enerDAG – Towards a DLT-based Local Energy Trading Platform

Christoph Groß[1] Mark Schwed[1] Stefan Mueller[1] Oliver Bringmann[1]

***Abstract*—Due to decreasing costs and less by-effects of renewables energy the energy production is getting more and more decentralized and therefore leads to more bottom-up loads on the grid. To reduce the stress on the grid local energy grids with smart energy trading are a possible solution. This paper describes the goals and concept of our flexible resilient local energy trading platform called enerDAG and how it can easily be extended through smart contracts. At the end enerDAG gets critically analyzed if it solves the critical requirements for a scalable smart grid platform. Nodes using this platform can get a financial advantage through cheaper local energy which might also lead to even more investment in own regenerative energy systems.**

***Index Terms*—Smart Energy, Smart Grids, Smart Contracts, Tangle, Distributed Ledger Technologies, Embedded Systems, Efficiency**

I. Introduction

The energy market is currently facing a change from big centralized and mostly fossil energy producers to more distributed and decentralized prosumers. This is accelerated through forced reduction of carbon emissions and the nuclear phase-out and hence opening up new opportunities in energy trading [1], [2]. Due to the nature of electricity needing to be consumed when it is produced to ensure the network stability, locally traded electricity can bring advantages for local participants and grid operators even though the reality is currently still looking different.

When installing a photovoltaic system on a roof there are only two options: consuming as much energy as possible or selling it to the local energy provider. The current remuneration for the sale of electricity is significantly below the purchase price in most cases. Therefore, a valid question is why it is not possible to sell overproduced electricity directly to neighbors at a price between the purchase and sales price. This results in several positive effects: Not only is the energy consumed where it is produced and relieves the high voltage transport networks. But this also creates an incentive to invest in renewable energies. Finally, energy suppliers can counteract blackouts by providing targeted incentives in the form of price changes.

With more and more photovoltaic and biogas plants added to the power grid, the grid becomes even more decentralized making it even harder for the energy providers to handle. This is especially apparent when at sunny times all photovoltaic system produce simultaneously at high rates; when the sun now is covered by some clouds, the energy flow just stops abruptly.

The idea of the neighborhood contract is to be able to trade energy with neighbors. The neighborhood is defined as the lowest level of connections in the energy network (at the 230 V level behind the local transformer station) because this allows for the lowest costs of energy transportation. So on average we are talking about 100 to 150 households that will be able to communicate with each other [3]. The challenge here is the low energy consumption of individual households. As a result, one transaction covering 15 minutes has an average value of less than 10 cents.

In this paper we set out to solve the problem of making a distributed, scalable and energy efficient local smart grids for the neighborhood. First we describe the current status and why classic blockchain technology like Bitcoin or Ethereum isn't able to solve this and then how our solution of Smart Contracts added to a special variant of a tangle can solve this.

II. Related Work

Since the goal of our project is to create a distributed local energy market we need a distributed database where we can have thousands of writers and not all of them are trusted. As a consequence we use distributed ledger technology (DLT) similar to the Bitcoin blockchain [4] as our distributed database. DLTs allow for a high throughput of data and a consensus about which data is correct.

There exist multiple papers discussing for which specific use case blockchains or more generally speaking DLTs are useful [5]–[7]. Our decision is based on the desire for a decentralized data structure that enforces every participant to keep up to date and follow the rules. For our system we use the Tangle as the underlying data structure.

The first bigger test field for local energy grids using blockchain technology was the *Brooklyn Microgrid* [8]. Since then multiple new projects have been testing blockchain-like DLTs in small local test fields like the Swiss project *Quartierstrom* [9]–[12] and in the "*East Harbour Prosumer Community*" inside the European *PARENT* project [13], [14]. But classical blockchain solutions have the problem of needing lots of energy if they are using Proof of Work (PoW), needing to store lots of data for at least a big recent part of the blockchain and everyone needs to have the completely same state of the whole chain. A better scalability separates the Tangle as a data structure from the classic blockchain.

[1]University of Tübingen, Faculty of Science, Department of Computer Science, Embedded Systems Group

This work was partially supported by the state of Baden-Württemberg (project number BWSGD 18011).

