (depending on the consensus model) portion of the network, cannot keep transactions out of the blockchain.

Public blockchains can grow quite large, and posses huge amounts of processing power, combined. This computational capacity can in turn be used for ...

As mentioned, public blockchains rely on cryptoeconomics which in turn requires computationally heavy consensus models. Proof of Work (PoW) is the most common protocol used to reach consensus in a public blockchain. PoW has its drawbacks, including vast amounts of computing power required, especially in the Bitcoin blockchain

Private Blockchains

Trusted parties Lost identification credentials - private keys Reverse verified transactions in event of e.g. system error, fraudulent activity, mistake in recipient address. Reduced access lowers risk of outsider attack Benefits of blockchain require some for of mistrust between users, otherwise a shared database might be a better solution Developers might change system without the users consensus Key issuing from e.g. government to be allowed participation - loses part of the aspect of being independent from third parties. Ability to revert transactions, e.g. property ownership in legal dispute Trust comes from being trusted to participate in verification No single point of failure, as not dependent on one single computer Cheaper due to no transaction fees Little or no chance of 51% attacks

Consortium Blockchains

Summary

In conclusion, the type of blockchain most beneficial for each applications will vary from industry to industry and from application to application. Public blockchains are most beneficial for individual users, as there is no need to trust anyone. Private and consortium blockchains are more beneficial to organizations or companies, as they lower cost and increase speed of transactions.

2.3 Merkel Tree

A merkel tree is a (binary) tree where all the leaf nodes are paired and hashed until a single root hash remains, which is the merkel root. Each transaction is appended to the merkel tree as a leaf node. The merkel root is stored in the header of a block in the blockchain.

SPACE SAVING Also used by private blockchains to ensure only valid transactions are added to the blockchain

Common in peer-to-peer systems including blockchains. Authenticate receiving data as part of file (etc.), verify transaction (that has happened previously). No need to store entire history - saves space and time. Merkel data created out of transaction data in each block. Enables auditing transactions in logarithmic time opposed to linear time.

A Merkle consistency proof lets you verify that any two versions of a log are consistent: that is, the later version includes everything in the earlier version, in the same order, and all new entries come after the entries in the older version. If you can prove that a log is consistent it means that no certificates have been back-dated and inserted into the log, no certificates have been modified in the log, and the log has never been branched or forked.

A Merkle audit proof lets you verify that a specific certificate has been included in a log. This is a critical verification task because the Certificate Transparency model demands that all TLS clients reject any certificates that do not show up in a certificate log.

2.4 Consensus Models

Consensus models are important in distributed systems, in order for ... correct version of the shared ledger. Reward for mining Safety, liveliness, fault-tolerance Two types of fault-tolerance - fail-stop and Byzantine faults

2.4.1 Proof of Work

Computationally expensive Requires very much energy Variable called nonce, which is adjusted in order to find a hash matching requirements of another variable called target Consensus used in Bitcoin untrustworthy peers wanting to modify past blocks have to work harder than peers wanting to add new blocks. Target value determines difficulty, the lower the target value the more attempts are needed on average to find ha hash beneath the target threshold. Mining to generate more crypto currency not needed or desirable in this use case Brute force Good scalability for participating nodes and completely decentralized Slow, energy waste, latency

Hardware centralization - ASCI

2.4.2 Proof of Stake

Tries to solve the energy problem in proof of work Rich get richer More assets in network - more likely to be chosen as validator for mining - reward Centralized

Casper Tendermint no coin creation (mining) exists in proof of stake. Instead, all the coins exist from day one, and validators (also called stakeholders, because they hold a stake in the system) are paid strictly in transaction fees.

2.4.3 Proof of Elapsed Time

2.4.4 Byzantine Fault Tolerance

Byzantine Fault Tolerance is a form of ”leader-follower” protocol. Private blockchains do not utilize proof of work as a consensus model. Rather, a Byzantine Fault Tolerance protocol can be used to ensure valid transactions. Requires more than one node in network

Ripple and Stellar use open-ended (in regards to node participation) Byzantine fault tolerance as a consensus model Ripple requires each node to have a list of trusted nodes, each nodes list must have a 40% overlap with other nodes in the network. Consensus in multiple rounds

Support high transaction rates Low scalability. Over 20 nodes causes huge overhead in messages

2.4.5 Summery of Consensus Models

This is perhaps the most challenging aspect in order to create a blockchain. Different types of blockchains require different types of consensus models. At this point, there does not exist a single best consensus model. Limitations and restrictions are present in regards to power consumption, scaling, security and transaction speed. These factors will be further discussed in the following section.

