|  |  |
| --- | --- |
| **eXchains**  CS4160 Blockchain Engineering (2017/18 Q2) | Project Report by:  Joseph Verburg 4018575  Michal Loin 4587324  Jetse Brouwer 4615964  Niels Hokke 4610148  Evgenia Domnenkova 4625633 |

# Introduction to Imbalance Market

The energy market consists of multiple markets with the main goal of balancing production and consumption. Each market works on a different timescale with different terms. The 4 most important markets are the Long-term market, day-ahead market, intra-day market and the unbalance market.

The imbalance market is the most volatile market. This market is led by the Transmission system operator (TSO, TenneT in the Netherlands) and serves the sole purpose of maintaining a stable frequency of the grid. When a imbalance arises TenneT has multiple reserves at his disposal. These reserves consist of parties who have the ability to regulate their consumption or production up or down relatively fast (depending on which reserve group they're in). The primary reserves will be used to solve immediate imbalance. The secondary reserve is the market that is used if the grid hasn't stabilized after the 15 minutes of using the primary reserves. At last there is a tertiary reserve, also called the emergency pool, for long term highpower regulation power.

# Problem Statement

With consumers of electricity now also becoming producers at a higher rate, more imbalance is created on the net. To maintain security of supply the imbalance should be kept within the limits of the grid. Currently the transmission system operator (TSO: Tennet in Netherlands) is responsible for maintaining this balance. A problem arises when the production becomes more and more decentralized, and the balancing is still the duty of a centralized body, it becomes harder to respond on the highly fluctuating consumption and production of the different actors.

The aim of this project is to design and prototype a decentralized market and a so called 'smart grid' where participants of the smart grid can dynamically match their supply and demand to minimize imbalances.

# General Approach to Balancing

The system should be able to be deployed in co-operation with the current electrical grid as it unrealistic to expect major changes in the grid topology in the foreseeable future.  The grid is currently laid out in a manner where several households (1 or more streets) are clustered together sharing a physical cable. These clustered cables join at small brickhouse-substations spread out through the city. These substations are clustered together on a 10KV network, where one cluster roughly serves a city. Between nearby cities a 50KV net grid is clustering those together, from there they are clustered on a 110 -150 KV grid and again at a 380KV grid (which also connects with neighboring countries). In the current situation, the grid is balanced by the TSO, who buys capacity to up- or downregulate the consumption or production at companies spread out over several levels of the grid, and does so when required.

For balancing the grid matching consumption and production at the highest level is not enough. Even when nationwide balance is achieved overload can happen due to unreasonably large imbalances on lower levels. Therefor we believe that balancing should, in the first place, happen within same cluster (e.g. a single street) before propagating it up to a higher level (e.g. district or city). When looking at an example of at the lowest, the spikey behavior of a single household is smoothened out by combining the households per cluster, and the smart grid aims to spread out the load equally over time. The remaining imbalance is then propagated to a higher level where bigger players (which more capacity but lower flexibility) can partake in the balancing game. For every step to a higher level the peaks become smoother and more predictable, which is very desirable for the big powerplants that will have to supply the energy in moments of energy shortage.

# Balancing Using Blockchain

To enable the actual balancing between the participants of the smart grid (which will be called clients from now on) a platform must be offered to gain knowledge of the current state, the expected state and to perform reach agreements on energy exchanges. For this project we will explore the possibilities of using a blockchain for providing such a platform.

As discussed earlier, the focus should first lie on balancing within a single cluster, therefor there is no need for connecting every client to every other client. This creates the possibility of creating a blockchain per cluster instead of having to design a blockchain for all households in the system, which reduces the amount of transactions per second per blockchain significantly.

The knowledge per cluster will then be aggregated and propagated as a single entry up to a higher level blockchain, where bigger parties can join the energy balancing game. The aggregation implies that individual information is lost when propagating to a higher level. However, this does not have to be a problem as the individuals in the cluster already did their best to solve the imbalance the best they could and the main task is now to balance out the remaining imbalance.

