future internet

MDPI

Article

A Blockchain-Based Real-Time Power Balancing Service for Trustless Renewable Energy Grids

Andrea Calvagna, Giovanni Marotta, Giuseppe Pappalardo and Emiliano Tramontana *

Dipartimento di Matematica e Informatica, University of Catania, 95125 Catania, Italy; andreamario.calvagna@unict.it (A.C.); giovanni.marotta@phd.unict.it (G.M.); pappalardo@dmi.unict.it (G.P.)
* Correspondence: tramontana@dmi.unict.it; Tel.: +39-095-7383008

Abstract: We face a decentralized renewable energy production scenario, where a large number of small energy producers, i.e., prosumers, contribute to a common distributor entity, who resells energy directly to end-users. A major challenge for the distributor is to ensure power stability, constantly balancing produced vs consumed energy flows. In this context, being able to provide quick restore actions in response to unpredictable unbalancing events is a must, as fluctuations are the norm for renewable energy sources. To this aim, the high scalability and diversity of sources are crucial requirements for the said balancing to be actually manageable. In this study, we explored the challenges and benefits of adopting a blockchain-based software architecture as a scalable, trustless interaction platform between prosumers' smart energy meters and the distributor. Our developed prototype accomplishes the energy load balancing service via smart contracts deployed in a real blockchain network with an increasing number of simulated prosumers. We show that the blockchain-based application managed to react in a timely manner to energy unbalances for up to a few hundred prosumers.

Keywords: blockchain; smart contract; green energy; power modulation

Citation: Calvagna, A.; Marotta, G.; Pappalardo, G.; Tramontana, E. A Blockchain-Based Real-Time Power Balancing Service for Trustless Renewable Energy Grids. *Future Internet* **2024**, *16*, 149. https://doi.org/10.3390/fi16050149

Academic Editor: Yuansong Qiao

Received: 16 March 2024
Revised: 10 April 2024
Accepted: 21 April 2024
Published: 26 April 2024

1. Introduction

Energy production is moving away from the paradigm whereby a central system controls and regulates the flow of energy from production to consumers towards a decentralized scenario whereby large communities of remotely distributed, small energy producers/consumers (i.e., prosumers) contribute to renewable energy production by means of plants on their premises, such as roof-installed solar panels, small wind turbines, home batteries, and electric vehicles, according to the crowd energy paradigm [1].

Such modern energy networks make use of low-cost digital smart meters to actually measure and control the energy provisioning data as a separate digital flow, independent of the energy transport network itself, thus allowing new entities to play a role in a large-scale market of energy management and provisioning services: the smart grid. One major role is that of the aggregator, an operator that aggregates the resources of a managed prosumer pool into a virtual power plant and facilitates the sale of excess energy to face fluctuations in the grid power provision [2]. Therefore, a data-management system is needed to record agreements and production data, as well as to regulate the flow of energy among the contractual parties. However, there is no standard nor pervasive digital infrastructure to allow operators to communicate with small-scale assets and their digital controllers, despite many initiatives having taken place to fill this gap, such as Equigy (https://equigy.com/, accessed 14 February 2024) and Energy Web (https://www.energyweb.org/, accessed 14 February 2024).

In this scenario, one of the key drivers is to provide crowd energy players with a trusted data-exchange platform to actively enter the smart grid market and effectively participate in grid flexibility. Blockchain-based systems perfectly fit these needs [3], allowing a

fully decentralized system to work and trust between parties to hold, without requiring any previous agreement on a common trust authority to guarantee that no one will tamper with the recorded data and computations. Blockchain-based solutions have, thus, been widely applied in the literature for the design of the next generation of distributed applications in a variety of different application domains, all sharing the common requirement of enabling Internet users or IoT devices to safely trade information and resources across untrusted environments and networks [4–7]. On the other hand, while blockchains and smart contracts are a powerful tool to automatically enforce the rules governing the interaction amongst mutually untrusted parties, their expressive power in terms of coordination ability, while rapidly evolving, is still limited [8]. As a consequence, it becomes quite difficult to confidently prefigure at design time the actual validity, or to assess the efficiency and scalability, of any complex blockchain distributed application without actually implementing and testing it.

In principle, a blockchain-based distributed ledger can support the implementation of all the aggregator management tasks, such as the negotiation and signing of contractual agreements, the energy profiling of prosumers, the handling of billing and payments, and the issuance of power modulation commands to the controlled energy sources. However, a legitimate question that has not yet been explored, and which we specifically addressed in this work, is whether a blockchain platform can actually support the scaling and timing requirements specific to time-critical services, i.e., the quick power modulation actions that an aggregator is expected to implement in response to any unpredictable power unbalancing events occurring in the grid.

In this paper, to answer this question, we evaluated the feasibility and performance of a blockchain-based implementation of the aggregate power modulation service of an aggregator's virtual power plant. This has been carried out by creating a prototype application, complete with front-end and back-end system components, which operates on a real blockchain and sends the modulation commands to the prosumer's devices. Nevertheless, the aggregator monitors the prosumer's smart meters directly and in real time, to gather the readings of the prosumer-generated power and continuously assess the ability of the prosumers to follow the requested operational mode, while avoiding any impact due to the blockchain.

The main contribution of the paper is in the way the commands handling the modulation of production are exchanged. The work gives a thorough analysis of the potential use of a blockchain to handle energy modulation commands, while ensuring that all parties receive the commands, and none of them can dispute the given commands. As a second contribution, we implemented a prototype application realizing the designed balancing operator service, and the paper presents the results of running the implemented prototype on an actual blockchain. Moreover, the developed prototype application provided means to reveal the technical issues that have to be considered to successfully design, implement, and deploy a scalable application on the Ethereum blockchain.

The remainder of the paper is structured as follows. Section 2 explores the main related works and compares them with our solution. Section 3 describes the represented real-world scenario. Section 4 details the actual system architecture that we have implemented. Section 5 analyses all the technical results collected when running the implemented test, and finally, in Section 6, we draw our conclusive remarks.

2. Related Work

The potential applications of the blockchain in the energy sector have always been regarded as crucial, as it allows managing complexity, data security, and ownership among grids, and especially, it enables new business models and marketplaces and the opportunity to actively engage small actors in the market of renewable energy sources (RESs) [9]. The blockchain enables the creation of energy communities in which the transparency and trust of the marketplace can guarantee accountability while preserving privacy requirements.

In addition to the many existing surveys on the application of blockchain to the energy sector [9–12], conceptual work has been also delivered on RES-specific topics that are related to the application investigated in this paper, in particular about the handling of the grid flexibility demand/response mechanism and the provision of platforms for more efficient billing processes. By using decentralized ledgers, end-users could benefit from the automation and flexibility of decentralized trade.

The main benefit of the decentralization introduced by the blockchain in this application is in the way the commands handling the modulation of production are exchanged. The work gives a thorough analysis of the potential use of a blockchain to handle energy-modulation commands, while ensuring that all parties receive the commands, and none of them can dispute the given commands. Moreover, the prosumers gain credit for actively contributing to implementing the power modulations, instead of having to trust (and pay) a central agent to actually manage this task. The only trusted central mechanism regulating the service is the smart contract deployed in the blockchain, which is public and checkable well before its use by the prosumers. The transparency of the agreed service is then guaranteed by the blockchain.

Decusatis and Lotay [13] have tackled security issues for an Ethereum blockchain that hosts a decentralized energy-management application, presenting an approach to digital identity management that would require smart meters to authenticate with the blockchain ledger and mitigate identity spoofing attacks.

Yang et al. [14] have explored the potentiality of the virtual power plant (VPP) by using a blockchain for managing RESs. Their proposed energy-management platform allows users to perform energy transactions in a way that they can trade energy for mutual benefits and provide network services, such as feed-in energy, reserve, and demand response, through the VPP. The conceptual work has been validated by a prototype blockchain network whereby the VPP energy-management scheme has been successfully tested with respect to users' energy trading and other network services. BloRin [15] is an academic and industrial initiative that aims to create a blockchain-based technology platform to favor the creation of solar smart communities and to encourage trustworthy interactions between prosumers. The use of the blockchain platform, based on Hyperledger Fabric, manages the accounting of energy flows and the automation of economic transactions. EFLEX [16] is a pilot service being carried out in Bulgaria and Romania whose main objective is to demonstrate the trading and flexibility of services amongst Transmission Service Operators (TSOs), Distribution Service Operators (DSOs), and small prosumers in a transparent, secure, and cost-effective manner using a blockchain-based flexible marketplace. The aim is to look for ways to help both DSOs and TSOs be more directly engaged in managing energy flows on the network by using the same decentralized platform with the aid of specifically designed smart contracts. Umar et al. [17] have studied the integration of decentralized battery storage equipped with smart meters to manage the distributed energy resources (DER) using a platform aiming at achieving a self-sustaining community. A case study was developed using the blockchain to provide secure trading among users.

Pop et al. [18] have proposed a blockchain-aided system that uses decentralized mechanisms for achieving transparent, secure, reliable, and timely energy flexibility. This has been pursued by adapting the energy demand profiles of all prosumers involved. For this, blockchain storage handles the energy exchange data collected from smart meters, and smart contracts implement the flexibility rules for each prosumer, with the handling of the associated rewards/penalties. The devised mechanisms have been validated using a prototype implemented in an Ethereum platform.

In the present work, in contrast to the existing literature, we examined a smart grid scenario where a virtual operator, the aggregator, administers an amount of distributed energy resources on behalf of transmission or distribution operators to cope with the planned or unexpected fluctuation of the grid power, in particular examining the feasibility and performance of an aggregator implementation relying on a blockchain to support its management services, specifically in the operational scenarios concerning the collective

implementation of changes in the aggregate power produced, to fulfil balancing requests from the grid operators. There is no existing work in the literature, to the best of our knowledge, that considers the exact identical type of assessment and application we studied here. In addition, besides the originality and intrinsic interest of the innovative industrial case study considered, please note that the result of this research has a strong significance as an assessment of the real pros/cons in the practical application of a blockchain not only as a secure and tamper-proof middleware, but also as a middleware to coordinate decentralized applications involving (even though not real-time) timing constraints on the interactions.

