Trabajo Fin de Máster
Máster en Ingeniería Industrial

Final Master Thesis
Master in Industrial Engineering

Design of a blockchain-based platform for peer-to-peer energy trading

Author: Manuel Sivianes Castaño
Supervisor: Carlos Bordons Alba

Dpto. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2021

Ingeniería de Sistemas y Automática

Trabajo Fin de Master
Master en Ingeniería Industrial

Design of a blockchain-based platform for peer-to-peer energy trading

Author:
Manuel Sivianes Castaño

Supervisor:
Carlos Bordons Alba
Catedrático

Dpto. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2021

Trabajo Fin de Master: Design of a blockchain-based platform for peer-to-peer energy trading

Autor: Manuel Sivianes Castaño

Supervisor: Carlos Bordons Alba

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2021

El Secretario del Tribunal

Agradecimientos

Quisiera hacer uso de estas líneas para agradecer a todos aquellos que, directa o indirectamente, me han acompañado durante esta etapa:

A mi familia y pareja, por su cariño, comprensión y apoyo permanente.

A mis amigos, por estar siempre.

A mi tutor Carlos, por brindarme tantas oportunidades, abrirme la puerta al mundo de la investigación y empujarme a que esta no sea la última sección de agradecimientos que deba escribir.

Gracias a todos,

Manuel Sivianes Castaño

Sevilla, 2021

Resumen

Este trabajo busca introducir las bases y posibles usos de la tecnología blockchain, y su implementación a través de un algoritmo de gestión de energía distribuido junto a una interfaz gráfica que aproveche al máximo las ventajas de esta tecnología. El algoritmo actúa como un programa de energía con un día de anticipación en el que cada nodo participante en la red puede realizar intercambios de energía seguros con el resto de agentes de la microrred. Blockchain hace el papel de un agregador global que verifica los intercambios de energía y evalúa la convergencia del algoritmo a través de iteraciones. El esquema propuesto se ha implementado en Ethereum y los beneficios derivados de este son comparados a través de la simulación de diferentes escenarios. La interfaz gráfica de usuario se construye usando React y, junto con web3, permite a sus usuarios interactuar con blockchain a través de Internet.

1. Introducción

Durante los últimos años la industria energética está siendo remodelada por la enorme expansión en el campo de las fuentes de energía renovables [1] y [2], dando lugar a que la producción de energía sea cada vez más dependiente del clima y, en consecuencia, más impredecible [3], y [4]. Además, debido a la reciente inclusión de los recursos energéticos distribuidos (RED), como son los paneles solares fotovoltaicos, los sistemas de almacenamiento de baterías o los vehículos eléctricos (VE), el sistema energético está experimentando una progresiva descentralización [5], y [6]. Todas estas incorporaciones hacen que la gestión centralizada sea cada vez más compleja. Estos desafíos deben ser abordados a través de la flexibilización del sistema eléctrico, otorgándole la capacidad de manejar este nuevo paradigma mientras se preserva la integridad y estabilidad del sistema [7].

En el sistema eléctrico tradicional, es obligatoria la existencia de una entidad comercializadora de energía centralizada que permita a los consumidores participar en los mercados minoristas de electricidad. El uso de un sistema comercial centralizado de este tipo se traduce en mayores costes de transacción y posibles ineficiencias. Además, estos sistemas presentan notables desventajas como la existencia de un único punto de fallo o infraestructuras costosas.

En los sistemas de energía desregulados, los agentes que utilizan RED pueden intercambiar energía gestionando su generación, almacenamiento y perfiles de consumo. Al permitir un comercio de energía entre pares es posible crear una economía impulsada entre hogares dentro de una red local. Por ejemplo, en [8], los usuarios que poseen RED intercambian energía con sus vecinos a través de una doble subasta continua habilitada por una plataforma en línea; en [9], una plataforma de comercio de energía entre pares permite a los compradores, vendedores, proveedores y operadores del sistema dirigir el comercio de energía en las redes eléctricas locales; y, en [10], se estudia el reparto de facturas, la tarifa media del mercado y una estrategia de precios basada en subastas para permitir intercambios flexibles en redes locales. En este contexto, los flujos de

energía multidireccionales están habilitados a diferencia con el comercio de energía centralizado, permitiendo a sus usuarios establecer preferencias energéticas. Sin embargo, estas novedades conducen a que sea necesario un mayor intercambio de datos e información de control entre los distintos usuarios, pudiendo implicar problemas de seguridad o privacidad.

Para hacer frente a estos desafíos y permitir la descentralización, proponemos utilizar blockchain. Con el objetivo de eliminar intermediarios y hacer posibles interacciones seguras entre pares, Satoshi Nakamoto creó blockchain junto con Bitcoin [11]. Esta tecnología reduce drásticamente la posibilidad de una violación y alteración de datos mediante el uso de criptografía y descentralización. Una red blockchain es una estructura de datos digitales compartida distribuida e inmutable, donde las transacciones entre agentes se registran permanentemente sin riesgos. Está conformado por paquetes de datos llamados bloques, cada uno de los cuales contiene múltiples transacciones. Cada bloque está vinculado criptográficamente con el anterior, excepto el primer bloque que se conoce como bloque génesis [12]. Al agregar el concepto de contratos inteligentes a las propiedades inherentes de blockchain, es posible borrar por completo la interfaz humana y, por lo tanto, a las terceras partes. En este escenario, los contratos inteligentes contienen las reglas preestablecidas para las transacciones de energía directa entre dos puntos finales en función de las preferencias de los consumidores locales, sin intermediarios [13].

Este trabajo propone una plataforma de gestión de energía que se centra en maximizar el bienestar global de sus agentes resolviendo un problema de optimización distribuido. Este problema es resuelto a través de pasos iterativos utilizando un contrato inteligente implementado en Ethereum, actuando como un agregador global que elimina la necesidad de un tercero que controle y distribuya los datos. El concepto de bienestar global se refiere a minimizar los costes financieros del mercado minorista de electricidad. Los agentes se pueden dividir en dos grupos: consumidores y prosumidores. El primero está formado por usuarios que solo realizarán pedidos a demanda en la microrred local por no poseer capacidad de generación. Por otro lado, los prosumidores son consumidores que pueden tener capacidad de almacenamiento que les permita almacenar y comercializar energía, o cualquier servicio de generación como paneles fotovoltaicos. Otros artículos que también siguen un enfoque descentralizado para la gestión de la energía utilizando blockchain son [14], [15] y [16]. En [14], se presenta un esquema basado en blockchain para el comercio de energía entre vehículos eléctricos y cargas críticas en una red lógica para satisfacer las demandas energéticas; en [15], el proceso de subasta y los precios dinámicos basados en la oferta y la demanda de energía son automatizados a través de contratos inteligentes de blockchain, y, en [16], se utiliza una blockchain de consorcio para proponer un sistema de comercio de energía que incluye un esquema de pago basado en crédito para apoyar un comercio de energía rápido.

El resto del trabajo se organiza de la siguiente manera. El Capítulo 2 presenta la tecnología blockchain. El Capítulo 3 introduce blockchain dentro del sector energético y describe el caso de estudio que este trabajo pretende cubrir. La aplicación desarrollada, así como las diferentes pruebas se presentan en los capítulos 4 y 5, respectivamente. Finalmente, las conclusiones se dan en el capítulo 6.

2. Blockchain

En este capítulo son expuestos los concetos fundamentales tras la tecnología blockchain junto a una breve introducción a su historia.

2.1 Historia de blockchain

Se muestra de manera resumida la evolución de blokchain desde el año 1991, donde Stuart Haber y W Scott Stornetta introdujeron por primera vez el término "*cadena segura de bloques*", hasta el año 2021.

2.2 Qué es blockchain

Blockchain es un tipo de estructura de datos digital, compartida, distribuida e inmutable. Contiene un registro de transacciones en constante crecimiento junto a su orden cronológico. Blockchain es, en otras palabras, un libro de contabilidad que contiene transacciones, marcas temporales y ejecutables. Estas transacciones se agrupan en bloques, cada uno de los cuales tiene una marca de tiempo y está unido criptográficamente al anterior [22]. A medida que se añaden nuevos bloques a la cadena, los bloques más antiguos son más difíciles

de modificar debido a sus vínculos criptográficos. Los nuevos bloques se distribuyen entre todos los nodos de la red, y cualquier conflicto que pueda ocurrir se resuelve utilizando un conjunto de reglas de consenso.

Blockchain funciona en redes digitales. La transmisión de datos en estas redes es similar a copiar información de un lugar a otro, es decir, en el entorno de las criptomonedas la "información copiada" son monedas digitales reales, que son transferidas entre carteras virtuales. La principal preocupación es garantizar que las monedas sólo se gasten una vez, y que no se produzca un doble gasto. La solución clásica a este reto es utilizar un punto central de autoridad, como un banco, que interpreta el papel de intermediario de confianza entre las partes que realizan las transacciones y cuya tarea es almacenar y proteger el libro de contabilidad que lleva el control del estado del sistema. En múltiples escenarios, la gestión centralizada puede ser inadecuada debido a que los costes de intermediación son demasiado elevados, además de requerir que los usuarios confíen en un tercero que controle el sistema. Además, los sistemas centralizados implican la existencia de un único punto de fallo, lo que los hace más vulnerables a problemas técnicos y ataques maliciosos [22].

El principal objetivo de Blockchain es eliminar la necesidad de terceras partes y sustituirlas por una red distribuida entre usuarios digitales que trabajen juntos para verificar las transacciones y preservar la integridad del estado del sistema. A diferencia de los sistemas centralizados, todos los participantes de la red tienen acceso al estado de ésta y a su registro inmutable de transacciones, de modo que cada miembro posee una copia del registro de todo lo que ha sucedido en la red o, al menos, puede acceder a él a través de la nube. Con este libro de contabilidad compartido, las transacciones se registran una sola vez, eliminando la duplicación de esfuerzos típica de las redes empresariales tradicionales [23]. Como resultado, cualquiera puede comprobar los registros de las transacciones y verificar su validez, lo que permite un alto nivel de transparencia. Esto plantea otra cuestión: ¿cómo encontrar una forma adecuada de consolidar y sincronizar múltiples copias del libro mayor sin una autoridad central? El proceso varía para los diferentes tipos de blockchain, sin embargo, en términos generales, los poseedores de las copias siguen un conjunto predeterminado de reglas y comparan sus versiones entre sí mediante un proceso similar al de una votación distribuida, en el que la versión que obtiene más votos de la red es considerada como la auténtica. Este proceso es repetido indefinidamente [24]. Estos mecanismos de validación se conocen como algoritmos de consenso distribuido, que se explicarán con detalle en las siguientes secciones.

Dos elementos fundamentales necesarios para proporcionar una mayor seguridad son las funciones hash y la criptografía de clave pública. Las funciones hash criptográficas son algoritmos que toman una entrada y la transforman en una salida de longitud fija, que se denomina salida hash [22]. El uso de estas funciones se basa en el hecho de que es prácticamente imposible recrear la entrada original sólo con la salida. La criptografía de clave pública, o criptografía asimétrica, es un esquema en el que un usuario posee un par de claves: una clave pública (que puede ser conocida por otros), y una clave privada (que nunca puede ser conocida por nadie más que su propietario). Estas claves están relacionadas matemáticamente de tal manera que cualquier usuario puede cifrar un mensaje utilizando la clave pública del receptor previsto, pero ese mensaje cifrado sólo podrá ser descifrado utilizando la clave privada del receptor [25]. Este proceso garantiza la autentificación: una transacción debe ser iniciada por la fuente de la que dice proceder; y la autorización: las acciones sólo pueden ser realizadas por los usuarios que tienen derecho a ello.

Para concluir esta primera aproximación a lo que es blockchain, se presentará una característica central de la misma: los contratos inteligentes. Según IBM [26], los contratos inteligentes son líneas de código que se almacenan en una blockchain y se ejecutan automáticamente una vez que se cumplen una serie de términos y condiciones requeridas. Esta funcionalidad puede ser muy beneficiosa en colaboraciones empresariales, donde los acuerdos se codifican explícitamente y todos los participantes pueden tener la certeza del resultado sin intermediarios. Esto permite a blockchain superar las limitaciones de las aplicaciones monetarias tradicionales.

2.3 Tipos de blockchain

Dependiendo del método de acceso a los datos, de cómo los usuarios pueden unirse al sistema y cómo éste funciona, blockchain se puede clasificar de la siguiente forma [27] [28]:

- Blockchain pública: estas blockchain no cuentan con permisos de acceso o restricciones: cualquier agente con conexión a Internet puede unirse y formar parte de ellas. El uso principal de este tipo de blockchain es el intercambio de criptomonedas y la minería. Mantiene la confianza entre toda la

comunidad de usuarios ya que los integrantes de la red son recompensados al trabajar juntos. Bitcoin y Ethereum son los dos ejemplos de blockchain públicas.

- Blockchain privada: es una blockchain restrictiva en la que son necesarios permisos para operar. Este tipo de blockchain se utiliza dentro de organizaciones en las que solo determinados miembros forman parte de la red. Se adapta mejor a las empresas y negocios que quieren utilizar blockchain para fines internos. Una blockchain privada está más centralizada debido a la existencia de una única autoridad que mantiene la red.

- Blockchain de consorcio: también llamada blockchain federada, es una solución para las organizaciones que necesitan tanto blockchain privadas como públicas. En lugar de un sistema abierto en el que cualquiera puede validar bloques o uno cerrado en el que sólo una entidad nombra a los productores de bloques, una arquitectura de consorcio contiene más de una organización que proporciona acceso a nodos preseleccionados para leer, escribir y auditar la blockchain. Dado que no hay una única autoridad que gobierne la blockchain, ésta mantiene una naturaleza más descentralizada en comparación con las privadas. Una blockchain de consorcio es beneficiosa en aquellos escenarios donde varias organizaciones trabajen en un mismo campo y necesiten una plataforma común en la que realizar transacciones o transmitir información.

2.4 Alcanzar consenso

Un algoritmo de consenso es un mecanismo que permite a un conjunto de usuarios o máquinas actuar de forma coordinada en un entorno distribuido. El objetivo principal es garantizar que todos los agentes del sistema se pongan de acuerdo en una única fuente de verdad, incluso si algunos agentes fallan. Para ello, dos problemas deben ser abordados: el doble gasto y el problema de los generales bizantinos [37] [38].

2.5 Algoritmos de consenso

Existen numerosos algoritmos de consenso, cada uno ofreciendo diferentes características con inherentes ventajas y desventajas. Sin embargo, hay varios requisitos que todos deben cumplir [34]:

- Acuerdo: todos los nodos honestos deben decidir el mismo valor.
- Terminación: todos los nodos honestos terminan la ejecución del proceso de consenso y llegan a una decisión.
- Validez: el valor acordado por todos los nodos honestos debe ser el mismo que el valor inicial propuesto por al menos un nodo honesto.
- Tolerancia a fallos: el algoritmo debe ser capaz de funcionar en presencia de nodos maliciosos o con fallos (BFT).
- Integridad: ningún nodo puede tomar la decisión más de una vez en un mismo ciclo de consenso.

Una vez que se tienen en cuenta estos requisitos, hay dos categorías principales de algoritmos de consenso: algoritmos de consenso basados en pruebas y algoritmos de consenso basados en votos.

2.5.1 Algoritmos de consenso basados en pruebas

En esta subsección se presentarán los algoritmos de consenso basados en pruebas. El trabajo original es Proof of Work (PoW), que fue propuesto por Satoshi Nakamoto [11]. El concepto central detrás de este tipo de algoritmos de consenso es que recompensa a los participantes que resuelven rompecabezas criptográficos con el fin de validar las transacciones y crear nuevos bloques.

2.5.2 Algoritmos de consenso basados en votos

Para implementar un algoritmo de consenso basado en votación los nodos que participen en la red deben ser conocidos para que el intercambio de mensajes sea más sencillo de realizar. Esta es la principal diferencia con los algoritmos basados en pruebas, en los que los nodos son libres de unirse o retirarse de la red de verificación. Además, todos los nodos de la red trabajan juntos para verificar las transacciones o los bloques. La comunicación entre ellos es necesaria antes de añadir un nuevo bloque a la cadena. Es habitual establecer un umbral que determinará el número mínimo de nodos que deben tener el mismo bloque propuesto para ser añadido.

2.6 Sistemas operativos blockchain

Una vez las bases de blockchain han sido presentadas se explicarán algunos de los sistemas blockchain más relevantes actualmente[1]: Bitcoin, Ethereum y Hyperledger.

3. Blockchain para intercambios energéticos

3.1 Blockchain en el sector energético

Se pueden encontrar múltiples aplicaciones de blockchain en el sector energético como la facturación a través de contratos inteligentes automatizados, plataformas de comercio descentralizadas, o respuesta a la demanda automatizada, entre otras.

En la siguiente sección se propone una plataforma de gestión energética basada en blockchain que se centra en maximizar el bienestar global de sus agentes mediante la resolución de un problema de optimización distribuido siguiendo pasos iterativos utilizando un contrato inteligente desplegado en la blockchain de Ethereum. Este contrato actúa como un agregador virtual que elimina la necesidad de un tercero que controle y distribuya los datos. El concepto de bienestar global se refiere a la minimización de los costes financieros del mercado minorista de electricidad. Los agentes pueden ser divididos en dos grupos: consumidores y prosumidores. Los primeros están formados por usuarios que sólo realizan pedidos de demanda a la microrred local al no poseer capacidad de generación. Los prosumidores son consumidores que pueden tener alguna capacidad de almacenamiento que les permita guardar e intercambiar energía, o capacidad de generación.

3.2 Modelo

La plataforma está diseñada para funcionar en una microrred en la que hay un número de prosumidores, que pueden tener acceso a paneles fotovoltaicos, vehículos eléctricos y baterías. En ella son calculados los consumos e intercambios energéticos con un día de antelación en base a las predicciones de consumo y generación fotovoltaicas, entre otras, y en la que cada hogar es considerado un nodo independiente de la red. Cada nodo puede compartir su excedente o déficit energético con el resto de los participantes utilizando blockchain. Esta información se utilizará para ajustar los intercambios de energía en la red con el fin de alcanzar el objetivo de bienestar global.

3.2.1 Mecanismo de comercio propuesto

Para la estrategia de intercambio se propone un modelo en el que cada hogar pueda intercambiar energía con cualquier agente para minimizar el esfuerzo económico global de la comunidad. Este sistema permite a sus usuarios compartir energía cuando les sobra y podría ser perdida en otro escenario, o adaptar el uso de la batería para evitar que otros hogares compren energía cuando es más cara de lo habitual. Estas premisas se recogen en la siguiente función de costes:

$$\min_{\forall\, p^g_{i,t}, p^t_{i,t}} \sum_{t=1}^{T} [C^g_{i,t}(p^g_{i,t}) + \sum_{j \neq i}^{n} c_t (p^g_{j,t} - p^t_{ij,t})] \qquad \forall\, i,$$

[1] Año 2021.

3.2.2 Algoritmo distribuido propuesto

El algoritmo distribuido para minimizar el problema de optimización (24)-(25) está compuesto por 5 pasos:

1. La primera iteración comienza con cada hogar resolviendo su problema de optimización local (14) que no tiene en cuenta al resto de la red:

$$P_i^g = \min_{\forall p_{i,t}^g} \sum_{t=1}^{T} C_{i,t}^g(p_{i,t}^g) \qquad \forall i,$$
$$s.a.\ (2)-(13),$$

   donde $P_i^g \in \mathbb{N}^{1\times T}$ contiene la energía que i necesita comprar $\forall t$ para el día siguiente, $p_{i,t}^g$.

2. La matriz de demanda global para el próximo día $P^d \in \mathbb{N}^{n\times T}$ se construye con cada P_i^g del paso anterior y debe ser conocida por todos los hogares.

3. $P^d = [P_1^g, \dots, P_i^g, \dots, P_n^g]$ es usada para configurar las restricciones superiores (17), (19) y (22) del problema de optimización (24), que es resuelto localmente:

$$P_i^t = \min_{\forall p_{i,t}^g, p_{i,t}^t} \sum_{t=1}^{T} [C_{i,t}^g(p_{i,t}^g) + \sum_{j\neq i}^{n} c_t(p_{j,t}^g - p_{ij,t}^t)] \qquad \forall i,$$
$$s.a. \quad (3)-(12), (16)-(23),$$

   donde $P_i^t \in \mathbb{N}^{n\times T}$ contiene la potencia $p_{ij,t}^t$ que el agente i pretende enviar a cada agente $\forall t$.

4. La matriz de intercambios globales $\Phi \in \mathbb{N}^{1\times T}$ es construida con cada P_i^t del paso anterior y, acto seguido, comienza un proceso de consenso para garantizar que la energía total recibida $\forall i, t$ no es superior a su correspondiente $p_{i,t}^g$ de P_i^g. Se sabe que cada intercambio individual $p_{ij,t}^t$ cumple con la restricción (22), pero la suma de todos los $p_{ij,t}^t$ puede violar esta premisa. Por ello, una vez la fase de consenso es finalizada, se calculan los intercambios comerciales confirmados $p_{ji,t}^{t_c}$ y se asegura que $\sum_{j\neq i}^{n} p_{ji,t}^{t_c} \leq p_{i,t}^g\ \forall i, t$, dando lugar a la matriz de intercambio global final Φ^c.

5. **Si** $\Phi^c - \Phi < \epsilon$, donde ϵ es la tolerancia permitida, todos los intercambios comerciales son factibles de acuerdo con el umbral establecido; **o** $iteración > \psi$, donde ψ es el máximo número de iteraciones, **el algoritmo finaliza.**

   En otro caso, todos los agentes recalculan localmente el problema de optimización del paso 1 añadiendo los intercambios comerciales de Φ^c. La energía recibida y enviada $\forall i, t$ es calculada como $\sum_j^n \Phi_{i,t,j}^c$ y $\sum_j^n \Phi_{j,t,i}^c$, respectivamente, e incluidas en (13):

$$p_{i,t}^g = p_{i,t}^l + p_{i,t}^{ev} + \sum_{j}^{n} \Phi_{j,t,i}^c - p_{i,t}^b - p_{i,t}^{pv_u} - \sum_{j}^{n} \Phi_{i,t,j}^c, \qquad \forall i, t.$$

Una vez el problema de optimización es calculado se obtiene un nuevo P_i^g que incluye todos los intercambios confirmados de Φ^c. **Seguir en paso 2**.

3.2.3 Implementación blockchain

El algoritmo distribuido de la sección 3.2.2 se ejecuta junto a una red blockchain para proporcionar una trazabilidad completa y la capacidad de auditar el proceso. Esto se consigue utilizando Ethereum, que es una blockchain pública y sin permisos que ofrece una función denominada contratos inteligentes. Ethereum permite a los usuarios crear contratos inteligentes utilizando un lenguaje de programación Turing completo.

4 Aplicación desarrollada

Como se describe en el capítulo 3, el objetivo principal de este trabajo es desarrollar una aplicación que permita a un conjunto de agentes comerciar energía libremente dentro de una microrred siguiendo un algoritmo de control distribuido cuyo agregador virtual es un contrato inteligente desplegado en la blockchain de Ethereum.

A lo largo de este capítulo se aborda el proceso de desarrollo de la aplicación comenzando por los diferentes elementos software necesarios, siguiendo por la implementación blockchain, y terminando por la interfaz gráfica de usuario que permitirá una fácil interacción entre los hogares y blockchain.

4.1 Componentes software

Remix – Ethereum IDE, Solidity, Node.js, Visual Studio Code, Web3.js, Ganache CLI, Truffle, Metamask, React, Infura.

4.2 Funciones de contrato inteligente

Las funcionalidades del agregador virtual se codifican en un contrato inteligente dentro Ganache, que es un emulador de blockchain, y se compilan utilizando Truffle. En esta sección se revisan las diferentes funciones que deben ser llamadas para cumplir con cada paso del algoritmo de consenso distribuido, excepto el algoritmo de consenso que se presenta en la Sección 3.2.3.

4.3 Interfaz gráfica de usuario

Se desarrolla una interfaz gráfica de usuario utilizando React para que los agentes puedan interactuar fácilmente con el contrato inteligente del agregador virtual. La portada es la siguiente:

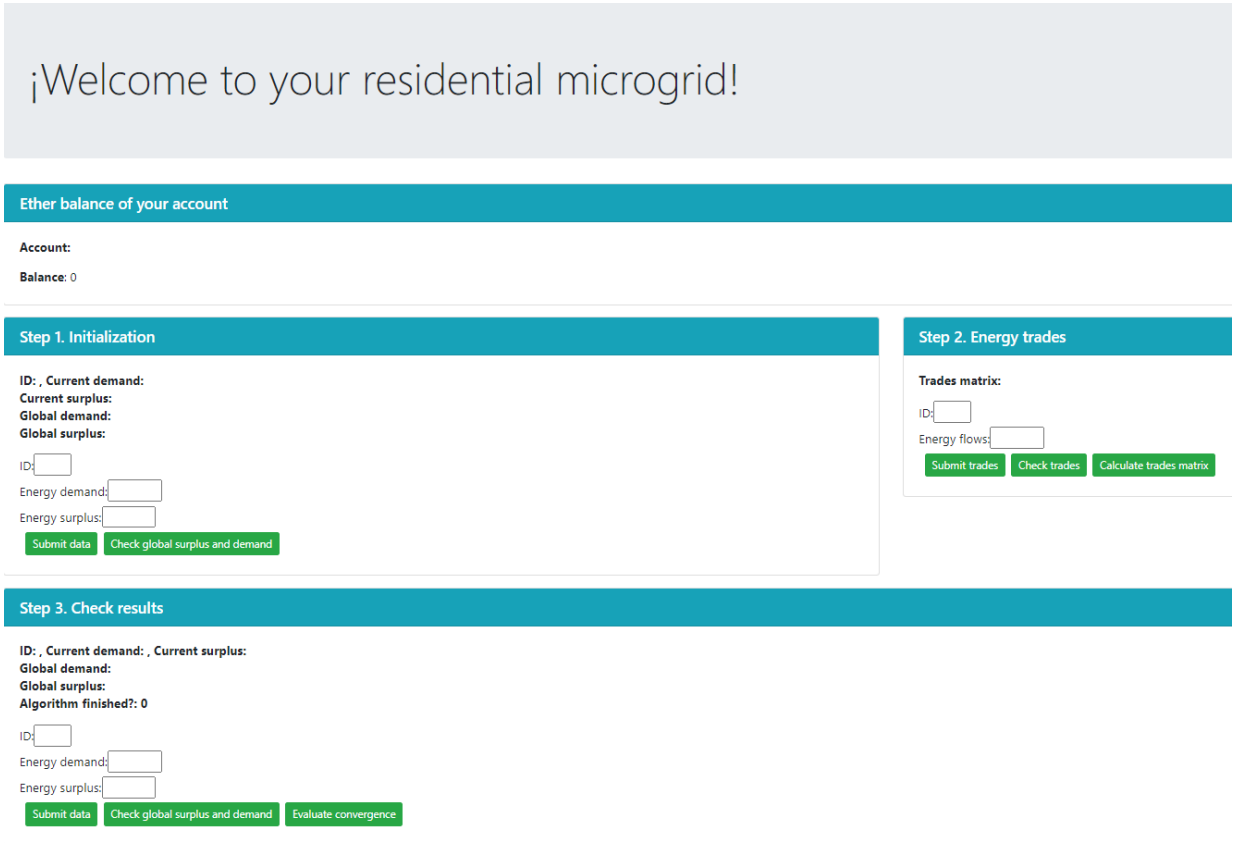

Figura 10: interfaz gráfica de usuario.

4.4 Ejemplo práctico del uso de la interfaz gráfica

Se recrea un escenario simple de tres agentes a lo largo de tres instantes. El agente 0 se define como prosumidor mientras que los agentes 1 y 2 son consumidores, lo que significa que sólo el primer agente podrá enviar energía a través de la microrred. No se darán más detalles sobre las baterías o los vehículos eléctricos ya que el objetivo de esta sección es proporcionar simplemente una visión sobre el uso de la interfaz gráfica de usuario.

5 Simulaciones y discusión

En este capítulo, se ilustran las ventajas de utilizar el algoritmo distribuido propuesto a través de blockchain mediante ejemplos numéricos. Se recrean dos escenarios para observar el impacto de utilizar una tarifa constante clásica (TC) o una tarifa de discriminación horaria (HDT).

5.1 Análisis numérico

La topología de la microrred considerada se representa en la Figura 22, donde 11 hogares forman parte de la red y 4 de ellos tienen instalaciones de generación y almacenamiento de energía. Para las cargas no controlables de los hogares, se utilizan varios perfiles de demanda con un consumo total diario de 9,55kWh, que es el consumo medio diario de energía eléctrica para un hogar español según el IDAE [65].

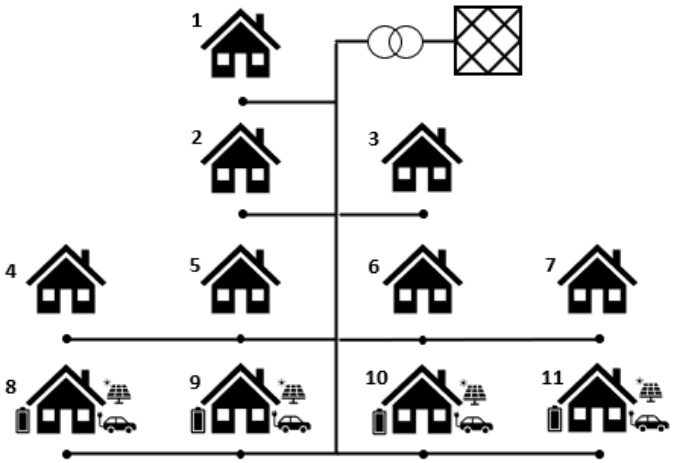

Figura 22: microrred utilizada.