Fig. 1. A tangle where every new node references two previous nodes

A. The Tangle Data Structure

The Tangle [15], [16] is a directed acyclic graph (DAG) that consists of transactions and their references. Each node can add transactions to the Tangle by following simple rules to create a transaction e.g. by referencing at least two older transactions called tips and then publishing it to its neighbors. All other nodes will then receive the transaction through the gossip protocol and will accept and include the received transaction in their view of the Tangle structure. The Tangle is a chronologically ordered data structure, as new transactions can only reference older ones and do so for a specific time. So over time the Tangle graph will grow into one direction as you can see in figure II-A.

This means, the Tangle is like a blockchain without blocks and without a chain and therefore overcomes a lot of the problems like fees, mining, and limited transactions per second from other DLTs. As the Tangle has a transaction flow with time, this allows to delete old transactions to free disk space as they will not be referenced in the far future and allows a chronicle order to check which transaction came first for the verifiability of transactions.

B. Smart Contracts in Distributed Ledgers

A smart contract is a computerized transaction protocol that executes the terms of a contract without the need of having a third party involved. The origin of this name and the idea behind it was first developed by Szabo [17] and further refined throughout the years [18]–[20]. This idea was among others further improved by Buterin in [21]. The goal of smart contracts are deterministic digitally executable contracts without the overhead of written contracts like the interpretation of a contract by a human but with at least the same security. Additionally, it provides

- a reduction of transaction costs and involved third parties,
- improved speed and immutability, and
- self-enforcing contract clauses and easy contract checks.

But the name "smart contracts" can be misleading. They are not smart in a sense that they adapt intelligently to environmental changes. The contracts are smart in a sense that they react to defined changes of their world, for example executing after a specific time or saving the amount of money different actors have sent to the smart contract in a state. They can save data and act on that data with predefined rules. Smart contracts also are no legal contracts. They should rather be thought of as a programs that execute at certain conditions.

Using a distributed ledger like a blockchain or the Tangle as the data structure of smart contracts leads to some special attributes:

Immutability When a smart contract is saved on a distributed ledger its source code is frozen to a specific address in the ledger and can never be changed again. New versions of a smart contract need to be saved on a different address. This way it is ensured that no one can alter a smart contract's source code.

Open Source Knowing the address of a smart contract one can read its code and in the best case simulate the outcome of a specific action. There is no room for law or contract interpretation by a human.

Decentralization Similar to the idea of a distributed ledger, smart contracts are decentralized and get executed decentralized. Therefore no trust in a single party with the result or the execution time of a contract is necessary. Instead the decentralized majority depending on the consensus algorithm of the specific DLT decides on the outcome of a smart contract.

C. Problems of Smart Contracts on a Tokenless Tangle

There are three requirements for smart contracts on distributed ledgers:

1) Byzantine-fault tolerant consensus algorithms that allow for security through decentralization. This guarantees that no single node tampers with the network by providing a wrong smart contract result,
2) programming languages with some level of Turing completeness built into the blockchain to allow creation of custom logic as smart contracts, and
3) a fee structure as an incentive for running other people's contracts on a owned node.

The consensus algorithm of a *Proof of Work* (PoW) blockchain is done by miners and a lot of hashing power to put a lot of transactions into a block and find the correct nonce so that the hash of the block including the nonce starts with x – set by the difficulty of the network – zeros. When a block is mined, all other nodes can verify that this nonce is correct and will add the block to their own blockchain. The issuer of a new block will then be rewarded the fees of all transactions included within his block and some additional (for example) Bitcoin or Ethereum.

With Solidity, a Turing-complete programming language is included in the Ethereum blockchain as the underlying idea of Ethereum was to allow smart contracts. Bitcoin uses a scripting language to publish smart contracts to allow the creation of custom logic.

On blockchains smart contracts get executed because the result of an execution is a transaction which can be included in the next mined block. This alone provides an incentive of fees. Furthermore, the execution of each line of contract code is rewared with a fee; this way it is lucrative for miners to execute contracts and include their results in the next block.

The problem with the Tangle data structure and smart contract is obvious: the Tangle in its IOTA origin and in enerDAG is feeless, there are no miners that try to achieve the next block to receive a reward, and there is also no programming language built into the Tangle. In addition, there are more problems of smart contracts in a Tangle environment:

No built-in programming language To solve this problem it is necessary to understand that within the idea of enerDAG it is not required that participants can create their own smart contract logic. They should not have to write their own contracts themselves as it is error-prone and highly technical. Households should rely on templates or even execution-ready contracts provided by a third party like a neighborhood maintainer.