The purpose of the tests carried out in this work is to determine, for the implemented case study, a few experimental outcomes:

i. The number of DERs that an aggregator can manage on a blockchain without introducing unacceptable delays in the service;
ii. To highlight any technological constraints of a blockchain-based implementation that can affect the scalability of the service.

3. Reproduced Real-World Scenario

We considered that an aggregator virtual operator implements an energy balance service. It administers an amount of distributed energy resources (DERs) on behalf of the transmission or distribution operators to cope with planned or unexpected occasional fluctuations of the grid power (unbalance events) by opportunely modulating its own power contribution.

The aggregator stipulates contracts with remote owners of distributed energy resources (DERs), of variable plant sizes, i.e., prosumers equipped with either small solar systems, wind turbines, or commercial batteries capable of delivering electric power in the range of a few KW to a MW. The aggregator classifies the managed sources in terms of a few parameters such as the following: (1) the produced power size; (2) the type of source (i.e., wind, solar, batteries, etc.), which can affect its overall reliability and availability; (3) the level of flexibility offered as a percentage of the contractual base power it is able to modulate up or down; (4) the reference time profile, that is the daily time slot(s) in which the power variation can occur.

We assumed the number of managed DERs as a parameter and that their contractual energy baselines are statistically distributed according to a Poisson curve. Thus, we will have a few large-sized DERs and increasingly many small-sized DERs contributing to the flows of managed energy and data. We will then increment the number of managed DERs and expect that some technical upper limit will be reached. Each DER has to periodically output to the aggregator a real-time reading of its produced power, where one Hz was deemed as a reasonable frequency for all DERs' periodic power readings.

Occasionally, DERs receive local power modulation requests, to be implemented by the underlying energy plant as an increase/decrease of power by a given amount, which is also parametric. Modulation requests are asynchronous requests issued by the aggregator and implemented evenly by all managed DERs: each will apply the same relative (percentage of) local modulation. A given aggregate modulation command has to be applied inside the same time window by all DERs, and no new command can be issued in time windows that overlap. The time window, or transient window, is defined by two parameters, `{start:datetime}` and `{end:datetime}`, indicating the start-time and end-time for the requested aggregate modulation to be implemented. Acceptable transients in the real world can range from a minimum of a few minutes up to a few hours, depending on the context, and the modulation start can be immediate or in the future, e.g., within the next twelve or twenty-four hours. In the investigated scenario, a fifteen-minute window size has been used.

In the scenarios we studied, the users that are only consumers do not play any active role; however, they might cause an unexpected energy unbalance event for excessive power draw. Being just consumers, they are not involved in the recovery action. Of course, some

consumers could proactively reduce their power draw to help with the recovery. In this case, they would have to interact with the aggregator to register their service contracts and log their actual contributions to the blockchain. However, in this paper, consumers are assumed to be just passive users.

A decentralized application (DApp) has been developed and deployed on an actual blockchain for the described scenario. Tests have then been executed to measure and analyze the blockchain subsystem's performance under variable stress conditions. These have been conceived to allow us to draw appropriate conclusions on the adequacy of the blockchain technology to support real-world applications with timing constraints, like the considered energy-grid-balancing scenario. Full details on the case study's actual implementation are given in the following section.

4. Prototype Decentralized Application

The software architecture of the proposed DApp consists of three parts: a set of components constituting the underlying blockchain subsystem; a back-end subsystem, which is a set of (TypeScript) classes and smart contracts created to interact with the blockchain and modeling the DERs' functional operations; and a web-based front-end, which is the point of view of the aggregator operator. The latter is the user interface to configure all operating parameters (e.g., the number of involved DERs) and allows us to run and monitor the interactions between the aggregator and the DERs through the blockchain. The proposed DApp was developed using the Electron framework, thus making it possible to be a multi-OS application.

The three-tiered architecture of the whole developed DApp is depicted in Figure 1 and explained in greater detail in the following.

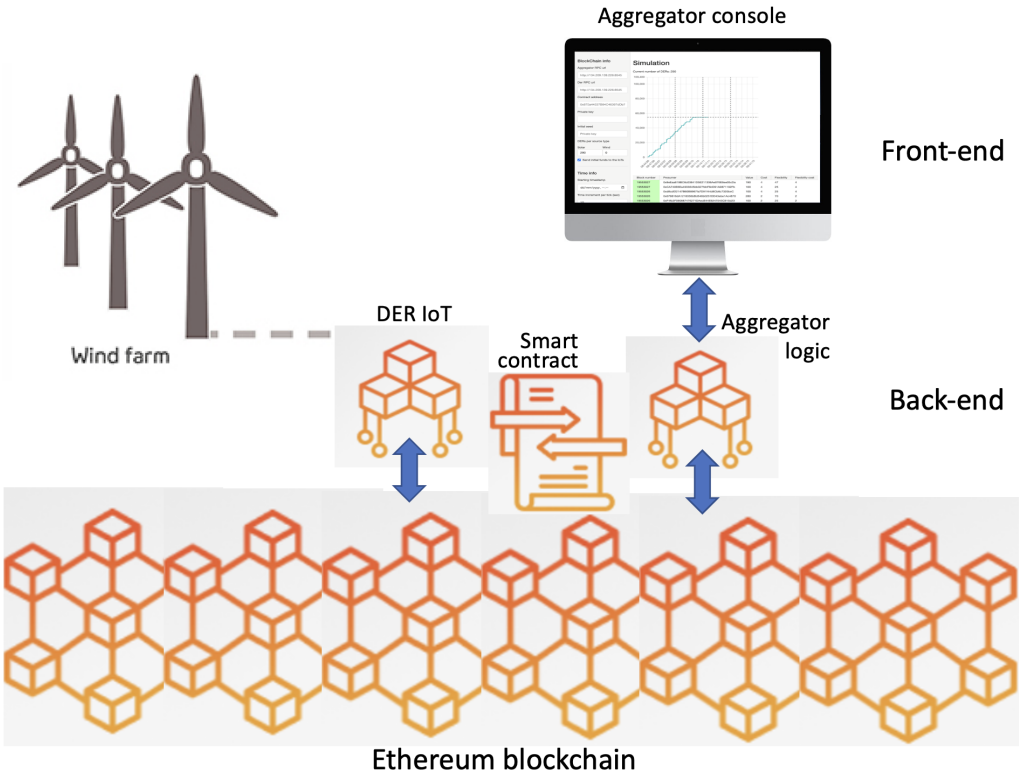


Figure 1. Three-tiered architecture of the developed DApp. The front-end tier is a set of interface components. The back-end tier is the set of components implementing the aggregator's and the DERs' sides, consisting of their respective blockchain accounts. The third tier is the Ethereum blockchain infrastructure, which the system relies on.

4.1. Blockchain Environment

The experimental blockchain testbed for our investigations is based on a permissionless Ethereum blockchain [19]. Specifically, Volta is one of the testnets deployed by the Energy Web project and was used since it allows testing any real-world DApps without using a real payment currency for the transactions.

The Ethereum blockchain is the underlying platform for the Energy Web Consortium, a well-established global initiative to promote a decentralized, green energy market. The consortium has proposed a framework consisting of a layered architecture on top of Ethereum for several energy-market-related tasks. In our previous works, we experimented with the Energy Web framework, and a further evaluation was needed, i.e., specifically crafted stress tests of the blockchain's capabilities. Therefore, the proposed prototype energy service was implemented in the same blockchain, as the goal was evaluating the feasibility of the proposed application on the Energy Web's blockchain.

In the deployed setting, the aggregator is equipped with a light Ethereum client implementation, based on the public code of the Nethermind client (https://nethermind.io/, accessed 14 February 2024). A light Ethereum client is a lightweight alternative to a full Ethereum node installation, missing the validation ability, which would require around 200 GB of storage space and demand high computational resources. A light node can still act as a Web3 Provider, that is the interface endpoint for accessing any Ethereum-based blockchain. The aggregator uses the light node to broadcast transactions to prosumers, requiring implementing local energy modulation actions.

A second dedicated light Ethereum node was introduced into the system to act as the reference access point for all DERs' IoT devices (the smart meters), exposing the dedicated RPC service to allow them to remotely access the Ethereum network as well. Each DER was given an identification account in the Ethereum test network, thus interacting with it through the Ethereum APIs. It also interacts with the DApp interface module through RPCs over the HTTP protocol. Note that the smart meters' additional hardware and software requirements to interface with an external blockchain access node through an HTTP or Web Socket channel are minimal.

4.2. DApp Front-End

The developed DApp has a front-end module providing the user with a web-based graphical interface to manage the balance service operation from the aggregator's point of view. That is, it allows the user to issue aggregated modulation (balance) commands to the associated DERs and configure and install in the blockchain a set of smart contracts that govern the interactions with the DERs. Finally, the front-end allows users to instantiate within the blockchain a set of identities, for an amount that is user-configured. This set of blockchain identities, i.e., accounts, corresponds to the blockchain endpoints of the pool of managed DERs. Thus, any transaction involving one of the said accounts corresponds to an interaction between the smart IoT device of a specific DER plant and the aggregator, and will, in turn, be the execution of a specific smart contract transaction implementing one of their business logic operations. Since Ethereum is a public blockchain, open to anybody to start transactions, to protect our system from tampering, our application's smart contract firstly checks that the originating account is either the aggregator or a known prosumer, i.e., the prosumer has to have an account already registered with the aggregator service.

A large part of the front-end graphical interface is reserved for the monitoring dashboard, which, based on the data continuously read from the blockchain, draws a graph showing to the user the current cumulative power output of the managed DERs' pool, in real time. Figure 2 shows a screenshot of the front-end during the system's initialization phase.