Los costes de extracción de energía de la red eléctrica c_t(€/kWh) se han obtenido de [66] y se representan en la Figura 26. El primer gráfico muestra el coste horario de la tarifa distribuida por horas, en el que se aprecian claramente dos regiones de coste diferentes. Por otro lado, la segunda imagen muestra una tarifa constante clásica donde el premio del kWh oscila entre aproximadamente 0,12 y 0,15€, mientras que en la HDT varía entre 0,08 y 0,017€, lo que da lugar a una mayor flexibilidad a la hora de resolver el problema distribuido.

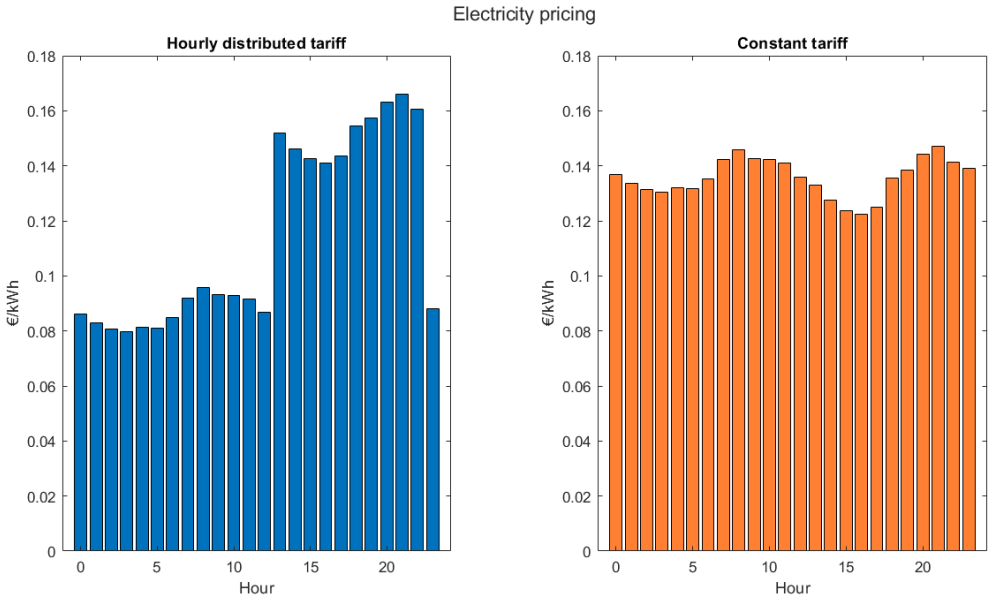

Figura 26: tarifas eléctricas.

Se establece un consumo energético diario de 7 kWh y una eficiencia de carga del 89% para los VE, y cada hogar elige libremente las horas de carga en las que es posible cargar su VE. Para la generación fotovoltaica, el perfil utilizado en este estudio se muestra en la figura 27. Las baterías de este estudio son capaces de almacenar 1250kWh y sus eficiencias de carga y descarga se fijan en el 94,5%.

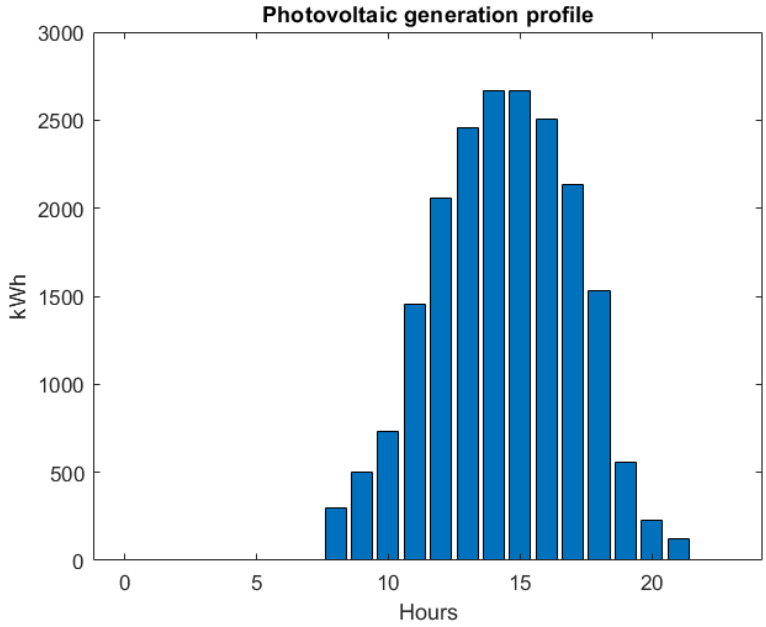

Figura 27: generación fotovoltaica.

5.2 Test 1: algoritmo distribuido utilizando la tarifa constante

En este test se simula la microrred mostrada en la Figura 22 con el algoritmo distribuido propuesto en el apartado 3.2.2 utilizando la tarifa constante de la Figura 26.

En el paso 1 se obtiene el P_i^g de cada hogar para el día siguiente resolviendo localmente el problema de optimización aislado (26), (27). Los resultados de todos los hogares se envían al contrato inteligente. En la Figura 28 y la Figura 29 se ilustran P_1^{grid} (consumidor) y P_8^{grid} (prosumidor). Se observa que la batería del agente 8 está entregando energía en aquellos instantes en los que el precio de la electricidad es más alto de lo

habitual y, viceversa, cargándola mientras es más barata.

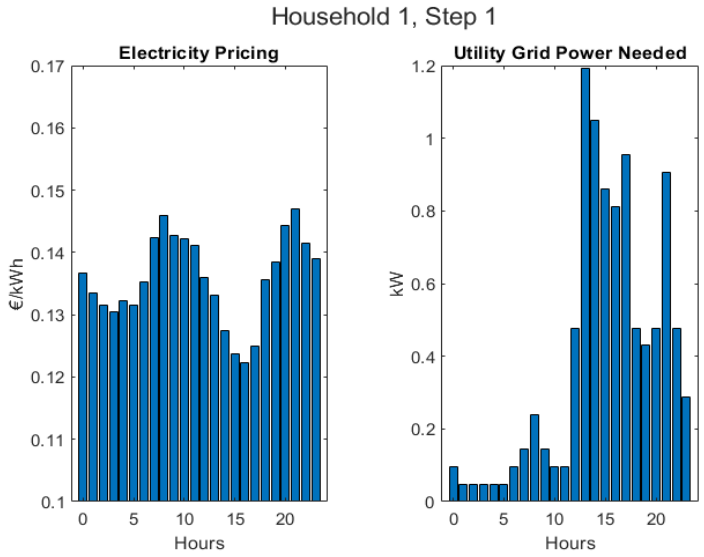

Figura 28: resultados del agente 1, primer paso, test 1.

P^d se construye dentro del contrato inteligente con los datos de la gráfica "utility grid power needed" de cada agente y se representa en la Figura 30. Obsérvese que los hogares con generación fotovoltaica e instalaciones de almacenamiento pueden mitigar sus consumos de energía en varias horas, especialmente en aquellas en las que la generación fotovoltaica es mayor.

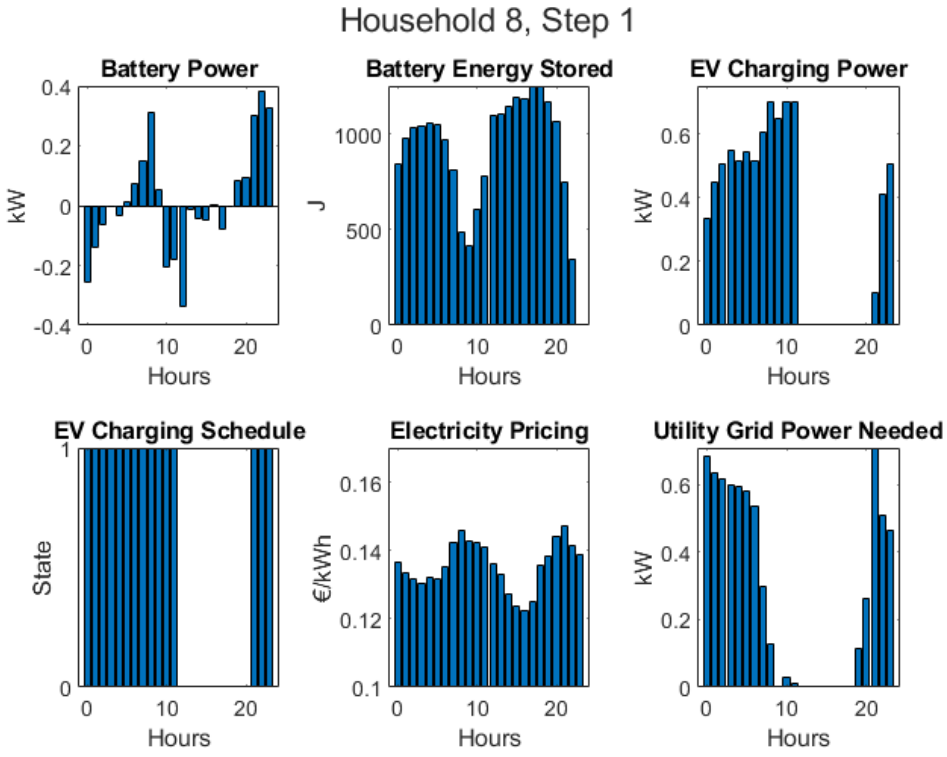

Figura 29: resultados del agente 8, primer paso, test 1.

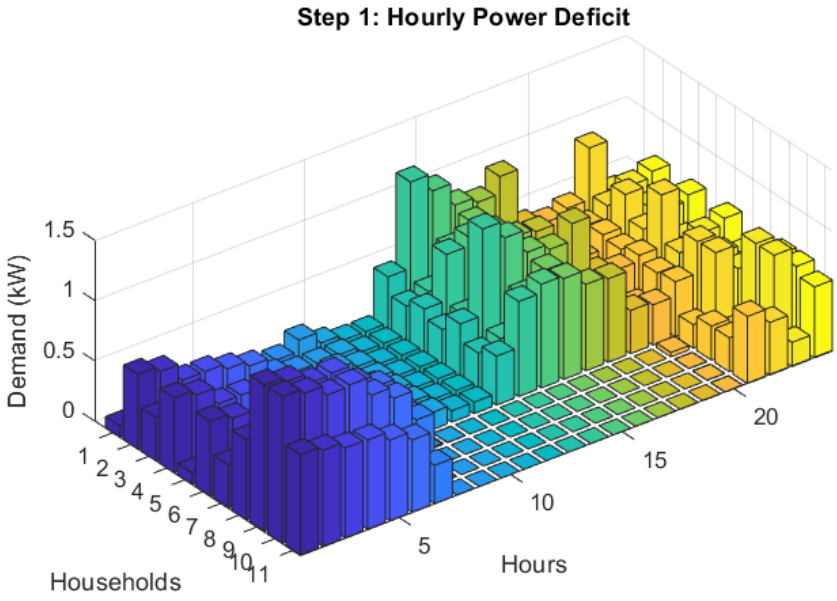

Figura 30: P^d, test 1.

El problema de optimización del paso 3 se resuelve localmente teniendo en cuenta la matriz P^d que es obtenida de blockchain. Los intercambios de energía $P_i^t \ \forall \ i,t$ son enviados al contrato inteligente. P_8^t se representa en la figura 31.

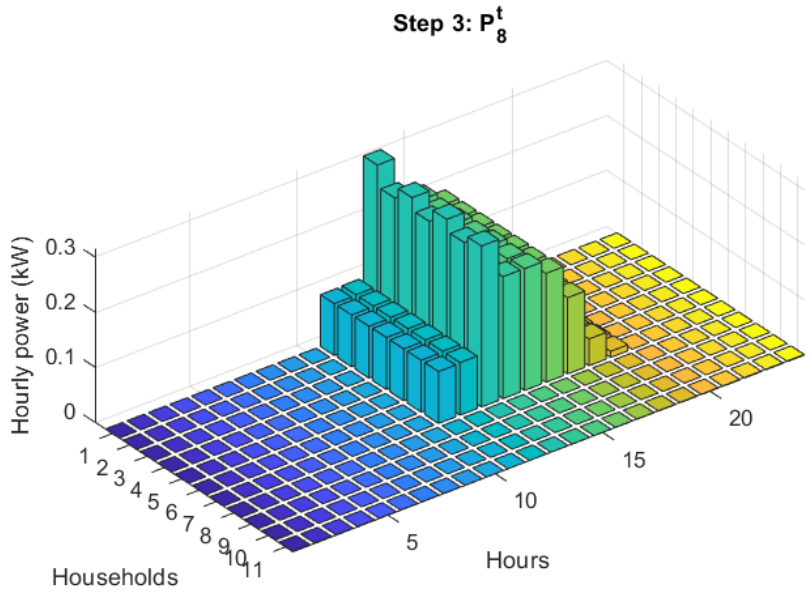

Figura 31: intercambios energéticos, paso 3, t =8, test 1.

Φ se construye dentro del contrato inteligente y la suma de energía horaria que se espera enviar $\forall i,t$ antes del consenso se calcula como $\sum_j^n \Phi_{j,t,i}$, y se representa en la Figura 32. A continuación, se ejecuta el método de consenso para calcular todos los $p_{ij}^{t_c}$ y construir Φ^c. La energía recibida tras el consenso $\sum_j^n \Phi_{j,t,i}^c$ se representa en la Figura 33.

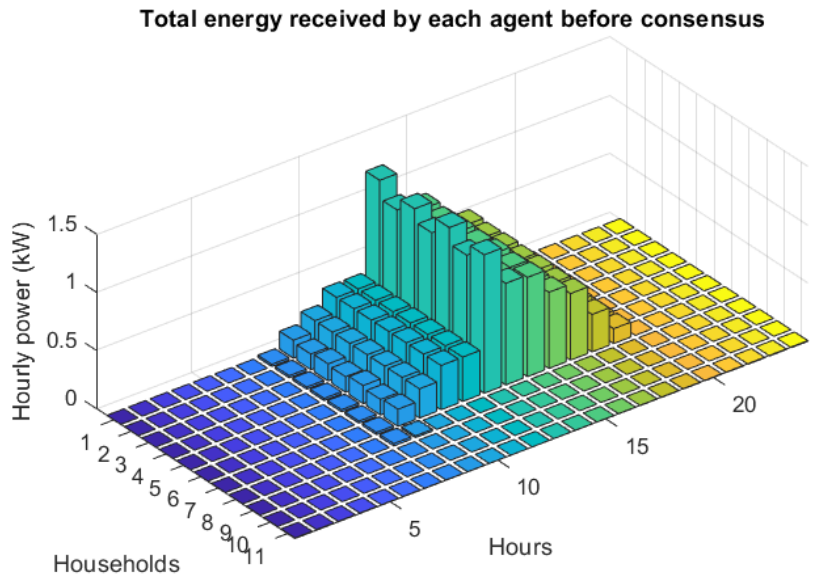

Figura 32: energía potencialmente recibida por los hogares antes del consenso, test 1.

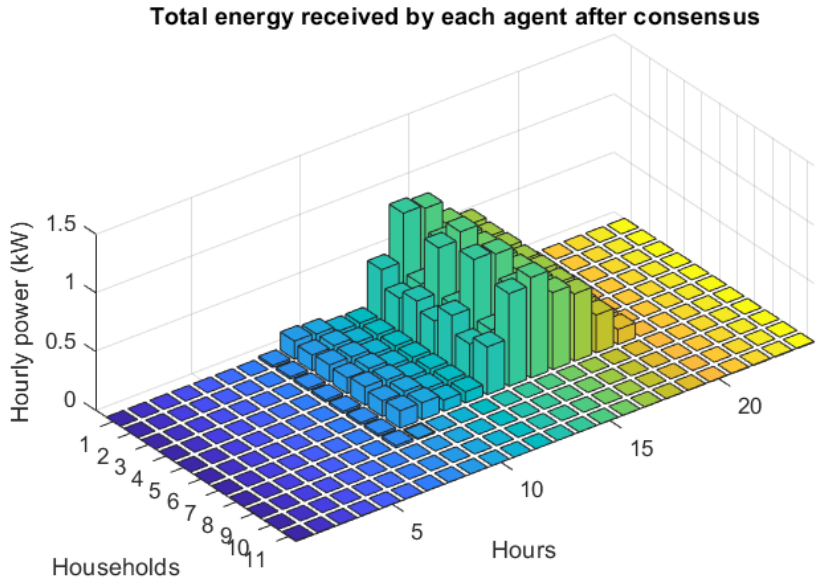

Figura 33: energía recibida por los hogares después del consenso, test 1.

Como $\Phi \neq \Phi^c$, se inicia la segunda iteración incluyendo Φ^c dentro del problema de optimización aislado. Gracias a los intercambios de energía aceptados se obtienen nuevas demandas P_i^g y se envían al contrato inteligente comenzando la siguiente iteración.

El algoritmo termina después de 4 iteraciones. Las operaciones finales aceptadas para los instantes 8 y 14 se muestran en la Figura 37 y la Figura 38, donde cada punto azul y triángulo púrpura representan a los consumidores y prosumidores, respectivamente. La evolución de los costes monetarios a lo largo de las iteraciones es recogida y mostrada en la Figura 39.

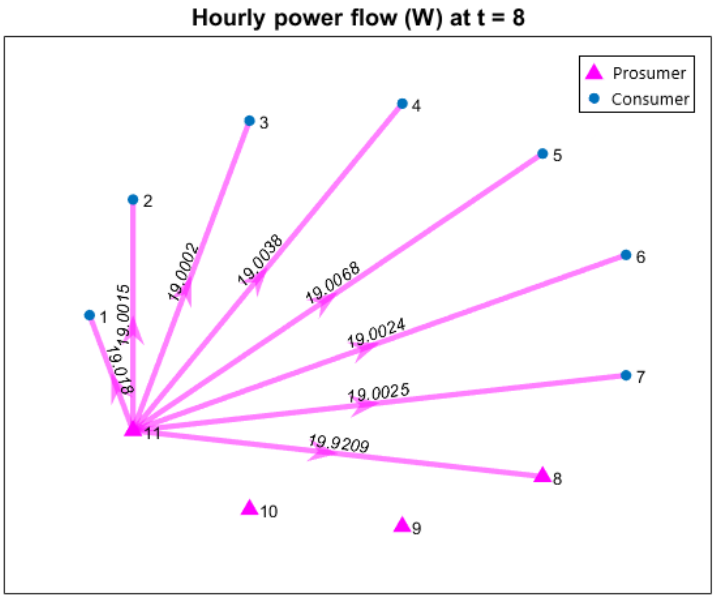

Figura 37: intercambios finales en la microrred, instante = 8, test 1.

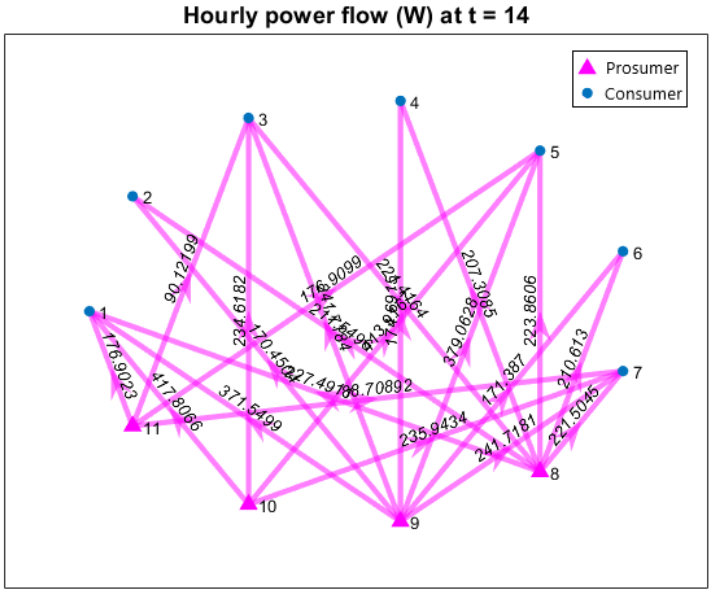

Figura 38: intercambios finales en la microrred, instante = 14, test 1.

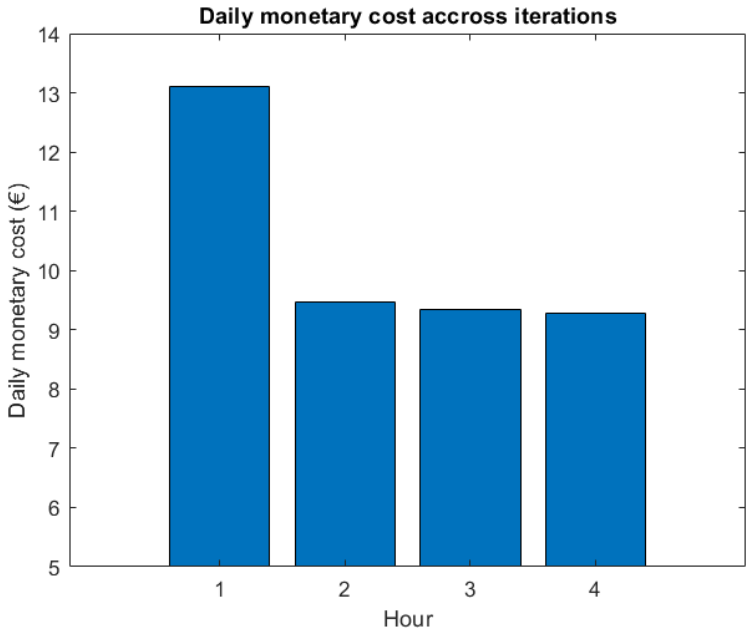

Figura 39: coste monetario en cada iteración, test 1.

5.3 Test 2: algoritmo distribuido utilizando la tarifa de discriminación horaria

En esta prueba, la tarifa eléctrica utilizada es la HDT de la figura 26 en lugar de la CT. Esta tarifa permite a los prosumidores adaptar sus cargas flexibles y baterías para minimizar no sólo su función de costes, sino también las de la microrred, ya que pueden evitar que los consumidores compren energía cuando es más cara.

El algoritmo finaliza tras 6 iteraciones. El coste monetario a través de las iteraciones se muestra en la Figura 51.

5.4 Comparación

En esta subsección se destaca el efecto positivo del uso de la plataforma propuesta comparando la función objetivo final, donde se habilita el comercio de energía entre pares, con su estado inicial, donde los hogares resuelven un problema de optimización aislado. En el test 1, donde se utiliza una tarifa CT, el coste monetario diario para la microrred derivado de la resolución del problema de optimización aislado (14) es de 13,10€, y, después de que el algoritmo finalice, se reduce un 29,08% a 9,29€, lo que corresponde a una reducción anual de 1390€. En el segundo test, en el que una tarifa HDT sustituye a la CT, el coste monetario diario inicial para la microrred es de 11,45€ y, tras 6 iteraciones, se reduce un 36,94% a 7,22€, lo que se traduce en 1544€ ahorrados anuales. Como se esperaba, el uso de la tarifa HDT conduce a unos costes monetarios inferiores ya que los prosumidores pueden adaptar sus cargas flexibles y la energía de las baterías en función a las distintas franjas de la tarifa con discriminación horaria.

6 Conclusiones

Este trabajo desarrolla una plataforma de gestión de energía distribuida dentro de una microrred que planifica los consumos e intercambios energéticos con un día de antelación con el objetivo de minimizar factura eléctrica del vecindario. Estas características se construyen en una red pública de blockchain para evitar depender de terceros, tener trazabilidad total de los datos compartidos y permitir interacciones entre pares seguras, entre otros beneficios.

Gracias al algoritmo distribuido propuesto, se reduce la factura de la red eléctrica y siempre se alcanza consenso en torno a las transacciones energéticas. En particular, cuando se utiliza la tarifa HDT, los DER pueden desplegar plenamente su potencial, ya que no sólo se utilizan para reducir la factura del usuario, sino también la de sus vecinos. Además, el uso de blockchain elimina la posibilidad de que cualquier agente

manipule el algoritmo en su propio beneficio, ya que el contrato inteligente y todas las transacciones son inmutables, lo que prueba a blockchain como una herramienta fiable para permitir la descentralización de forma transparente y segura.

En cuanto a la investigación futura, planeamos añadir una respuesta a la demanda intradiaria para abordar el posible desequilibrio entre la predicción del día anterior y la demanda de energía en vivo. Además, podría ser interesante utilizar una red de blockchain privada/híbrida para estudiar su rendimiento y viabilidad.

Abstract

This work aims to provide some insight about the blockchain technology and to propose a distributed energy management algorithm that takes full advantage of the blockchain technology through a graphic user interface. This platform serves as a one day-ahead energy schedule where each networked entity is allowed to make peer to peer (P2P) safe power trades with the rest of the microgrid agents. Blockchain serves as a global aggregator that verifies power trades and evaluate convergence across iterative steps. The proposed scheme has been implemented within Ethereum blockchain and its benefits are compared simulating different scenarios. The graphic user interface is built using React and, in conjunction with web3, allows its users to interact with blockchain through the internet.

Contents

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

The energy industry is being reshaped by a huge expansion during the past years within the field of renewable energy sources (RES) [1], and [2], which leads into energy production being more reliant on the weather, and consequently more unpredictable, as well as producing potential energy imbalances [3], and [4]. Furthermore, the energy system is becoming more decentralized due to the recent inclusion of distributed energy resources (DERs), such as photovoltaic solar panels, battery storage systems or electric vehicles (EV) [5], and [6]. All these incorporations make central management increasingly daring. These challenges must be addressed by enhancing the electricity system flexibility, granting it the ability of handling this paradigm while maintaining the system integrity and stability [7].

In the traditional electricity system, it is mandatory the existence of a centralized energy trading entity to enable consumers to take part of retail electricity markets. The usage of such centralized trading system derives in higher transaction costs and inefficiencies. In addition, this type of system presents notable disadvantages such as the existence of a single point of failure or expensive infrastructures.

In deregulated power systems, agents using DERs can potentially trade energy by handling their own generation, storage capacity and consumption profile. By enabling a peer-to-peer (P2P) energy trading, it is possible to create a share driven economy between consumers and prosumers within a local network. For example, in [8], users that own DERs trade energy with their neighbors through a continuous double auction enabled by an online platform; in [9], a P2P energy trading software platform based enables buyers, sellers, suppliers and system operators to direct energy trading in local power networks; and, in [10], where bill sharing, mid-market rate and an auction based pricing strategy are studied to allow flexible P2P trades within local customers. In this context, multidirectional flows of energy are enabled in contrast with centralized energy trading, granting users the possibility of establishing energy matching preferences. This leads into more data and control information been exchanged between end points, which can involve security or privacy issues.

To deal with these challenges, and allow decentralization, we propose to use blockchain. With the goal of removing intermediaries and making peer to peer safe interactions possible, blockchain was created alongside with Bitcoin by Satoshi Nakamoto [11]. This technology reduces drastically the possibility of a breach and alteration of data by using cryptography and decentralization. A blockchain is a digital data structure, which is shared, distributed and immutable, where transactions from two agents are recorded without any risk permanently. It is conformed of data packages, called blocks, each of which contains multiple transactions. Every block is cryptographically linked with the previous one, except the first block which is known as the genesis block [12]. By adding the concept of smart contracts to the inherent properties of blockchain, it is possible to erase completely the human interface and thus, third parties. In this scenario, the smart contracts contain the pre-established rules for direct energy transactions between two endpoints based on local consumer preferences, with no intermediaries [13].

This work proposes an energy management platform that focuses on maximizing the global wellness of their agents by solving a distributed optimization problem following iterative steps using a smart contract deployed in the Ethereum blockchain, serving as a global aggregator that erases the need of a third party controlling and distributing the data. The concept of global wellness refers to minimizing the financial costs from the electricity retail market. Agents can be divided in two groups: consumers and prosumers. The former is made

up of users that will only place demand orders into the local microgrid due to not owning any generation capacity. In the other hand, prosumers are consumers that may have some storage capacity that enables them to store and trade energy, or any sort of generation utility such as photovoltaic panels. Other articles also following a decentralized approach for energy management using a blockchain are [14], [15], and [16]. In [14], a blockchain-based scheme for energy trading between electric vehicles and critical loads in a logical network to meet temporary energy demands is presented; in [15], the bidding process and dynamic pricing based on supply and demand for energy are automated through blockchain smart contracts, and, in [16], a consortium blockchain is used to propose an energy trading system that includes a credit-based payment scheme to support fast and frequent energy trading.

The rest of the work is organized as follows. Chapter 2 presents the blockchain technology. Chapter 3 introduces blockchain within the energy sector and describes the case study that this work aims to cover. The application developed as well as different tests are presented in chapter 4 and 5, respectively. Finally, conclusions are given in chapter 6.

2 BLOCKCHAIN

In this chapter the fundamentals concepts beyond the blockchain technology are covered, as well as a brief introduction to its history.

2.1 The History of Blockchain

Blockchain term emerged in 2008, inside Bitcoin project but it was much earlier when the founding concepts were described. It was in 1991 when Stuart Haber and W Scott Stornetta described a cryptographically secured chain of blocks for the first time [17].

The concept of "smart contracts" was created by Szabo with the objective of improving the electronic commerce protocols between untrusted agents on the Internet. It was in 1998 when Szabo developed a system for a decentralized digital currency that he called "bit gold", although it was never implemented [18].

In 2000, Stefan Konst published his theory of cryptographically secured chains and suggested a set of ideas for its implementation. A few years later, in 2008, a developer or a group of developers that worked under the pseudonym Satoshi Nakamoto, published a white paper stating the model for a blockchain. It was here when he proposed a "purely peer-to-peer version of electronic cash that would allow online payments to be sent directly from one party to another without going through a financial institution" [11]. Its (or their) real identity remains unknown. The design was implemented the following year by Nakamoto as a core component of the cryptocurrency, nowadays well known, Bitcoin [19]. The first transaction in Bitcoin took place in the 170^{th} block, opening the door for the first real-world transaction, which happened on 22 May 2010, two pizzas were bought for 10 thousand BTC[2], becoming the first real world transaction ever made [20].