The decision is to use private smart contracts and save the code off-chain or off-Tangle. This means that each node has their own copy of a smart contract they want to participate in. As the contracts are written in simple language they are still open source and readable by anyone. Nodes can verify they have the same contract by comparing a simple hash of the contract. Each node can only change their own contracts which will possibly lead to a different result when executing the contract but only on this specific node.

When sharing the result with the network, nodes that altered their contracts will just be a "normal" malicious node trying to influence the contract's result and will get caught by the Byzantine-fault tolerant algorithms.

No fees Running anyone's code costs money through the used energy. To still trigger the interest in participating one needs to have a financial benefit in doing so. There is no advantage in forwarding a trade or an offer to others if the exchange of energy is build on a first-come-first-serve policy.

Additionally after the release of an offer, there would be a flood of bidding transactions and all actors besides the winner would still need to send more energy purchase requests to elsewhere. Another aspect would be the problem that nobody besides the two partners of that energy exchange would want to invest enough energy to recalculate the results of the smart contract which could lead to malicious actors and an easy misuse of the consensus algorithm.

To overcome these problems there we introduce the neighborhood market contract in the form of partially centralized markets.

III. Method

The Tangle idea we use for the system is inherited from IOTA's Tangle concept [15] but we make significant changes to the IOTA protocol as for example our system has no tokens (IOTAs) and we use a binary instead of a ternary system. For the first concept study, we are trading in the past as we cannot predict our production and usage of energy easily.

A centralized market can be set up as a smart contract that is executed in fixed intervals, for example every 5 or 15 minutes. Households that want to participate in the energy market in a desired time frame send their energy balances and prices for buying and selling to the smart contract. At the execution time a market algorithm clears the market and the result of the contract are the happening energy trades. A node benefits from the smart contract if the following conditions are fulfilled:

- It participates with an offer in the time frame,
- the smart contract result is one where the nodes overproduction / need is traded, and
- the result containing the trade gets accepted by the overall network as the majority result.

To participate in the market a node needs to send a transaction with their energy overproduction or needs and prices to the central smart contract.

By definition the marketplace allows each node to send only one transaction per time frame. This reduces the amount of transactions to a minimum. The market algorithm of the smart contract (that is explained in III-H) then matches buy and sell offers. All Nodes have a financial incentive to calculate the smart contract result if they take part in the smart contract within the time frame. As this will be normal operation, nodes will participate in the calculation and securing of the result. The market is planned to span over a neighborhood defined by a energy transformer station (because of transport costs), containing about up to 150 participants within one neighborhood smart contract.

For this reason it is acceptable if a node does not take part in the smart contract caused by connection issues, exchanges, etc. because a lot of participants will execute the result calculations and a consensus mechanism will find the overall result. With every node that takes part in a round of energy trading via the smart contract there is an additional player that has an incentive to behave correctly, calculate the correct result of the market algorithm, and share it with his neighbors.

Due to enerDAG being a tokenless private decentralized ledger system there are special tasks that must be carried out reliably like validation and billing of energy flow. Therefore energy or infrastructure providers can run a node within each neighborhood that is called the neighborhood maintainer node.

A. Contract Composition

Smart contracts in the enerDAG system are stored on the nodes that participate in the contract. Like any other node in the network a contract is identified by its address. For additional security in the field of contract immutability the contracts address is its own SHA-256 hash. To interact with a specific contract, transactions can be sent to the contract's address as `receiverAddress` in a transaction. The contract itself is a program that is executed in a given interval.

A contract has two pre-defined functions: The `receiveTransaction(t)` function is called when a transaction with the contracts address is set as `receiverAddress`. As parameter the function gets the transaction itself, allowing the contract to process the transaction. The `executeContract()` function is called when the contract is executed at the specific times. The return

value is the contract's result value that then gets sent to the node itself for further processing and majority voting.

B. Contract Results and Majority Voting

When a node calculates the result of a smart contract that is not "None", it will try to verify this result with the network to reach a consensus with the network. This could mean that the node itself calculated the result wrong (maybe it missed some transactions within the contract time frame because of down time, loss of connection, etc.) and now wants to get the correct result from other nodes. It could also mean that it has the correct result and another node (maliciously intended or not) does not have the correct result and the network then tries to convince the other node to adapt the correct result.