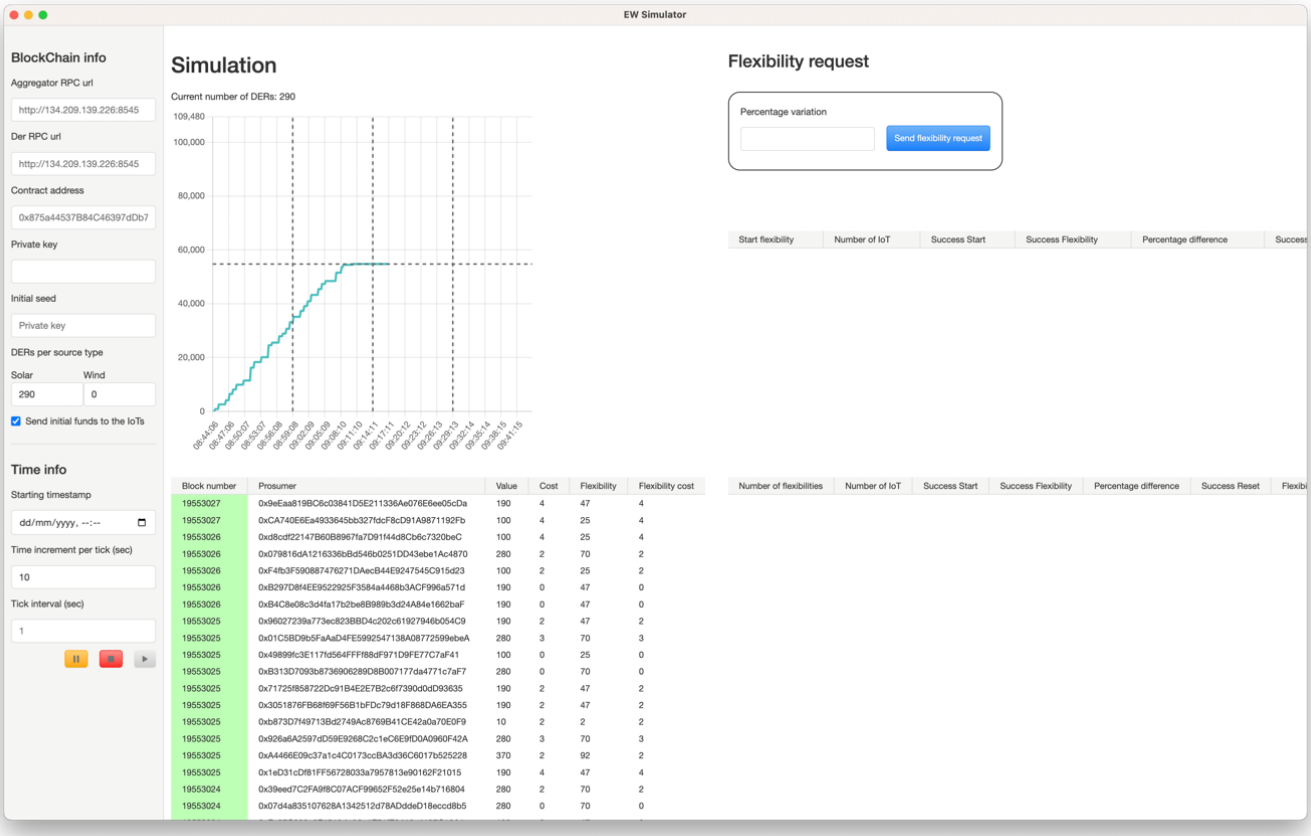


Figure 2. Front-end page of the virtual balance operator DApp, shown in the DERs' pool set-up phase (last accessed on 28 September 2023).

While DERs interact with the aggregator exclusively through the blockchain, a set of corresponding modules, named "IoTs", is instantiated for each DER in the DApp front-end, to represent the data flow out of smart metering IoT devices in the DERs. The aggregator then acquires those readings via secure and private Internet links (that is, outside the blockchain) to have a real-time monitoring of the DERs' plant energy production.

For the real-time monitoring of produced/consumed power, the aggregator relies on its own smart meters, installed on the premises of the prosumers' or consumers' sites. These data flows are assumed to travel on a separate network, i.e., encoded over the powerline or a suitable digital link, and flow unidirectionally towards the aggregator. This is a choice motivated by several considerations: (1) blockchains do allow trustless "agreements", but do not provide a way to trust "unilateral" statements: the meter readings can only be trusted by the aggregator if coming from its own meters; (2) the meter readings are relatively frequent and continuous, and blockchains are not ideal tools to handle fast-paced, sustained, real-time data flows; (3) the meter readings have a very short lifespan, and logging them to the blockchain would be pointless unless the application requires a public and undisputable log to be created for future reference. Other than that, they are only temporarily interesting for the aggregator in order to perform a reality check, to verify the correctness of the power data claimed by the prosumers in their recent transaction requests.

Please note that relying on an external communication network for this one-way monitoring data-path complements the blockchain interactions and does not compromise the trustless relation that the blockchain implements between the large community of users and the aggregator for the realization of the balancing service, which is the core of the proposed design.

To simulate the DERs' physical smart meters, required to test our DApp, a corresponding number of IoT software objects was instantiated to generate simulated data with an algorithm corresponding to one of (currently) two types of renewable energy sources, solar and wind, with a reading frequency that can be tuned from the DApp front-end.

The DERs' IoTs objects were instantiated by the DApp in the set-up phase at once, and the generation of their DERs' blockchain accounts was performed according to a pseudo-random algorithm ("Hierarchical Key Generation"), which is based on a seed and a deterministic mechanism for pseudo-casual number generation. In this way, the DApp can reuse the same seed and always generate the same set of accounts. This allowed us to test-run the DApp many times while making it possible to compare the transaction logs for the same DER account, for performance analysis and debugging purposes.

4.3. Aggregator Smart Contract

A smart contract was designed to regulate the interactions between the aggregator and its managed DERs, exhibiting the following functionalities: (i) registration and updating of the contractual agreements of a DER; (ii) the transmission of modulation commands to registered DERs and accounting for the implemented modulations on the DERs' blockchain accounts. The DApp set-up phase takes care of deploying the developed smart contract catering to the tasks to be executed in the blockchain, and the aggregator is the account owner of the smart contract implemented in Solidity.

The smart contract management of contractual agreements was implemented in the following operations: `registerAgreement()`, `reviseAgreement()`, and `cancelAgreement()`, which are used to govern the creation of a contractual agreement with a DER, to later alter its terms, or to allow deleting it, respectively. Such operations were executed as part of the transaction requests started by the DERs. The contractual agreements will take into account data about the energy source type, the expected amount and price of the produced energy, and the flexibility that the DER is expected to provide. Upon successful termination of one of this type of transaction, a smart contract internal data structure named `agreements`, containing the data of registered DERs' contracts, will be correspondingly updated.

The second main functionality of the smart contract regards the power modulations. Such transactions will be started by the aggregator and will use the following data:

- Flexibility: the percentage of modulation that the network must produce with respect to the baseline;
- Start: the deadline by which all DERs must reach the required modulated value;
- End: the indication of the moment from which the DERs have fifteen minutes to return to baseline.

The DApp allows a Transmission System Operator (TSO) to issue a modulation request command specifying the baseline percentage to be altered. The Ethereum blockchain event mechanism was used to allow the aggregator to collectively notify an aggregate modulation request to the whole pool of DERs while storing the relative instance data in the blockchain transaction logs. This dramatically improves performance, as the aggregator avoids having to referto a long sequence of transactions for any time-critical interaction with the DERs. The system reaction for an aggregate modulation request sent to IoT devices is shown in the sequence diagram sketched in Figure 3, and the corresponding sequence of interactions is also detailed in the following. `IPC Handler` is a component letting the Electron-framework-based front-end communicate with the server side. `Tracker` handles the global timing for the events.

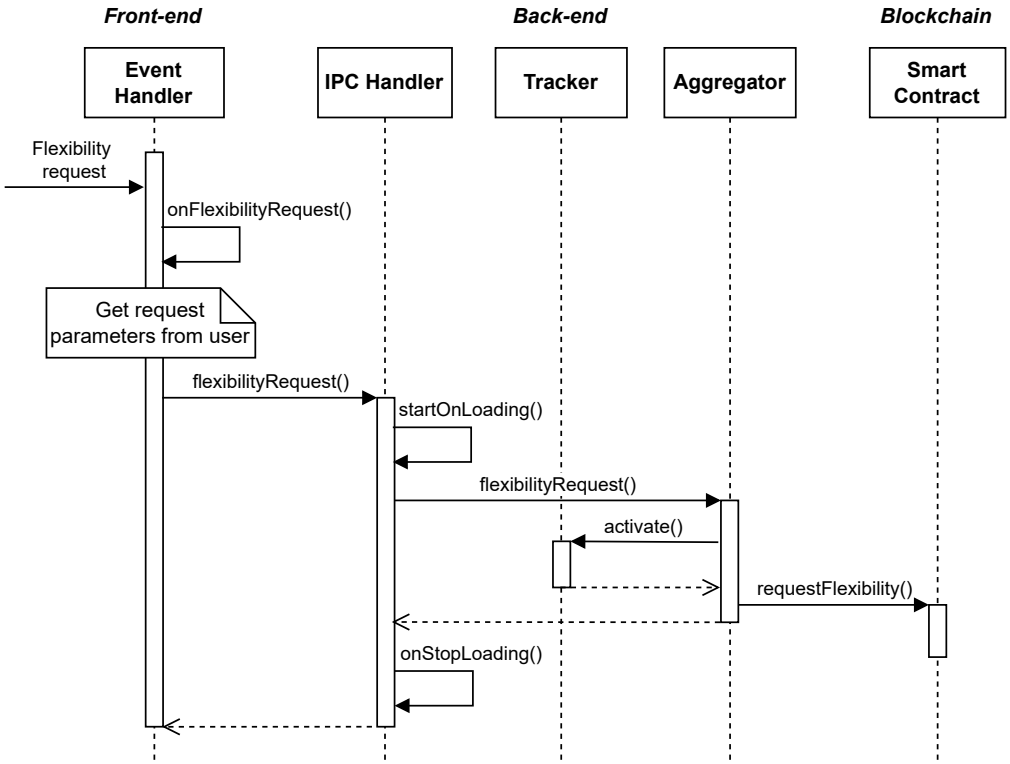


Figure 3. Sequence diagram of interactions to implement an aggregate modulation request issued.