Four years later, in 2014, the blockchain technology splits from the currency and Blockchain 2.0 was born, opening a new horizon of applications beyond currency. This upgrade is achieved by the implementation of a programming language that allows any user to develop more complex smart contracts, that could potentially create automatic payments when a shipment arrives or distribute certificates that automatically send dividends to their owners if certain conditions are met.

The evolution of the daily number of transactions for the most relevant blockchains during the last years is shown in Figure 1.

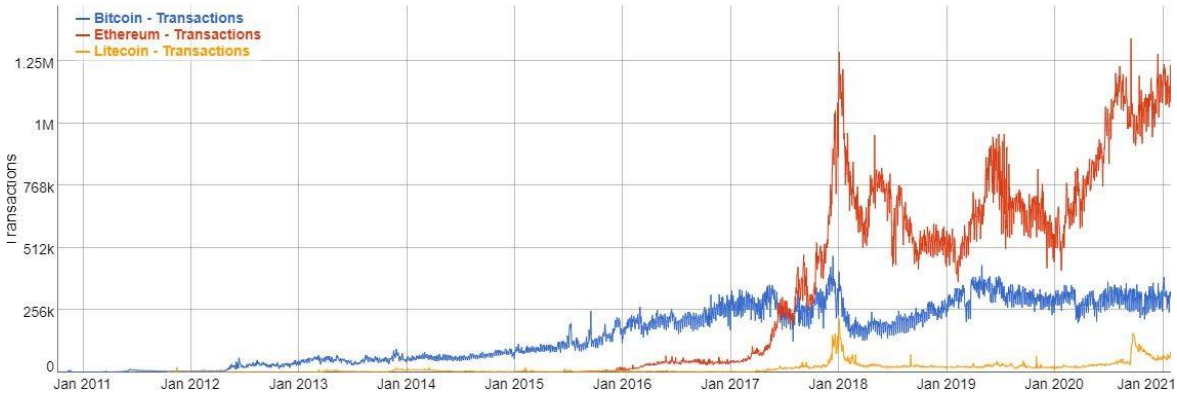

Figure 1: Bitcoin, Ethereum and Litecoin transactions per day (January 2011 - January 2021) [21].

[2] 10,000 BTC = 399,170,000 $ on 21 May 2021.

2.2 What is blockchain

A blockchain is a type of digital data structure, which is shared, distributed and immutable. It contains an ever-growing log of transactions and their chronological order. A blockchain is, in other words, a ledger that contains transactions, timestamps and executable. These transactions are grouped into blocks, with each one having a time stamp and being cryptographically bonded to the previous one, forming the blockchain itself [22]. When new blocks are appended to the chain, older blocks become more difficult to be modified due to their cryptographic links. The brand-new blocks are distributed across copies of the ledger in the network, and any conflicts that could possibly happen are solved by using an established set of rules.

Blockchain works on digital networks. Transmission of data in these networks is similar to copying information from one place to the other, i.e., in the cryptocurrency environment the "information copied" are actual digital coins, which are transferred from one user's electronic wallet to another's. The principal concern is ensuring that coins are only spent once, and no double-spending is occurring. The classical solution towards this challenge is using a central point of authority, such as a bank, who interprets the role of trusted intermediary between transacting parties and whose task is to store and protect the ledger that keeps track of the system state. In multiple scenarios, central management may be inadequate due to intermediary costs being too high and requiring users to trust a third party to control the system. Also, centralised systems imply the existence of a single point of failure, which make them more vulnerable to technical problems and malicious attacks [22].

Blockchain's main objective is to erase the need of third parties and replace them with a network distributed among digital users who work together to verify transactions and preserve the integrity of the ledger. In contrast with centralised systems, all network participants have access to the distributed ledger and its immutable record of transactions, so every member holds a copy of the whole ledger or, at least, it can be accessed through the open cloud. With this shared ledger, transactions are recorded only once, eliminating the duplication of effort that is typical of traditional business networks [23]. As a result, anyone can check the transactions logs and verify their validity, enabling a high level of transparency. This arises another question: how to find a proper way to consolidate and synchronise multiple copies of the ledger without a central authority? The process varies for different types of blockchain, however in general terms, copyholders follow a predetermined set of database management rules and compare their versions together using a process similar to distributed voting, in which the version that gets the most votes from the network is considered as the authentic, being this process repeated indefinitely [24]. These validations mechanisms are known as distributed consensus algorithms, which will be thoroughly explained in the following sections.

Two crucial core elements needed to provide an enhanced security are hash functions and public-key cryptography. Cryptographic hash functions are algorithms that take an input and transform it into an output of a fixed length, which is called the hash output [22]. The usage of these functions relies on the fact that it is barely impossible to recreate the original input given the output alone. Public-key cryptography, or asymmetric cryptography, is a scheme in which a user holds a pair of keys: a *public key* (which can be known to others), and a *private key* (which can never be known by anyone but its owner). These keys are mathematically related in such way that any user can encrypt a message using the intended receiver's *public key*, but that encrypted message will only be decrypted using the receiver's *private key* [25]. This process ensures authentication: a transaction must be initiated by the source it claims to be from; and authorisation: actions can only be performed by users who have the right to.

To conclude this first approach to *what is blockchain*, it will be presented a core feature of it: smart contracts. According to IBM [26], smart contracts are lines of code that are stored on a blockchain and automatically execute once a set of required terms and conditions are met. This functionality can be highly beneficial in business collaborations, where agreements are explicitly coded, and all participants can have certainty of the outcome with no intermediary. This enables blockchain to transcend the limitations of currency applications.

2.3 Types of blockchain

Depending on how data can be accessed, how users can join the system and how it operates, blockchain can be categorised as below [27] [28]:

- Public blockchain: these blockchains are permisionless, non-restrictive, distributed ledger system where any agent with internet connection can join them and become a part of it. The main use of this genre of blockchains is for exchanging cryptocurrencies and mining. It maintains trust among the whole community of users as everyone in the network feels incentivized to work towards the improvement of the public network. Bitcoin and Ethereum are the two examples of public blockchains.
- Private blockchain: it is a restrictive blockchain where permissions are needed in order to operate within a closed network. This kind of blockchain is used within an organization in which only particular members take part of the network. It fits better for enterprises and business that want to use blockchain for internal purposes. A private blockchain is more centralized due to the existence of a single authority maintaining the network.
- Consortium blockchain: also called federated blockchain, is a solution for organizations where both private and public blockchains are needed. Instead of an open system where anyone can validate blocks or a closed one where only a single entity appoints block producers, a consortium architecture contains more than one organization providing access to pre-selected nodes for reading, writing and auditing the blockchain. Since there is no single authority governing the blockchain it maintains a more decentralized nature compared with private ones. A consortium blockchain would be highly beneficial in a scenario where multiple organizations work in the same field and need a common platform on which to carry out transactions or relay information.

Blockchain type	**Public**	**Private**	**Consortium**
Permissionless	Yes	No	No
Consensus determination	The set of miners	One organization	A set of nodes
Immutability	Almost impossible to tamper	Could be tampered	Could be tampered
Read permission	Anyone	Invited users only	Depends
Write permission	Anyone	Approved participants	Approved participants
Centralized	No	Yes	Partial
Participants known	No	Yes	Yes
Transaction speed	Slow	Fast	Fast

Table 1: different blockchain properties comparison.

2.4 Blockchain components

In this subsection the main components of blockchain are introduced, which will enable the reader to further understand this technology.

2.4.1 Cryptographic Hash Functions

The usage of cryptographic hash functions (CHF) is a crucial process in blockchain technology. These functions consist of algorithms that take an arbitrary amount of data input and produce a relatively unique fixed-sized output called hash or message digest. Any user can take an input data, hash it, and obtain the exact same result every time (if the input data has not changed). Even the smallest change to the input (e.g., modify one bit) will result in a completely different output, as shown in Figure 2. A CHF is a one-way function, which means that it is practically impossible to obtain an input given its output.

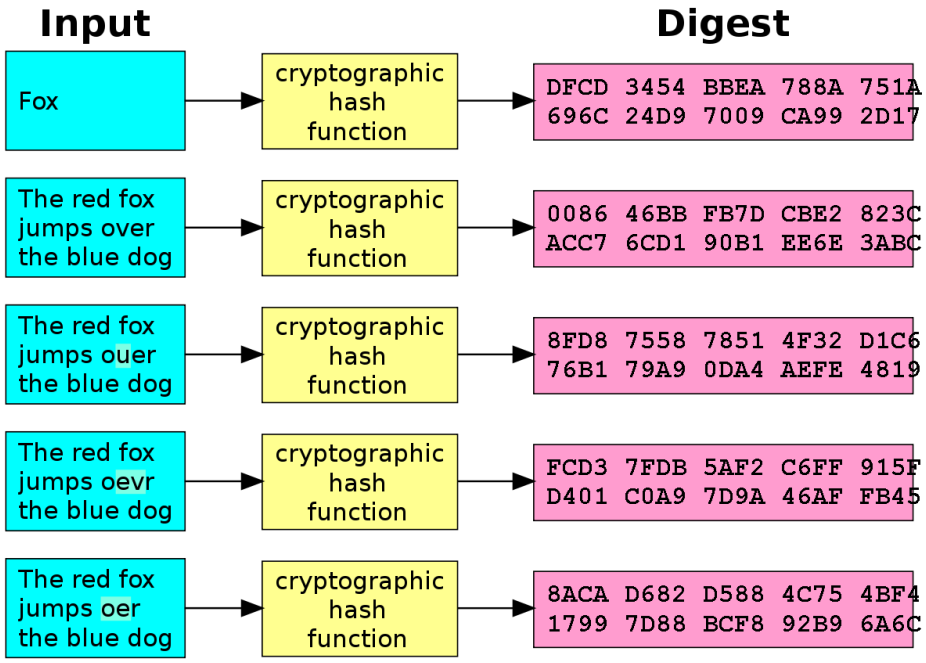

Figure 2: cryptographic hash function [29]

The cryptographic hash function presents the following main properties [30]:

- A given input to the function always results in the same hash.
- It is quick to compute the hash value for any given message.
- They are *preimage resistant*. This means that is infeasible to reverse the process that generated a given hash value (e.g., obtain *x* given *h*, being *h=hash(x)*).
- They are *second pre-image resistant.* It is infeasible to find different messages with the same message hash value.
- They are collision resistant. Any minimum change to a message should result in a totally uncorrelated new hash value (avalanche effect[3]).

One of the most cryptographic hash functions used in blockchain implementations is the Secure Hash Algorithm (SHA) with an output size of 256 bits (SHA-256). This algorithm has an output of 32 bytes which is normally displayed as a 64-character hexadecimal string, which means that there are 2^{256}or *115,792,089,237,316,195,423,570,985,008,687,907,853,269,984,665,640,564,039,457,584,007,913,129,639,936* possible hash values [31].

[3] In cryptography, the avalanche effect is the desirable property of cryptographic algorithms, typically cryptographic hash functions, where if an input is changed slightly (for example, flipping a single bit), the output changes significantly (e.g., half the output bits flip) [68].

2.4.1.1 Cryptographic Nonce

In security engineering, nonce is an abbreviation of number used once. It is usually a random or pseudo-random number provided in an authentication protocol which ensures that old communications cannot be reused in replay attacks[4] [32]. Nonces are used in Proof of Work protocol (see 2.6.1.1) to produce different hash outputs without changing the data:

$$hash(data + nonce) = message\ digest$$

2.4.2 Public-Key Cryptography

As introduced in 2.2, public-key cryptography, or asymmetric cryptography, is a cryptographic system where every user owns a private and a public key. The generation of such key pairs depends on cryptographic algorithms which are based on one-way functions. These keys are mathematically related without reducing the security of the system: the private key cannot efficiently be determined based on the public key. While the public key can be freely distributed without risking security, it is mandatory to preserve secretly the private key.

In this system, any user can encrypt a message using his public key, but that encrypted message can only be decrypted with the receiver's private key, as can be seen on Figure 3:

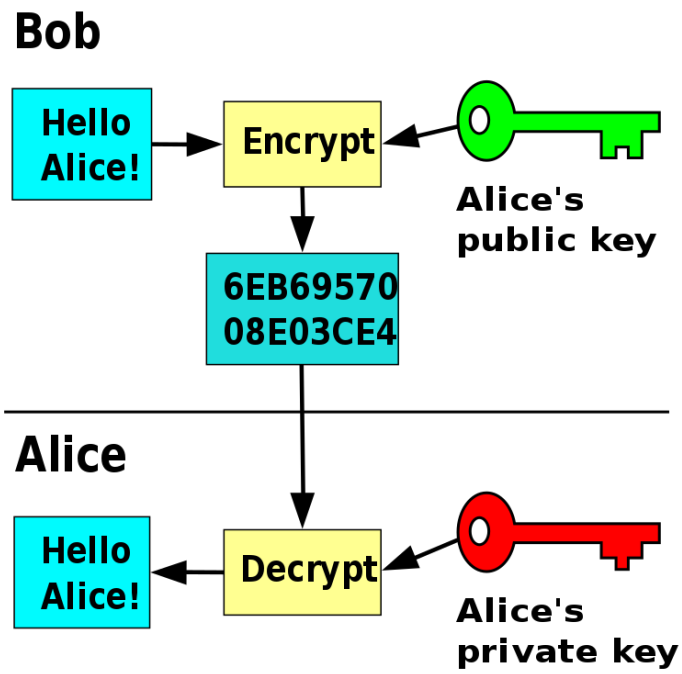

Figure 3: public-key cryptography [33]

This process enables users to maintain trusted relationships with untrusted nodes. Blockchain makes extensive use of public key cryptography. Here it is shown an example of how useful this system is:

User 'A' decides to pay user 'B' one bitcoin. This can be done with A's hardware wallet or any software wallet like Metamask, using A's private key. This key is stored in A's device and it will never leave it. This means that, if 'A' wants to send a transaction to 'B', both A's private key and B's public key are needed. A's public key is then used to verify whether that message actually did come from the wallet that says it did. In

[4] An attack on a security protocol using a replay of messages from a different context into the intended (or original and expected) context, thereby fooling the honest participant(s) into thinking they have successfully completed the protocol run [71].

contrast to fiat currency, if 'A' happens to lose his private key, his money is not lost. All this money is recorded on the blockchain and, as long as 'A' can recover this key somehow[5], he will be able to get access and continue trading.

2.4.3 Address

Addresses are unique identifiers used within a blockchain transaction to determine senders and recipients. A common pattern to obtain an address is deriving it from a public key using a one-way function. Addresses are not secret and are shorter than public keys.

Depending on the type of blockchain implementation the process of deriving an address may be different. For permissionless blockchain networks, where anonymous account creation is allowed, a blockchain network user can generate as many asymmetric-key pairs, and therefore addresses as desired, enabling a varying degree of pseudo-anonymity.

Blockchain network users are not the only source of addresses in blockchain networks. An address it is also required to interact with any smart contract deployed within the network. For instance, in Ethereum it exists a special address for this case termed *contract account*. This address is created automatically once a smart contract is deployed and it allows for the contract to be executed whenever a transaction is received [31].

2.4.4 Transactions

Transactions constitute the fundamental and smallest building block of a blockchain system. It represents a transfer of value between two addresses. The transaction data structure is shown in Table 2: transaction structureTable 2 and an example of a verified transaction output is given in Figure 4.

Field	**Size**	**Information**
Version number	4 bytes	It specifies the rules to be used by the miners and nodes for transaction processing.
Input counter	1-9 bytes	The number of inputs contained in the transaction.
List of inputs	Variable	Each input is composed of multiple fields. The first transaction in a block is called *coinbase transaction*. It specifies one or more transaction inputs.
Output counter	1-9 bytes	Represents the number of outputs.
List of outputs	Variable	Outputs within the transaction.
Lock time	4 bytes	It determines the earliest time when a transaction becomes valid.

Table 2: transaction structure [34].

[5] If a user loses completely a private key, any digital asset bonded with it is virtually lost due to how computationally infeasible is to regenerate the same private key. The assets will still be recorded in the blockchain, but unless the user manages to find his private key, they are completely locked.

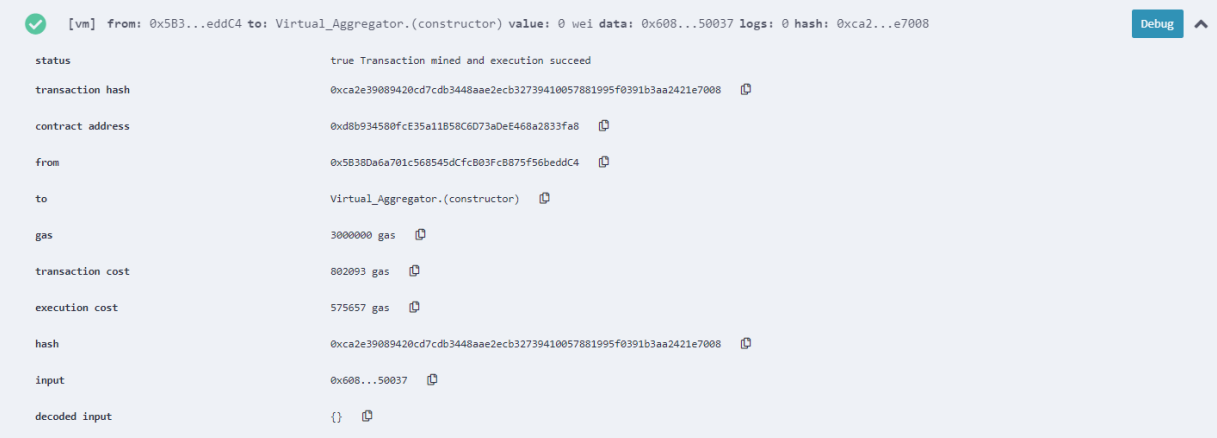

Figure 4: transaction data structure from Remix.

2.4.5 Blocks

Each block contains several verified transactions, and they are cryptographically related with each previous block of the chain. In the next table the structure of a block is presented:

Field	**Size**	**Information**
Block size	4 bytes	Represents the size of the block.
Block header	80 bytes	It includes multiple fields that will be explained in Table 4.
Transaction counter	1-9 bytes	It contains the total number of transactions in the block.
Transactions	Variable	Every transaction included in the block.

Table 3: block structure [34].

2.4.5.1 Block header

The block header is a block feature that acts as a summary of the whole block. It is built with all the metadata contained in the block, including the Merkle[6] root of the new transactions added, and the nonce, among others [35]. Each block includes a unique header, whose structure is shown in Table 4 [36]:

Field	Size	Description
Version	4 bytes	It determines the block validation rules that must be followed.
Previous block's header hash	32 bytes	Double SHA-256 hash from the previous block's header.
Merkle root hash	32 bytes	Double SHA-256 hash of the Merkle tree of every transaction within the block.
Timestamp	4 bytes	It contains the time when the miner started mining the header.

[6] The Merkle Hash Tree is a tree-based data structure in which every non-leaf node is labelled with the cryptographic hash of the labels of its child nodes, and every leaf node is labelled with the hash of a data set. This scheme provides an efficient way of verifying the content of large data sets since all data can be related with the Merkle root (the base of the tree) [72].

Difficulty target	4 bytes	Current difficulty target of the network/block.
Nonce	4 bytes	Random number that miners need to keep changing to produce a hash lower than the difficulty target.

Table 4: block header structure.

2.4.5.2 Blocks interconnection

In the following image it is described the blockchain archetype, where every block is linked with its previous one through the hash. Any change in any transaction would lead to a new hash, which will be notified and discarded by the rest of nodes:

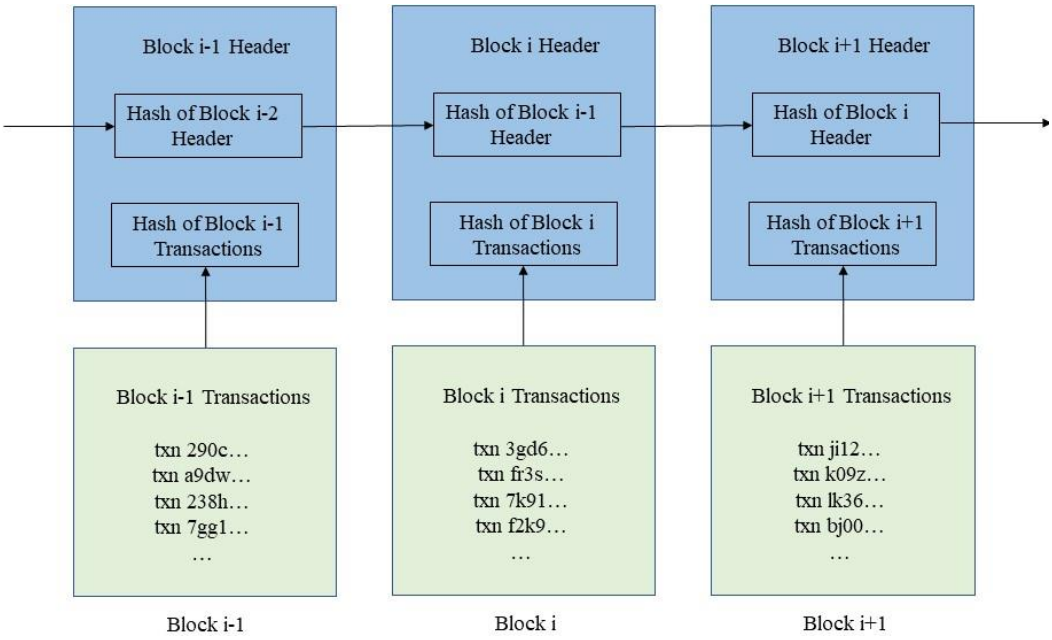

Figure 5: blocks interconnection.

2.5 Reaching agreement

A consensus algorithm is a mechanism that allow a set of users or machines to act coordinately in a distributed setting. The main goal is to ensure that all agents in the system agree on a single source of truth, even if some agents fail. Two problems must be addressed: double spending and Byzantine Generals Problem [37] [38].

2.5.1 Double spending problem

Double spending is the risk of a digital currency being spent twice. It is a potential problem unique to digital currencies because digital information can be relatively easy reproduced. Traditional currency does not face this issue due to its physical condition, making them difficult to replicate. This problem is traditionally fixed in the Internet through centralized trusted institution. In order to escape from this centralized approach while avoiding the double spending problem, blockchain technology incorporates consensus algorithms where every single transaction is verified by several distributed nodes before being accepted.

2.5.2 Byzantine Generals Problem

Byzantine Generals Problem was conceived in 1982 as a dilemma in which a group of Byzantine generals could have communication problems when trying to agree on their next action. The dilemma assumes that each general owns an army and that each group is located in different locations, being necessary the usage of a

courier to communicate between each other. The generals have to agree on either attacking or retreating. It does not matter whether they decide the first or the second, as long as all generals reach consensus.

Therefore, the following requirements are found:

- Each general has to decide to attack or to retreat.
- After the decision is made, it cannot be changed.
- All generals have to reach an agreement and execute it in a synchronized manner.

As stated before, a courier es needed for each general, consequently, the core challenge of the Byzantine General's Problem is that the messages can get somehow delayed, destroyed or lost. In addition, even if the courier reaches its destination safely, there is a chance of one or more generals to act maliciously and send a fraudulent message to confuse the other generals, leading to a failure.

Transposing this dilemma into blockchain, each general is featured by a network node, and these nodes are responsible of reaching consensus on the current state of the system. This means that, in order to reach consensus within these distributed systems, it is necessary to exist at least ⅔ or more trustworthy network nodes. On the other hand, if the majority of the network decides to act maliciously, the system is susceptible to failures and attacks (such as the 51% attack that will be covered in section 2.6.1.1).

As a result, it is named Byzantine fault tolerance (BFT) as the property of a system that is able to resist the class of failures derived from the Byzantine General's Problem, which means that a BFT system is able to continue operating even if some of its participants act maliciously [39].

2.6 Consensus algorithms

Some of the issues that must be addressed have been presented. There is an enlarging amount of distributed consensus algorithms being developed, each one providing different features with inherent advantages and disadvantages.

There are various requirements that must be accomplished [34]:

- Agreement: all honest nodes must decide on the same value.
- Termination: all honest nodes terminate execution of the consensus process and eventually reach a decision.
- Validity: the value agreed by all honest nodes must be the same as the initial value proposed by at least one honest node.
- Fault tolerant: the algorithm must be able to work in the presence of malicious or faulty nodes (BFT).
- Integrity: no node can make the decision more than once in a single consensus cycle.

Once these requirements are taken into account, two main consensus algorithms categories can be found: proof-based and voted-based consensus algorithms.

2.6.1 Proof-Based Consensus Algorithm

In this subsection it will be introduced the proof-based consensus algorithms. The original work is Proof of Work (PoW), which was proposed by Satoshi Nakamoto [11]. The core concept behind this kind of consensus algorithms is that they reward participants who solve cryptographic puzzles in order to validate transactions and create new blocks: in other words, among many nodes joining the network, the node that performs sufficient proof will earn the right to append a new block to the chain, and thus, receive the reward.

2.6.1.1 Proof of Work

A proof of work is a piece of data which is difficult (costly, time-consuming) to produce but easy for others to be verified and which satisfies certain requirements. To produce a proof of work it is typically needed to invest a reasonable amount of resources in a process of trial and error. Bitcoin uses the Hashcash proof of work

system, which was proposed in 1997 by Adam Back. Hashcash is a cryptographic proof of work algorithm based on the usage of hashes, which need a certain amount of work in order to be computed, but its output can be verified in an very efficient manner. An example of this can be found in the email context, where an encoded hashcash stamp can be added to the email's header to ensure that the remitter has spent some CPU time computing the stamp before sending the message. By applying this process, spammers will not be able to send large numbers of emails since it is highly time/cost demanding. Receivers can check if the remittent made such investment filtering the emails' headers [40].

Bitcoin uses hashcash proofs of work within the block generation process. Before a block is accepted by network agents, miners need to fulfill a proof of work that contains all the information stored in the block. The rate at which blocks are generated depends on the target difficulty, which can be adjusted in order to maintain constant (one block is accepted every 10 minutes). The high number of miners trying to succeed in the block generation process makes highly unpredictable to guarantee which miner will win the race in yielding the next block.

For a block to be valid its hash must be a value less than the current target, as shown in the following expression:

$$H(N \,||\, P_{hash} \,||\, Tx \,||\, Tx \,||\, \ldots \,||\, Tx) < Target$$

Where N is a nonce, P_{hash} is the hash of the previous block, Tx represents transactions in the block, and *Target* is the current target difficult value [34]. New blocks contain the hash of the preceding one, which creates a chain of cryptographically linked blocks that together involve a huge amount of work. Modifying a block, which can only be fulfilled by creating a new block that presents the same predecessor, involves regenerating every following block and hashing all data they contain. This property makes blockchain tamper proof [41].

The mining algorithm in Bitcoin consists of the following steps [34]:

1. Every miner retrieves the previous block's header from the Bitcoin network.
2. A set of broadcasted transactions on the network are assembled into a block to be proposed.
3. Compute the double hash[7] of the previous block's header combined with a nonce and the newly proposed block using the SHA-256 algorithm.
4. Verify if the new hash is lower than the target. If this is true, PoW is solved and the discovered block is broadcasted to the network. Miners fetch their rewards.
5. If the new hash is greater than the target the process is repeated after incrementing the nonce.

[7] Double hashing is used to safeguard against birthday attacks. A birthday attack is a scenario where an attacker is able to produce the same hash by using a completely different input (called a collision). With the SHA-256 function, the probability of this attack happening is infinitely small. However, other hash functions have been "broken" in the past. In order to safeguard against this happening to SHA-256 in the future (and effectively breaking the security model of Bitcoin) it's best to hash the hash. This halves the probability of a collision occurring, making the protocol that much more secure [67].

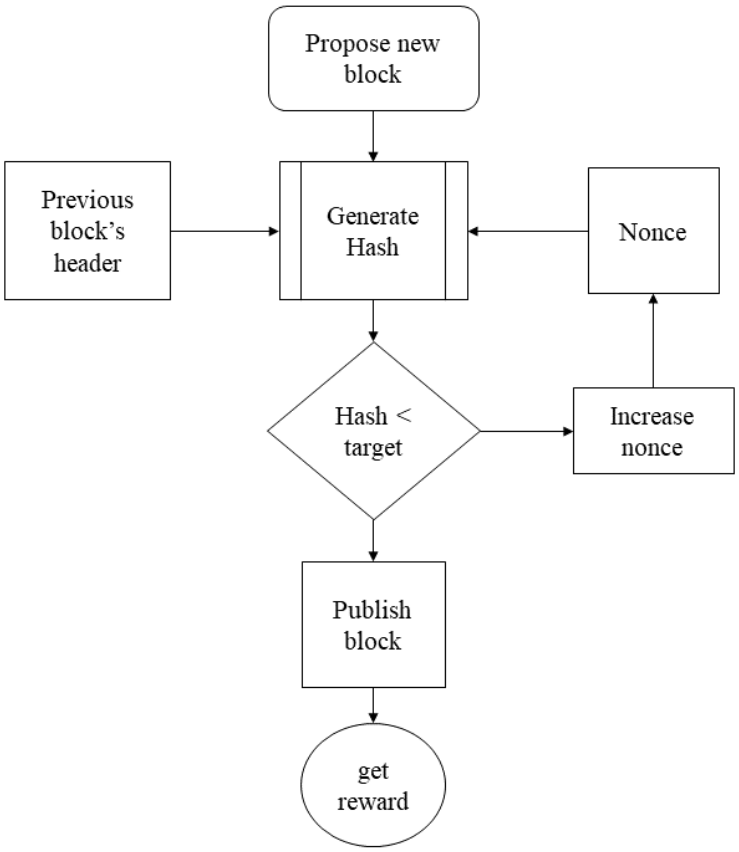

Figure 6: Minning process.

Other nodes accept the newly generated block after verifying the transaction set, ensuring all transactions are valid (every transaction is properly signed, coins are not double spent and/or are created out of thin air). These nodes will double-hash the block header and check whether the resulting hash is below the block's included target value. If it is valid, these nodes will keep propagating the block until every node has an updated ledger.