The network has a majority requirement ($\frac{2}{3}$ by default) that determines how many neighbor nodes of those that are online at the time need to send the same result to convince the node that this result is correct. Every time a node receives a contract result from another nodes or itself, it checks if the contract is present on the system before proceeding. It then hashes the provided result and selects the previous result hashes from the database using the contract address and contract date. The resulting hashes are saved associated with their sender in the database. The majority voting takes place between arrival of the contract result and the database insertion.

Majority voting works as follows: If the first received contract result is from another node, nothing is done. If the first contract result is from the own node, it will be gossiped to all neighbor nodes. If the received contract result does not change the majority votes nothing is done. If the received contract result does change the majority votes the node changes its opinion or vote, saves the majority result as the contract result and sends its updated view on the result to its neighbors. If the received vote is from the own contract result calculation and there is no majority yet on any result hash, the own calculated contract result is saved as the contract result and sent to the neighbor nodes.

This process of calculations, sharing of opinion, and re-thinking the own opinion will eventually lead to a common contract result within the network given a good network neighbor structure, enough honest nodes, and some fast rounds of contract result sharing.

C. Gossip Protocol

The transaction protocol is some kind of verified gossip protocol. A node in the network creates a transaction and sends it to its neighbors. The neighbors receive that transaction but do not know who created the transaction. They check the transaction for formal verification (see 2f in section III-H) and if everything is correct, they save it to their own database and forward it to their neighbors excluding the one they got the transaction from. If a node receives an already known transaction, nothing happens. This results is a snowball scheme where a transaction is forwarded through the network of neighbors as fast as possible. It is the neighborhood maintainer node's task to keep the network in a state where the gossip protocol works and no two split neighborhoods or bottleneck nodes are formed.

D. Sync State

When a node receives a transaction that refers to tips of the Tangle which are not known, it saves the unknown transaction in a separate list for a defined amount of seconds. If within this time frame it receives the transaction from the majority of online neighbors, it is assumed that the node is not up to date with the Tangle and needs to be synced. It will then request the missing tip from its neighbors. Upon receiving this transaction from a neighbor, the node will then insert it into the database, check if it knows the referred tips, and if not, also request those.

Once the sync state is entered, a requested transaction needs to be provided by only one neighbor to insert the response into the database. This is fast and secure as the transactions are requested and no contract functions are executed as part of receiving requested transactions.

E. Neighborhood Market Contract

To buy and sell units of energy between the participants there is a need for a marketplace where all offers and demands are listed and cleared at the end of a time frame. This is realized with a smart contract which connects all participants from the neighborhood including producers, prosumers, and consumers. Within a time frame everyone posts their energy balances and buy and sell prices to the smart contract. During execution of the contract, an algorithm define the trades happening within the time frame and sends the result back to the Tangle. This market deals with the fixed amounts of produced and consumed energy of the last time frame, which eliminates the need to make any assumptions on the future energy balances. The design goals of the neighborhood market are:

Anonymity & Pseudonymity Nodes should be able to stay anonymous to prevent the creation of energy profile by another household. One exception is the neighborhood maintainer node which must be able to identify them for controlling purpose.

Security Only members of the defined neighborhood can participate in the market or even read the transactions. The neighborhood maintainer node should be able to allow nodes for trading as well as stop nodes from participation in the market.

Correctness Each household should be able to post their energy balance once within a time frame and set their buy and sell price as they wish. The market algorithm should be one that incentives the participation in the neighborhood market and correct behavior.

Fairness No node should be able to gain an advantage by behaving in a special way like waiting for all bids and sending last.

F. Bid and Clearing Prices

Following the Quartierstrom project [9]–[11], the idea of a market clearing algorithm based on a double auction mechanism was adopted. Each participant sends the following information to the market:

energyBalance Positive, if in the given time frame more energy was produced than consumed by the household

maxBuyPrice A maximum price in $\mathrm{ct/kWh}$ that the node is willing to pay for energy

minSellPrice A minimum price in $\mathrm{ct/kWh}$ that the node is willing to sell its energy for

After receiving all `energyBalances` and prices, the market is then cleared. For that, the algorithm searches the positive energy balance with the lowest `minSellPrice` and the negative energy balance with the highest `maxBuyPrice`. The amount of energy traded is either until all is sold from the seller or until the needs of the buyer are satisfied. The price of the trade is set as the mean of `maxBuyPrice` and `minSellPrice`. This leads to both buyer and seller having the same advantage from the trade and encourages all households to set a low sell price and a high buy price.