i. Upon receipt of a user-generated aggregate modulation request from the front-end user interface, parameterized by the overall modulation amount and the two timestamps for the beginning and end of a desired duration, the aggregator saves such data for later in an internal `flexibilityRequestdata` data structure, and upon the aggregate modulation, start: time, will invoke the `requestFlexibility()` smart contract function.

ii. Operation `requestFlexibility()` emits a `RequestFlexibility()` event to be published on the blockchain logs, to notify about the modulation request each DER in the managed pool, in parallel. The DERs' IoT devices (simulated by the IoT objects) intercept the notifications of this event, by means of a specific event watcher implemented in the smart contract (an event watcher is a specific Web3 functionality that allows a requester to be notified when a filtered event occurs in a blockchain log).

iii. Upon event notification, every notified DER has to implement the modulation, for three times or stages: a modulation start transient, where the DER output power will rise from the contractual baseline value to the requested modulated value; a modulated regime stage, where the DER will hold this modulated power value steady; and a modulation end transient, where the DER will gradually return to its contractual baseline. Upon completion, the DER will call the smart contract's `provideFlexibility()` function, passing it the modulation amplitude it declares to have accomplished, which potentially could turn out to be either compliant or non-compliant with the issued request.

iv. The aggregator, by monitoring the parallel real-time data flow from the DERs' smart meters, can check and determine whether the modulation declared by each DER is truthful (matches the real-time smart meter readings) and whether it conforms to the requested modulated amplitude or not. Similarly, at the aggregate modulation *end:* deadline, it will invoke the `endFlexibilityRequest()` smart contract function, with the list of collected results. This, in turn, will save those data in a `flexibilityResults` data structure and emit the `EndFlexibilityRequest()` event to notify about this result the DERs.

v. Upon this last notification, the DERs invoke the `provideFlexibilityFare()` function to register their contribution and claim a corresponding contractual reward, by means of a non-repudiation blockchain transaction. This transaction will have the aggregator as the counterpart and will actually be successful if and only if there has been an agreement on the conformity of the DERs' implemented modulation. The aggregator pays a credit to the prosumers for their contribution in restoring the unbalanced power network. The prosumers can then use this credit to pay for their next transactions or their own eventual power consumption bills. The aggregator would take this money out of its profits for energy selling.

The smart contract's internal storage is only used for the state variables required by its functions and includes the data structures shown in Listing 1.

Listing 1. State variables for the smart contract governing DERs.

```
address public immutable aggregator; //aggregator's account
int256 public energyBalance; //instant aggregate power
mapping(address=>Prosumer) public prosumers; //prosumers' states
mapping(address=>Agreement) public agreements; //prosumers' contracts
address[] public prosumerList; //prosumers' accounts
FlexibilityRequest public flexibilityRequest;//requested aggregate modulation
mapping(address=>int256) public flexibilityResults; //prosumers' modulation
    outcomes
```

The smart contract defines and makes use of the event types in Listing 2.

Listing 2. Events defined and used by the smart contract governing DERs.

```
event RegisterAgreement(...);
event ReviseAgreement(...);
event CancelAgreement(...);
event RequestFlexibility(...);
event EndRequestFlexibility(...);
event FlexibilityProvisioningSuccess(...);
event FlexibilityProvisioningError(...);
event RewardProduction(...);
```

Listing 3 provides a code snippet simulating the invocation of the smart contract's `requestFlexibility()` function.

Listing 3. Test snippet to simulate a smart contract's function call.

```
describe("requestFlexibility", function () {
  it("create a new flexibility request", async function () {
    await contract.requestFlexibility(start, end, gridFlexibility);
    const request = await contract.flexibilityRequest();
    expect(request.start.toNumber()).to.equal(start);
    expect(request.end.toNumber()).to.equal(end);
    expect(request.gridFlexibility.toNumber()).to.equal(gridFlexibility);
  });
  it("revert on unauthorized use with 'UnauthorizedAggregatorError(msg.sender
      )'", async function () {
    await expect(iot1Contract.requestFlexibility(start,end,gridFlexibility))
    .to.be.revertedWithCustomError(contract, ContractError.
        UnauthorizedAggregatorError)
    .withArgs(iot1Addr);
  });
});
```

For the management of the data within the blockchain, the smart contract uses both its associated storage, for storing the ephemeral variables representing the contract's state, and the transaction logs, for a persistent and much less expensive record keeping of public information related to the occurrence of triggered events.

The smart contract has been developed using the Hardhat framework (https://hardhat.org/ accessed 14 February 2024), a development environment for smart contracts that also supports debugging, compilation, and testing. Furthermore, Hardhat includes the functionalities to deploy the smart contract on a blockchain through a simple configuration script (see Listing 4), which we provided with the private key associated with the aggregator's blockchain account (with sufficient credit to pay the transaction) and the address of the RPC-API reference light node to connect to.

Listing 4. Script to deploy the aggregator's smart contract on Energy Web chain.

```
import { ethers } from "hardhat";
async function main() {
  const cntrcFactory = await ethers.getContractFactory("AggregatorContract");
  const contract = await cntrcFactory.deploy({ maxPriorityFeePerGas: 7 });
  await contract.deployed();
}
main().catch((error) => {
  console.error(error);
  process.exitCode = 1;
});
```

The Hardhat framework additionally provides a series of libraries for quick and easy testing of smart contracts internally in a simulated blockchain environment, so as to validate the expected behavior of the contract's functionalities before deployment. Tests include checking the return value of the invoked function, verifying the correct alteration of the status of the contract, and ensuring that a due exception is produced in the case of invalid inputs. Putting all these tests together, considerable coverage of the smart code was achieved.

4.4. Back-End Architecture

The DApp back-end is the software module implemented to support the DApp set-up at launch time and to simulate the real DERs' operation. DERs have an active role in supporting the modulation service by actually interacting with the aggregator through the blockchain, either by synchronous smart contract transactions or asynchronous event logging, as detailed in the previous subsection. The positioning of the back-end tier in the whole architecture clearly shows how it supports the system protocol implementation on the actual blockchain.

Among the implemented classes, the `Tracker` class is used by other classes to have a common timeline reference, which is tick-configurable and reset upon simulation start. The aggregator uses a `IoTFactory` class to initialize a size-configurable collection of `IoT` instance objects, modeling the DERs' smart metering IoT devices, each storing its own blockchain account, that is a (private-key, public-key) pair.

The DApp set-up phase includes resetting the smart contract status (its temporary data storage) and sending funds to each created DER account to enable its operation on the blockchain. In fact, the first task of each IoT instance is to trigger a blockchain transaction to register their DER's service agreement with the aggregator, by means of smart contract functions.

Once registered, the DER is added to the aggregator's managed pool and contributes its power to the overall managed power. The contributed amount is periodically updated in the IoT instance, simulating periodic smart meter readings of the energy produced in the underlying plant. At this stage, the graph window in the DApp front-end module shows in real time the aggregate power value coming from all so-far registered DERs. Once all DERs are registered with the aggregator, this value shall reach a cumulative baseline value and stay steady, until an aggregate modulation request is issued. This is clearly shown in the DApp front-end graph window in Figure 2, which is a snapshot of the DApp's status just after completion of the set-up stage, when running with a configured pool of 290 simulated DERs, totaling 58 MW of baseline power.

5. Assessment of the Results

The presented solution was specifically designed and developed as a proof of concept to assess the advantages and drawbacks of using a blockchain for supporting the operating protocol of a real-world decentralized application with basic time constraints, namely the energy-balancing operator service. Such a service requires the whole set of contributing actors to react in a timely manner to a collectively shared service request and implement their part of a (simple) collective behavior within a specific time frame, despite any communication delay that the blockchain itself might add.