There is a chance of multiple blocks being validated at the same time in different nodes, which results in the appearance of multiple chains, as it is shown in Figure 7:

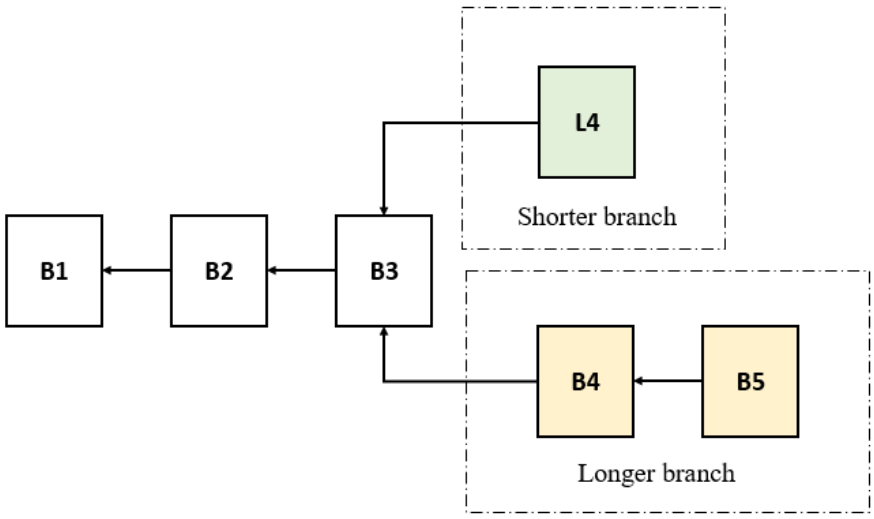

Figure 7: Fork.

In this situation, the network stores every new branch of the chain. At some point, one of the new forks will get longer than the rest, which will automatically set this fork as the authentic one and every node will abandon all other chains but the longest, which is assumed to have been produced by a network majority of computational power and thus, to represent the most valid state of the ledger. This leads into malicious attackers being continuously override by the honest agents if they cannot control more than 51% of the computational power of the network. If dishonest nodes manage to gain more than the 51% processing power, they can regenerate the blockchain history modifying the transactions registry [22].

2.6.1.2 Proof of Stake

The Proof of Stake (PoS) model is an alternative to PoW in which the computational consumption is replaced by a random selection process, where the chance of mining the block is proportional to the wealth of validators. The idea behind this methodology is that the more stake is invested by users into the system, the more likely they will want the system to succeed, and the less likely they will try to collapse it. Stake is normally an amount of cryptocurrency that the user has invested into the system, and it is no longer available to be spent. This amount of stake determines the chances for a node to successfully mine the block.

Due to the above-mentioned process, there is no need to perform highly consuming tasks (which involve time, electricity, and computing power) as in PoW models. Depending on how the stake is used, different approaches can be found [31]:

- If the choice of the block publisher is a random choice (*chain-based proof of stake*), the algorithm will notice every node which owns a stake and select among them based on their ratio of stake to the total amount of cryptocurrencies staked: if a user owns the 33% of the network stake, he/she will be chosen, on average, 33% of the time.
- When the choice of the publisher is a multi-round voting system (sometime referred to as *Byzantine fault tolerance proof of stake*), several staked users are selected to generate proposed blocks. These agents will emit votes during several voting rounds until consensus is reached, leading to the creation and appending of the proposed block. By applying this method the publisher is granting all staked users to have power in the block generation process.
- Using a delegate system (also known as *delegated proof of stake*) where users vote for nodes to become publishing. The influence of the user's votes depends of their stake so the larger it is, the heavier their vote is. The elected nodes become publishing nodes and can validate and publish blocks. Blockchain network users can also vote against an already elected publishing node, to try to erase them from the set of publishing nodes. This voting scenario is continuous so publishing nodes are incentivized to act honestly, otherwise their privilege (and rewards) will be removed. In addition, users of the network vote for delegates, who participate in the governance of the blockchain proposing changes, and improvements, which will be voted on by blockchain network users.

2.6.1.3 Proof of Elapsed Time

Proof of elapsed time (PoET) is a blockchain network consensus mechanism algorithm where each publishing node requests a wait time from a secure hardware time source within their computer system.

The working of the PoET algorithm is quite simple: every participating node in the network must wait for a random amount of time, and the first one that completes the established waiting time wins the new block. All nodes in the blockchain network generate a random wait time and enter sleep mode for that specified duration. The one that wakes up first earns the right to append a new block to the blockchain, broadcasting the new information to the whole peer network.

Two crucial elements need to be ensured by the PoET consensus algorithm. The first one is to ensure that every agent is waiting a time that is actually random and not a shorter duration in order to win fraudulently. Consequently, the second element is guaranteeing that the winner has indeed waited for the generated random time. These requirements are solved by executing software in a trusted execution environment found on some computer processors (such as Intel's Software Guard Extensions, AMD's Platform Security Processor, or ARM's TrustZone) [42] [31].

2.6.1.4 Proof of Authority

Block generation in Proof of Authority (PoAu) based systems requires special permissions. For example, one node holding a special key may be responsible for generating and appending every new block. This mechanism can be seen as a modification of PoS algorithm where validator's strength resides on their identity, in contrast with PoS, where validators' power is strictly tied to their stake. Network's users deposit their trust on authorized nodes and blocks are accepted if the majority of these authorized nodes sign it with their vote.

Despite this method represents a more centralized approximation, typical from governing or regulatory institutions, it is also proving popular with utility companies in the energy sector. This consensus algorithm can also be useful in cases where the security and integrity of the system can not be risked under any circumstances.

2.6.1.5 Proof of Activity

Proof of Activity (PoA) is a hybrid protocol that combines proof of work and proof of stake. Initially, a block template without transactions is proposed with a suitable nonce using a traditional PoW approach. Then, *N* nodes will be chosen randomly depending on their stake in the network. Block validation finishes once all signatures from the group have been collected. This consensus protocol embraces both advantages and disadvantages from PoW and PoS models.

2.6.1.6 Proof of Burn

The idea behind Proof of Burn (PoB) is to '*burn'* cryptocurrency instead of wasting computational resources as PoW. The more coins are burnt, the more likely a node will earn the privilege of mining the next block. This validation process relies on the willingness to waste money, which avoids centralization issues due to hardware equipment, unlike PoW [22].

2.6.1.7 Proof of Capacity

Proof of Capacity (PoC) consensus algorithm makes use of hard disk space as a resource in order to mine the blocks. This strategy is significantly different from PoW algorithms, where computational resources are used. In PoC, the *mining* term is replaced by *hard drive mining,* since it is the actual resource used to mining.

In contrast with PoW schemes where each miner needs to alter repeatedly the nonce in order to reach a valid solution, in PoC a set of possible set of solutions are already stored on the node's hard drive before the mining process starts. The larger the hard drive is, the more possible will it be to contain a hash value from the set that matches the target difficulty, leading into more chances to succeed in the process [43].

2.6.2 Voted-Based Consensus Algorithms

To implement a voting based consensus algorithms it is required that nodes taking part in the network are known and adjustable, so that the exchange of messages is easier to perform. This is the main difference compared to the proof-based ones, where nodes are free to join or to withdrawn from the verifying network. Besides this, all the nodes in the network work together to verify transactions or blocks. Communication between them is needed before appending any new block to the chain. It is a common pattern to establish a threshold which will determine the minimum number of nodes that should have the same proposed block in order to be appended.

2.6.2.1 Practical Byzantine Fault Tolerance

Practical Byzantine Fault Tolerance (PBFT) algorithm was proposed by Castro and Liskov [44], and it was used for Hyperledger Fabric. It focuses on solving the already explained Byzantines generals problem in section 2.5.2. PBFT works under the assumption that less than one-third of the peers are faulty, which are denoted as f. This means that the network should be formed of at least $n = 3f + 1$ peers to handle f faulty nodes. From this equation it can be extracted that $f = [\frac{n-1}{3}]$, which means that the network requires $2f + 1$ peers to agree on the block of transactions.

In PBFT, there are two types of nodes: validating peers and one leader node. Clients make transaction requests to different validating peers, which validate and broadcast them to other peers. A few seconds later, the leader creates a block that contains these new transactions (ordered by their timestamps) and proceeds to broadcast it to the validating peers. Consensus is reached if $2f + 1$ peers agrees with the state of the proposed block. Then, validating peers execute every transaction and append the block as the next block of their private ledger [45].

2.6.2.2 Federated Byzantine Agreement

The Federated Byzantine Agreement algorithm (FBA) is a form of PBFT where there is no *leader*. FBA presents a high throughput, network scalability, and low transaction costs. Highly known cryptocurrencies using the Federated Byzantine Agreement include Stellar and Ripple.

FBA algorithm requires nodes to be known and verified. These nodes deposit their trust on a subset of validators that each member may consider trustworthy, and eventually quorums of nodes emerge from decisions made by the individual nodes making up the FBA network. A block is accepted if it is signed by a specific quorum of validators, defined as a sufficient set of nodes required to reach consensus [46].

2.7 Operating blockchain systems

Once blockchain basis have been presented, it will be explained some of the most relevant blockchain systems that are operating today[8].

2.7.1 Bitcoin

Bitcoin is the first application of blockchain technology. Since its introduction in 2008 by Satoshi Nakamoto and its practical implementation in 2009, Bitcoin has earnt massive popularity. Nowadays it is the most successful cryptocurrency in the world with an approximate market capitalization of 736 bn [47].

The motivation and objectives of Bitcoin are shown in its white paper, where it is stated that Bitcoin provides 'a purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution'. It is argued there that digital signatures can grant this functionality, but the main benefits of using them are lost since they require a third party preventing the double-spending issue. Consequently, they propose a P2P network to address this problem, where transactions are timestamped through hashing them within 'an ongoing chain of hash-based proof-of-work' that forms a network state that cannot be modified unless redoing the proof of work. The paper points out that the longest chain serves both as a witness of every event happened in the network, and as an evidence of the computational cost invested to verify every transaction. Regarding malicious agents, as long as the majority of computational power is held by honest nodes, they will keep generating the longest chain and will outpace any possible attacker. To conclude, the network structured is described as *minimal,* and agents can leave or rejoin the network at any time, accepting the longest PoW chain as evidence of the events that happened while they were not part of the network [11].

As stated in a technical report by the European Commision, 'Besides an effective procedure to transfer an amount of virtual currency from one user (account) to another user (account), the major and indeed an essential contribution of the concept of the Bitcoin is the solution to the general problem how to establish trust between two mutually unknown and otherwise unrelated parties to such an extent and certainty that sensitive and secure transactions can be performed with full confidence over an open environment, such as Internet' [48].

2.7.2 Ethereum

Ethereum, as well as Bitcoin, is a public blockchain. Its introductory paper was originally published in 2013 by Vitalik Buterin, the founder of Ethereum, before the project's launch in 2015. Ethereum, like many

[8] 2021 year.

community-driven and open-source software projects, has evolved since its beginnings [49].

This project was born to enable users to create smart contracts by using a built-in fully fledged Turing-complete programming language. This way, users are allowed to create any system beyond the pure monetary transaction.

The design behind Ethereum is intended to fulfill the following premises [49]:

- **Simplicity**: the Ethereum protocol is aimed to be as simple as possible, even at the cost of some data storage or time inefficiency. This means that an average programmer should be able to follow and implement the entire specification of his project.
- **Universality**: Ethereum does not provide any features. Instead, Ethereum brings to its users an internal Turing-complete scripting language, which can be used to create any smart contract or transaction type that can be mathematically defined.
- **Modularity**: the parts of the Ethereum protocol should be designed to be as modular and separable as possible.
- **Agility**: details of the Ethereum protocol may be changed in order to improve scalability or security.
- **Non-discrimination** and **non-censorship**: no categories of usage will be actively restricted or prevented by the protocol. Every regulatory mechanism shall be developed to mitigate the harm, instead of opposing specific unwanted applications. For example, any user can run an infinite loop script within Ethereum as long as the tariff fees are being paid.

2.7.2.1 State of Ethereum's ledger

In this blockchain, the state is conformed by a set of objects called "accounts". Each account has a 20-byte address and the transfer of data between these objects leads into state transitions of the ledger. Every Ethereum is made up of four fields:

- The nonce, a number that it used to make sure that a transaction is only processed once.
- The account's ether balance.
- The account's contract code, only if it exists.
- The account's storage.

Ether is the main cryptocurrency found in Ethereum and it is needed to pay transaction fees. Two main categories can be made around the concept of "Ethereum account":

- Externally owned accounts, which belong to private users and are controlled exclusively by their private keys. These accounts do not present any code, and any agent can send data or any message by creating and properly signing a transaction.
- Contract accounts, which are only controlled by their contract code. Any user can interact with these contract accounts by sending messages, which will activate the code inside the contract and enabling it to read or write to its internal storage, to send messages or to create and deploy new contracts.

2.7.2.2 Ethereum blockchain itself and mining

Ethereum is very similar to the Bitcoin blockchain, but some differences can be found. The most important difference among these blockchains is that, regarding their blockchain architecture, every Ethereum block contains a copy of the most recent state of the ledger, as well as the transaction list (in Bitcoin every block only contains a copy of the new set of transactions). In addition, there are two other fields that are included in each Ethereum block: the block number and the difficulty. The block validation algorithm is carried out in the following manner:

1. Verify whether the previous block exists and is authentic.
2. Verify that the timestamp of the new block is greater than the previous one but not larger than 15

minutes into the future.

3. Verify that the following block fields are valid: difficulty, block number, transaction root, gas limit and uncle root.
4. Verify the proof of work given.
5. Update the state at the end of the previous block with the new set of transactions. If any error is encountered, an error is returned.
6. Verify whether the Merkle tree root of the new state is equal to the state recorded in the proposed block header. If all these conditions are met, the block is valid, otherwise it is discarded.

The reason behind keeping the data in this tree structure is that after every block only a small part of the tree is changed. Which means that, in general, the trees of two consecutive blocks are genuinely similar. This leads into data being saved once and referenced later using pointers.

Other important concern is where the contract code is executed. The answer is simple: the execution of contract code is part of the state transition function, which belongs to the block validation algorithm. This means that if a transaction is pumped into a block called *k*, the code execution generated by that transaction will be accomplished by every node, from now until the end of days, that download and validate the block *k* [49].

2.7.3 Hyperledger

To conclude this operating blockchain system list it is included Hyperledger, which is an open-source collaborative project carried out to promote blockchain for enterprises. Hyperledger, hosted by The Linux Foundation, started in 2015 when a set of companies decided to combine efforts in order to develop something valuable. Resources from these companies were pooled to create an open source blockchain technology accessible to anyone. This collaboration among the developers across multiple initiatives aimed to reach standardization, reuse, and interoperability between different blockchain technologies developed under the project. The different business blockchain technologies, libraries and tools developed within Hyperledger project are shown in Figure 8.

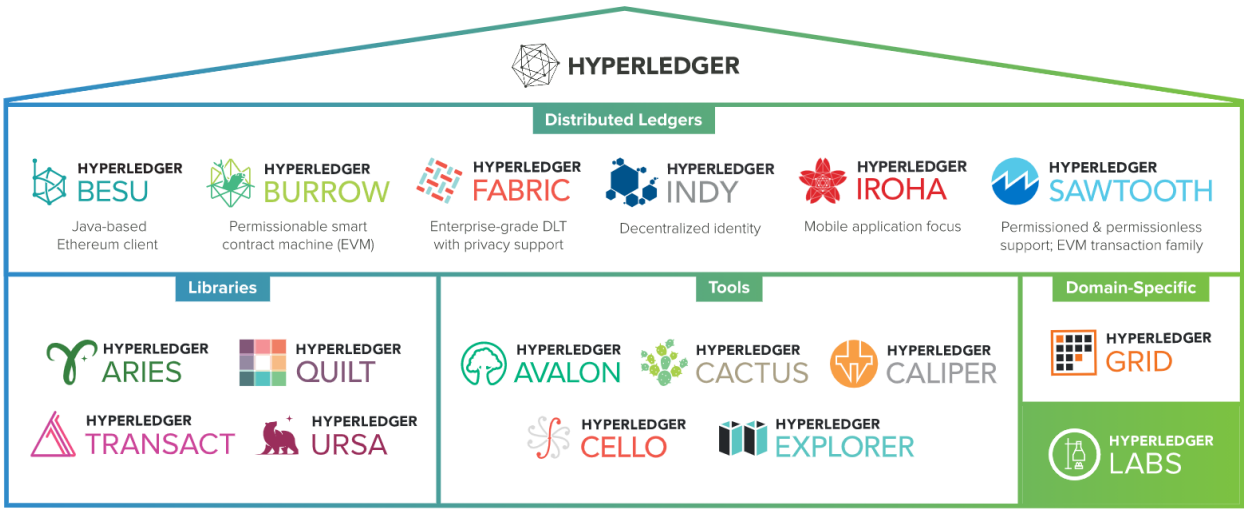

Figure 8: Hyperledger projects [50].

3 BLOCKCHAIN FOR ENERGY TRADING

3.1 Blockchain in the energy sector

As previously mentioned, the energy sector is experiencing a structural shift due to the recent inclusion of RES, and DERs. Also, power is no longer being exclusively transferred in one direction, but, in both. The microgrid concept emerges to handle this archetype by splitting the power grid into smaller portions based on distributed schemes. A microgrid is a group of loads, distributed energy resources, and energy storage systems that act coordinately to deliver energy in a reliable manner. According to [51], microgrids can work both connected to the main grid or completely isolated. Switching between these operation modes implies sophisticated control to maintain stability and an economically efficient operation. In this paradigm, the inherent decentralized nature of blockchain provides a possible solution to control and handle this increasingly decentralized and complex energy systems, as well as enabling P2P safe energy trading.

Multiple applications of blockchain can be found on the energy sector such as billing through automated smart contracts; decentralized trading platforms; automated demand response; communication asset among smart devices, data transmission or storage; enhancing security and privacy; and transparency.

Blockchain evolution can be summarized in three phases: blockchain 1.0, 2.0, and 3.0. Phase 1.0 starts with the inclusion of cryptocurrency as a payment option along as other classical payment systems; phase 2.0 incorporates the smart contract features, enabling higher quality and more complex procedures to be deployed within blockchain; and phase 3.0 will start with blockchain applications beyond currency, economics or markets, such as the combination between blockchain and Big Data. Big Data's predictive analysis could be highly improved with the automatic execution triggers from blockchain smart contracts [52], [53].

Some of the potential blockchain use cases in the energy sector are summarized below [22]:

1. Blockchain in energy company operations: many use cases in this field could be improved by using blockchain such as automated billing; sales and marketing, incorporating artificial intelligence and machine learning; blockchain-enabled markets; green certificates trading; control of decentralized energy systems; smart grid applications; grid management; and transparency.
2. Wholesale energy trading: smart contracts allow P2P energy trading through erasing the middle-man unit.
3. Imbalance settlement: potential exact tracking of which generator and consumer create an imbalance while real time billing is allowed. Latency and scalability are challenges to be addressed here.
4. IoT platforms, such as smart houses, where machine to machine communication can be enabled through P2P blockchain transactions, or data exchanges among smart gadgets.

In the following section it proposed a blockchain-based energy management platform that focuses on maximizing the global wellness of their agents by solving a distributed optimization problem following iterative steps using a smart contract deployed in the Ethereum blockchain, serving as a virtual aggregator which erases the need of a third party controlling and distributing the data. The concept of global wellness refers to minimizing the financial costs from the electricity retail market. Agents can be divided in two groups: consumers and prosumers. The former is made up of users that will only place demand orders into the local microgrid due to not owning any generation capacity. In the other hand, prosumers are consumers that may have some storage capacity that enables them to store and trade energy, or any sort of generation utility such as photovoltaic panels.

3.2 Model

The platform is designed to work in a microgrid where there is a number of prosumers, who may have access to photovoltaic panels (PV), electric vehicles (EV) and batteries. It works as a one day-ahead energy scheduling program in which every household is considered to be an independent node of the network. Every

node can publish their surplus or deficit of energy budget among the rest of agents using blockchain. This information will be used to adjust the energy trades through the network in order to reach the global wellness goal.

3.2.1 Electric model

The microgrid considered in this work is represented by a collection of n nodes, indexed by $i = 0, \dots, n$, placed in a complete graph K_n, meaning that every pair of nodes is connected by a pair of unique edges in each direction. The network is modeled over T number time steps indexed by $t = 0, \dots, T$. Every agent is connected to the microgrid and the external grid. Power imported from the latter is defined as $p_{i,t}^{g}$. The cost function for each node i in timestep t is modeled as:

$$C_{i,t}^{g}(p_{i,t}^{g}) = c_t p_{i,t}^{g} \Delta t \qquad \forall\, i, t. \quad (1)$$

where c_t is the monetary cost of purchasing each kWh and $\Delta t = 24/T$ determines the amount of hours in each timestep.

Each household presents a fixed load demand $p_{i,t}^{l}$ that is uncontrollable. The power generation through solar panels $p_{i,t}^{pv}$ is divided into the actual photovoltaic energy used, $p_{i,t}^{pv_u}$, and the photovoltaic energy surplus, $p_{i,t}^{pv_s}$, defined in (2). $p_{i,t}^{pv}$, $p_{i,t}^{pv_u}$ and $p_{i,t}^{pv_s}$ are constrained as can be seen from (3) to (5).

$$p_{i,t}^{pv} = p_{i,t}^{pv_u} + p_{i,t}^{pv_s} \qquad \forall\, i, t. \quad (2)$$

$$0 \le p_{i,t}^{pv} \le \overline{p_{i,t}^{pv}} \qquad \forall\, i, t, \quad (3)$$

$$0 \le p_{i,t}^{pv_u} \le \overline{p_{i,t}^{pv}} \qquad \forall\, i, t, \quad (4)$$

$$0 \le p_{i,t}^{pv_s} \le \overline{p_{i,t}^{pv}} \qquad \forall\, i, t. \quad (5)$$

The availability of batteries and electric vehicles incorporate additional constraints. EV are modeled as flexible controllable charges that allow their users to choose the charging power for each timestep t, $p_{i,t}^{ev}$. This charging power is constrained as follows:

$$0 \le p_{i,t}^{ev} \le \delta_{i,t} \overline{p_{i,t}^{ev}} \qquad \forall\, i, t. \quad (6)$$

where $\delta_{i,t}$ is a binary variable that represents the possibility to charge the EV at timestep t. The EV charging efficiency is defined by η^{ev} and the EV daily total charge must accomplish the charging energy demand E_i^{ev} as follows:

$$\sum_{t=0}^{T} \eta^{ev} p_{i,t}^{ev} \Delta t = E_i^{ev} \qquad \forall\, i. \quad (7)$$

The net battery power $p_{i,t}^{b}$ represents the difference between the discharging power $p_{i,t}^{bd}$ and the charging power $p_{i,t}^{bc}$ of agent i at timestep t as seen below:

$$p_{i,t}^{b} = p_{i,t}^{bd} - p_{i,t}^{bc} \qquad \forall\, i, t. \quad (8)$$

Energy stored within batteries is defined as $E_{i,t}^{b}$, and the efficiency of both charging and discharging are defined as η_c^b and η_d^b, respectively. The mathematical relation among them can be seen in (9).

$$E_{i,t}^{b} = E_{i,t-1}^{b} + \left(\eta_c^b p_{i_1^t}^{bc} - \frac{p_{i,t}^{bd}}{\eta_d^b}\right)\Delta(t) \qquad \forall\, i,t. \qquad (9)$$

The upper and lower restrictions for these variables are defined below:

$$\underline{p_{i,t}^{bd}} \leq p_{i,t}^{bd} \leq \overline{p_{i,t}^{bd}} \qquad \forall\, i,t, \qquad (10)$$

$$\underline{p_{i,t}^{bc}} \leq p_{i,t}^{bc} \leq \overline{p_{i,t}^{bc}} \qquad \forall\, i,t, \qquad (11)$$

$$\underline{E_{i,t}^{b}} \leq E_{i,t}^{b} \leq \overline{E_{i,t}^{b}} \qquad \forall\, i,t. \qquad (12)$$

The power balance for every agent is defined in (13), where offer must accomplish demand $\forall t$, and $p_{i,t}^{g}$ is calculated.

$$p_{i,t}^{g} = p_{i,t}^{l} + p_{i,t}^{ev} - p_{i,t}^{pv_u} - p_{i,t}^{b} \qquad \forall\, i,t. \qquad (13)$$

Taking into consideration these constraints the optimization problem to be solved for each household is as follows:

$$\begin{aligned} &\min_{\forall\, p_{i,t}^{g}} \sum_{t=1}^{T} C_{i,t}^{g}(p_{i,t}^{g}) \\ &s.t. \quad (2)-(13) \end{aligned} \qquad (14)$$

3.2.2 Proposed trade mechanism

For the trading strategy it is proposed a model where every household can trade energy with any agent in order to minimize the global economic effort of the community. This system enables users to share energy when they have excess and could be lost in other scenario or adapt the battery usage to prevent other households from purchasing energy when it is more expensive than usual. These premises are included in the following cost function:

$$\min_{\forall\, p_{i,t}^{g}, p_{i,t}^{t}} \sum_{t=1}^{T} [C_{i,t}^{g}(p_{i,t}^{g}) + \sum_{j \neq i}^{n} c_t(p_{j,t}^{g} - p_{ij,t}^{t})] \qquad \forall\, i, \qquad (15)$$

where $p_{ij,t}^{t}$ is the power sent by i to j at timestep t. Note that the sign of the $p_{ij,t}^{t}$ trade is negative since it is preventing agent j from purchasing this power from the utility grid, and the optimization problem (14) needs to be solved first in order to know $p_{j,t}^{g}\ \forall j,t$ from (15). This new objective function implies adding new features into the model presented in Section 3.2.1:

- $p_{i,t}^{bd}$ is divided into $p_{i,t}^{b_u}$ and $\sum_{j \neq i}^{n} p_{ij,t}^{b_t}$, as shown in (16). $p_{i,t}^{b_u}$ represents the energy that is extracted from the battery and is consumed without being transferred to any agent. On the other hand, $p_{ij,t}^{b_t}$ is the energy sent from i to j in t. Note that (17) forces $p_{ij,t}^{b_t}$ to be lower than the energy that j needs to purchase from the external grid in t, $p_{j,t}^{g}$.

$$p_{\mathrm{i,t}}^{bd} = p_{\mathrm{i,t}}^{\mathrm{b_u}} + \sum_{\mathrm{j}\neq \mathrm{i}}^{\mathrm{n}} p_{ij,t}^{b_t}, \qquad \forall i,t, \qquad (16)$$

$$0 \leq p_{ij,t}^{b_t} \leq p_{j,t}^{g}, \qquad \forall i,t. \qquad (17)$$

- The solar panel energy balance from (2) is modified to (18) by adding $\sum_{j\neq i}^{n} p_{ij,t}^{pv_t}$, which represents the photovoltaic energy traded from i to the rest of agents in every timestep t. Again, every $p_{ij,t}^{pv_t}$ trade is bounded in (19) to ensure that they are not greater than every individual power deficit for $\forall j,t$.

$$p_{i,t}^{pv} = p_{i,t}^{pv_u} + \sum_{\mathrm{j}\neq \mathrm{i}}^{\mathrm{n}} p_{i,t}^{pv_t} + p_{i,t}^{pv_s}, \qquad \forall i,t, \qquad (18)$$

$$0 \leq p_{i,t}^{pv_t} \leq p_{j,t}^{g}, \qquad \forall i,t. \qquad (19)$$

- The power balance (13) is reformulated as follows:

$$p_{i,t}^{g} = p_{i,t}^{l} + p_{i,t}^{ev} + p_{i,t}^{bc} - p_{i,t}^{pv_u} - p_{i,t}^{b_u}, \qquad \forall i,t. \qquad (20)$$

- $p_{ij,t}^{t}$ from (15) is defined in (21) and constrained in (22) and (23):

$$p_{ij,t}^{t} = p_{ij,t}^{pv_t} + p_{ij,t}^{b_t} \qquad \forall j,t, \qquad (21)$$

$$0 \leq p_{ij,t}^{t} \leq p_{j,t}^{g} \qquad \forall j \neq i,t, \qquad (22)$$

$$p_{ii,t}^{t} = 0 \qquad \forall t. \qquad (23)$$

Taking into consideration the updated electric and trading model, the following optimization problem is obtained:

$$\min_{\forall\, p_{i,t}^{g}, p_{i,t}^{t}} \sum_{t=1}^{T} [C_{i,t}^{g}(p_{i,t}^{g}) + \sum_{j\neq i}^{n} c_t(p_{j,t}^{g} - p_{ij,t}^{t})] \qquad \forall i, \qquad (24)$$
$$s.t. \qquad (3)-(12), (16)-(23)$$

3.2.3 Proposed distributed algorithm

The distributed algorithm to minimize the optimization problem (24) (25) is composed of 5 steps:

6. First iteration is started with every household solving locally its own optimization problem (14), which does not take into consideration the rest of the network:

$$P_i^g = \min_{\forall p_{i,t}^g} \sum_{t=1}^{T} C_{i,t}^g(p_{i,t}^g) \qquad \forall i, \qquad (25)$$
$$s.t.\ (2)-(13),$$

   where $P_i^g \in \mathbb{N}^{1\times T}$ contains the energy that i needs to purchase $\forall t$ for the next day, $p_{i,t}^g$.

7. The global demand matrix for the next day $P^d \in \mathbb{N}^{n\times T}$ is built with every P_i^g from the previous step and must be known for all households.

8. $P^d = [P_1^g, \dots, P_i^g, \dots, P_n^g]$ is used to configure the upper bounds (17), (19), and (22)from the optimization problem (24), which is solved locally:

$$P_i^t = \min_{\forall p_{i,t}^g, p_{i,t}^t} \sum_{t=1}^{T} [C_{i,t}^g(p_{i,t}^g) + \sum_{j\neq i}^{n} c_t(p_{j,t}^g - p_{ij,t}^t)] \qquad \forall i, \qquad (26)$$
$$s.t. \qquad (3)-(12), (16)-(23),$$

   where $P_i^t \in \mathbb{N}^{n\times T}$ contains the power $p_{ij,t}^t$ that agent i aims to send to each agent at each timestep.