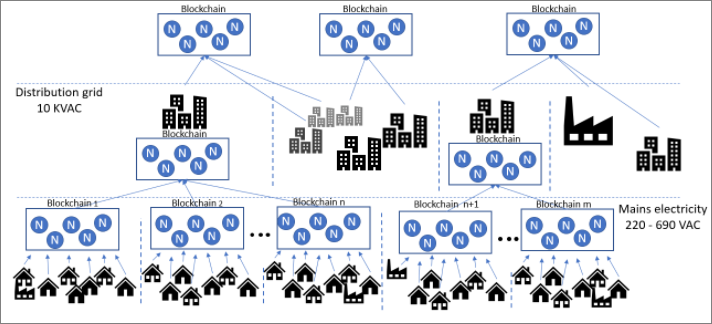


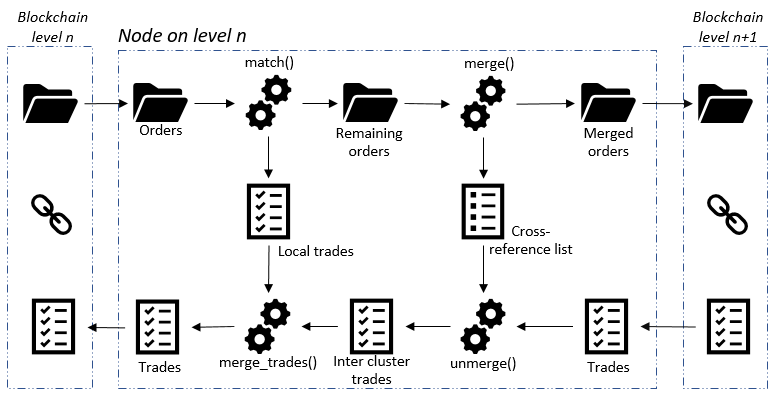
Figure 1: A possible data structure with a blockchain per cluster.

By following the physical lay-out for clustering the nodes we end up with a structure that could be visualized as a tree structure where the dataflows from node to node are only through their parent. A figure depicting such a tree is given in Figure 1. Here the circles containing the letter 'N' are the nodes that host the blockchain, the black symbols depict the clients (which can be households, industries and companies specialized in balancing).

# Balancing Algorithm

The main idea behind the balancing algorithm is that users (smart meters, aka clients, and intelligent household agents behind them) submit to the blockchain their expected consumption, production and theflexibility they can afford/offer for the coming balancing timeslot. The flexibility is the indication from the client to what extend they are ready to increase/decrease their production/consumption and for how much. The more flexibility there is on offer, the easier it is to balance the grid. The prices given in the flexibility data are not fixed prices but limits similar to exchange markets. ‘Ask’ price is the maximum price the client is willing to pay for that amount of energy, where the ‘Bid’ price is the minimum the client wants to sell that amount of their energy for.

As the model is currently expected to be compatible with the existing grid, no user is expected to be denied the requested energy within the predicted consumption. Therefore, every user, before being able to join the balancing, is required to have a contract with a back-up supplier of the grid. The back-up supplier is introduced into the model for the worst-case scenarios when the whole grid cannot balance itself and the the back-up supplier is there to cover for the imbalance. The idea is that the back-up will be used as little as possible, but needs to be there in case of emergency.



When it comes to matching supply(‘bid’) and demand (‘ask’), the buyer almost always pays less than their maximum, and the seller almost always makes more than their minimum. To encourage clients to place their bids at their personal extremes, the algorithm first matches highest buyer with lowest seller (just as the stock market) as those orders generate overlap and

therefore enable more matching, and,consequently, help to balance the grid as a whole. The matching first happens at the cluster level. Then the consolidated imbalance is sent up to the next cluster of balancing and the same matching is executed at the next level. This is repeated until imbalance it zero at all clusters, or the root is reached.

# Considerations

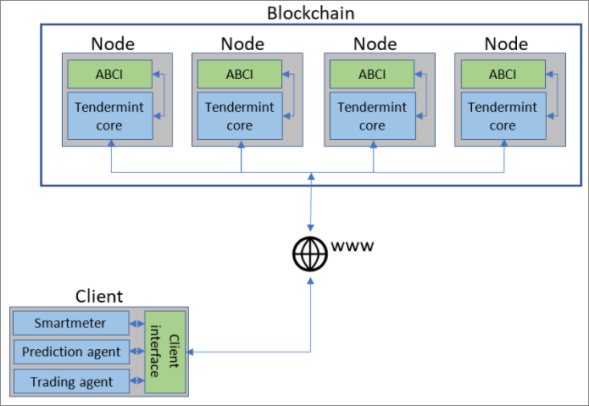
As part of the project, the team familiarised themselves with the following material and information:

* [Princeton blockchain lectures](https://piazza.com/princeton/spring2015/btctech/resources)
* [Blockchain market research](https://github.com/Tribler/tribler/issues/2559)
* [Blocklab paper](http://blocklab.nl/media/uploads/2017/09/Beta-Paper-Blockchain-x-Energy-online_opmaak.pdf?x54716)
* [Lecture notes Distributed Algorithms](https://github.com/NielsHokke/DEM/blob/master/LectureNotes-IN4150-20170109.pdf)
* [Master thesis of SjorsHijgenaar](https://repository.tudelft.nl/islandora/object/uuid%3Ac4a8d2be-6c4e-4bfb-8a37-8a65bd7fe50b?collection=education)
* [Satoshi paper](https://bitcoin.org/bitcoin.pdf)
* [Tendermint and PoS problems](https://blog.cosmos.network/consensus-compare-casper-vs-tendermint-6df154ad56ae)
* [Tendermint](https://tendermint.readthedocs.io/en/master/)
* [Graph illustrating the consensus mechanism of Tenderminder](https://github.com/devcorn/hackatom/blob/master/tminfo.pdf)
* [Tendermint thesis paper](https://atrium.lib.uoguelph.ca/xmlui/bitstream/handle/10214/9769/Buchman_Ethan_201606_MAsc.pdf)
* [Blockchain as a service system based on Tendermint](https://github.com/monax/monax)
* [Cosmos.Network](https://cosmos.network/faq)
* [Etherium](http://www.ethdocs.org/en/latest/)
* [Casper](https://blockonomi.com/ethereum-casper/)
* [Hyperledger-fabric](https://hyperledger-fabric.readthedocs.io/en/release/)
* [Ripple](https://ripple.com/)
* [Corda](https://www.corda.net/)
* [Iota](https://dev.iota.org/)
* [Waves](https://github.com/wavesplatform/Waves/wiki/Waves-Node-REST-API)
* [Bitshare](http://docs.bitshares.org/)