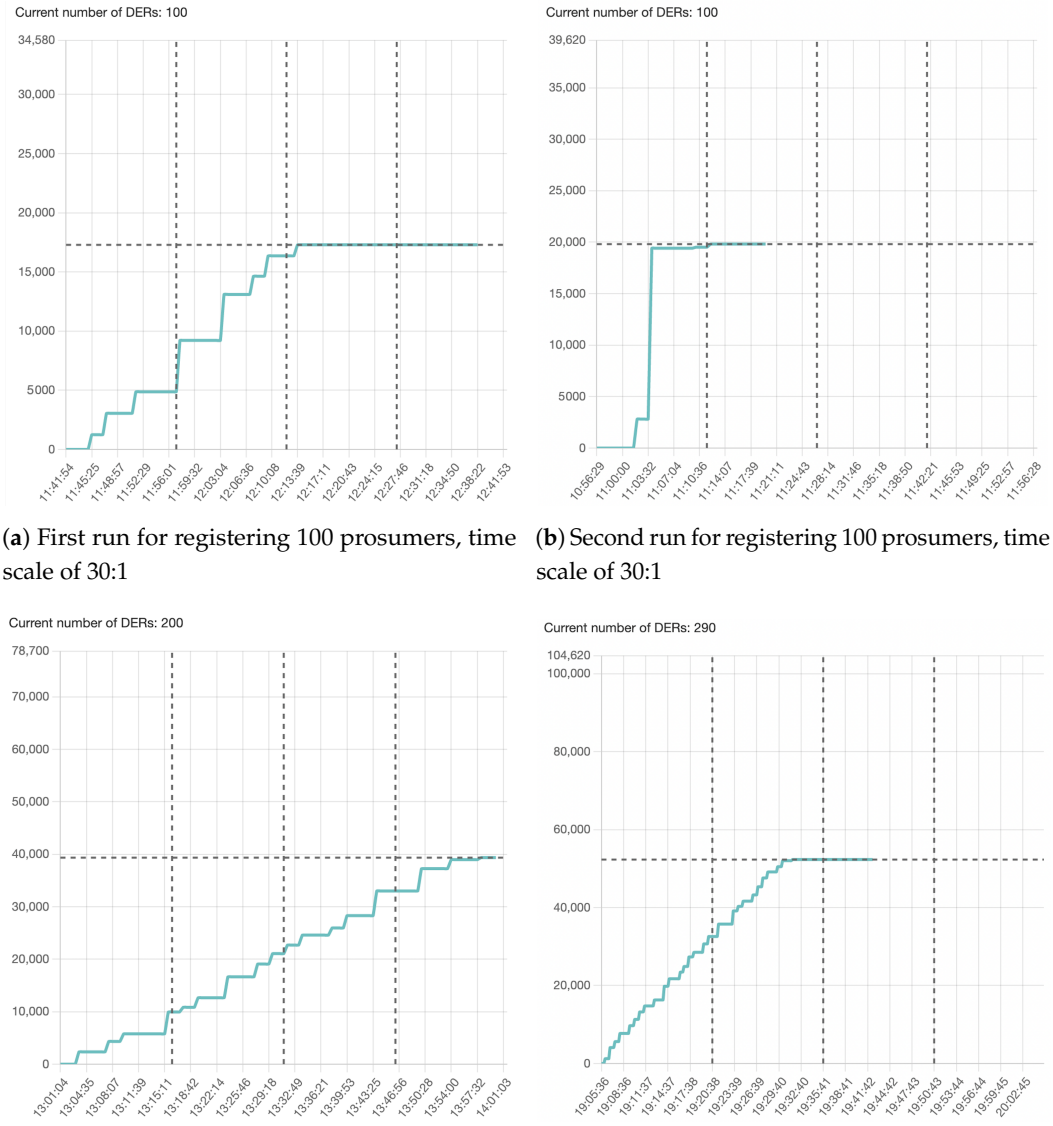
This raises two research questions: How high can the size of this set of contributing actors be scaled up? How much will the blockchain actually affect their interaction delays? To answer these questions, a very large set of tests was run to profile the behavior of the blockchain network in the implemented case study, to identify any factors affecting its scalability and, whenever possible, suggest strategies to overcome them.

The first actual data obtained from the tests showed that the developed DApp could not scale up to a managed pool size greater than 290–300 DERs, with slightly varying results depending on the considered DApp prototype version. Beyond that upper boundary, the DApp showed unpredictable variations and/or stalls in the DApp protocol interactions. This upper bound was reached gradually, starting with a few tens of DERs, and then repeatedly testing the DApp with increasingly complex scenarios, by steps of ten additional DERs each. During the research experiments, as soon as we had identified a solution to overcome a specific technical implementation issue behind the observed scalability limits, we gradually integrated it into the DApp prototype. As a consequence, only the final DApp version is presented and discussed in this paper, which, as we said, works correctly and interacts reliably with the blockchain testnet for up to 290 managed DERs. The reasons for this behavior were thoroughly examined and are discussed throughout this section, while in the following, we present several example system behaviors observed in the DApp test runs.

Figure 4 shows several plots of the ramp-up of the aggregated power, for 100, 200, and 290 registered prosumers, at the completion of the application setup (registration) phase, for successful completion cases. Figure 5 shows some failure cases, that is when the programmed number of prosumers that should have registered with the aggregator service was not reached, since their transaction request issued to the blockchain failed somehow.

Figures 6 and 7 show several aggregated power graph screenshots representative of the observed behavior in response to some of the more interesting cases of the tested modulation requests for some success and some failure cases, respectively. A modulation was successful when the target modulated power was fully reached (with the contribution of all prosumers) in a transient time less than the (real-world inspired) max lapse of 15 min. It was unsuccessful when it did not, due to some (even very few) prosumers failing to correctly interact with the aggregator by means of the blockchain.

Figures 4–7 show a graphic area that was originally designed to represent a 60 min long period, as well as four consecutive 15 min long partitions separated by three dotted vertical lines. The three lines indicate in the graph the occurrence of three milestone events in the implemented power modulation duty cycle, respectively: (1) the first line (from the left) is the end of the allowed transient from the baseline to the target modulated power; (2) the second line denotes the end of the steady state and the start of the transient from target power back to baseline; (3) the third line is the maximum allowed time to end this latter transient and have, again, a steady baseline power.



(**a**) First run for registering 100 prosumers, time scale of 30:1

(**b**) Second run for registering 100 prosumers, time scale of 30:1

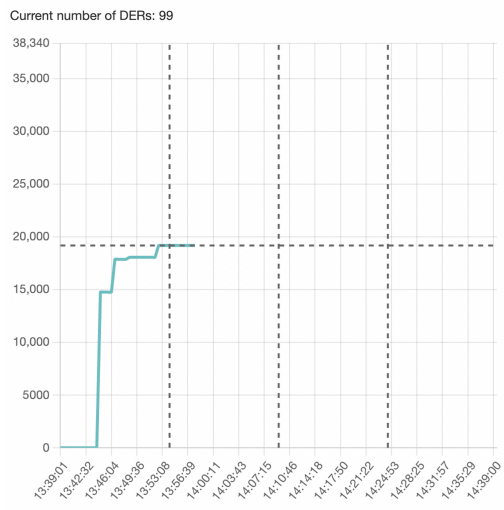
(**c**) Third run for registering 200 prosumers, time scale of 30:1

(**d**) Fourth run for registering 290 prosumers, time scale of 10:1

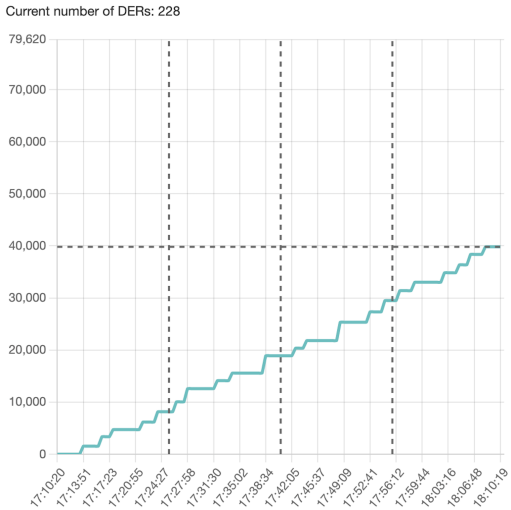
Figure 4. Plots of several runs of the set-up phase, where prosumers register their agreement with the aggregator by starting a transaction on the blockchain. The resulting behavior can be different in relation to the current load in the blockchain, as for the first run (**a**), many more new blocks were actually added with respect to the second case (**b**), while attempting to register the very same number of prosumers; the registered prosumer count was gradually increased up to 200 (**c**) and 290 (**d**). The y-axis shows the (simulated) aggregate power in KW. The whole panel is divided into four equal parts by vertical dashed lines. The X-axis shows the elapsed time scaled to make it easier to see the steps in the power ramp, which, in turn, correspond each to an additional block of confirmed transactions in the blockchain, including several of our DApp transactions.

The graph areas report the simulation time of day on the X-axis and the value of the current aggregate power on the Y-axis. In some cases, we observed a quick reaction with respect to the allowed transients, then the depicted graph ramps were almost vertical. A zooming factor for the X-axis was added in the application front-end to allow the user to change the 1:1 time scale to a more convenient N:1 scale and magnify the graphic ramp to appreciate its single steps of increase. Each graph caption reports its time scale factor, so that it is possible to correctly interpret and compare the reported graph data. The reaction times were widely distributed; in some cases, the blockchain was unloaded

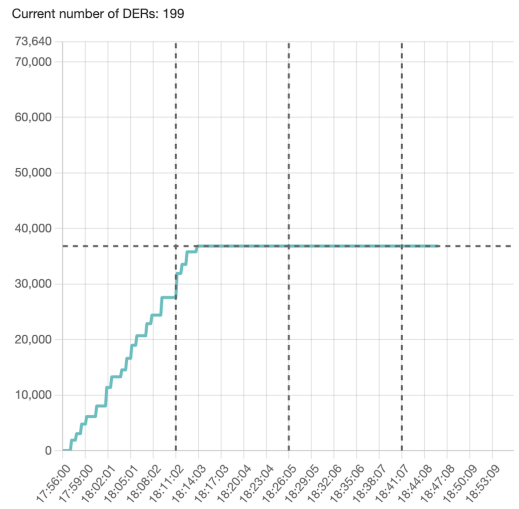
and everything worked really quickly and smoothly; in some other cases, the requested transactions were processed in lots, with totally unpredictable delays between the lots, and unacceptable delays for the considered service, which has a 15 min max duration allowed for transients.