9. The global trade matrix $\Phi \in \mathbb{N}^{1\times T}$ is built with every P_i^t from the last step and a consensus process is started to guarantee that the total energy received $\forall i,t$ is not higher than the respective $p_{i,t}^g$ from P_i^g. It is known that every individual $p_{ij,t}^t$ complains with constraint (22), but the sum of every $p_{ij,t}^t$ can violate it. Thus, after this consensus phase is completed, the compliant trades $p_{ji,t}^{t_c}$ are calculated and it is ensured that $\sum_{j\neq i}^{n} p_{ji,t}^{t_c} \leq p_{i,t}^g \ \forall i,t$, leading to the consensus trade matrix Φ^c.

10. **If** $\Phi^c - \Phi < \epsilon$, where ϵ is the permitted tolerance, all proposed trades are feasible according to the established threshold; **or** $iteration > \psi$, where ψ is the maximum number of iterations, **the algorithm finishes.**

    Else, all agents recalculate locally the optimization problem from step 1 adding the energy trades from Φ^c. Energy received and sent $\forall i,t$ are calculated as $\sum_j^n \Phi_{i,t,j}^c$ and $\sum_j^n \Phi_{j,t,i}^c$, respectively, and included in (13):

$$p_{i,t}^g = p_{i,t}^l + p_{i,t}^{ev} + \sum_{j}^{n} \Phi_{j,t,i}^c - p_{i,t}^b - p_{i,t}^{pv_u} - \sum_{j}^{n} \Phi_{i,t,j}^c, \qquad \forall i,t. \qquad (27)$$

   Once the optimization problem is solved a new P_i^g is obtained that includes the compliant energy trades from Φ^c. **Go to step 2**.

3.2.4 Blockchain implementation

The distributed algorithm from Section 3.2.3 is executed along with a blockchain network to provide full traceability and the ability to audit the process. This is achieved using Ethereum to make full usage of which is a public and permissionless blockchain that provides a feature called smart contracts. Ethereum enable users to create smart contracts by using a built-in fully fledged Turing-complete programming language.

The smart contract coded with Solidity programming language and implemented in the Ethereum blockchain fulfills the following functions:

1. Control flow of the distributed algorithm.
2. Exchange of information between all participants.
3. Execute the consensus algorithm for Φ^c.

Adding the smart contract functionalities to the proposed algorithm leads to the following modification within the distributed algorithm:

- The optimization variables P_i^g obtained in steps 1 and 6 are uploaded to the smart contract, where it is built the global demand matrix P^d and it is available to be checked by every household.
- Once P^d is built, agents access the smart contract and call for different methods that provide them the data required for step 3. This means that every agent knows the energy deficit of their neighbors.
- When every P_i^t from step 3 is submitted to the smart contract the Φ matrix is built and the Algorithm 1 is executed, starting the consensus process that leads to Φ^c.

Algorithm 1 Consensus process for Φ^c

Initialization
$\Phi \leftarrow$ *global trade matrix*, $n \leftarrow$ number of households, $T \leftarrow$ number of timesteps
$\Gamma \leftarrow$ *zero matrix* $\in \mathbb{N}^{n \times T}$, $\Phi^c \leftarrow \Phi$, $i \leftarrow 1$, $t \leftarrow 1$;
for $i := 1; i \leq n$ **do**
  for $t := 1; t \leq T$ **do**
    for $k := 1; k \leq n$ **do**
      $\Gamma(i,t) = \Gamma(i,t) + \Phi(i,t,k)$ ▷ Γ contains the energy received $\forall i, t$
for $i := 1; i \leq n$ **do**
  for $t := 1; t \leq T$ **do**
    $Trade \leftarrow$ *zero matrix* $\in \mathbb{N}^{1 \times T}$
    if $\Gamma_{i,t} > P_{i,t}^{grid}$ **then** ▷ i would receive more power than needed at t
      $exit = 0$
      $deficit = P_{i,t}^{grid}$
      $FinalTrade \leftarrow$ *zero matrix* $\in \mathbb{N}^{1 \times T}$
      for $k := 1; k <= n$ **do**
        $Trade_k \leftarrow \Phi_{i,j,k}$ ▷ contains every trade $\forall k \in n$ to i at t
        if $exit == 0$ **then**
          $aux = deficit - Trade_k$;
          if $aux < 0$ **then**
            $Trade_k = deficit$
            $exit = 1$
          if $aux > 0$ **then**
            $deficit = deficit - Trade_k$;
          if $aux = 0$ **then**
            $exit = 1$
          $FinalTrade_k = Trade_k$
      for $ii = 1; ii \leq n$ **do**
        $\Phi_{i,t,ii}^c \leftarrow FinalTrade_{ii}$

- Termination condition $\Phi - \Phi^c < \epsilon$ or $iteration > \psi$ are verified.

4 APPLICATION DEVELOPED

As described in chapter 3, the main goal of this work is to develop an application that will enable a set of agents to freely trade energy within a microgrid following a distributed control algorithm whose virtual aggregator is a smart contract deployed in the Ethereum blockchain.

Across this chapter it is covered the process of developing the application starting from the different software assets needed, followed by the blockchain implementation, and finishing with the graphic user interface that will enable an easy interaction between households and blockchain.

4.1 Software components

- **Remix – Ethereum IDE**

  It is an open-source web and desktop application that permits a rapid development of smart contracts for the Ethereum blockchain. Remix presents multiple modules for testing, compiling, debugging, and deploying smart contracts in a practice virtual machine, which serves as a first step into the process of validating the developed smart contract.

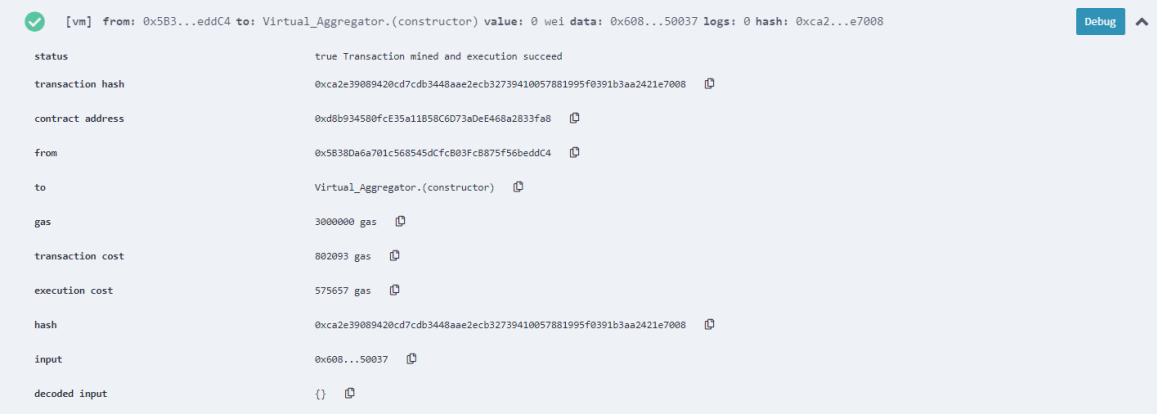

Figure 9: Output from Remix transaction

- **Solidity**

  Solidity is the programming language used for writing smart contracts that run on the Ethereum Virtual Machine (EVM). It is object-oriented and it is designed around the ECMAScript syntax in order to keep it familiar for web developers [54].

- **Node.js**

  Node.js is an asynchronous event-driven JavaScript runtime designed to build scalable network applications, which is built on Chrome's V8 JavaScript engine [55]. Node.js enables developers to use JavaScript for both writing command line tools and server-side scripting. It unifies the web-application development on a single programming language [56].

- **Visual Studio Code**

  Visual Studio Code, also known as VS Code, is a highly popular free open source text editor developed by Microsoft. It is available for Windows, Linux and macOs. Multiple features are included within this editor including debugging, snippets, code refactoring or embedded Git, among others.

- **Web3.js**

  Web3.js is a set of libraries that allow users to interact with a local or remote Ethereum node using HTTP, IPC or WebSocket [57]. It serves as a bridge between Ethereum's JSON RPC interface and

JavaScript, which makes it straight usable in web technology. In addition, web3.js is usually used on the server side in Node.js applications [58].

- **Ganache CLI**

  Ganache CLI is a customizable blockchain emulator that allows to make calls to the blockchain without needing to run an actual Ethereum node. It is ideal for deploying blockchain applications during their early stages due to its following features [59]:

  - No transaction cost and instant "mining".
  - Accounts can be reset with a fixed amount of available Ether.
  - Blockchain inherent parameters such as gas price or mining speed can be altered.
  - Presents a graphic user interface which provides an overview of testchain events.

- **Truffle**

  Truffle is a development environment, testing framework and asset pipeline for blockchains using the EVM, which aims to simplify the process of developing a blockchain project. Truffle provides, among others [60]:

  - Built-in smart contract compilation, linking, deployment and binary management.
  - Tools for automated contract testing.
  - Interactive console which enables direct contract communication.
  - Network management for deploying to different public and private networks.
  - Scriptable deployment and migrations framework.

- **Metamask**

  Metamask is a wallet manager in Ethereum that acts as a bridge between distributed applications and the browser without running a complete Ethereum node. It is available as a browser extension or mobile application, Metamask equips its users with a key vault, secure login, token wallet and token exchange [61].

- **React**

  React, also known as React.js or ReactJS, is a JavaScript library used for building user interfaces. It is sustained by Facebook and a community of individual developers and companies, being one of the most used web frameworks during the last years as it is reflected in the 2020 Developer Survey carried out by Stack Overflow [62].

  React framework is specialized in building dynamic applications through the usage of the virtual DOM, state, lifecycle methods or JSX, which is an extension to the JavaScript language syntax with a similar appearance to HTML [63].

- **Infura**

  Infura grants a set of tools and infrastructure that enable developers to easily upgrade their blockchain application from testing to scaled deployment. It bypasses the need of running an Ethereum node which can take hours or days in order to sync it with the ledger state. As long as more nodes are needed to expand a project infrastructure, more expensive it gets. Infura solves these problems by providing their users access to its already functional Ethereum nodes [64].

4.2 Smart contract functions

The functionalities of the virtual aggregator mentioned in Section 3.2.4 are coded in a smart contract within the Ganache blockchain emulator and compiled using Truffle. In this section it will be reviewed the different functions that needs to be called in order to fulfil every step from the distributed consensus algorithm, except for the consensus algorithm which was already presented in Section 3.2.4. The object of this study is not to provide a complete insight of the Solidity programming language, but some annotations will be done to clarify the proposed syntax.

1. Every agent has obtained a $P_i^g \in \mathbb{R}^{1xT}$ array with the energy deficit or surplus for the next day divided in T time slots. These data are uploaded into Ethereum blockchain by calling $Step1(\text{uint ID, uint[] PBuy, uint[] PSell})$ function:

```
function Step1 (uint ID, uint[] PBuy, uint[] PSell) public{
  Household storage household=household_init[msg.sender];
  Household.ID = ID;
  Household.PBuy = PBuy;
  Household.PSell = PSell;
  Households_initialized += 1;
  for (uint i=0; i<T; i++){
    BuyMatrix[ID][i] = PBuy[i];
  }
  for (i=0; i<T; i++){
    SellMatrix[ID][i] = PSell[i];
  }
}
```

   Agents will make use of Step1 function by introducing their identifying ID, demand array and offer array. These arrays will be stored in both Buy and $Sell$ matrix.

2. Once $Households\_initialized = n$, the first step is concluded and global purchase and sell matrixes are available to be seen through $checkPurchaseOfferMatrix()$ function:

```
function checkPurchaseOfferMatrix() public view returns(uint[][]
PurchaseMatrix, uint[][] OfferMatrix){
    require(Households_initialized == n);
  PurchaseMatrix = BuyMatrix;
  OfferMatrix = SellMatrix;
  return(PurchaseMatrix, OfferMatrix)
}
```

   The $require$ syntax makes this function callable only when $Households\_initalizated = N$, which means that every household has completed step 1. Once both matrices have been returned by this function, every household knows the energies surplus and deficit across the network.

3. Households perform the optimization problem (24), obtaining the energy trades to be sent to the microgrid. These trades are uploaded through $uploadTradeMatrix(uint\ ID, uint[][]\ trades)$ function:

```
function uploadTradeMatrix(uint ID, uint[][] trades) public{
  for (uint i=0; i<n; i++){
    for(uint j=0; j<T; j++){
       TradeMatrix [ID][i][j] = trades[i][j];
    }
  }
  EnergyTradesSubmitted += 1;
```

```
  }
```

4. Once $EnergyTradesSubmitted = n$, it is possible to call $CalculateTradeMatrix()$ function, which transforms $TradeMatrix \in \mathbb{R}^{n \times n \times T}$ into $globalTradeMatrix \in \mathbb{R}^{n \times T}$ by executing the Algorithm 1 from 3.2.4. Once the $globalTradeMatrix$ is obtained, a notification in the graphic user interface will appear informing about its completion.
5. The $globalTradeMatrix$can be obtained calling $checkglobalTradeMatrix()$:

```
  function checkglobalTradeMatrix() public view returns(uint[][][]
globalTradeMatrix){
      require(globalTMcalculated = 1);
    return(globalTradeMatrix);
  }
```

6. Optimization problem (24) is solved and both new Buy and Sell matrixes are uploaded to Blockchain through $Step2$(uint ID, uint[] PBuy, uint[] PSell) function:

```
  function Step2 (uint ID, uint[] PBuy2, uint[] PSell2) public{
    Households_initialized_2 += 1;
    for (uint i=0; i<T; i++){
      BuyMatrix2[ID][i] = PBuy2[i];
    }
    for (i=0; i<T; i++){
      SellMatrix2[ID][i] = PSell2[i];
    }
  }
```

7. Once $Households\_initialized\_2 = n$, the recalculated global purchase and sell matrixes are available to be seen through $checkPurchaseOfferMatrix2()$ function:

```
  function  checkPurchaseOfferMatrix2()  public  view  returns(uint[][]
PurchaseMatrix2, uint[][] OfferMatrix2){
      require(Households_initialized_2 == n);
    EvalConvergence = 1;
    PurchaseMatrix2 = BuyMatrix2;
    OfferMatrix2 = SellMatrix2;
    return(PurchaseMatrix2, OfferMatrix2)
  }
```

8. $EvalConvergence$ is a flag that indicates the possibility of verifying if the algorithm has concluded by calling $EvaluateConvergence()$ function:

```
  Function EvaluateConvergence() public {
      require(EvalConvergence == n);
    EvalConvergence = 1;
    uint ConsensusReached = 1;
    for (uint i=0; i<n; i++){
      for(uint j=0; j<T; j++){
        if(globalTradeMatrix [i][j] - globalConsensusTradeMatrix
>= Threshold) ConsensusReached = 0;
      }
    }
```

```
    if(ConsensusReached == 1 || Iteration >
MaxIterations)AlgorithmFinished = 1;
  }
```

4.3 Graphic user interface

A graphic user interface using React is developed to enable agents to interact easily with the virtual aggregator smart contract. The front page is as follows:

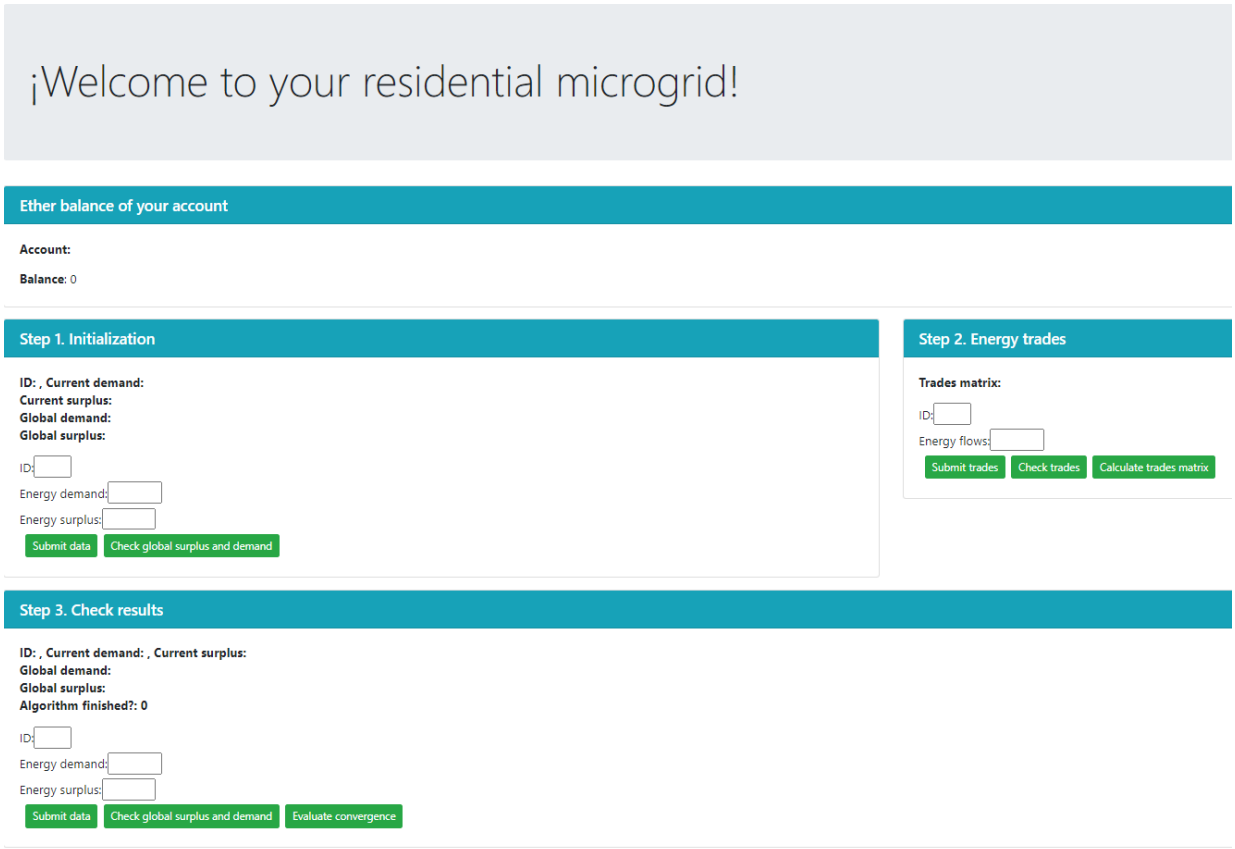

Figure 10: application home page.

As is seen in Figure 10,the web page is divided in four panels:

1. The first one, titled as "Ether balance of your account", shows the public wallet address and the ether balance of the current user visiting the page.
2. "Step 1. Initialization" involves the process of submitting the required information to build both global demand and surplus matrixes.
3. "Step 2. Energy trades" is responsible of gathering every energy trade and displaying them through $Trades\ matrix$:.
4. The last panel shows the recalculated global demand and surplus matrixes, executes the consensus algorithm, and checks whether convergence has been met.

In the first panel no information regarding the user's address or balance is being shown, this is because the React app is not connected with the Ethereum blockchain. To achieve this, it is needed to use web3, which was introduced in 4.1. To make use of web3 functionalities through the browser, it is used a chrome extension called Metamask. Metamask is a crypto wallet manager and gateway to blockchain that enables interacting with distributed applications through the browser without having to run a full Ethereum node. It includes an interface to manage user's identities and sign blockchain transactions. To connect Metamask with Ganache it

is needed to specify the Ganache RPC URL, which is HTTP::///127.0.0.1:7545 as seen in Figure 11:

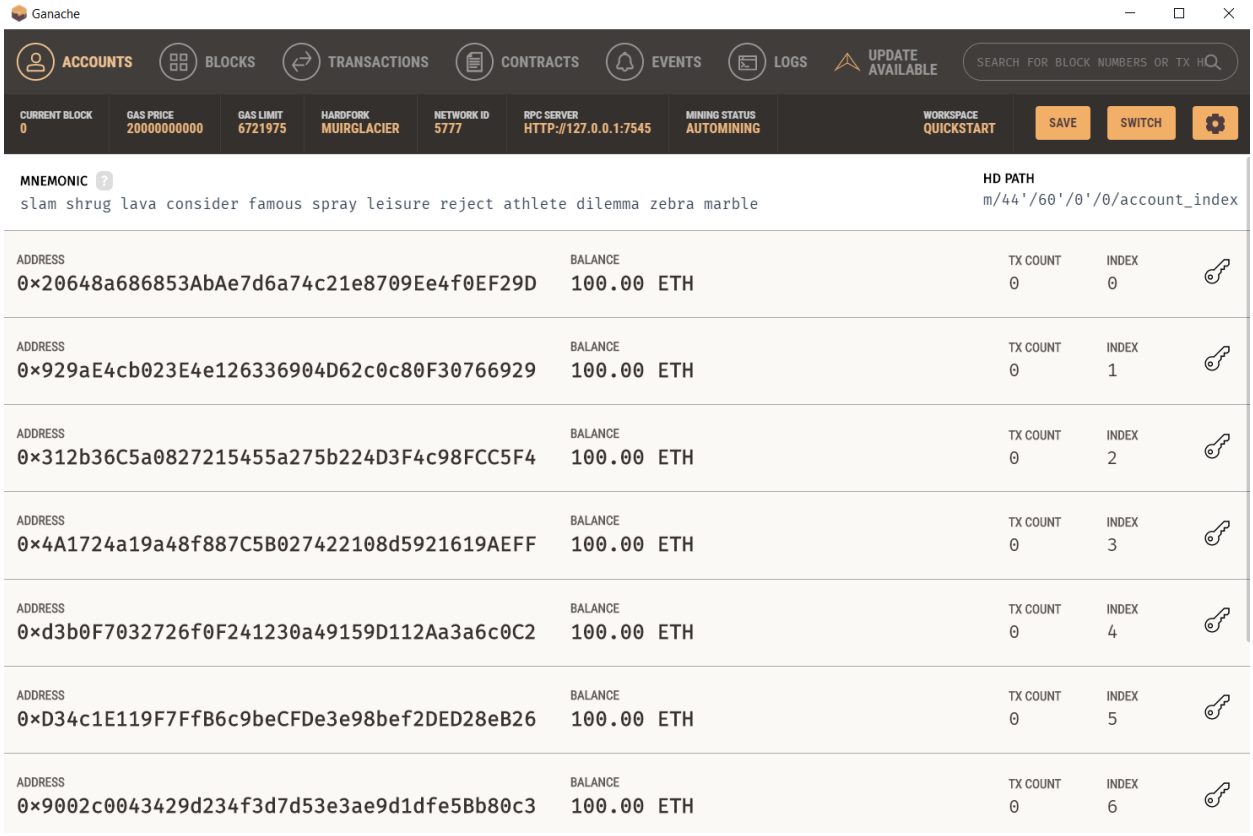

Figure 11: Ganache.

At this point it is possible to interact with the Ethereum blockchain emulator using one of the accounts provided by Ganache through Metamask.

The next step consists in defining a JavaScript function inside the project that permits to instantiate the web3 object. This is achieved using the following *getWeb3* function:

getWeb3.js:

```
import Web3 from 'web3';

const getWeb3 = () => {
  return new Promise( (resolve, reject) => {
    window.addEventListener('load', function(){
      let web3 = window.web3;
      if(typeof web3 !== undefined) {
        web3 = new Web3(web3.currentProvider);
        resolve(web3);
      }else {
        console.error("No provider found, please install Metamask");
        reject();
      }
    })
  });
};

export default getWeb3;
```

Next, an instance of the smart contract is created within the React application to access the functions defined in

4.2. This is achieved by retrieving the json generated from the Truffle compilation and importing it into JavaScript. This json contains the smart contract ABI[9] and bytecode.

VirtualAggregator.js

```
  import VirtualAggregatorContract from "../build/contracts/Virtual_Agg
regator.json";
  import contract from "truffle-contract";

  export default async(provider) => {
    const virtualaggregator = contract(VirtualAggregatorContract);
    virtualaggregator.setProvider(provider);

    let instance = await virtualaggregator.deployed();
    return instance;//return the smart contract instance
  };
  10
```

Finally, making use of web3 and Metamask features, it is possible to call the web3 methods $eth.getAccounts$ and $eth.getBalance$ to interact with the Ganache blockchain through Metamask and retrieve the user information stored inside the blockchain. User's account and balance are displayed using React, as seen in Figure 12.

```
  this.web3 = await getWeb3();//web3 instance
  …
  var account = (await this.web3.eth.getAccounts())[0];//retrieves  the
Metamask account's address
  …
  let weiBalance = await this.web3.eth.getBalance(this.state.account);/
/retrieves the weiBalance
```

[9] ABI stands for The Contract Application Binary Interface. It is the standard procedure to interact with smart contracts within the Ethereum ecosystem.

[10] 10^{18} Wei = 1 Ether

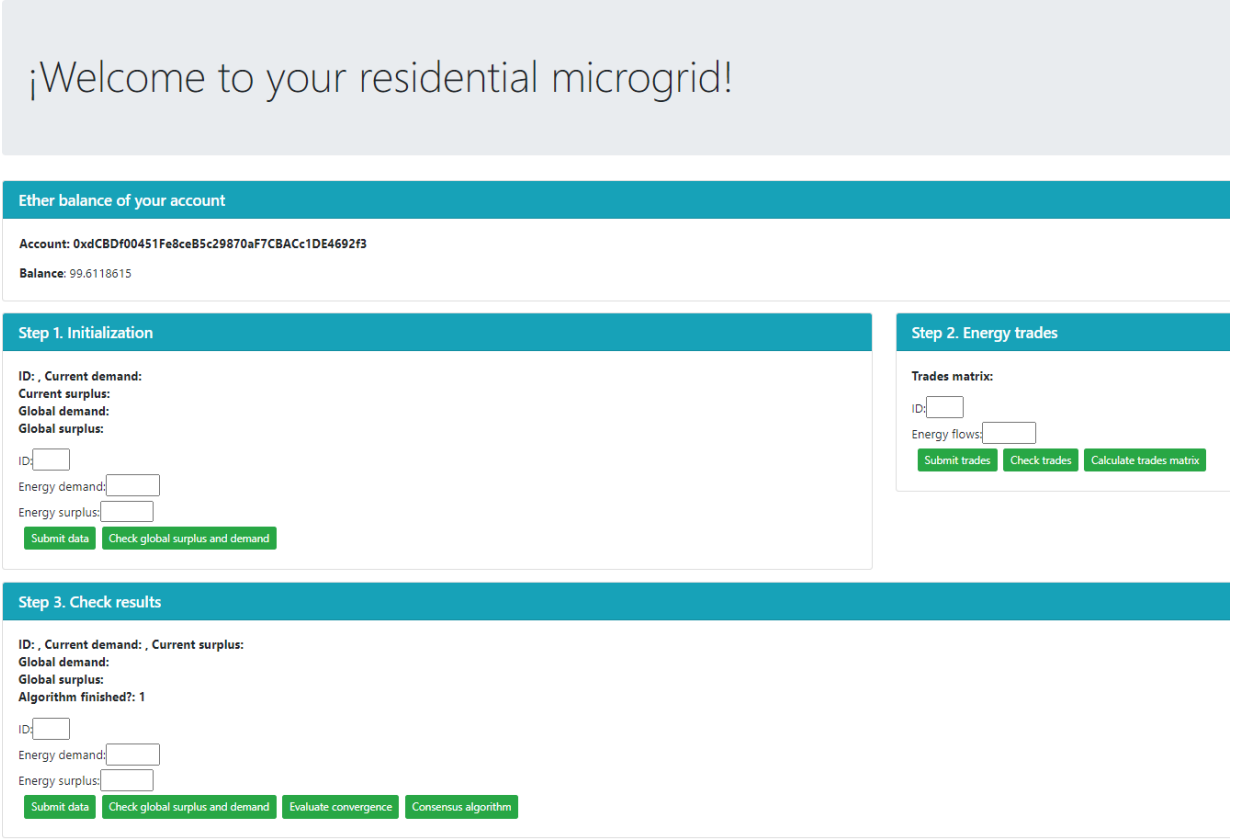

Figure 12: application home page with Metamask web3 provider.

In the next section it is illustrated an example of the process that an agent would have to follow in order to participate in the distributed algorithm proposed in Section 3.2.3.

4.4 Practical example of graphic user interface usage

A simple scenario of three agents is recreated across three timestamps. Agent 0 is defined as prosumer while Agent 1 and 2 are consumers, meaning that only the first agent will be able to send energy across the microgrid. No further details about batteries or electric vehicles will be given since the objective of this section is to provide an insight about the graphic user interface usage.

It will be described the process from Agent 0's point of view:

1. Account address and balance from Agent 0 is displayed once Metamask is connected to Ganache, Figure 13:

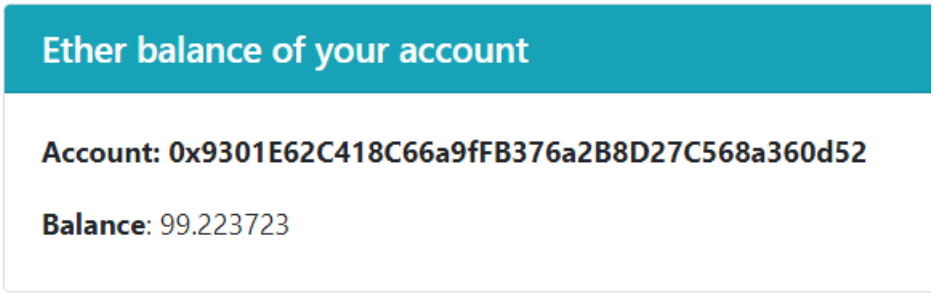

Figure 13: Agent 0's account and balance.

No information is yet available from "Step1. Initialization".

2. Optimization problem (14) is solved by each agent and data is submitted. Agent 0's demand and surplus for the next day are $[0,4600,0]$ (W) and $[2480,0,3560](W)$, respectively.