2.5 Security and Weaknesses in Blockchain technology

2.6 Blockchain Applications

2.6.1 Bitcoin

Black market because of openness when everyone can read and write transactions Centralized the mining to countries with cheap electricity - vulnerable to changes in policy on electricity subsidies

On average 10 minutes per block to avoid double spending, adjust difficulty to keep this pace. Fully verification takes about 1-2 hours due to possibility of forks - possibility of "double spending" Wallets One block contains 1 MB worth of transaction data

We define an electronic coin as a chain of digital signatures.

Speed of transfer TCP/IP email analogy Transaction chain - history of ownership and blockchain - transaction ordering

2.6.2 Ethereum

Blocks can contain code which becomes a smart contract New blocks every 15 seconds Build apps on ethereum or access and interact with smart contracts Currently uses PoW but plan in changing to PoS Turing-complete programming language for smart contracts

Three types of applications on Ethereum platform; financial, semi-financial and non-financial.

Smart Contracts

Problem that they might not be hack safe

2.6.3 Hyperledger

Hyperleger is an open-source platform for blockchains hosted by the Linux Foundation. It consists of many project, two of which will be further evaluated here. Fabric and Sawtooth. Consortium with identities verified and registered Linux foundation Permissioned blockchain Smart contracts Two consensus models - PBFT and SIEVE Not a cryptocurrency

2.7 Related work

There already exist some examples of blockchains being used in the energy market.

[4] : Current trading of renewable energy without blockchain – pay ahead. NRGcoins directly correlating to energy transferred to grid – pay for use Calculations for buying and selling energy. First to propose this kind of system?

https://solarcoin.org/en/frequently-asked-questions Interbit BTL Innogy

2.7.1 Brooklyn Microgrid

Virtual/physical grid Based on ethereum, but not scalable so making its own

2.8 Other Use Cases

Land ownership Sharing economy Transferring money (abroad) Intellectual property rights - music, kodak etc. Crowd funding via smart contracts

Chapter 3

Specifications

This chapter contains the specifications for the blockchain system to be implemented.

The blockchain will be permissioned, with a central authority, e.g. the project owner, validating which nodes are allowed to join the network. The blockchain will not be completely private, but rather a consortium where all the nodes allowed access to the network are participating in the validation of new blocks. To avoid malicious nodes, or other unwanted participants in the network, the project owner also handles the distribution of private and public keys used for signing transactions. By having a central authority control key distribution, it is possible to keep a record of user identities corresponding to the keys.

(An alternative approach for accepting new nodes in to the network, is to make it part of the consensus model. For instance, say the network is started by an initial group of nodes. If a new node is to be introduced into the network, M of N nodes must validate the node before it can enter the network. This could work in a system such as a microgrid, where participants are physically close, possibly acquaintances in real life, and the system is finite. Key distribution would then be handled differently e.g. by a key generating function. This potentially provides higher user privacy, which might not be ideal in such a system.)

Each new block will contain all pending transactions. The transactions will be stored in a merkel tree, to save space in the block header. The merkel tree is also used to verify transactions, e.g. by ensuring that no tokens are used for double spending. Each block/transaction contains a timestamp to help prevent double spending of tokens or double selling of electricity(?)

The consensus algorithm is some form of Byzantine Fault Tolerance (TBA) where M of N nodes must agree before a block is validated. As previously mentioned, all nodes in the network participate in the consensus process of validating new blocks. This is to prevent centralization of the blockchain. Since the blockchain is not public, there is no reason to rely on cryptoeconomics for incentives to keep the blockchain correct. The users have already been approved, so there is little to no risk of an attack. BFT requires less computing power than PoW or PoS and can also be implemented to perform much faster transactions.