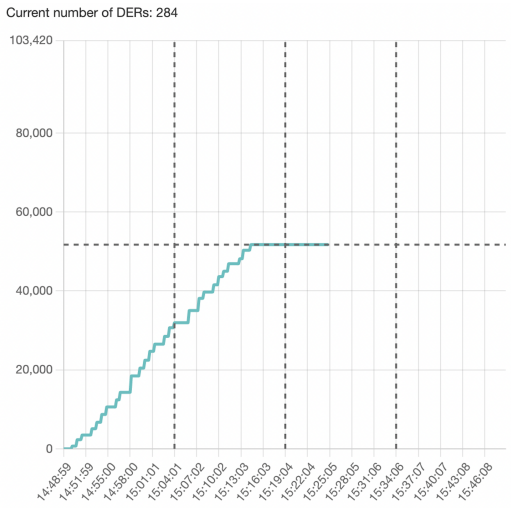
(**a**) Failing set-up phase, originally scheduled for 100 prosumers; the time scale is 30:1



(**b**) Failing set-up phase, originally scheduled for 250 prosumers; the time scale is 30:1



(**c**) Failing set-up phase for 200 prosumers; the time scale is 10:1

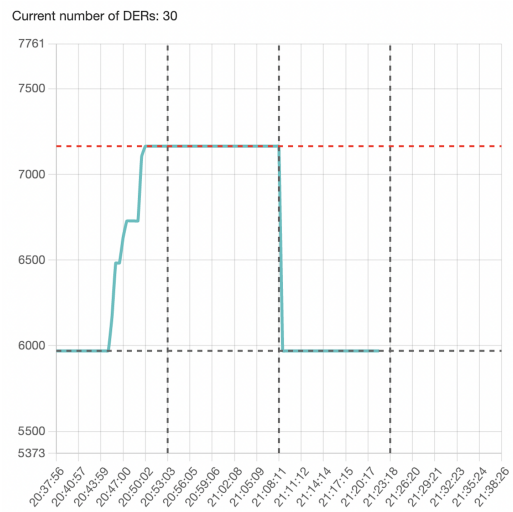


(**d**) Failing set-up phase for 290 prosumers; the time scale is 20:1

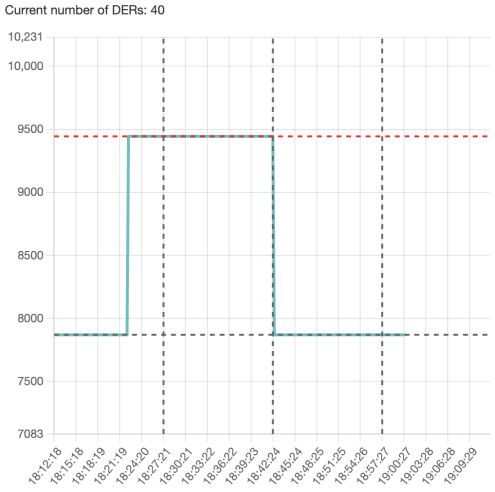
Figure 5. Plots of several runs reporting failing service set-up phases: the count of registered prosumers does not reach the planned total, as it was set in the simulation control panel; a few prosumer's transaction requests did not get finalized, due to one of the technical reasons undermining the successful service execution.

By observing the plots showing the prosumers' agreements in the set-up phase (see Figures 4 and 5) or those regarding the implemented power modulations (see Figures 6 and 7), it is interesting to note how the resulting behavior changes substantially even when running in identical or very similar settings. The different results are the effect of the load of the underlying blockchain, when the blockchain is overloaded and transactions are processed in lots, with unpredictable delays between them (see, e.g., Figure 4a,b), or randomly causing some transactions to stay in pending status for an indefinitely long time, or to fail immediately (see, e.g., Figures 5a,b and 7a,b), or vice versa, when the blockchain

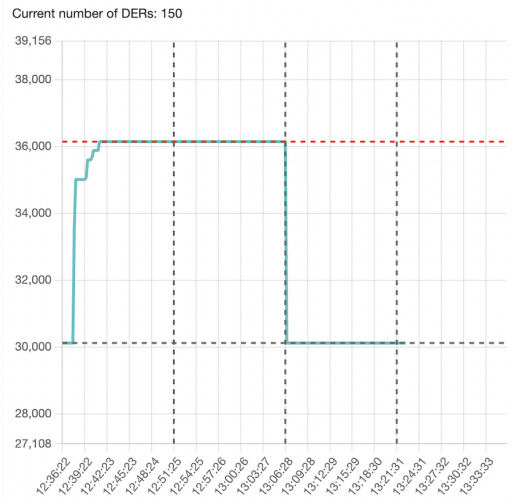
is almost unloaded, it responds quickly, smoothly, and reliably, confirming all transactions in a few blocks (see, e.g., Figures 4b and 6). As the Ethereum Volta testnet is a public access ledger, just as Ethereum itself, it is not possible to control the network load and/or predict with precision its behavior in a certain time frame. Of course, failing transactions correspond to missing interaction steps between the applications actors, which, in turn, cause the application's business logic to fail irremediably, at least as it is currently. One further step of development could be to redesign the application to be more resilient or robust, if possible, with respect to these unpredictable network failures.



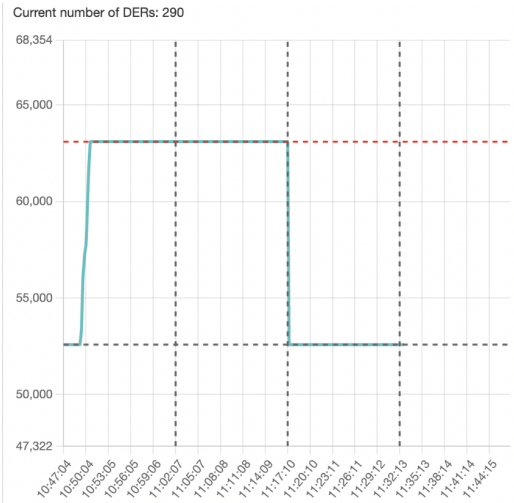
(**a**) Modulation by 30 prosumers; the time scale is 30:1



(**b**) Modulation by 40 prosumers; the time scale is 30:1

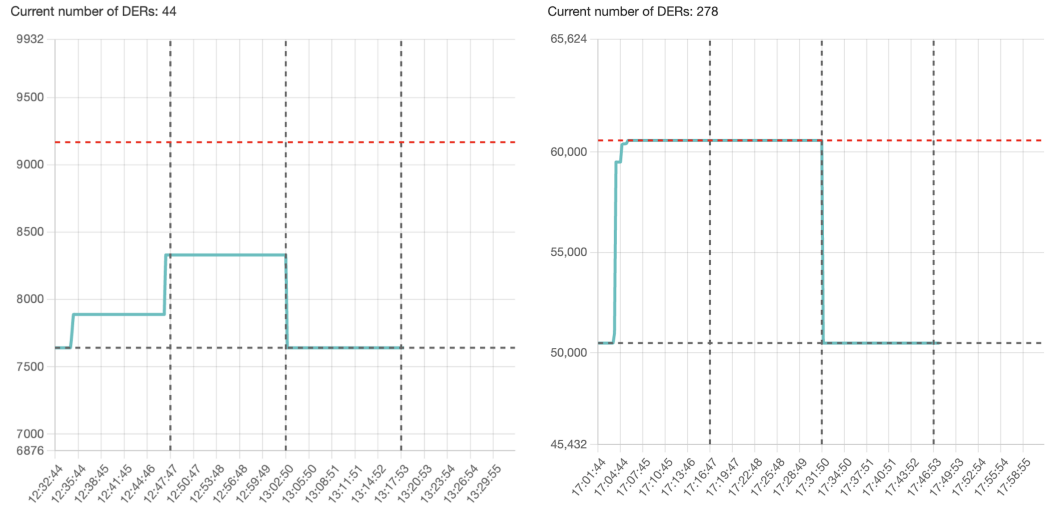


(**c**) Modulation by 150 prosumers; the time scale is 12:1



(**d**) Modulation by 290 prosumers; the time scale is 10:1

Figure 6. Plots showing power modulation when issuing commands in different scenarios of increasing size in terms of participating prosumers. The dashed red line is the reference modulated power target, while the horizontal dashed black line is the power baseline. Test runs in very similar initial conditions showed different dynamic behaviors, due to the unpredictable responsiveness of the underlying blockchain.



(**a**) Failed modulation with 100 prosumers; the time scale is 12:1

(**b**) Failed Modulation by 300 prosumers; the time scale is 12:1

Figure 7. Plots of the runs for failing power modulations when only 44 over 100 and 278 over 300 overall prosumers, respectively, responded to the aggregator's issued modulation command. The dashed red line is the reference modulated power target, while the horizontal dashed black line is the power baseline.

5.1. Transaction Confirmation Time

The first technical aspect we had to confront in our DApp test runs was the unexpected delays in the duration of successful transactions' execution. A healthy blockchain should exhibit a transaction validation time no higher than five or six seconds. A healthy state is a condition in which the majority of the blockchain nodes are in agreement with each other, i.e., they share the same blocks, which, in turn, requires them to have adequate computing and network resources. However, while the transaction validation is performed locally, a transaction confirmation, indicating the inclusion of a validated transaction in a newly created block, takes variable time. The transaction processing of Ethereum-like platforms [19] is inherently competitive and relies on the gas-fee mechanism: pay more for higher processing priority. In fact, the requester of a transaction can freely offer to pay an established gas price to obtain confirmation, to push the mining nodes to process the transaction before less remunerating ones (see Figure 8). High offered gas prices (e.g., 30 GWei in Volta, with Wei the unit measure of the utility currency) can, therefore, make the network become very selective with transactions willing to pay a fairer and lower price.

```
Registering agreement Error: processing response error
(body="{\"jsonrpc\":\"2.0\",\"error\":{\"code\":-
32010,\"message\":\"FeeTooLowToCompete\"},\"id\":52}", error={"code":-32010},
requestBody="{\"method\":\"eth_sendRawTransaction\",\"params\":[\"0x02f901158301204
5047f464f070a0179368947efd4676b023d605db59680d888\"],\"id\":52,\"jsonrpc\":\"2.0\"}
", requestMethod="POST", url="http://192.168.45.65:8545", code=SERVER_ERROR,
version=web/5.6.1)
  reason: 'processing response error',
  code: 'SERVER_ERROR',
  body: '{"jsonrpc":"2.0","error":{"code":-
32010,"message":"FeeTooLowToCompete"},"id":52}',
  error: Error: FeeTooLowToCompete
    code: -32010,
    data: undefined
  },
```

Figure 8. Error message from the RPC interface of the Energy Web Volta test network, showing the problem of gas cost for the transaction, preventing it from execution, that is preventing a prosumer from registering his/her service contract with the aggregator. This is due to it being not competitive enough at that moment to compete with the load of other users racing for transactions.