G. Authenticity and Correctness of Bids

To ensure a fair and smooth process, trading is divided into multiple phases, which is explained in detail in the next subsection III-H. In the first phase, consumption and prices are encrypted by the participants and transmitted to the market after a random waiting period. To authenticate this information, it is signed with a hash of a unique `validationSeed` and the current `contractTime`. This hash can be compared to a list of valid hashes which is distributed by the maintainer node M regularly. This also allows to block malicious nodes by sending a block list, excluding them from future lists and investigating their wrong doings through the neighborhood node. This offers a spam protection because the market accepts hashes only once and also prevents malicious nodes from placing multiple offers to gain a undue advantage:

1) Once at startup neighborhood maintainer node M generates `validationSeeds` $S_1, \ldots, S_n$ and sends S_i to each node N_i with n being the amount of nodes in this neighborhood.
2) The maintainer node then regularly (e.g. daily) publishes a list containing the sub lists of the o included time slots $L_{T_1 \ldots T_o}$ which is decryptable for each participant of this neighborhood. Each sublist L_{T_y} is constructed the following way with H being the SHA256 hash function:

$$L_{T_y} = \{H(H(S_1 + T_y)), \ldots, H(H(S_n + T_y, \ldots))\}$$

   The order of $S_1, \ldots, S_n$ is randomized to reduce the retracability from a doubly hashed value to a specific node
3) Now each node calculates the `validationKey` $= H(S_1 + T_y)$ for each time slot and appends it to its bidding. Other nodes now check later if the once more hashed `validationKey` exists in the list L_{T_y} of this time slot.

H. Neighborhood Market Algorithm

The neighborhood market algorithm fulfills the above mentioned goals in section III-E. The bidding part smart contract SC_B and the receiving and execution of the neighborhood market smart contract $SC_{R\&E}$ are both done by all participants. So all of them will compute their result by collecting encrypted bids and decryption keys before running the market algorithm, sharing the results, and doing majority voting.

Each node needs to have the following information for its neighborhood: the smart contract addresses it uses, the symmetric key, and its own `valdidationSeed` for this neighborhood. Since everyone can read all transactions on the Tangle we are using a key to encrypt all messages within a neighborhood. Only the nodes of this neighborhood and the neighborhood maintainer node know it.

Figure 2 shows that the second and third thirds of a the time frame are saved for the neighborhood market contract and how they look separated in phases. The first period is reserved for the chaining of the last time frame results (as described in section III-I) and possible later use in for example an apartment house contract.

1) **Phase 1** The events inside Phase 1 happen in the blue parts of Fig. 2. Each node has a random timer so that the messages are evenly distributed and spikes are reduced.
   a) Every node checks if the result of a previous market execution hash has already been posted to the Tangle and matches it to its own `previousResult` based on the majority voting and calculations. If the result is accepted the previous trades are seen as final, else it shares its own result.
   b) Following this the new bid is created with time frame, **energyBalance**, **maxBuyPrice**, **minSellPrice**, and the `validationKey`. It is then encrypted with a newly generated symmetric key.
   c) The message is then encrypted with the neighborhood symmetric key, put into the normal transaction structure, and then sent to the Tangle using the neighborhood market contract's address as the `receiverAddress`.
   d) Upon receiving encrypted bids, nodes decrypt the outer layer using the neighborhood symmetric key and then conduct some check like if the arrival time is still within Phase 1.
2) **Phase 2** In the second phase no more bids are accepted by the market. The nodes have to send the encryption key for their information sent in the first phase. The actions of Phase 2 happen during the orange time frame of Fig. 2.
   e) The symmetric key generated in Phase 1 is now added to a message, encrypted with the neighborhood key, and then sent to the Tangle from the same `senderAddress` as the encrypted bid before.
   f) Upon receiving a decryption key, nodes decrypt the outer layer and perform time checks. Then they check if the arrival time is within the bounds of Phase 2 and if they received an `encryptedBid` in Phase 1