The current electricity price is visible to the consumers/prosumers at all times, and is updated every minute. Payment for consumed electricity will be handled by consumers making deposits into an account. Deposits will be made with actual currency (e.g. USD or NOK) and be converted into tokens (not to be confused with cryptocurrency), where the token value directly correlates to value of the currency with minute resolution. The tokens can at any point be withdrawn from the account. Tokens are transferred from consumer to producer when electricity is bought by consumer.

A consumer signs a contract with a producer/prosumer for a given amount of time, e.g. a month. The consumer can at any time decide how much electricity they want to buy, or if the want to buy anything at all. Fluctuations in price will likely influence this decision. Once a consumer signs a contract from one producer/prosumer, it cannot sign a new contract with someone else. However, it can opt out of using the electricity for a period by e.g. using a battery, or simply using less energy for a period of time, if it is expensive. This might help regulate pricing if it becomes too high due to high demands.

To ensure precise transactions of tokens/electricity, smart contract functionality is used. A consumer puts tokens into a contract with limitations on how much electricity they want to buy and at which prize. If current electricity price matches the limitations, the computing node (virtual machine) automatically "forwards" the right amount of energy to the right node, and the tokens in return. The contract locks the tokens until electricity has been consumed. In case of a prosumers/producers failure to provide electricity, tokens are transferred back to consumer. This functionality prevents double spending of tokens and double selling of electricity.

The contracts will be stored in the blockchain, and automatically executed when the right conditions are met. A token contains a state indicating whether it is locked in a smart contract or not. An important reason for using smart contract functionality is that there are two assets being transferred in this system: tokens and electricity. In a cryptocurrency blockchain with only one asset, the sender of the asset initiates the transaction. As a result of the bi-directional transfer of assets, the smart contract is used to ensure that neither side can withdraw from the transaction when it is in progress.

TODO: messaging in system, user identification, ...

The implementation of this system is focused around the software; the hardware is out of scope. However, some form of computer is needed to monitor the microgrid, as well as running a virtual machine for validation of blocks, creation and execution of smart contracts, and running and monitoring the blockchain. For the purpose of this project, these features will be simulated.

Chapter 4

Implementation

Class for blocks

Class for blockchain

Class for transactions

hashlib.sha256 for hashes

Socket vs HTTP

TCP/UDP - UDP would be preferred in local LAN, could make WLAN for blockchain in finite area, but TCP for more general implementation

4.1 Network Communication

Peer to Peer network implemented with Python framework Twisted. Asynchronous. Event based. The initial node starts a server and connects to itself. Other nodes connect to a server already running on a given port. Hello message is sent from node trying to connect and receives and hello ack if connection is successful. Connecting node receives a list of peers from the node it connected to and tries to connect to them in the same manner.

Reactor Deferreds, callback and errback chain

Lock for transactions?

No protection against brute force attacks on network

Store all peers since finite size of network?

Send hello Ack hello Start ping Respond with pong If 3 pings unresponded - remove peer Also drop peer in peer list if connection is dropped and notified Connecting node receives other peers in network, connects to peers and repeat above process until no unconnected peers or max peers connected Starts a new client for every peer it connects to Peers receive connections on server if they are being connected to Initial node starts blockchain Connecting nodes compare local head block to peer If not the same, receive blockchain if other peer has connecting peers head block in chain Validate blockchain and query other peers for head block and receive blockchain if its not the same When blockchains are the same start creating blocks When new block created, it is proposed to the other peers, they validate block and accept it and sets it to be local head block

Chapter 5

Testing

Chapter 6

Result

Chapter 7

Discussion

Necessary to store all transactions?

Appendices

Bibliography

[1] Satoshi Nakamoto. *Bitcoin: A peer-to-peer electronic cash system.* 2009. URL: http://bitcoin.org/bitcoin.pdf.

[2] Michael Nofer et al. "Blockchain". In: *Business & Information Systems Engineering* 59.3 (June 2017), pp. 183–187. ISSN: 1867-0202. DOI: 10.1007/s12599-017-0467-3.

[3] Vitalik Buterin. *On Public and Private Blockchains.* Blog. Aug. 2015. URL: https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains/.

[4] M. Mihaylov et al. "NRGcoin: Virtual currency for trading of renewable energy in smart grids". In: *11th International Conference on the European Energy Market (EEM14).* 2014, pp. 1–6. DOI: 10.1109/EEM.2014.6861213.