In addition, the more pending (unconfirmed) transactions, the more dramatic will be the impact of this competitive mechanism. A great number of unconfirmed transactions

will tend to push the average gas fees to achieve transaction confirmation up towards higher values, leaving low paid transactions starving for hours or days (see Figure 9). The combined result is that, since the current level of congestion of a publicly open blockchain network is not controllable nor predictable, it is possible to have a transaction confirmed after a few seconds, a few minutes, or several days. This issue is discussed in more thorough detail below, in Section 5.3.

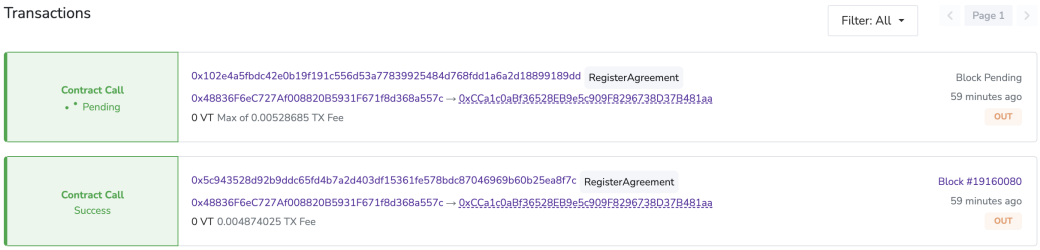


Figure 9. The occurrence of one technical issue probably due to the blockchain currently being overloaded and resulting in an issued transaction left pending even for days, waiting to be finally confirmed. In this case, it was a transaction to register a new prosumer within the aggregator service; this prosumer will then be missing in the total count of the simulation's managed prosumers.

In our prototype, we then tried to set the gas price to be always competitive, even in congested situations, thus making it unlikely that validated transactions wait excessively before their confirmation (i.e., in the order of minutes in the worst cases). In its last tested version, the blockchain integrated in the prototype responds to all modulation commands well within the limits of fifteen minutes, which has been set as the maximum accepted transient window duration in modulation service requests.

In the largest possible scenario, having 290 DERs (see Section 5.2 for the determination of this limit), the observed times to complete the initial registration of all DERs/IoTs in the set-up phase were between three to five minutes, whereas the reaction times to the modulation requests exhibited by the DERs ranged from a few seconds to about a minute or two for each DER. Figure 10 shows the developed DApp behavior at maximum DER load during the execution of a modulation command.

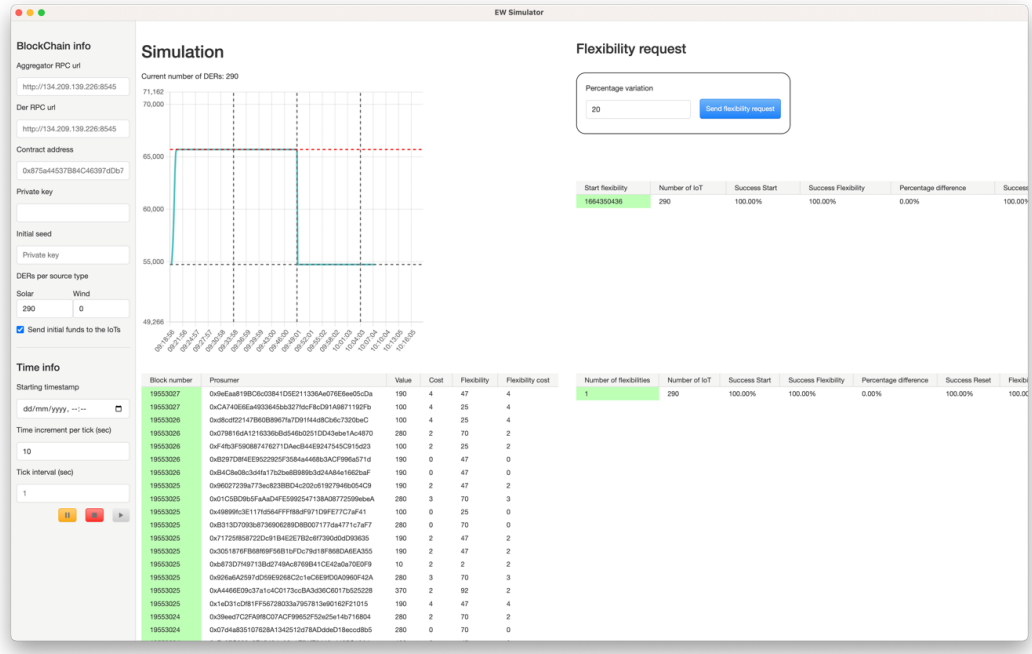


Figure 10. DER responses to a modulation request command (last accessed 28 September 2023).

5.2. Gas Limit

A second major issue we faced is that transactions may occasionally fail by exhibiting a "gas limit exceeded" error code and then be rejected. This happens when their processing cost exceeds a system-wide set threshold, i.e., the gas limit (8 M gas units for Volta), which is a prescriptive feature of Ethereum-like blockchains, introduced in the decentralized protocol for security reasons [20].

The transaction cost, in terms of gas units, cannot be determined as an exact value. It can only be estimated because it is related not only to the complexity of the transaction itself, but also to the number and size of the actual input parameters, which, in turn, can heavily affect the storage operations. Please note that several smart contract operations in the considered DApp make use of parameters, which are collections of data (i.e., the DERs list, to cite one example; see Section 4.3). In our case, the number of DERs concurrently managed in a single transaction can cause this faulty situation.

The gas limit exception represents a sturdy obstacle to scale the number of manageable DERs beyond an upper threshold (see Figure 11). This limit is due to the implementation of the smart contract, where there are some transactions that are affected by this problem. E.g., if the transaction that simulates the uploading of funds of IoTs' accounts is launched for a number of devices higher than 290, it experiences an execution fail, and consequently, the system cannot activate the IoTs. However, this will not be a problem in a real scenario (instead of a simulated one), as the prosumers will be in charge of refunding their own accounts, thus making the said transaction unnecessary.

In broader terms, the gas limit exception requires a fine-tuned design of the smart contracts, which favors as few state variables as possible in the smart contract's account storage, since the transactional costs to create, update, and delete them are far more expensive than using other storage, like the transaction logs.

Furthermore, the smart contract should not have functions that handle a large number of data structures of any kind at a time. Since the operations and data structures of the said DER administration scenario could be kept separate from the ones of the balancing scenario, splitting the aggregator's smart contract into two separately designed smart contracts could also contribute to a more effective design.

```
[WARN] aggregator - Error resetting contract Error:
reason: 'processing response error',
  code: 'SERVER_ERROR',
  body: '{"jsonrpc":"2.0","error":{"code":-32010,"message":"Transaction cost
exceeds current gas limit. Limit: 8000000, got: 8893482. Try decreasing supplied
gas."},"id":65}\n',
  error: Error: Transaction cost exceeds current gas limit. Limit: 8000000, got:
8893482. Try decreasing supplied gas.
```

Figure 11. Execution logs from the simulation tests showing an error message from the RPC interface to the Energy Web Volta test network. It explains the occurrence of a problem actually impeding the correct execution of the simulation set-up phase. Specifically, going past the 290 prosumers limit exceeds the max limit for the gas cost of one transaction, the one the simulator uses to reset its simulated IoT data registered in the ledger before restarting.

5.3. Network Congestion

Congestion conditions in the blockchain network proved to change rapidly and unpredictably, thus affecting the performance of our application in a significant way. As previously mentioned, the transaction confirmation times are highly susceptible to network overloads not only for obvious delays in the network components, but also for a demand/supply mechanism that causes the gas price to rise, thus causing the overall gas cost of transactions to follow suit. This circumstance has two major consequences: on the one hand, the transaction emitted with a lower gas price than the current average one, determined by the congestion conditions, risks not being confirmed by the mining

protocol within acceptable time frames; on the other hand, a higher transaction cost can undermine the balance of blockchain accounts or even prevent them from transacting, in case of insufficient funds (see Figure 12).

```
Registering agreement Error: timeout
(requestBody="{\"method\":\"eth_getBlockByNumber\",\"params\":[\"latest\",false],\"
id\":46,\"jsonrpc\":\"2.0\"}", requestMethod="POST", timeout=120000,
url="https://volta-rpc.energyweb.org", code=TIMEOUT, version=web/5.6.1)
  reason: 'timeout',
  code: 'TIMEOUT',
  requestBody:
'{"method":"eth_getBlockByNumber","params":["latest",false],"id":46,"jsonrpc":"2.0"
}',
  requestMethod: 'POST',
  timeout: 120000,
  url: 'https://volta-rpc.energyweb.org'
```

Figure 12. A blockchain network delay error run into a time-out and unexpectedly preventing a prosumer transaction agreement registration with the aggregator to get issued.

A massive reduction in gas price volatility and transaction costs may be introduced by permissioned blockchain technologies, where the network access can be regulated by selected nodes, so as to prevent intolerable congestion situations. Better scalability and security performances in permissioned blockchains are obviously obtained at the detriment of the decentralization dimension, as stated by the "blockchain trilemma". However, in a public blockchain, a certain decrease of decentralization can be regarded as critical, not as much in enterprise environments where, with the same decrease percentage, the system can still exhibit an elevated degree of decentralization towards the authorized participants.

Whereas the application logic would expect a rapid and sequential transaction processing, the system may experience a disordered and discontinuous response, based on the congestion conditions of the validator nodes and, in particular, in our prototype, of the RPC interface node. In the worst cases, crossing public networks can favor the occurrence of situations in which some transactions, already in the transaction pool for confirmation in a definitive block, may randomly remain "pending" for an indefinitely long time without any feedback to the sender.

To limit the above problem, it can be convenient to connect the aggregator, which generates transaction bursts by design, to the blockchain network with more reliable mechanisms, such as an internal Inter-Process Communication (IPC) channel, a private connection, or an IP tunnel (i.e., a VPN). We have observed that this actually relieves the problem, as this guarantees that the packets will be received by the blockchain network sequentially.