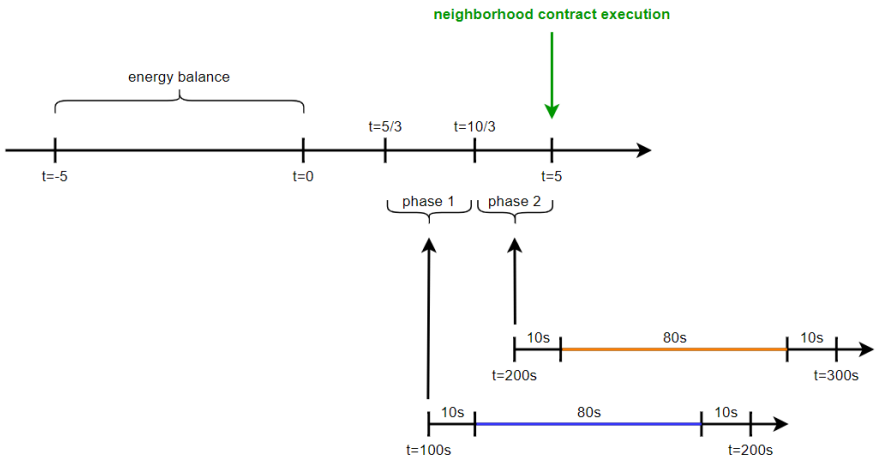

Fig. 2. Phases inside a 5 minutes trading period

with the same `senderAddress` as the decryption key. If they find a match, they decrypt it and then run several checks: (1) Is the bid's time frame matching the execution time frame of the contract (so it is no old bid), (2) is the `hash(validationKey)` used also in the `hashedValidationKeys` list provided by the neighborhood maintainer node, and (3) has the `validationKey` never been used before? If any check fails, the bid is dropped.

3) **Contract Execution**

g) For neighborhood contract execution the algorithm loops through all decrypted bids looking if there is at least one positive energy balance for selling and a negative energy balance for buying. As long as there is, the trading loop is run.

h) The best buyers – with maximum buy price – and the best sellers – with minimal sell price – are searched within all bids. Those can be multiple participants using the same **maxBuyPrice** or **minSellPrice**. These are the participants of the first round of trades.

i) To be "fair", all traded values will be split equally between same price sellers and same price buyers. First it is checked if in this round there is more energy to sell or more demand:

- If more energy could be sold than bought, the total energy amount to buy is calculated and divided on all best sellers of that round. If a seller reaches its energy balance with such a trade, the remaining volume is again divided on all other best sellers.
- If more energy could be bought than sold, the total energy amount to sell is calculated and divided on all best buyers of that round. If a buyer reaches its energy balance with such a trade, the remaining volume is again divided on all other best buyers.

j) These steps are repeated as long as there is still energy left to sell **and** to buy.

I. Chaining the Results in the Tangle

In the third phase the result of the second phase, which is composed of `decryptedBids`, `previousResultHash`, `previousResultTransactionHash`, and the trades, is chained into the Tangle. This works by sending it into majority voting as described in section III-B and takes place after phase 2 until the phase 1 of the next trading period starts.

It could also occur that a node sends the wrong `previousResultHash` maliciously or because its majority voting did not work well. In that case, there may be multiple previous results in the Tangle for a specific time frame but over the next executions of the neighborhood contract and majority voting there will be a result that is accepted by all nodes referencing a previous result transaction and so on. This is like "the longest chain wins" when comparing to blockchain architecture, where we have a separate chain for each neighborhood result.

J. Tasks of a Maintainer Node

This node has the following tasks:

- Adding new nodes to the neighborhood and changing the neighbor structure. The restructuring assigns every node randomly selected neighbors which minimizes the effect that malicious node clusters can try to influence honest nodes while new nodes are perfectly embedded into the network. The limit set here is five neighbors to allow for nice majority voting while not flooding the network with messages between them
- Publishing of validation keys for the neighborhood market
- Collect and validate Data on energy consumption of each household

- Generating bills

IV. Evaluation

enerDAG and its concepts allow local energy trading inside neighborhood environments. To test the neighborhood market prototype it is co-simulated with a Kerber Vorstadtnetz model included in the pandapower package [22]. This power grid contains a representative number of houses with and without photovoltaic. Both simulations are orchestrated by the mosaik co-simulation framework [23] to synchronize power production / consumption with the transactions on the tangle. It can easily be extended by just adding producers, prosumers and consumers. This neighborhood is designed to feature all of the ideas that the enerDAG software has:

Anonymity & Pseudonymity Through the neighborhood encryption it is ensured that only participants of the specific neighborhood and their maintainer node can read the bids. Additionally we have introduced the concept of switching addresses for each transaction, which minimizes the risk of being able to link households to transactions for normal participants. Bigger households with very specific energy prosumption characteristics will still be identifiable.