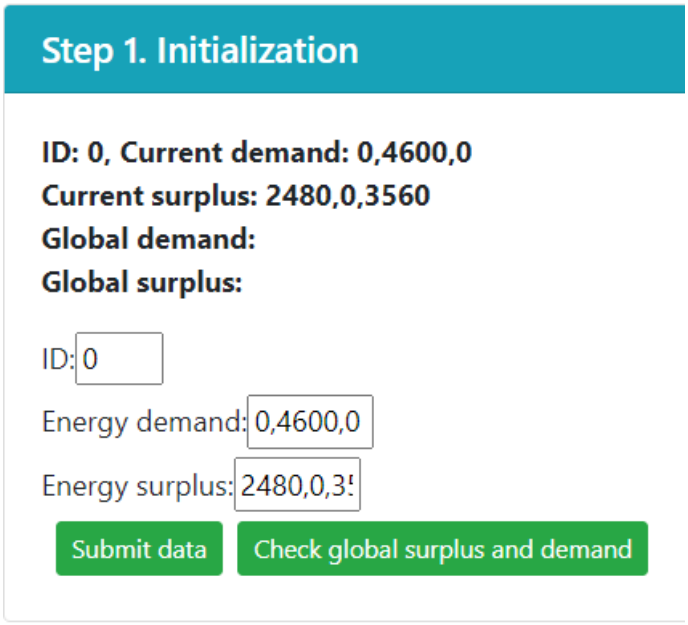

Figure 14: Agent 0's inputs for Step1.

Clicking on the "Submit data" button will trigger the Metamask extension to confirm and sign the transaction, as shown in Figure 15:

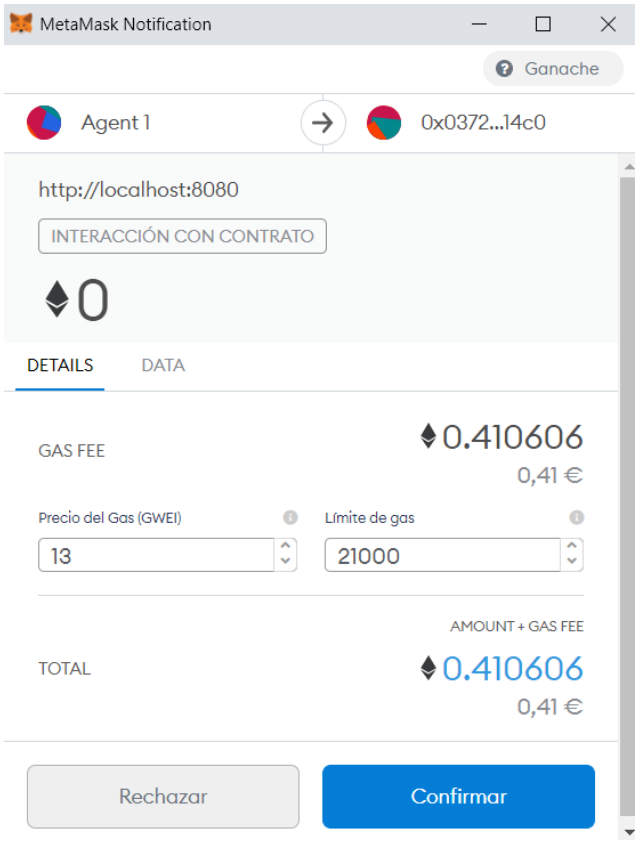

Figure 15: Metamask notification, GUI example.

It is worth noting that the receiver address "$0x0372 \ldots 14c0$" is the actual smart contract address. Once the transaction is mined by a node from the blockchain, and included in the next block, it will appear a notification from Metamask confirming it:

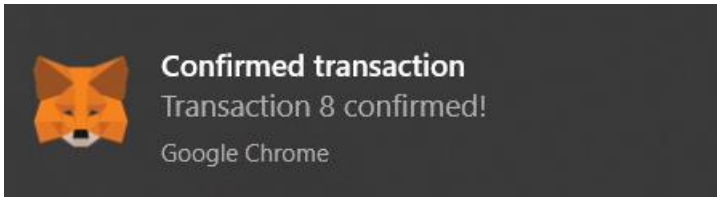

Figure 16: Metamask transaction confirmation.

3. Both global demand and surplus matrixes are retrieved by clicking on the "Check global surplus and demand" button, as seen in Figure 17:

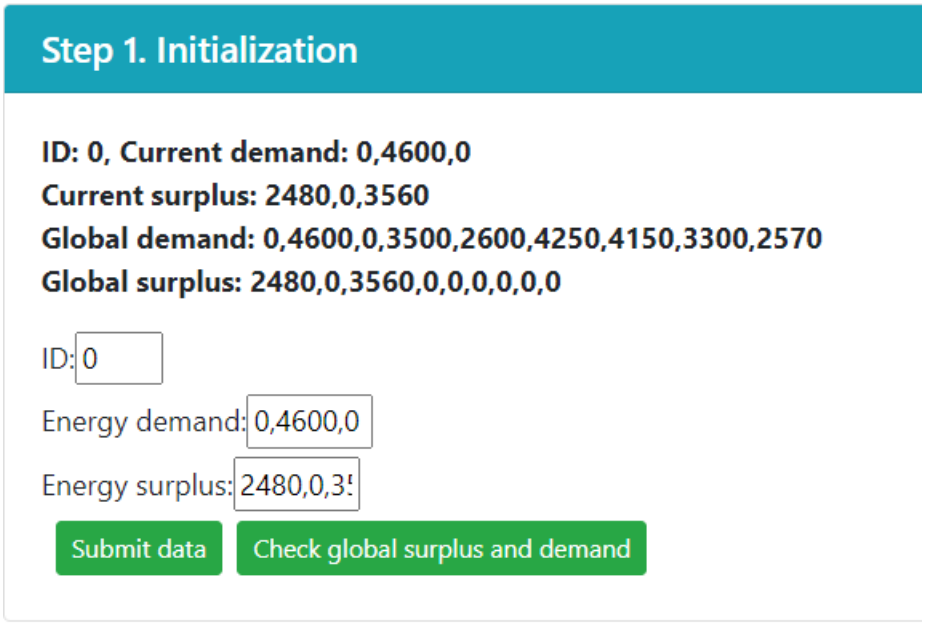

Figure 17: Step 1 global demand and surplus within the microgrid.

From Figure 17 it is subtracted that Agent 1 and Agent 2 have a demand of $[3500,2600,4250]W$, and $[4150,3300,2570]W$ for the next day, and a surplus of $0\ W$ across all timesteps.

4. Energy trades are obtained by solving the optimization problem (24) with Agent 1, and 2 demand and surplus data:

$$P_0^{b_{trade}} + P_0^{PV_{sp}} = [180 + 1240, 0, 150 + 1780] = [1420, 0, 1930]$$

$$P_1^{b_{trade}} + P_1^{PV_{sp}} = [180 + 1240, 0, 150 + 1780] = [1420, 0, 1930]$$

Then, energy flows to be introduced in the "Energy flows" box from "Step 2. Energy trades" are: $[[0,0,0],[1420,0,1930],[1420,0,1930]]$. The first T terms refer to energy flows from $Agent\ 0 \rightarrow Agent\ 0$, which are null. These trades are submitted and then, it is clicked the "Calculate trades matrix" button. Finally, the trade matrix can be checked through the "Check trades" button, as seen in Figure 18:

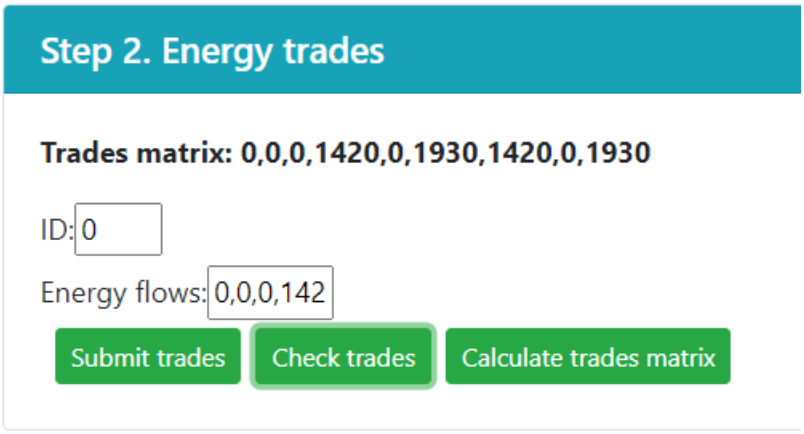

Figure 18: Step 2, practical example GUI.

5. As expected, no trade has been sent to Agent 0 since it is the only prosumer in this scenario. Optimization problem (14) is recalculated adding the energy transactions from the Trades matrix, Figure 19:

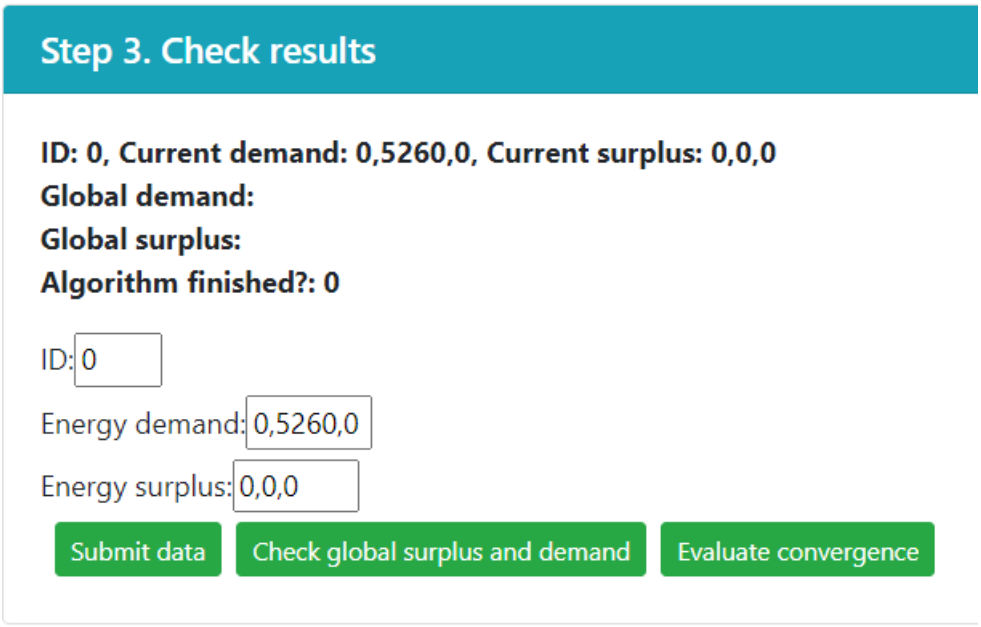

Figure 19: Step 3, practical example GUI.

Checking the global and surplus demand it is observed that the demand of both Agent 1 and 2 is reduced due to the energy sent by Agent 0, as seen in Figure 20:

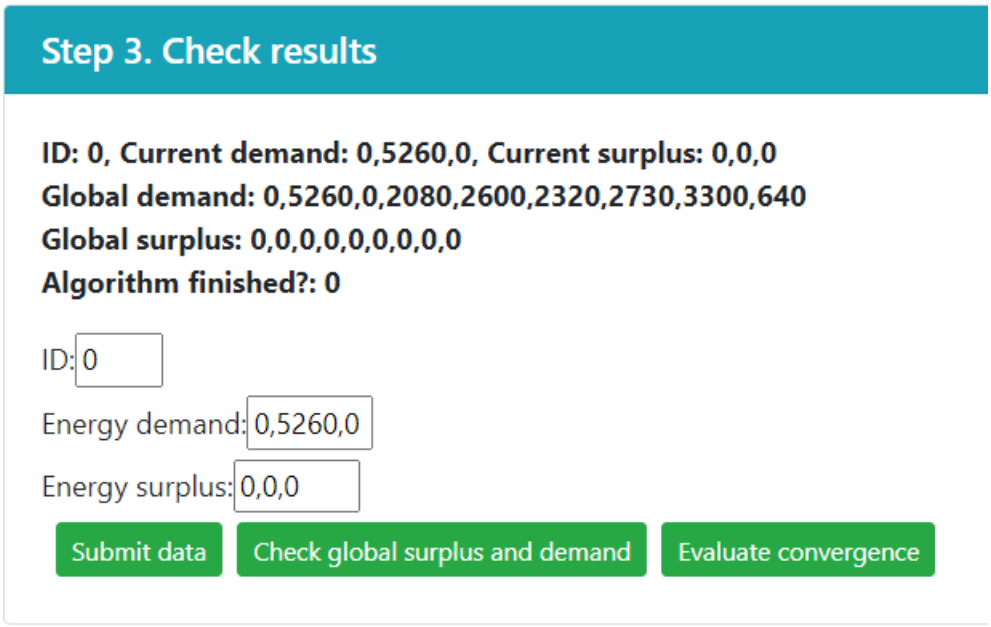

Figure 20: Step 3.1, practical example GUI.

Since no energy surplus exist on any timestamp the algorithm will finished once the "Evaluate convergence" button is pressed, which will change the state of the variable "Algorithm finished?" to 1, as can be seen in Figure 21.

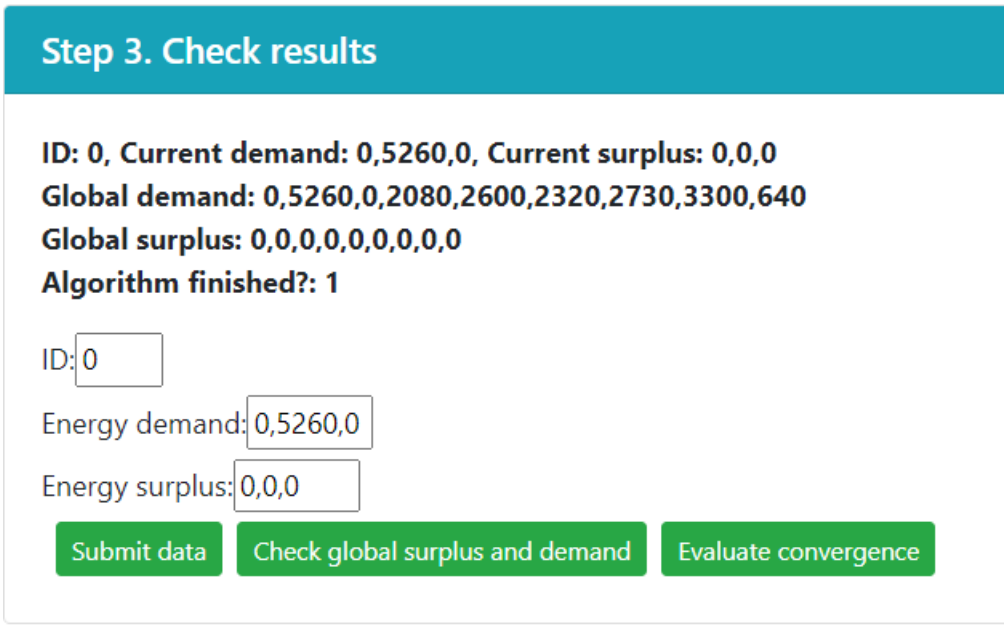

Figure 21: Algorithm finished, practical example GUI.

5 SIMULATIONS AND DISCUSSION

In this chapter, the benefits of using the proposed distributed algorithm through blockchain are illustrated through numerical examples. Two scenarios are recreated to observe the impact of using a classic constant tariff (CT) or an hourly discrimination tariff (HDT).

5.1 Numerical analysis

The topology of the considered microgrid is depicted in Figure 22, where 11 households take part within the network and 4 of them have power generation and storing facilities. For the uncontrollable load of households, various demand profiles are used with a total daily consumption of 9.55kWh, which is the average daily electric energy consumption for a Spanish household according to IDAE [65]. Three load profiles are used within the tests as shown from Figure 23 to Figure 25. The first profile corresponds to households which do not present a significant nightly consumption, whereas in the second profile it is found a profound nightly energy consumption. Load profile 3 represents an average profile between 1 and 2.

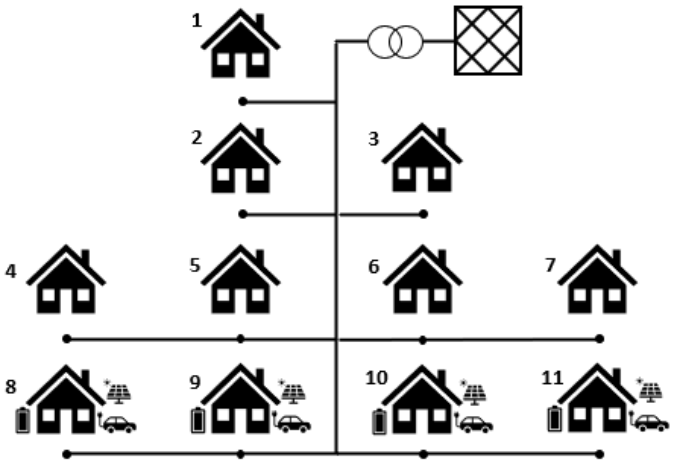

Figure 22: Microgrid simulated.

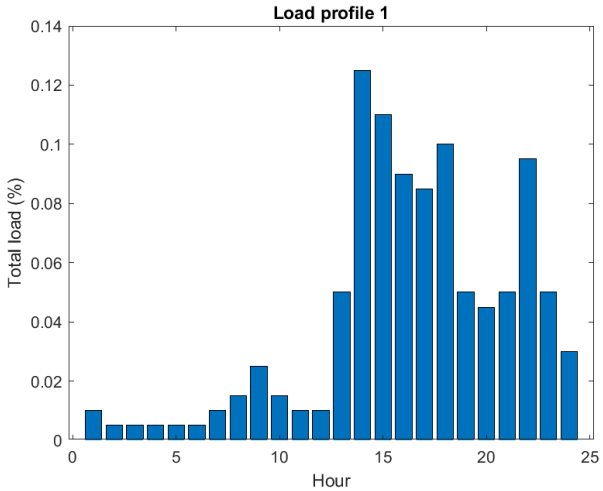

Figure 23: load profile 1.

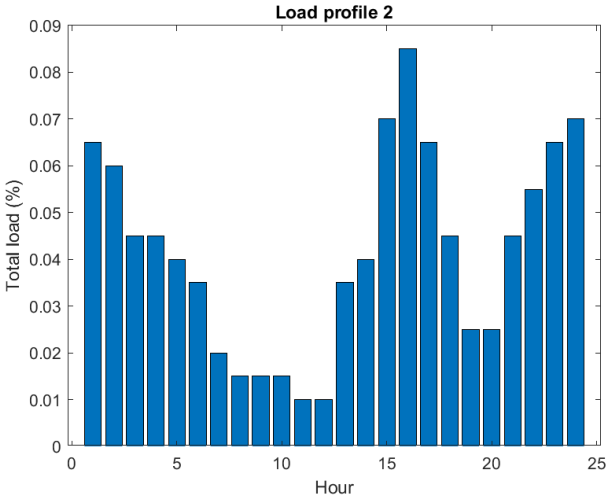

Figure 24: load profile 2.

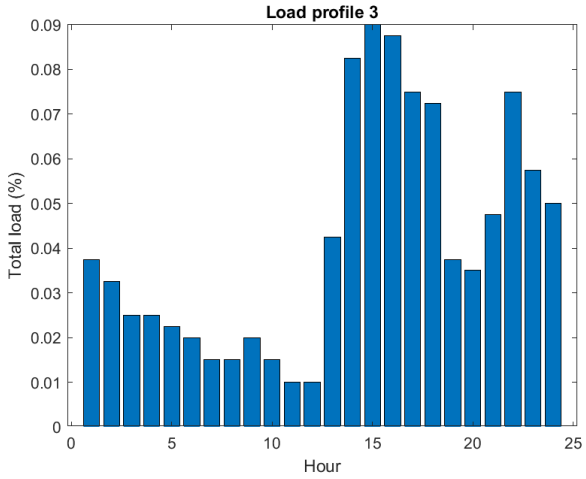

Figure 25: load profile 3.

The cost of withdrawing power from the utility grid c_t(€/kWh) are gathered from [66] and depicted in Figure 26. The first graphic shows the hourly cost for the hourly distributed tariff, where two different cost regions can be clearly spotted. On the other hand, the second picture brings a classical constant tariff where the kWh prize ranges from approximately 0.12 to 0.15€, whereas in the HDT one it goes from approximately 0.08 to 0.017€, which is translated into more flexibility while solving the distributed problem.

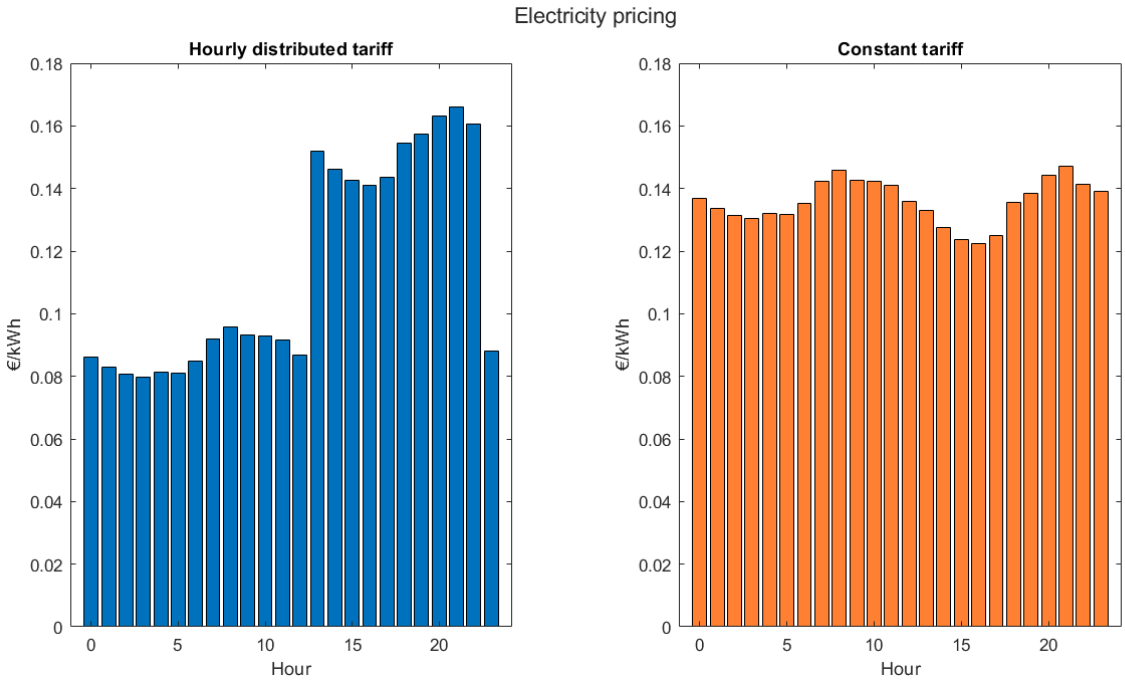

Figure 26: electricity pricing.

A daily energy consumption of 7 kWh and a charging efficiency of 89% are set for EVs, and every household freely chooses the charging hours where their EVs may be charged. For the photo-voltaic generation, the profile used within this study is shown in Figure 27. Batteries in this study are able to store $1250kWh$ and their charge and discharge efficiencies are set at 94.5%.

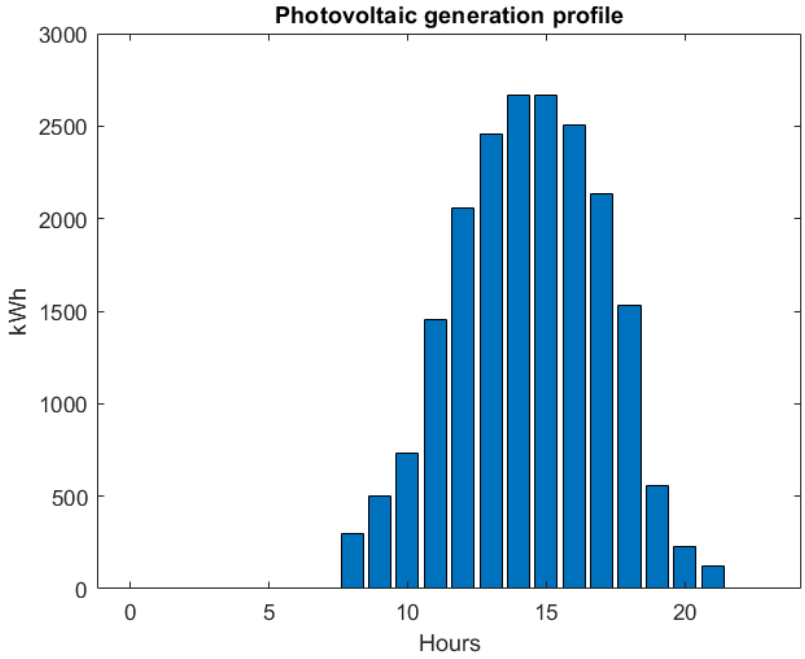

Figure 27: photovoltaic generation profile.

5.2 Test 1: distributed algorithm using constant tariff

In this test the microgrid showed in Figure 22 is simulated with the proposed distributed algorithm from 3.2.3 using the CT from Figure 26.

From step 1 it is obtained the P_i^g from each household for the next day by solving locally the isolated optimization problem (26), (27). Results from all households are sent to the smart contract. Results for agent 1 (consumer) and agent 8 (prosumer) are illustrated in Figure 28 and Figure 29. It is observed that agent 8's battery is delivering power in those timesteps where electricity pricing is higher than usual and vice versa, charging it while it is cheaper.

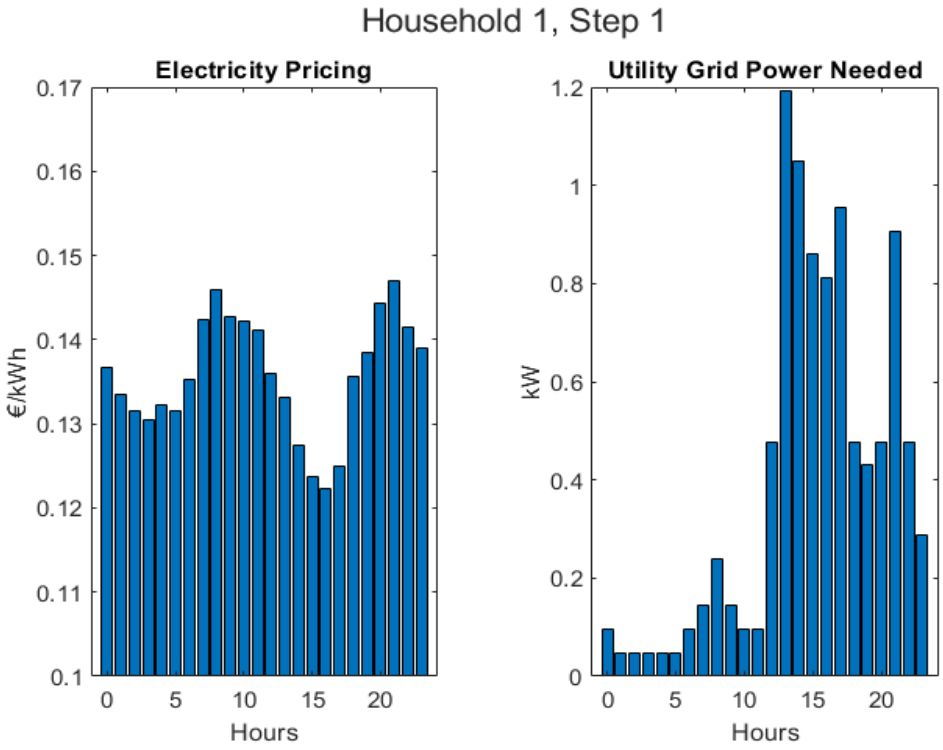

Figure 28: step 1, household 1, test 1.

P^d is built within the smart contract with the 'Utility Power Needed' graphics data of every agent and depicted in Figure 30. Note that households with photovoltaic generation and storing facilities are able to mitigate their energy consumptions in various hours, especially in those where the photovoltaic generation is higher.

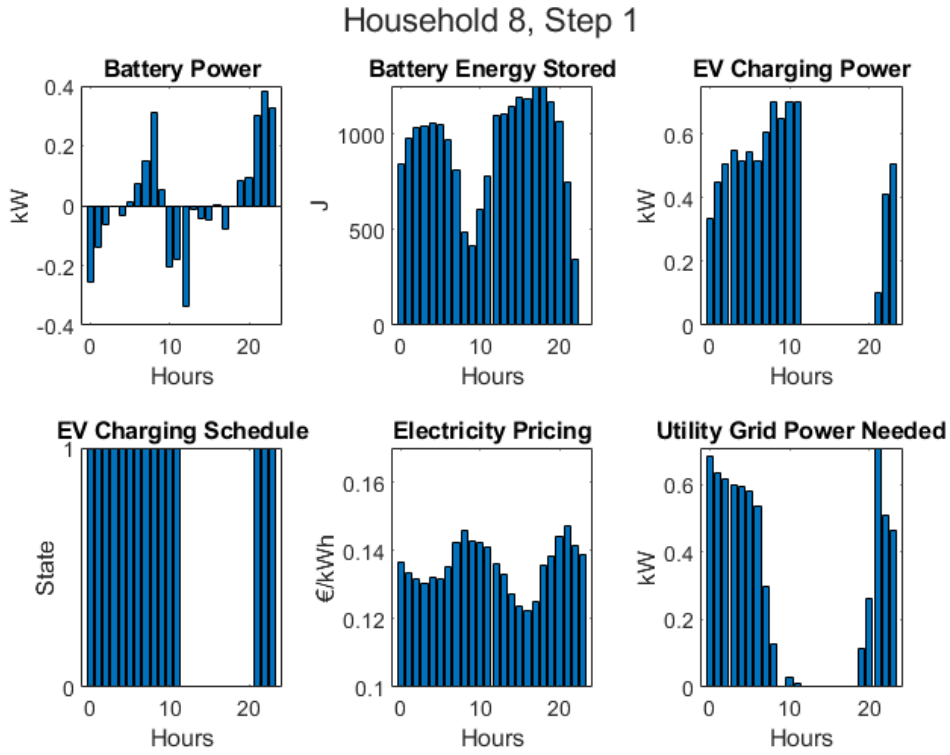

Figure 29: step 1, household 8, test 1.