# Technical Prototype Overview

**Layers:**

* Client Layer: Smart metertransactions simulated by a python script with help of a static csv file for data simulation
* Blockchain Layer: Tendermint Core and Custom ABCI written in python + Validator VMs in Docker
* Visualisation Layer: Thin client written in Node.js to visualise blockchain process



**Why use blockchain for grid balancing:**

* Balancing was selected as the use case as it fits the parameters of general blockchain use case:
  + Several parties without any one authority
  + Lack of trust among those parties
* Effective balancing of the grid is also one of the major challenges of the distributed energy market in a market where any consumer can also be a supplier

**Why Tendermint as the blockchain platform:**

* Ability to run permissioned blockchain for:
  + Ability to maintain similarity with the current physical infrastructure
  + Ability to hold consensus servers accountable
  + Ability to trace clients/users in the system
* Ability to run blockchain with BFT-based PoS design:
  + Ability to reach the performance level required by the target field of implementation with limited number of nodes (e.g. to serve up to thousands of clients with only up to 20 nodes)
  + In addition, BFT blockchains are per definition permissioned, requiring the sharing of identities amongst the servers, in support of the requirement related to permissioned blockchain
* Ability to be flexible over the blockchain application functionality so that any block/contract logic (ABCI application) can be defined depending on the field of implementation

Some technical challenges tough:

* Lack of technical knowledge and learnings from other developers available online due to Tendemint being a young technology
* Lack of Tendermint supporting material for Python developments
* Migration to new versions of Tendermint during the project
* Lack of hooks into Tendermint Core (e.g. not being able to identify the current validator) made it more difficult to implement the balancing rounds

# Conclusions and Next Steps

**Project Conclusions:**

* The working prototype demonstrated that the core ideas, that the team developed about the approach, are technically feasible
* Tendermint’s idea about the BFT Consensus Core and Custom ABCI is a good concept, however Tendermint as a platform needs to mature to make it easier to develop and deploy on it

**Possible Next Steps:**

* Enhance ABCI from prototype to a more deployable solution, including future enhancements of Tendermint that may make implementation simpler
* Enhance balancing algorithm logic
* Implement smart contract features
* Implement intelligent agents running on the smart meters
* Test the solution on physical infrastructure (e.g. real smarts meters, real households, real validators from the balancing community, etc)
* Pilot the process/technology with real world parties (e.g. households, energy companies, Tennet, etc)

# Tendermint core concepts

To facilitate the blockchain, Tendermint is used. Tendermint consists out of chief technical components; the consensus layer and the ‘Application BlockChain Interface’ (ABCI). The first part is responsible for receiving, verifying and reaching consensus on transactions, where the second one is designed by the developers of the project that has to run ‘top’ of the blockchain. Every time someone wants to place information in the blockchain it sends a transaction to the core layer of one of the nodes. This node will make a call to verifyTX() in the ABCI layer to verify the validity of the transaction (based the state of the blockchain and other factors). When the receiving node considers it a valid transaction he will then gossip it to other nodes who, on their turn, will verify it as well. The Tendermint core then elects a node who will propose a new block, after multiple rounds of commits and votes the block will be accepted or refused.

As the balancing takes place in rounds, and we don’t want to rely on absolutely times system clocks, relative timing is used. As soon as a new balancing round is announced the nodes start a timer for 10 minutes. In these 10 minutes clients can report their usages to the blockchain. After a node’s timer expires it posts a ‘Begin Balancing’ transaction to the blockchain, if a node receives this message and agrees the timing is right, it will accept it. If enough nodes agree on this, a block will be committed with the balancing message in it, informing both the nodes and the clients that no more reports will be accepted and the nodes have 5 minutes to balance the trades.

To ensure that no failures are made during the balancing end up in the blockchain all of the nodes run the balancing algorithm, but only one is allowed to send transactions. All the other nodes verify that the transactions are in line with what they expect. Since the balancing is designed in a fully deterministic way a certain unique input will always result in the exact same unique outcome, and therefore mistakes can be easily detected. The node making the transaction is picked based on a random number generator seeded with the hash of the previous block, to ensure unpredictable but fair and deterministic chances of being the next balancer.

After the 5 minutes have passed the node will submit a ‘new round’ transaction that, if the other nodes agree it is time, will trigger a new round to be started by having the ‘new round’ transactions committed to a block.