Decentralization As far as possible the energy trading is decentralized. All participants validate the trading events and even a temporary outage of the maintainer node and a few other nodes doesn't hinder the trading. The maintainer node is needed to permit entry to a neighborhood and to generate bills in this tokenless environment.

Reliability & Availability Through the decentralization a high availability is granted. Multiple parties can be unavailable or be newly joining without hindering the functionality of the trading platform. Even the maintainer node can be unavailable for a certain time only a wide-area outage of the energy and communication network can lead to outages of the system. Only the sync mode described in III-D can lead to a problem in very small neighborhoods when multiple new nodes join at the same time and therefore a new transaction with unknown `approvedTips` isn't getting forwarded by a majority of the neighbors since they also don't know these references.

Correctness The correctness of the trading platform is secured through the design decision of using tangles as data storage and through the permissioned energy trading rights through a central maintainer. This leads to valid trading transaction in defined user groups, which get validated through the consensus mechanism and stored immutably in the tangle.

Security & Risk analysis Since only permissioned members can participate in each neighborhood the risk of of a malicious neighbor is reduced. Additionally the maintainer node can check for correct bids and compare it to the results of the metering point operator. If a malicious behavior is detected the bad actor can be taken in recourse and can be stopped from further trading. For getting economic advantages by changing the market algorithm over a majority of over two thirds of the nodes would needed. Additionally no tokens or similar things are exchanged which also reduces the attack surface. Still there needs to be further security analysis to be done.

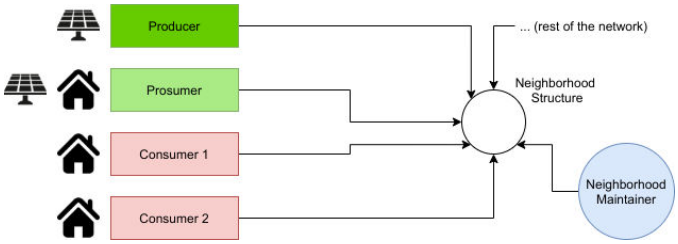

Fig. 3. Example of a small neighborhood

Fairness Through the specific design of two phases for each trading event no node can get an advantage through specific behavior.

Energy- and Costefficency Since expensive calculations like the PoW in Bitcoin or Ethereum or the IOTA Tangle tip selection algorithm are avoided or improved enerDAG should work better than current State of the Art but there is no concrete data to rely on for comparison. For storage enerDAG supports the option of regular snapshots but could still be more aggressive in removing irrelevant old transactions while still keeping the chain of market results. A problem is the size of the `validationKeys` list. This list would be approximate 900kB for 100 participants neighborhood with 5 minute trading periods and would need to be distributed to each participant daily.

Scalability enerDAG is based on the tangle which in itself already offers great throughput and scalability. Additionally there exists the options to separate bigger local areas in different parallel tangle flows so the small nodes only get reduced traffic.

V. Conclusion and Future Work

As we showed in IV we have developed a flexible and extendable local energy platform and have fulfilled most of the goals and requirements for a useful smart grid and big scale deployment.

The composition of the energy price with taxes, grid charges, current financial and costs for accounting is still an open question.

In the future we will add a apartment house contract to make it simple for house owners to sell their produced energy first inside the house to tenants before it goes to the neighborhood without any big setup overhead for the owner.

Another focus is on bringing this implementation down to low power platforms by adding hardware acceleration low power communication channels to maximize the benefit of the system.

References

[1] B. Mika and A. Goudz, *Blockchain-Technologie in der Energiewirtschaft: Blockchain als Treiber der Energiewende*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2020. [Online]. Available: http://link.springer.com/10.1007/978-3-662-60568-4

[2] O. Abrishambaf, F. Lezama, P. Faria, and Z. Vale, "Towards transactive energy systems: An analysis on current trends," *Energy Strategy Reviews*, vol. 26, p. 100418, Nov. 2019. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2211467X19301105

[3] G. Kerber, "Aufnahmefhigkeit von Niederspannungsverteilnetzen fr die Einspeisung aus Photovoltaikkleinanlagen," Ph.D. dissertation, Technische Universitt Mnchen, 2011.