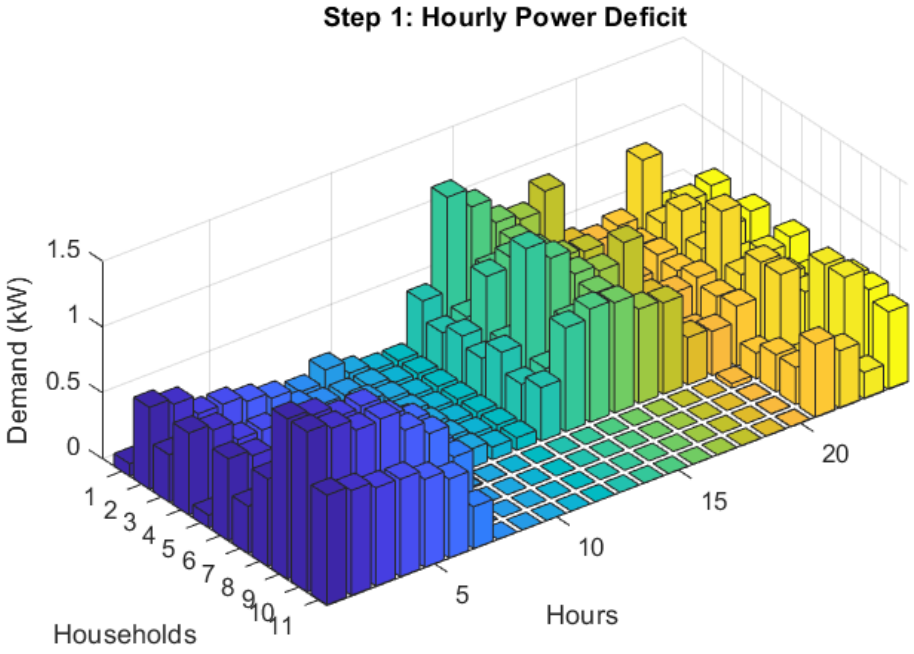

Figure 30: global demand, step 1.

Optimization problem from step 3 is solved locally taking into consideration the P^d matrix which is retrieved from blockchain. Power trades $P_i^t \ \forall \ i,t$ are submitted to blockchain. P_8^t is depicted in Figure 31.

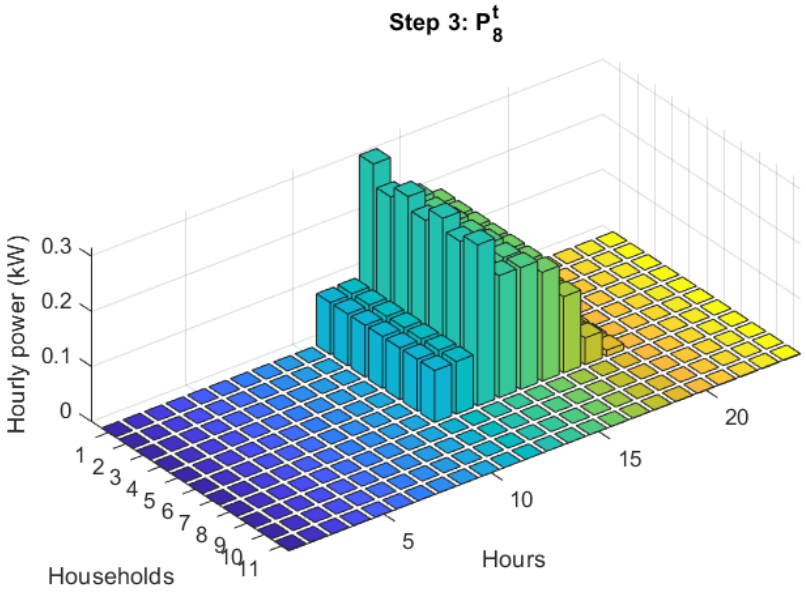

Figure 31: household 8 trades before consensus, step 1.

Φ is built within the smart contract and the hourly power sum expected to be sent $\forall i,t$ before consensus is calculated as $\sum_j^n \Phi_{j,t,i}$, and depicted in Figure 32. Then, the consensus method is executed to calculate all $p_{ij}^{t_c}$ and build Φ^c. The energy received after consensus $\sum_j^n \Phi_{j,t,i}^c$ is depicted in Figure 33.

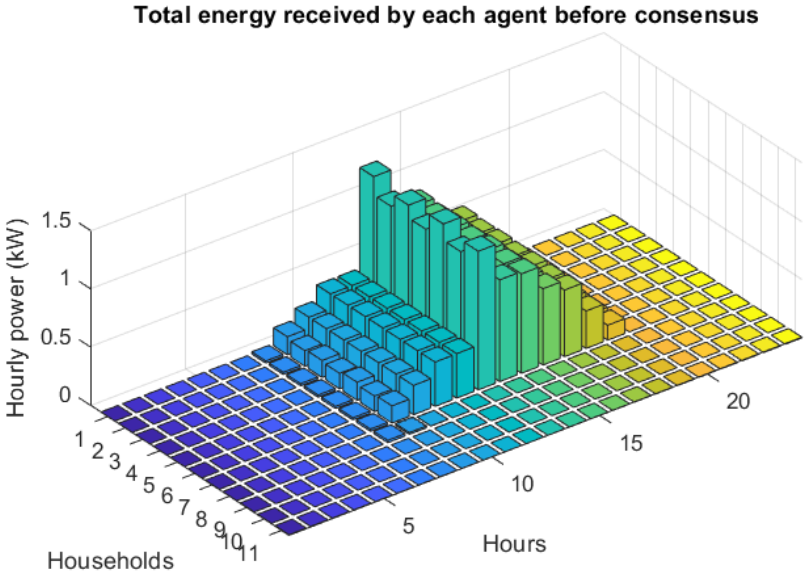

Figure 32: $\sum_j^n \Phi_{j,t,i}$, first iteration, step 1.

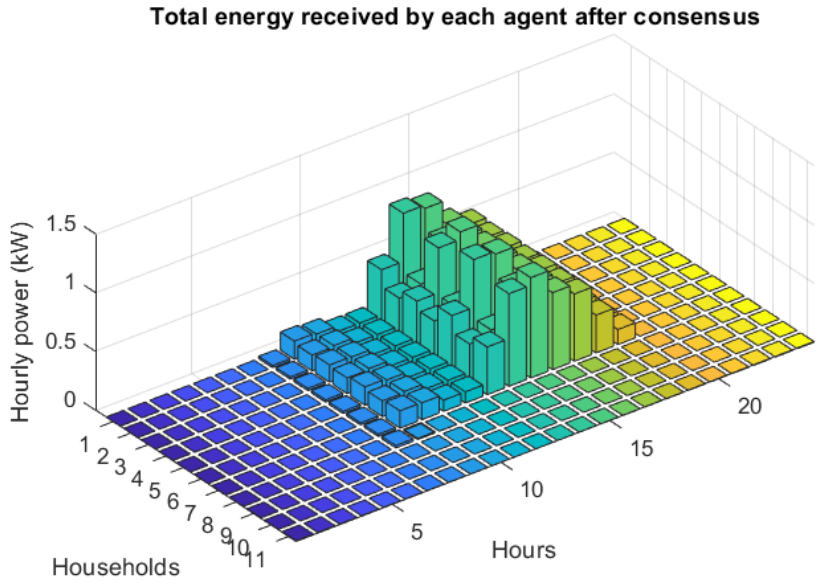

Figure 33: $\sum_j^n \Phi_{j,t,i}^c$, first iteration, step 1.

Since $\Phi \neq \Phi^c$, second iteration is started including Φ^c within the isolated optimization problem. Due to the accepted energy trades, new P_i^g demands are obtained and submitted to the smart contract. Again, P^d is built with every P_i^g and shown in Figure 34.

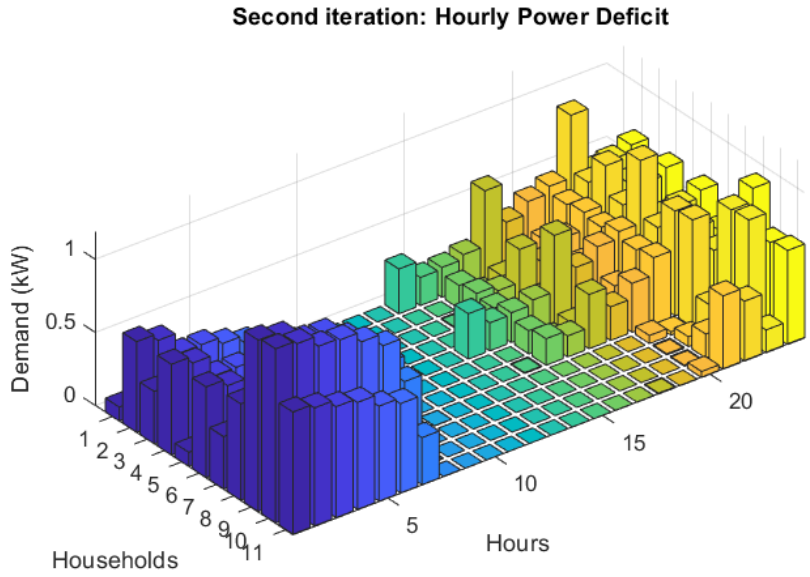

Figure 34: P^d, step 2.

Step 3 optimization problem is solved and new P_i^t are calculated and submitted to the blockchain. Second iteration is concluded once Φ and Φ^c are recalculated and depicted in Figure 35 and Figure 36.

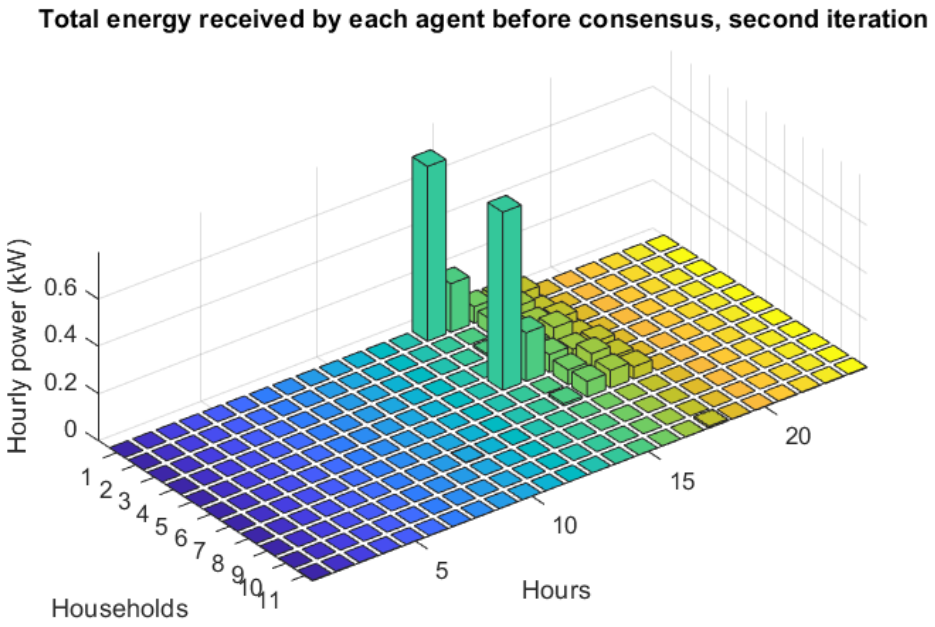

Figure 35: $\sum_j^n \Phi_{j,t,i}$, second iteration.

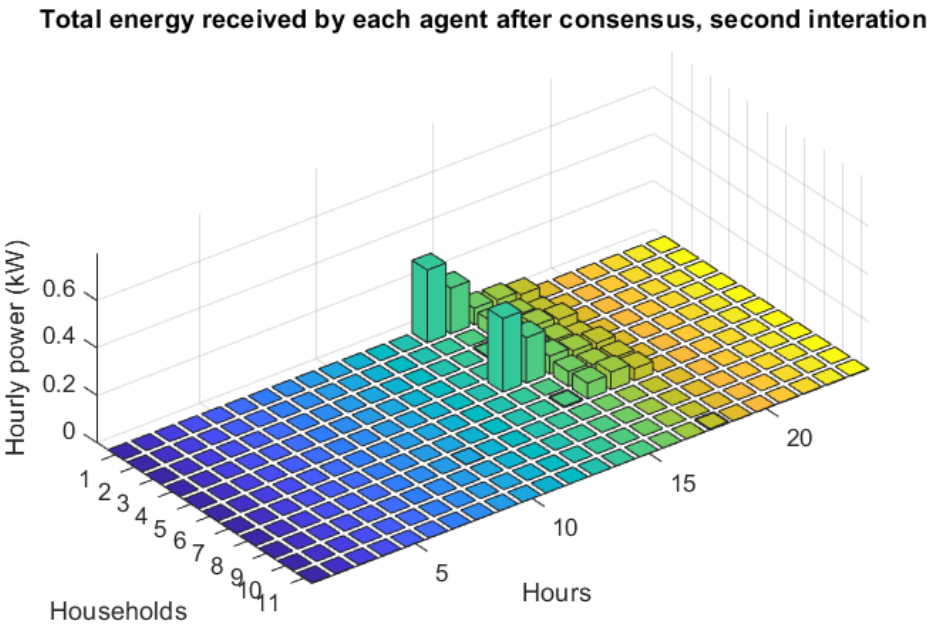

Figure 36:$\sum_j^n \Phi_{j,t,i}^c$, second iteration.

The algorithm finished after 4 iterations. The final accepted trades for the 8^{th} and 14^{th} timesteps are depicted in Figure 37 and Figure 38, where every blue dot and purple triangle represent consumers and prosumers households, respectively. The monetary cost evolution across all iterations is gathered and shown in Figure 39.

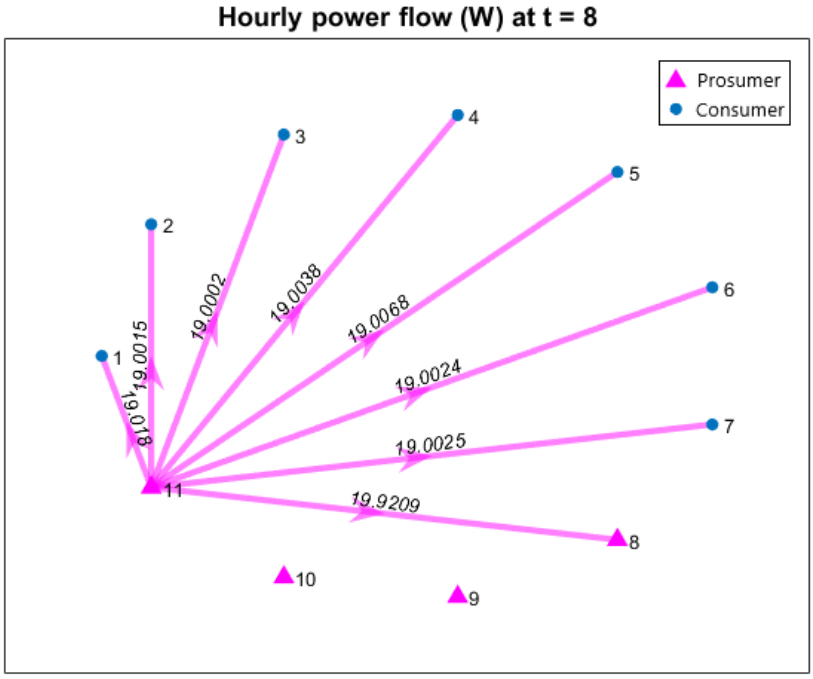

Figure 37: hourly power trades at t = 8.

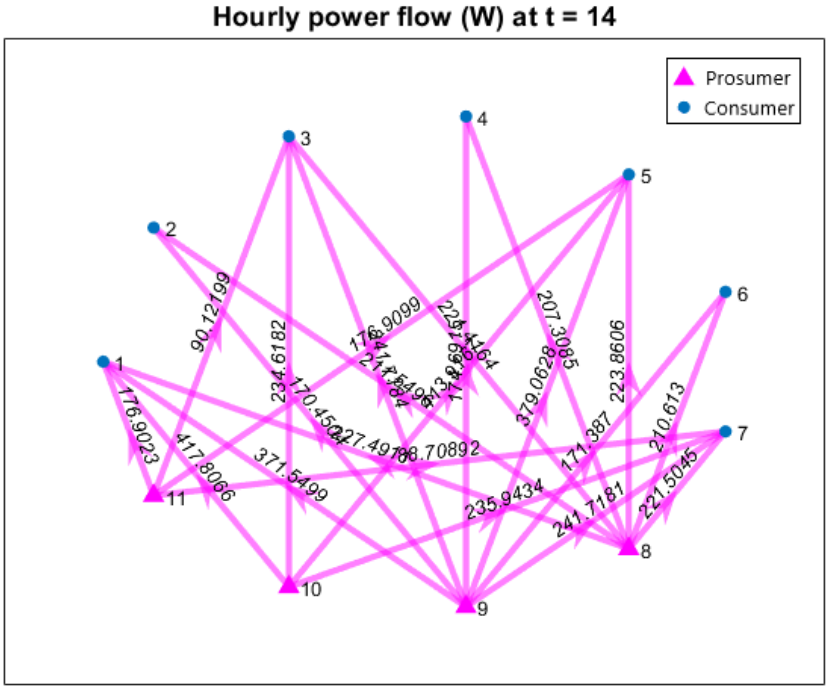

Figure 38: hourly power trades at t = 14.

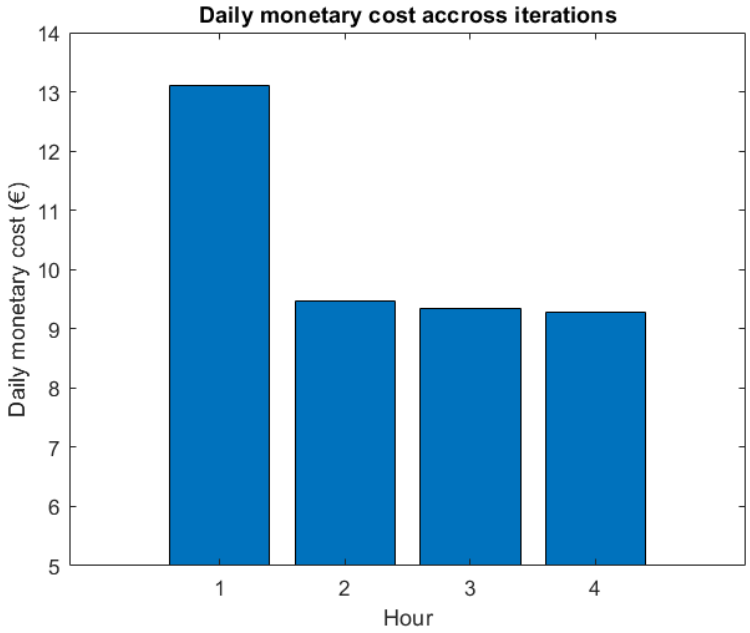

Figure 39: monetary cost across all iterations, test 1.

5.3 Test 2: distributed algorithm using hourly distributed tariff

In this test the electricity pricing tariff used is the HDT from Figure 26 instead of the CT. This tariff enables prosumers to adapt their flexible loads and batteries to minimize not only their cost function but also the microgrid's, since they can prevent consumers from purchasing power in expensive timesteps.

$P_i^g \ \forall i$ for the next day is obtained from step 1. P_1^{grid} (consumer) and P_8^{grid} (prosumer) are illustrated in Figure 40 and Figure 41.

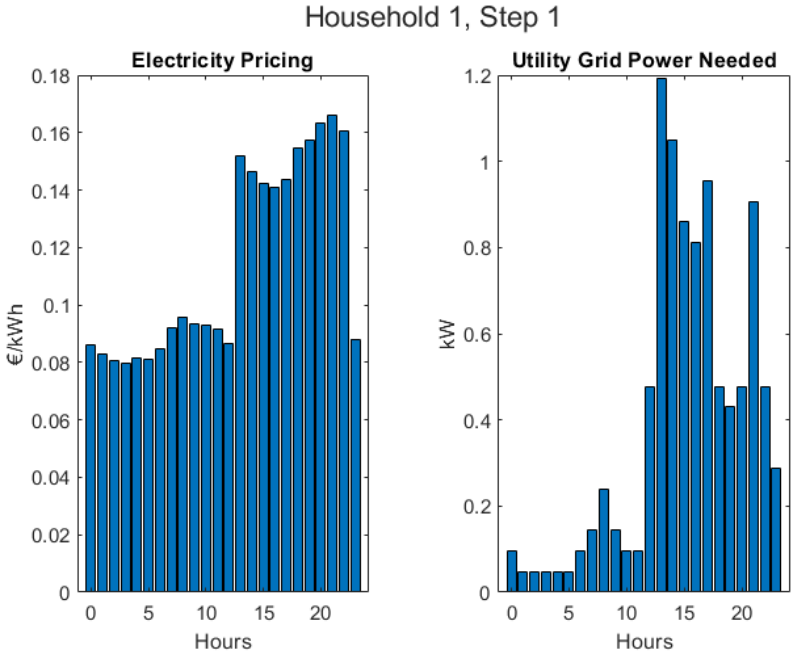

Figure 40: step1, household 1, test 2.

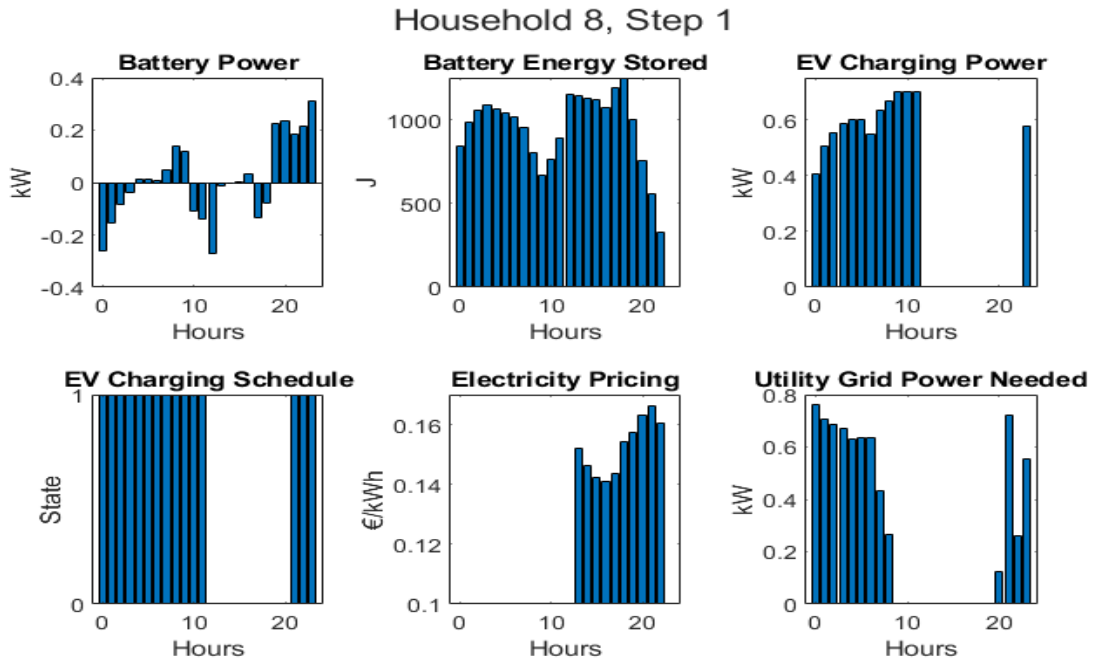

Figure 41: step1, household 8, test 2.

P^d retrieved from the smart contract is depicted in Figure 42. Again, prosumers manage to mitigate their power demands in various hours, especially in those where the photovoltaic generation is higher.

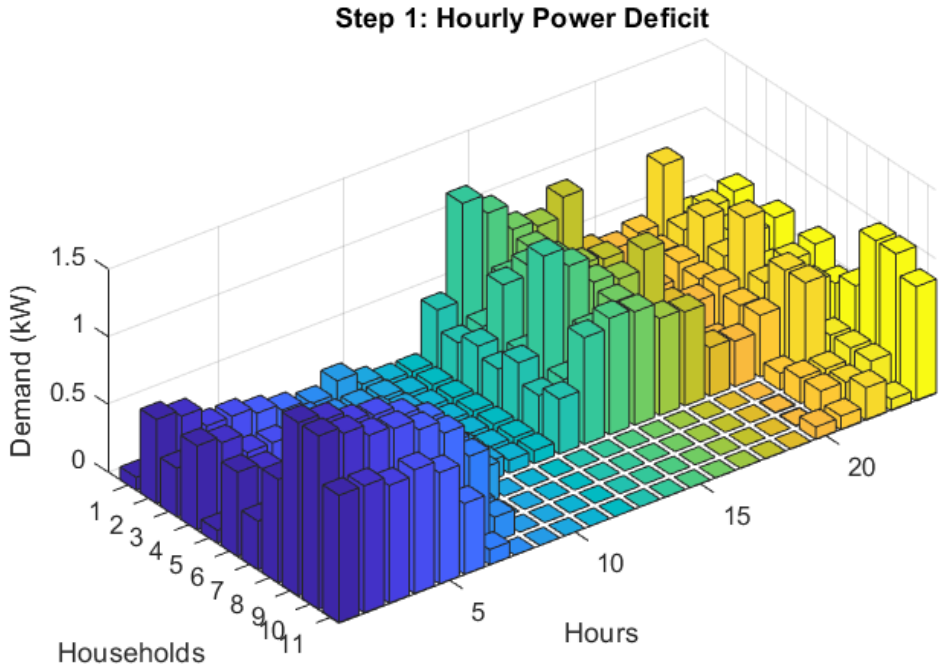

Figure 42: P^d, iteration 1, test 2.

Optimization problem from step 3 is solved locally taking into consideration the P^d matrix which is retrieved from blockchain. Power trades $P_i^t \ \forall \ i,t$ are submitted to blockchain. P_8^t is depicted in Figure 43.

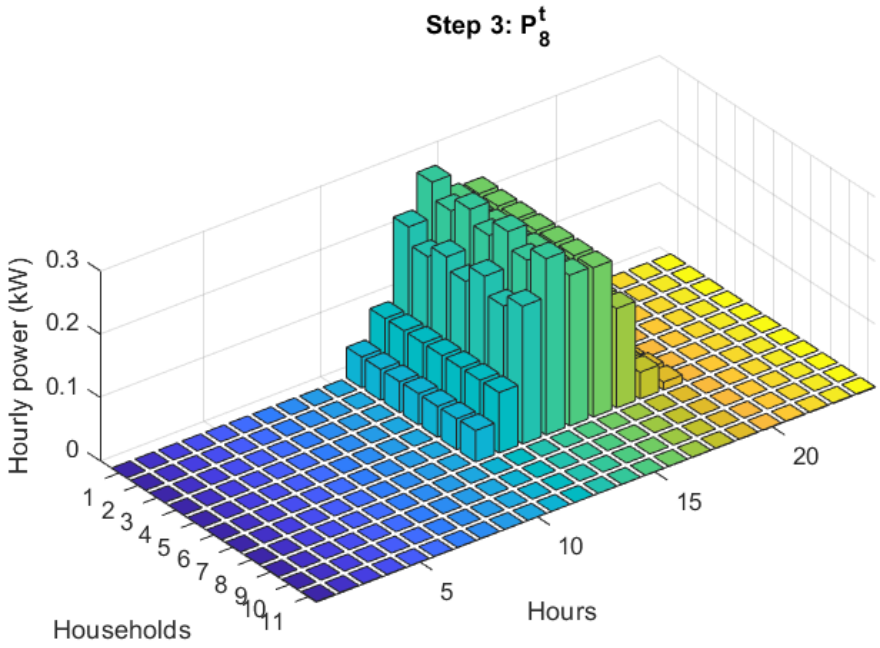

Figure 43: household 8 trades before consensus, step1, test 2.

Φ is built within the smart contract, and the consensus algorithm is executed to calculate all $p_{ij}^{t_c}$ and build Φ^c. The energy received before $\sum_j^n \Phi_{j,t,i}$, and after consensus $\sum_j^n \Phi_{j,t,i}^c$ are depicted in Figure 44, and Figure 45, respectively.

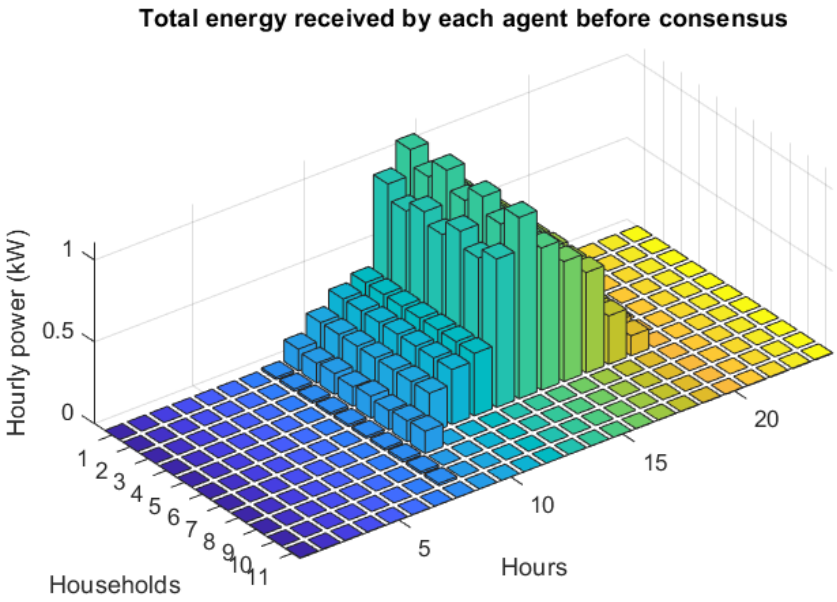

Figure 44: $\sum_j^n \Phi_{j,t,i}$, first iteration, test 2.