5.4. Overloaded RPC Node

When the computing resources of the ingress RPC blockchain node are overloaded by the incoming call flow, the default behavior of the node is dropping the excess calls, thus discarding packets containing transactions. A too-intense burst of packets can cause a very significant loss of part of them, thus compromising the system operation. The problem is technically a consequence of the lack of a synchronization mechanism in the flow of calls between the source (application front-end for the aggregator) and destination (RPC node of the blockchain), which is a single logical sequence at the application level, but is managed on the blockchain network as a sequence of independent TCP connections (see Figure 13).

Although the RPC node can be configured to increase the limit of the maximum number of accepted concurrent TCP connections, there is a hard constraint given by the actual number of parallel threads that the node can run simultaneously to open and process data on these connections. This undesired situation becomes evident as RPC request timeouts or network error responses to the RPC requests. A plausible solution could be the artificial reduction of the intensity of transaction bursts, by introducing acknowledgments and micro-delays in the application logic according to the processing resources of the RPC node, thus bringing the system back to sustainable operational conditions.

```
[WARN] aggregator - Error resetting contract Error:
requestMethod="POST", url="http://192.168.45.68:8545", code=SERVER_ERROR,
version=web/5.6.1)
  reason: 'processing response error',
  code: 'SERVER_ERROR',
  body: '{"jsonrpc":"2.0","error":{"code":-
32010,"message":"AlreadyKnown"},"id":58}',
  error: Error: AlreadyKnown
    code: -32010,
    data: undefined
  },
```

Figure 13. A blockchain network error related to sync problems between our Ethereum access node and the blockchain. The request issued had been labeled with a "nonce", which should be unique for this node, but has instead been reused, due to a previous request originally numbered with the same "nonce" being still unprocessed. This is a clear sign of the underestimated computational resources of the access node, which is not able to stay in sync with the continuously updating distributed ledger status.

A further issue with a congested node is the inability to keep up with the continuous updating of the status of the blockchain, thus losing sync and not being able to respond correctly to remote calls. When this issue occurs, it is detected as an anomalous error, which indicates that a received transaction request has been marked with the same identification number as a previous one. Having jammed the mechanism of synchronization with the global state of the blockchain, the node stops processing blocks and will no longer have a consistent internal state. One of the alarm bells of this situation is that the ordinal count of the transactions also hangs, so the same value of the "nonce" field is used for all the transactions requests, which, by definition, should instead be a unique guaranteed sequential identifier. Please note that the "nonce" is the order of acceptance of transactions originating from the same account, which, thus, must be respected for their definitive inclusion in the chain. Unfortunately, if a node goes out of sync, the only practical chance to recover from this adverse situation is to shut it and reinstall it in a better performing hardware platform.

6. Conclusions

In this study, we explored the challenges and benefits of adopting a blockchain-based decentralized architecture as a scalable, trustless interaction platform between a set of prosumers and a virtual distributor operating an energy-balance service. A prototype implementation has been realized, implementing the service via smart contracts running on the DERs' IoTs and deployed in a real blockchain network with an increasing number of simulated prosumers. This work aimed precisely to assess the suitability of the Ethereum blockchain as a scalable platform to support this decentralized service with timing requirements. We designed, implemented, and deployed the prototype DApp in order to study this research question in practice and in deep, concrete detail. We showed that the blockchain application could correctly react to energy unbalancing events of the tested amplitudes for up to a few hundred prosumers. Additionally, we observed that the load balancing completion times actually can vary from a few seconds up to a few minutes, and occasionally stall for days, as they will be limited by each IoT's availability of gas and strongly affected by unpredictable fluctuations of the transaction cost in the blockchain.

This work has highlighted how unpredictable the gas cost is, which, in turn, has a major impact on the transaction costs and on the application behavior and performance. We could observe and understand the technical reasons for the delay in the processing and storage and, therefore, the unpredictable application outcomes. The deployment of our smart contract in a public Ethereum network could then be detrimental for a real industrial deployment, where timing constraints are paramount. Hence, a real setting could make use of a large number of parties in a consortium, which provides each a blockchain node. Nevertheless, the developed smart contract and the whole application could be deployed elsewhere as a public blockchain, without hindering the work of the developers, as the

underlying Ethereum platform can be replicated in an environment where the load can be more strictly and reliably estimated.

Semi-decentralized blockchain architectures are more suited than full decentralized ones for the purposes of our considered application. In fact, they allow the load of available mining resources to be controlled and a minimum threshold of performance to always be guaranteed, so that the transaction times can be predictable. This, in turn, suggests having a closed blockchain network with the ratio between the participating DERs/IoT (light) nodes vs the infrastructure (mining) nodes as an always monitored and controlled parameter.

Author Contributions: Conceptualization, A.C., G.M., G.P. and E.T.; methodology, A.C. and E.T.; software, A.C.; validation, A.C. and E.T.; writing—original draft preparation, A.C., G.M. and E.T.; writing—review and editing, A.C. and E.T.; supervision, A.C. and E.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Generated data are given in Section 5.

Acknowledgments: The authors acknowledge the support of the University of Catania PIACERI 2020/22 Project "TEAMS".

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Teufel, S.; Teufel, B. The crowd energy concept. *J. Electron. Sci. Technol.* **2014**, *12*, 263–269.
2. IRENA. Aggregators; Innovation Landscape Brief. 2019. Available online: https://www.irena.org/-/media/Files/IRENA/Agency/Publication/2020/Jul/IRENA_Aggregators_2020.pdf (accessed on 14 February 2024).
3. Teufel, B.; Sentic, A.; Barmet, M. Blockchain energy: Blockchain in future energy systems. *J. Electron. Sci. Technol.* **2019**, *17*, 100011. [CrossRef]
4. Shi, J.; Zeng, X.; Han, R. A Blockchain-Based Decentralized Public Key Infrastructure for Information-Centric Networks. *Information* **2022**, *13*, 264. [CrossRef]
5. Xevgenis, M.; Kogias, D.G.; Karkazis, P.; Leligou, H.C.; Patrikakis, C. Application of Blockchain Technology in Dynamic Resource Management of Next Generation Networks. *Information* **2020**, *11*, 570. [CrossRef]
6. Sun, S.; Du, R.; Chen, S. A Secure and Computable Blockchain-Based Data Sharing Scheme in IoT System. *Information* **2021**, *12*, 47. [CrossRef]
7. Awan, S.M.; Azad, M.A.; Arshad, J.; Waheed, U.; Sharif, T. A Blockchain-Inspired Attribute-Based Zero-Trust Access Control Model for IoT. *Information* **2023**, *14*, 129. [CrossRef]
8. Ciatto, G.; Mariani, S.; Maffi, A.; Omicini, A. Blockchain-Based Coordination: Assessing the Expressive Power of Smart Contracts. *Information* **2020**, *11*, 52. [CrossRef]
9. Casino, F.; Dasaklis, T.K.; Patsakis, C. A systematic literature review of blockchain-based applications: Current status, classification and open issues. *Telemat. Inform.* **2019**, *36*, 55–81. [CrossRef]
10. Andoni, M.; Robu, V.; Flynn, D.; Abram, S.; Geach, D.; Jenkins, D.; McCallum, P.; Peacock, A. Blockchain technology in the energy sector: A systematic review of challenges and opportunities. *Renew. Sustain. Energy Rev.* **2019**, *100*, 143–174. [CrossRef]
11. Mollah, M.B.; Zhao, J.; Niyato, D.; Lam, K.Y.; Zhang, X.; Ghias, A.M.; Koh, L.H.; Yang, L. Blockchain for future smart grid: A comprehensive survey. *IEEE Internet Things J.* **2020**, *8*, 18–43. [CrossRef]
12. Kirli, D.; Couraud, B.; Robu, V.; Salgado-Bravo, M.; Norbu, S.; Andoni, M.; Antonopoulos, I.; Negrete-Pincetic, M.; Flynn, D.; Kiprakis, A. Smart contracts in energy systems: A systematic review of fundamental approaches and implementations. *Renew. Sustain. Energy Rev.* **2022**, *158*, 112013. [CrossRef]
13. DeCusatis, C.; Lotay, K. Secure, decentralized energy resource management using the ethereum blockchain. In Proceedings of the IEEE International Conference on Trust, Security and Privacy in Computing and Communications/International Conference on Big Data Science and Engineering (TrustCom/BigDataSE), Rotorua, New Zealand, 31 July–3 August 2018; pp. 1907–1913.
14. Yang, Q.; Wang, H.; Wang, T.; Zhang, S.; Wu, X.; Wang, H. Blockchain-based decentralized energy management platform for residential distributed energy resources in a virtual power plant. *Appl. Energy* **2021**, *294*, 117026. [CrossRef]
15. Sciumè, G.; Riva Sanseverino, E.; Gallo, P.; Augello, A.; Sciabica, G.; Tornatore, M. Blorin Blockchain Platform. In *A Practical Guide to Trading and Tracing for the Energy Blockchain*; Springer International Publishing: Cham, Switzerland, 2022; pp. 139–170.
16. Mladenov, V.; Chobanov, V.; Seritan, G.C.; Porumb, R.F.; Enache, B.A.; Vita, V.; Stănculescu, M.; Vu Van, T.; Bargiotas, D. A Flexibility Market Platform for Electricity System Operators Using Blockchain Technology. *Energies* **2022**, *15*, 539. [CrossRef]
17. Umar, A.; Kumar, D.; Ghose, T. Blockchain-based decentralized energy intra-trading with battery storage flexibility in a community microgrid system. *Appl. Energy* **2022**, *322*, 119544. [CrossRef]

18. Pop, C.; Cioara, T.; Antal, M.; Anghel, I.; Salomie, I.; Bertoncini, M. Blockchain based decentralized management of demand response programs in smart energy grids. *Sensors* **2018**, *18*, 162. [CrossRef] [PubMed]
19. Wood, G. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Proj. Yellow Pap.* **2014**, *151*, 1–32.
20. Antonopoulos, A.M.; Wood, G. *Mastering Ethereum: Building Smart Contracts and Dapps*; O'Reilly Media: Sebastopol, CA, USA, 2018.