[4] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," p. 9, 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[5] K. Wst and A. Gervais, "Do you need a Blockchain?" p. 7.

[6] D. Yaga, P. Mell, N. Roby, and K. Scarfone, "Blockchain technology overview," National Institute of Standards and Technology, Gaithersburg, MD, Tech. Rep. NIST IR 8202, Oct. 2018. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/ir/2018/NIST.IR.8202.pdf

[7] T. Koens and E. Poll, "What Blockchain Alternative Do You Need?" in *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, ser. Lecture Notes in Computer Science, J. Garcia-Alfaro, J. Herrera-Joancomart, G. Livraga, and R. Rios, Eds. Cham: Springer International Publishing, 2018, pp. 113–129.

[8] "Brooklyn Microgrid | Community Powered Energy." [Online]. Available: https://www.brooklyn.energy

[9] L. Ableitner, A. Meeuw, S. Schopfer, V. Tiefenbeck, F. Wortmann, and A. Wrner, "Quartierstrom – Implementation of a real world prosumer centric local energy market in Walenstadt, Switzerland," *arXiv:1905.07242 [cs]*, May 2019, arXiv: 1905.07242. [Online]. Available: http://arxiv.org/abs/1905.07242

[10] A. Brenzikofer, A. Meuw, S. Schopfer, A. Wrner, and C. Drr, "QUARTIERSTROM: A DECENTRALIZED LOCAL P2P ENERGY MARKET PILOT ON A SELF-GOVERNED BLOCKCHAIN," p. 5, 2019.

[11] A. Wrner, A. Meeuw, L. Ableitner, F. Wortmann, S. Schopfer, and V. Tiefenbeck, "Trading Solar Energy within the Neighborhood: Field Implementation of a Blockchain-Based Electricity Market," p. 12, 2019.

[12] A. Brenzikofer and N. Melchior, "Privacy-Preserving P2P Energy Market on the Blockchain," *arXiv:1905.07940 [cs]*, May 2019, arXiv: 1905.07940. [Online]. Available: http://arxiv.org/abs/1905.07940

[13] G. van Leeuwen, T. AlSkaif, M. Gibescu, and W. van Sark, "An integrated blockchain-based energy management platform with bilateral trading for microgrid communities," *Applied Energy*, vol. 263, p. 114613, Apr. 2020. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0306261920301252

[14] "PARENT Project | Helping communities reduce their energy consumption," library Catalog: www.parent-project.eu. [Online]. Available: https://www.parent-project.eu/

[15] Popov, Serguei, "The Tangle," 2018.

[16] S. Popov, O. Saa, and P. Finardi, "Equilibria in the Tangle," *Computers & Industrial Engineering*, vol. 136, pp. 160–172, Oct. 2019, arXiv: 1712.05385. [Online]. Available: http://arxiv.org/abs/1712.05385

[17] N. Szabo, "Smart Contracts," 1994. [Online]. Available: http://www.fon.hum.uva.nl/rob/Courses/~InformationInSpeech/CDROM~/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html

[18] ——, "Smart Contracts: Building Blocks for Digital Markets," 1996. [Online]. Available: http://www.fon.hum.uva.nl/rob/Courses/~InformationInSpeech/CDROM~/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart_contracts_2.html

[19] ——, "The Idea of Smart Contracts," 1997. [Online]. Available: http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM~/Literature/LOTwinterschool2006/szabo.best.vwh.net/idea.html

[20] ——, "A Formal Language for Analyzing Contracts," 2002. [Online]. Available: http://www.fon.hum.uva.nl/rob/Courses/~InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/contractlanguage.html

[21] V. Buterin, "A next-generation smart contract and decentralized application platform," *white paper*, p. 36, 2014.

[22] L. Thurner, A. Scheidler, F. Schäfer, J. Menke, J. Dollichon, F. Meier, S. Meinecke, and M. Braun, "pandapower an open-source python tool for convenient modeling, analysis, and optimization of electric power systems," *IEEE Transactions on Power Systems*, vol. 33, no. 6, pp. 6510–6521, Nov 2018.

[23] "mosaik." [Online]. Available: https://mosaik.offis.de