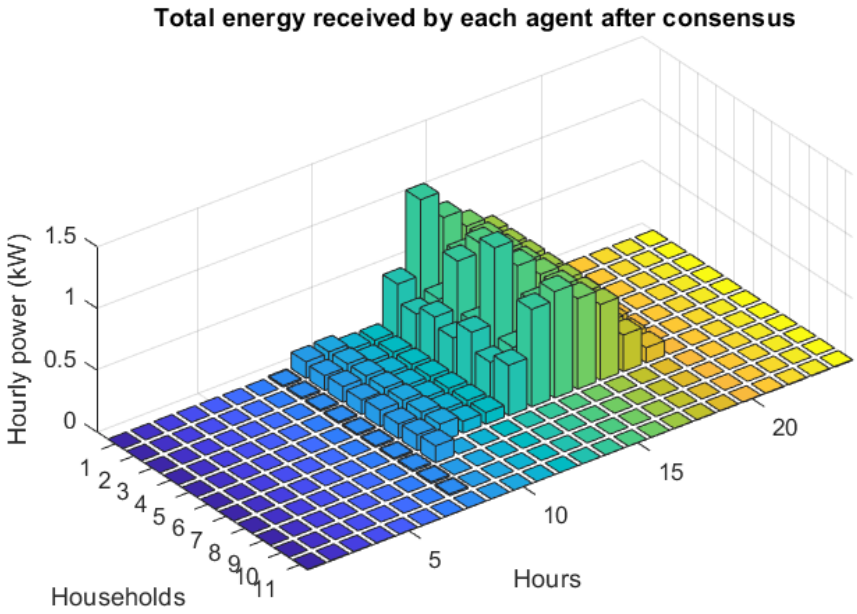

Figure 45: $\sum_j^n \Phi_{j,t,i}^c$, first iteration, test 2.

Since $\Phi \neq \Phi^c$, second iteration is started including Φ^c within the isolated optimization problem. Due to the accepted energy trades, new P_i^g demands are obtained and submitted to the smart contract. Again, P^d is built with every P_i^g and shown in Figure 46.

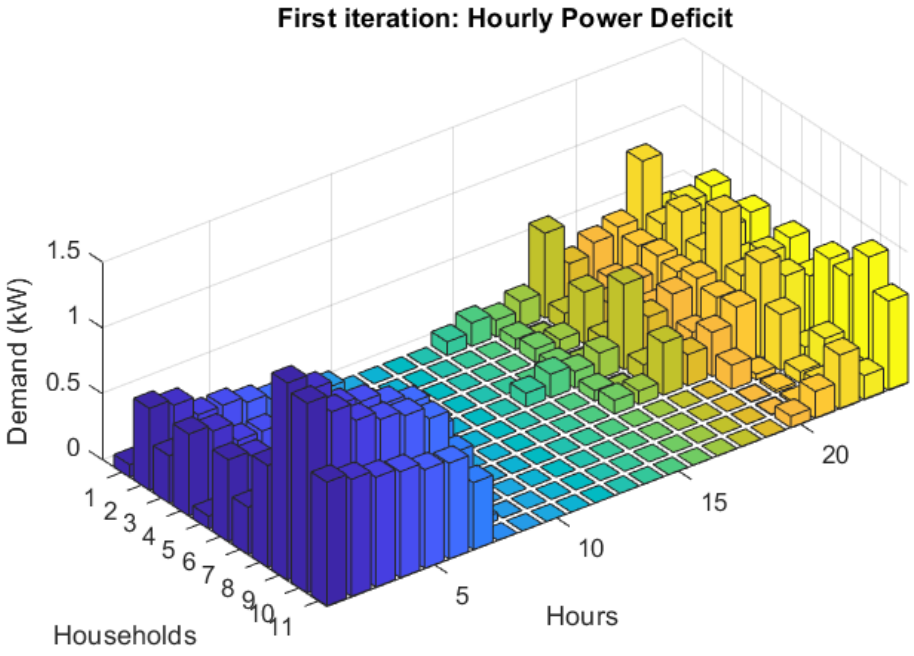

Figure 46: P^d, step 2, test 2.

Step 3 optimization problem is solved and new P_i^t are calculated and submitted to the blockchain. Second

iteration is concluded once Φ and Φ^c are recalculated and depicted in Figure 47, and Figure 48, respectively.

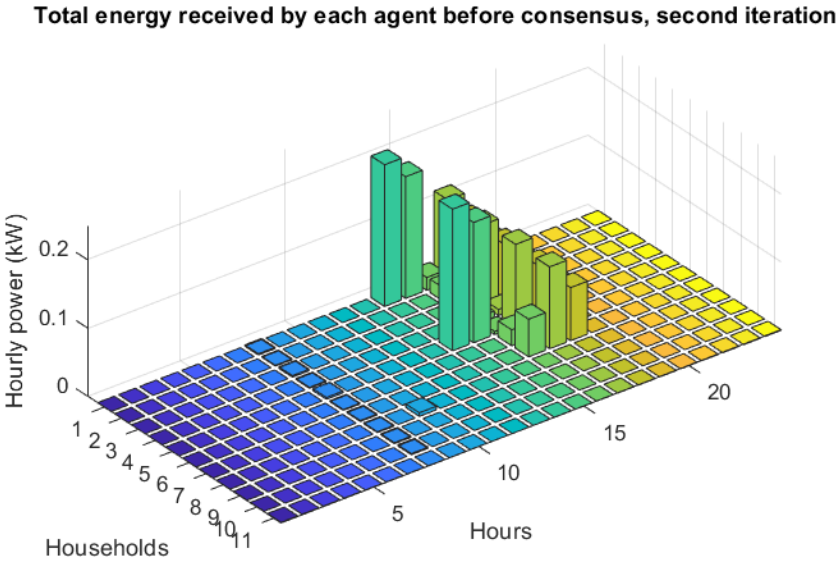

Figure 47: Φ, step 2, test 2.

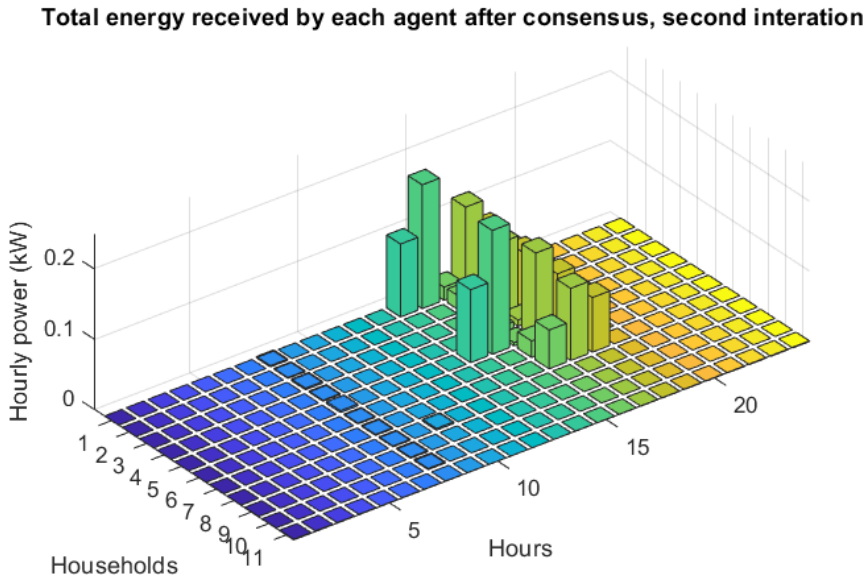

Figure 48: Φ^c , step 2, test 2.

The algorithm finished after 4 iterations. The final accepted trades for the 7^{th} and 14^{th} timesteps are depicted in Figure 37 and Figure 38, where every blue dot and purple triangle represent consumers and prosumers households, respectively. The monetary cost evolution across all iterations is gathered and shown in Figure 51: monetary cost across all iterations, test 2.Figure 51.

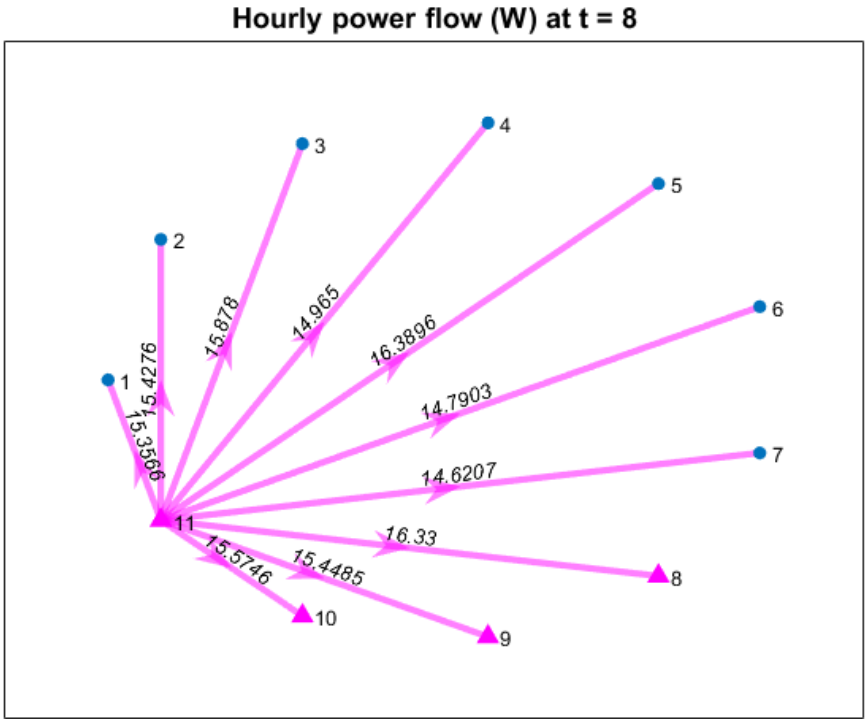

Figure 49: hourly power trades at t = 8, test 2.

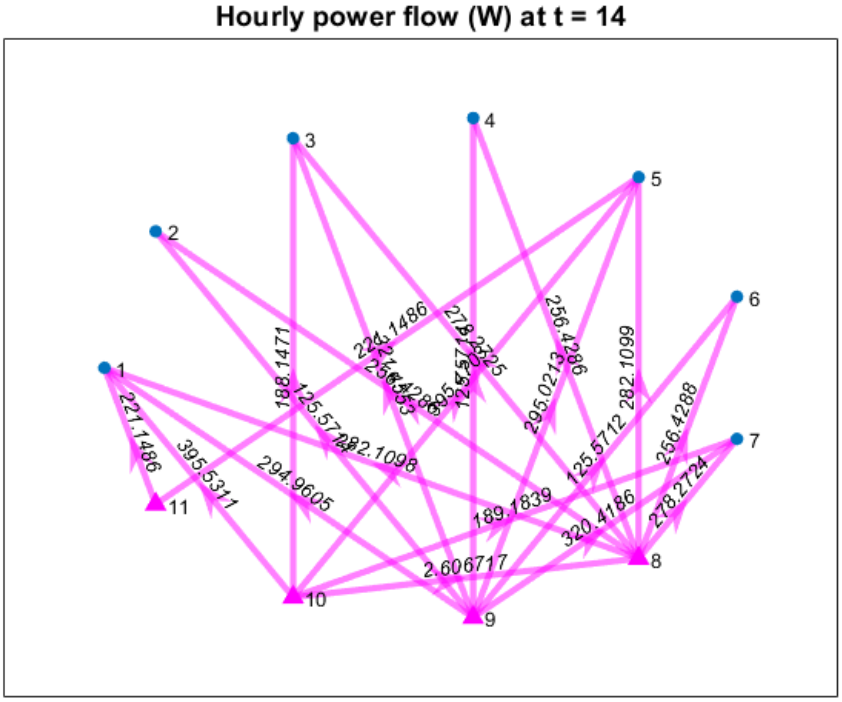

Figure 50: hourly power trades at t = 14, test 2.

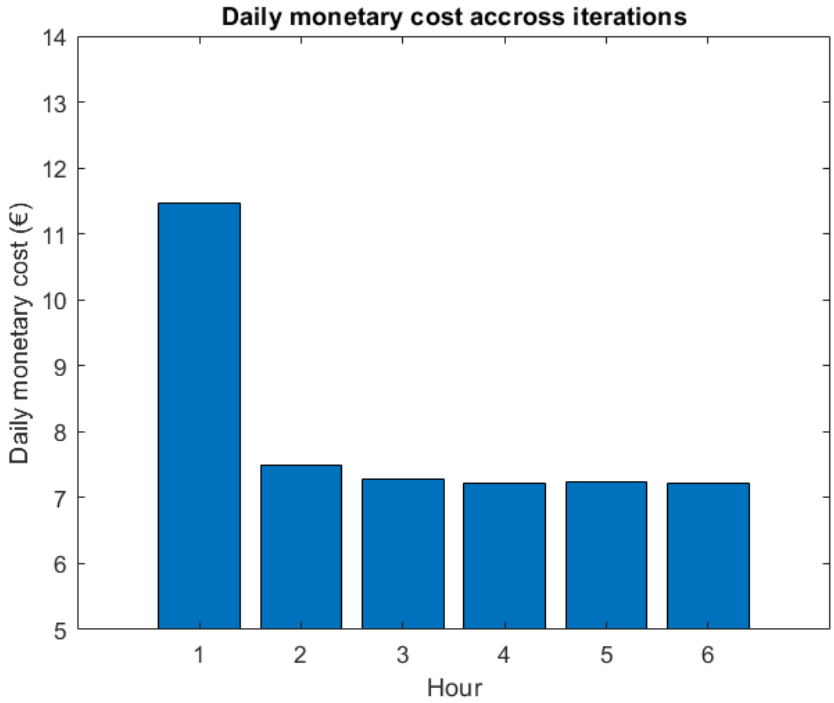

Figure 51: monetary cost across all iterations, test 2.

The algorithm finished after 6 iterations. The final accepted trades for the 8^{th} and 14^{th} timesteps are depicted in Figure 49 and Figure 50, respectively. The monetary cost across evolution for all iterations is shown in Figure 51. As expected, the usage of HDT leads to lower monetary costs since prosumers can adapt their flexible loads and battery power to take full advantage of those timesteps when the electricity price is lower.

5.4 Comparison

In this subsection it is highlighted the positive effect of using the proposed distributed P2P blockchain enabled sharing platform comparing the final objective function, where P2P energy trading is enabled, with its initial state, where households solve an isolated optimization problem. In Test 1, where a CT tariff is used, the daily monetary cost for the microgrid from solving the isolated optimization problem (14) is 13,10€, and, after the algorithm reaches finalization, is reduced 29,08% to 9,29€, which corresponds to 1390€ reduced per year. In Test 2, where a HDT tariff replaces the CT used in Test 1, the initial daily monetary cost for the microgrid is 11,45€, and, after 6 iterations, it is reduced 36,94% to 7,22€, which becomes into 1544€ annualy. As expected, the usage of HDT leads to lower monetary costs since prosumers can adapt their flexible loads and battery power to take full advantage of those timesteps when the electricity price is lower.

6 CONCLUSIONS

This work has developed a distributed energy management platform within a microgrid that serves as a day-ahead energy scheduling program that minimizes the utility bill of a neighborhood by enabling power trades among households. These features are built in a public blockchain network to avoid relying on third parties, have full traceability of shared data, and enable safe P2P interactions, among other benefits.

Thanks to the distributed algorithm proposed, the utility grid bill is reduced, and consensus around power trades is always reached. In particular, when an HDT is used, DERs can fully unleash their potential since not only they are used to reduce its user bill but also its neighbors'. Furthermore, the usage of blockchain erases the possibility of any agent manipulating the algorithm for their own benefit since the smart contract, and every transaction are immutable, proving blockchain a reliable tool to enable decentralization in a transparent and safe manner.

Regarding future research, we plan to add an intra-daily demand response to address possible imbalance between the day-ahead prediction and the live power demand. Also, it could be of interest using a private/hybrid blockchain network to study performance and viability.

REFERENCES

[1] Herzog, A. V, Lipman, T. E, Kammen and D. M, "Renewable energy sources," *Encyclopedia of life support systems (EOLSS),* 2001.

[2] Panwar, NL, S. Kaushik and S. Kothari, "Role of renewable energy sources in environmental protection: A review.," *Renewable and sustainable energy reviews,* vol. 15, no. 3, pp. 1513-1524, 2011.

[3] Banshwar, Anuj, N. K. Sharma, Y. R. Sood and R. Shrivastava, "Real time procurement of energy and operating reserve from Renewable Energy Sources in deregulated environment considering imbalance penalties," *Renewable Energy,* vol. 113, pp. 855--866, 2017.

[4] W.-Y. Chiu, H. Sun and V. H. Poor, "Energy imbalance management using a robust pricing scheme," *IEEE Transactions on Smart Grid,* vol. 4, no. 2, pp. 896--904, 2012.

[5] H. Jiayi, J. Chuanwen and X. Rong, "A review on distributed energy resources and MicroGrid," *Renewable and Sustainable Energy Reviews,* vol. 12, no. 9, pp. 2472--2483, 2008.

[6] M. F. Akorede, H. Hizam and E. Pouresmaeil, "Distributed energy resources and benefits to the environment," *Renewable and sustainable energy reviews,* vol. 14, no. 2, pp. 724--734, 2010.

[7] C. Eid, P. Codani, Y. Perez, J. Reneses and R. Hakvoort, "Managing electric flexibility from Distributed Energy Resources: A review of incentives for market design," *Renewable and Sustainable Energy Reviews,* vol. 64, pp. 237--247, 2016.

[8] J. Guerrero, A. C. Chapman and G. Verbič, "Decentralized P2P energy trading under network constraints in a low-voltage network," *IEEE Transactions on Smart Grid,* vol. 10, no. 5, pp. 5163--5173, 2018.

[9] C. Zhang, J. Wu, Y. Zhou, M. Cheng and C. Long, "Peer-to-Peer energy trading in a Microgrid," *Applied Energy,* vol. 220, pp. 1-12, 2018.

[10] C. Long, J. Wu, C. Zhang, L. Thomas, M. Cheng and N. Jenkins, "Peer-to-peer energy trading in a community microgrid," in *2017 IEEE power & energy society general meeting*, IEEE, 2017, pp. 1-5.

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

[12] M. Nofer, P. Gomber, O. Hinz and D. Schiereck, "Blockchain," *Business & Information Systems Engineering,* vol. 59, no. 3, pp. 183-187, 2017.

[13] M. Wohrer and U. Zdun, "Smart contracts: security patterns in the ethereum ecosystem and solidity," in *2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE)*, IEEE, 2018, pp.

2-8.

[14] I. A. Umoren, S. S. A. Jaffary, M. Z. Shakir, K. Katzis and H. Ahmadi, "Blockchain-Based Energy Trading in Electric-Vehicle-Enabled Microgrids," *IEEE Consumer Electronics Magazine,* vol. 9, no. 6, pp. 66-71, 2020.

[15] H. A. Khattak, K. Tehreem, A. Almogren, Z. Ameer, I. U. Din and M. Adnan, "Dynamic pricing in industrial internet of things: Blockchain application for energy management in smart cities," *Journal of Information Security and Applications,* vol. 55, p. 102615, 2020.

[16] Z. Li, J. Kang, R. Yu, D. Ye, Q. Deng and Y. Zhang, "Consortium blockchain for secure energy trading in industrial internet of things," *IEEE transactions on industrial informatics,* vol. 14, no. 8, pp. 3690-3700, 2017.

[17] Icaew, "History of blockchain," [Online]. Available: https://www.icaew.com/technical/technology/blockchain/blockchain-articles/what-is-blockchain/history.

[18] Wikipedia, "Nick Szabo," [Online]. Available: https://en.wikipedia.org/wiki/Nick_Szabo.

[19] N. Soni, "Evolution of Blockchain," Medium, [Online]. Available: https://medium.com/@nehasoni1812/evolution-of-blockchain-f243f7509fe6#:~:text=Stefan%20Konst%20published%20his%20theory,linked%20together%20using%20cryptographic%20methods..

[20] Thought Leadership Zen, "Blockchain industry overview," [Online]. Available: https://thoughtleadershipzen.blogspot.com/2019/10/blockchain-industry-overview.html.

[21] HocusPocus00. [Online]. Available: https://commons.wikimedia.org/w/index.php?curid=99450981.

[22] M. Andoni, V. Robu, D. Flynn, S. Abram, D. Geach, D. Jenkins, P. McCallum and A. Peacock, "lockchain technology in the energy sector: A systematic review of challenges and opportunities," *Elsevier.*

[23] IBM, "What is blockchain technology?".

[24] M. Juri, "The Blockchain Phenomenon – The Disruptive Potential of Distributed Consensus Architectures," [Online]. Available: https://www.researchgate.net/publication/313477689_The_Blockchain_Phenomenon_-_The_Disruptive_Potential_of_Distributed_Consensus_Architectures.

[25] Wikipedia, "Public-key cryptography," [Online]. Available: https://en.wikipedia.org/wiki/Public-key_cryptography.

[26] IBM, "What are smart contracts on blockchain?," [Online]. Available: https://www.ibm.com/blogs/blockchain/2018/07/what-are-smart-contracts-on-blockchain/. [Accessed 9 febrero 2021].

[27] B. Council, "TYPES OF BLOCKCHAINS EXPLAINED- PUBLIC VS. PRIVATE VS. CONSORTIUM," TOSHENDRA KUMAR SHARMA, [Online]. Available: https://www.blockchain-council.org/blockchain/types-of-blockchains-explained-public-vs-private-vs-consortium/. [Accessed 12 02 2021].

[28] Binance, “Private, Public, and Consortium Blockchains - What's the Difference?,” [Online]. Available: https://academy.binance.com/en/articles/private-public-and-consortium-blockchains-whats-the-difference. [Accessed 12 02 2021].

[29] Wikipedia, “Cryptographic hash function,” [Online]. Available: https://en.wikipedia.org/wiki/Cryptographic_hash_function. [Accessed 17 2 2021].

[30] Wikipedia, “Cryptographic hash function,” [Online]. Available: https://en.wikipedia.org/wiki/Cryptographic_hash_function. [Accessed 17 2 2021].

[31] D. Yaga, P. Mell, N. Roby and K. Scarfone, “Blockchain Technology Overview,” National Institute of Standars and Technology, [Online]. Available: https://nvlpubs.nist.gov/nistpubs/ir/2018/NIST.IR.8202.pdf. [Accessed 16 2 2021].

[32] Fandom, “Cryptographic nonce,” [Online]. Available: https://cryptography.fandom.com/wiki/Cryptographic_nonce.

[33] Davidgothberg, “Public-key cryptography figure,” [Online]. Available: https://commons.wikimedia.org/w/index.php?curid=1028460.

[34] I. Bashir, Mastering Blockchain.

[35] Binance, “Block header,” [Online]. Available: https://academy.binance.com/en/glossary/block-header. [Accessed 19 05 2021].

[36] W. Kenton, “Block Header (Cryptocurrency),” [Online]. Available: https://www.investopedia.com/terms/b/block-header-cryptocurrency.asp.

[37] Binance, “What Is a Blockchain Consensus Algorithm?,” [Online]. Available: https://academy.binance.com/en/articles/what-is-a-blockchain-consensus-algorithm. [Accessed 12 2 2021].

[38] D. Mingxiao, M. Xiaofeng, Z. Zhe, W. Xiangwei and C. Qijun, “A Review on Consensus Algorithm of Blockchain,” in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Banff Center, Banff, Canada, 2017.

[39] Binance, “Byzantine Fault Tolerance Explained,” [Online]. Available: https://academy.binance.com/en/articles/byzantine-fault-tolerance-explained. [Accessed 12 2 2021].

[40] Wikipedia, “Hashcash,” [Online]. Available: https://en.wikipedia.org/wiki/Hashcash.

[41] Bitcoin Wiki, “Proof of work,” [Online]. Available: https://en.bitcoin.it/wiki/Proof_of_work. [Accessed 15 2 2021].

[42] Investopedia, “Proof of Elapsed Time (PoET) (Cryptocurrency),” Jake Frankenfield, [Online]. Available: https://www.investopedia.com/terms/p/proof-elapsed-time-cryptocurrency.asp#:~:text=Proof%20of%20elapsed%20time%20(PoET)%20is%20a%20consensus%20algorithm%20developed,block%20winners%20and%20mining%20rights.&text=The%20PoET%20algorithm%20generates%20a,to%20. [Accessed 16 2 2021].

[43] Investopedia, “Proof of Capacity,” Adam Hayes, [Online]. Available: https://www.investopedia.com/terms/p/proof-capacity-

cryptocurrency.asp#:~:text=Proof%20of%20capacity%20allows%20the,to%20mine%20the%20available%20cryptocurrencies.&text=Burstcoin%20is%20a%20cryptocurrency%20that,Storj%2C%20Chia%2C%20and%20SpaceMint.. [Accessed 16 2 2021].

[44] M. Castro and B. Liskov, "Practical Byzantine Fault Tolerance," in *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, New Orleans, USA, 1999.

[45] H. Sukhwani, J. M. Martínez, X. Chang, K. S. Trivedi and A. Rindos, "Performance modeling of PBFT consensus process for permissioned blockchain network (hyperledger fabric)," in *2017 IEEE 36th Symposium on Reliable Distributed Systems*, 2017.

[46] Golden, "Federated Byzantine Agreement (FBA)," [Online]. Available: https://golden.com/wiki/Federated_Byzantine_Agreement_(FBA). [Accessed 17 2 2021].

[47] CoinMarketCap, [Online]. Available: https://coinmarketcap.com/currencies/bitcoin/. [Accessed 19 05 2021].

[48] S. Muftic, "Overview and Analysis of the Concept and Applications of Virtual Currencies".

[49] Ethereum, "Ethereum Whitepaper," 9 2 2021. [Online]. Available: https://ethereum.org/en/whitepaper/. [Accessed 8 3 2021].

[50] Hyperledger, [Online]. Available: https://www.hyperledger.org/use. [Accessed 08 03 2021].

[51] C. Bordons, F. Garcia-Torres and M. Ridao, Model Predictive Control of Microgrids, Springer, 2020.

[52] C. Burger, J. Weinmann, A. Kuhlmann and P. Richard, "Blockchain in the energy transition. A survey among decision-makers in the German energy industry.," *DENA German Energy Agency,* 2016.

[53] M. Swan, Blockchain: Blueprint for a new economy, O'Reilly Media, 2015.

[54] Wikipedia, "Solidity," [Online]. Available: https://en.wikipedia.org/wiki/Solidity. [Accessed 16 3 2021].

[55] Node.js, "About Node.js®," [Online]. Available: https://nodejs.org/en/about/. [Accessed 16 3 2021].

[56] Wikipedia, "Node.js," [Online]. Available: https://en.wikipedia.org/wiki/Node.js. [Accessed 16 3 2021].

[57] web3.js, "web3.js - Ethereum JavaScript API," [Online]. Available: https://web3js.readthedocs.io/en/v1.3.4/. [Accessed 3 16 2021].

[58] S. Beyer, "What is Web3.js? A Detailed Guide," [Online]. Available: https://www.mycryptopedia.com/what-is-web3-js-a-detailed-guide/. [Accessed 16 3 2021].

[59] Nethereum, "Ganache CLI Configuration and usage," [Online]. Available: https://docs.nethereum.com/en/latest/ethereum-and-clients/ganache-cli/. [Accessed 16 3 2021].

[60] Truffle Suite, "TRUFFLE OVERVIEW," [Online]. Available: https://www.trufflesuite.com/docs/truffle/overview. [Accessed 16 3 2021].

[61] Metamask, "Metamask," [Online]. Available: https://metamask.io/index.html. [Accessed 16 3 2021].

[62] Stack overflow, “2020 Developer Survey,” [Online]. Available: https://insights.stackoverflow.com/survey/2020#overview. [Accessed 3 17 2021].

[63] Wikipedia, “React (JavaScript library),” [Online]. Available: https://en.wikipedia.org/wiki/React_(JavaScript_library). [Accessed 17 3 2021].

[64] Infura, “Infura Frequently Asked Questions,” [Online]. Available: https://infura.io/faq. [Accessed 17 3 2021].

[65] IDAE, [Online]. Available: https://www.idae.es/uploads/documentos/documentos Documentacion. [Accessed 18 05 2021].

[66] System Operator Information System., “Término de facturación de energía activa del PVPC,” [Online]. Available: https://www.esios.ree.es/es/pvpc. [Accessed 06 05 2021].

[67] FreeCodeCamp, “How Bitcoin mining really works,” Subhan Nadeem, [Online]. Available: https://www.freecodecamp.org/news/how-bitcoin-mining-really-works-38563ec38c87/#:~:text=At%20a%20very%20high%20level,specific%20output%20the%20network%20accepts.. [Accessed 15 2 2021].

[68] Wikipedia, “Avalanche effect,” [Online]. Available: https://en.wikipedia.org/wiki/Avalanche_effect. [Accessed 17 2 2021].

[69] IDAE, “Consumos del Sector Residencial en España,” [Online]. Available: https://tarifasgasluz.com/sites/default/files/pdf/documentacion_basica_residencial_unido.pdf. [Accessed 29 03 2021].

[70] Protokol, “Top 5 Blockchain Use Cases in Energy and Utilities,” [Online]. Available: https://www.protokol.com/insights/top-5-blockchain-use-cases-in-energy-and-utilities/. [Accessed 17 05 2021].

[71] S. Malladi, J. Alves-Foss and R. B. Heckendorn, “On preventing replay attacks on security protocols,” 2002.

[72] Wikipedia, “Árbol de Merkle,” [Online]. Available: https://es.wikipedia.org/wiki/%C3%81rbol_de_Merkle.